UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Resolution in Internet Identifiers: Towards Context-Awareness and Customisation

by Andrew Martin Babakian

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Emeritus Professor Robin Braun and Industry Associate Professor Justin Lipman

University of Technology Sydney
Faculty of Engineering and Information Technology

June 2024

Certificate of Original Authorship

I, Andrew Martin Babakian, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Production Note:
Signature removed prior to publication.

Signature:

Date: 01 June 2024

Abstract

Identity systems are the key to any network. Just as a language is defined by a consistent set of symbols and their usage, a network is defined by a consistent set of identities and their usage. For users to find this network useful, they must be able to communicate references using human-friendly names that are easy to remember and be confident that these names refer to the same digital objects in multiple contexts. This is only achievable if names are allocated from a unified namespace and the resolution of the name into network addresses is consistent and coherent. Market competition dynamics for digital presence naturally encourage us to be bound within this unified namespace in the Domain Name System (DNS).

Originally, the DNS modelled a telephone directory, a static system published periodically with invariant lookup keys. This directory was non-customisable and uniform for all users, permitting anyone to use the same lookup key and retrieve identical information. One could share a key, and others could access the same database to retrieve identical results, making it possible to refer to entities via their keys. However, with the evolution of the Internet, the concept of identity systems also evolved, and the use of identifiers within the DNS began to shift. In this evolution, one can consider the broader concept of DNS as a marketplace and the identity system as a market. It is not the name that determines the market; rather, it is the behaviour of the name – its ability to customise and contain attributes beyond its static translation – that determines the usefulness of the name. This reduces the utility of the objective uniform identifier system. However, we rely on customisation to cater to user preferences and needs, which is a feature of today's Internet.

This research contributes to the literature by examining identifier systems within and beyond the Internet context. Additionally, this study introduces a review of the DNS, revealing a broad range of parameters governing its operation. Based on these foundational principles, this research proposes a computational model to concurrently enhance the system's flexibility, speed and privacy, introducing alternative behaviours in the DNS. The impetus behind this study was the imperative to extend DNS functionality, thereby preserving the unified namespace and highlighting the importance of avoiding fragmentation that would compromise the integrity and erode the value of a cohesive, unified communication network.

Acknowledgements

Firstly, I must thank my supervisor, Robin Braun, whose positive impact on my life cannot be overstated. Robin reinforced my passion for research and instilled insights into principles of creating good research, navigating through obscurity until the findings become apparent, and, importantly, taking feedback and using it as a portal of discovery to improve continuously.

In this journey, Robin identified my strengths, enabling me to harness them for adaptively learning, improving my writing, and developing techniques for this research, ultimately allowing me to deliver my best work. At the same time, he was consistently there to support me through times of difficulty and provided a unique perspective that allowed me to continue moving forward. It is hard to imagine a better person to guide me on this journey, and I am truly privileged to have had Robin as my supervisor.

I would also like to thank my co-supervisor, Justin Lipman, whose guidance, mentorship and encouragement have been invaluable throughout my journey. I am truly grateful for Justin's insightful approaches to advancing my study.

Embarking on research within the intricate field of Internet identifiers, I quickly realised the rich history and the complex interplay of diverse systems it encompasses. I would like to thank Geoff Huston from the Asia-Pacific Network Information Centre (APNIC), my subject matter expert and mentor, for his invaluable contribution to my learning and the development of this research. In the middle of his work and other commitments in the Internet standards realm, Geoff, without hesitation, would always be available to share his knowledge, provide feedback, distil the essence of this research, and explain why it is important for the Internet's future. In addition, I would also like to thank Lixia Zhang from the University of California, Los Angeles (UCLA), who also, in between her work commitments and supervising other research students, unhesitatingly offered support and took me on as another student to help guide me through the literature, sharing her knowledge and providing invaluable feedback and encouragement.

During my studies, I worked full-time and was privileged to integrate my research with an industrial perspective. I would like to thank Pere Monclus from the Office of the CTO at VMware Inc. for developing my thinking in approaching the challenges in systems networking. No matter how busy he was, he would spend time providing guidance and direction on how to evolve this study to highlight the impact from an industry point of view. In addition, I would like to thank Emad Benjamin from the Office of the CTO for his guidance and mentoring, especially in helping me learn more about approaches to solving intricate software engineering challenges of this research. I would also like to thank Saidulu Adlas from Intel Corporation for the shared collaboration and research in 5G core signalling and future application of identifier systems.

Throughout developing and testing the proposed model, I gained new insights into software

development and scientific validation methods. I would like to thank Aarohi Arora for this learning and his fantastic questions and deep inquisitiveness about this field of study. His guidance, expertise in software development, and support were immensely valuable.

Also, thank you to Matthew Sidebotham, who copyedited the draft thesis in accordance with university policy and the *Guidelines for Editing Research Theses*.

Outside the work and research context, friends who are like family is something I am truly blessed to have in my life. In the middle of the challenges that both PhD research and life throw on one's path, I would like to thank Garth Law for his unstinting encouragement when stress was getting the better of me and for consistently showing up to assist in reviewing my papers and experiments day or night; he is truly deserving of an award for this. I also appreciated Garth and his wife Natasya for their hospitality, with Garth making dinner several nights to ensure I focused on thesis writing. Also, I would like to thank Arup Deb for being there for me; he would call me during difficult times and usually find a way to make me laugh from the other side of the world. On many occasions Arup travelled from Singapore to visit and spend time with me, for which I am very thankful, as well as for his encouragement and belief that great things will happen with this research. I would also like to thank Kit Colbert, Richard James, Seepahn Ingilizian and Gerard Chami for being amazing friends and supporting me through this program.

I would like to thank my family, particularly Romen, Marlin, Steven and Clayton Babakhan, for their support, especially through the most challenging periods of this research program while I was in Australia. I owe a special thank you to Steven Babakhan for his extraordinary efforts to ensure I remained focused on the PhD, constantly checking to ensure I was okay and being available for a chat anytime I needed it. His perspectives were invaluable, and I am privileged to have such an amazing family.

Lastly, I thank my parents, Martin and Enaam Babakian, for their incredible support. No matter the challenge or journey I embarked on, they have always been there to support me through it. My dad, an avid learner and a devoted meteorologist, taught me to aim high and give it my best effort. I owe him for being the greatest role model a son could ask for and for instilling in me a devotion to learning and problem-solving, which I have applied to my research.

Likewise, I owe my mother the gift of determination – *"when you start something, you must finish it. No matter how challenging it is, you keep trying until you succeed"*. This perseverance until success has been a significant factor in reaching this point.

I could not have asked for more amazing parents, and I dedicate this PhD to Martin and Enaam Babakian.

Publications and Research Impact

Accepted Publications / Issued Patents

1. **Dynamic Virtual Machine Network Policy for Ingress Optimization**
   A. Babakian, *United States Patent 11,204,791, Dec. 2021.*
   Brief Description: Enhances IP namespace classification through attribute association.
   Relevance to PhD: Originates from IP namespace research, contributing to the development of emerging topic review in Chapter 3.

2. **A Retrospective on Workload Identifiers: From Data Center to Cloud-Native Networks**
   A. Babakian, P. Monclus, R. Braun, J. Lipman, *IEEE Access, vol. 10, pp. 105518-105527, 2022.*
   Brief Description: Reviews network architectures' evolution of identifiers and locators.
   Relevance to PhD: Central to forming the research proposition, highlighting the extended role of the DNS, outlined in Chapters 3 and 4.

3. **Internet Identifiers: Survey of History, Challenges, and Future Perspectives**
   A. Babakian, G. Huston, R. Braun, J. Lipman, *IEEE Access, vol. 12, pp. 51919-51941, 2024.*
   Brief Description: Establishes research directions through a multidisciplinary review.
   Relevance to PhD: Outlined in Chapter 5A, this publication strengthens the research proposition, guiding the DNS model development in Chapter 6B.

4. **Cloud-Native Service Mesh Readiness for 5G and Beyond**
   S. Aldas, A. Babakian, *IEEE Access, vol. 11, pp. 132286-132295, 2023.*
   Brief Description: Shared primary authorship in collaboration with Intel Corporation. Explores 5G in cloud-native environments and Service Mesh enhancements.
   Relevance to PhD: Reinforces the research application in future identifier system architectures, detailed in Chapter 7A.

5. **Context-Aware Domain Name System (DNS) Query Handling**
   A. Babakian, E. Benjamin, P. Monclus, *United States Patent 11,799,899, Oct. 2023.*
   Brief Description: Proposes a context-aware approach to DNS query handling.
   Relevance to PhD: Reinforces the research application in future context-aware identifier systems, detailed in Chapter 7B.

Contents

I Introduction and Development of the Proposition 2

1 Introduction 3
1.1 Background . . . 3
1.2 Research Motivations . . . 4
1.3 Research Objectives . . . 5
1.4 Progressive and Integrative Research Structure . . . 6
1.5 Research Scope . . . 9
1.6 Key Contributions . . . 10

2 From Lessons Learned to Research Inquiry: A Preliminary Step 12
2.1 Practical Invention and Academic Inquiry . . . 12
2.2 Background . . . 12
2.3 Problem Statement . . . 13
2.4 Proposed Model . . . 13
2.5 Solution Design . . . 14
2.6 Challenges Observed in Proposed Solution . . . 15
2.7 Need for Continued Research . . . 16
2.8 Conclusion: Topical Review Required . . . 16
References for Chapter 2 . . . 16

3 Emerging Topical Review: Identifiers and Locators Evolution 18
3.1 Purpose of this Chapter . . . 18
3.2 Introduction . . . 18
3.3 Location/ID Separation Protocol . . . 19
3.3.1 Semantic Overload in the IP Address . . . 19
3.3.2 LISP for Virtualised Data Centres and the Virtual Machine Artefact . . . 20
3.3.3 Identity and the IP Namespace . . . 20
3.4 Network Virtualisation . . . 21
3.4.1 Core vs. Edge Split . . . 21
3.4.2 Network Virtualisation for Cloud . . . 21
3.4.3 Identity and the IP Namespace . . . 22
3.5 Identifier Locator Addressing . . . 22
3.5.1 Container Artefact and the Enterprise Cloud . . . 22
3.5.2 Identifier Locator Addressing (ILA) Namespace . . . 22
3.6 Cloud-Native Service Mesh . . . 23
3.6.1 Microservices and Service Mesh . . . 23
3.6.2 Service Mesh Namespace for Discovery and Routing . . . 24

3.6.3 Service Mesh Identity and the URI Namespace . . . 25
3.7 Comparisons and Future Directions . . . 26
3.7.1 Comparison between Discussed Techniques . . . 26
3.7.2 Comparisons with the State-of-the-art . . . 28
3.7.3 Future Directions and Investigations . . . 29
3.8 Conclusions . . . 30
References for Chapter 3 . . . 30

II Evidence and Validation of the Proposition 35

4 Research Proposition 36
4.1 Background to the Proposition . . . 36
4.1.1 Givens: . . . 37
4.1.2 We propose the following: . . . 37
4.1.3 We will give weight to the proposition as follows: . . . 37
4.1.4 Future work as evidence: . . . 38
4.1.5 We will conclude as follows: . . . 39

5A Systematic Literature Review: Evolution of Internet Identifiers 40
5A.1 Purpose of this Chapter . . . 40
5A.2 Introduction . . . 40
5A.3 Systematic Literature Review Planning . . . 41
5A.3.1 Research Questions . . . 42
5A.3.2 Source Selection . . . 43
5A.3.3 Study Selection and Evaluation Process . . . 44
5A.3.4 Search Screen Results . . . 45
5A.4 Background . . . 45
5A.4.1 Development and History of Internet Identifiers . . . 46
5A.4.2 The Role of a Namespace . . . 46
5A.4.3 Flat and Hierarchies . . . 47
5A.4.4 Internet Considerations . . . 48
5A.4.5 Technological Influences on Identity and Location: . . . 50
5A.5 Identifiers versus Locators: Parallel Insights into Library and Internet Systems . . . 51
5A.5.1 International Standard Book Number . . . 52
5A.5.2 Digital Object Identifier . . . 52
5A.5.3 URI, URL, and URN . . . 53
5A.5.4 Internet Protocol Address . . . 55
5A.6 Search Versus Identification: Comparing Directory and DNS Approaches . . . 59
5A.6.1 HOSTS.TXT . . . 59
5A.6.2 X.500 . . . 60
5A.6.3 The Domain Name System . . . 61
5A.7 Evolutionary Pressures and Future Perspectives . . . 66
5A.7.1 Metadata and Query Qualification . . . 68
5A.7.2 DNS as Computational Resource . . . 69
5A.7.3 Multicast DNS and Apples and the '.local' Namespace . . . 69
5A.7.4 TOR Network and the '.onion' Namespace . . . 70
5A.7.5 Ethereum Name Service and the '.eth' Namespace . . . 71
5A.7.6 GNU Name System and the '.gnu' namespace . . . 71

5A.7.7 Innovation versus Integration: A Discussion . . . 72
5A.8 Conclusions . . . 73

5B Proof By Argument: Alternate Paths for the DNS 74
5B.1 Purpose of this Chapter . . . 74
5B.2 The Case for an Evolutionary DNS Innovation . . . 74
5B.3 Customised Responses: Query Qualification and Context-Awareness . . . 75
5B.4 Temporal Identities: for Permanent vs. Impermanent Names . . . 75
5B.5 DNS Data: Storage vs. Real-Time Computation . . . 76
5B.6 DNS as Computational Resource . . . 76
5B.7 Computational Integration with the DNS . . . 77
5B.8 Conclusions . . . 77
References for Chapters 5A and 5B . . . 77

6A The Five DNS Parameters: A Categorisation-Based Review 86
6A.1 Purpose of this Chapter . . . 86
6A.2 Introduction . . . 86
6A.3 The Domain Name System . . . 87
6A.3.1 Scalability . . . 88
6A.3.2 Flexibility . . . 88
6A.3.3 Currency and Speed . . . 91
6A.3.4 Privacy . . . 92
6A.4 Conclusions . . . 93

6B Proposed Model, Experiments, and Results: DNS and URL Computation 94
6B.1 Purpose of this Chapter . . . 94
6B.2 Problem Statement . . . 94
6B.3 Proposed Model . . . 96
6B.4 Implementation of Proposed Solution . . . 98
6B.4.1 Search Engine Service . . . 98
6B.4.2 Locational Service: Search By Country Method . . . 99
6B.4.3 Locational Service: Retrieve URLs For Country Method . . . 99
6B.4.4 URL Computational Service: Precompute URL Method . . . 100
6B.4.5 Parallelisation of DNS Resolution and URL Transformation . . . 100
6B.5 Deployment Methodology . . . 101
6B.5.1 Environment Setup . . . 102
6B.5.2 Azure Infrastructure Environment . . . 102
6B.5.3 DNS Infrastructure . . . 102
6B.5.4 Client Setup . . . 104
6B.5.5 Public Accessibility and Application Behavior . . . 104
6B.6 Comparative Methodology for Evaluating the Proposed Model . . . 105
6B.6.1 Considerations of Test Methodology . . . 105
6B.6.2 User Behaviour and Test Methodology . . . 105
6B.6.3 Data Collection Method . . . 106
6B.6.4 Environment Setup . . . 106
6B.6.5 Latency Measurement . . . 106
6B.6.6 Statistical Analysis - Comparison Between New and Old Method . . . 107
6B.6.7 Data Collection Over Time . . . 108
6B.6.8 Test Scenario Goals . . . 108

6B.7 Test Scenario and Results . . . 109
6B.7.1 Test Scenario Description . . . 109
6B.7.2 Australia East Region . . . 109
6B.7.3 United States West Region . . . 112
6B.7.4 United Kingdom Region . . . 115
6B.7.5 Proposed Model Evaluation and Future Considerations . . . 118
References for Chapters 6A and 6B . . . 122

III Future Work and Conclusions 125

7A Future Work A: Identifier Systems in 5G Core Networks 126
7A.1 Purpose of this Chapter . . . 126
7A.2 Identifier Systems in Cloud-Native 5G . . . 126
7A.2.1 Background and Challenges in Cloud-Native 5G . . . 126
7A.2.2 Session Communication Proxy and 5G Core Services . . . 127
7A.3 Considerations and Requirements . . . 128
7A.3.1 Proposed Solution . . . 129
7A.4 Service Discovery as Identifier System . . . 131
7A.5 Future Work: Shifting Roles in Service Discovery . . . 132
7A.6 Conclusions . . . 134

7B Future Work B: Context-Aware DNS Query Handling 135
7B.1 Purpose of this Chapter . . . 135
7B.2 Forward Looking Challenges in Network Security . . . 135
7B.3 Considerations and Requirements . . . 136
7B.4 Proposed Solution . . . 138
7B.5 Future Work: Towards a Semantic DNS . . . 138
References for Chapters 7A and 7B . . . 139

8 Conclusions 142

A Dynamic Virtual Machine Network Policy for Ingress Optimization 145

B A Retrospective on Workload Identifiers: From Data Center to Cloud-Native Identifiers 169

C Internet Identifiers: Survey of History, Challenges, and Future Perspectives 180

D Cloud-Native Service Mesh Readiness for 5G and Beyond 204

E Context-Aware Domain Name System (DNS) Query Handling 215

List of Figures

1.1 Thesis Organisation. 7

2.1 Identity Management Module. 13
2.2 Flowchart of Solution Design. 14

3.1 LISP Endpoint Identifier and Dynamic Locator. 20
3.2 Core versus Edge Split: The Newly Revised Edge. 21
3.3 IPv6: Identifier and Locator Transformation. 23
3.4 Cloud-Native Service Mesh: The Application-level Identifier for Name-based Routing. 24
3.5 PKI: Mutual Authentication within a Trust Domain. 26
3.6 Persistent Identifier for Serverless in Hyperscaler Environments. 28

4.1 Navigating the Research Proposition. 39

5A.1 Categorisation of Citations by Source. 45
5A.2 Namespace Scope and Roles. 47
5A.3 Global Namespace Coherence and Fragmentation Risks. 49
5A.4 Role of an IP Address: Example Implementations for Identity and Location Separation. 57
5A.5 Apple Relay: Anonymising Client's Identity. 58
5A.6 Ingress and Egress Relay: IP Address Identity, and Location [81]. 58
5A.7 X.500 and DNS: Search versus Identification. 62
5A.8 DNS Lookup Process: CNAME and NAPTR Records. 64
5A.9 Service Binding Record. 66
5A.10 Evolutionary Pressures and Expanding Namespaces. 67
5A.11 Evolution in Customised Responses. 68
5A.12 Classifier for Multi Namespace Forwarding. 73

6B.1 Recursive Resolvers: Local versus Open. 95
6B.2 DNS Resolvers Utilization: Source Asia Pacific Network Information Center [16]. 95
6B.3 Sequence Diagram: Component Interactions. 96
6B.4 UDP Versus DOH and DOT: Source Asia Pacific Network Information Center [20]. 97
6B.5 Systems Architecture: Precompute URL. 98
6B.6 Environment Setup. 103
6B.7 Environment Setup: DNS Infrastructure. 104
6B.8 Test Scenario: Australia East Region. 110
6B.9 Australia: Total Latency of Search Queries and User Navigation. 111

6B.10 Australia: Average Computation Time of Proposed Model versus Average Total DNS Lookup Time. 112
6B.11 Test Scenario: United States West Region. 113
6B.12 United States: Total Latency of Search Queries and User Navigation. 114
6B.13 United States: Average Computation Time of Proposed Model versus Average Total DNS Lookup Time. 115
6B.14 Test Scenario: United Kingdom South Region. 116
6B.15 United Kingdom: Total Latency of Search Queries and User Navigation. . . . 117
6B.16 United Kingdom: Average Computation Time of Proposed Model versus Average Total DNS Lookup Time. 118
6B.17 Global Average: Comparison Between the Proposed and Traditional Method. . 119
6B.18 Global: User Perception Latency. 120
6B.19 A Layered Approach: Evolving the Proposed Model. 121
6B.20 Sequence Diagram: Proposed Future Model. 122

7A.1 5G Control Plane Reference Architecture and the Inclusion of SCP. 127
7A.2 Enhanced Envoy Architecture. 130
7A.3 Scenario-3: SCP as Distributed Envoy Proxy. 132
7A.4 Service Discovery: Client Side Candidate Selection. 134

7B.1 User and Machine Process of Context-Aware Query Handling. 136
7B.2 Proposed Solution. 137
7B.3 Future State of Context-Aware DNS Query Handling. 139

List of Tables

3.1 Supporting Network Architectures, Namespace, Identifiers and Insertion Points. 27

5A.1 Search Strategy and Search String. 43
5A.2 Identity Models and Comparisons. 54
5A.3 Summary of Responses to Evolutionary Pressures. 72

6B.1 Millisecond and Percentage Improvement of Proposed Model Over Traditional Model for the Australian Dataset. 111
6B.2 Improvement and Percentage Improvement of Proposed Model Over Traditional Model for the United States Dataset. 114
6B.3 Improvement and Percentage Improvement of Proposed Model Over Traditional Model for the European Dataset. 117
6B.4 Improvement and Percentage Improvement of Proposed Model Over Traditional Model for the Global Dataset 119

Part I

Introduction and Development of the Proposition

Chapter 1

Introduction

1.1 Background

Often perceived as an interconnected web of content and information exchange, the Internet is a marketplace as distinct from a passive platform. Like any marketplace, its success is determined by active participation and the ability to respond to consumer demands. Competition within this marketplace depends on the use of identifier systems, a key component of the Internet infrastructure, to functionally align with consumer preferences and desires. The result is an identifier system constantly adapting to an evolving market as competitors differentiate their solutions to gain relative advantage. As a technology trend, this creates a powerful impetus for continued change.

When considering the identifier system's influence, it is not the identifier that determines the market; it is the complex relationship between the behaviour of the identifier and the system's ability to customise responses to users that determine the impact of the system.

The core of this research asserts that identity systems are the key to any network. Just as a language is defined by a consistent set of symbols and their usage, a network is defined by a consistent set of identities and their usage. However, when an identifier system is under pressure not to treat identifiers as constant indices in an invariant system but, rather, to incorporate behaviour, customisability and adaptability within the identity space, two dilemmas arise:

1. *How can an identity system simultaneously prioritise flexibility, privacy and performance when each parameter introduces system trade-offs?*

2. *How does a network maintain coherence when developing alternative identifier systems that potentially fracture the unified namespace that maintains the Internet's public communication system?*

These are the paradoxical tensions that reside in today's Internet. No one is in charge; it is just a market. The inflow of ideas in a competitive context creates strong pressures to offer unique solutions that are differentiated from the past. The result is that differentiation is neither coherent nor controlled, highlighting that the Internet does not have to be bound to a single identifier system.

The remainder of this chapter outlines the research motivations, detailing the preliminary steps that shaped the formulation of the research objectives, which are also included in this section.

An integrative thesis structure is presented, which is crucial for understanding the interrelationships and organisation of the overall thesis. Additionally, the research scope, delineating the foundational tenets of this study, is discussed. Finally, the key contributions of this research are highlighted, providing readers with a comprehensive understanding of what this chapter encompasses.

1.2 Research Motivations

Referring back to the commencement of this research – discussed further in Chapter 2 – the initial focus was exploring the complex semantics of identity and location deeply rooted in the Internet Protocol (IP) address namespace. This initial research proposal focused on a persistent and durable endpoint identifier that could allow the decorating of attributes on the namespace for categorisation purposes. The architecture incorporated mobility as a core network construct, allowing the proposed model to maintain endpoint tracking irrespective of location changes. However, this method revealed several limitations, including the ephemeral nature of IP addresses, which prevents the maintenance of a consistent identity. Moreover, the routing system was continually burdened with frequent updates due to the duality expressed in the IP identifier, serving both as an identifier – 'who' you are – and a locator – 'where' you are – leading to an overwhelmed routing system. Additionally, privacy concerns arose due to the overexposure of persistent identity tracking at the IP address namespace.

This initial path, a preliminary research step, led to a critical realisation: while the focus on identity and location was an important principle, addressing these challenges extended beyond the scope of the IP namespace. Therefore, the research evolved, maintaining its essence but shifting the approach to a higher level of abstraction. This research shift did not abandon the original goals but, rather, extended the study, focusing on how users and consumers interact in the digital world.

This research shift closely examined the nuances of distinguishing between identifiers and locators in the context of defining digital entities coherently. The discussion raises the question:

> *How do we identify a digital ephemera that exists in multiple locations yet is logically viewed as a single entity?*

The Internet's approach to this challenge has been through Uniform Resource Locators (URLs), which describe a document by specifying how to access it. This method is analogous to providing directions to a physical location. The URL is effective where there is a one-to-one mapping between the location and the resource that is being identified. However, similarly to the limitations in the IP namespace, the URL namespace also conflates identity and location. In contrast, the Uniform Resource Identifier (URI) is a persistent identifier that describes the document, irrespective of its location. While URIs answer the fundamental question to some degree, the development of location-independent identifications has proven challenging, leading to continued reliance on URLs for web references.

The human-friendly name within the URL scheme is derived from the Domain Name System (DNS). The use of names within the references determines how to access resources. In theory, a name passed between two entities should resolve to the same digital resource. This is only achievable because names are allocated from a unified namespace, and the resolution of the name into network addresses is consistent and coherent. Market competition dynamics for digital presence naturally encourage all participants to be bound within this unified namespace residing

in the DNS. Moreover, for the Internet to continue functioning as a public communications system, it is essential to remain within the same namespace to maintain coherence.

The observation that emerged from this research was that the advent of individualised or autonomous identifier systems risks fragmenting the Internet's namespace, eroding the value of a unified and cohesive communication network. Therefore, examining various identifier system scopes and their role is essential to understanding the breadth of literature within and beyond the Internet namespace.

Proposals for alternative root systems introduce new namespaces to the Internet and have displayed potential fragmentation risks, creating scenarios wherein participants may find themselves in namespaces A or B. This research does not explore these potentially disruptive revolutionary paths, as radical changes that disrupt an incumbent system often do not integrate seamlessly (such as by maintaining referential consistency between network namespaces). This issue is further highlighted by the difficulties in referencing or transferring identifiers from one context to another, as they are locally significant. For a new system to disrupt the incumbent, it must demonstrate superiority in multiple orders of magnitude. However, it is understood that no technology is considered immutable, and the DNS is no exception; it will continue to experience an array of pressures to evolve.

Instead of investigating revolutionary paths such as alternate namespaces, the aim of this research is to examine both the DNS as an identifier allocated from a uniform namespace and how it can respond to evolutionary pressures, which are further discussed in Chapter 5A. This area of research requires a multi-disciplinary review to compare and contrast various identifier systems, to enhance the understanding of this field of study, and to initiate a discussion about the research field's future direction.

1.3 Research Objectives

The research objective aims to prove the **research proposition detailed in Chapter 4**, which includes the background, arguments and validation support. The research proposition statement asserts that a computational DNS resource can enhance the identifier system's flexibility without compromising performance and privacy. To address this proposition, we introduce the questions raised in Section 1.1:

1. *How can an identity system effectively prioritise flexibility, privacy and performance when these parameters inherently involve trade-offs?*

2. *How does a network sustain its coherence when the development of individual identity systems potentially disrupts the fundamental nature of maintaining a public communication system?*

In alignment with the research proposition in Chapter 4, as well as the commencing research questions, the following research objectives are included.

Objective 1: Present Systematic Literature Review
Conduct a systematic literature review to explore various identifier systems, including and beyond the traditional DNS, assessing their functional purposes and contributions. This aims to examine the evolutionary pressures faced by identifier systems.

Objective 2: Present Proof by Argument within the DNS Framework
Present proof by argument for integrating alternative features within the identity space, proposing DNS as a computational resource.

Objective 3A: Propose DNS as Computational Model
Propose DNS as a computational model designed to address practical challenges in the current Internet environment. This involves enhancing system flexibility without compromising privacy and speed, detailing the implementation, system architecture, and its integration with web applications and Internet infrastructure.

Objective 3B: Deployment of Proposed DNS Computational System
Deploy the proposed DNS computational system in real-life network conditions to evaluate its performance, including customised web engine/DNS responses and reduced end-to-end client-server latency within a globally distributed network.

Objective 3C: Comparison of the Proposed DNS Computational System
Validate the performance of the proposed DNS computational system compared to traditional DNS systems, aiming to demonstrate improvements and efficiencies in real-world applications.

Objective 4: Demonstrate the Impact of Future Work
Demonstrate the application of this research on future work, particularly its relevance to next-generation 5G telecommunications systems and the DNS. This explores how the findings may influence the evolution of Internet infrastructure and telecommunications technology.

1.4 Progressive and Integrative Research Structure

In addition to outlining the thesis structure, this section highlights the progressive research structure achieved through the sequential building of each chapter. While there are eight chapters, the thesis is organized into three categories, transitioning from identifying challenges and evolving the research purposes to formulating and validating actionable propositions, culminating in discussions of future work and conclusions.

Chapter 1 – Overview: In this opening chapter, readers are introduced to the research area, including the motivations behind the study, its intended contributions, and the research approach. It sets the stage for understanding the research progression and how the thesis is structured throughout the remaining chapters.

Chapter 2 – Initial Investigation and Lessons Learned: The commencement of the research investigated the role of Software-Defined Networking (SDN) systems to act as a global identity controller for the IP address namespace. Leveraging the SDN controller's global topology visibility, this study examines the management of virtual network interfaces and their attachment to endpoints, focusing on mobility. Despite achieving success in specific virtualised environments, this approach reveals significant shortcomings within the broader Internet context. The challenge extends beyond applying the proposed identity controller model in SDN, pointing to a fundamental issue in asserting identity at the IP address level.

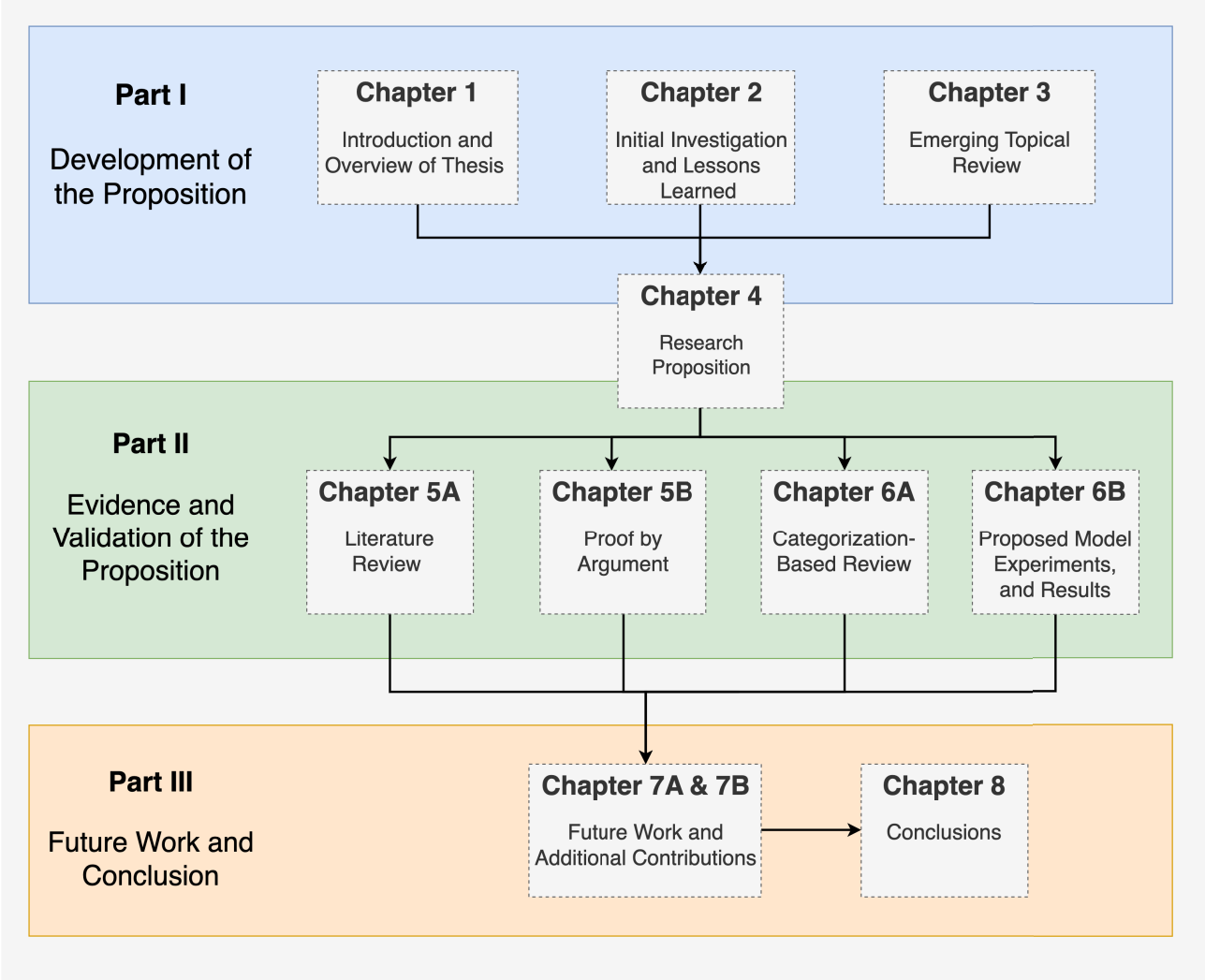


Figure 1.1: Thesis Organisation.

Specifically, there are areas for improvement in the proposed model's attempt to overcompensate for the semantic overload in IP addresses through a mapping function and state tracking of endpoints to their physical hosts across different data centres. The challenges include an overwhelmed routing system burdened with excessive routing advertisements and notifications and privacy concerns due to the overexposure of mapping identity to the IP address.

While this study reveals several lessons learned, it also opens avenues for research inquiry into how Internet identifiers have evolved to address these challenges.

Chapter 3 – Emerging Topical Review: Challenges identified in Chapter 2 prompted an inquiry into the evolution of identifiers and locators within network architectures. This chapter offers a topical review of the evolution of identifiers and locators. It examines how various architectures have responded to the overloaded semantics of the IP address, specifically addressing the challenges outlined in Chapter 2. It highlights a trend towards location-agnostic identifiers supported by names in the DNS, which goes beyond a simple lookup system for name-to-address mapping. This topic warranted further examination into identifier systems and the DNS evolved roles.

Chapter 4 – Research Proposition: Observing the insights from Chapter 3 regarding the DNS adaptation and evolving roles, this chapter introduces the research proposition. It outlines the challenges associated with the DNS and proposes focusing on the evolutionary path of the DNS rather than pursuing revolutionary changes that could disrupt the utility of the network as a public system and lead to namespace fragmentation. The research is based on the customisation observed in the DNS, proposing that the DNS as a computational resource could incrementally introduce new functionalities without trade-offs observed in the traditional DNS. This chapter discusses the research methodology behind this proposition, including a mixed (qualitative and quantitative) research approach, which is further discussed in Chapters 5A/5B and Chapter 6A/6B.

Chapter 5A – Literature Review: To begin proving this proposition, a systematic literature review was conducted. This chapter introduces initial research questions that guide the examination supporting the proposition, focusing on the field's current state. It extends the inquiry beyond Internet systems to other disciplines, such as digital libraries and their approach to identification systems. The questions generated saw this research evolve into a multidisciplinary study. The literature review explores how the role of the DNS has shaped and influenced other identifier systems, revealing that alternative identifier systems are isomorphic to the DNS. It outlines the importance of understanding namespaces beyond the DNS and their functional purposes. The literature review then surfaces non-obvious examples of evolution in DNS lookup and namespace, indicating potential evolutionary directions it may take.

Chapter 5B – Proof by Argument: The literature review in Chapter 5A revealed potential evolutionary directions for the DNS. This chapter extends the exploration, asserting that no technology is immutable and all will inevitably experience pressures for change, with the DNS no exception. It argues that the evolving Internet marketplace will continue to pressure identifier systems such as the DNS to evolve their traditional role from simple lookup systems to computational resources. The support for this argument provides examples of functionalities that a computational DNS could introduce. Given the limited literature on this topic, this chapter uniquely contributes by commencing the dialogue for further discussion.

Chapter 6A – Categorisation-Based Review: This chapter progresses the discussion on computational DNS by examining the current DNS system by categorising the analysis based on five parameters, treating this chapter similar to a taxonomy. This approach prepares for quantitative analysis in Chapter 6B, strengthening the proposition by prioritising these parameters through a computational module within the DNS infrastructure.

Chapter 6B – Proposed Model, Experiment and Implementation: Building on the qualitative insights from previous chapters, this chapter adopts a quantitative approach to provide evidence for the proposition. It explores the practical challenges a computational DNS approach attempts to solve, presenting a proposed model and implementation details. The effectiveness of the proposed model is evaluated through a deployment methodology, which involves integrating the model's solution into a real-life global network. This setup tests the model's performance across a geographically distributed network. The chapter outlines the quantitative methods used for data collection, analysing the data, and comparing the traditional DNS model with the proposed computational DNS model through a set of results. Furthermore, key considerations regarding potential directions are discussed.

Chapter 7A – Future Work A – Identifier Systems in 5G Core Networks: This chapter focuses on the future applications of this research, outlined in the IEEE Access publications

detailed in D). In collaboration with Intel Corporation, the research contributed to designing a resilient and low-latency network platform for 5G applications. In such environments, an identifier system is essential for prefetching the connection parameters required to establish connections with target workloads. The study revealed an absence of key functionalities in service discovery. Addressing this gap, the application of this research introduces an adaptive identifier system and its potential future directions.

Chapter 7B – Future Work B – Context-aware DNS Query Handling: This chapter focuses on the future applications of this research, including a semantic DNS created from a patent (detailed in Appendix E). It explores the application of a computational DNS to evolving identity models. Unlike traditional systems that provide uniform answers to everyone, this system delivers answers relative to the querier's context. This work stems from the thesis outlined in Chapter 5B and its potential direction for future research.

Conclusion: This concluding chapter synthesises the research findings, validating the research proposition. It highlights the insights gained through the development of the computational DNS and its potential advantages, the imperative for continued investigation in this field, and who could benefit from the proposed model. Central to this study, two insights have emerged which inform the continuation of this research journey. Fundamental to connectivity, the first insight concerns the role of a unified namespace and the implications of fragmentation. The second insight centres on the concept of relative identity and context-sensitive framework and could potentially redefine the role of identifier systems, warranting further exploration.

References: Finally, rather than consolidating references at the end of the document, this thesis presents references after each chapter. A consolidated list would have surpassed 200 entries, making it difficult to review and determine their relevance to specific chapters. Specifically, for chapters divided into two parts – 5A and 5B, 6A and 6B, and 7A and 7B – references are located at the end of the 'B' sections.

1.5 Research Scope

Internet identifiers and supporting systems for resolution are a broad field of study. In addition to the research objectives, which define the research outcomes, this section describes the parameters bound to the field of study. Rather than technology confinement, the parameters reflect the nature of the expectations of an identifier system and its interaction with the Internet and its users. The categorisation is as follows:

1. **Exposure versus Anonymity:** Firstly, a classic tension is the balance between exposure and privacy. For communication between two parties, the initiator must reveal their identification to the receiver while avoiding overexposure in time and generality, which elevates risk. The research scope considers the level of publication required, given the present Internet identifiers and supporting systems in use. Within the initial research proposal in Chapter 2, privacy research scope was required to be included, and is further outlined in Chapters 5A, 6A, and 6B.

2. **Namespace Role:** The role of an identifier is determined by the functional purpose of the namespace. Commonly, identifiers within a namespace are understood to distinguish distinct instances of objects. In addition to distinguishing objects, the namespace may include a 'locator.' This implies that the identifier states what the object is and where to find it, as described in Section 1.2. In essence, the identifier serves a dual function of

identity and location. However, concerning this study, the scope of the namespace must be further defined. This is further outlined in the literature study in Chapter 5A.

3. **Network Coherence and Reference Integrity:** In the context of a communication network, referential integrity is paramount. As described in Section 1.2, when an identifier is passed from one entity to another, it must consistently refer to the same object. If the identifier refers to different objects at different times, the essence of a reliable communication network is lost. Maintaining referential integrity is essential to ensure that namespaces are interpreted uniformly across the entire network. Therefore, this principle is within the scope of the research, which supports DNS research as the uniform namespace that maintains the network as a public communication system.

4. **Identifier System – Query Handling and Custom Responses:** Directing Internet users to the closet content to optimise their application experience requires custom handling of their initial queries. Customisation is included in the research scope as a core principle to ensure flexibility without trade-offs. For instance, a DNS query must disclose aspects of the client's identity to drive behaviour within the system and return an optimal response. While this approach enhances speed and user experience, it raises important privacy concerns.

1.6 Key Contributions

Key contributions and outcomes from this research are highlighted through several publications, each aligned with the chapters in this thesis, as follows.

1. **Dynamic Virtual Machine Network Policy for Ingress Optimization (Issued Patent)**
   Publication Venue: United States Patent and Trademark Office
   Brief Description: Assertion of Identity at the Internet Protocol that can associate attributes in the namespace for classification.
   Relevance to PhD: This patent resulted from the initial research on identifier systems challenges in the IP namespace. The proposed model was successful in scoped environments; however, while maintaining the original essence of the Internet namespace, the question of the research question deepened. This chapter drew from lessons learned and the academic inquiry to develop an emerging topic review in Chapter 3.
   Associated Appendix: Publication can be found in Appendix A.

2. **A Retrospective on Workload Identifiers: From Data Center to Cloud-Native Networks (Accepted)**
   Publication Venue: IEEE Access
   Brief Description: An emerging topic review of network architectures supporting identifiers and locators and their evolution to cloud-native.
   Relevance to PhD: This chapter was pivotal in developing the research proposition. The chapter gives evidence of the evolution of workload identifiers from IP addresses to names and the importance of the Internet Naming System, the DNS, the only namespace that keeps the Internet together. This contribution is outlined in Chapter 3.
   Associated Appendix: Publication can be found in Appendix B.

3. **Internet Identifiers: Survey of History, Challenges, and Future Perspectives (Accepted)**
   Publication Venue: IEEE Access
   Brief Description: This systematic literature review both contributes to and seeks to answer a defined set of research questions. Secondly, this chapter synthesises a diverse field of study to provide a multi-disciplinary view of Internet identifiers and their supporting systems.
   Relevance to PhD: The contributions of this study in Chapter 5A strengthened the research proposition and offering a clear direction for the proposed DNS model outlined in Chapter 6B.
   Associated Appendix: Publication can be found in Appendix C.

4. **Cloud-Native Service Mesh Readiness for 5G and Beyond (Accepted)**
   Publication Venue: IEEE Access
   Brief Description: Shared primary authorship in collaboration with Intel Corporation. This study discusses the complexities of 5G networks in cloud-native deployments and service mesh enhancements. This research contributes by highlighting identifier system requirements for inclusion in future 3GPP guidelines.
   Relevance to PhD: This study reinforces the proposition, highlighting its application as a foundational component for future research in 5G cloud-native systems, outlined in Chapter 7A. *Associated Appendix:* Publication can be found in Appendix D.

5. **Context-aware Domain Name System (DNS) Query Handling (Issued Patent)**
   Publication Venue: United States Patent and Trademark Office
   Brief Description: Introduces context-aware approach to DNS query handling and its application to relative identities.
   Relevance to PhD: This study strengthens the proposition, highlighting its application as a foundational component for DNS context-aware systems, outlined in Chapter 7B. *Associated Appendix:* Publication can be found in Appendix E.

Chapter 2

From Lessons Learned to Research Inquiry: A Preliminary Step

2.1 Practical Invention and Academic Inquiry

This chapter provides the synthesis between practical application and academia. As outlined in Chapter 1, the initial research for addressing the identifier systems challenges commenced at the IP namespace. The proposed model was successful in specific virtualised environment cases and is an issued U.S. patent titled *"Dynamic Virtual Machine Network Policy for Ingress Optimisation"* (detailed in Appendix A). However, the observation of the proposed model revealed several limitations, as described in Sections 2.7 and 2.8, offering lessons to consider when broadening the model's scope to accommodate the Internet's requirements. Furthermore, the related work is discussed in Chapter 3. Disclaimer: The descriptions and explanations provided in this chapter are limited to the content explicitly detailed in Appendix A.

2.2 Background

IP addresses are foundational to any communication system. They provide instructions for a service to deliver a message – for example, informing a postal service where mail should be delivered. In telephony, an address references a handset to initiate a phone call between two parties. In the context of the Internet, IP addresses are attached to the network interface of every computing machine. When transmitting messages on the network, the IP address determines the path a packet will take to reach the destination. The consistent aspects of an addressing scheme across all communication systems surround endpoint identity and its location.

Today, identity and location have become increasingly important topics, with issues distinct from when the Internet was first designed. Consider the digital environment; in the era of the mainframe, these machines were located on various floors and buildings. When transferring files between source and destination, the identity and location in the address were firmly enmeshed. The addressing scheme could make this gross simplification of conflating identity with location as the endpoints were assumed to be fixed. Fast forward to the megatrends in the era of cloud computing [1], application artefacts evolved to be uncoupled from their underlying resources through the technique of virtualisation [2]. Mobility became an attractive property as

application designers have flexibility and choice for migrating applications across multiple cloud providers. To provide this mobility awareness, a constant address was maintained irrespective of endpoint migrations. This was required as application security treated the IP address as an identity.

2.3 Problem Statement

In essence, the routing system and the addressing plan make the gross simplification that an IP address is both 'who' and 'where' you are.

As described in section 2.2, the advent of cloud-specific application constructs, such as Virtual Machines (VMs) [2] or Containers [3], demonstrated the ability to migrate between locations or cloud platforms and simultaneously introduce network considerations. For instance, a summarised prefix would ensure traffic is routed to the correct location in site-specific routing. However, there are two challenges to moving a workload between platforms or regions. Firstly, a sub-optimal path introduces latency due to traffic returning to the primary location. Secondly, should such migration events become frequent, ingress and egress tunnel mapping techniques must be employed for edge location mapping.

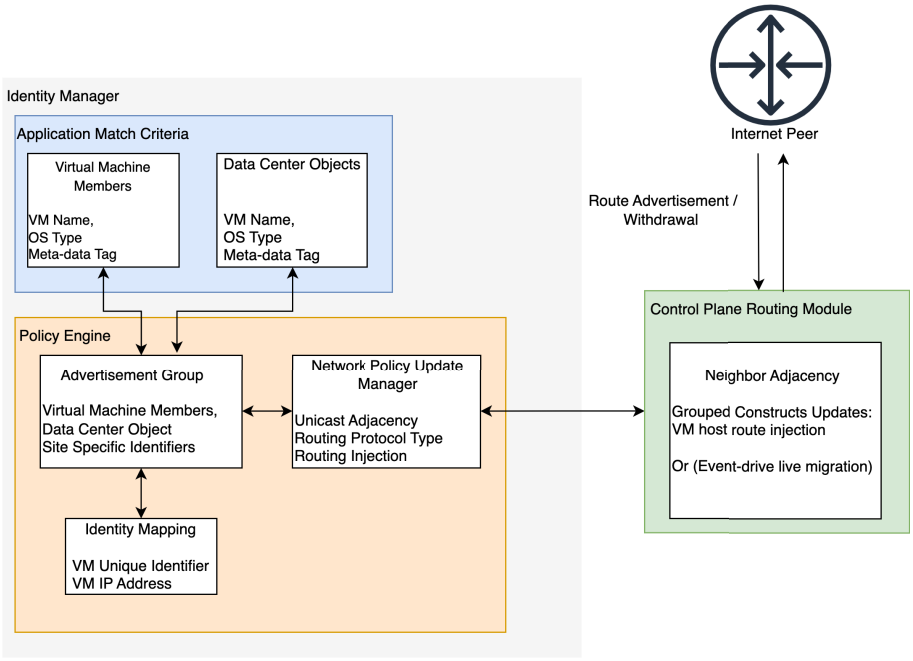


Figure 2.1: Identity Management Module.

2.4 Proposed Model

The proposed model includes a logically centralised Software-Defined Network controller (SDN) [4], [5] that coordinates the IP namespace and serves its function as an identity controller. The primary objective of this approach is to promote a location-agnostic separation architecture at the IP namespace. Notably, VMs retain a consistent address despite their migrations across the network. Consequently, the design integrates a namespace designed specifically to classify workloads as VMs retained a constant address. This namespace is central in computing the

division of prefixes to guarantee that when applications are moved between locations, the network's routing metadata will undergo the requisite modifications. The overarching intent was to ensure that the reassessment of the forwarding state across wide-area network (WAN) boundary routing components navigates traffic accurately to its destination. Appendix A describes the method and implementation.

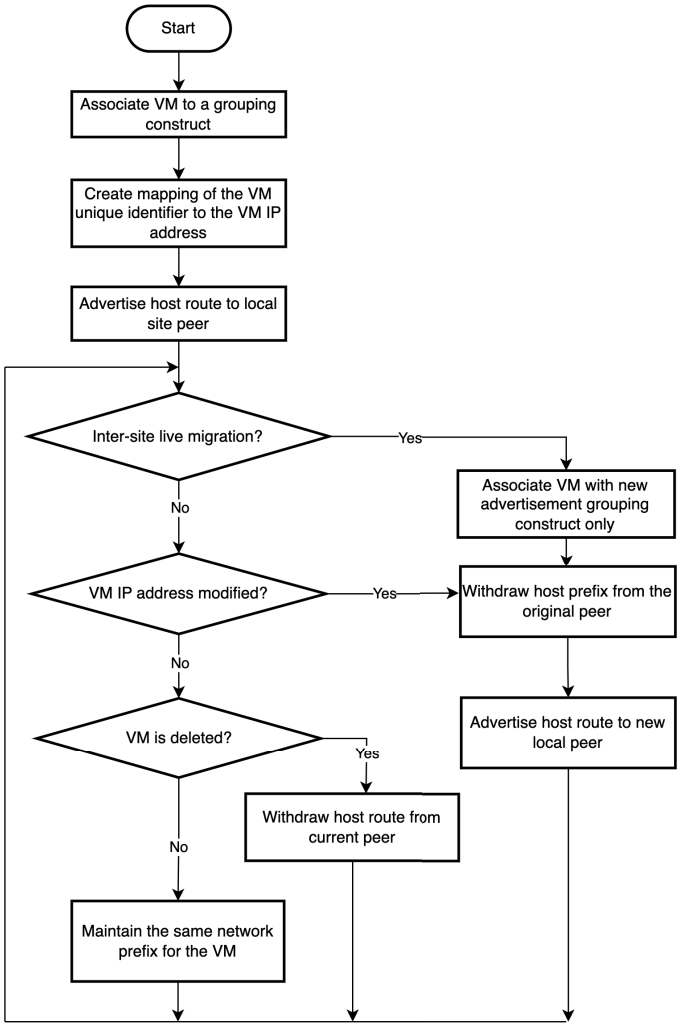


Figure 2.2: Flowchart of Solution Design.

2.5 Solution Design

The presented solution offers a method for orchestrating ingress traffic engineering in active/active cloud platforms to maintain workload identity. The approach centres around the classification of site residency for virtual machines, particularly those essential for external network advertisement. Identified virtual machines are subsequently aligned with their site-specific awareness, enabling advertisement via dynamic routing policies.

As depicted in Figure 2.1, the identity manager contains two sub-systems: the application matching criteria and the policy engine. Within this framework, the application matching criteria assimilate the virtual machine members' list and data centre objects, orchestrating a match with the application containers for network policy for route advertisement.

The solution is further structured as follows:

1. **Application Matching:** This container abstracts applications from their physical network addressing, enabling matches with virtual machine members and data centre objects.

   The site-specific ID (SSI) is bound to the application, ensuring members of the application container and only members without this construct get advertised. The SSI, a unique identifier for the specific cloud platform site, is classification-mapped to either hypervisor UUID, virtual machine manager objects, or metadata labels/tags.

   The benefits of dynamic classification and mapping include:

   (a) Dynamic grouping of applications for site-local peer advertisement, independent of traditional network segmentation.

   (b) A centralised management API framework for provisioning virtual machine applications with advertisement policies.

2. **Policy Engine and Route Advertisement:** Linked to the advertisement group, the policy engine triggers an API call to the logical control plane routing module with a list of virtual machine IP addresses, enabling advertisement through the desired routing protocol. Subsequently, a /32 host route is advertised per VM via the Wide Area Network (WAN) router, as depicted in Figure 2.2.

2.6 Challenges Observed in Proposed Solution

As outlined, the solution ensures a constant workload identity by maintaining /32 prefixes. This approach allows workloads to migrate while retaining network reachability, irrespective of location. However, a notable challenge arises when a workload or endpoint relocates to a different network location. In such cases, the routing system must be updated to accurately reflect the new location of the /32 IP address.

The primary implication of this dynamic is the need for routing tables within the network to be dynamically updated. These updates involve either adding or removing /32 routes from the routing table to ensure accurate routing information. For example, consider the following scenario.

Set of All /32 Allowable Prefixes:

- R: The set of all /32 prefixes currently in the routing tables.
- R_{add}: The set of /32 prefixes to be added due to workload migration.
- R_{remove}: The set of /32 prefixes to be removed due to workload migration.

Upon Workload Migration:

- The routing table R is updated by incorporating R_{add} and excluding R_{remove}.
- Consequently, the total number of /32 prefixes in the routing table after the update is given by $|R| + |R_{\text{add}}| - |R_{\text{remove}}|$.

This scenario highlights a critical aspect: as the frequency of routing table updates and the volume of routable addresses increases, the burden of maintaining a constant identity for each workload shifts significantly to the routing system. This is primarily due to the semantic

overload inherent in the use of IP addresses as both identifiers and locators [6]. Such fluctuations in the routing table present considerable challenges, impacting the routing system's efficiency and stability.

2.7 Need for Continued Research

In the realm of network systems research, the practical implementation of theoretical concepts frequently reveals unexpected challenges. In this case, a burden emerged on the underlying routing system, which became inundated with an unprecedented volume of route modifications due to the surge in host-specific prefixes. This development paradoxically reduced the efficiency of aggregability of route prefixes, a fundamental benefit of routing design [6].

In the context of the rapidly transforming digital landscape, a foundational aspect warranting re-examination is the interplay between identity, location and the core infrastructure of the Internet – the IP addressing system. The Internet's original design principles enmeshed the concepts of 'identity' and 'location', inherently tying an entity's identity to its locational address. While this binary approach was pivotal in the Internet's early development, it has surfaced as a potential bottleneck limiting mobility and cloning. The evolution of application artefacts, now composed in advanced programming languages, has reduced their dependency on foundational platforms. Projecting this trend into the future suggests an evolving paradigm wherein identity becomes increasingly decoupled from physical infrastructure, morphing into an intrinsic attribute of the application itself. This topic is relevant in an environment where applications traditionally bound to specific hardware platforms evolve into more abstracted, platform-agnostic entities.

2.8 Conclusion: Topical Review Required

As described in this chapter, the conceptualisation of IP addresses as identities and the assertion of such at the network layer presents many drawbacks. Principal among these is the undue burden placed on routing systems due to the advertising of more specific host prefixes, which disrupts the efficiency of prefix aggregation.

The modern trend towards applications developed with complex componentry introduces further complexities. Such applications often deploy multiple replicas performing identical functions, necessitating the involvement of middleware such as network-based load balancers. This involvement renders the IP identifier weak and mutable, subject to transformations by network middleware that rewrites source and destination IP addresses.

Moreover, a critical aspect of this discourse is privacy. Asserting identity at the IP level exposes users and applications to potential passive monitoring, a practice that can be construed as a network attack. This raises significant privacy concerns, particularly in an era in which digital security is paramount.

The subsequent phase of this research aims for a deeper understanding of the levels at which identifiers and locators are exposed within the protocol stack. Chapter 3 contributes to this research by thoroughly examining a broad spectrum of evolving network architectures and related work. It explores the evolution of identifiers and locators across various levels, thereby aiding in the development of focused research questions and defining the scope of the study.

References for Chapter 2

[1] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, "The characteristics of cloud computing," in *2010 39th International Conference on Parallel Processing Workshops*, IEEE, 2010, pp. 275–279.

[2] B. Walters, "Vmware virtual platform," *Linux journal*, vol. 1999, no. 63es, 6–es, 1999.

[3] A. Tosatto, P. Ruiu, and A. Attanasio, "Container-based orchestration in cloud: State of the art and challenges," in *2015 Ninth international conference on complex, intelligent, and software intensive systems*, IEEE, 2015, pp. 70–75.

[4] T. Koponen, K. Amidon, P. Balland, *et al.*, "Network virtualization in multi-tenant datacenters," in *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, 2014, pp. 203–216.

[5] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian, "Fabric: A retrospective on evolving sdn," in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 85–90.

[6] D. Meyer, L. Zhang, and K. Fall, "Report from the iab workshop on routing and addressing," RFC Editor, RFC 4984, Sep. 2007, `http://www.rfc-editor.org/rfc/rfc4984.txt`. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc4984.txt`.

Chapter 3

Emerging Topical Review: Identifiers and Locators Evolution

3.1 Purpose of this Chapter

Building upon the findings in Chapter 2, which highlighted the array of challenges associated with identity at layer three, this chapter explores the evolution of workload identifiers and locators to aid the development of the research proposition. It examines how network architectures have responded to the semantic overload in IP addresses. This analysis resulted in a publication titled *"A Retrospective on Workload Identifiers: From Data Center to Cloud-Native"* which was accepted and published in IEEE Access (found in Appendix B).

This chapter, alongside the publication, discusses an emerging topic by focusing on how application migration to multiple clouds has transformed the network into a reactive element that supports cloud consumption and application needs. It explores how, through each generation of network architecture, the types of identifiers and locators have evolved at different levels of the protocol stack. The identifiers and locators type is defined by the isolation boundary and how the architecture considers semantic overload in the IP address. As discussed in this chapter, these considerations are critical to understanding how identifiers impact the overall network architecture. This chapter contributes an industrial retrospective and topical review of how the schemes and mechanisms for identifying and locating network entities have evolved in traditional data centres and how they match cloud-native application requirements. Specifically, each application artefact that forced necessary changes in the identifiers and locators is evaluated. Finally, the common themes are highlighted from observations to determine the investigation areas that may play an essential role in the future of cloud-native networking.

3.2 Introduction

In today's dynamic environment, the network is under pressure to deliver connectivity and identity across multiple clouds in an agile manner. The architectural support for identifiers and locator split has solidified their role as an essential function to keep pace with application growth and flexible cloud computing models. From a network access-control perspective, IP addresses could be viewed as identity labels for authorisation in a network zone model and, in parallel, as routing instructions for networked elements to guide a packet to the destination. In

each generation of network architectures, there is a realisation that the IP space cannot serve two purposes of identity and location in virtualisation and cloud computing environments. A technique to overcome the semantic overload of IP addresses was to split the IP space for persistent identity and dynamic location through encapsulation protocols as a point of indirection. However, as the new generation of cloud applications emerges, the IP space has become weak and mutable to maintain its role as a persistent identifier. For example, microservice applications heavily depend on network proxies, known as sidecars, to provide application availability and resiliency. In such network rendezvous architectures, the IP address has no lasting semantics [1]. One observation that can be made is that the cloud-native architectures have shifted workload identifiers and created new namespaces to support service discovery, routing and identification for applications to extend administrative boundaries.

This chapter contributes to a discussion of the schemes and techniques for identifying and locating network entities that have evolved in traditional data centres and how they match cloud-native application requirements. The distinction this chapter makes is noting how each application artefact has forced a shift in workload identifiers required for discovery, routing and identification, which is not obvious from previous work from a comparison point of view. For example, the rationale behind identifiers, locators and insertion points required for today's cloud-native networks is emphasised rather than focusing on identity and location split in its generality. In addition, this chapter contributes to an increased awareness of evolved cloud application artefacts that lead to the functionality required for service mesh and its impact on the use of location-independent identifiers.

Finally, the chapter closes by considering the discussed techniques, state-of-the-art opportunities and directions for future investigation into the critical elements foundational to a cloud-native network.

3.3 Location/ID Separation Protocol

3.3.1 Semantic Overload in the IP Address

As background, at the Internet Architecture Board (IAB) workshop [2] concerns centred around the current routing information base (RIB) size of the border gateway protocol (BGP) [3]. One of the underlying causes of routing scalability was identifier/locator overload in the IP address. Enterprises were advertising provider-independent prefixes across sites to avoid server renumbering. From a firewall point of view, the server's IP address was its identity; therefore, IP renumbering would be an operational burden.

With the current IP architecture, it seemed impossible to have a single IP space that could serve two purposes, and a split seemed necessary to scale the routing system. This conclusion was the supporting material that led to the creation of the locator/ID separation protocol (LISP) [4]. There are several other proposals to separate identifiers and locators, such as HIP [5], Mobile IP [6] and RINA [7]. To get started, this study selected LISP, which addressed the workload mobility challenges in traditional data centres.

The initial technique of LISP was to provide topological aggregation, leveraging a locator approach for the IP routing system. The network's edge contains more specific details needed to reach the endpoint. Provider-independent networks resided behind LISP networks and behaved similarly to network address translation (NAT) [8]. However, in contrast to NAT, the translation state in LISP was not held in a network device but stored in the original IP packet. In

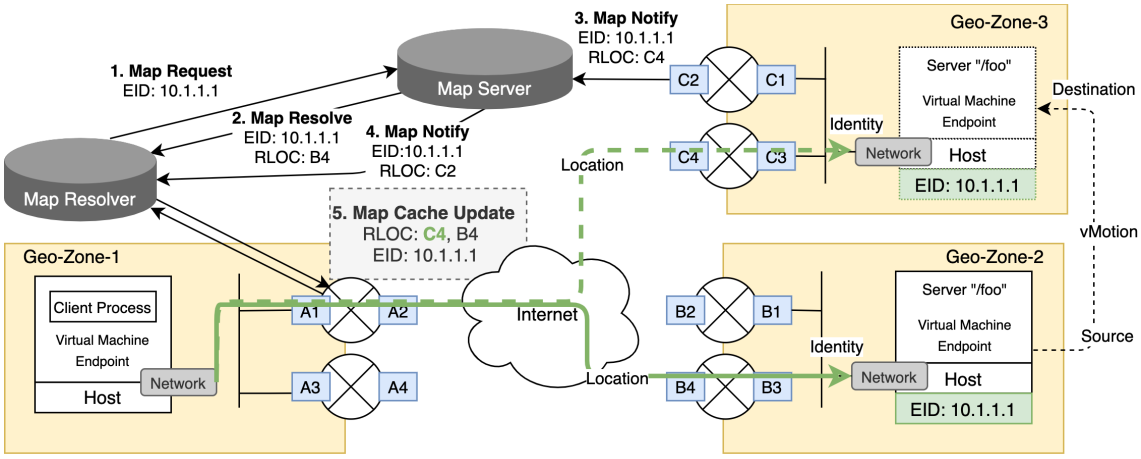


Figure 3.1: LISP Endpoint Identifier and Dynamic Locator.

LISP, the original IP packet is encapsulated with an outer IP header that contains the locator information.

3.3.2 LISP for Virtualised Data Centres and the Virtual Machine Artefact

While the initial motivation of LISP was to solve a route scalability issue, it became a tool to address mobility challenges created by virtualised data centres (VDCs) [9]. VDCs refer primarily to a hosted virtualisation platform, where servers are decoupled from their underlying hardware resource. This disaggregation allows the infrastructure layer to isolate memory, CPU, and storage to each virtual machine (VM) instance managed by the hypervisor [10]. Applications within the VMs artefact remained unaware of this abstraction. As VDCs became mainstream, VM portability between physical hosts became an appealing function for disaster recovery and operations maintenance, as well as a method to utilise capacity in active-active data centres (DC).

3.3.3 Identity and the IP Namespace

LISP naturally addressed the VDC's needs by allowing VM migration between data centres without burdening the routing system. The insertion point of LISP was at the edge of the data centre core network, where the endpoint identifiers (EIDs) – in this case, VMs – are mapped to routing locations (RLOCs) at the geozone level. The VMs are not globally routable and are topologically independent from the underlying network. For example, in Figure 3.1 VM "/foo" migrates from Geo-Zone-2 to Geo-Zone-3 without disrupting the existing transport-layer session from a client in Geo-Zone-1. Through LISP control-plane notifications, Geo-Zone-1 receives a Map-Notify to inform that "/foo's" IP address of "10.1.1.1" has migrated to Geo-Zone-3. The LISP-speaking router "A1" updates "10.1.1.1" entry to map to "C4" in Geo-Zone-3. This map and encapsulation with a dynamic approach allow the VM's identity namespace to be preserved during the migration procedure.

In summary, LISP introduced a disjointed IP namespace, wherein IP is separated as an identity (i.e., EID) attribute and IP is used for location. This separated namespace technique in the data centre network architecture allowed VMs to preserve their identity while taking advantage of the virtualisation operational model, which proved to minimise service disruption based on

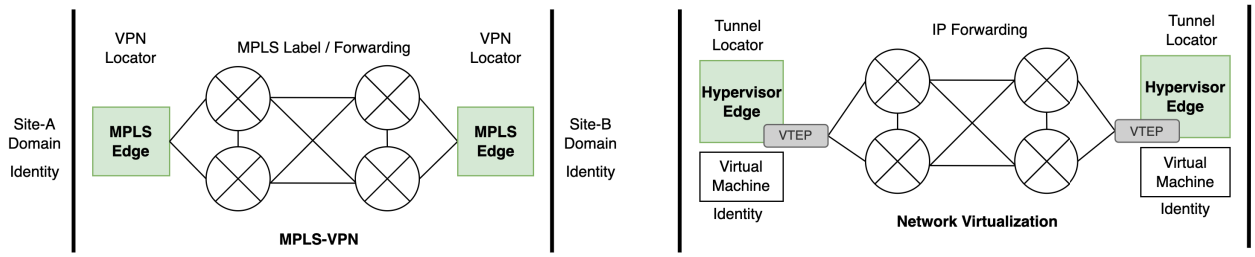


Figure 3.2: Core versus Edge Split: The Newly Revised Edge.

its mobility usecase. LISP's mapping system introduced a new approach to providing network control at the edge of the data centre. The arguments for the separation of edge vs. core become an important approach for the IP routing system [11] and for providing a network insertion point to support ephemeral and mobile data centre workloads.

3.4 Network Virtualisation

3.4.1 Core vs. Edge Split

In the section above, LISP's identifier mappings are only relevant at the edge (i.e., VMs), while the core of the network is forwarding IP packets to the correct Geo-Zone (i.e., location). This architecture is similar to Multi-protocol Label Switching (MPLS) [12]. In MPLS-VPN [13], the provider core (P) is solely focused on forwarding packets based on MPLS labels, while the provider edge (PE) has specific details of the neighbour domains (i.e., identifier). This separation allowed the PE to create multiple virtualised paths, which cater to overlapping address spaces over the same physical network.

3.4.2 Network Virtualisation for Cloud

In the quest for a purpose-built network for a cloud computing architecture, network virtualisation (NV) [14] replicated the same separation of core and edge demonstrated in the LISP and MPLS design and solved new challenges created by cloud computing.

In the technique of NV, the network becomes a reactive element to the application and cloud needs, promulgating the hypervisor vSwitch [15] as the newly revised edge, depicted in Figure 3.2. This is also known as the Network Virtual Edge (NVE) [16]. Furthermore, cloud computing forced the network architecture to create a unifying abstraction across hypervisors. In contrast to LISP and MPLS-VPN, NV introduced the dynamic creation of virtual networks, each with independent address spaces and topologies that coexist over the same physical network.

In cloud computing, self-service consumption, rapid VM provisioning and lifecycle management are subsumed in the NV architecture. The NVE allowed tenants to dynamically create network topologies to support the virtualised applications' or VM's lifecycle state. The virtual switch (vSwitch) within the hypervisor becomes the first network control point at the edge. In an NV model, the vSwitches are programmed by a logically centralised control plane responsible for synchronising the forwarding state across all NVEs. The controller cluster is responsible for the overall state management, including the VM location. In addition, the control plane efficiently disseminates the VM identifiers to locator mappings using an advertisement-based model.

3.4.3 Identity and the IP Namespace

In the context of NVEs, vSwitches are responsible for providing connections over IP tunnels as paths without modifications to the physical network. While there are many variations of overlays [17] [18], the industry has primarily adopted the Virtual eXtensible Local Area Network (VXLAN) [19] encapsulation protocol as the de facto standard in the data centre and cloud. VXLAN embeds tenant-specific identifiers for isolation to ensure packets are differentiated between tenants that cohabit on a common platform.

VXLAN preserves the original IP space as the VM's identity. The encapsulation contains an outer IP space to be used as a dynamic locator known as virtual-tunnel endpoints (VTEPs), which is a routable interface bound to the hypervisor.

In summary, NV adopted an application-first principle, unlike LISP, where the NVE dynamically reacts to the consumption needs of self-service clouds. Rapid VM provisioning and management of lifecycle state requires the control plane to disseminate identifiers and locator mappings to hypervisors on a need-to-know basis. Like LISP, NV created a disjoint IPv4 namespace, separating the role of identifier and locator as independent addresses. Due to the prevalence of internal network-based middleware such as firewalls, the IP address must maintain its role as the core identifier for the VM.

3.5 Identifier Locator Addressing

3.5.1 Container Artefact and the Enterprise Cloud

There has been a great increase in the adoption of delivery applications in the format of containers [20]. It is appealing for cloud-native applications to be delivered in container-based virtualisation [21] due to their efficient resource utilisation and portability between cloud platforms. However, there are two new challenges that containers create for networking. Firstly, every container process requires a single IPv4 address, which leads to exhaustion in CIDR allocation due to the scale of processes compared to virtual machines. Secondly, the container's runtime layer is above the hypervisor, requiring a new networking control point.

3.5.2 Identifier Locator Addressing (ILA) Namespace

To address these challenges, identifier locator addressing (ILA) [22] was proposed. Similar to NV, as described in the previous section, ILA is an overlay network focused on separating core and edge networks. However, ILA differs from NV in two ways. ILA focuses on the layer above server virtualisation, to which NV becomes transparent. In addition, ILA is an IPv6 overlay solution that takes advantage of a larger address space to modify the identifier, thus removing the encapsulation overhead that VXLAN and LISP introduced. Its primary motivation is to support container virtualisation use cases such as scheduling, task migration and container orchestration [23] while preserving container identity.

The fundamental principles of ILA were borrowed from earlier work in the identifier-locator network protocol (ILNP) [24] and GSE (8+8) [25], which also addressed the challenges of overloaded semantics in the IP address. The ILNP approach demonstrated a way of using addressing by creating a distinction between the core routing and end systems.

While ILNP had the design properties to fit the dynamic behaviour of cloud workloads, the im-

plementation required modifications to the TCP stack to perform dynamic changing of locator bindings. ILA addressed this challenge by implementing an IPv6 [26] transformation technique that ensures the transport layer remains unmodified with an immutable 64-bit identifier and SIR prefix. There is an allocation of a /64 prefix per host representing the locator within the ILA scheme. As described in Figure 3.3, when a client/server session is initiated between two containers, the ILA host will perform an outside-of-process stateless IP address transformation on the higher-order 64 bits, the locator, to forward packets to the container's destination. On the receiving side, the ILA host rewrites the destination IP address from the locator prefix to the original identifier address and subsequently delivers the packet to the container application.

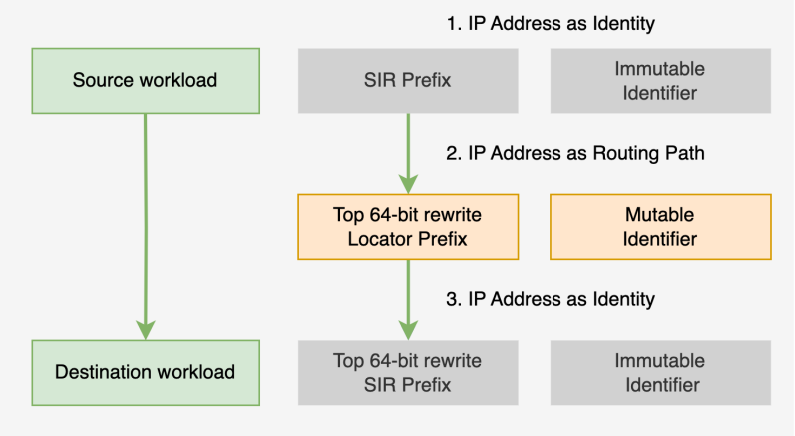


Figure 3.3: IPv6: Identifier and Locator Transformation.

The control plane of this architecture disseminates identifiers and locator mappings to all ILA host routing elements. The ILA control plane is a distributed key-value store that manages mappings of identifiers to locators in a network. As tasks or containers are registered, the mappings are published to the control plane and disseminated to all ILA hosts through its push-based model, which differs from the LISP pull-based model.

In summary, while the ILA solution applies creative techniques through IPv6 transformations, the administrative boundaries of this architecture assume the control of the host stacks to allow an insertion strategy to preserve the original container identifiers and modify the locator instructions within the packet. In addition, it will become increasingly challenging to preserve the end-to-end identity that IPv6 advocates across heterogeneous cloud providers, given that NAT, load balancers and network-based middleware will force identifiers higher in the protocol stack.

3.6 Cloud-Native Service Mesh

3.6.1 Microservices and Service Mesh

Previous sections discussed the networking challenges in cloud computing models that deployed N-tier applications in VM or container artifacts. This section discusses the network requirements for a new generation of cloud-native applications called microservices [27]. A microservice is a collection of networked services, with each service performing a specific business function.

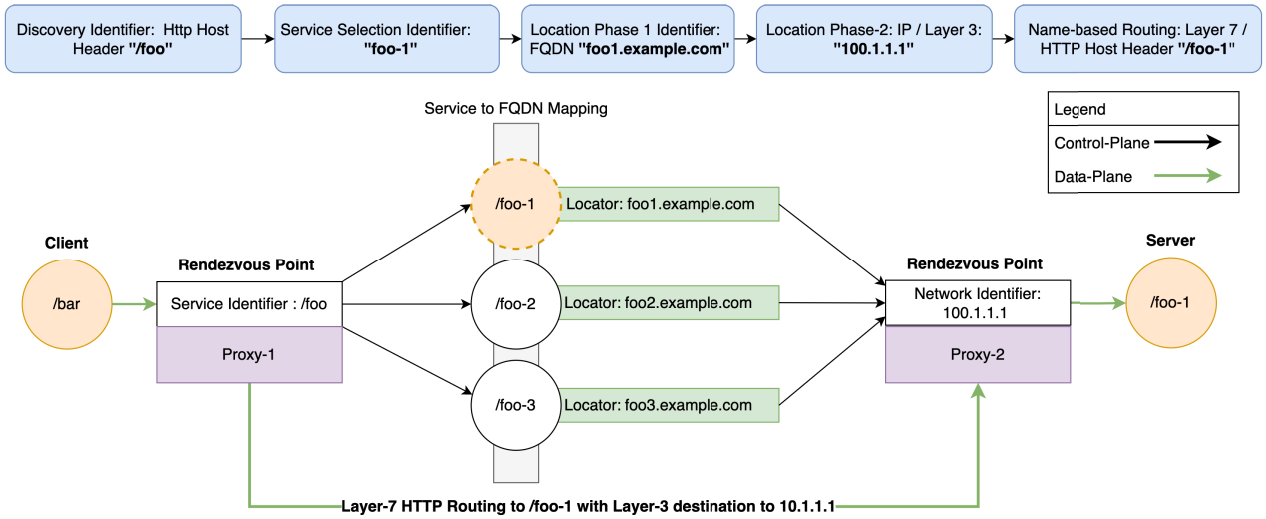


Figure 3.4: Cloud-Native Service Mesh: The Application-level Identifier for Name-based Routing.

Services are built independently from each other and delivered in the format of containers. The value is that developers can alter and upgrade the application without a full redeployment.

A single cloud-native application can be represented by thousands of services that are continually changing states. The probability of communication failure between services increases as cloud-native applications grow exponentially. Application resiliency leads to the need for a dedicated infrastructure layer to handle service-to-service communication. Service mesh [28], [29] addresses this challenge by injecting a proxy service, known as a sidecar, distributed within an application or microservice boundary. The service mesh ensures an optimal application-level response time by handling requests, retries and timeouts, such as circuit breaking, load balancing and traffic engineering. Each application request is routed optimally and is load-balanced without the knowledge of the application. The following sections describe the refinements of identifiers and locators, considering that an application residence is in multiple cloud platforms.

3.6.2 Service Mesh Namespace for Discovery and Routing

The service mesh is responsible for discovering, locating and routing between proxies to establish application connections, as depicted in Figure 3.4. In this example, the client "/bar" service requests to communicate with the server "/foo" service. Firstly, the client sends an HTTP [30] request to query for foo, modifying the host header field to "Host: foo". The service name "/foo" is an anycast identifier [31] mapped to multiple instances or replicas of the service. The service mesh has a selection criterion to map the service identifier to a locator. In this example, the query for "/foo" is based on the proximity and latency between the client and server, and therefore, "foo-1" is matched against the Fully Qualified Domain Name (FQDN) "foo1.example.com". The resolution of this FQDN is IP address "100.1.1.1", which is the proxy rendezvous point. Furthermore, proxy-1 sets up a TCP connection with Proxy-2. Proxy-1 modifies the HTTP host header field to "Host: foo-1" to connect with the foo-1 service. In this example, the HTTP host header is a namespace that shifted the unique identifier from Layer-3 to Layer-7. Having the namespace distinguish services is not limited to HTTP. With HTTP/S [32], the host header is encrypted, and the server name indicator (SNI), which is an extension of the TLS protocol [33], provides the namespace to distinguish services.

These name-based techniques were borrowed from earlier work in virtual-host networking [34], where web servers were consolidated to a common platform, overloading a single IP address with multiple service aliases [35]. A client would replicate the name previously learned from the Domain Name System (DNS) [36] into the HTTP application namespace – e.g., Host header field – to provide an application demuxer beyond the network layers. The HTTP/S protocol became the vehicle for carrying names and has been widely known as a name-based routing approach [37]. Content Delivery Networks (CDNs) also adopted name-based virtual hosting to disentangle the conventional bindings between IP addresses, the applications they represent, and the servers to which they are assigned to [38].

3.6.3 Service Mesh Identity and the URI Namespace

Regarding identity, it is important to note that there are three distinct layers: the platform layer, the host layer and the process/service layer. IP addresses have been the common identifier for modelling host-to-host communication. However, the scenarios where applications span multiple clouds present three challenges to model identity at the host layer:

- Workloads are ephemeral in nature, and the IP address is short-lived as an identifier.
- Workloads are deployed outside traditional administrative boundaries, such as the public cloud, which does not intersect with the internal enterprise network.
- Given that the service mesh proxies TCP connections and the network is riddled with network-based middleware boxes, the IP address becomes a weak and mutable identifier.

The objective of workload identity is to model service-to-service communication without preserving the IP address as the identifier. Applications that span administrative domains and cloud realms require identifiers to be uncoupled from the underlying infrastructure.

For cloud-native applications, the secure production identity framework for everyone (SPIFFE) addresses the workload identity challenge [39] using its naming schema independent of the discovery and routing described in this section. There are three components of SPIFFE: a specification and identifier used as a referral for a service (SPIFFE-ID), a SPIFFE Verifiable Identity Document (SVID) for embedding the SPIFFE-ID that is signed by a trusted authority, and the Workload API, which is an agent running within the cloud that provides a method of obtaining the SVID. The SPIFFE-ID is a uniform resource identifier (URI) [40] that includes the scheme "spiffe://", a trust domain, and a path that specifies the name of the service. For example, the foo service is a billing system whose administrative domain is in the production environment. The identifier can be represented as the following:

spiffe://production.example.com/billing/foo

The SPIFFE ID is codified into an SVID, which is an extension of the x509 certificate [41], signed by an authority inside the trust domain. Within the specification, the SPIFFE ID is set to the URI type in the subject alternative name extension (SAN) [41]. Thus, each workload will have a SPIFFE bundle used for path validation against all the SVIDS that belong to the trust domain. SVIDs allow clients and servers to gain mutual knowledge by asserting their provable cryptographic identity for mutual TLS authentication [42], [43].

As depicted in Figure 3.5, an X509 certificate is issued to services "/bar" and "/foo". Both services are part of the trust domain of "production.example.com" and have a common root

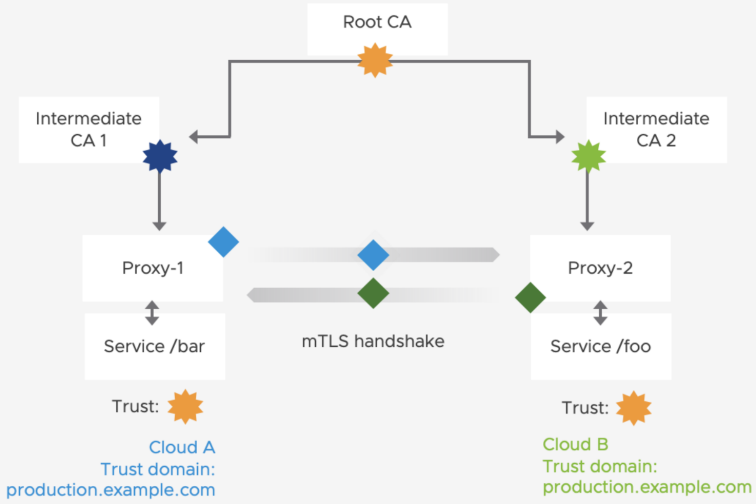


Figure 3.5: PKI: Mutual Authentication within a Trust Domain.

of trust in the PKI hierarchy [44]. Both services will undergo the verification process in this scenario and exchange their x509 certificate as their identity document for mutual TLS authentication. The trust domain within the SVID identifier allows the administrative domain the flexibility to expand beyond its platform and across multiple clouds, where control of identity at the IP level is harder to achieve.

In summary, in the cloud-native service mesh, IP addresses are no longer identifiers for applications, and the distinguishing factors are names. Names do not represent a host, VM or container; they refer to services inside an application. Services are assigned a persistent name as the identifier with a flexible administrative scope spanning multiple cloud environments and organisations. Therefore, it is essential to delineate naming systems for identifiers and locators in the cloud-native service mesh. First, the service name is derived from the HTTP header for the discovery and locating phase, which provides a mapping to a concrete FQDN. The URI within the SVID provides the verification phase for digital identity.

3.7 Comparisons and Future Directions

This chapter has discussed how the iteration of each network architecture addressed a new set of challenges that were non-existent in the previous generation. The chapter includes an observation that the application artifact and cloud consumption model forced changes in identifiers and locators at various layers of the protocol stack. Firstly, this section will summarise how the core vs. edge evolved based on the network control required to meet the cloud-native needs. After that, the chapter discusses future directions and provides topical areas that may play an essential role in future cloud-native networking.

3.7.1 Comparison between Discussed Techniques

Table 3.1 summarises the association between the generation of the application artefact, identity controller, identifier namespace and the network architecture insertion point. LISP and NV were the generation architectures that intersected with server virtualisation. The insertion point in both approaches assumes control of the network platform and, therefore, takes a bottom-up approach to identifiers and locators anchored to the IP layer.

Application Artifact	Identity Controller	Identity Namespace	Distinguished Identifier	Insertion Point
Single-Tier: Virtual Machine	Locator/Identifier Seperation Protocol	Disjoint IPv4	IPv4 Address	Data Center Core Network
Multi-Tier: Virtual Machines	Network Virtualization	Disjoint IPv4	IPv4 Address	Hypervisor Network
Multi-Tier: Containers	Identifier Locator Addressing	Separated IPv6	IPv6 Address	Container OS - Network Namespace
Microservice: Containers	Cloud-Native Service Mesh	HTTP/S Host Header	Service & Canonical Name	Container Sidecar Proxy
Microservice: Hyperscale	SPIFFE	X509	URI	PKI Certificate Issuance

Table 3.1: Supporting Network Architectures, Namespace, Identifiers and Insertion Points.

In contrast to LISP, NV shifted the edge to the hypervisor, projecting an application-first architecture supporting the dynamic characteristics of enterprise cloud computing. ILA adopted similar techniques to NV by adding a layer of indirection with IPv6 to preserve the container identifier. However, the ILA edge model shifted to a layer above NV by inserting control at the container OS network namespace level.

The public cloud created a new challenge that was not present in previous approaches. LISP, NV and ILA are approaches whereby the insertion point can assert identity independent of location, as long as there is network control of the platform. However, as the public cloud infrastructure layers are hidden from consumers, it negates the ability of the network-centric approaches to provide a persistent identity at the IP layer using overlay techniques. A new generation of applications is designed to be less dependent on the underlying platform. The artefact comprises ancillary services known as a sidecar, which provides reachability and resiliency. The cloud-native service mesh addressed this challenge by creating an insertion point as distributed proxies to be the new edge. The abstractions for identity evolved to be an attribute of the application. Therefore, it did not serve the purpose of structuring the IP address plan for identity and location as demonstrated in predecessor architectures. The use of naming as a point of indirection demonstrated that network flows could be distinguished in the context of the application-level session.

SPIFFE was separated into its own category with the cloud-native workload identities framework. Many application instances could represent a service in a plurality of cloud platforms. SPIFFE addresses this challenge by disseminating identifiers in the form of the URI to genuine instances of a service. The x509 document underpins the SPIFFE solution; therefore, cloud-native workloads must have a common root of trust to authenticate mutually. If two services are under different trust hierarchies, then a third-party broker is required to exchange trust bundles between authority domains for mutual authentication.

Based on the observations of the cloud-native network requirements, there are four distinct roles of naming used as identifiers:

1. **Service.** A public-like persona to discover and identify services needed to compose an application.

2. **Locator.** The core or canonicalised name and the locator identifier to resolve the IP

address of the service instance.

3. **Routing.** An application-level identifier replicated in the application protocol namespace to distinguish network sessions over a common IP address target.

4. **Identification.** Verifiable identifier to provide assurance it is a genuine copy or application instance.

Using these identifiers is only possible with the strict governance of naming inside an isolation boundary. Therefore, a namespace is required to ensure the scope of discovery, identifying, and locating workloads is within the trust domain.

3.7.2 Comparisons with the State-of-the-art

In recent years, developers have shifted their focus to constructing and deploying microservices without considering the underlying infrastructure requirements [45]. Serverless computing is an example of this paradigm that allows developers to focus on the business logic of the application without the consideration of provisioning and scaling of the entire system. Serverless functions are seen as event-driven, triggering a temporary execution environment only when required. Amazon Web Services (AWS) Lambda, Microsoft Azure Functions, Google Functions and Cloudflare Workers have introduced serverless capabilities, known as Function-as-a-Service (FaaS) [46].

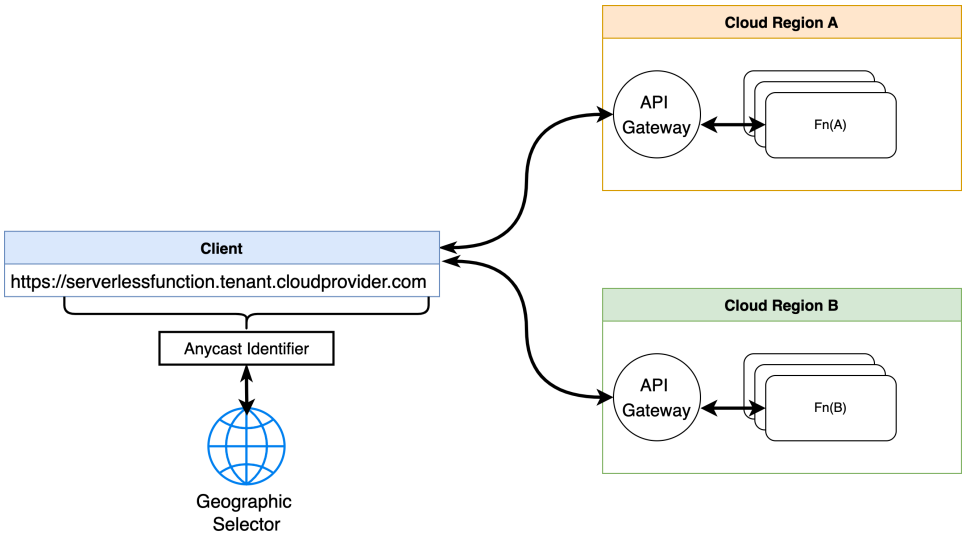


Figure 3.6: Persistent Identifier for Serverless in Hyperscaler Environments.

In the event the serverless function is called, a virtualised workload is provisioned with application files and dependencies required to execute the function and terminates after the operations are performed [47]. In most implementations, containers are used as the primary artefact for their lightweight application process isolation and resource provisioning [48], [49]. Although serverless was originally enabled for cloud environments, it gained traction and also cemented its position in edge computing [50], where computational resources are closer to the data source [51]. However, as the edge environments consist of resource-constrained devices and diverse CPU architectures, WebAssembly (Wasm) [52] has emerged as a promising artefact for future serverless support at the cloud-edge [47], [48], [53], [54].

The serverless model has created new challenges in cloud-native networking. In comparison, cloud-native service mesh provides end-to-end network control through distributed proxies. Each proxy is an ambassador for each service-to-service communication, using the above four identifiers. Unlike service mesh, serverless computing offered by cloud providers does not expose the naming and addressing of the infrastructure platform [55], which thwarts the ability to bind the identifiers and locators for network control.

Irrespective of the serverless artifact type, the cloud providers only expose a publicly accessible Universal Resource Locator (URL) [56] identifier, which is used in the HTTP protocol as an access mechanism to a serverless function. The URL namespace describes the path instructions to a location of the available service. For clarity, serverless applications do not reside in a single location but are replicated globally; therefore, the same URL is also required to access multiple instantiations or replicas.

In Figure 3.6, the domain name portion of the URL is referred to as an anycast identifier [31], which is persistent and resolves to multiple locators or IP addresses. The domain name to IP address translation is coordinated by the cloud providers' DNS-based global traffic management system (GTM) [57]. The DNS-based GTM acts as a geographic site-selector, directing client connections to the closest serverless function [58]. This enables the URL to remain immutable, allowing the DNS to be the identity-like and public persona, returning customised answers based on the information from the client queries [59], [60].

3.7.3 Future Directions and Investigations

A view of future directions using Internet identifiers to support cloud-native services is emerging. Globally decentralised cloud-native applications rely on URLs to mimic and behave like URIs to refer to the service independent of its location. This suggests three investigative areas to provide identifiers and the underlying infrastructure support that will play an essential role in cloud-native networks.

Persistent Identifier. In the digital library space, the Digital Object Identifier (DOI) [61] is a persistent identifier that refers to a book or document residing in multiple locations at the same time. Analogous to a library, a cloud-native application also contains instances simultaneously running in multiple environments and locations. However, the complexity is that cloud-native applications today require four separate identifiers to be referred to and verified as the given instance. Cloud-native architectures must adopt similar principles from the digital libraries; all application service instances should be referred to as a single identifier. What an application is should be separated from where it is.

Furthermore, investigation is needed to provide referential consistency that anchors application instances to a common identification scheme. This would comprise a directory service whereby the identifier could be passed between parties to provide locator mappings to cloud resources. In addition, the identifier is decorated with additional metadata such that the identifier would remain persistent. Unlike copies of books, applications evolve; therefore, providing additional metadata to adorn the query is a possible technique to obviate the need to create a new identifier.

Identity Rendezvous. The identity rendezvous provides two functions. The first is identity attestation to analyse credentials from all workloads to assure they are qualified application replicas or instances, and the second is connecting and mutually authenticating source and

destination service. This approach is only possible when there is control over the workload deployment pattern, e.g., the container workload and its ancillary service. When microservices are also constructed with serverless functions, they expose new identity challenges that require investigation into two areas. Firstly, as described in this section, cloud providers hide the interconnections of a serverless environment. It is possible to provide identity rendezvous as a sidecar or proxy as long as it is outside the providers' realm. The challenge it may create is additional network latency costs, which are worth exploring. Secondly, the execution of a serverless function has a temporary runtime environment, which creates performance vs. security pressure to minimise application startup delays. In the context of the SPIFFE example, the framework assumes control of the underlying operating system of where the application is running. Since the serverless environment does not provide access, it is unclear how the attestation process will be completed to ensure the application is a genuine instance and the true owner of the identifier it passes before mutual authentication. This usecase is not limited to serverless but also includes future applications where only the API is exposed without access to the underlying infrastructure.

Dissemination of Identities. To construct complex applications between network and cloud realms, isolation boundaries can span multiple clouds and be created on demand. This ensures that application instances within an isolation boundary are uniquely rooted in a common trust model. Suppose the industry wishes to continue using a SPIFFE-like model, where the certificate chain of trust underpins the identity framework. In that case, a challenge will arise when applications are not within a common isolation boundary. While federation techniques are well understood in the industry, with various approaches to exchanging bundles [62], the architecture lacks a third-party identity broker to manage and distribute trust bundles for general use and on a need-to-know basis.

3.8 Conclusions

This chapter has highlighted how each generation of network architectures addressed a challenge created by evolving application artefacts. It highlights that cloud-native applications have shifted identifiers from the IP layer to become an attribute of the application layer used to distinguish workloads and provide location-independent identification schemes. In addition, this chapter has reviewed the current state-of-the-art application deployment patterns and the future identifier challenges that may arise. This is an important topic that requires further investigation under three topical areas. The first is evaluating existing persistent identifiers in the digital space that could be applied for broader use. Regardless of location, application instantiations should be referred to as the single identifier. Secondly, the verification process, where credentials are analysed to provide identity attestation and issuance of its imprimatur, should be consistent across any cloud and edge computing platform. Lastly, the delivery method to control and disseminate trust anchors to allow application communication between trust realms will underpin the success of a common identity system. It is hoped this retrospective of identifiers may influence the future design of identifiers and locators used in cloud-native architectures.

References for Chapter 3

[1] G. Huston and C. S. APNIC, "In defence of nats," *Internet Protoc. J*, vol. 20, no. 3, pp. 25–33, 2017.

[2] D. Meyer, L. Zhang, and K. Fall, "Report from the iab workshop on routing and addressing," RFC Editor, RFC 4984, Sep. 2007, `http://www.rfc-editor.org/rfc/rfc4984.txt`. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc4984.txt`.

[3] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (bgp-4)," RFC Editor, RFC 4271, Jan. 2006, `http://www.rfc-editor.org/rfc/rfc4271.txt`. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc4271.txt`.

[4] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "The locator/id separation protocol (lisp)," RFC Editor, RFC 6830, Jan. 2013, `http://www.rfc-editor.org/rfc/rfc6830.txt`. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc6830.txt`.

[5] P. Nikander, A. Gurtov, and T. R. Henderson, "Host identity protocol (hip): Connectivity, mobility, multi-homing, security, and privacy over ipv4 and ipv6 networks," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 2, pp. 186–204, 2010.

[6] C. E. Perkins, "Mobile ip," *IEEE communications Magazine*, vol. 35, no. 5, pp. 84–99, 1997.

[7] V. Ishakian, J. Akinwumi, F. Esposito, and I. Matta, "On supporting mobility and multihoming in recursive internet architectures," *Computer Communications*, vol. 35, no. 13, pp. 1561–1573, 2012.

[8] K. B. Egevang and P. Francis, "The ip network address translator (nat)," RFC Editor, RFC 1631, May 1994, `http://www.rfc-editor.org/rfc/rfc1631.txt`. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc1631.txt`.

[9] M. F. Bari, R. Boutaba, R. Esteves, *et al.*, "Data center network virtualization: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 909–928, 2012.

[10] B. Walters, "Vmware virtual platform," *Linux journal*, vol. 1999, no. 63es, 6–es, 1999.

[11] D. Jen, M. Meisel, H. Yan, *et al.*, "Towards a new internet routing architecture: Arguments for separating edges from transit core.," in *HotNets*, 2008, pp. 103–108.

[12] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," RFC Editor, RFC 3031, Jan. 2001, `http://www.rfc-editor.org/rfc/rfc3031.txt`. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc3031.txt`.

[13] E. Rosen and Y. Rekhter, "Bgp/mpls ip virtual private networks (vpns)," RFC Editor, RFC 4364, Feb. 2006.

[14] T. Koponen, K. Amidon, P. Balland, *et al.*, "Network virtualization in multi-tenant datacenters," in *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, 2014, pp. 203–216.

[15] B. Pfaff, J. Pettit, T. Koponen, *et al.*, "The design and implementation of open vswitch," in *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, 2015, pp. 117–130.

[16] M. Lasserre, F. Balus, T. Morin, N. Bitar, and Y. Rekhter, "Framework for data center (dc) network virtualization," RFC Editor, RFC 7365, Oct. 2014.

[17] B. Davie and J. Gross, "A stateless transport tunneling protocol for network virtualization (stt)," IETF Secretariat, Internet-Draft draft-davie-stt-08, Apr. 2016.

[18] J. Gross, I. Ganga, and T. Sridhar, "Geneve: Generic network virtualization encapsulation," RFC Editor, RFC 8926, Nov. 2020.

[19] M. Mahalingam, D. Dutt, K. Duda, *et al.*, "Virtual extensible local area network (vxlan): A framework for overlaying virtualized layer 2 networks over layer 3 networks," RFC

Editor, RFC 7348, Aug. 2014, http://www.rfc-editor.org/rfc/rfc7348.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7348.txt.

[20] G. Banga, P. Druschel, and J. C. Mogul, "Resource containers: A new facility for resource management in server systems," in *OSDI*, vol. 99, 1999, pp. 45–58.

[21] A. Tosatto, P. Ruiu, and A. Attanasio, "Container-based orchestration in cloud: State of the art and challenges," in *2015 Ninth international conference on complex, intelligent, and software intensive systems*, IEEE, 2015, pp. 70–75.

[22] T. Herbert and P. Lapukhov, "Identifier-locator addressing for ipv6," IETF Secretariat, Internet-Draft draft-herbert-intarea-ila-01, Mar. 2018.

[23] A. Khan, "Key characteristics of a container orchestration platform to enable a modern application," *IEEE cloud Computing*, vol. 4, no. 5, pp. 42–48, 2017.

[24] R. Atkinson and S. Bhatti, "Identifier-locator network protocol (ilnp) architectural description," RFC Editor, RFC 6740, Nov. 2012.

[25] M. D. O'Dell, "Gse - an alternate addressing architecture for ipv6," IETF Secretariat, Internet-Draft draft-ietf-ipngwg-gseaddr-00, Feb. 1997.

[26] S. E. Deering and R. M. Hinden, "Internet protocol, version 6 (ipv6) specification," RFC Editor, RFC 2460, Dec. 1998, http://www.rfc-editor.org/rfc/rfc2460.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2460.txt.

[27] N. Dragoni, S. Giallorenzo, A. L. Lafuente, *et al.*, "Microservices: Yesterday, today, and tomorrow," *Present and ulterior software engineering*, pp. 195–216, 2017.

[28] W. Li, Y. Lemieux, J. Gao, Z. Zhao, and Y. Han, "Service mesh: Challenges, state of the art, and future research opportunities," in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, IEEE, 2019, pp. 122–1225.

[29] A. E. Malki and U. Zdun, "Guiding architectural decision making on service mesh based microservice architectures," in *European Conference on Software Architecture*, Springer, 2019, pp. 3–19.

[30] R. T. Fielding, J. Gettys, J. C. Mogul, *et al.*, "Hypertext transfer protocol – http/1.1," RFC Editor, RFC 2616, Jun. 1999, http://www.rfc-editor.org/rfc/rfc2616.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2616.txt.

[31] B. Carpenter and S. Brim, "Middleboxes: Taxonomy and issues," RFC Editor, RFC 3234, Feb. 2002, http://www.rfc-editor.org/rfc/rfc3234.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3234.txt.

[32] E. Rescorla, "Http over tls," RFC Editor, RFC 2818, May 2000, http://www.rfc-editor.org/rfc/rfc2818.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2818.txt.

[33] D. Eastlake, "Transport layer security (tls) extensions: Extension definitions," RFC Editor, RFC 6066, Jan. 2011, http://www.rfc-editor.org/rfc/rfc6066.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6066.txt.

[34] C. Karayiannis, "The apache web server," in *Web-Based Projects that Rock the Class*, Springer, 2019, pp. 1–37.

[35] E. Lear and R. Droms, "What's in a name:thoughts from the nsrg," IETF Secretariat, Internet-Draft draft-irtf-nsrg-report-10, Sep. 2003.

[36] P. Mockapetris and K. J. Dunlap, "Development of the domain name system," in *Symposium proceedings on Communications architectures and protocols*, 1988, pp. 123–133.

[37] T. Koponen, M. Chawla, B.-G. Chun, *et al.*, "A data-oriented (and beyond) network architecture," in *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, 2007, pp. 181–192.

[38] M. Fayed, L. Bauer, V. Giotsas, *et al.*, "The ties that un-bind: Decoupling ip from web services and sockets for robust addressing agility at cdn-scale," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 433–446.

[39] "Secure production identity framework for everyone." (), [Online]. Available: `https://www.spiffe.io` (visited on 05/08/2021).

[40] T. Berners-Lee, R. T. Fielding, and L. Masinter, "Uniform resource identifiers (uri): Generic syntax," RFC Editor, RFC 2396, Aug. 1998, `http://www.rfc-editor.org/rfc/rfc2396.txt`. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc2396.txt`.

[41] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile," RFC Editor, RFC 5280, May 2008, `http://www.rfc-editor.org/rfc/rfc5280.txt`. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc5280.txt`.

[42] T. Dierks and C. Allen, "The tls protocol version 1.0," RFC Editor, RFC 2246, Jan. 1999, `http://www.rfc-editor.org/rfc/rfc2246.txt`. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc2246.txt`.

[43] A. Goel and B. Thangaraju, "Authenticating distributed systems using spire over kubernetes cluster," in *2022 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, IEEE, 2022, pp. 1–6.

[44] S. Chokhani, W. Ford, R. Sabett, C. Merrill, and S. Wu, "Internet x.509 public key infrastructure certificate policy and certification practices framework," RFC Editor, RFC 3647, Nov. 2003.

[45] M. Roberts and J. Chapin, *What is Serverless?* O'Reilly Media, Incorporated, 2017.

[46] P. Castro, V. Ishakian, V. Muthusamy, and A. Slominski, "The rise of serverless computing," *Communications of the ACM*, vol. 62, no. 12, pp. 44–54, 2019.

[47] A. Hall and U. Ramachandran, "An execution model for serverless functions at the edge," in *Proceedings of the International Conference on Internet of Things Design and Implementation*, 2019, pp. 225–236.

[48] P. K. Gadepalli, G. Peach, L. Cherkasova, R. Aitken, and G. Parmer, "Challenges and opportunities for efficient serverless computing at the edge," in *2019 38th Symposium on Reliable Distributed Systems (SRDS)*, IEEE, 2019, pp. 261–2615.

[49] E. Jonas, J. Schleier-Smith, V. Sreekanti, *et al.*, "Cloud programming simplified: A berkeley view on serverless computing," *arXiv preprint arXiv:1902.03383*, 2019.

[50] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[51] M. S. Aslanpour, A. N. Toosi, C. Cicconetti, *et al.*, "Serverless edge computing: Vision and challenges," in *2021 Australasian Computer Science Week Multiconference*, 2021, pp. 1–10.

[52] A. Rossberg, B. L. Titzer, A. Haas, *et al.*, "Bringing the web up to speed with webassembly," *Communications of the ACM*, vol. 61, no. 12, pp. 107–115, 2018.

[53] P. K. Gadepalli, S. McBride, G. Peach, L. Cherkasova, and G. Parmer, "Sledge: A serverless-first, light-weight wasm runtime for the edge," in *Proceedings of the 21st International Middleware Conference*, 2020, pp. 265–279.

[54] J. Ménétrey, M. Pasin, P. Felber, and V. Schiavoni, "Webassembly as a common layer for the cloud-edge continuum," in *Proceedings of the 2nd Workshop on Flexible Resource and Application Management on the Edge*, 2022, pp. 3–8.

[55] K. Kaffes, N. J. Yadwadkar, and C. Kozyrakis, "Centralized core-granular scheduling for serverless functions," in *Proceedings of the ACM symposium on cloud computing*, 2019, pp. 158–164.

[56] T. Berners-Lee, L. Masinter, and M. McCahill, "Uniform resource locators (url)," RFC Editor, RFC 1738, Dec. 1994, `http://www.rfc-editor.org/rfc/rfc1738.txt`. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc1738.txt`.

[57] C. Huang, D. A. Maltz, J. Li, and A. Greenberg, "Public dns system and global traffic management," in *2011 Proceedings IEEE INFOCOM*, IEEE, 2011, pp. 2615–2623.

[58] K. Schomp and R. Al-Dalky, "Partitioning the internet using anycast catchments," *ACM SIGCOMM Computer Communication Review*, vol. 50, no. 4, pp. 3–9, 2020.

[59] C. Contavalli, W. van der Gaast, D. Lawrence, and W. Kumari, "Client subnet in dns queries," RFC Editor, RFC 7871, May 2016.

[60] K. Schomp, O. Bhardwaj, E. Kurdoglu, M. Muhaimen, and R. K. Sitaraman, "Akamai dns: Providing authoritative answers to the world's queries," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 465–478.

[61] N. Paskin, "Digital object identifier (doi®) system," *Encyclopedia of library and information sciences*, vol. 3, pp. 1586–1592, 2010.

[62] M. S. L. d. SILVA *et al.*, *Integrating SPIFFE and SCONE to enable universal identity support for confidential workloads.* Universidade Federal de Campina Grande, 2021.

Part II

Evidence and Validation of the Proposition

Chapter 4

Research Proposition

4.1 Background to the Proposition

The previous chapter examined the evolving role of the DNS, highlighting the increasing significance of prioritising unique names over unique addresses to distinguish distinct resource instances. Currently, the DNS encompasses a broader role in service discovery, containing additional data associated with the lookup key or name, such as a connection profile. This data provides all the necessary information to establish a connection, thereby minimising the need for multiple round trips to improve latency. These initial observations from the topical review hint that an evolution is occurring, indicating that the DNS system's complexity extends beyond these immediate enhancements.

It is assumed that no technology is immutable, and as market needs exert pressures for identifier systems to evolve, systems are increasingly at risk of being replaced, or multiple instantiations of various identifier systems may emerge to address different challenges. DNS is no exception. It has experienced, and will continue to experience, a range of pressures for change. Therefore, it is essential to continue examining these pressures and understanding their expression.

Originally, DNS was modelled similarly to a telephone directory, and, as with a directory published annually, infrequent publishing negated the need for customisation. It mattered little who opened the telephone book; the data was there for anyone to look up and retrieve the same answer. These properties were highly valued in a digital identifier system, where users can pass a key or name between parties, referring to things by this key and being assured that it is referencing the same object. However, as the Internet has evolved, so too have the requirements of what is needed in an identity system. Continuous examination of which parts need to be altered and the implications of those alterations is central to the dialogue of this research.

The DNS system, no longer viewed as a static lookup and now with the understanding that identifiers are not constant indices in an invariant system, illustrates that it is not a matter of one answer for everyone; the optimal answer is based on the attributes of the questioner, meaning the answer is relative to who is asking.

There are two paths for research in this field of study. The first is a revolutionary path, the development of a new or alternative system that directly confronts the incumbent with a strategy so radical it could potentially create fragmentation of the Internet, forcing participants into Internet A or Internet B.

This research advocates for an evolutionary approach over a revolutionary one, underpinned by the belief that even incremental improvements in areas such as flexibility, speed and privacy hold the potential to challenge the current status quo. While the literature presents several well-documented alternative approaches to novel naming systems, which is the revolutionary path, the discussion of alternative behaviours lacks depth, both in thought and documentation. There are very few literature examples to draw from, indicating a gap in the scholarly exploration of this topic.

4.1.1 Givens:

1. **Underexplored DNS Literature:** The existing literature on the flexibility within DNS resolution systems and namespaces is limited, with several underexplored areas. For instance, the dynamics between the client and publisher, especially in contexts where the publisher is a public Internet resource but the client's access attempts are intended to be confidential, have not been extensively discussed. Moreover, topics such as the temporal nature of domain names, their persistence over time, and the systems' capability to customise responses based on specific names are areas that existing research barely addresses.
2. **DNS Constraints:** The DNS as an identifier system in its current state is constrained within a framework of five key parameters: speed, currency, scalability, flexibility and privacy. Prioritising one or more of these parameters in this framework inevitably shifts the technological burden elsewhere, introducing technological trade-offs.
3. **Network Coherence:** The Internet serves as a public communications system; it is imperative that the DNS namespace is structured to ensure the network maintains its coherence and utility as a public service.

4.1.2 We propose the following:

We propose a computational DNS approach aimed at enhancing system flexibility, navigating the balance between customisability and maintaining privacy, a dynamic referred to as the *flexibility* $\leftrightarrow$ *privacy tension*. This approach proposes to optimise flexibility and privacy without compromising on speed, illustrating our strategy of reconciling the competing objectives of an adaptable identifier resolution system.

4.1.3 We will give weight to the proposition as follows:

To substantiate this proposition, a mixed-methods approach incorporating both qualitative and quantitative methodologies will be employed. The following outlines the contributions.

Chapter 5A – Systematic Literature Review as Research Methodology:

Given the limited literature in this domain and the multidisciplinary nature of Internet identifiers, a systematic literature review is essential. This chapter synthesises a variety of data sources, including publications from W3C, RFC, IEEE, ACM and government documents from libraries that address similar challenges. The aim is to achieve an improved understanding of the various categories of identifiers and examine how resolution systems and namespaces are designed. This study will explore the myriad identifier systems beyond the DNS, each scoped for specific purposes, exploring their functional purpose. In addition, the research contribution

offers insights into non-obvious literature examples and provides a perspective on the potential evolutionary directions an identifier system, such as the DNS, could embark upon.

Chapter 5B – Proof by Argument on 5A Findings:

Building on the literature review, Chapter 5B advances the qualitative inquiry by presenting a proof by argument for the integration of alternative behaviours within DNS. This is achieved through an examination of various DNS extensions that address key operational parameters: flexibility, speed, currency and privacy. These examples demonstrate the DNS's capacity as a computational resource, providing qualitative evidence of published techniques.

Chapter 6A – Categorisation-Based Review:

Chapter 6A further explores the DNS landscape by conducting a categorisation-based review, similar to a taxonomy, to understand the constraints of the current DNS system. This review offers insights into the rationale and motivations behind each DNS design decision, highlighting the system's operation within constraints defined by five parameters. This chapter serves as a critical precursor to Chapter 6B, establishing an understanding of existing limitations prior to introducing the proposed model in Chapter 6B.

Chapter 6B – Implementation and Experimentation of the Proposed DNS Model:

Chapter 6B outlines a method to address real-world challenges, leveraging global DNS resolver data to demonstrate the advantages of prefetching DNS responses and precomputing URL strings. It presents the proposed model for enhancing latency and privacy by prefetching DNS answers and precomputing URLs. This method circumvents the privacy issues associated with direct user queries and improves resolution times. Additionally, this chapter details how this approach fits within the existing Internet architecture and describes the deployment methodology that emulates global network conditions. The solution will be implemented as a module within a search engine application, where DNS prefetching and precomputing of the URL occurs. This search engine answer page will display content and URLs that have been precomputed and ready for the user to select.

The validation of this proposition involves testing the precomputed DNS model against traditional DNS usage. This process, detailed in Chapter 6B, discusses the data collection techniques to utilise network and applications tools for measuring end-to-end latency, including the search engine query, search engine answer page and click to the resource. This will provide quantitative evidence of the model's effectiveness.

4.1.4 Future work as evidence:

Chapters 7A and 7B, while categorised as future work, contribute to lending weight to the research position. Chapter 7A outlines the contributions of identifier systems for 5G core network signalling and offers insights and recommendations to the 3GPP rev 16. It highlights the functionality of identifier systems required in future service discovery systems. This work has been published in IEEE Access as referenced in Appendix D. In addition, Chapter 7B is a practical application of context-aware DNS leveraging computational aspects described in the literature study and proof by argument. This work is recognised as an issued patent, attached in Appendix E.

4.1.5 We will conclude as follows:

The analysis presented in Chapters 5A and 5B contributes to setting up the continued examination into Chapters 6A and 6B. This research provides a theoretical exploration of the practical application and validation of the proposed DNS model. Outlined in Figure 4.1, this research approach connects each chapter, leading to a validation of suggested evolutionary changes in the DNS system and potential responses.

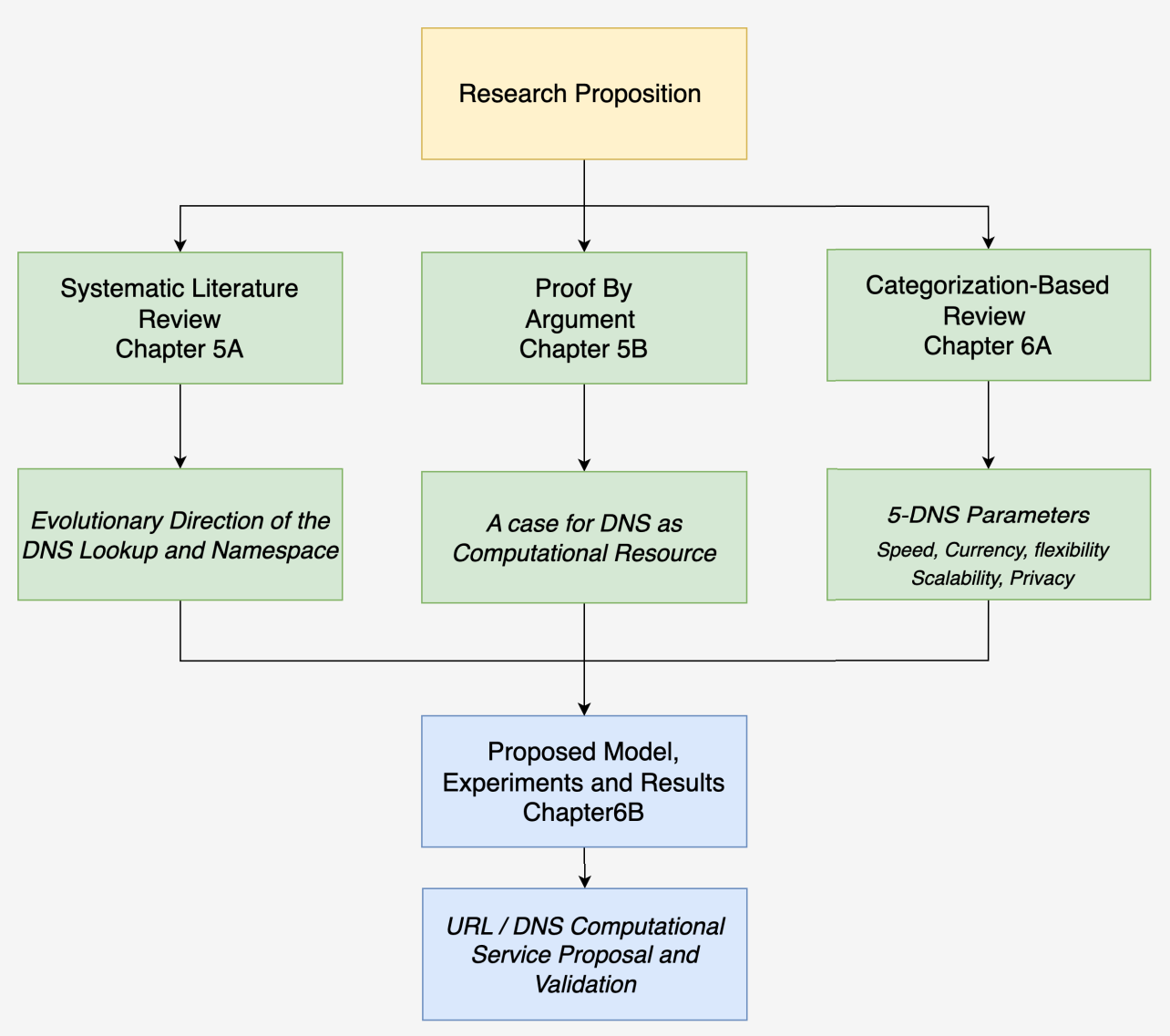


Figure 4.1: Navigating the Research Proposition.

Chapter 5A

Systematic Literature Review: Evolution of Internet Identifiers

5A.1 Purpose of this Chapter

This chapter supports the research proposition's initial contributions by establishing a set of research questions the literature review aims to answer. Furthermore, the chapter synthesises a diverse field of study to provide a multi-disciplinary view of Internet identifiers and their supporting systems. This analysis resulted in a publication titled *"Internet Identifiers: A Survey of History, Challenges and Future Perspectives"*, which was accepted and published in IEEE Access (found in Appendix C). The aim of this work was to identify gaps in the literature and provide possible directions for the field of study. Section 5B expands the systematic literature investigation by outlining a proof-by-argument of alternative behaviours necessary for the DNS resolution system and namespace. These contributions establish the foundation for answering the research proposition and offer a clear direction for the DNS and its proposed model outlined in Chapters 6A and 6B.

5A.2 Introduction

Analogous to a common language, the Internet relies on a common set of symbols for communication efficacy. Identifier systems serve as the backbone of the digital environment, facilitating interactions between diverse entities within the digital realm. The continuous adaptation of these identifiers reflects the need for efficient and flexible techniques to identify and interact with resources, services and users.

This study explores the evolution of Internet identifiers through a comprehensive review of the relevant literature. It includes a historical overview, analysis of existing studies and discussions of current challenges and prospects in this field. Given the multi-disciplinary nature of the topic, this chapter draws perspectives from a range of authoritative sources. These include the World Wide Web Consortium (W3C), Internet Architecture Board (IAB) Workshops, Requests for Comments (RFCs) and academic literature. Additionally, it incorporates significant contributions from entities involved in digital library systems, such as the Library of Congress, and the development of key registry systems to support the Digital Object Identifiers (DOIs).

The aim is not to produce an exhaustive list of identifiers but, rather, to focus on those critical to

the Internet's infrastructure and influential in digital handle systems. The aim is to synthesise a broad range of literature concerning digital identifiers in the Internet namespace and consider potential future directions influenced by evolutionary pressures. This synthesis seeks to enhance the understanding of the Internet's developmental trajectory.

The exploration begins by examining the fundamental roles of namespaces in identifying, locating and classifying entities. Moreover, the unique considerations of the Internet are given attention, primarily the importance of maintaining coherence in the public namespace. Such coherence hinges on a consistent symbol interpretation, universal reachability and referential integrity. Subsequently, the critical dimensions of identity and location within the Internet context are examined, exploring how their significance has been influenced by advances such as mobility and digital cloning.

The narrative progresses to trace and emphasise the roles played by Universal Resource Identifiers (URIs), Universal Resource Locators (URLs) and Universal Resource Names (URNs) within the realm of identifiers and locators. To transcend the boundaries of networked systems, this study's exploration extends to identifier systems utilised in broader contexts, such as library systems. These include the International Standard Book Number (ISBN) used in publishing and Digital Object Identifiers (DOIs) for digital resources. The chapter then juxtaposes these identity models with those at the Internet's heart, such as the URI versus URL and the IP address namespace.

Subsequently, the study examines the thematic topic of search versus identification, exploring the evolution of directory lookup systems. This includes a chronological analysis, commencing with the earlier network namespaces, such as HOSTS.TXT and X.500. Examining the X.500 Directory Naming Framework is important to compare its directory system approach with the Domain Name System (DNS), particularly the system differences between searchability and uniqueness of resolving names. The study transitions to the emergence and importance of the DNS. The chapter draws insights from the DNS predecessors to highlight its evolutionary journey. The successive generation of new resource records highlights this evolution, enhancing the system's flexibility to address emerging challenges.

Furthermore, the study analyses the shift of DNS from a static lookup system to a dynamic, computation-oriented resource. In addition, this study further examines the evolutionary pressures that catalysed the creation of other namespaces that operate outside the global scope of the DNS. These alternative namespaces, established for specific purposes, signify another facet of the Internet evolutionary process and highlight the necessity for a cohesive approach across varied namespaces to mitigate the risk of fragmentation.

In conclusion, this study offers insights into the progression of identifier systems in the Internet context, emphasising the critical role of DNS in supporting Internet identifiers. It also highlights the ongoing transformations driven by evolving needs and technological advances. This discussion clarifies the complex interplay between identifier systems, the necessity of namespace cohesion and the adaptation to evolutionary pressures.

5A.3 Systematic Literature Review Planning

This section aims to demonstrate the strategy adopted to perform the SLR. This approach seeks to define the method of how the review was conducted, including the research questions and how the sources are categorised for this study.

5A.3.1 Research Questions

As the literature covering Internet identifiers and their resolution is multi-disciplinary in nature, we focus on research questions that aim to categorise the role and class of identifiers respective to the digital objects to which they refer. In addition, the research aims to answer the system's considerations in a broader Internet context, such as the purpose of namespaces that exist within and outside of the Internet as well as the directions the Internet naming system may require to continue operating as a cohesive unified namespace.

1. **How have the roles of identifiers allocated from namespaces been conflated in digital systems?**
   This question seeks to explore the evolution of namespaces and how their role and function have become conflated in both Internet and digital library systems.

2. **What are the important principles of referential integrity and semantic consistency in Internet namespaces, and what factors could introduce namespace fragmentation?**
   This question aims to capture the foundational principles that ensure network coherence through a unified namespace to ensure consistent symbol interpretation and risks that could introduce namespace fragmentation.

3. **How have mobility and digital cloning influenced the concepts of identity and location, and what specific challenges do they introduce?**
   This question examines the impact of mobility and cloning on identity and location. This question seeks to explore identity and location and how the burden may shift between network systems to support the goal of the identifier.

4. **How do digital identifiers and locators in Internet and library systems compare, and how can they be categorised?**
   This question focuses on the similar challenges in Internet and library systems, seeking to understand their categorisation and functional disparities.

5. **How does the X.500 Directory System differ from the Domain Name System?**
   This question investigates the distinction between search versus uniqueness between the X.500 system and the DNS. The question will explore the chronology of these systems and the evolution of the DNS.

6. **How has the DNS lookup system and namespace evolved to support Internet identifiers and user queries?**
   This question explores the successful generations of the DNS lookup system to accommodate the growing needs of the Internet. The question seeks to extract how the DNS system should be perceived, i.e., dictionary, directory and computational service.

7. **What potential evolutionary paths could DNS follow, and what alternative namespaces exist outside DNS, including their purposes?**
   This question speculates the future direction the DNS could take in response to the evolutionary pressures of today's Internet. The question also explores the existence and roles of alternative namespaces beyond the Internet and their motivations and purposes.

Category	Keywords	Swim lane Search String
Internet Identifiers	URL, URN, URI, IP Address	("URL" AND "URN" AND "URI" OR "IP Address") AND '**Category**'
Internet Naming Systems	DNS, namespace, resource records, X.500, HOSTS.TXT, identifiers	("DNS" AND "namespace" AND "resource records" AND ("x.500" OR "HOSTS.txt")) AND '**Category**'
Digital Object Architecture	Persistent Identifiers, Durable Identifiers, Handle System, DOI, ISBN	(("persistent identifiers" OR "durable identifiers") AND ("DOI" OR "ISBN")) AND '**Category**'
Identity and Location	Digital Identity, Digital Identifier, Identity Location Separation	("Digital Identity" OR "Digital Identifier" OR "Identity Location Separation") AND '**Category**'
Alternate Namespace	.local, .onion, .gnu, .eth, name systems	(".local" OR ".onion" OR ".gnu" OR ".eth") AND 'Name Systems and **Category**'

Table 5A.1: Search Strategy and Search String.

5A.3.2 Source Selection

Investigating Internet identifiers requires diverse sources to address the research questions thoroughly. In addition to Internet systems research, the research explored digital identifiers used in library systems. This research into journals and government papers that describe metadata archival approaches for digital preservation and the importance of persistent, durable identities is required. A categorisation-based approach was used to create a selection criterion for the following literature.

- **RFC specifications:** Requests for Comments (RFCs) are publications that contain technical information about the specifications and standards for implementing protocols on the Internet.
- **Academic and Conference Papers:** Publications either presented at conferences or peer-reviewed on specific studies relevant to Internet identifiers are included in this category. Topics vary from theoretical discussions to systems that are implemented in the field.
- **Technical Reports and White Papers:** Technical reports include documents from research institutions with in-depth findings of specific research projects. In addition, White Papers were included in this category for specific solution guides that are not necessarily found in academic databases.
- **Internet Drafts:** Documents from the Internet Engineering Task Force (IETF) serve as a reference for discussing evolving standards in relation to the field of study.
- **Preprints and Online Articles:** Sources from reputable websites that published technical insights and preprints of current publications were considered in this category.
- **Books and Book Chapters:** Specific book chapters that provide in-depth coverage of

particular topics were considered due to the fundamental nature of the topic of naming system research.

Keywords were assigned per category to construct a search string. Boolean logic (AND, OR) combined keywords to enhance the accuracy of the search results. A **category** within this model determines the relevant technology scope, which maps the keywords. As per Table 5A.1, the search utilised a composite swim lane model in which the search string contains the keywords within that given swim lane. The category at the end of each search string allows keywords to be added from a different swim lane for cross-disciplinary searches.

5A.3.3 Study Selection and Evaluation Process

The study selection and screening process aims to develop a series of gates to filter literature based on its potential to address aspects of the research questions and offer insights into the field of study.

Stage 1: Review Titles and Abstracts

- Objective: Filter out irrelevant studies.
- Inclusion: Relevance to Internet identifiers, namespaces and selected keywords as per Table 5A.1.
- Exclusion: Literature that does not address aspects of the research questions.

Stage 2: In-depth Screening of Selected Articles

- Objective: Evaluate selected literature background and motivation to determine the relevance of research questions.
- Inclusion: Addresses part or whole of the research question.
- Exclusion: Literature does not partly or fully answer the research questions.

Stage 3: Assessment and Relevance

- Objective: Assess selected studies for completeness in the topic.
- Inclusion: Direct contribution in the field of study through synthesis of the topics covered.
- Scoring: Adopt a scoring system based on the following quality criteria to determine the papers' inclusion to be used in this study:
  - QC1: Are the problem statement, motivation and research objectives clear?
  - QC2: Is the proposed model's architecture and machinery detailed?
  - QC3: Are the arguments justifying the approach of the proposed model sound, and do they contain a related work section?

For simplicity, each criterion has two possible responses: Yes or No. "Yes" results in (1) one point, and "No" results in (0) zero points. As described in the following equation, a score greater than or equal to two (2) is required for literature acceptance.

$$\text{Accepted if } (\text{QC1} + \text{QC2} + \text{QC3}) \geq 2$$

5A.3.4 Search Screen Results

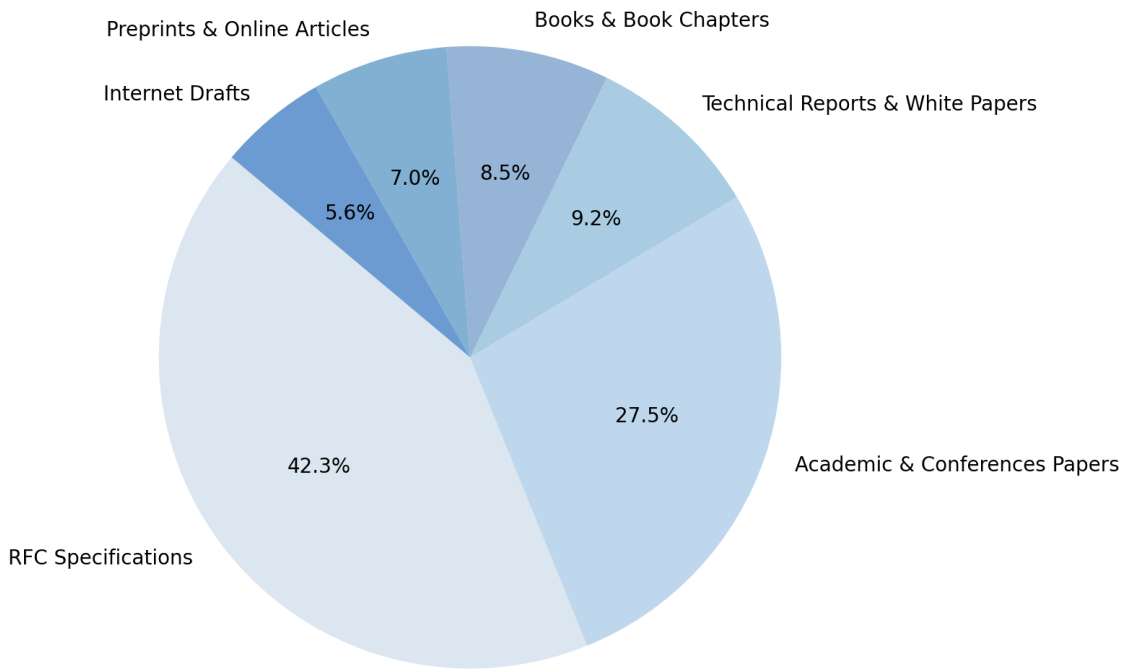


Figure 5A.1: Categorisation of Citations by Source.

Developing a categorisation approach within a framework to define a search strategy proved highly effective in reducing the time to identify appropriate articles for this study. The search string is based on keywords in tandem with the categorisation to filter articles that are aligned to address the research question. Figure 5A.1 highlights the percentage of selected articles per category. 42.3% of the articles were from RFC specifications, which address most of the fundamentals. These covered a mixture of technology standards, historical notes, implementation details and discussions addressing the concerns of topics relating to Internet identifiers. 27.5% resulted from academic literature – predominantly from IEEE and ACM databases– that addressed the fundamental topics of the DNS and Internet namespaces. Technical Reports accounted for 9.2%, including standards documentation, primarily from W3C regarding web references such as URIs, URLs and URNs. In addition, library reports on digital identifiers were also captured. Books and book chapters constituted 8.5% and mainly addressed topics related to Internet architecture and the fundamental design of distributed naming services. Online and preprint articles, which made up 7.0%, were especially important. These include public articles from the Library of Congress and their perspective on using persistent identifiers. Finally, Internet-Drafts (5.6%) provided insight into the literature on experimental approaches for using identifiers and supporting systems.

5A.4 Background

Various factors contribute to the functionality and design of the complex field of Internet identifiers, making it essential to understand these elements thoroughly. This section establishes the foundation of the chapter by tracing the developmental history of Internet identifiers, discussing namespaces and their role in structuring digital communication, and acknowledging their use across multiple disciplines. It clarifies the concept of a namespace, which is crucial to grasping

the following discussions in this chapter. Moreover, this section investigates Internet identifiers' specific considerations and challenges, focusing on maintaining coherence and referential integrity. Finally, this section examines the interplay between identity and location, clarifying these concepts in the context of Internet infrastructure and library systems.

5A.4.1 Development and History of Internet Identifiers

The Internet did not develop in isolation; it was influenced by contemporaneous network protocols and the interplay between the Internet, DECnet and other proprietary network protocols. A common feature of these protocols was their limited scope, so the inclination to associate natural language aliases with network-connected hosts was not considered a significant issue. The prevailing solution was to associate a 'word' with a protocol address, assemble this information for all hosts, and distribute a copy to every connected host.

Two methods ensured uniqueness in constructing identifiers: one involved using a statistically unique random number, and the other was a managed identifier space through a hierarchical system, which is further discussed in section 5A.4.3. In the case of DECnet, scalability challenges were encountered early in its development, leading to the adoption of a hierarchical, two-level routing structure comprising of 'areas' and 'local networks' and a corresponding two-level name alias system. The system used a namespace format of two labels separated by a double colon (AREA::NAME), tightly integrated with the network's connection topology [1].

In parallel, the rise of store/forward messaging systems introduced additional requirements. These systems, often not permanently connected, evolved without central coordination and rapidly expanded. Initial attempts to use single-level naming face collision issues, prompting a shift to a generalised approach to the two-layer naming system using an n-layer hierarchy. The Internet adopted this hierarchical naming approach coinciding with the transition from the HOSTS.TXT file system to the Domain Name System (DNS), which is further detailed in sections 5A.6.1 and 5A.6.3.

5A.4.2 The Role of a Namespace

Names categorise the world by assigning unique identifiers to concepts and entities. These names serve as a crucial means of conveying a shared understanding of a concept in communication, serving as essential components in maintaining referential integrity and enabling clear references. The total set of allowable names within a broad class is called a namespace. Namespaces enable the establishment of defined and limited environments in which the uniqueness of names is ensured within a certain context. For instance, a person's first name may be sufficient to distinguish them within a close-knit community of friends, family and coworkers. However, when the context expands to a larger population, such as a city, state or country, additional information, such as the middle and last name, is necessary to guarantee the uniqueness of the name. In human use contexts, a significant level of ambiguity is often tolerated as opposed to automated systems.

Automated systems are no exception regarding the importance of the role and coordination of naming within namespaces. A namespace creates a boundary for the unique use of names and requires strict constraints to eliminate ambiguities. These namespaces are often formally specified to ensure consistency, ensuring that names are unique within a certain context. However, the conflation of multiple roles of names to indicate 'who', 'where' and 'what' in automated systems has led to an overload of their semantic role [2]. This study highlights the three roles

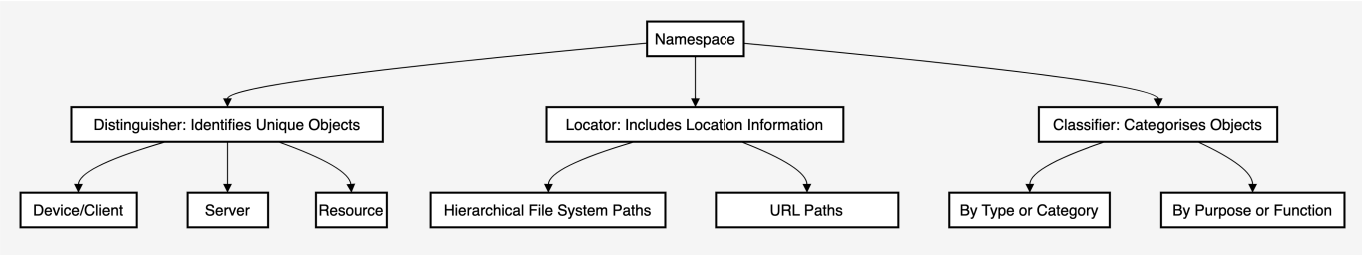


Figure 5A.2: Namespace Scope and Roles.

played by a name in a namespace for automated systems. Figure 5A.2 illustrates these multiple roles.

First, a name identifies and refers to a specific object, including a broad set of networked entities such as devices/clients, content, resources or servers. They are used to distinguish distinct instances of objects within a broad class [3]. Techniques to disambiguate larger data sets concerning Internet considerations are discussed in Section 5A.4.4.

Second, the scope of identity in a namespace may include a locator. This is often observed in hierarchical file systems, where the prefix before the file name functions as a location path [4], [5]. However, it may not be associated with a location context, such as the IMEI in mobile networks, which has no location context. It is important to note that incorporating location into an identity name can assist in the uniqueness of the name, particularly for automated systems. However, this is a convenient shortcut for automated systems, whereas, for human systems, there is an implicit location context of 'here' and 'now' in everyday conversation. The significance of this distinction is discussed in Section 5A.4.5 to compare and contrast the Internet challenges in terms of identity and location.

Finally, including a classifier in the namespace is optional and depends on the context of use. If the identity system is designed for a specific purpose, the context of its use is already defined. However, if the system is intended for multiple-use contexts, a classifier or 'what' parameter may be required. The classifier can be included as an attached qualifier or embedded in the identity token, depending on the system's requirements and efficiency. While embedding the classifier may be cheaper and more efficient in automated systems, it is not a mandatory feature of identity systems but rather a convenient shortcut in certain situations. The primary role of the classifier is to categorise objects across broad sets of classes using attributes within the namespace [6]. For example, library systems use classification techniques [7], [8] to logically categorise physical and electronic documents based on the subject and discipline, allowing users to browse and locate them.

5A.4.3 Flat and Hierarchies

To ensure that identifiers are uniquely allocated to entities within a flat namespace, a fundamental operation involves enumerating each identifier across the namespace to prevent conflict. A conflict arises when multiple entities are assigned the same identifier within a namespace. When this occurs, it becomes impossible for the namespace to resolve the identifiers unambiguously, as there are no distinguishing factors between conflicting entities. Despite these limitations, certain systems have found ways to operate efficiently within flat namespaces. For instance, The Onion Router (TOR) [9] and Distributed Hash Table (DHT) [10] are examples of systems that have managed to maintain the uniqueness of identifiers even as the size of the namespace

grows. However, as they become larger and more complex, the cost of administration and maintenance of identifiers and their bindings becomes non-trivial. Initially, flat namespaces were not considered viable because of concerns about their ability to preserve uniqueness under scaling pressure. Hierarchies have emerged as a solution to the challenges posed by flat namespaces in maintaining the uniqueness of identifiers as the environment grows [11].

A hierarchical system is composed of interrelated subsystems, with each subsystem being hierarchic in structure until the lowest level of the elementary system is reached [12]. A divide-and-conquer approach was employed using a hierarchical tree to manage large namespaces. This approach allows the namespace to grow beyond the constraints of any single node. This contrasts with the finite size of flat namespace environments and effectively circumvents any limitation on the number of names that a single node can hold.

The identifiers in this context contain a discernible structure by grouping objects that represent their positions in the hierarchy [13]. Hierarchy has several advantages over flat spaces. First, the separation of concerns in large namespaces can be achieved by creating smaller administrative domains, each of which can allocate identifiers from a namespace that does not collide with others in the hierarchical tree [14]. This also enables the independent governance of the namespace subtrees for administrative and scalability purposes. Second, while resolution through the hierarchical technique may sometimes be slow, it provides an efficient search for identifiers and their associated pointers in the namespace. This hierarchy allows for simple search functions that do not require the querier to assemble a local copy of the entire space for resolution. During a query, only the identity of the name server for the queried label must be locally stored.

Its tree-like structure allows navigation to traverse each parent and child node until the identifier is matched. Unlike hierarchical namespaces, flat namespaces inherently store identifiers sequentially. However, the task of searching in an unstructured flat space may not pose a significant challenge, assuming an appropriate sorting mechanism is employed. The transformation of an unstructured data set into a structured or sorted set serves as the key to efficient search operations. The index can then be used for searches, thereby reducing the time required to find an identifier. In contrast, hierarchies provide an alternative approach that does not require the assembly of all data at a single point in time.

Finally, the reusability of the identifiers in a hierarchical structure is an important advantage. Each part of a hierarchical name is resolved relative to a different context of relatively small size. The same identifier can be used with different semantic meanings in different contexts to suit a variety of uses [13]. In the DNS context, it is important to note that there is a distinction between names and labels. A name is an ordered sequence of labels, with each label acting as a node in the name hierarchy. The equivalence of two names is established if they both comprise the same set of labels in the same sequence. This facilitates the utilisation of a single label for multiple names without conflicts or ambiguity.

5A.4.4 Internet Considerations

There are two essential preconditions for effective communication between the two entities. First, communication is defined by a common symbol space and its usage. Second, a common semantic interpretation of the composed symbols is essential for preserving the intended or original meaning of the message. Failure to satisfy these two preconditions results in communication failure [15], [16]. A communication network relies on a unique symbol space with consistent

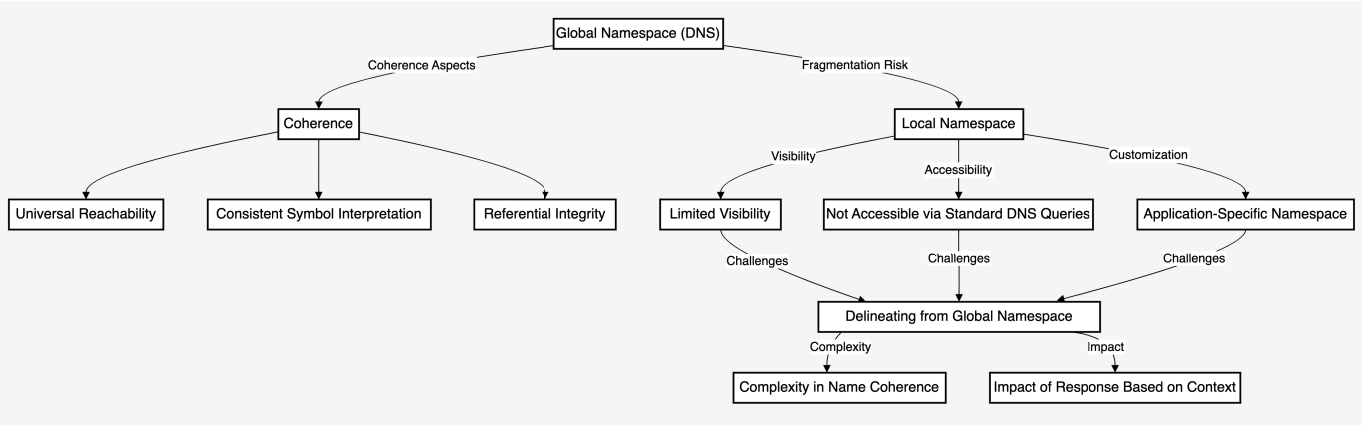


Figure 5A.3: Global Namespace Coherence and Fragmentation Risks.

semantic interpretation to ensure that any entity can understand the intended meaning of a message received. These strict rules are crucial to prevent symbols from being interpreted differently, leading to the loss of the original meaning [15].

The uniqueness of a common symbol space, the namespace, is crucial for ensuring the unambiguous usage of Internet identifiers, allowing them to be passed on, referred to and reused without semantic ambiguity. However, the growth and scaling of the Internet have led to the evolution of its name system mapping functions, allowing a name to refer to a constant digital object with a location relative to the querier rather than a single global locator. In addition, when a namespace avoids enclosing a unique location into its name value, issues related to the replication and distribution of digital objects become easier to reconcile. For example, if a digital object is assigned a unique identifier that is not location-specific, replication and distribution can occur without updating it.

Referential integrity in Internet namespaces ensures that a reference, when transferred from one entity to another, consistently signifies the same object across the Internet. This eliminates ambiguity in using these references and ensures a shared understanding of communicated terms as valid references. The system's capacity to accommodate Internet dynamism is of equal importance by regularly updating association pointers relevant to the location of the querying entity [17], [18].

For the Internet, as a public communications medium, to maintain its utility as a public service, its associated identifier namespace must be operated as a coherent public namespace [18]. In this context, fragmentation refers to the potential division of this unified namespace into smaller, disjointed parts. Such fragmentation poses a significant risk to the future of the Internet as it could lead to inconsistencies in symbol interpretation across different systems. Specifically, fragmentation can challenge the coherence of the Internet, which hinges on universal reachability, consistency in symbol interpretation and referential integrity [19]. Figure 5A.3 depicts the practice of using the '.local' top-level domain, a common strategy employed by Apple to create a variety of context-specific local namespaces. These namespaces are intentionally limited in visibility and cannot be accessed by standard DNS queries in the public namespace. This approach offers a unique method for creating application-specific namespaces yet concurrently introduces complexities in delineating the global public namespace of DNS from these local domains.

These complexities intensify when the constraints on these local namespaces fail or in scenarios

contemplating the delegation of the '.local' domain as the global scope domain. Therefore, ensuring name coherence in local and global network environments is an important factor. The issue of namespace fragmentation highlights the significance of identifiers in a broader context. Name systems have traditionally existed in a realm associated with a common infrastructure, where any application can use this common identity system, irrespective of the context of the query into the system, the result the name system provides in response is the same. 'Who' is asking does not materially alter the response. However, adding context and qualification to a query in an identification system can affect the response received because the identity of the querier is a factor, similar to how the 'Client Subnet' in the DNS [20] can impact the response.

It is debatable whether a single infrastructure name system can effectively handle various needs and applications or if a customisable namespace is required. Customised identification systems can be more specific to the realm of use and can provide more useful utility models for automated applications. However, the translation problem persists when attempting to transfer an identifier from one context to another.

5A.4.5 Technological Influences on Identity and Location:

Identity and location have gained increasing importance in the digital realm. This can be attributed to two key factors, mobility and cloning. The early digital environment consisted of mainframe computers dispersed across various floors and buildings, with network connectivity predicated on a fixed model that firmly enmeshed identity and location. However, with the advent of portable computers, the Internet has needed to support an always-connected model that accommodates endpoint mobility. The evolution of mobile cellular services exposed an aspirational goal for endpoints to retain a constant service and context, even when transitioning between different network connections, each with a unique IP context.

This has led to the challenge of IP addresses being viewed interchangeably by different network elements, making them difficult to distinguish. For example, from the routing system perspective, an IP address serves as a locator, whereas from the Domain Name System (DNS) [21] perspective, names are isomorphic to IP addresses, and the IP address acts as an identity that is treated as such by applications [22]. In scenarios where mobility is a critical attribute of the network architecture, the care-of addresses serve as temporal locators only, as endpoints transition between locations and providers, necessitating a new IP address for the device. However, this frequent change in IP addresses disrupts the continuity of ongoing sessions, thereby introducing a unique challenge for mobile networks. Section 5A.5.4 discusses this challenge and explores strategies to address frequent changes in IP addresses, considering the duality of identity and location inherent in mobile networks.

In contrast, when a file is copied between two systems, it generates two separate named instances. Each file's identification incorporates a distinct path namespace, leading to unique references despite the identical content. This raises challenges in identifying digital objects in a world where duplication and replication are necessary to optimise content delivery on the Internet. In the digital world, the proliferation of cloning has raised challenges in identity and location. For instance, within a library system, there may be numerous physical or digital replicas of the same book, all referred to as a single instance. Regardless of their location, these books are identified by a common identifier, often as a persistent URL. Library systems commonly employ key registry systems such as the International Standard Book Number (ISBN) and Digital Object Identifier (DOI).

Although the URL model is more intrinsically useful for locating the object being sought, it is less useful when the content might exist at multiple locations. Content Delivery Networks (CDNs) address this challenge by replicating content in proximity to user requests. This involves employing aggressive caching, geo-replication, and traffic engineering [23]. The fundamental challenge in digital environments is maintaining multiple instances of the same entity at different locations. This issue is compounded when the aim is to steer clients optimally towards the instance of a digital object that delivers the best efficiency or speed for each individual client.

Two strategies have been developed to address this problem. The first strategy involves a selective response to DNS queries for the named digital object. The system returns the IP address corresponding to the 'optimal' instance of the object for the client. Akamai illustrates this method well, using DNS as a geo-locator and modifying the response depending on the assumed identity of the querier [24]. In this context, IP addresses serve as geographical selectors.

Cloudflare's approach represents the second strategy. This involves using a small set of IP addresses announced into the routing system from multiple origination points. This strategy, known as anycast, allows the routing system to locate the optimal instance of a named object, leaving the DNS unaltered [25]. Both strategies enable URLs to mimic and behave like Uniform Resource Identifiers (URIs) [26]. This allows the establishment of a persistent identity while allowing the domain portion of the URL to serve as a dynamic locator.

In summary, the concepts of location and identity can often be interchanged using various mapping system functions. For example, in a mobility scenario, a device maintains its constant identity while its location changes. However, when digital objects are cloned or replicated, they simultaneously maintain a consistent identity across multiple locations. To support location and identity on the Internet, the burden shifts to different systems. For instance, in the case of mobility, updating the DNS to reflect the new IP address mapping when a device changes its location or informing the routing system of the device's new location helps to retain its IP address. In a scenario where content entities are cloned or exist in multiple locations, certain strategies are necessary to maintain the URL as a persistent identifier. One strategy involves using DNS to provide customised responses to clients' queries. This can include appending meta-data to the query resolution, such as using the source IP address to infer the querier's location. An alternative strategy is the anycast approach, in which the routing system independently guides the client to the optimal instance of the content entity. Each discussed approach for mobility, and cloning has its own design trade-offs, making it suitable in certain contexts and not others.

5A.5 Identifiers versus Locators: Parallel Insights into Library and Internet Systems

This section examines the roles of identifiers and locators in object identification, focusing on the differences between library and Internet systems. Starting with an analysis of the International Standard Book Number (ISBN) and Digital Object Identifier (DOI), this discussion explores how these persistent identifiers operate outside the typical Internet namespace, drawing insights from digital libraries and publishing. The section then shifts to U-systems, including the URL, URI, URN and IP namespace, to compare identity models across these varied contexts. A summary of distinct approaches to identifiers is presented in Table 5A.2.

5A.5.1 International Standard Book Number

This section explores important registry key systems that publish persistent identifiers outside the Internet namespace to provide insight into information management and resource identification for digital and non-digital resources. The examination of registry key systems such as the International Standard Book Number (ISBN) and Digital Object Identifier (DOI) [27] in Section 5A.5.2 are useful examples of the challenges they solve in the transformation of various media types.

ISBN is a separate model from the publishing industry used to create a unique identifier independent of the media type, e.g., hardback, paper or digital. The purpose of ISBN is to associate content with a unique identifier [28]. The International ISBN Agency manages the central registration of ISBNs, ensuring that each allocated ISBN is unique and corresponds accurately to its intended media. Importantly, the ISBN space is internally structured, facilitating the unique identification of the media.

When there are multiple copies and media types, identifying a specific instance without a unique identifier can be challenging. For example, if two books have the same title, author and publisher but different publication dates and editions, it may be difficult to discern which book is being referred to. The ISBN solves the distinguishing problem of assigning a unique key to a book. While numerous copies and media types may exist, the ISBN serves as a unique identifier for each book edition. However, the ISBN does not provide information on how to locate or retrieve a specific media instance. To find the media and its type, a separate catalogue search system or database that contains a mapping function to determine the availability and retrieval options is required. In addition, to resolve the ISBN, one must use its unique identifier as an input to retrieve publishing information related to the identified entity [27].

Once an ISBN is assigned to a book or monograph, it identifies the exact entity to be referred to. This identifier remains static, and should the content be revised, a new ISBN is issued. This contrasts sharply with the characteristics of a Uniform Resource Locator (URL) [29] on the World Wide Web (WWW). Unlike an ISBN, a URL does not specify the exact content retrieved but points to the location where it can be found. Moreover, the content associated with the URL is dynamic and can change over time. Content may also change locations, which poses a challenge for assigning persistent identifiers, as discussed in Section 5A.5.2.

5A.5.2 Digital Object Identifier

The Digital Object Identifier (DOI) system assigns persistent identifiers to content entities, whether physical, digital or abstract [27]. Although both ISBN and DOIs offer persistent identifiers for content entities, the DOI system was developed as an extension of the ISBN system to address challenges in the digital landscape. The handle system influenced the DOI system [30], developed by the Corporation for National Research Initiatives (CNRI) as a framework for managing digital information. The handle system includes a global naming service that stores persistent identifiers, called handles, for digital resources and resolves those handles into the information needed to locate, access and utilise the resources. The handle remains a persistent identifier, whereas its associated values can change to reflect the current state of the identified entity.

Libraries and publishing communities were among the first to adopt Digital Object Architecture (DOA) [31], [32] because they recognised the need to uniquely and unambiguously identify content entities rather than referencing them by location, which is an issue with existing iden-

tifiers such as URLs. Although some URLs are stable, they lack version control mechanisms to track changes to specific versions users might be interested in. Furthermore, when digital objects identified by URLs move between locations over time, these URLs can become invalid, causing additional challenges in updating the citation indices. This problem of maintaining a consistent identity with the associated metadata, exacerbated by the potential instability of URLs, is especially relevant for libraries responsible for digital preservation [33].

The handle system was developed outside the DNS system, enabling the creation of location-agnostic qualification strings, thus forming a separate namespace [34]. However, extending the DNS as a general-purpose name service encountered two main challenges. First, DNS operates on a zone-level management structure that lacks administrative capabilities at the individual name level. This presents a considerable challenge regarding the fine-grained control and governance of digital identifiers, especially for entities such as publishers who require individual control over a multitude of digital objects. Second, administrative controls cater to network administrators, making them suitable for domain administration but less so for general-purpose administration [30], [35].

The DOIs are governed and administered by the International Digital Object Identifier Foundation (IDF), which publishers use. The DOI system builds on the handle system, an integral component of the Digital Object Architecture (DOA) [33]. Similarly to the handle system, DOIs were initially mapped outside of the DNS, which required specialised client-side software to resolve handles [30]. However, as publishers and service providers were responsible for making DOIs resolvable to digital entities, the namespace was eventually integrated with the DNS using a URL scheme. By mapping DOIs to DNS, users can access and retrieve information through a common web browser without requiring client-side software, making the DOI system more accessible to a broader audience.

In summary, the identity representation of the ISBN and DOI is a persistent, unique identifier for a specific instance of a manifestation. All instances within the same identifier are considered complete and precise clones. The ISBN and DOI serve as identifiers that distinguish a member of a particular class (e.g., a book) from others. This system is location-independent, meaning that the location of an entity is not essential for identification. In this context, ISBN and DOI systems rely on a categorisation-based identity that is exclusionary in nature because content entities without an ISBN or DOI are not considered part of the specified class.

5A.5.3 URI, URL, and URN

In networked systems, a resource identifier serves as a compact surrogate for a resource, enabling identification, retrieval or both. Resource identifiers are represented as textual strings of characters that must be resolvable to yield the associated resource [36]. In the context of the World Wide Web (WWW), a Uniform Resource Identifier (URI) [37], [38] was introduced to define a method for identifying digital object instances, such as documents, images, videos, web pages or services that exist anywhere. Similarly to ISO Object Identifiers (OIDs), URIs were primarily intended to serve as pure identifiers rather than retrieve, dereference or locate a resource [38]–[40]. In addition to ISBN and DOI, a URI is considered a location-independent identifier, reinforcing the concept of a web object identity as a generic identifier [41]. URIs provide resource identification via a naming scheme or other resource attributes. Generic URI syntax specification does not imply the inclusion of names and addresses from various namespaces to be mapped in the URI strings [29]. URIs are general constructs in which the initial "scheme" part of the URI determines the structure and semantics of the remainder of the URI

Approach	Focus	Identifier Type	Advantages	Disadvantages
Location-Independent Identification	Describes 'what' the identity. Identity by categorization or classification. Identifiers that are exlusionary in nature.	Unique and Persistent identifiers (URI, URN DOI, ISBN)	Persistent identity, independent of location, effective for managing content across multiple instances	May not be suitable for resolving dynamic content, can be complex to implement and maintain, may provide multiple answers, including misleading ones, due to the vast amount of information associated with locators.
Location-Dependent Identification	Definition of 'where' the entity is. An Algorithmic-Based approach.	Locators with Specific Server-Side Instructions (URL)	Simple and efficient for one-to-one mapping of host to resource..	Not effective for content in multiple locations, can be vulnerable to changes in location. Requires CDN and DNS support for resolving URL to multiple locators.
Hybrid-Identification	Complementation of 'what' and 'where' the entity is.	IP Addresses, unique identifiers and locators	Duality expressed in the IP namespace is complementation between the two identity models.	Overloaded semantics, significant privacy concerns due to persistent identity tied to IP, potential for tracking and profiling.

Table 5A.2: Identity Models and Comparisons.

string. The following example demonstrates an abstract identifier consisting of a sequence of components referred to as the scheme, followed by a (":") and then a string (scheme-specific-part) [38].

scheme : scheme-specific-part

The URI syntax does not impose a specific structure or set of rules for the scheme-specific part of the URI. URIs are not widely used, with their limited traction possibly attributable to their amorphic and insubstantive nature, offering less immediate utility than URLs' tangible benefits. Conversely, the URL, a subclass of the URI, has been widely adopted as it appears easier to use. The URL identifies a resource by representing its primary access mechanism [42], thus establishing it as a functional form of identity that is location-dependent. The URL syntax enforces a specific structure to represent hierarchical relationships within a namespace. The following HTTP [43] URL example consists of a sequence of components referred to as the scheme, authority, path, query and fragment.

https://example.com/articles/technology?view=summary

In this example, the scheme defines the Hypertext Transfer Protocol Secure ('HTTPS') protocol [44], which is used to access the resource. The authority includes the domain name 'example.com', which resolves the identifier to a specific host. The subsequent path, such as '/articles/technology', represents an internal path within the local filestore of the identified server. Finally, the query component provides additional information to the server. In this case, 'view=summary' indicates a request for a summary view of the technology article. In the example provided, the URL builds on the DNS system. The DNS name is contextualised as a reference point, while the rest of the URL has semantic significance and is subject to server-side translation. In the context of URLs, a domain name reliably identifies the designated host. However, changes in resource location pose challenges. The DNS has solutions for this, with Canonical Name (CNAME) records mapping an alias to a true domain name and Delegation Name (DNAME) records redirecting an entire domain subtree to another domain. These features ensure stability at the domain level despite changes in resource locations. DNS

also incorporates advanced mechanisms such as Naming Authority Pointer (NAPTR) records. Rather than merely redirecting, a key role of NAPTR records is to phrase new queries through name transformation. More detail on these DNS techniques is provided in Section 5A.6.3.

In contrast to the challenges associated with the instability of URLs, Uniform Resource Names (URNs) are defined as a subclass of URIs that provide a persistent identifier for resources, independent of their location and access method [45], [46]. The resolution process for URNs is more complex than that of URLs. With URLs, the access mechanism is subsequently used to dereference the locator, retrieving a representation of the associated resource, such as a document [38], [45]. In contrast, URN resolution involves mapping a URN to one or more URLs, which still requires dereferencing the mapped locators to one or more resource representations. The key distinction in this identity system is persistence [45].

As a conceptual example, when a user interacts with a URN, the browser returns a set of candidate URLs, selecting one based on the location or access method to retrieve the resource while remaining entirely transparent to the user of the system [39]. However, similar to the challenges with URLs, if the locators change, such as the DNS name, the system must refresh and update the list to ensure that the URN remains resolvable and continues to yield the desired resource. The URN example identifier includes the scheme, namespace identifier (NID) and namespace-specific string (NSS):

urn:namespace-identifier:namespace-specific-string

In this example, the scheme identifies the type as URN. The URN includes a NID maintained by the Internet Assigned Numbers Authority (IANA) [47]. Examples of namespace identifiers include "NBN" for National Bibliography [48], "ISBN" for International Standard Book Number [49] and "ISSN" for International Serial Standard Number [50]. The NSS contains an opaque flat string within the NID that provides the format rules – for instance, a unique bibliographic identifier from the NBN.

There are several reasons URNs have not been widely adopted. One is that retrofitting the Internet to include the complex mapping of URNs to dynamic locators, which was introduced relatively late in the development of the Internet, created maintenance challenges regarding how URLs would be updated and refreshed as resources changed locations or cloned [51]. Additionally, URNs attempt to perform too many roles at once without a clear focus on one purpose, such as providing a persistent URL, location independence, a resolution system, or a pure identifier [39].

5A.5.4 Internet Protocol Address

The previous sections discussing ISBN, DOI and URI have revealed two distinct approaches to identity. The first approach involves assigning identifiers, such as ISBN or DOI, to each instance of a content entity that is an exact clone. In this model, identity is determined by the entity's membership within a specific class, unlike other existing classes. The second approach is an algorithmic-based method for identifying the location where the content entity can be found. In this model, identity is location-dependent. Aspects of identity, unique location and unique categorisation are essential for defining an object. These two models of identity, 'what' and 'where', complement each other in providing a useful namespace. The IP address [52] can be considered the ultimate complementation of these two identity models. However, IP

addresses are heavily overloaded semantically because they attempt to serve multiple purposes simultaneously [22], [53].

Fundamentally, an IP address [52] is used to identify the point of attachment of a host to a network [54]. Unlike prior address spaces such as DECnet, where all interfaces share a common identifier to represent the host [55], an IP address does not equate to a host. A deliberate decision was made in IP to adopt a more functional view of the address and address numbering. The function of this identifier is to indicate how the network interface of a host is reached. For example, when the forwarding system matches a packet's destination address, it directs a packet to the network interface associated with the host [56]. As traditional networking views assume a fixed model, the identifier is spatially and temporally unique and identical to its locator [57].

Consequently, an IP address not only identifies the host's interface but also serves as a means to access it [18], [54]. The dual role of IP addressing, serving as an identifier and locator, is inherently advantageous but also introduces the potential for confusion. This duality has been widely embraced, largely owing to the static nature of hosts, which permits a level of permanence in address assignment.

Mobility was never considered an attribute that required attention until the realisation that all endpoints – devices, services and content – might need to be able to frequently change network attachment [58], [59]. Including mobility as a network capability involves two approaches when an endpoint changes location. The first approach involves updating the network's forwarding information to reflect the new network location, which maintains service continuity but imposes a burden on the routing system. Alternatively, the address of the attachment point could be adjusted to reflect its new location. This approach reduces the impact on the routing system but may cause service disruptions during location changes.

Although IP addresses are heavily overloaded semantically, deliberate efforts have been made to develop various techniques for disambiguating location and identity to address mobility challenges. These approaches, extensively covered in the literature [60]–[64], share three key components in their design principles: a stable identifier for a mobile endpoint, a locator representing the IP address of the current location, and a mapping function to associate the identifier with the locator [60]. Refer to Figure 5A.4 for a representation of these principles.

Within the context of architectural support for mobility, there are two main techniques: handling host relocation, which is transparent to the network using a level of indirection in the routing architecture, and transport-level approaches, which ensure that sessions are sustained in the event of an endpoint IP address change. Although numerous methods have been proposed, the following examples illustrate the use of the IP address as a network namespace.

Mobile IP [65]–[67], the Home Address (HA) is elevated from its role as an interface identifier to a constant endpoint identifier, independent of the host's point of attachment within the network domain [68]. The Care-of Address (CoA) serves as a temporary locator, changing as the endpoint moves between the network domains. The home agent maintains a binding between the HA and CoA, which the architecture utilises when an endpoint transitions between the locations. The home agent is responsible for managing the mobility within the network, primarily by establishing tunnels [69], [70]. This facilitates seamless packet delivery to the relocated endpoint [71]. Importantly, this added layer of indirection localises the implications of a location change, affecting only the devices engaged in communication with the mobile entity. This approach avoids burdening the entire routing system with an incremental load, thereby promoting efficiency and continuity in mobile networking environments.

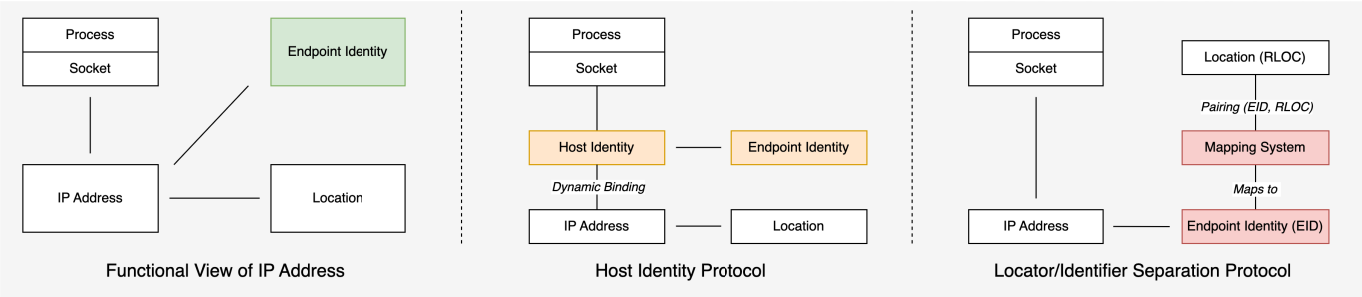


Figure 5A.4: Role of an IP Address: Example Implementations for Identity and Location Separation.

The Host Identity Protocol (HIP) [72], [73], described in Figure 5A.4, creates a new namespace between the IP layer and transport protocols. In this namespace, the host identifier is a public cryptographic key that allows hosts to authenticate to their peer hosts directly using their host identity [74]. The HIPs identity namespace replaces the traditional role of IP addresses in binding communication sessions, providing a stable connection if the endpoint's IP address changes [74]. The role of the IP address in the model is for routing purposes, creating a location/identity separation in the architecture.

The Locator/ID Separation Protocol (LISP) [75], described in Figure 5A.4, creates a disjoint IP namespace where the Endpoint Identifier (EID) is decoupled from the locator (RLOC) and is stored in its mapping system. LISP provides topological aggregation, leveraging a dynamic locator approach for IP routing, similar to the earlier effort of Pseudo-IP (PIP) [76]. LISP employs UDP encapsulation for its delivery mechanism, with the outer IP header containing the destination Routing Locator (RLOC). Based on this RLOC, the network core forwards packets to the destination's LISP-speaking gateway at the edge. The edge gateway is responsible for de-encapsulating the packet and subsequently forwarding it to the endpoint interface by utilising the EID to determine the original destination.

Despite various proposals for separating endpoint identity from location, there are known disadvantages. For example, there is the added cost of administrative overhead, additional infrastructure, and the need for extra levels of indirection through several encapsulation techniques, each with inherent challenges in the Internet's routing architecture [77]. Another concern is privacy, as IP addresses visible in IP packet headers serve as identifiers and can be associated with a client's identity. A skilled attacker in the path between the client and destination can potentially infer the client's identity and activity on the network through passive monitoring [78]. To address this privacy issue, efforts have been made to create temporal identities by generating randomised identifiers, particularly in IPV6 [79], [80].

In a recent effort to address the mass consumer market to ensure privacy protection is more accessible to Internet users, Apple's iCloud Private Relay was designed to safeguard user privacy online. This service protects user privacy by concealing their online activities from network intermediaries and passive observers. The principle of a Private Relay is that IP addresses identifying users should be separated from the names of websites users access [81]. As described in Figure 5A.5, the architecture consists of two tiers: ingress and egress. The responsibility of the ingress relay is for clients to establish a connection in which the original IP address is visible; however, the website names requested by the client are encrypted. Proxying traffic through the ingress relay shields the client's IP address from the egress relay, third-party intermediaries and destination server [82]. The technique employed by an ingress relay hides the client's IP address

while exposing sufficient information for the egress relay to understand the client's location for address allocation purposes. The ingress relay converts the client's IP address into a 'geohash', representing the user's location coordinates. Consequently, as the client communicates with the egress relay, the client's IP address is not forwarded. Refer to Figure 5A.6 for a visual illustration of this sequence.

The egress relay performs three primary functions. Firstly, it receives the 'geohash' from private relay clients, then consults its database to identify and assign the nearest corresponding IP address to the client. Second, it decrypts the website name, requested by the client. Finally, it completes the connection between the client and the requested website. The egress relay is not privy to the user client's original IP address. The egress relay only receives location information to assign the client the appropriate locator, which maps to the region from which the request originated.

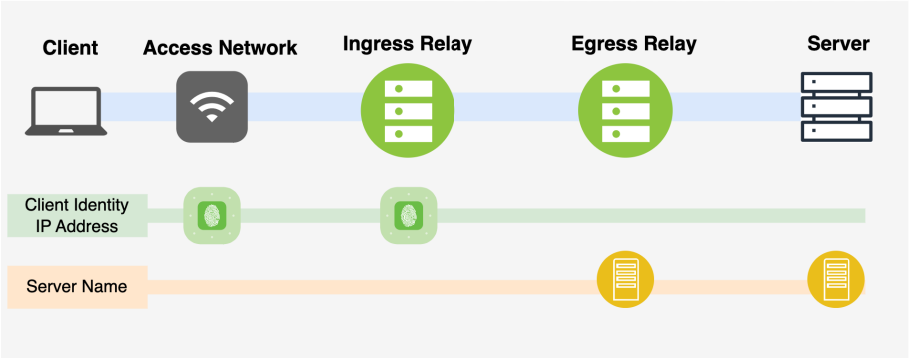


Figure 5A.5: Apple Relay: Anonymising Client's Identity.

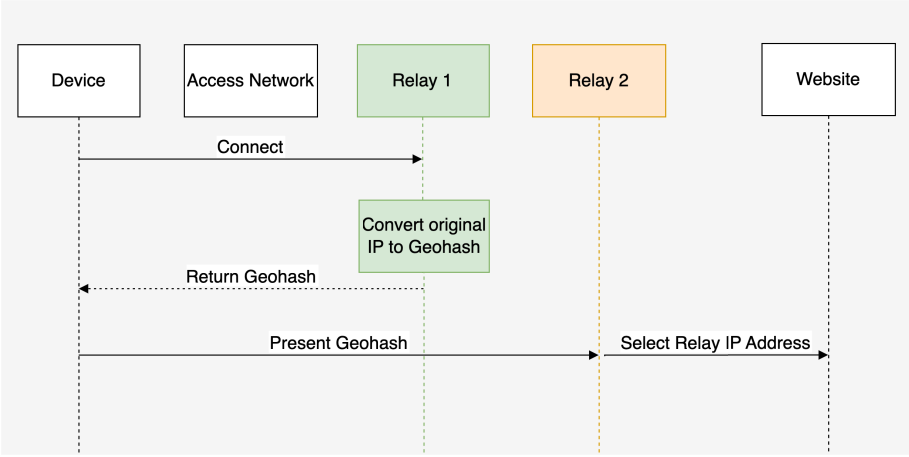


Figure 5A.6: Ingress and Egress Relay: IP Address Identity, and Location [81].

Because the egress relay has no knowledge of the original IP address of the user, the Relay IP addresses rotate over time and between sessions, preventing their use as stable identifiers for the user. In addition, the iCloud Private Relay utilises QUIC [83] to connect to the ingress relay. The tunnel to the egress employs a secure proxying protocol using the HTTP/3 [84], [85] proposed by the Multiplexed Application Substrate over QUIC Encryption (MASQUE) IETF working group [86]. Using QUIC as the transport protocol, MASQUE provides a secure connection that can combine multiple connections within a single proxy connection [82]. As part of this architecture, QUIC introduces a unique approach for sustaining transport-level

connections, which is particularly useful in mobility scenarios for roaming clients. Unlike previous transport-level approaches [68], [87], QUIC provides a semantic distinction between the identity of a connection to an endpoint and the current IP address and port numbers, similar to HIP [88] and Serval [89]. QUIC's connection endpoint identifier allows sessions to survive if the IP addresses and ports change [88]. For instance, if a client switches from a cellular to a Wi-Fi network, the QUIC connection ID allows the session to be recognised as an ongoing stream, irrespective of changes in the source IP address or UDP port numbers [88].

In summary, the fundamental changes brought about by mobility exert pressure on the IP namespace. To complement location and identity, distinct roles are assigned to IP addresses, requiring additional mapping systems to create disjoint namespaces, thus decoupling identity from location. In essence, names become isomorphic to IP addresses, and IP serves not only as a locator but also as an identity.

5A.6 Search Versus Identification: Comparing Directory and DNS Approaches

This section reviews the transformation of HOSTS.TXT into a distributed database and provides an overview of the initial namespaces that were critical to the early stages of Internet development, such as HOSTS.TXT and X.500. The focus is on a thorough exploration of the theme of search versus identification, offering a comparison and contrast between the X.500 and the DNS. The section then examines the evolution of the DNS in lookup design, including the adoption of successive generations of resource records that meet evolving business needs for flexibility, currency, and speed.

5A.6.1 HOSTS.TXT

In the early development of ARPAnet [90], the HOSTS.TXT file served as a directory to provide a human mnemonic for identifying resources, e.g., the printer [91]–[93]. Each line in the host table file contains a human-readable label, a network address and protocol-specific information supported by the resource [91]. An identical copy of this host table exists in each host. The Network Information Center (NIC) at the Stanford Research Institute (SRI) was responsible for managing the registrations and distribution of the HOSTS.TXT file. As each host was registered to the network or changes were made to label mappings, the NIC would ensure that the host table was current and distribute it to every host on the network.

However, as ARPAnet grew in popularity, the increased frequency of host table updates and distribution became burdensome and prone to failure. Consequently, the following challenges surfaced when using the HOSTS.TXT file: First, the host table approach assumes that changes occur infrequently and that the environment is constrained to a small number of hosts. The increased frequency of host table updates results from an increase in the number of hosts growing on the network. The increased frequency of updates posed a challenge to maintaining HOSTS.TXT so that network hosts could receive the latest version in a timely manner. Subsequently, the central server where HOSTS.TXT resided was overwhelmed by the number of download attempts, exacerbated by the HOSTS.TXT growing in size [3]. Second, the flat architecture of the HOSTS.TXT namespace introduced a significant challenge. Registrants who did not have the most up-to-date version of the host file could potentially attempt to register an already assigned name. Therefore, the registration process necessitated rigorous checks to

maintain the uniqueness of entries in the host file [94].

While the NIC could uniquely assign addresses from the address space, it had no controls to prevent an administrator from erroneously entering a host with a conflicting name. Moreover, host table specifications were established to provide names with a discernible structure and promote ease of recall. The specifications include format rules, such as limiting names to no more than 24 characters composed of case-insensitive letters (A–Z), digits (0–9), and hyphens ("-"). Additionally, names were prohibited from starting or ending with a hyphen and containing spaces or any other special characters [95]. In summary, the growing success of ARPAnet illuminates the inefficiencies and limitations of the host table file approach, highlighting its potential points of failure. The need for a more efficient and timely system became apparent, shifting the focus toward developing an on-demand query resolution system using a distributed data model. This new direction aimed to replace the static distribution of a host file with a dynamic model, facilitating timely updates and improved overall efficiency. This represented a critical step in the search for a scaleable, manageable and globally available system that could handle the increasing demands of a rapidly expanding digital network [96].

5A.6.2 X.500

The Open Systems Interconnection (OSI) reference model was standardised by the International Organization for Standardization (ISO) to create a vendor-neutral network architecture. The primary objective of this initiative was to devise an abstract network architecture that promoted inter-operation rather than fostering dependency on a particular vendor's implementation.

Thereafter, the X.500 Directory Service was developed as an integral part of the OSI's effort to manage network resources effectively. To address this issue, the X.500 directory service was created and standardised in ISO/IEC 9594 [97]. The X.500 directory is a logical database containing information about objects that model the physical world. The directory service manages names and their associated attributes and provides access to read and retrieve information in the database [98]. An important distinction of this directory system is that the returned answers are independent of the identity or location of the client that initiated the query [99]. Therefore, similarly to a telephone directory, the same answer is provided irrespective of the querier.

Although X.500 shares similarities with a telephone directory, key differences and limitations exist when applying these models in the digital space. For instance, a telephone directory can only return a telephone number when the complete name and address of the provider are given [100]. Otherwise, the telephone directory cannot facilitate the query; these detailed attributes are necessary to disambiguate entries. Additionally, physical telephone directories are published no more than once a year, resulting in incorrect and outdated information that cannot be refreshed until the next publication. X.500 addressed these challenges through its global namespace design, allowing a hierarchical search to facilitate incomplete queries based on its structured information model. If additional information is required, the attributes can be updated promptly to ensure the presence of distinguishing factors. For example, if two entries have the same first and last names, the supplementary information can be updated [98].

The implementation of the X.500 directory is based on the Directory Information Base (DIB) and Directory Information Tree (DIT). DIB is a structured information model composed of information that classifies objects called entries. Each entry consists of one or more attributes. In addition, each attribute type contains one or more values that define the attribute charac-

teristics [99]. DIT defines the relationships between entries to allow for a hierarchical search. The Distinguished Name (DN) uniquely identifies each entry in the Directory Information Tree (DIT) based on its location within the DIT hierarchy and the associated attributes, which are expressed through the namespace assigned to the object [101].

Each attribute associated with the entry is called a Relative Distinguished Name (RDN). To find a given entry's distinguished name (DN), the search starts at the tree root and traverses top-down until the entry is found. The DN concatenates each entry's RDNs traversed [99]. For example:

countryName = Australia

organization = UTS

organizationalUnit = Faculty of Engineering and IT

commonName = Andrew Martin

As described in the above example, the upper levels of the hierarchy are based on the country, organisation and organisational unit. This approach was designed to describe the geographic location of an object. Country, organisational and organisational units are helpful for hierarchical search, allowing users to locate candidate objects to choose from [100]. In contrast, other lookup systems, such as the DNS hierarchy, are based on naming conventions using a sequence of labels to construct a Fully Qualified Domain Name (FQDN) that is not necessarily location-based. Additionally, the DNS is not a searchable database and only returns an answer based on the exact query provided. The fundamental difference between these approaches lies in the distinction between search and identification, as depicted in Figure 5A.7. In a directory system, the emphasis is on strict search capabilities that enable users to perform targeted queries and retrieve specific information. Conversely, the DNS operates on a system of exact matches, where all queries and corresponding responses are exact [102].

In summary, the X.500 directory service has many attribute sets to describe its resources and users, which requires coordination to manage multiple points of authority at different levels of the hierarchy. Despite documented enhancements, deployments and case studies [103]–[105], the complexity of the Directory Access Protocol (DAP) limits its widespread adoption. However, it is important to note that the development of a Lightweight Directory Access Protocol (LDAP), a simplified version derived from DAP, resulted in significant advancements and widespread adoption. With its lightweight and streamlined approach, LDAP has been successful in various systems, including the Microsoft ecosystem [106], [107].

5A.6.3 The Domain Name System

The Internet employs multiple identities, each of which has distinct roles. As discussed in Section 5A.5.4, IP addresses illustrate a duality by functioning as identifiers for naming resources and services while also acting as locators essential for packet forwarding within the network. The DNS identity within URLs, detailed in Section 5A.5.3, maintains the identity of the service, requiring no modification even when the associated IP address changes. The domain name introduces an abstraction layer over the IP addresses. This allows URL identifiers to point to services without directly specifying their present location on the Internet [18].

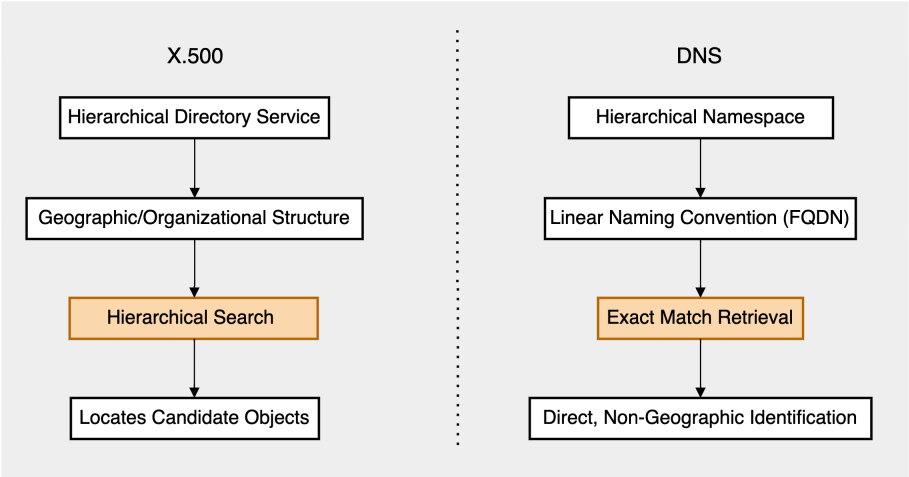


Figure 5A.7: X.500 and DNS: Search versus Identification.

The DNS can be described as method-independent [108], [109]. In essence, it assigns names to hosts regardless of the specific type of server or service in use. It functions as a naming system for naming computers or services, similar to how individuals are named in a telephone network, highlighting the notion of endpoint identity. The primary motivation of the DNS was to establish human-friendly identifiers for protocol addresses, while its hierarchical structure was designed to ensure scalability. The DNS established itself as being of critical importance to the Internet early on in its development owing to several important properties:

- The **Decentralised nature** of the distributed database was used to populate the DNS namespace, eliminating the need for a centralised organisational infrastructure to operate the entire namespace. Each entity operating a delegated part of the namespace can do this autonomously. This decentralisation allows the DNS to scale without creating critical bottlenecks, operating with limited coordination with others, which aligns precisely with the broader operation of the Internet itself.

- The **DNS resolution function** operates deterministically, using an exact match between the queried names and DNS names. There is no intended scope within the DNS for ambiguity in the name resolution function, rendering it highly suitable for automation in a deterministic computation environment. This is important because it can support a single unique naming space across the entire network where the same query generates the same response regardless of the location or identity of the querier. This determinism is the most important reason the DNS is critical to the Internet.

- **Resilience and Efficiency** in name resolution are the defining features of the DNS. The system avoids relying on any single point of failure. Caching results for a predetermined period using the TTL field reduces the load on the DNS resolver, thereby enhancing the efficiency of the system.

- The **Syntax** of DNS names involves concatenating individual labels to construct a DNS name and using a limited set of characters. This design permits the inclusion of natural, human-readable words within DNS labels, although it is constrained to only support ASCII characters in its standard format [108], [109].

The emergence of the DNS represented a significantly simpler form when compared to X.500 in Section 5A.6.2; however, it drew insights from the latter. It also incorporates elements from the

antecedents of the DECnet naming system, which used a two-level hierarchy. As described in section 5A.4.1, each cluster in DECnet was assigned an area name, permitting the allocation of unique names within that specific area. An identical name could be reused in different areas due to its namespace 'AREA::NAME' hierarchy. In contrast to the two-level hierarchy, the DNS design sought to move beyond these constraints. Recognising the limitations of a two-level hierarchy, DNS is deliberately designed to accommodate an arbitrary number of hierarchical levels. Each node in the hierarchical structure has an associated label. A unique domain name associated with a node is formed by assembling all labels from a specific node at the root of the tree [21], [110].

In the predecessor HOSTS.TXT [11] approach, the host table file operated on a just-in-case principle, providing all possible host mappings as needed. The maintenance of the HOSTS.TXT file was challenging, particularly when hosts were added, removed or had their IP addresses changed. This process required every file instance to be updated and distributed to each host, a procedure discussed in more detail in Section 5A.6.1. DNS pioneered a just-in-time query protocol, enabling efficient retrieval of specific records when necessary. Users can query a particular resource and obtain the corresponding answer instead of retrieving all the mappings. This approach assumes a robust and highly available network that enables specific database record queries on demand. One such specific record is the Address (A) record [109], which created a synthesis between the directory and protocol-level address whereby the namespace gained greater ascendancy. Addressing the deficiencies of HOSTS.TXT, the A record ensures that when a new host is added or an IP address changes, only the relevant record needs updating, eliminating the need to modify the entire host mapping file.

As the Internet's complexity increased, so did the requirements for DNS identifiers. While the A record, which is fundamental to the system, continued to play its central role, it was complemented by successive generations of DNS identifiers. These new iterations introduced additional abstractions and enhanced functionality, meeting the evolving needs of the Internet landscape and, importantly, serving the demands of developing an identifier system. Among these needs is the need to accommodate changes in the location of resources while ensuring the permanence of names. With these evolving requirements and DNS advancements, the following subsections will explore three main themes: first, the shift in control of a name in its entirety through outsourcing an indirection to a third party; second, the introduction of a persistent identifier that allows publishers greater control over the resolution process; and lastly, DNS's role as a rendezvous control point, fostering service-level connectivity based on intent.

Outsource Control: Canonical Name Records

As Internet outsourcing expanded, the need to relocate resources without altering their names became increasingly apparent. There was a growing need for entities to map individualised names into the infrastructure managed by specialised service providers. Although delegating control of a DNS zone to a third party was possible, transferring control of the name proved challenging. This constituted a significant limitation in maintaining a name while letting a third party run the corresponding service; a straightforward solution was required. To address the challenge of shifting control of a name to a third party, a Canonical Name (CNAME) was utilised [109], [111]. With a CNAME record, a name can point to another namespace without official delegation of the zone. This was instrumental in mapping names to a third-party infrastructure. Consider specialised service providers that host numerous entities. As described in Figure 5A.8, these entities could utilise the CNAME record to establish an alias in the service provider's domain by outsourcing an indirection. This strategy enables entities

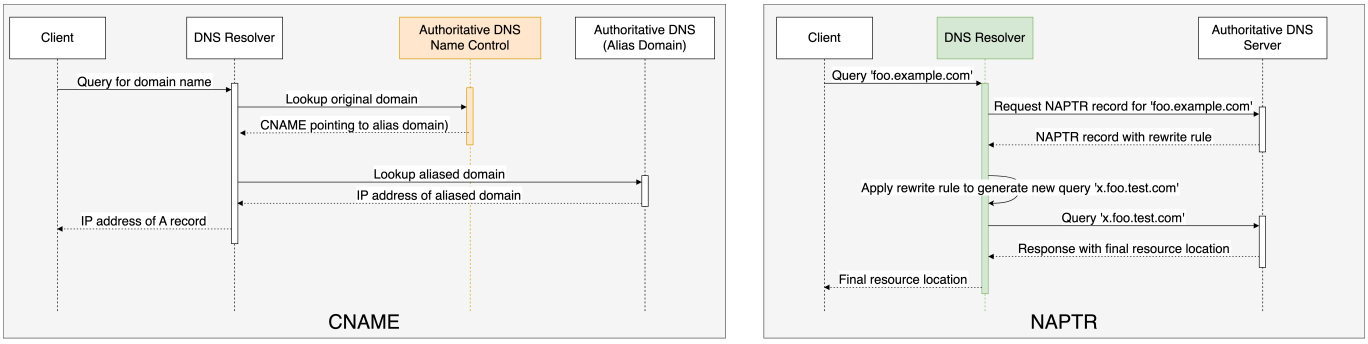


Figure 5A.8: DNS Lookup Process: CNAME and NAPTR Records.

to preserve their individual identities. Simultaneously, the naming system redirects queries to the resources managed by the hosting provider, which contributes to efficient query resolution. Importantly, this process operates in a manner that remains invisible to the end user.

However, the use of CNAMEs introduces unique considerations that have been the focus of industry discussion. Significantly, a CNAME alias controls the name in its entirety. This presents the issue of handling the apex of a delegation, as CNAMEs cannot reside at the apex; they must be situated at least one level down. Moreover, it is imperative to recognise that CNAMEs are affiliated with a distinct name, not a hierarchy. In contrast, the DNAME record [112] executes a broader role by redirecting an entire sub-tree of the DNS hierarchy to another domain. Effectively, a DNAME assimilates a name with all its descendants and maps it onto a disparate point within the namespace.

Publisher Control: Pointer Records

Building on the earlier discussion of DNS operations, this section delves into the challenges related to the control of the resolution path, which is an aspect intrinsic to DNS as a lookup service. These challenges stem from the widespread caching of DNS information. When a DNS name propagates, its mapping achieves a certain level of persistence, which can complicate efforts to modify or customise the associated responses. Disseminating a DNS name involves a commitment to resolvability; however, this is not total or indefinite. Publishers retain control via the record's 'time to live' fields, which can regulate how long a specific DNS record remains cached.

As described in Section 5A.5.3, a key factor in maintaining a URL's stability is the enduring existence of its associated DNS name, which is a critical consideration when the resource changes locations. In this context, the functionality of the Naming Authority Pointer (NAPTR) [113], a distinct DNS record type, is apparent. The NAPTR record enables dynamic translation into designated target names. It establishes a separate layer that maintains independence between naming and the resolution system, along with its associated protocols and services. This ensures that changes in one system do not directly impact the other [114]. NAPTR records provide a pivotal mechanism to facilitate redirection or transformation, thus reinforcing the permanence of URLs. For example, in Figure 5A.8, if a domain holder intends to relocate a resource from 'foo.example.com', the NAPTR records serve to redirect incoming requests. Rather than directly resolving 'foo.example.com', the DNS instructs the resolver to phrase a new query, such as 'x.foo.test.com'. This method ensures the integrity of the original URL, facilitating seamless redirection to the new resource location.

This NAPTR process involves the DNS server returning a regular expression pattern and a rewrite rule to the resolver. The resolver then applies this pattern to the original query name and, following the rewrite rule, generates a completely new query. It is crucial to note that this translation lies within the control of the publisher and not the user. This arrangement allows the publisher to relocate content while ensuring a persistent identity for the domain name, thereby transforming the domain resolution process from a semi-permanent to a dynamic state. This technique provides considerable flexibility in terms of rewriting rules.

The Straight-forward-Naming Authority Pointer (S-NAPTR) was introduced as a simplified alternative to the NAPTR, which utilises regular expressions for name transformations based on specific patterns [115]. In contrast, S-NAPTR focuses on service discovery by working in conjunction with SRV records [116] to provide explicit service details [115]. NAPTR also differs in approach compared to CNAME and DNAME. While CNAME translates one name into another in an iterative process, and DNAME maps all names under one delegation point to another, NAPTR provides a more nuanced approach. The NAPTR offers an adaptable class of names that adheres to a specified pattern and maps them to target names. Although not identical to the original, these target names must conform to the provided pattern. This is in contrast to DNS records such as DNAME, where the query labels to the left of the DNAME point are fixed. In contrast, the NAPTR allows these labels to undergo transformation according to a defined set of rules.

Despite their potential, NAPTR records have not seen widespread adoption owing to their complexity and the perception of unnecessary indirection. Many users find it simpler to customise the namespace directly instead of complicating the resolution process. In the identity system space, the publisher bears the burden, not the user. Systems that impose significant costs or burdens on the user are likely to fail, given the availability of free alternatives that transfer the entire burden onto the publisher. This principle is why systems such as ISBN and DOI, which could have imposed costs on users, continue to exist – because they do not impose additional name resolution burdens on the user.

Rendezvous as Intent: Service Records

The advent of Service Binding (SVCB) records [117] in DNS has marked a significant shift from previous DNS record types. Initially, service names were bound to domain names, meaning that all services sharing a logical name were grouped under a common domain name, offering limited flexibility. Services might share a common domain name for various human-use reasons, but there may be a need to disambiguate services and direct them differently, invisible to end users.

The primary goal of the SVCB record is to allow a client to gain all necessary information to establish a connection to a service with a single DNS query. Moreover, SVCB records provide a DNS aliasing functionality, which is especially useful at the apex of a domain. This functionality allows for the separation of service identity from location, eliminating the need for additional CNAME records and reducing the round trips required for name resolution.

Although SVCB and SRV [116] records exhibit shared characteristics, significant differences exist. An SRV record is non-extensible with a fixed number of fields. Any alterations to these fields require the creation of a new record type. In contrast, SVCB records can contain any number of key-value pairs, allowing for extensibility with new keys and values. Moreover, SRV records require an extra resolution step because they do not include elements comparable to the IPv4 or IPv6 hints found in the SVCB records. As described in Figure 5A.9, these hints

allow clients to initiate connections more rapidly by reducing the number of necessary DNS queries.

In the context of SVCB records, service binding requires encoding the service name, which is prepended in front of the domain name. This follows the Attrleaf naming pattern [118], [119] – for example, _service._port.example.com. With the introduction of SVCB records, DNS transitioned from being a simple locator to acting as a rendezvous broker, enabling more precise, user-intention-driven service discovery. Instead of translating a name to an IP address, DNS interprets the user's intent to determine the specific service they wish to access. This service is defined by a tuple of parameters such as port, protocol, and label, providing a precise description of the service point.

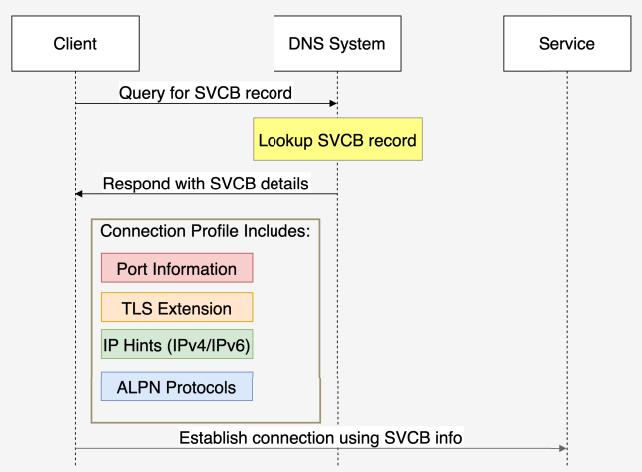


Figure 5A.9: Service Binding Record.

In this evolved model, when a query is made, DNS considers the user's location and the locations of all the potential service points. It then returns a connection profile with detailed information about the requested service, including aspects such as Application-Level Protocol Negotiation (ALPN) [120] and the necessary TLS parameters [121] for the subsequent connection [122].

In essence, the DNS evolved into a rendezvous broker, offering more than a simple IP address. It now provides a means of accessing specific services, thus significantly expanding its role. It is no longer just about objects or locations; it is about guiding users to the services they seek based on their intentions. This concept is an abstraction beyond the basic model, making DNS an arbiter of service access within a namespace.

5A.7 Evolutionary Pressures and Future Perspectives

Originally, the DNS modelled the telephone directory, a static system published periodically with invariant lookup keys. This directory was non-customisable and uniform for all users, permitting anyone to use the same lookup key and retrieve identical information. This approach has the appealing properties of a digital identity system. One could share a key, and others could access the same database to retrieve identical results, making it possible to refer to entities via their keys. However, with the evolution of the Internet, the concept of identity systems also evolved. The use of identifiers, especially within the DNS, began to shift. Instead of treating identifiers as constant indices in an invariant database, there was a growing need to customise these identifiers according to the needs of individual users. In the currently evolving DNS,

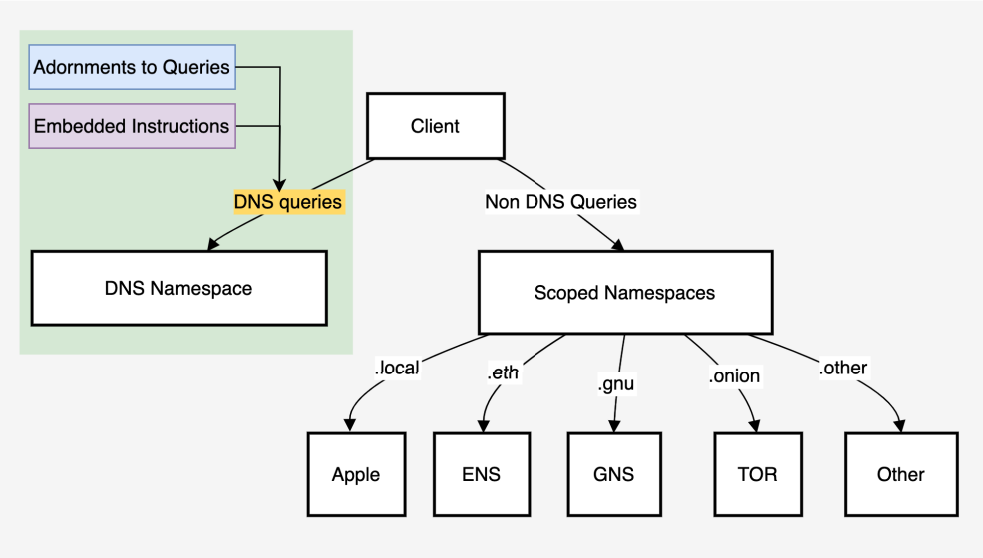


Figure 5A.10: Evolutionary Pressures and Expanding Namespaces.

queries are also based on the intention to generate a more comprehensive set of information, outlining the specific steps required to fulfil that intention. These intention-oriented queries consider a user's location and motivation, resulting in responses tailored to individual users. The extent to which intentions can be parameterised within these queries remains an active area of exploration.

Evolutionary pressures are driving the transformation of the DNS from a simple, efficient single lookup system into a more flexible computational resource that tailors answers to queries. In this context, query names no longer function merely as static lookup keys in a local database; instead, they can be regarded as microcode instructions, with the server operating as a computational engine. Furthermore, evolutionary pressures within the DNS, encompassing issues of uniqueness, time, performance and evolving service models, are reshaping the definition of an identifier within a namespace. From a computing perspective, this creates a shift in which identifiers' behaviour increasingly aligns with roles, services and clients' identities rather than simply distinguishing one resource from another.

The first two subsections below examine these changes in detail. First, the discussion of metadata and query qualification will explore the increasingly sophisticated metadata usage in DNS queries. Second, DNS as a Computational Resource observes query names as microcode instructions, a fundamental shift in how the DNS is utilised. These strategies, born from evolutionary pressures, carry distinct implications and properties, influencing various aspects of the DNS.

Beyond these transformations within the DNS, the broader Internet landscape has been evolving outside the conventional DNS realm, as per Figure 5A.10. These developments have led to the emergence of alternative identifier systems that can cater to specific needs and scenarios. Following the DNS-centric discussion, the scope expands to explore alternative identifier systems in their respective subsections. These include Apple's usage of the '.local' domain, TOR's '.onion' service, Ethereum's '.eth' approach, and the alternative GNU name system. Each of these systems offers a unique perspective in addressing the Internet namespace challenges outside the DNS, providing insights into the potential evolutionary directions of digital identifier systems. Please refer to Table 5A.3 for a summary of the subsections below.

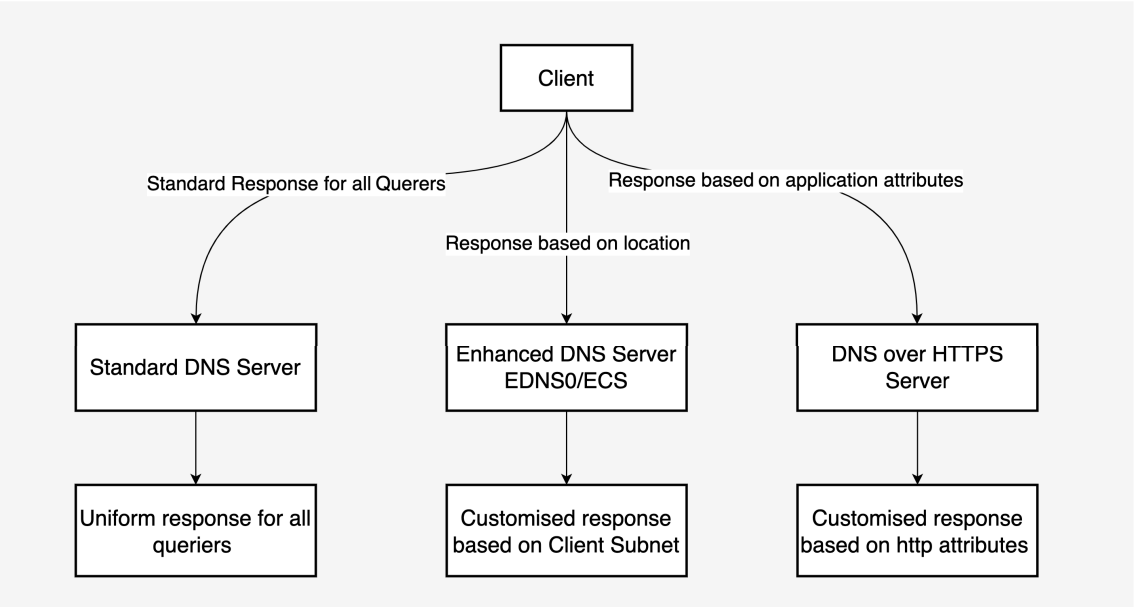


Figure 5A.11: Evolution in Customised Responses.

5A.7.1 Metadata and Query Qualification

In conventional DNS operations, a client's identity typically remains undisclosed. However, with the advent of Extension Mechanisms for DNS (EDNS0) [123] and the inclusion of the Client Subnet (ECS), [20], a client's IP address prefix is exposed and utilised as a part of the retrieval process. A DNS query goes beyond simply requesting a name in an evolved model. It now encompasses the parameters dictated by EDNS0 extensions and the client's subnet (ECS) capability. This process necessitates a form of information trade wherein the client provides its assumed identity, encapsulated in its subnet address, to receive an optimally proximate resource based on the server's estimation.

Consequently, the previously invariant DNS response is now tailored to these additional query attributes. This progression introduces a distinct tension as the traditionally anonymous DNS unveils more information about the client. However, this transition also brings its own set of advantages. The precision of programmatic DNS responses can be enhanced by exposing the aspects previously concealed by the DNS, as illustrated in Figure 5A.11.

With the advent of DNS over HTTPS (DoH), adding query adornments could further customise the query by including various HTTP attributes. The objective is not to overload the label space with customisation code but to integrate this customisation into the query as adornments that can refine or limit the server's responses, ensuring each response is uniquely customised. This shift in the operational paradigm of the DNS suggests a dichotomy. On the one hand, the namespace is viewed as static, providing invariant responses. On the other hand, the DNS query is decorated with metadata that determines the identifier behaviour. The server operates as a computational engine, and the DNS response is relative and customised to each query. This transformation underscores the tension between preserving client anonymity and the drive towards enhanced customisation, a key challenge in the DNS and a possible evolutionary direction.

5A.7.2 DNS as Computational Resource

A proposal known as Oblivious DNS (DNS) [124] has been introduced, although it has not been adopted by DoH or DNS over TLS (DOT) [125]. In the original DNS model, the query name undergoes encryption, yielding a new query name encoded in Base64. This encoded query name is directed not to the upper-level tree but to a specific domain name: 'obliviousdns.net'. This process involves a query string encrypted from the initial query. When the query reaches the oDNS server, acting as an authoritative server, it decrypts the encoded query name and retrieves the complete query name without prior knowledge of the query initiator. Essentially, the recursive resolver poses the question, not the original client. The authoritative server then switches its roles and becomes a stub resolver. It fully resolves the requested record, encrypts the result using the same key, sends it back to the recursive resolver, and subsequently delivers the answer to the client.

The primary focus in this example is not the identity-hiding mechanism of DNS but, rather, the embedded information within the query name. The server assigned to that query name now performs computations on behalf of the client. This situation marks a departure from the traditional DNS lookup. It now involves the query name posing a computational task, which the server executes.

To extend this concept, a DNS server can dynamically construct the response from the context established in the query name. The query name could contain instructions that prompt a specific reaction from the server, effectively transforming the DNS server into a programmatic, systematic entity. This concept could steer the future direction of DNS. This technique allows names to become more elastic, representing not only a static lookup but also a computational task. Consider the DNS as a microcode language, where the query effectively acts as a program for the server. In this scenario, the identifier namespace is not just a dictionary or directory but evolves into a distributed computational platform. From an evolutionary standpoint, shifting the dynamism of the namespace to convert DNS into a computational vehicle is a plausible future direction worth exploring.

5A.7.3 Multicast DNS and Apples and the '.local' Namespace

With Zero Configuration Networking (Zeroconf), Apple aimed to make technology more user-friendly by automating the discovery and communication process between devices and eliminating manual configuration. This vision was actualised through Bonjour, Apple's proprietary implementation of Zeroconf [126], [127].

Central to Bonjour's operation is the '.local' pseudo-domain, a namespace intentionally designed to streamline local networking environments. This namespace, primarily employed within smaller networks, is deliberately designed with restricted visibility to foster seamless device-to-device communication. Multicast DNS (mDNS), a protocol adept at providing name resolution services in localised networks where conventional DNS would be inefficient, was utilised to achieve this [128]. The mDNS executes multicast queries with a time-to-live (TTL) value of 1, effectively confining network traffic within its originating subnet [126]. Within this namespace, user-friendly devices such as printers can broadcast their capabilities, enhancing their discoverability by other network clients.

Unlike public DNS domains, which ensure global uniqueness, '.local' domains are designed to be link-local and not globally unique [129]. Hostnames within the '.local' domain are managed cooperatively by mDNS responders present within the local network, preserving the local nam-

ing scope and allowing the reuse of the same name in different contexts [126], [130]. In the event of a name conflict within the local network, Bonjour's design ensures automatic reassignment to a new, unique name while maintaining the integrity and consistency of network references.

However, this approach presents the Internet not as a single coherent namespace but as a plethora of local namespaces. These are intentionally limited and inaccessible through standard DNS queries in the public namespace. This unique strategy for creating application-specific namespaces brings forth challenges in distinguishing between the global namespace of DNS and these local domains. The '.local' domain is considered a special-use domain, not intended for use on the public Internet [131]. If this domain were to be delegated into the public space, it would disrupt the intentional constraints that Apple has implemented. While Apple's implementation with the '.local' domain is a noteworthy example, it is important to highlight that this strategy of localised, scoped visibility is part of a broader trend and an evolutionary pressure pushing towards a need for systems that limit the visibility of names to specific, defined sub-ranges rather than making them universally visible, as is the case with traditional DNS.

5A.7.4 TOR Network and the '.onion' Namespace

The motivation for creating The Onion Router (TOR) network was to establish an Internet environment that safeguards user privacy. The TOR architecture employs multilayered encryption to anonymise the end user. Each layer of encryption ensures that the user's profile remains confidential and cannot be exposed [132].

The TOR network uses public DNS infrastructure to handle regular DNS queries. However, it employs a technique to protect the identities of authoritative servers responsible for resolving popular domains on the public Internet and the identities of clients accessing the network. A name within the '.onion' namespace is derived from the public key of the asymmetric key pair [132]. When an onion service is set up, it automatically generates a private key and a corresponding public key. The public key is then hashed and truncated. The resulting hash is encoded using Base32, which yields a character string. This string is then suffixed with '.onion' to create the fully qualified domain name of the onion service, indicating that it is accessible only over the Tor network.

To enhance the recognisability of an onion address, vanity domains generate key pairs until the hash of the public key aligns with the desired string, thereby making the domain name reflective of the desired content [132]. It is important to note that the '.onion' domain does not rely on traditional DNS for resolution. Instead, queries for the '.onion' namespace are handled by the TOR protocol, which maps names to their corresponding network addresses within the TOR network.

The TOR was not originally intended to extend or augment the DNS. The primary goal was to respond to the pressures of providing anonymous communications. Similarly to Apple's '.local' namespace method, the '.onion' namespace is a special-use Top-Level Domain (TLD) that serves a distinct purpose [133]. It is deliberately designed to avoid resolution through the traditional DNS infrastructure. This deliberate separation ensures that the hidden TOR services remain concealed within the TOR network and are not exposed to the global public DNS namespace. Consequently, the Special-Use Domain Name Registry maintains a registry for these types of domain names reserved for special purposes, ensuring they are not utilised for general purposes.

5A.7.5 Ethereum Name Service and the '.eth' Namespace

Enforcement of uniqueness is a key aspect of online identities and transactions. This is typically maintained via centralised systems, where names are recorded on a ledger, ensuring that another cannot claim it once a name is used. However, alternatives exist to this standard model, one of which is blockchain-based systems. For instance, the Ethereum Name Service (ENS) leverages smart contracts to manage the decentralised registry of '.eth' domain names across the Ethereum network [134].

The ENS manages domains through three smart contracts: the Registry, Registrar and Resolver. The ENS Registry contains three key pieces of information: the owner of the ENS domain, the Resolver associated with the domain, and the time-to-live (TTL) for the domain records. The Registrar contract owns and allocates domains and subdomains under pre-defined rules in its smart contract. Finally, the Resolver synthesises the domain names into associate pointers, such as cryptocurrency addresses [135].

The ENS system ensures the uniqueness of each domain name through an implicit collision mechanism, which is activated when there is an attempt to register a claimed name. This decentralised approach mirrors the functionality of a central entity coordinating the uniqueness of domain names [136]. However, it is important to note that while the ENS operates on a decentralised framework, decisions regarding the registration, renewal and governance of top-level domains (TLDs) such as '.eth' are centralised.

This distinctive architecture presents an intriguing set of evolutionary pressures with advantages and disadvantages. Decentralised systems, such as ENS, offer benefits such as increased user pseudonymity. However, the inherent transparency of these systems can become challenging in public communication settings, distinct from private ones, where the lack of complete anonymity can pose significant issues. Striking a balance between pseudonymity, where users have a persistent but not directly identifiable presence, and full anonymity, analogous to sending mail without a return address, often leads to societal dilemmas. Consequently, a degree of residual accountability, such as a return address for mailed letters, is often seen as a positive attribute in public systems.

5A.7.6 GNU Name System and the '.gnu' namespace

GNS uses '.gnu', a pseudo-top-level domain (TLD), to distinguish names resolved in the GNS namespace from those resolved in the DNS namespace [137]. This unique structure leads to a distinctive approach in name resolution, where, unlike hierarchical DNS, every user's zone is treated as an independent root. This results in all zones becoming equal starting points for the resolution process [138]. The resolution process in the GNS is transitive, following delegations from one zone to another based on public keys. These keys, pivotal for navigating different zones, are located using a Distributed Hash Table (DHT), a decentralised data structure that efficiently maps keys to values [138]. In contrast, DNS uses an iterative resolution process, which begins at a globally recognised root zone and continues through a sequence of authoritative servers from the TLD to the subdomain to resolve the IP address of the requested resource. Although both systems use delegation, their resolution methodologies are fundamentally different. Like DNS, GNS allows owners complete control over their namespaces. Additionally, users can delegate a part of their namespace, requiring any label within that domain to map to another identity. This feature mirrors DNS, easing the transition to this alternative architecture [138].

Evolutionary Response	Namespace	Application Domain	Usecase
Metadata Enhanced DNS Queries	DNS	Content Steering, Internet Infrastructure	Optimize content delivery.
DNS as Computation Resource	DNS	Dynamic Content Delivery, Internet Infrastructure	Compute the answer, rather than the location of the answer.
Apple mDNS	.local	Local Network Services	Discovery and communication on local networks.
TOR Network	.onion	Privacy and Secure Communication	Anonymous access to TOR services
Blockchain-based Naming	.eth, .bit, .crypto, etc.	Blockchain, Decentralized Web	Provide readable names to wallets and websites.
GNS Name System	.gnu	Censorship Resistant Name Service	Alternate name system peer-to-peer network for storing and resolving names.

Table 5A.3: Summary of Responses to Evolutionary Pressures.

The emergence of alternative naming systems outside the traditional DNS has amplified evolutionary tensions toward the interoperability of various Internet namespaces. For instance, the '.gnu' TLD is a distinguishing label that prompts a GNS-aware resolver to transition from the traditional DNS resolution process to the GNS protocol. This mechanism allows names intended for resolution in the GNS namespace to be handled correctly, thus mitigating conflicts with DNS and other name systems.

5A.7.7 Innovation versus Integration: A Discussion

The use of names within references determines how resources are accessed. A name passed between two entities should resolve to the same digital resource. This is only achievable if names are allocated from a unified namespace and the resolution of the name into network addresses is consistent and coherent. Market competition dynamics for digital presence naturally encourage all participants to be bound within this unified namespace residing in the DNS. Moreover, for the Internet to continue functioning as a public communications system, it is essential to remain within the same namespace to maintain coherence. This study outlines examples of alternate namespaces and notes the caveat that the parallel use of such alternate name systems risks fragmenting the Internet's namespace, eroding the value of a unified and cohesive communication network.

One approach to support cohesion across diverse namespaces involves augmenting name resolvers with classifier functionality for various name suffixes such as '.eth', '.bit', '.onion', '.gnu' and '.local'. As described in Figure 5A.12, this method introduces a distinguisher label to specify the namespace that is used. From an architectural point of view, this approach may provide a potential avenue toward interoperability among multiple namespaces, necessitating minimal changes to the existing global DNS infrastructure [139]. However, the challenge lies in determining who decides which distinguisher labels are used by each of these namespaces. This will only be an effective approach with a coherent and universally accepted method for organising such distinguishing labels.

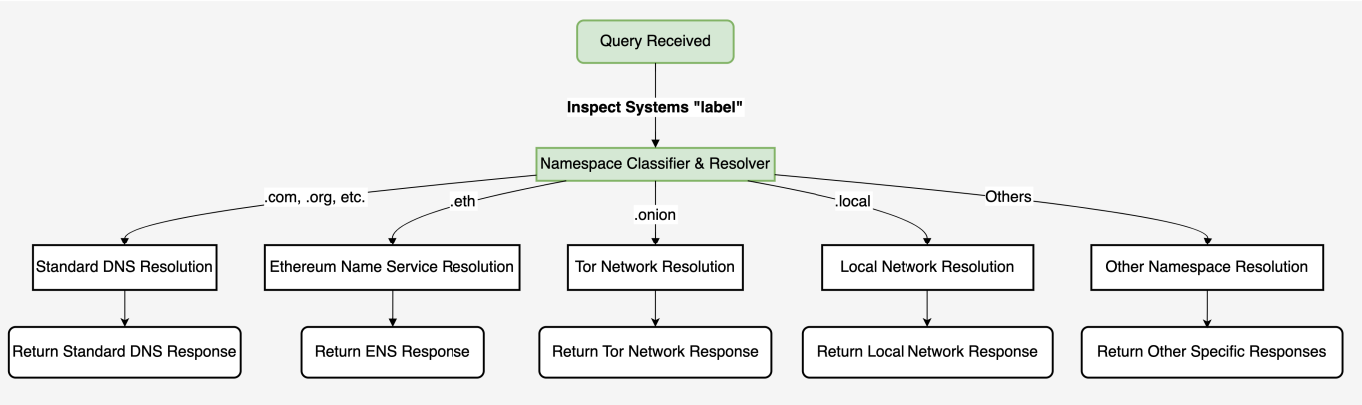


Figure 5A.12: Classifier for Multi Namespace Forwarding.

5A.8 Conclusions

The evolutionary pressures that shaped the alternative namespace have not strayed far from the root of the DNS and remain fundamentally mappable to the DNS; the key challenge lies in preserving cohesiveness.

The foundation of trust in Internet interactions primarily stems from confidence in the naming system, a need made more acute by the indeterminism of the routing and addressing system, influenced in the latter case by IPv4 address exhaustion. In this context, the trust placed in Internet transactions relies on the exchange of credentials between the engaging parties. This crucial process enables the receiving party to verify that the responses they receive are from an entity that controls the domain name of the accessed service.

Navigating the evolution of Internet identifiers is intricate, as it is critical to accommodate various aspects of technical innovation in a gradual and incremental manner while simultaneously preserving the coherence of the identifier space. These may appear to be conflicting goals, as they often stand in opposition.

Chapter 5B

Proof By Argument: Alternate Paths for the DNS

5B.1 Purpose of this Chapter

This chapter expands the systematic literature investigation and outlines a proof-by-argument of alternative behaviours necessary for the DNS resolution system and namespace as a second research contribution. The chapter develops the argument to support the need to review the alternative behaviours that the DNS ecosystem could adopt to maintain the incumbent as a unified namespace for the Internet. This chapter is critical as the usages of the described DNS modes are absent in the Internet literature. These contributions establish the foundations for answering the research proposition and offer a clear direction for the proposed DNS model outlined in Chapters 6A and 6B.

5B.2 The Case for an Evolutionary DNS Innovation

It is recognised that no technology is immutable, irrespective of its efficiency, precision and performance. The DNS is no exception; it has experienced and will continue to experience an array of pressures for change. It is essential to continue examining these pressures and understand their expression. In the context of the DNS, the central discussion is 'where' and 'why' the DNS is heading. The first response is that it may be perceived as perfect. The second response is that there is a need for change through one of two approaches. The first strategy advocates a radical change that would be marked as a direct confrontation with the incumbent; without seamless integrations, users will find themselves inside separate Internets, A or B. If such a disruptive approach is expected to compete with the DNS, the system's overall benefits should be orders of magnitude greater than the DNS rather than offering merely marginal improvement.

This chapter argues that if a revolutionary approach, described in Chapter 5A, is not deemed feasible, then a more nuanced strategy is essential. This involves observing the evolutionary DNS system usages and developing a set of responses that address pressures, thereby enabling co-existence within the existing Internet infrastructure.

In the following sections, the article will discuss the alternative behaviours and varying degrees of freedom within the namespace that can alter the behaviour of the system. This includes

the dynamic relationship between client and publisher, the temporal nature of the name, the persistence of the name, and the applicability of the name as adjustable aspects of the system. Given the limited literature on these topics, this section employs a thought experiment to outline potential marginal approaches and responses through adjustments that avoid direct confrontation while introducing necessary features to the modern DNS.

5B.3 Customised Responses: Query Qualification and Context-Awareness

As described in Chapter 5A, a client's identity typically remains undisclosed in DNS operations. However, with the inclusion of the Client Subnet (ECS), [20], the client's IP address prefix is exposed and utilised as a part of the retrieval process. This progression introduces a distinct tension as the traditional DNS reveals more information about the client. However, this transition also brings its own set of advantages. The precision of programmatic DNS responses can be enhanced by exposing aspects previously hidden by the DNS.

DNS over HTTPS (DoH) could offer a promising approach to adorn the query with various HTTP attributes. The objective is not to overload the label space with customisation code but to integrate this customisation into the query as adornments that can refine or limit the server's responses, ensuring each response is uniquely customised. This shift in the operational paradigm of the DNS suggests a dichotomy. On the one hand, the namespace is viewed as static, providing invariant responses. On the other hand, the DNS query is decorated with metadata that drives the identifier behaviour based on the decorated queries. In this case, a computational element is required to customise the response for each query. This transformation highlights the tension between preserving client anonymity and the drive towards enhanced customisation, a key challenge in the DNS and a possible evolutionary direction.

5B.4 Temporal Identities: for Permanent vs. Impermanent Names

In the security industry, the use of one-time tokens being valid for a single transaction or session has greatly enhanced data security and prevented unauthorised access, promoting confidentiality. Consider translating this thought into the DNS, where names are usable only once. Impermanent names could be an effective tool for ensuring privacy. Non-replayable names could be more effective for ensuring acute privacy. Even under passive monitoring, inspecting a transaction that is nonreproducible by accessing the name holds no value as there is nothing to inspect retrospectively or use.

This concept borrows from the cryptographic system, where session keys are employed. Once the session key is generated and used, there is no further use for that key. Names could likewise reflect this usage and behaviour. By utilising cryptographic techniques, the system can introduce a one-time flag where it remains unused if a particular flag is not acknowledged in a transaction, thus creating a replay-resistant name.

A network is defined by a common addressing and common naming system. However, in the context of ephemeral names, there is a unifying principle that transcends individual transactions. When a name is published, it is without foresight into who will request it in subsequent interactions. Interestingly, it is these subsequent queries that delineate the scope of the network

in which the individual operates. Furthermore, the ephemeral name evolves into an entity that fundamentally shapes the network. Finally, investigating alternative behaviours, such as adjusting the lookup design and namespace to reveal ephemeral names on a need-to-know basis, could present a promising direction to address privacy and confidentiality concerns.

5B.5 DNS Data: Storage vs. Real-Time Computation

In the earlier stages, the DNS operated on a 'compute once, use many' principle, implying that storage was cheaper than computation. Hence, the computation results can be stored and reused numerous times. This originated in a period when computational resources were scarce, which led to the strategy of distributing DNS information closer to the receivers despite resulting in a loss of control over its answers. However, this was acceptable as long as the computation that generated the answer was accurate, allowing for confident reuse of results.

Today, however, a paradigm shift is occurring in the DNS realm. Computation resources have become abundantly available, challenging the initial principle by leaning towards real-time computation. This shift indicates that it might be more efficient and flexible to compute data in real-time rather than storing it for repeated use, primarily due to the drastically decreased cost of recomputing data.

Cloudflare's approach to IP address agility is a notable example of this paradigm shift [140]. Address agility refers to the ability to reuse, reassign and reallocate IP addresses in an isolated and independent manner from the names they are commonly bound to. In the conventional DNS, name-to-IP mappings are static and stored in a record or lookup table. However, with Cloudflare's system, the binding occurs on-query, meaning the address is returned in response to a DNS query. This implies that addresses are bound to names when a response is returned rather than being stored within any record. This distinction, though subtle, is significant.

Instead of having IP addresses pre-assigned to names, Cloudflare's authoritative DNS operates based on policies. It adopts a policy-first approach, selecting an address pool assigned to the respective policy, and ignores the name entirely.

When a DNS query arrives, attributes such as the user location of the originating query and the account type are matched within the policy. Consequently, the IP addresses are computed and assigned during runtime or query time. This demonstrates that the binding has become ephemeral and can change without regard to previous bindings, resolvers or clients.

In this new approach within the DNS, pulling back answers once stored closer to the receivers and computing the questions as they are asked allows for greater control and flexibility to customise answers individually.

5B.6 DNS as Computational Resource

In terms of the computational example in the DNS, the Oblivious DNS (oDNS) proposal [124] mentioned in Chapter 5A serves as one of the primary examples. The computational element found in this technique is the embedded information within the query name. The server assigned to that query name now performs computations on behalf of the client. This approach marks a shift from the traditional DNS lookup as it now involves the query name posing a computational task, which the server executes.

The DNS as a computational server, could dynamically construct the response from the context established in the query name. The query name could contain instructions that prompt a specific reaction from the server, effectively transforming the DNS server into a programmatic, systematic entity. This concept could steer the future direction of DNS. This technique allows names to become more elastic, representing not just a static lookup but a computational task. Consider the DNS as a microcode language, where the query effectively acts as a program for the server. In this scenario, the identifier namespace becomes not just a dictionary or directory but evolves into a distributed computational platform. From an evolutionary standpoint, shifting the dynamism of the namespace to convert DNS into a computational vehicle is a plausible future direction that requires continued exploration.

5B.7 Computational Integration with the DNS

Consider the transformation in the search world. For instance, search interprets and responds in the query language instead of directing users to resources. The relative importance of the original web resource from which these answers are derived diminishes when direct answers are provided instead of simple pointers. In this evolution, the search engines interact with the web page, not the end users. As computational needs integrate with the DNS, it could become capable of discerning the underlying intent of a query. Instead of merely providing a location for the answer, it can deliver the answer directly. This evolution positions the DNS as a computational resource. Chapter 6B provides an instantiation of this model.

5B.8 Conclusions

The above examples demonstrate the increasing dependency on computational methods to extend the functionality of the DNS. Customising DNS responses relies on computing client-specific answers to enhance performance. Similarly, the creation of temporary identity involves computing one-time names to enable privacy from third-party monitoring. The need for real-time computation in constructing dynamic resource records outlines the demand for answers. Furthermore, computational methods have invoked privacy-hiding mechanisms like ODNS to protect client identities. Futuristic approaches that seek to integrate computation into the DNS could benefit clients; rather than DNS merely pointing to a location, users could get answers based on the query's intent.

There will be a clear shift in where efforts are directed, where value increases and where value is lost. Augmenting the computational aspect within the DNS is an emerging direction. It will advantage agile players with sufficient scale and resources to perform this service at speed. Conversely, it will disadvantage the existing infrastructure of a distinction between name resolution and computational service. While it is unlikely to receive unanimous support, this development represents an additional element of the ongoing disruption.

References for Chapters 5A and 5B

[1] T. C. Bartee, *ISDN, DECnet, and SNA Communications*. H.W. Sams, 1989, p. 414, ISBN: 9780672225123.

[2] M. Bowman, S. K. Debray, and L. L. Peterson, "Reasoning about naming systems," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 15, no. 5, pp. 795–825, 1993.

[3] N. R. Council *et al.*, *Signposts in cyberspace: the Domain Name System and internet navigation*. National Academies Press, 2005.

[4] B. Walker, G. Popek, R. English, C. Kline, and G. Thiel, "The locus distributed operating system," *ACM SIGOPS Operating Systems Review*, vol. 17, no. 5, pp. 49–70, 1983.

[5] A. Adya, W. J. Bolosky, M. Castro, *et al.*, "{Farsite}: Federated, available, and reliable storage for an incompletely trusted environment," in *5th Symposium on Operating Systems Design and Implementation (OSDI 02)*, 2002.

[6] R. Kumbhar, *Library classification trends in the 21st century*. Elsevier, 2011.

[7] L. M. Chan, S. S. Intner, and J. Weihs, *Guide to the Library of Congress classification*. ABC-CLIO, 2016.

[8] W. A. Wiegand, "The" amherst method": The origins of the dewey decimal classification scheme," *Libraries & culture*, pp. 175–194, 1998.

[9] D. Goldschlag, M. Reed, and P. Syverson, "Onion routing," *Communications of the ACM*, vol. 42, no. 2, pp. 39–41, 1999.

[10] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, and S. Shenker, "Rofl: Routing on flat labels," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, 2006, pp. 363–374.

[11] D. Mills, "Internet name domains," RFC Editor, RFC 799, Sep. 1981.

[12] H. A. Simon, "The architecture of complexity," in *Facets of systems science*, Springer, 1991, pp. 457–476.

[13] G. Coulouris, J. Dollimore, and T. Kindberg, *Distributed systems: Concepts and design edition 3*, 2001.

[14] S. Paul, J. Pan, and R. Jain, "A survey of naming systems: Classification and analysis of the current schemes using a new naming reference model," *WUSTL Technical Report*, 2009.

[15] "Iab technical comment on the unique dns root," RFC Editor, RFC 2826, May 2000.

[16] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[17] M. P. Evans and S. Furnell, "The resource locator service: Fixing a flaw in the web," *Computer Networks*, vol. 37, no. 3-4, pp. 307–330, 2001.

[18] P. Fältström and G. Huston, "A Survey of Internet Identities," Internet Engineering Task Force, Internet-Draft draft-iab-identities-02, Dec. 2004, Work in Progress, 40 pp. [Online]. Available: `https://datatracker.ietf.org/doc/draft-iab-identities/02/`.

[19] G. Huston, *Dns evolution: Innovation or fragmentation?* `https://blog.apnic.net/2022/09/30/dns-evolution-innovation-or-fragmentation/`, Accessed: January 18th, 2024, 2022.

[20] C. Contavalli, W. van der Gaast, D. C. Lawrence, and W. ". Kumari, "Client Subnet in DNS Queries," Tech. Rep. 7871, May 2016, 30 pp. DOI: `10.17487/RFC7871`. [Online]. Available: `https://www.rfc-editor.org/info/rfc7871`.

[21] P. Mockapetris and K. J. Dunlap, "Development of the domain name system," in *Symposium proceedings on Communications architectures and protocols*, 1988, pp. 123–133.

[22] D. Meyer, L. Zhang, and K. Fall, "Report from the iab workshop on routing and addressing," RFC Editor, RFC 4984, Sep. 2007, http://www.rfc-editor.org/rfc/rfc4984.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4984.txt.

[23] G. Peng, "Cdn: Content distribution network," *arXiv preprint cs/0411069*, 2004.

[24] K. Schomp, O. Bhardwaj, E. Kurdoglu, M. Muhaimen, and R. K. Sitaraman, "Akamai dns: Providing authoritative answers to the world's queries," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 465–478.

[25] M. Calder, A. Flavel, E. Katz-Bassett, R. Mahajan, and J. Padhye, "Analyzing the performance of an anycast cdn," in *Proceedings of the 2015 Internet Measurement Conference*, 2015, pp. 531–537.

[26] A. Babakian, P. Monclus, R. Braun, and J. Lipman, "A retrospective on workload identifiers: From data center to cloud-native networks," *IEEE Access*, vol. 10, pp. 105 518–105 527, 2022.

[27] N. Paskin, "Digital object identifier (doi®) system," *Encyclopedia of library and information sciences*, vol. 3, pp. 1586–1592, 2010.

[28] S. Griffiths, "Isbn: A history," *Information Standards Quarterly*, vol. 27, no. 2-3, pp. 46–48, 2015.

[29] T. Berners-Lee, L. Masinter, and M. McCahill, "Uniform resource locators (url)," RFC Editor, RFC 1738, Dec. 1994, http://www.rfc-editor.org/rfc/rfc1738.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1738.txt.

[30] S. Sun, L. Lannom, and B. Boesch, "Handle system overview," RFC Editor, RFC 3650, Nov. 2003.

[31] R. E. Kahn, *An open architecture for a digital library system and a plan for its development*. Corporation for National Research Initiatives, 1988.

[32] R. Kahn, R. Wilensky, *et al.*, "A framework for distributed digital object services," *International Journal on Digital Libraries*, vol. 6, no. 2, pp. 115–123, 2006.

[33] J. Hakala *et al.*, "Persistent identifiers-an overview," *KIM Technology Watch Report*, 2010.

[34] A. Durand, "Digital object architecture and the handle system," *Los Angeles: ICANN—Office of the CTO*, 2019.

[35] S. Sun, S. Reilly, and L. Lannom, "Handle system namespace and service definition," RFC Editor, RFC 3651, Nov. 2003.

[36] G. Janée, *Resource identifier*. 2009.

[37] T. Berners-Lee, R. T. Fielding, and L. Masinter, "Uniform resource identifiers (uri): Generic syntax," RFC Editor, RFC 2396, Aug. 1998, http://www.rfc-editor.org/rfc/rfc2396.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2396.txt.

[38] T. Berners-Lee, R. T. Fielding, and L. Masinter, "Uniform resource identifier (uri): Generic syntax," RFC Editor, STD 66, Jan. 2005, http://www.rfc-editor.org/rfc/rfc3986.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3986.txt.

[39] "About uris." (2007), [Online]. Available: https://www.loc.gov/standards/uri/about.html (visited on 04/13/2007).

[40] T. Berners-Lee, "Universal resource identifiers in www: A unifying syntax for the expression of names and addresses of objects on the network as used in the world-wide web," RFC Editor, RFC 1630, Jun. 1994.

[41] T. Berners-Lee, *The world wide web: Past, present and future*, Aug. 1996. [Online]. Available: https://www.w3.org/People/Berners-Lee/1996/ppf.html.

[42] M. Mealling and R. Denenberg, "Report from the joint w3c/ietf uri planning interest group: Uniform resource identifiers (uris), urls, and uniform resource names (urns): Clarifications and recommendations," RFC Editor, RFC 3305, Aug. 2002.

[43] R. T. Fielding, J. Gettys, J. C. Mogul, *et al.*, "Hypertext transfer protocol – http/1.1," RFC Editor, RFC 2616, Jun. 1999, http://www.rfc-editor.org/rfc/rfc2616.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2616.txt.

[44] E. Rescorla, "Http over tls," RFC Editor, RFC 2818, May 2000, http://www.rfc-editor.org/rfc/rfc2818.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2818.txt.

[45] P. Saint-Andre and J. Klensin, "Uniform resource names (urns)," RFC Editor, RFC 8141, Apr. 2017.

[46] K. Sollins, "Architectural principles of uniform resource name resolution," RFC Editor, RFC 2276, Jan. 1998.

[47] "Uniform resource names (urn) namespaces." (2023), [Online]. Available: https://www.iana.org/assignments/urn-namespaces/urn-namespaces.xhtml (visited on 03/14/2023).

[48] J. Hakala, "Using national bibliography numbers as uniform resource names," RFC Editor, RFC 8458, Oct. 2018.

[49] J. Hakala and H. Walravens, "Using international standard book numbers as uniform resource names," RFC Editor, RFC 3187, Oct. 2001.

[50] S. Rozenfeld, "Using the issn (international serial standard number) as urn (uniform resource names) within an issn-urn namespace," RFC Editor, RFC 3044, Jan. 2001.

[51] L. of Congress, "The relationship between urns, handles, and purls," 1997.

[52] J. Postel, "Internet protocol," RFC Editor, STD 5, Sep. 1981, http://www.rfc-editor.org/rfc/rfc791.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc791.txt.

[53] B. E. Carpenter and B. E. Carpenter, "General Identifier-Locator Mapping Considerations," Internet Engineering Task Force, Internet-Draft draft-carpenter-idloc-map-cons-01, Jun. 2007, Work in Progress, 22 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-carpenter-idloc-map-cons/01/.

[54] J. F. Shflch, "Inter-network naming, addressing, and routing," 1978.

[55] R. Perlman, *Interconnections: bridges, routers, switches, and internetworking protocols*. Addison-Wesley Professional, 2000.

[56] J. H. Saltzer, *On the Naming and Binding of Network Destinations*, RFC 1498, Aug. 1993. DOI: 10.17487/RFC1498. [Online]. Available: https://www.rfc-editor.org/info/rfc1498.

[57] Y. Rekhter, J. Crowcroft, and B. E. Carpenter, *IPv4 Address Behaviour Today*, RFC 2101, Feb. 1997. DOI: 10.17487/RFC2101. [Online]. Available: https://www.rfc-editor.org/info/rfc2101.

[58] A. Venkataramani, J. F. Kurose, D. Raychaudhuri, K. Nagaraja, M. Mao, and S. Banerjee, "Mobilityfirst: A mobility-centric and trustworthy internet architecture," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 74–80, 2014.

[59] E. Lear and R. Droms, "What's In A Name:Thoughts from the NSRG," Internet Engineering Task Force, Internet-Draft draft-irtf-nsrg-report-10, Sep. 2003, Work in Progress, 24 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-irtf-nsrg-report/10/.

[60] R. Wakikawa, Z. Zhu, and L. Zhang, *A Survey of Mobility Support in the Internet*, RFC 6301, Jul. 2011. DOI: 10.17487/RFC6301. [Online]. Available: https://www.rfc-editor.org/info/rfc6301.

[61] M. Komu, M. Sethi, and N. Beijar, "A survey of identifier–locator split addressing architectures," *Computer Science Review*, vol. 17, pp. 25–42, 2015.

[62] W. Ramírez, X. Masip-Bruin, M. Yannuzzi, R. Serral-Gracia, A. Martínez, and M. S. Siddiqui, "A survey and taxonomy of id/locator split architectures," *Computer Networks*, vol. 60, pp. 13–33, 2014.

[63] L. Zhang, A. J. Mankin, J. W. S. III, D. T. Narten, and D. M. Crawford, "Separating Identifiers and Locators in Addresses: An Analysis of the GSE Proposal for IPv6," Internet Engineering Task Force, Internet-Draft draft-ietf-ipngwg-esd-analysis-05, Oct. 1999, Work in Progress, 52 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-ipngwg-esd-analysis/05/.

[64] A. C. Snoeren, H. Balakrishnan, and M. F. Kaashoek, "Reconsidering internet mobility," in *Proceedings Eighth Workshop on Hot Topics in Operating Systems*, IEEE, 2001, pp. 41–46.

[65] C. E. Perkins, *IP Mobility Support*, RFC 2002, Oct. 1996. DOI: 10.17487/RFC2002. [Online]. Available: https://www.rfc-editor.org/info/rfc2002.

[66] C. Perkins, "Ip mobility support for ipv4," RFC Editor, RFC 3344, Aug. 2002, http://www.rfc-editor.org/rfc/rfc3344.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3344.txt.

[67] C. Perkins, D. Johnson, and J. Arkko, "Mobility support in ipv6," RFC Editor, RFC 6275, Jul. 2011, http://www.rfc-editor.org/rfc/rfc6275.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6275.txt.

[68] A. C. Snoeren and H. Balakrishnan, "An end-to-end approach to host mobility," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, 2000, pp. 155–166.

[69] C. E. Perkins, *IP Encapsulation within IP*, RFC 2003, Oct. 1996. DOI: 10.17487/RFC2003. [Online]. Available: https://www.rfc-editor.org/info/rfc2003.

[70] T. Li, D. Farinacci, S. P. Hanks, D. Meyer, and P. S. Traina, *Generic Routing Encapsulation (GRE)*, RFC 2784, Mar. 2000. DOI: 10.17487/RFC2784. [Online]. Available: https://www.rfc-editor.org/info/rfc2784.

[71] C. E. Perkins, "Mobile ip," *IEEE communications Magazine*, vol. 35, no. 5, pp. 84–99, 1997.

[72] P. Nikander and R. Moskowitz, "Host Identity Protocol Architecture," Internet Engineering Task Force, Internet-Draft draft-moskowitz-rfc4423-bis-00, Dec. 2009, Work in Progress, 25 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-moskowitz-rfc4423-bis/00/.

[73] R. Moskowitz, P. Jokela, T. Henderson, and P. Nikander, *Host Identity Protocol*, RFC 5201, Apr. 2008. DOI: 10.17487/RFC5201. [Online]. Available: https://www.rfc-editor.org/info/rfc5201.

[74] P. Nikander, A. Gurtov, and T. R. Henderson, "Host identity protocol (hip): Connectivity, mobility, multi-homing, security, and privacy over ipv4 and ipv6 networks," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 2, pp. 186–204, 2010.

[75] D. Farinacci, V. Fuller, D. Meyer, D. Lewis, and A. Cabellos-Aparicio, *The Locator/ID Separation Protocol (LISP)*, RFC 9300, Oct. 2022. DOI: 10.17487/RFC9300. [Online]. Available: https://www.rfc-editor.org/info/rfc9300.

[76] P. Francis, *Pip Near-term Architecture*, RFC 1621, May 1994. DOI: 10.17487/RFC1621. [Online]. Available: https://www.rfc-editor.org/info/rfc1621.
[77] T. Li, *Recommendation for a Routing Architecture*, RFC 6115, Feb. 2011. DOI: 10.17487/RFC6115. [Online]. Available: https://www.rfc-editor.org/info/rfc6115.
[78] S. Farrell and H. Tschofenig, *Pervasive Monitoring Is an Attack*, RFC 7258, May 2014. DOI: 10.17487/RFC7258. [Online]. Available: https://www.rfc-editor.org/info/rfc7258.
[79] D. S. E. Deering and B. Hinden, *Internet Protocol, Version 6 (IPv6) Specification*, RFC 8200, Jul. 2017. DOI: 10.17487/RFC8200. [Online]. Available: https://www.rfc-editor.org/info/rfc8200.
[80] D. T. Narten, R. P. Draves, and S. Krishnan, *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*, RFC 4941, Sep. 2007. DOI: 10.17487/RFC4941. [Online]. Available: https://www.rfc-editor.org/info/rfc4941.
[81] *Icloud private relay overview*, Dec. 2021. [Online]. Available: https://www.apple.com/privacy/docs/iCloud_Private_Relay_Overview_Dec2021.PDF.
[82] P. Sattler, J. Aulbach, J. Zirngibl, and G. Carle, "Towards a tectonic traffic shift? investigating apple's new relay network," in *Proceedings of the 22nd ACM Internet Measurement Conference*, 2022, pp. 449–457.
[83] J. Iyengar and M. Thomson, *QUIC: A UDP-Based Multiplexed and Secure Transport*, RFC 9000, May 2021. DOI: 10.17487/RFC9000. [Online]. Available: https://www.rfc-editor.org/info/rfc9000.
[84] M. Bishop, *HTTP/3*, RFC 9114, Jun. 2022. DOI: 10.17487/RFC9114. [Online]. Available: https://www.rfc-editor.org/info/rfc9114.
[85] T. Pauly, E. Rosenberg, and D. Schinazi, "QUIC-Aware Proxying Using HTTP," Internet Engineering Task Force, Internet-Draft draft-pauly-masque-quic-proxy-06, Mar. 2023, Work in Progress, 24 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-pauly-masque-quic-proxy/06/.
[86] *Multiplexed application substrate over quic encryption*, May 2020. [Online]. Available: https://datatracker.ietf.org/doc/charter-ietf-masque/.
[87] J. Iyengar, C. Raiciu, S. Barre, M. J. Handley, and A. Ford, *Architectural Guidelines for Multipath TCP Development*, RFC 6182, Mar. 2011. DOI: 10.17487/RFC6182. [Online]. Available: https://www.rfc-editor.org/info/rfc6182.
[88] *A look at quic use*, Jul. 2017. [Online]. Available: https://www.potaroo.net/ispcol/2022-07/quic.html.
[89] E. Nordström, D. Shue, P. Gopalan, *et al.*, "Serval: An end-host stack for service-centric networking," in *Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, 2012, pp. 85–98.
[90] M. Hauben, "History of arpanet," *Site de l'Instituto Superior de Engenharia do Porto*, vol. 17, pp. 1–20, 2007.
[91] L. Deutsch, "Host names on-line," RFC Editor, RFC 606, Dec. 1973.
[92] M. Kudlick, "Host names on-line," RFC Editor, RFC 608, Jan. 1974.
[93] P. Karp, "Standardization of host mnemonics," RFC Editor, RFC 226, Sep. 1971.
[94] D. E. Comer and L. L. Peterson, "Issues in using darpa domain names for computer mail," in *Proceedings of the ninth symposium on Data communications*, 1985, pp. 158–164.
[95] K. Harrenstien, M. Stahl, and E. Feinler, "Dod internet host table specification," RFC Editor, RFC 952, Oct. 1985.
[96] C. Liu and P. Albitz, *DNS and Bind.* " O'Reilly Media, Inc.", 2006.

[97] I. Rec, "X. 509— iso/iec 9594-8," *Information Technology-Open Systems Interconnection-The Directory: Authentication Framework, Editor's DRAFT*, 1993.

[98] D. M. Piscitello and A. L. Chapin, *Open systems networking: TCP/IP and OSI*. Addison-Wesley Reading, MA, USA, 1993.

[99] M. T. Rose, *The open book: a practical perspective on OSI*. Prentice-Hall, Inc., 1990.

[100] J. K. Reynolds, S. Heker, and C. Weider, "Technical Overview of Directory Services Using the X.500 Protocol," Tech. Rep. 1309, Mar. 1992, 16 pp. DOI: 10.17487/RFC1309. [Online]. Available: https://www.rfc-editor.org/info/rfc1309.

[101] G. W. Neufeld, "Descriptive names in x. 500," in *Symposium proceedings on Communications architectures & protocols*, 1989, pp. 64–71.

[102] P. Vixie, "What dns is not," *Communications of the ACM*, vol. 52, no. 12, pp. 43–47, 2009.

[103] N. Meulemans, "A yellow pages service based on x. 500," *Computer networks and ISDN systems*, vol. 28, no. 14, pp. 1939–1946, 1996.

[104] C. Weider and R. Wright, "A survey of advanced usages of x. 500," Tech. Rep., 1993.

[105] D. L. Silver, J. W. Hong, and M. A. Bauer, "X. 500 directory schema management," in *Proceedings of 1994 IEEE 10th International Conference on Data Engineering*, IEEE, 1994, pp. 393–400.

[106] T. Howes, M. Smith, and G. S. Good, *Understanding and deploying LDAP directory services*. Addison-Wesley Professional, 2003.

[107] R. G. Lavender and M. Wahl, *Internet directory services using the lightweight directory access protocol*. 2004.

[108] *Domain names - concepts and facilities*, RFC 1034, Nov. 1987. DOI: 10.17487/RFC1034. [Online]. Available: https://www.rfc-editor.org/info/rfc1034.

[109] *Domain names - implementation and specification*, RFC 1035, Nov. 1987. DOI: 10.17487/RFC1035. [Online]. Available: https://www.rfc-editor.org/info/rfc1035.

[110] U. of Western Ontario. Dept. of Computer Science, *Naming and Name Management Systems: A survey of the state of the art*. [London]: Department of Computer Science, University of Western Ontario, 1989.

[111] R. Elz and R. Bush, *Clarifications to the DNS Specification*, RFC 2181, Jul. 1997. DOI: 10.17487/RFC2181. [Online]. Available: https://www.rfc-editor.org/info/rfc2181.

[112] S. Rose and W. Wijngaards, *DNAME Redirection in the DNS*, RFC 6672, Jun. 2012. DOI: 10.17487/RFC6672. [Online]. Available: https://www.rfc-editor.org/info/rfc6672.

[113] M. H. Mealling and D. R. Daniel, *The Naming Authority Pointer (NAPTR) DNS Resource Record*, RFC 2915, Sep. 2000. DOI: 10.17487/RFC2915. [Online]. Available: https://www.rfc-editor.org/info/rfc2915.

[114] D. R. Daniel and M. H. Mealling, *Resolution of Uniform Resource Identifiers using the Domain Name System*, RFC 2168, Jun. 1997. DOI: 10.17487/RFC2168. [Online]. Available: https://www.rfc-editor.org/info/rfc2168.

[115] A. Newton and L. Daigle, *Domain-Based Application Service Location Using SRV RRs and the Dynamic Delegation Discovery Service (DDDS)*, RFC 3958, Jan. 2005. DOI: 10.17487/RFC3958. [Online]. Available: https://www.rfc-editor.org/info/rfc3958.

[116] A. Gulbrandsen and D. L. Esibov, *A DNS RR for specifying the location of services (DNS SRV)*, RFC 2782, Feb. 2000. DOI: 10.17487/RFC2782. [Online]. Available: https://www.rfc-editor.org/info/rfc2782.

[117] B. M. Schwartz, M. Bishop, and E. Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs)," Internet Engineering Task Force, Internet-Draft draft-ietf-dnsop-svcb-https-12, Mar. 2023, Work in Progress, 61 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-dnsop-svcb-https/12/.

[118] D. Crocker, *Scoped Interpretation of DNS Resource Records through "Underscored" Naming of Attribute Leaves*, RFC 8552, Mar. 2019. DOI: 10.17487/RFC8552. [Online]. Available: https://www.rfc-editor.org/info/rfc8552.

[119] D. Crocker, *DNS Attrleaf Changes: Fixing Specifications That Use Underscored Node Names*, RFC 8553, Mar. 2019. DOI: 10.17487/RFC8553. [Online]. Available: https://www.rfc-editor.org/info/rfc8553.

[120] A. Hutton, J. Uberti, and M. Thomson, *The ALPN HTTP Header Field*, RFC 7639, Aug. 2015. DOI: 10.17487/RFC7639. [Online]. Available: https://www.rfc-editor.org/info/rfc7639.

[121] S. Friedl, A. Popov, A. Langley, and S. Emile, *Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension*, RFC 7301, Jul. 2014. DOI: 10.17487/RFC7301. [Online]. Available: https://www.rfc-editor.org/info/rfc7301.

[122] J. Zirngibl, P. Sattler, and G. Carle, "A first look at svcb and https dns resource records in the wild," in *International Workshop on Traffic Measurements for Cybersecurity 2023*, 2023.

[123] J. da Silva Damas, M. Graff, and P. A. Vixie, *Extension Mechanisms for DNS (EDNS(0))*, RFC 6891, Apr. 2013. DOI: 10.17487/RFC6891. [Online]. Available: https://www.rfc-editor.org/info/rfc6891.

[124] P. Schmitt, A. Edmundson, A. Mankin, and N. Feamster, "Oblivious dns: Practical privacy for dns queries: Published in popets 2019," in *Proceedings of the Applied Networking Research Workshop*, 2019, pp. 17–19.

[125] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. E. Hoffman, *Specification for DNS over Transport Layer Security (TLS)*, RFC 7858, May 2016. DOI: 10.17487/RFC7858. [Online]. Available: https://www.rfc-editor.org/info/rfc7858.

[126] S. Cheshire and D. Steinberg, *Zero configuration networking: The definitive guide*. " O'Reilly Media, Inc.", 2006.

[127] A. Inc., *Net services programming guide*, Accessed: 2023-07-24, 2013. [Online]. Available: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/NetServices/Introduction.html.

[128] S. Cheshire and M. Krochmal, *Multicast DNS*, RFC 6762, Feb. 2013. DOI: 10.17487/RFC6762. [Online]. Available: https://www.rfc-editor.org/info/rfc6762.

[129] A. Inc., *Resolving dns domain names*, Accessed: 2023-07-24, 2013. [Online]. Available: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/NetServices/Articles/domainnames.html#//apple_ref/doc/uid/TP40002460-SW1.

[130] M. Bartosh and R. Faas, *Essential Mac OS X Panther Server Administration: Integrating Mac OS X Server Into Heterogeneous Networks*. " O'Reilly Media, Inc.", 2005.

[131] S. Cheshire and M. Krochmal, *Special-Use Domain Names*, RFC 6761, Feb. 2013. DOI: 10.17487/RFC6761. [Online]. Available: https://www.rfc-editor.org/info/rfc6761.

[132] R. Dingledine, N. Mathewson, P. F. Syverson, *et al.*, "Tor: The second-generation onion router.," in *USENIX security symposium*, vol. 4, 2004, pp. 303–320.

[133] J. Appelbaum and A. Muffett, *The ".onion" Special-Use Domain Name*, RFC 7686, Oct. 2015. DOI: 10.17487/RFC7686. [Online]. Available: https://www.rfc-editor.org/info/rfc7686.

[134] P. Xia, H. Wang, Z. Yu, *et al.*, "Challenges in decentralized name management: The case of ens," in *Proceedings of the 22nd ACM Internet Measurement Conference*, 2022, pp. 65–82.

[135] B. Rajendran, G. Palaniappan, R. Dijesh, S. Sudarsan, *et al.*, "A universal domain name resolution service–need and challenges-study on blockchain based naming services," in *2022 IEEE Region 10 Symposium (TENSYMP)*, IEEE, 2022, pp. 1–6.

[136] P. Xia, H. Wang, Z. Yu, X. Liu, X. Luo, and G. Xu, "Ethereum name service: The good, the bad, and the ugly," *arXiv preprint arXiv:2104.05185*, 2021.

[137] M. Schanzenbach, C. Grothoff, and B. Fix, *The gnu name system*, 2020.

[138] M. Wachs, M. Schanzenbach, and C. Grothoff, "A censorship-resistant, privacy-enhancing and fully decentralized name system," in *Cryptology and Network Security: 13th International Conference, CANS 2014, Heraklion, Crete, Greece, October 22-24, 2014. Proceedings 13*, Springer, 2014, pp. 127–142.

[139] G. Huston, P. Koch, A. Durand, and W. ". Kumari, "Problem Statement for the Reservation of Special-Use Domain Names using RFC6761," Internet Engineering Task Force, Internet-Draft draft-adpkja-dnsop-special-names-problem-06, Sep. 2016, Work in Progress, 10 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-adpkja-dnsop-special-names-problem/06/.

[140] M. Fayed, L. Bauer, V. Giotsas, *et al.*, "The ties that un-bind: Decoupling ip from web services and sockets for robust addressing agility at cdn-scale," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 433–446.

Chapter 6A

The Five DNS Parameters: A Categorisation-Based Review

6A.1 Purpose of this Chapter

Before developing the proposed model, a categorisation-based taxonomy is required for the DNS. This chapter serves this purpose by exploring the five parameters of the DNS system. This is essential, as prioritising one or more DNS parameters could lead to trade-offs in others. This study contributes to the research proposition by improving the understanding of the five parameters to navigate the proposed model in Chapter 6B.

6A.2 Introduction

Over the past four decades, Domain Name System (DNS) [1], [2] transformation has been anchored by five guiding principles: speed, privacy, currency, scalability and flexibility. This chapter discusses how DNS advancement has navigated and, at times, challenged these five attributes, resulting in the system recognised today. The emphasis is on the 'why' behind the evolution, offering insights into the driving forces of this lookup and resolution system. While this chapter discusses the pivotal transformations that have shaped the DNS, it does not encompass an exhaustive account of every change to the system.

The discourse begins with the initial challenges inherent to HOSTS.TXT, subsequently tracing its progression to the contemporary DNS. This narrative highlights the transition from the foundational just-in-case lookup model to the refined just-in-time query model, detailing its advantages and the associated challenges. Along this trajectory, the role of the recursive resolver was introduced to address scalability challenges, although at the expense of flexibility.

The chapter continues its exploration of the flexibility introduced in the DNS lookup system, detailing the motivations and business imperatives behind it. The discussion begins with distributed service provisioning models, emphasising the utility of the Canonical Name (CNAME), specifically in scenarios requiring outsourcing name control. This segues into an exploration of name transformations using the Naming Authority Pointer (NAPTR) resource record to illustrate the complexities and motivations that influenced its development. The challenges of integrating multilingualism into the DNS through Internationalised Domain Names (IDNs) are then addressed. The chapter discusses an added facet of flexibility, exploring the customi-

sation of DNS responses. Within this context, the Client Subnet DNS extension emerges as a technique to enhance the precision of DNS replies, noting certain reservations concerning privacy.

The evolution of resource records is then explored, leading to newer iterations such as the SVC record. These service-centric records are intentionally designed to expedite the delivery of answers by reducing the need for multiple query round-trips, thereby enhancing both speed and currency. Shifting to privacy, the chapter analyses the intricate dynamics of obfuscation, highlighting the critical nature of separating client identities from query names, a cornerstone of ensuring privacy and confidentiality.

To conclude, an emerging theme is spotlighted as the DNS continues to be subjected to many pressures. This theme suggests a potential shift in DNS's foundational role, alluding to transitioning from a straightforward lookup system to a more intricate computational resource.

6A.3 The Domain Name System

The naming system emerged to address challenges facing conventional directory lookup. In this conventional system, one would use a lookup key, such as a name in a telephone directory, and retrieve a corresponding value, such as a telephone number. Similarly, the Domain Name System (DNS) was conceived to map domain names to IP addresses. While on the surface, this appears straightforward, actualising such a system in the evolving Internet poses unique challenges. Beyond the basic mapping of names to IP addresses, there are additional considerations: ensuring speed, upholding privacy, maintaining currency, handling the vast scale of user requests, and retaining the flexibility to adapt to emerging technological and service models. Thus, this chapter will employ the five central principles of speed, privacy, currency, scalability and flexibility as guiding parameters to discuss the motivations that influenced the evolution of the DNS.

Revisiting the antecedents of the DNS begins with the original HOSTS.TXT model. This host table file contained a human-readable label, a network address and protocol-specific information that the resource supported. One positive attribute of this original model was its privacy: each user received its own copy of the host mappings table, and all lookups were local, preventing external observers. However, the host table required frequent updates, leading to susceptibility to currency issues. Scalability also presented a challenge. The central server distributing the HOSTS.TXT became inundated with download requests, an issue that intensified as the HOSTS.TXT grew. Moreover, the static nature of the HOSTS.TXT made it highly inflexible. Adding new types of data to the evolving file structure proved limiting.

The solution to these problems was initially based on the provisioning model, leading to DNS as an outgrowth of HOSTS.TXT. Given the inconsistencies and outdated copies of the HOSTS.TXT, the original model underwent further refinement. This transition departed the static dissemination of a host file to a dynamic model encompassing two main components: an on-demand query resolution and a distributed data architecture. It shifted from a just-in-case approach, where users could access all potential host mappings for private lookup, to a just-in-time. In this model, users did not hold the entire host mapping information. Instead, they would query a specific resource and obtain the corresponding answer when required.

Adjusting to this model influenced various parameters, yielding improvements and introducing new challenges. The just-in-time approach enhanced currency with up-to-date mappings, but

lack of privacy emerged as a concern, with queries being directed to a logically centralised system where they could be observed. Additionally, speed and scalability posed a challenge; if numerous users simultaneously queried the central holder of domain names, the system risked being overwhelmed.

6A.3.1 Scalability

The Recurisve Resolver

Of these challenges, scalability was the most pressing issue. The initial evolution of this model saw the introduction of intermediaries, such as the caching recursive resolver. This was a pivotal development in the DNS, representing an amalgam of 'just-in-case' and 'just-in-time' approaches. When a query is made, if caching is employed across a distributed set of uncoordinated recursive resolvers, that information can be retained, anticipating that users nearby might pose the same query.

Another significant attribute of the caching recursive resolver pertains to privacy. Recursive resolvers are aware of users' activities by their inherent nature and function. When they query authoritative servers, they do so under their distinct identities. As a result, the individual's need for an answer is not broadcasted.

While the query model represented the initial approach to address the currency challenges, the caching recursive resolver addressed speed and scalability. However, there is a trade-off. While the caching mechanism ensures consistent responses, it maintains a static mode of naming. Therefore, benefits in speed and scalability come at the expense of flexibility. The following section introduces the successive generations of resource records that expanded the functionality and flexibility of the DNS.

6A.3.2 Flexibility

Outsource Control: CNAME

As the Internet evolved, scaling pressures prompted the need to outsource certain service provisions, and the prevailing DNS model was not fully equipped to accommodate this shift. The CNAME [2], [3] mechanism emerged as a solution, allowing the delegation of control of a specific name to a target entity without altering control of the entire DNS domain, a feature that the industry quickly adopted.

The CNAME mechanism provides the flexibility required for outsourcing, catering to scalability and specific business needs. For instance, if an external entity were managing a particular webpage and sought to control the domain name's resolution to align with their own infrastructure, the CNAME could be employed to transfer control of just that specific name without affecting the overall tree, delegation or zone.

Introducing CNAME was a pivotal step in the evolution of the DNS. It disrupted established models of delegation and administrative oversight, adapting to align closely with emergent business requirements. Individual domain names could now be managed with increased adaptability.

Akamai was among the first to commercialise the use of CNAME to effectively navigate the intricacies of DNS control in the new outsourced Internet model. This technique catered to the growing need to map individualised names onto infrastructures managed by specialised service

providers without altering the original names. Consider specialised service providers that host numerous entities. As described in Chapter 5A, these entities could utilise the CNAME record to establish an alias on the service provider's domain through outsourcing an indirection. This strategy enables the entities to preserve their identities while the queries are redirected to the resources managed by the hosting provider. Thus, the emphasis is on seamless service redirection. This process enhances the interoperability and flexibility of online services and operates in a manner that remains virtually invisible to the end user.

Publisher Control: NATPR

NAPTRs [4] gained importance for two main reasons, the first being largely mechanical. The objective of mapping E.164 telephone numbers into the DNS presented a unique challenge. Interestingly, the structure of telephone numbers aligns closely with the DNS's delegation hierarchy, although in the opposite sequence: telephone numbers read from left to right, whereas the DNS hierarchy is from right to left. Mapping each individual telephone number is possible, but integrating the entire telephony number system into the DNS's hierarchy is problematic due to this directional discrepancy. A solution to this challenge involved integrating query name translation directly into the DNS, bridging the gap between the queried name and the DNS record. NAPTRs facilitated this translation, enabling functions such as number reversals.

The second reason stemmed from a perceived need to transition from Universal Resource Locators (URLs) to Universal Resource Names (URNs). URLs inherently pinpoint a specific path to a file on a designated host. Once set, altering the path or host without causing disruptions becomes challenging. For instance, if an object were relocated to a different URL, all previously cached data associated with the original URL would become outdated and incorrect. The prevailing idea was for a method to name resources that were not tied to a specific location. One technique envisioned was translating the server's name to its location and embedding that within the DNS. This concept formed the foundation of NAPTRs. As the need for translation grew, the DNS was recognised as the optimal system to address this requirement.

Despite their potential, NAPTRs faltered due to the rapid decline in the use of E.164; the assumption that telephone numbers would remain prevalent proved incorrect. Consequently, the relevance of E.164 waned. Furthermore, the industry was reluctant to invest significant effort in URNs as location-independent alternatives to URLs. For transitioning resources between domain names, the preference leaned towards using CNAME with reduced caching.

While NAPTRs did not achieve widespread adoption, they serve as a pertinent example of the flexibility in the DNS, and the concept offers valuable insights. As described in Chapter 5A, one of the primary challenges in DNS management is its extensive caching. Once a domain name circulates, it becomes fixed due to widespread caching, complicating the process of rapidly altering mappings on demand. This inhibits the ability to generate customised responses, creating a significant challenge in relocating resources while maintaining a persistent and resolvable identifier.

The NAPTR record enables dynamic translations into designated target names. It establishes a separate layer that maintains the independence between the naming and the resolution system, along with its associated protocols and services. This ensures that changes in one system do not directly impact the other [5].

The NAPTR process involves the DNS server returning a regular expression pattern and a rewrite rule to the resolver. The resolver then applies this pattern to the original query name

and, following the rewrite rule, generates a completely new query. Notably, this technique provides considerable flexibility in rewriting rules as the translation lies within the publisher's control, not the user's, transforming the resolution process from semi-permanent to dynamic.

NAPTR differs in approach compared to CNAME. While CNAME translates one name into another in an iterative process, NAPTR provides a more nuanced approach. NAPTR offers an adaptable class of names that adhere to a specified pattern, mapping these to target names. While not necessarily identical to the original, these target names must conform to the provided pattern. Furthermore, NAPTR allows the query labels to undergo transformation according to a defined set of rules.

From an implementation perspective, NAPTRs were deemed too complex, with unnecessary levels of indirection and a multitude of rewrite rules. Additionally, the onus should be on the publication side rather than the user in identifier systems. In the context of the NAPTR implementation, the resolver handles the transformations, not the authoritative server.

Internationalised Domain Names

Building on the discussion of NAPTRs and their role in enhancing DNS flexibility through translation, a similar discussion on DNS's adaptability emerges in multilingualism. An imperative arose to accommodate domain names in various languages, not solely English. This introduced new challenges, particularly with the advent of Internationalised Domain Names (IDNs) [6], [7].

Despite the existence of formats such as Unicode Transformation Formats (UTF), especially UTF-16 and UTF-32, the DNS has a deep-rooted legacy in ASCII, and there is no universal standard to accommodate all characters. Furthermore, Unicode's flexibility compared to ASCII's strict character representation can lead to deceptive situations. For instance, certain Unicode strings can visually mimic their ASCII counterparts, posing risks of misleading users.

The onus of mappings was externalised to address these challenges instead of integrating the mappings directly into the DNS. This decision stemmed from the DNS's vast size and scale, which rendered changes challenging. Addressing the issues at the stub resolver level offered a simpler solution and represented the path of least resistance. Specifically, the client-side application, utilising the IDN resolver libraries, now handles these translations. In this system, the entirety of the DNS continues to operate on ASCII, while IDNs are encoded using Punycode.

This strategy relies on a crucial assumption that there is uniformity of Punycode implementations across different platforms. Various rules have been put in place to combat the issue of homoglyphs, which are visually similar characters represented differently in encoding. While IDNs are not thriving within the DNS ecosystem, their continued existence highlights the lack of a more efficient alternative.

This surfaces an important observation: the name queried in the DNS, and the name stored in the DNS system can differ and change over time. With CNAME records, the focus is on the point of control. NAPTR records introduce flexibility, accommodating transformations from left to right and right to left and later evolving to handle URN transformations. In the case of IDNs, the emphasis shifts to what users can see, not just what can be stored within the DNS. The core issue is whether to modify the lookup key and where to make those modifications. There are two main approaches: Internal DNS Modification (e.g., NAPTRs) involves carrying out transformations in a systematic fashion within the DNS system, leveraging the DNS in-

frastructure. Alternatively, external DNS Modification (e.g., IDNs) entails processing queries externally to achieve the desired results, leaving the DNS structure unaltered.

Customised Answers: EDNS Client Subnet

In conventional DNS operations, the identity of the client generally remains undisclosed. Due to privacy constraints, service providers such as Akamai often made suboptimal decisions in their DNS responses for content steering. However, there was a deliberate effort to poke privacy holes in the DNS to extrude parts of the user's information towards the authoritative servers. The primary motivation for this alteration was to enhance performance through optimal content steering, directing users to the nearest edge server. The Extension Mechanisms for DNS (EDNS0) [8] was introduced, and with it, the inclusion of the Client Subnet (ECS) [9] became a key technique for this solution.

In ECS, the client's IP address prefix is exposed and utilised as a part of the retrieval process. A DNS query goes beyond simply requesting a name in this evolved model. It now encompasses parameters dictated by EDNS0 extensions and the client's subnet (ECS) capability. This process requires a form of information trade whereby the client provides its assumed identity to receive an optimal resource proximate based on the server's estimation. Consequently, the previously invariant DNS response is now tailored to these additional query attributes. The objective is not to overload the label space with customisation code but to integrate this customisation into the query as adornments that can refine or limit the server's responses, ensuring each response is uniquely customised. The traditional DNS is characterised by its static nature and invariant responses. In contrast, with the advent of ECS, DNS has evolved to deliver responses tailored to specific client queries.

While ECS is widely used, it is not well tolerated due to the significant compromise it represents. It highlights a distinct tension as the traditional private DNS now divulges more about the client. However, this compromise introduces its own set of advantages. By revealing facets previously hidden by the DNS, the precision of flexible DNS responses benefits accuracy.

6A.3.3 Currency and Speed

More Efficient Rendezvous: SVCB

The **resource record** was conceived to cater to the distinct roles servers played, ensuring that each server could be pinpointed by its primary function and the array of services it provided. This differentiation was crucial, especially with the inception of early Internet services such as electronic mail, FTP, Telnet, USENET and the web. The resource records addressed this need, permitting the same name to have different mappings for diverse services, represented by types such as MX, A, AAAA, TXT and CNAME records. This approach was initially effective. However, applications began to require information from multiple record types, necessitating separate DNS queries for each type. As the demand for diverse record types grew, the system's flexibility came at the cost of speed, given the increase in multiple queries.

Speed cannot be compromised when balancing the demands for currency and flexibility. This is where the Service Binding record (SVCB) distinguishes itself [10]. As a broader interpretation of the resource record, it consolidates answers traditionally requiring iterative queries into one cohesive response. This approach reinstates flexibility in the zone publisher and enhances speed for the clients, allowing them to obtain a quick resolution with a single enriched query.

By integrating SVC attributes, queries can be customised, elevating the DNS's capability to provide rapid, tailored responses.

As described in Chapter 5A, the DNS evolved from a simple locator to a rendezvous broker with the introduction of SVCB records, enabling more precise, user-intention-driven service discovery. Instead of translating a name to an IP address, DNS now interprets the user's intent to determine the specific service they wish to access. This service is defined by a tuple of parameters such as port, protocol, and label, which precisely describe the service point. It is not just about objects or locations anymore; it is about guiding users to the services they seek based on their intentions. This concept is an abstraction beyond the basic model, making DNS the arbitrator of service access within a namespace.

6A.3.4 Privacy

Privacy was not the original goal of the DNS. It was a byproduct of the HOSTS.TXT system and was enhanced with the use of recursive resolvers. Over time, the Internet segment between the client and the chosen DNS resolver has been abused for passive observation and active interception. To address this, multiple strategies have been proposed. Initial responses focused on encrypting the DNS channel between the client and the resolver. Two notable methods are DNS over TLS (DoT) and DNS over HTTPS (DoH) [11], [12]. While both encrypt the communication between the client and the recursive resolver, they have distinct adoption paths. DoT's integration necessitated operating system-level changes, requiring platforms such as Android or iPhone to add support.

Given that users often leave the default settings intact, adoption remained low. DoT's limited success underscored the need to elevate such functionality to the browser level, leading to the development of DoH. DoH builds upon DoT by encapsulating the DNS query and response within an HTTP object. This shift allowed browsers to handle DNS queries directly, ensuring a private session between the browser and the resolver.

However, while DoT and DoH can hide the content of DNS exchanges, they cannot completely mask the identity of clients due to the visibility of the client's IP address. This challenge led to innovations that further obscured the correlation between the requester and the query. Simple query name obfuscation proved unfeasible due to constraints such as fixed field lengths.

A more sophisticated solution emerged with double-encoded MASQUE [13] and DNS obfuscation techniques using Oblivious DNS (oDOH) [14], both of which have been adopted by Apple's iCloud Private Relay [15]. In this approach, a user sends an encrypted, opaque bundle to the intermediary agent. While the agent can identify the sender based on the IP address, the specifics of the request remain concealed. The agent then forwards this bundle to the recursive resolver, acting as the user. Although the resolver can decrypt the bundle to respond to the request, it does not have the means to identify the original requester. The system's efficacy hinges on the assumption that the agent and the recursive resolver do not collude. This ensures that the user's identity and the query content remain mutually exclusive in knowledge, which is a promising direction for enhancing privacy and confidentiality.

6A.4 Conclusions

This chapter has examined the evolution of DNS in the context of broader Internet pressures and how these pressures are expressed in DNS, focusing on five key parameters: speed, currency, scalability, privacy, and flexibility.

To ask a question of a public resource, the querier information is disclosed to a certain extent. The richer the querier information, the more flexible and customised the DNS answer, but the greater the compromise of privacy. Providing uniform answers to all queries ensures speed. However, this speed often comes at the expense of flexibility and currency. Each DNS extension discussed in this chapter emphasises one of the five parameters as its primary benefit, but compromises are evident in the others. Therefore, the challenge is to optimise performance across all these aspects simultaneously.

The ongoing evolution of the DNS reflects our varying levels of concern regarding each of these parameters. Emphasising and prioritising one or more of these parameters today steers the focus in that direction. This, in turn, has increased costs elsewhere in the ecosystem, and thus, there is no perfect answer.

Chapter 6B

Proposed Model, Experiments, and Results: DNS and URL Computation

6B.1 Purpose of this Chapter

This chapter contributes to the research proposition by highlighting the challenges the current DNS needs to address by proposing a new model. Firstly, a categorisation-based review conducted in the previous chapter reveals the current limitations in the DNS ecosystem that are used as a foundational basis for this chapter. The proposed model, along with the systems and software architecture, is detailed herein. The chapter discusses the deployment methodology for implementing the proposed model. It also includes the research and validation methodology, the measurement testing and testing scenarios, and the results. As the tests were conducted globally, a breakdown of the geographic regions and a summary of the findings are also provided. The chapter concludes with future considerations for evolving this proposed model.

6B.2 Problem Statement

The Domain Name System (DNS) is divided into two parts: the namespace and the resolution protocol. The resolution protocol is a method for mapping those names onto an attribute the querier seeks, typically an IP address. This process is known as resolution. Figure 6B.1 depicts the conventional architecture of DNS resolution. In this model, the resolver initiates queries using the standard DNS protocol. These queries are forwarded to the recursive resolver. The key function of the recursive resolver is to maintain a broader knowledge base about the queried names compared to the stub resolver and to cache the answers it retrieves. This caching mechanism benefits all connected stub resolvers by ensuring performant response times. In essence, the recursive resolver acts as an intermediary, retrieving answers from authoritative servers on behalf of the stub resolvers. With benefits, the current DNS model introduces three challenges.

The first challenge pertains to the speed of DNS resolution. In an era when immediate responses are expected, DNS resolution can appear excessively time-consuming. The variability in interaction with the cache can lead to disparities in response times, resulting in a highly unpredictable user experience.

The second challenge is fundamentally tied to privacy concerns, specifically concerning the

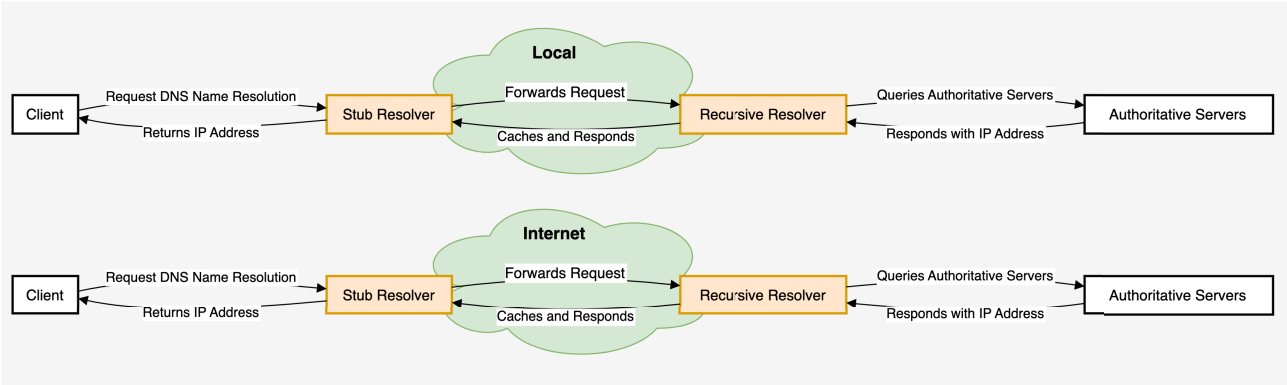


Figure 6B.1: Recursive Resolvers: Local versus Open.

Figure 6B.2: DNS Resolvers Utilization: Source Asia Pacific Network Information Center [16].

exposure of DNS queries. When these queries are analysed, they can reveal information about an individual's Internet usage, including both the identity of the user and the nature of their online activities [17]. This aspect of privacy becomes increasingly important as the evolution of DNS resolvers has shifted from local networks to being an Internet hop away. Prominent examples of this transition are Google's 8.8.8.8 and Cloudflare's 1.1.1.1, which are open recursive resolvers. Such changes in the resolver architecture have implications for the overall DNS architecture. They heighten the risks of passive third-party monitoring, as these queries, now traversing broader Internet spaces, are susceptible to interception, especially when open UDP DNS queries are employed. The APNIC DNS measurements [16] display the pattern of this shift, highlighting a trend where users are increasingly selecting open resolvers over local ISP-provided alternatives. As elaborated in Chapter 6A, there are three primary strategies proposed to enhance DNS privacy: DNS-over-TLS (DoT), DNS-over-QUIC (DoQ), and DNS-over-HTTPS (DoH). These techniques offer promising solutions to the privacy issues inherent in recursive resolvers situated more than a single hop away. However, it is crucial to acknowledge that while these methods may mitigate privacy concerns, the speed of DNS resolution is still an essential factor that must not be overlooked.

The third challenge is in the realm of flexibility. As discussed in Chapter 6A, prioritising more than one parameter of the DNS system can compromise others. The balance between privacy and flexibility is a good example: the richer the querier information through customisation, the greater the compromise of privacy. Flexibility through customisation of the namespace is an important parameter in content networking, allowing the DNS to resolve queries based on an approximation of the client's location.

Section 6B.3 presents the proposed DNS model to prioritise privacy, speed, and flexibility in the DNS resolution process, aiming to achieve a balance in the DNS ecosystem. Section 6B.4 outlines the implementation details of the proposed model. Section 6B.5 describes the setup process for testing the model's effectiveness. Section 6B.6 outlines the research methodology used to validate this effectiveness. Finally, Section 6B.7 examines the test scenarios and the resulting data to assess the proposed model's performance.

6B.3 Proposed Model

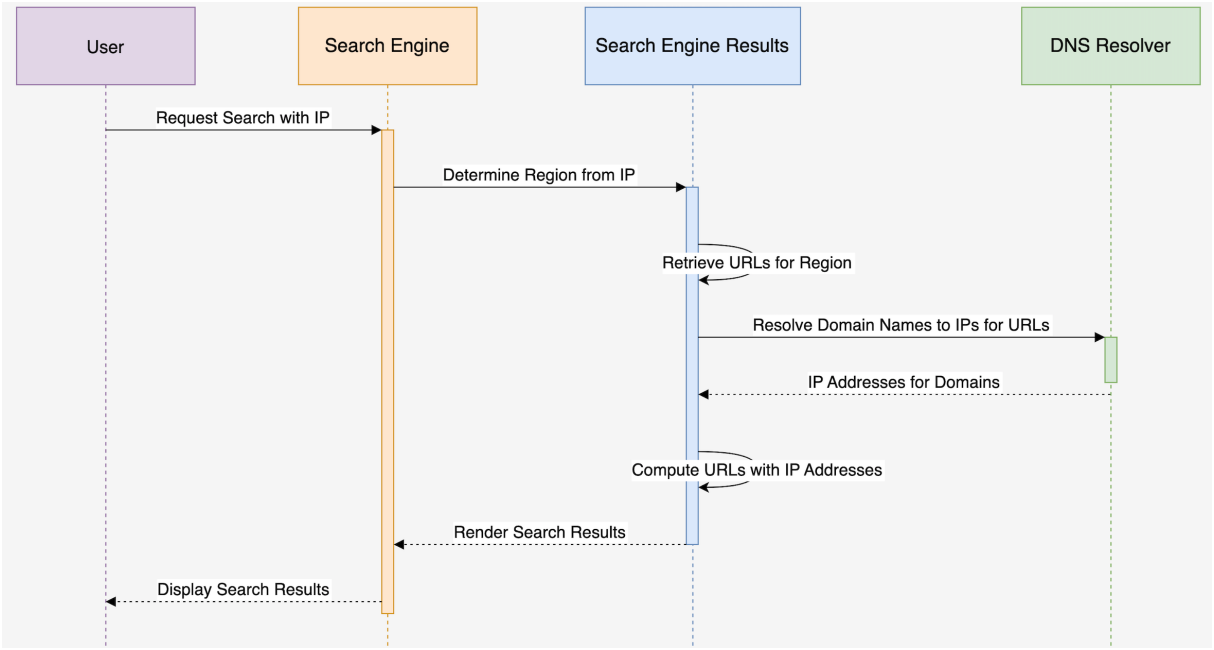


Figure 6B.3: Sequence Diagram: Component Interactions.

In traditional DNS resolution systems, the stub resolver in Figure 6B.2, residing within the client, initiates queries for the desired name. The proposed model shifts this stub resolver responsibility from the client to the server or application level to perform this function on behalf of the client. In his talk at DNS-OARC 39 on Day 1, Geoff Huston succinctly described how shifting responsibilities from the client to the server side of DNS resolution can enhance privacy and speed up responses [18]. This shift of name resolution has surfaced in the literature describing the concept as 'resolverless', where privacy and speed are also detailed as key motivators [18], [19]. This proposed model is an extension of the literature, employing computational techniques that offer three main advantages to address the original problem statement.

First, concerning the speed parameter, if caching is beneficial for performance, then prefetching could enhance it further. Consider the proposed sequence flow in Figure 6B.3. In this model, the client initiates a search query to the search engine. The search engine answer page response

includes precomputed results for URLs suggested on the search results page. This involves preresolving the fully qualified domain name to its corresponding IP address and applying a transformation technique to the URL. In contrast to the traditional method, every URL click requires name resolution for each user interaction. By eliminating this step, the new method allows users to reach the web resource directly, removing the additional DNS latency. Shifting this resolution responsibility from client to server could enhance user experience by reducing load times.

Second, in terms of privacy, this server-centric DNS resolution model minimises the risk of passive third-party monitoring, as the client's identity is not disclosed to the network or the DNS server. This structure preserves privacy and facilitates on-demand name resolution without client initiation. Hence, this approach is private and performant, addressing the critical challenges of enhancing privacy and speed in DNS resolution.

Finally, in addition to simultaneously prioritising privacy and speed, this model addresses the flexibility parameter. Based on client attributes performed at the application level, the application can refine or limit responses based on the client's information. This achieves flexibility by ensuring the response is tailored to the client's information while upholding privacy and performance.

Figure 6B.4: UDP Versus DOH and DOT: Source Asia Pacific Network Information Center [20].

Figure 6B.4, displaying the use of unencrypted queries in contrast to the adoption of DOH and DOT, identifies a significant gap in the adoption of privacy-hiding protocols but also indicates an ongoing shift towards the use of these protocols. Pending an industry consensus on a universally adopted technique, this model provides additional privacy support for current unencrypted query issues. The central focus of this model is to enhance privacy by shifting querying responsibilities to the server and facilitating the precomputation of URLs. This thesis aligns with the privacy-enhancing efforts discussed in the DOH and recent advances in query obfuscation with a double-encoded masque to increase privacy measures. This proposed model does not support the continuation of unencrypted queries and is designed to incorporate these privacy-hiding techniques in future implementations.

6B.4 Implementation of Proposed Solution

Figure 6B.5 illustrates the overarching system architecture for the proposed model's implementation. The architecture is further detailed in the subsequent subsections, each focusing on a key component of the system.

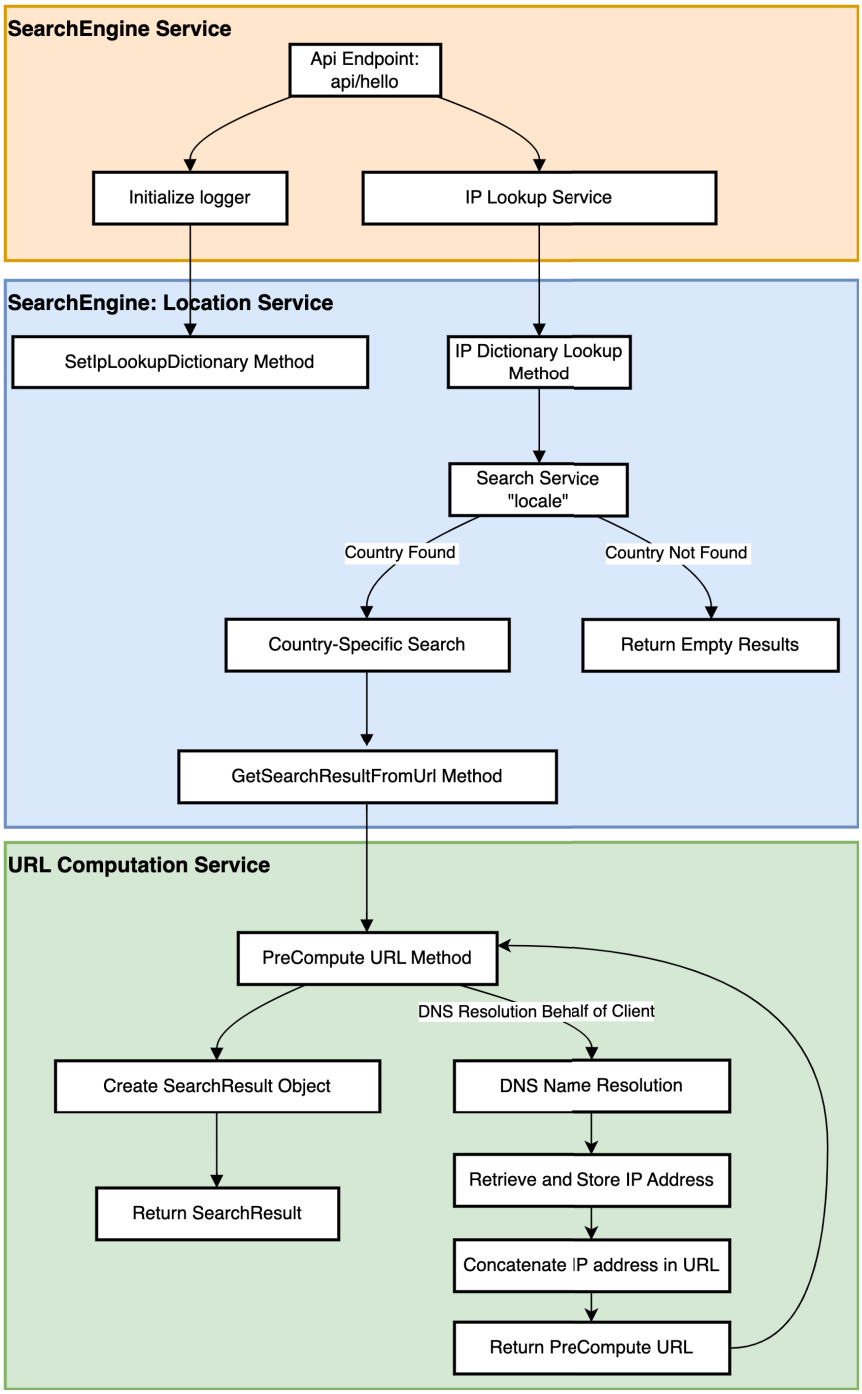


Figure 6B.5: Systems Architecture: Precompute URL.

6B.4.1 Search Engine Service

The SearchEngine Service handles incoming search requests at the API/hello endpoint, emulating the search engine page. It performs key operations, including extracting the client's IP

address from the request and storing it for geographic-based search. The following pseudocode outlines its functionality:

SearchEngine Service

```
Initialize api/hello endpoint
Initialize logger
Require: clientRequest is the incoming search request
  clientIP ← Extract IP address from clientRequest
  Store clientIP for IP Dictionary lookup
  Proceed with search operations using clientIP
```

6B.4.2 Locational Service: Search By Country Method

The client's IP address is passed to the SearchEngine: Location Service to determine the client's locale. The service tailors responses to the client's region, ensuring optimal resource access. The SearchByCountry method retrieves country-specific search results based on the country code. Its pseudocode is:

SearchByCountryMethod

```
Require: country is the country code
  results ← empty list of SearchResult
  urls ← RetrieveURLsForCountry(country)
  for each url in urls do
      searchResult ← GetSearchResultFromUrl(url, country)
      Add searchResult to results
  end for
  return results
```

6B.4.3 Locational Service: Retrieve URLs For Country Method

The RetrieveURLsForCountry method generates a list of specific URLs based on the country code. It ensures that clients from different regions receive different search results. Its pseudocode is:

RetrieveURLsForCountry

```
Require: country is the country code
  urlList ← new List of strings
  if country = "US" then
      urlList ← specific URLs for US
  else if country = "UK" then
      urlList ← specific URLs for UK
  else if country = "AU" then
      urlList ← specific URLs for AU
  end if
  return urlList
```

6B.4.4 URL Computational Service: Precompute URL Method

After obtaining the URL list, the URL computation method is invoked. It passes the list to the PreComputeUrl method, which performs IP address resolution. This involves creating a URI object, extracting the DNS name from the URL and querying authoritative servers to retrieve IP addresses. The resolved IP addresses are mapped to the original URL, and URL transformation is performed. This ensures that clients can access resources directly without DNS resolution delay. The following is the pseudocode for these methods:

GetSearchResultFromUrl

Require: *url*, *countryCode*
  searchResult ← new SearchResult
  searchResult.Url ← *url*
  searchResult.IpAddressList ← GetIpFromUrl(*url*)
  searchResult.Country ← *countryCode*
  searchResult.PreComputedUrl ← PreComputeUrl(*searchResult*)
  return *searchResult*

GetIpFromUrl

Require: *url*
  result ← empty list of strings
  uri ← new Uri(*url*)
  ipAddresses ← Dns.GetHostAddressesAsync(*uri*.Host)
  for each *ipAddress* in *ipAddresses* **do**
    Add *ipAddress*.ToString() to *result*
  end for
  return *result*

PreComputeUrl

Require: *searchResult*
  uri ← new Uri(*searchResult*.Url)
  return *uri*.Scheme + "://" + *searchResult*.IpAddressList[0] + "/"

6B.4.5 Parallelisation of DNS Resolution and URL Transformation

Single handling of a URL transformation would not demonstrate the computational aspects required for performance and latency improvements. This system is designed to deliver multiple URLs as part of the search engine answer page. To achieve this, the system design demonstrates URL retrieval and computation, typically when multiple URIs are retrieved, requiring concurrent operations. The system parallelises the resolution of multiple DNS queries for concurrent fetching of domain names. Once the IP address is retrieved, parallel URL computations are performed, which replace the domain name component for an IP address in the URL scheme, which is then presented as part of the search engine results page. Specific to the implementation, the **GetSearchResultsFromUrls** function enhances simultaneous operations by creating a task for each **GetSearchResultFromUrl**, awaiting their completion. This approach ensures that multiple URLs can be processed in parallel.

For DNS lookups, the **GetIpFromUrl** method asynchronously calls **Dns.GetHostAddressesAsync** to ensure that the resolution process of converting domain names into IP addresses for each URL is non-blocking and that the process serves multiple DNS resolutions simultaneously. Subsequently, each IP is sent using the URL computation method as described in the previous section. The **StopWatch** within the function is used as part of measurement testing to determine the computation of reach request during the runtime of the application. The following pseudocode describes this process:

Retrieval of Asynchronous Search Results

```
1: function GetSearchResultsFromUrls(urls, countryCode)
2:     if urls is null then
3:         return new list of SearchResult
4:     end if
5:     searchResults ← new list of SearchResult
6:     tasks ← new list of Task
7:     Start stopwatch
8:     for each url in urls do
9:         tasks.add(GetSearchResultFromUrl(url, countryCode))     ▷ asynchronously
10:    end for
11:    AwaitAll(tasks)                                  ▷ wait for all tasks to complete
12:    for each task in tasks do
13:        searchResults.add(task.result)
14:    end for
15:    Stop stopwatch and log time
16:    return searchResults
17: end function
18: function GetSearchResultFromUrl(url, countryCode)
19:    searchResult ← new SearchResult with url and countryCode
20:    searchResult.IpAddressList ← GetIpFromUrl(url)          ▷ asynchronously
21:    searchResult.PreComputedUrl ← PreComputeUrl(searchResult)
22:    return searchResult
23: end function
24: function GetIpFromUrl(url)
25:    result ← new list of strings
26:    Perform DNS lookup for url and add IP addresses to result     ▷ asynchronously
27:    return result
28: end function
```

6B.5 Deployment Methodology

This section describes the deployment methodology used to implement the proposed solution and the validation of the model within a global environment. The aim is to demonstrate how the implementation of the proposed solution, as described in the previous section, is instantiated in a real-life environment. The deployment is provisioned on the Microsoft Azure cloud platform [21], which has been selected for its extensive global network and computing coverage, especially across the United States, Europe, and Australia. In addition to the Microsoft Azure platform, two authoritative DNS servers were employed to replicate the mechanics of hosting DNS and the

redirections/translations that typically occur in practical content-delivery networking scenarios. This setup also aims to recreate the varied response times experienced on the public Internet. The environment and deployment process are designed to ensure the validation process is rigorously tested across diverse geographic regions and under varied network latencies.

6B.5.1 Environment Setup

A dedicated application has been deployed in each of these regions, with content replicated. This setup emulates a real-life use case where static and dynamic content is dispersed across a content delivery network, reflecting typical geographic distribution patterns. The primary deployment of the proposed solution is based in the United States, within the Microsoft Azure platform. This central deployment, along with the replicated content in other regions, is integral to demonstrating the validity of the model across different geographic areas. Additionally, the choice of Microsoft Azure was influenced by its comprehensive DNS coverage. The environment also includes public DNS names, which are used to replicate real-time latency and performance metrics from the public Internet, further validating the methodology.

6B.5.2 Azure Infrastructure Environment

As described earlier, validating the proposed DNS resolution model required deploying multiple Virtual Machines (VMs) across three regions: the United States, Europe and Australia. This deployment strategy was implemented to simulate a realistic global network environment. The deployment comprised four VMs, each serving a distinct function within this environment:

- **searchEngine-vm:** The primary VM hosting the implemented solution is located in the West US 2 and 3 regions of the Microsoft Azure platform. It functions as the central component of the test environment.
- **us-web-vm:** Deployed as a cluster of five VMs also situated in the West US 3 region. This VM validates the DNS model's effectiveness for users in the United States.
- **uk-web-vm:** Deployed as a cluster of five VMs in the UK South region. This VM mirrors the function of the us-web-vm, facilitating validation efforts for European users.
- **au-web-vm:** Deployed as a cluster of five VMs in the Australia East region. This VM extends the assessment to the Asia-Pacific region, performing the same validation functions as the US and UK VMs.

Notably, all regional web VMs us-web-vm, uk-web-vm, and au-web-vm maintain identical operating systems and size specifications to ensure uniformity in testing parameters and response behaviour. This intentional design mitigates any regional performance bias, as the primary VM, searchEngine-vm, also resides in the US.

6B.5.3 DNS Infrastructure

To simulate network latency and support the validation of the proposed model, two authoritative DNS name servers are utilised, and 11 test domains are hosted on Cloudflare. The regional VMs utilise this domain, appending a region-specific label — such as US, UK, or AU — to indicate the geographical area.

The DNS resolution process is initiated through iterative queries that reach Cloudflare's name servers, which perform the name resolution. Within these name servers, CNAME records have

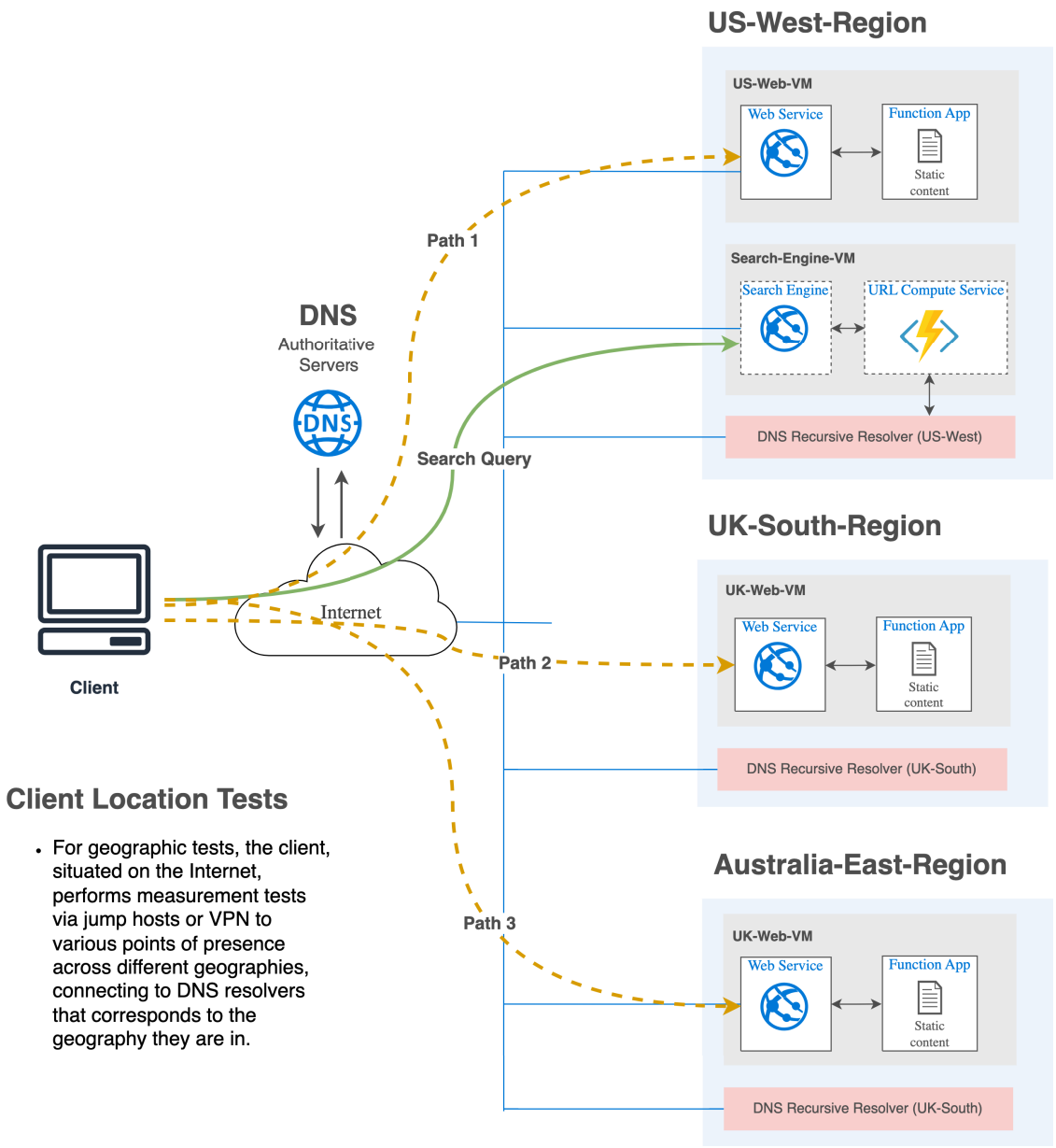


Figure 6B.6: Environment Setup.

been configured as a standard method for domain redirection and are associated with the regional VMs. These records allow a persistent identifier to redirect queries to the appropriate authoritative server for IP address resolution.

In typical content delivery network scenarios, several redirections may occur before an IP address is resolved. However, in this setup, the redirection is streamlined to a single level, directing to the domain experiment domains, which then resolves the name to the corresponding IP address of the regional VMs. The Time-To-Live (TTL) for the DNS cache to hold these records is 600 seconds.

This configuration replicates the latency experienced from the initial lookup to the final resolution. The effectiveness of this latency simulation is a key aspect that is evaluated in this environment.

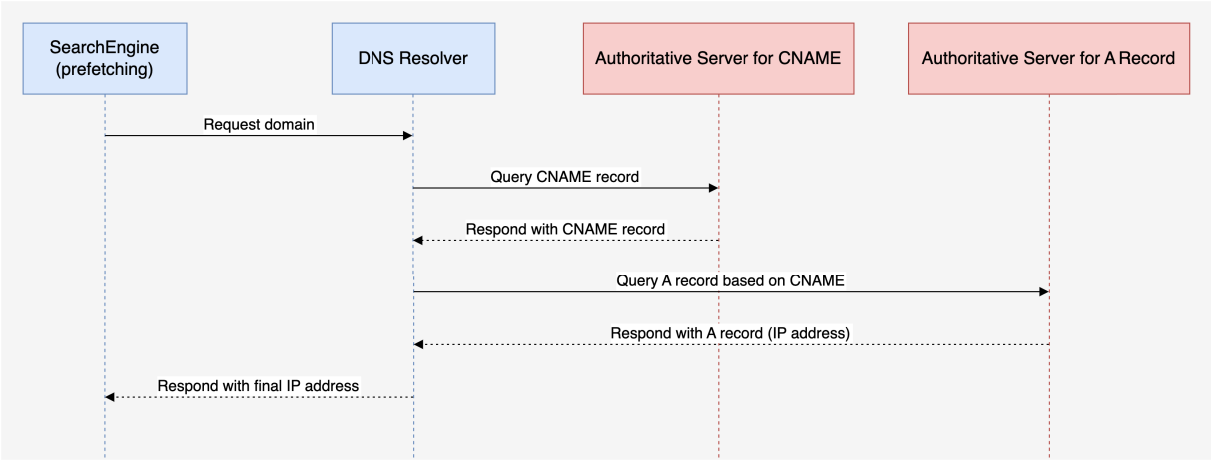


Figure 6B.7: Environment Setup: DNS Infrastructure.

6B.5.4 Client Setup

The client is set up to validate two scenarios. First, the client is deployed across all three regions to ensure that the response times and network conditions reflect real-world scenarios. This setup emulates the client's geographic proximity to the VMs in these regions, enabling the proposed solution to correctly select the appropriate VM based on the perceived location. Second, to test the effectiveness of sharing a common resolver of the proposed solution, the client is placed as VM form-factor in the Azure environment across three regions. For a practical setting where the client is associated with a local ISP DNS resolver, the client will be a physical machine-connected to the local ISP. However, it is important to note that the primary focus of this setup is on the relative improvement demonstrated by the proposed solution rather than on the absolute latency values.

6B.5.5 Public Accessibility and Application Behavior

The regional VMs are each assigned a public IP address to ensure they are globally reachable, which is key to the validation process. This deployment model emulates a real-world content delivery network, where users are directed to the geographically nearest application to latency. The IP addresses in this model are viewed as geographic site selectors.

Each regional VM hosts static content that specifies its geographic location. For instance, when a user accesses the VM located in the US, the static content delivered will include a message such as 'Hello from the US', confirming the regional source of the data. Similarly, the UK and Australia VMs have content unique to their regions, clearly indicating the origin of the content for the user.

As part of the application behaviour of the proposed solution, it integrates its search engine with its customised response mechanism. When a user performs a search, the search engine answer page includes URLs specific to the user's region. Upon clicking the press-commuted link, the user is automatically directed to the VM closest to their physical location.

6B.6 Comparative Methodology for Evaluating the Proposed Model

The proposed model's effectiveness is validated through user interactions with the search engine and the content delivery from regional VMs. Key metrics for validation include the time it takes for users to receive a response after clicking on a search result link and the accuracy of regional targeting. Live network traffic across three regions is used to simulate a real-world environment, enhancing the validation's accuracy. By analysing response times and verifying that user traffic is correctly directed to the appropriate regional VM, the model can be assessed in a live setting. With the deployment strategy outlined previously, we have established a comprehensive methodology to evaluate its impact. This methodology consists of two parts: an immediate performance assessment and an extensive analysis over a one-week business period.

6B.6.1 Considerations of Test Methodology

The methodology for validating the effectiveness of the proposed testing in live conditions creates additional complexity compared to controlled testing in a lab environment. This section considers the trade-offs between lab simulations and live measurements in real conditions.

A lab setup would involve creating a DNS infrastructure, including a stub resolver, recursive resolver and authoritative domains with a tree structure to reflect the hierarchy from the top-level domain to the specific child domain where testing occurs. The advantages of this approach include control over caching, which impacts test frequency, and the ability to automate the setup, thus allowing for a larger dataset. However, this method lacks measurement accuracy because it cannot accurately reproduce the Internet traffic conditions experienced during peak and off-peak periods, which influence overall latency. Controlling the DNS authoritative servers in this environment thus does not accurately reflect their response times for answering queries. A critical point is the behaviour of recursive resolvers, as each implementation – such as ISPs or open resolvers like Microsoft, Google and Cloudflare – behaves differently. Relying on lab simulations means the opportunity to observe the impact of these variables is missed.

Implementing the proposed model in live Internet conditions with live DNS systems offers advantages in terms of the accuracy of latency measurements for client requests, DNS responses and search engine results. However, the trade-off is that not all aspects of the test can be automated, and due to DNS answers being cached by optimised DNS recursive resolvers, the frequency of measurements per test is reduced. Despite this trade-off, prioritising accuracy over the simulated lab model is crucial. Given this consideration, the decision was made to conduct tests in live network conditions, obtaining measurements across Australia, the United States and Europe. This approach aims to obtain measurements that are as close to real-world as possible to strengthen the validity of the research proposition.

6B.6.2 User Behaviour and Test Methodology

This experiment must also consider user behaviour when interacting with the proposed model search engine page. Research suggests that while many users focus their interactions on the top three results, with the first result being clicked approximately 28% of the time [22], a variety of user patterns, including prolonged interaction with the search results page, should be within experimentation scope. Given the trend toward more personalised and contextually relevant search results based on individual user behaviour and history, it is reasonable that user patterns

involving prolonged interaction with the search engine could evolve. Therefore, this experiment will focus on five user clicks to assess both the traditional and proposed methods, reflecting a broadened scope of user behaviour.

6B.6.3 Data Collection Method

The latency measurement tests will be performed across three regions, the United States, Europe and Australia, to ensure an accurate dataset reflective of diverse network conditions, DNS and user behaviours. These metrics are recorded during peak and off-peak hours. There are two primary methods for obtaining data for this latency measurement test:

1. Postman, an API performance measurement tool, will capture the performance metrics such as overall latency, DNS lookup latency, TCP handshake, transfer start, and download times will be captured.
2. Stopwatch within the URL computation code path will be enabled during the SearchEngine service's runtime. This method analyses the per-request computation performed on the URL.

6B.6.4 Environment Setup

The testing environment, globally distributed in three Microsoft Azure regions as detailed in section 6B.5, ensures uniformity and reliability in the testing procedures. Live DNS authoritative servers and recursive resolvers are employed to accurately reflect live network conditions.

6B.6.5 Latency Measurement

The total latency measurement, L_{total}, encompasses the time from initiating the search to the user accessing a webpage from the search results. It comprises:

- **Search and Query Latency** ($L_{\text{search-query}}$): The time it takes a client to send a 'test' query to the search engine and receive results with URLs listed.
- **Initial Click Latency** ($L_{\text{click, first}}$): The time it takes for a user to click a URL in the search results and for the webpage to be displayed and accessible.

The initial load time (L_{initial}) is used to determine the first click immediately after search results are displayed and is calculated as follows:

$$L_{\text{initial}} = L_{\text{search-query}} + L_{\text{click, first}}$$

For subsequent clicks ($L_{\text{click, subsequent}}$), the latency is derived from the time taken to connect to the resource. The total latency for opening multiple web pages is calculated as follows:

$$L_{\text{total}} = L_{\text{initial}} + (n-1) \times L_{\text{click, subsequent}}$$

Note: This formula excludes the user decision time ($L_{\text{user-decision}}$); however, this is further discussed in the results section.

6B.6.6 Statistical Analysis - Comparison Between New and Old Method

1. **Performance Improvement Analysis**: To assess the immediate realisation of efficiency gains, the percentage improvement metric is employed. This metric quantifies the reduction in total latency achieved by the proposed method compared to the traditional approach. The formula is as follows:

$$\text{Improvement} = L_{\text{total, traditional}} - L_{\text{total, new}}$$

$$\text{Percentage Improvement} = \left(\frac{L_{\text{total, traditional}} - L_{\text{total, new}}}{L_{\text{total, traditional}}} \right) \times 100$$

2. **Mean Calculation Summarisation and Comparison between Methods:** The mean calculations provide a dual purpose. First, the calculation summarises the data distribution across the three collected regions, giving insight into the observed method's behaviour. Secondly, the mean calculation compares the mean latency measurements between the proposed new method and the old method, allowing for elimination and the level of impact the newly proposed method has made.

3. The **Average and Total Average Methods** involve calculating the average computation time and the total average DNS lookup time. These metrics assess the performance impact of the proposed concurrency approach to DNS and URL precomputing.

   The average computation time ($\overline{T_{comp}}$) is determined by:

$$\overline{T_{comp}} = \frac{1}{n} \sum_{i=1}^{n} T_{comp,i} \tag{6B.1}$$

   Where:

   - $T_{comp,i}$ represents the DNS and URL computation time for the i^{th} test,

   Versus the average DNS lookup time ($\overline{T_{DNS}}$), which is calculated by summing the DNS lookup times across the five user clicks within each test and then averaging these totals across all tests. The formula is below:

$$\overline{T_{DNS}} = \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{5} T_{DNS,ij} \right) \tag{6B.2}$$

   Where:

   - $T_{DNS,ij}$ represents the DNS lookup time for the j^{th} click within the i^{th} test, and the inner sum computes the total DNS lookup time for each subsequent test.

4. **Standard Error Calculation:** To quantify the variability of the averages, the error bars are added to the graphical representation in the test scenarios. The standard error for each metric is given by:

$$SE = \frac{\sigma}{\sqrt{n}} \tag{6B.3}$$

   This provides an estimate of the uncertainty associated between the mean values of $\overline{T_{comp}}$ and $\overline{T_{DNS}}$.

5. **Visualisation**: Graphs will represent both the performance improvements and the median latency comparisons.

6B.6.7 Data Collection Over Time

The data collection captures diverse network conditions in both off-peak and peak hours in the different geographies. Due to latency measurements being performed in real network decisions, the data collection time considers the optimised caching in the DNS resolvers. This could introduce systematic bias by maintaining copies of the DNS records of interest, potentially skewing the results. Furthermore, the proposed model is alternated between data centres in the United States West region to capture the effects of cloud resource sharing in the overall performance measurements. Statistically, these modifications to the data collection ensure reliability in data, mitigating the bias introduced by caching. This strategy aims to both support the variability influenced by the caching mechanisms in DNS resolvers and provide a dataset that accurately reflects the latency between traditional and new methods.

6B.6.8 Test Scenario Goals

The goal of the test scenarios is to demonstrate the improvement of web browsing by concurrently prefetching the DNS queries and precomputing the URL displayed on the search engine answer page in comparison to the traditional method, where DNS queries are initiated after clicking the URLs. The test scenario in Section 6B.7 provides geographic coverage of Australia, the United States and the United Kingdom, with each region focusing on different observed latency differences.

6B.7 Test Scenario and Results

6B.7.1 Test Scenario Description

To ensure measurement consistency, clients and the proposed model are associated with the common Microsoft DNS resolvers, except for the US client, which is associated with Google's resolver. This approach aims to isolate the impact of the DNS infrastructure performance, focusing on the computational improvements the proposed method yields. Testing involves the client querying the proposed system's search engine, where precomputation is performed, and recording the latency of the search results, denoted as $L_{\text{search-query}}$. The client also performs an identical search query using the traditional method, where latency is recorded without precomputation. For both the proposed and traditional methods, the user clicks the first URL, denoted as $L_{\text{click, first}}$, the latency of which is recorded. The initial latency, L_{initial}, is then calculated as the sum of these latencies using the following equation, referencing section 6B.6.5:

$$L_{\text{initial}} = L_{\text{search-query}} + L_{\text{click, first}}$$

Post L_{initial}, the test scenario entails user interaction with the search engine results, tracking up to five URL clicks. This scenario is performed for both the proposed and traditional methods to compare the total latencies from $L_{\text{search-query}}$ to the search result and the five subsequent clicks. The latency of subsequent clicks, denoted as $L_{\text{click, subsequent}}$, measures the time from the user's click to the connection with the web resource. The cumulative total latency for accessing multiple web resources is calculated using the following equation:

$$L_{\text{total}} = L_{\text{initial}} + (n-1) \times L_{\text{click, subsequent}}$$

The following subsections present the results for each region, Australia, the United States and the United Kingdom. Additionally, the data have been aggregated to calculate a global average of latencies, which is then compared between the proposed and the traditional model.

6B.7.2 Australia East Region

Architecture of Test Scenario

Figure 6B.8 illustrates the architecture of the differences between the proposed and traditional methods. The proposed method depicted on the left represents the communication between the two regions. The client connects to the proposed solution directly over the Internet using a physical laptop and a VM in the cloud. The client employs a variety of recursive resolvers, which include local ISP, Azure's local resolver, Google's 8.8.8.8, Cloudlare's 1.1.1.1, OpenDNS 208.67.222.222 and Quad9's 9.9.9.9. The diversity of client devices and resolvers is crucial for capturing application performance under various conditions.

In this experiment, the client accesses the search engine in the US region and enters a query string. The engine inspects the IP packet and infers from the geographic dictionary contained in the system that the client is from the Australian region. The search engine customises the results for all Australian-based URLs associated with the query and populates the URLs into the list. The search engine integrates with its DNS subsystem and concurrently queries all five domain names associated with the query and resolves them to IP addresses. Each IP address is precomputed into the URL, and the search answer page is returned to the client. In this model, the client accesses all five resources directly without querying the DNS to the Australia East web application.

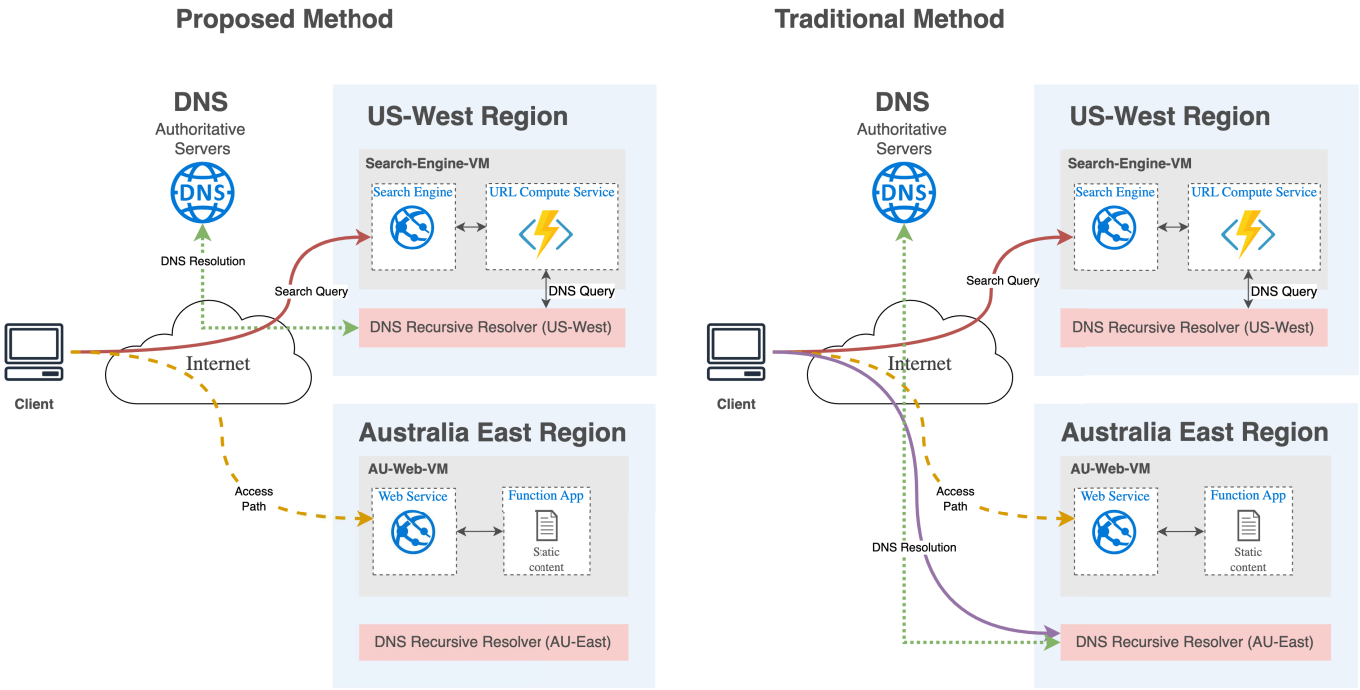


Figure 6B.8: Test Scenario: Australia East Region.

The traditional method similarly queries the search engine, but the computational latency is removed in this test and the answers are returned with the unaltered URL string for the user to click on. The user then clicks on each click, which triggers a DNS query per URL clicked.

Results of Test Scenario

To interpret Figure 6B.9, the following equation is applied to the Australian dataset:

$$\overline{L}_{\text{total, AU}} = \overline{L}_{\text{search-query, AU}} + \overline{L}_{\text{click, first, AU}} + (n-1) \times \overline{L}_{\text{click, subsequent, AU}}$$

The line graph illustrates the cumulative average of the latency as the number of user clicks increases, providing insight into the latency and response time differences between the two methods. The line that shows 'new method' is the cumulative total latency over a series of user clicks, depicted in dark orange. The royal blue also presents the cumulative total latency for the traditional model across a series of clicks. This directly compares the two methods, highlighting the improvements demonstrated by the proposed model. As illustrated in this graph, as the number of clicks increases, the total latency improves significantly compared to the old method. Initially, at the first user click, both methods exhibit comparable performance levels. This similarity can be attributed to the new method's initial computational overhead as it concurrently prefetches DNS queries and integrates the computed IP addresses into the URL strings. However, from click two onwards the new method's improvements become more pronounced, demonstrating the benefit of computing the answers prior to user clicks, illustrated in Table 6B.1. Notably, at click five, there is a 65.13% improvement.

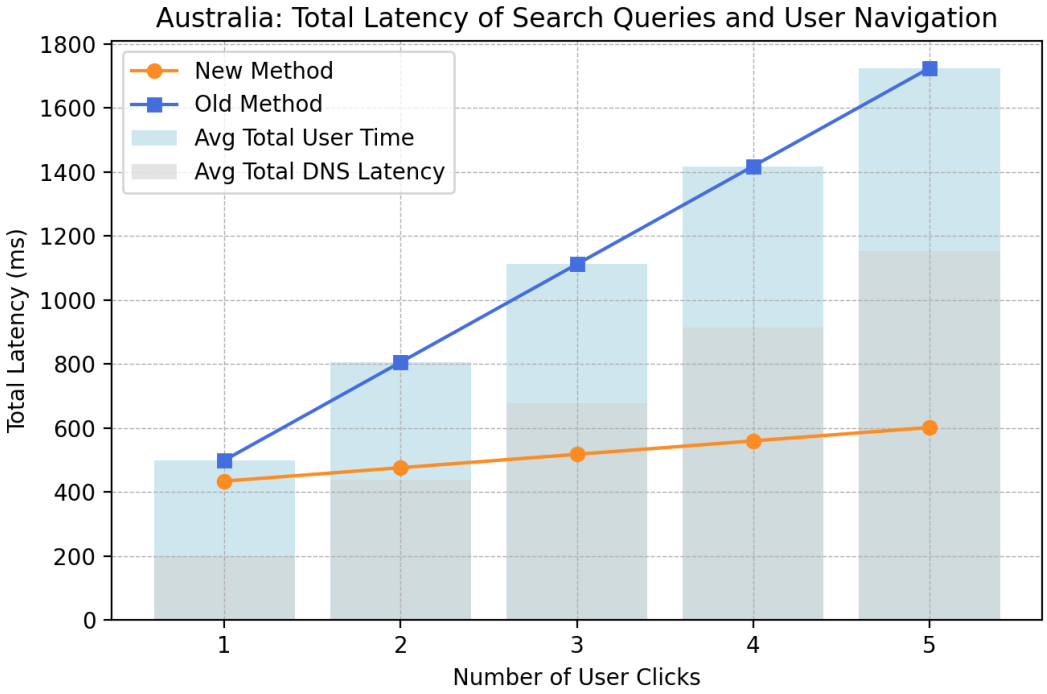


Figure 6B.9: Australia: Total Latency of Search Queries and User Navigation.

Number of User Clicks	Improvement (ms)	Percentage Improvement (%)
1	65.28	13.09
2	329.63	40.94
3	593.98	53.45
4	858.33	60.55
5	1122.68	65.13

Table 6B.1: Millisecond and Percentage Improvement of Proposed Model Over Traditional Model for the Australian Dataset.

In Figure 6B.10, the bar chart compares the average computation time ($\overline{T_{comp}}$) of the new method, calculated as the mean of computation times across all tests, as detailed in Section 6B.6.5:

$$\overline{T_{comp}} = \frac{1}{n}\sum_{i=1}^{n} T_{comp,i} \tag{6B.4}$$

The bar representing $\overline{T_{comp}}$, illustrated in dark orange, corresponds to the initial concurrent DNS queries and URL computation performed by the system to replace DNS queries with precomputed URL strings. In contrast, the bar illustrated in royal blue represents $\overline{T_{DNS}}$ as the average total DNS lookup time, encompassing the latencies from five clicks within each test, as detailed in Section 6B.6.5:

$$\overline{T_{DNS}} = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{5} T_{DNS,ij}\right) \tag{6B.5}$$

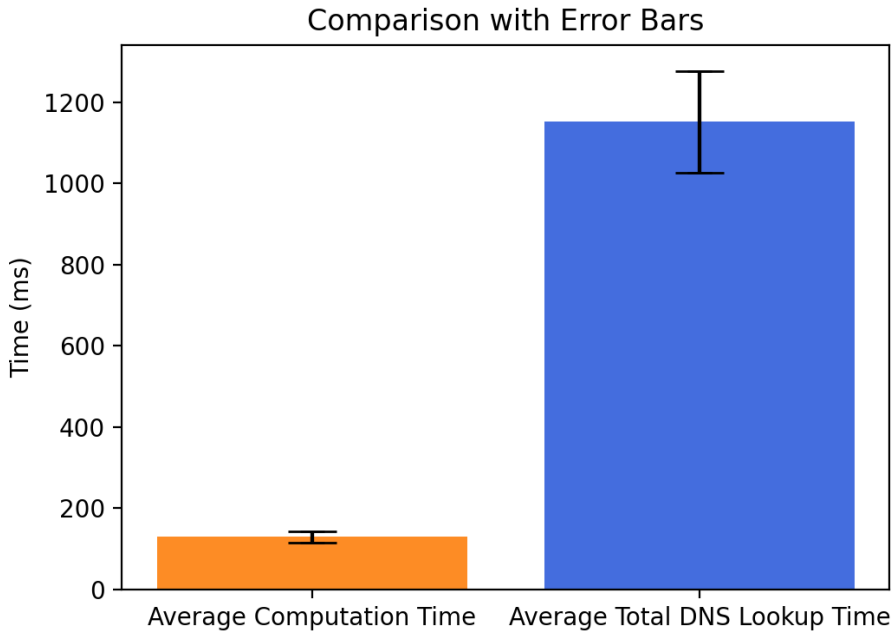


Figure 6B.10: Australia: Average Computation Time of Proposed Model versus Average Total DNS Lookup Time.

where $T_{DNS,ij}$ represents the DNS lookup time for the j^{th} click within the i^{th} test. The average total DNS lookup time is calculated by first summing the DNS lookup times for all five clicks within each test and then averaging these sums across all tests.

This comparison outlines that $\overline{T_{comp}}$, the concurrent method for DNS prefetching and precomputation of URLs at the search query phase, significantly impacts the efficiency, as evidenced by a comparison with the traditional method's total average DNS lookup time ($\overline{T_{DNS}}$). The proposed model demonstrated an 88.84% reduction in latency compared to the traditional DNS lookup method. The error bars in this analysis validated the results by displaying the standard error for $\overline{T_{comp}}$ and $\overline{T_{DNS}}$ illustrate confidence in the observed improvements via the computational method.

6B.7.3 United States West Region

Overview of Test Scenario

Figure 6B.11 illustrates the architecture of the differences between the proposed and traditional models. In contrast to Europe and Australia, the US communication path remains within the United States. The client connects to the proposed solution directly over the Internet using a physical laptop and a VM in the cloud. The client employs a variety of recursive resolvers, which include local ISP, Azure's local resolver, Google's 8.8.8.8, Cloudlare's 1.1.1.1, OpenDNS 208.67.222.222 and Quad9's 9.9.9.9. The diversity of client devices and resolvers is crucial for capturing application performance under various conditions.

In the proposed model, illustrated in Figure 6B.11, the client accesses the search engine in the US region and enters a query string. The engine inspects the IP packet and infers from the geographic dictionary contained in the system that the client is from the United States region. The search engine customises the results for all United States-based URLs associated with the query and populates the URLS into the list. The search engine integrates with its

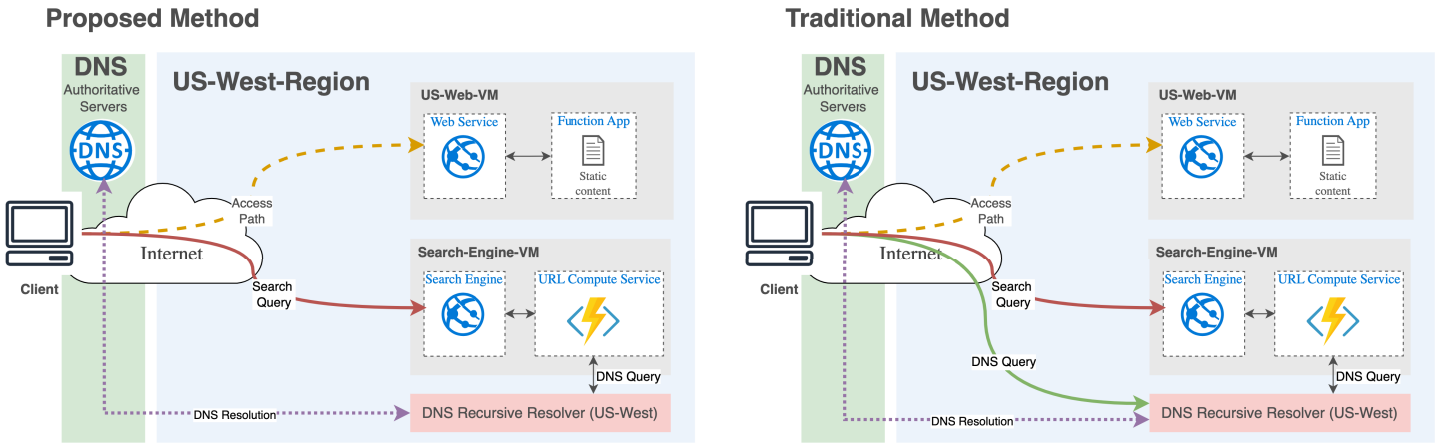


Figure 6B.11: Test Scenario: United States West Region.

DNS subsystem and concurrently queries all five domain names associated with the query and resolves them to IP addresses. Each IP address is precomputed into the URL, and the search answer page is returned to the client. In this model, the client accesses all five resources directly without querying the DNS to the United States West region.

The traditional method similarly queries the search engine, but the computational latency is removed in this test, and the answers are returned with the unaltered URL string for the user to click on. The user then clicks on each string, which triggers a DNS query per URL clicked.

Results of Test Scenario

To interpret Figure 6B.9, the following equation is applied to the United States dataset:

$$\overline{L}_{\text{total, US}} = \overline{L}_{\text{search-query, US}} + \overline{L}_{\text{click, first, US}} + (n-1) \times \overline{L}_{\text{click, subsequent, US}}$$

The line graph illustrates the cumulative average of the latency as the number of user clicks increases, providing insight into the latency and response time differences between the two methods. The line that shows 'new method' is the cumulative total latency over a series of user clicks, depicted in dark orange. The royal blue also presents the cumulative total latency for the old method across a series of clicks. This directly compares the two methods, highlighting the improvements demonstrated by the proposed model. As shown in the graph, the initial click shows the old method performing better due to the new method's initial computational overhead, which involves prefetching DNS queries and integrating computed IP addresses into URL strings. However, from click three onwards, the total latency for the proposed model shows improvement compared to the old method, demonstrating the benefit of computing the answers prior to user clicks, illustrated in Table 6B.2. Notably, at click five, there is a 20.24% improvement.

In Figure 6B.13, the bar chart compares the average computation time ($\overline{T_{comp}}$) of the new method, calculated as the mean of computation times across all tests, as detailed in Section 6B.6.5:

$$\overline{T_{comp}} = \frac{1}{n} \sum_{i=1}^{n} T_{comp,i} \tag{6B.6}$$

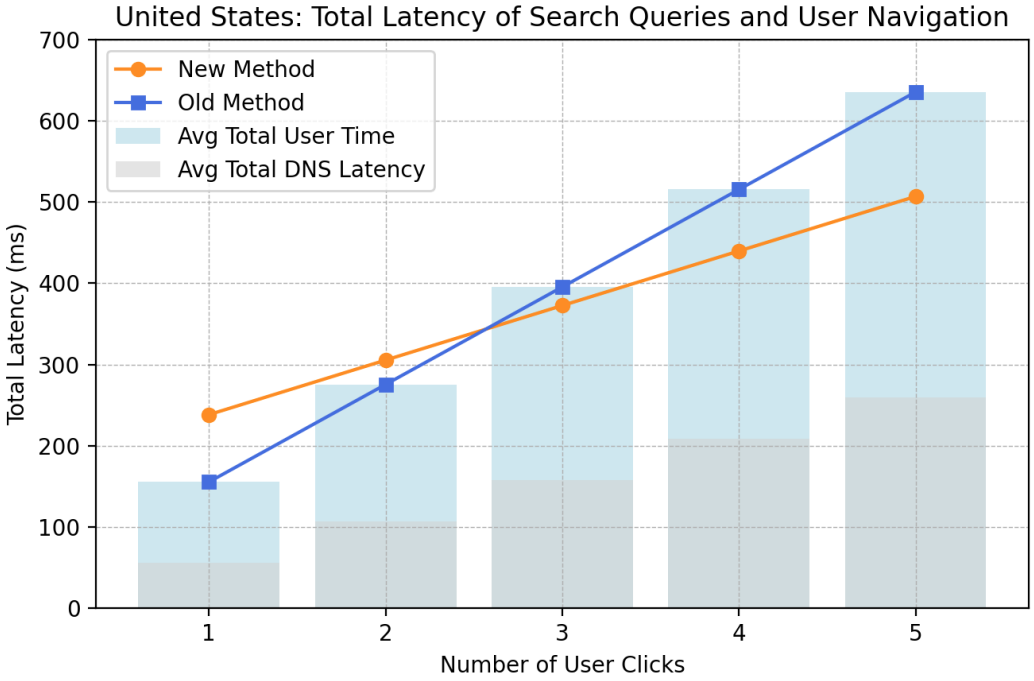


Figure 6B.12: United States: Total Latency of Search Queries and User Navigation.

Number of User Clicks	Improvement (ms)	Percentage Improvement (%)
1	-82.85	-53.35
2	-29.99	-10.89
3	22.86	5.78
4	75.72	14.69
5	128.58	20.24

Table 6B.2: Improvement and Percentage Improvement of Proposed Model Over Traditional Model for the United States Dataset.

The bar representing $\overline{T_{comp}}$, illustrated in dark orange, corresponds to the initial concurrent DNS queries and URL computation performed by the system to replace DNS queries with precomputed URL strings. In contrast, the bar illustrated in royal blue represents $\overline{T_{DNS}}$ as the average total DNS lookup time, encompassing the latencies from five clicks within each test, as detailed in Section 6B.6.5:

$$\overline{T_{DNS}} = \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{5} T_{DNS,ij} \right) \tag{6B.7}$$

where $T_{DNS,ij}$ represents the DNS lookup time for the j^{th} click within the i^{th} test. The average total DNS lookup time is calculated by first summing the DNS lookup times for all five clicks within each test and then averaging these sums across all tests.

This comparison outlines that $\overline{T_{comp}}$, the concurrent method for DNS prefetching and precomputation of URLs at the search query phase, significantly impacts the efficiency, as evidenced by its comparison with the traditional method's total average DNS lookup time ($\overline{T_{DNS}}$). The proposed model demonstrated a 46.43% reduction in latency compared to the traditional DNS

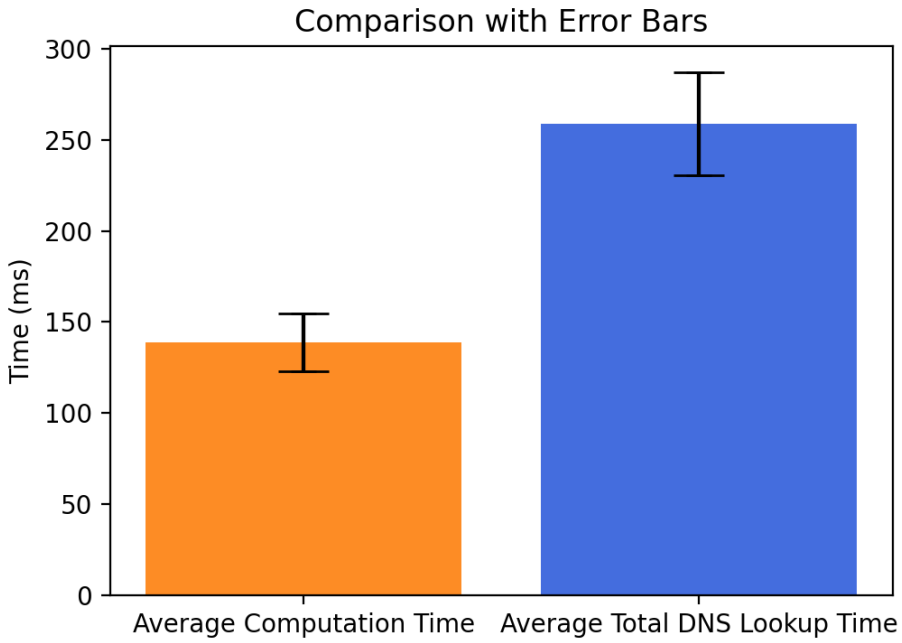


Figure 6B.13: United States: Average Computation Time of Proposed Model versus Average Total DNS Lookup Time.

lookup method. The error bars in this analysis validated the results by displaying the standard error for $\overline{T_{comp}}$ and $\overline{T_{DNS}}$ illustrate confidence in the observed improvements via the computational method.

6B.7.4 United Kingdom Region

Architecture of Test Scenario

Figure 6B.14 illustrates the architecture of the differences between the proposed and traditional models. The new method, on the left, represents the communication between the two regions. The client connects to the proposed solution directly over the Internet using a physical laptop and a VM in the cloud. The client employs a variety of recursive resolvers, which include local ISP, Azure's local resolver, Google's 8.8.8.8, Cloudlare's 1.1.1.1, OpenDNS 208.67.222.222 and Quad9's 9.9.9.9. The diversity of client devices and resolvers is crucial for capturing application performance under various conditions.

In this experiment, the client accesses the search engine in the US region and enters a query string. The engine inspects the IP packet and infers from the geographic dictionary contained in the system that the client is from the UK region. The search engine customises the results for the client locale and populates the URLs into the list. The search engine integrates with its DNS subsystem, concurrently queries all five domain names associated with the query and resolves them to IP addresses. Each IP address is precomputed into the URL, and the search answer page is returned to the client. In this model, the client accesses all five resources directly without querying the DNS of the webpage located in the United Kingdom's South region.

The traditional model involves the client querying the search engine using a qualification string. The search engine then returns an answer page with five URLs. In this test scenario, the computation latency is removed and the answers are provided with the original URL string for

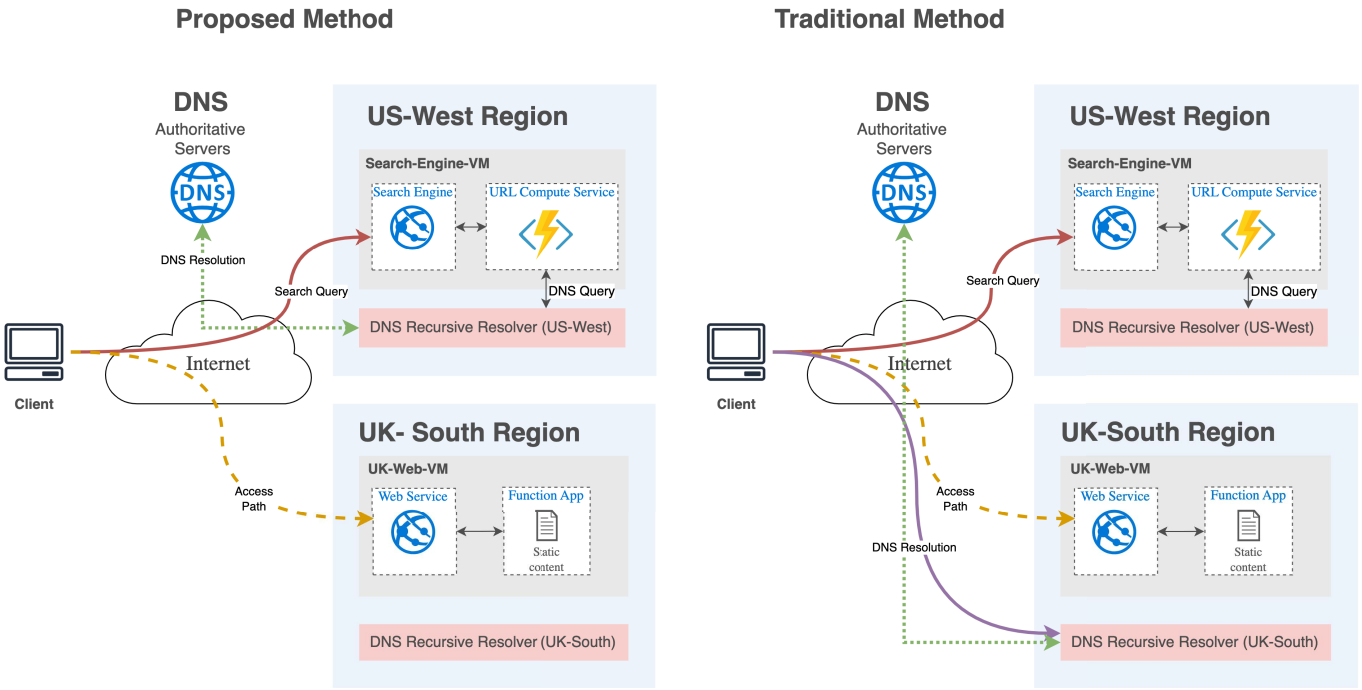


Figure 6B.14: Test Scenario: United Kingdom South Region.

the user to click. Each click by the user triggers a DNS query for the URL clicked.

Results of Test Scenario

To interpret Figure 6B.15, the following equation is applied to the United Kingdom dataset:

$$\overline{L}_{\text{total, UK}} = \overline{L}_{\text{search-query, UK}} + \overline{L}_{\text{click, first, UK}} + (n-1) \times \overline{L}_{\text{click, subsequent, UK}}$$

The line graph illustrates how the latency evolves as the number of clicks increases, providing insight into the latency and response time differences between the two methods. The line showing 'new method' is the cumulative total latency over a series of user clicks, depicted in dark orange. The royal blue also presents the cumulative total latency for the "old method" across a series of clicks. This directly compares the two methods, highlighting the improvements demonstrated by the proposed model. Mirroring the trends observed in Australia and the United States, this graph illustrates that as the number of clicks increases, the total latency improves significantly compared to the old method. Similarly to the United States, the initial click shows the old method performing better, but by the second click, performance levels become comparable. This is due to the proposed model's initial computational overhead from concurrently prefetching DNS queries and integrating computed IP addresses into URL strings. However, from the third click onward the new method's improvements become more pronounced, demonstrating the benefits of computing the answers prior to user clicks, as illustrated in Table 6B.3. Notably, at click five, there is a 21.67% improvement.

In Figure 6B.16, the bar chart compares the average computation time ($\overline{T_{comp}}$) of the new method, calculated as the mean of computation times across all tests, as detailed in Section 6B.6.5:

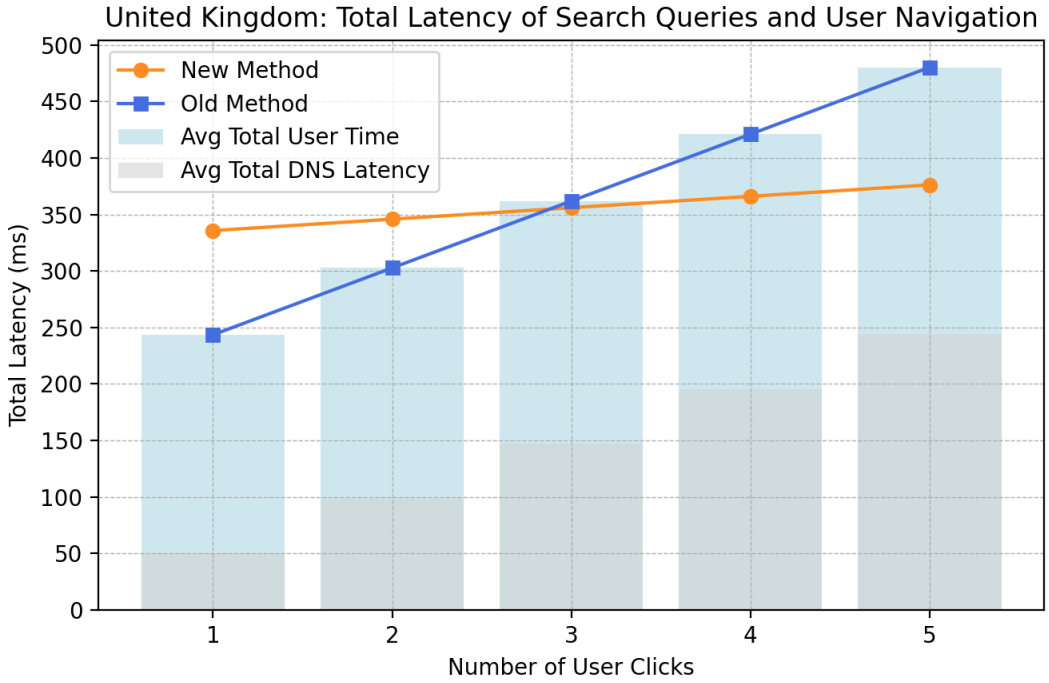


Figure 6B.15: United Kingdom: Total Latency of Search Queries and User Navigation.

Number of User Clicks	Improvement (ms)	Percentage Improvement (%)
1	-92.23	-37.87
2	-43.16	-14.26
3	5.91	1.63
4	54.98	13.06
5	104.05	21.67

Table 6B.3: Improvement and Percentage Improvement of Proposed Model Over Traditional Model for the European Dataset.

$$\overline{T_{comp}} = \frac{1}{n} \sum_{i=1}^{n} T_{comp,i} \tag{6B.8}$$

The bar representing $\overline{T_{comp}}$, illustrated in dark orange, corresponds to the initial concurrent DNS queries and URL computation performed by the system to replace DNS queries with precomputed URL strings. In contrast, the bar illustrated in royal blue represents $\overline{T_{DNS}}$ as the average total DNS lookup time, encompassing the latencies from five clicks within each test, as detailed in Section 6B.6.5:

$$\overline{T_{DNS}} = \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{5} T_{DNS,ij} \right) \tag{6B.9}$$

where $T_{DNS,ij}$ represents the DNS lookup time for the j^{th} click within the i^{th} test. The average total DNS lookup time is calculated by first summing the DNS lookup times for all five clicks within each test and then averaging these sums across all tests.

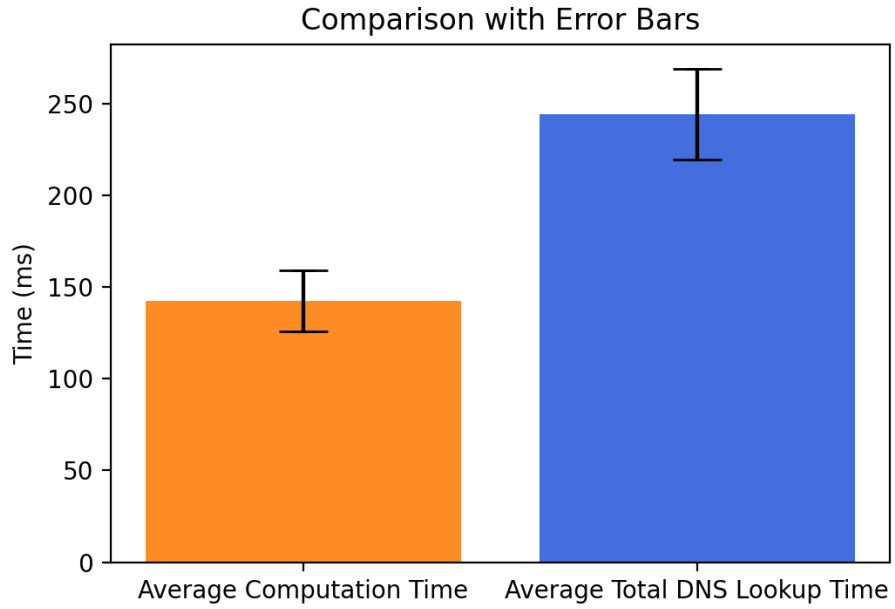


Figure 6B.16: United Kingdom: Average Computation Time of Proposed Model versus Average Total DNS Lookup Time.

Similarly to the United States, this comparison outlines that $\overline{T_{comp}}$, the concurrent method for DNS prefetching and precomputation of URLs at the search query phase, significantly impacts the efficiency, as evidenced by its comparison with the traditional model's total average DNS lookup time ($\overline{T_{DNS}}$). The proposed model demonstrated a 41.66% reduction in latency compared to the traditional DNS lookup method. The error bars in this analysis validated the results by displaying the standard error for $\overline{T_{comp}}$ and $\overline{T_{DNS}}$ illustrate confidence in the observed improvements via the computational method.

6B.7.5 Proposed Model Evaluation and Future Considerations

Summary of Test Scenario

As illustrated in Figure 6B.17, a significant distinction exists between the proposed and traditional models. The proposed model demonstrates one-time computation, enhancing system flexibility without compromising speed and privacy, thus supporting the original research proposition. Figure 6B.17 illustrates the global average of the results, highlighting a significant improvement of 46.55% by the fifth click with the concurrent prefetching method. The average computation reveals that one-time computation to address DNS queries demonstrates a 74.37% efficiency versus the subsequent iterative DNS lookups. Observation indicates that the proposed method typically requires one or two highly latent DNS lookups in a series of five clicks to outperform the traditional model. Across the three regions, Australia's results were more pronounced, most likely due to the recursive resolver performing several round trips to resolve the queries. Observations in the United States and the United Kingdom, in particular, may suggest that recursive resolvers appear more optimised, possibly due to periodically refreshing of resource records, contributing to faster response times.

A consistent observation in the northern hemisphere was that the new method demonstrated performance improvements from the third user click onward. The traditional model may seem more effective initially, but the performance differs from subsequent clicks. The observed con-

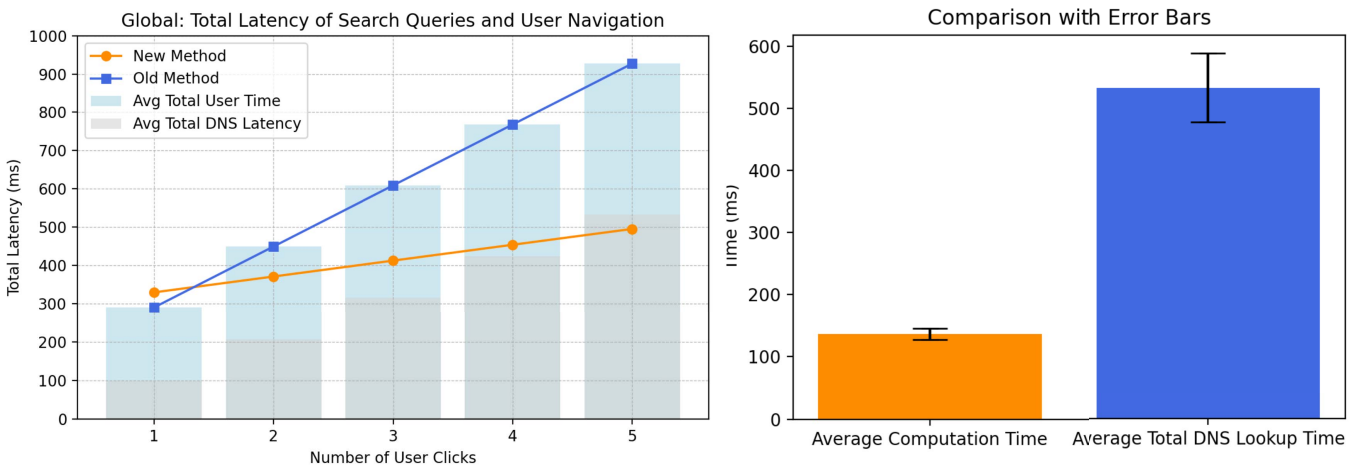


Figure 6B.17: Global Average: Comparison Between the Proposed and Traditional Method.

Number of User Clicks	Improvement (ms)	Percentage Improvement (%)
1	-39.49	-13.61
2	78.27	17.42
3	196.04	32.21
4	313.80	40.87
5	431.57	46.55

Table 6B.4: Improvement and Percentage Improvement of Proposed Model Over Traditional Model for the Global Dataset

sistency across all three regions was statistically significant through the continuous capture of latency measurements and subsequent click measurements. This consistency was maintained when measuring across various network conditions through the time of day and when altering the proposed method location between two data centres to assess the level of contributing latency beyond just network factors, including computational processing latency. The low variance in latency measurements observed from the proposed method and the homogeneity of results across different environmental conditions indicate that the dataset is statistically sufficient from the experiment conducted.

An important factor is considering the decision time in user behaviour, where performance is more about user-perceived latency than actual latency. A study found that, on average, it takes 14.6 seconds for a user on the Google search answer page to decide and click their first link, with 50% of users deciding within 9 seconds [23]. This finding highlights that the computational model in the proposed method is likely to complete its process before a user decides which favours the proposed model. With user behaviour as another variable, consider the following:

- **Computation Latency** L_{comp}: The time it takes for prefetching DNS and precomputing URL.
- **User Decision Time** $L_{\text{user-decision}}$: The period during which the user decides which link to click.
- **Click latency** L_{click}: The time from when the user decides to click a link to when the content is rendered.

Suppose the computation (L_{comp}) completes before the user's decision time ($L_{\text{user-decision}}$). Therefore, this method becomes invisible to the user. The perceived latency from the user's perspective after they decide to click is represented by:

$$T_{\text{perceived}} = L_{\text{click}}$$

This revised perceived latency equation emphasises that the only latency contributing to the user's perception of system responsiveness is L_{click}, the time it takes to access a resource after they decide to click, provided that:

$$L_{\text{comp}} \leq L_{\text{user-decision}}$$

Although the experiment primarily focused on performance metrics, it acknowledges user behaviour as a crucial factor, as emphasised above. To highlight this, Figure 6B.18 removes the search query latency data from the traditional and proposed methods to shift the focus directly to the user's perceived performance. In the proposed method, users clicking a link is sent directly to the resource, demonstrating a significant reduction in latency, unlike the traditional method, which involves a series of round-trip times to the DNS before reaching the resource. This distinction is reflected in the perceived latency shown in Figure 6B.18.

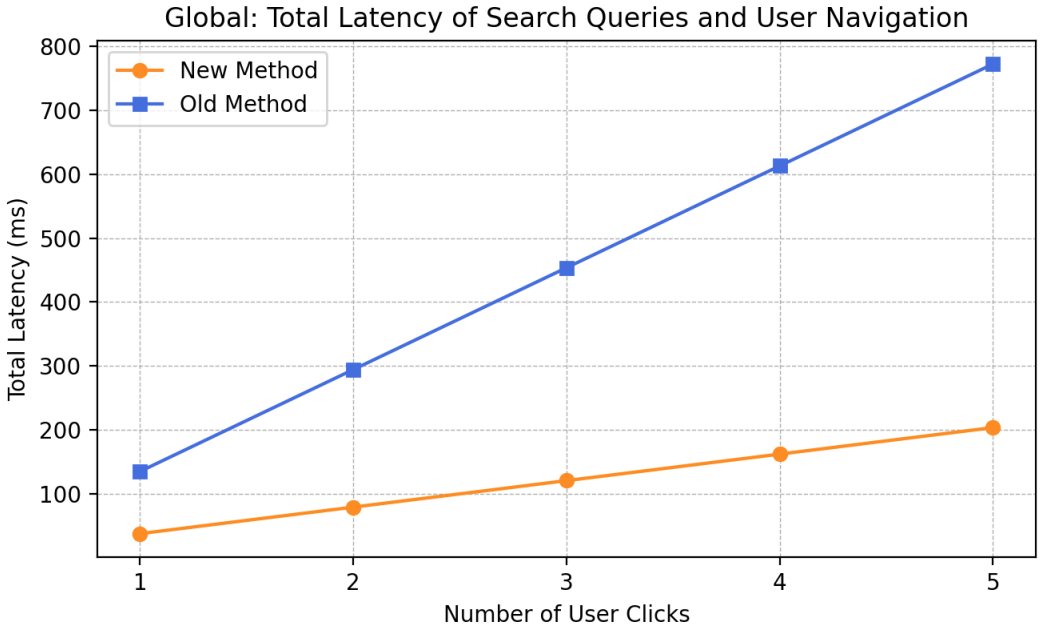


Figure 6B.18: Global: User Perception Latency.

Limitations, Considerations and Improvements of Proposed Model as Future Work

A key contribution of this study was the significant shift in identifier resolution from client to server, placing the onus on the server to provide computational for speed, privacy, and custom responses. This foundational work, depicted in Figure 6B.19 and 6B.20, highlights the layers needed post-research, focusing on handling virtual hosting, as well as SSL and SNI processing. Trust is also an important issue in ensuring responses are validated. To continue the customisation journey, there is an opportunity to enrich queries with attributes as part of

the application-level session. Considering these factors, the areas listed below are identified for further exploration to enhance the proposed model for future work.

Custom Responses and Flexibility: In the realm of content networking, the widespread data and content replication introduces a significant geographical challenge in computing the most accurate answers in terms of proximity to user queries. To ensure users are directed to the optimal result for requested material, it is crucial to preemptively identify and select the most geographically proximate and relevant answer from among all the possible answers. However, there is the probability of precomputing an answer that could be deemed sub-optimal, degrading the user experience.

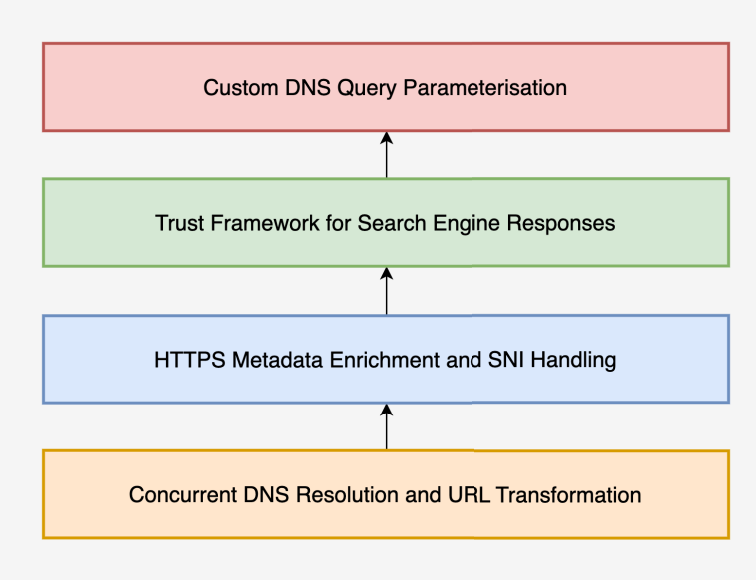


Figure 6B.19: A Layered Approach: Evolving the Proposed Model.

In the proposed model, a dictionary within the search engine service maintains users' current location to ensure answers are customised to the user's geographic region. The continued research builds upon this approach via an adaptive model wherein client/server interactions can invoke real-time re-calculations of URLs to direct the users to the most appropriate resource based on the client's attributes, which could improve response accuracy and use context to meet the user's preferences.

Trust and Validation: In this model, the emphasis on trust is significant. In an example using DNSSEC, if the system can push the DNS answer, it can concurrently push the DNSSEC validation chain, with the Key Signing Key (KSK) playing a crucial role in validating the responses [24]. This implies not only that it can deliver the resolution of a name but also the reasons why it can be trusted. With DNSSEC, the importance shifts to validation capability — it matters less how the information was learned, what is crucial is the ability to validate it. The point of whether the recursive resolver can validate it becomes irrelevant; trust centres on the individual or application level, enabling client applications to directly perform validation. In this proposed mode, all responses must be signed; otherwise, the trust framework collapses.

Addressing Virtual Hosting, SNI, and TLS/SSL Challenges: The value of a transition to server-side precomputation was the main argument of this chapter. Computing the URL or sending the DNS are options. However, this study clarifies that using an IP address in the

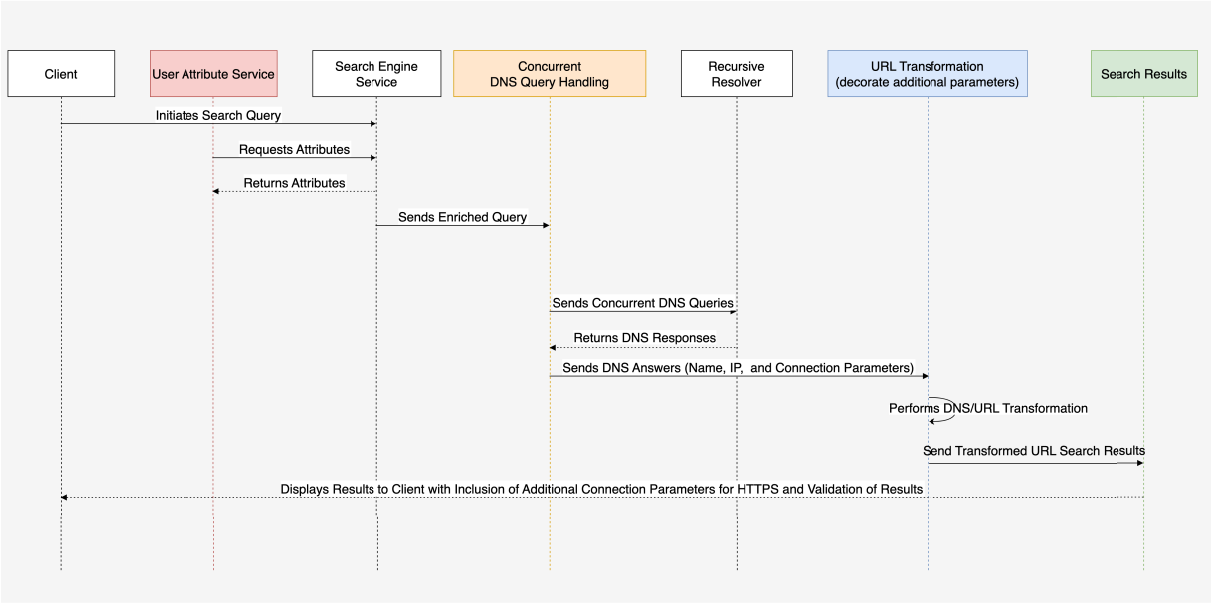


Figure 6B.20: Sequence Diagram: Proposed Future Model.

URL was an experiment to demonstrate parallelisation from server-side DNS fetching. This introduces three issues to be addressed:

- In the context of virtual hosting, web servers are consolidated under a host that maps to the same IP address, and the 'Host' header in the HTTP request is used as an application-level distinguisher. As part of precomputation, the URL could be associated with metadata to include this field.
- With SSL/TLS certificates issued for a domain name rather than an IP address, a mismatch between the domain in the HTTP request and the certificate's domain can cause the certification validation to fail, as the certificate is expected to contain the domain name queried. A possible solution could be to associate metadata with the URL, which requires the inclusion of the domain name even though the answer is precomputed.
- In HTTPS, the SNI field allows multiple domain names to be mapped to a single IP address as a method to distinguish between virtual hosts. As in the previous discussion regarding the use of domain names, the SNI field must also contain the domain name. This potential solution could also ensure this domain name is integrated into the SNI field.

Architectural Improvements with DoH: To further refine user experiences in the application-level session, adopting DNS over HTTPS (DoH) could offer a path [18]. DoH allows DNS results to be treated as web objects, which means, when combined with the Server Push feature intrinsic to HTTP/2 and HTTP/3 [25], that servers can push answers to clients dynamically. This integration reduces latency in displaying search answer page results by overcoming traditional search query latency constraints. Furthermore, the answers could be dynamically updated as the user scrolls through the search answer page, allowing for the immediate display of updated content without additional computational time during the search query phase. This approach researches the interplay between DOH as web objects, prefetching DNS answers and dynamically rendering the results invisible to users.

References for Chapters 6A and 6B

[1] *Domain names - concepts and facilities*, RFC 1034, Nov. 1987. DOI: 10.17487/RFC1034. [Online]. Available: https://www.rfc-editor.org/info/rfc1034.

[2] *Domain names - implementation and specification*, RFC 1035, Nov. 1987. DOI: 10.17487/RFC1035. [Online]. Available: https://www.rfc-editor.org/info/rfc1035.

[3] R. Elz and R. Bush, *Clarifications to the DNS Specification*, RFC 2181, Jul. 1997. DOI: 10.17487/RFC2181. [Online]. Available: https://www.rfc-editor.org/info/rfc2181.

[4] M. H. Mealling and D. R. Daniel, *The Naming Authority Pointer (NAPTR) DNS Resource Record*, RFC 2915, Sep. 2000. DOI: 10.17487/RFC2915. [Online]. Available: https://www.rfc-editor.org/info/rfc2915.

[5] D. R. Daniel and M. H. Mealling, *Resolution of Uniform Resource Identifiers using the Domain Name System*, RFC 2168, Jun. 1997. DOI: 10.17487/RFC2168. [Online]. Available: https://www.rfc-editor.org/info/rfc2168.

[6] D. J. C. Klensin, *Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework*, RFC 5890, Aug. 2010. DOI: 10.17487/RFC5890. [Online]. Available: https://www.rfc-editor.org/info/rfc5890.

[7] D. J. C. Klensin, *Internationalized Domain Names in Applications (IDNA): Protocol*, RFC 5891, Aug. 2010. DOI: 10.17487/RFC5891. [Online]. Available: https://www.rfc-editor.org/info/rfc5891.

[8] J. da Silva Damas, M. Graff, and P. A. Vixie, *Extension Mechanisms for DNS (EDNS(0))*, RFC 6891, Apr. 2013. DOI: 10.17487/RFC6891. [Online]. Available: https://www.rfc-editor.org/info/rfc6891.

[9] C. Contavalli, W. van der Gaast, D. C. Lawrence, and W. ". Kumari, "Client Subnet in DNS Queries," Tech. Rep. 7871, May 2016, 30 pp. DOI: 10.17487/RFC7871. [Online]. Available: https://www.rfc-editor.org/info/rfc7871.

[10] B. M. Schwartz, M. Bishop, and E. Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs)," Internet Engineering Task Force, Internet-Draft draft-ietf-dnsop-svcb-https-12, Mar. 2023, Work in Progress, 61 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-dnsop-svcb-https/12/.

[11] P. E. Hoffman and P. McManus, *DNS Queries over HTTPS (DoH)*, RFC 8484, Oct. 2018. DOI: 10.17487/RFC8484. [Online]. Available: https://www.rfc-editor.org/info/rfc8484.

[12] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. E. Hoffman, *Specification for DNS over Transport Layer Security (TLS)*, RFC 7858, May 2016. DOI: 10.17487/RFC7858. [Online]. Available: https://www.rfc-editor.org/info/rfc7858.

[13] D. Schinazi, "The MASQUE Protocol," Internet Engineering Task Force, Internet-Draft draft-schinazi-masque-02, Jan. 2020, Work in Progress, 6 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-schinazi-masque/02/.

[14] E. Kinnear, P. McManus, T. Pauly, T. Verma, and C. A. Wood, *Oblivious DNS over HTTPS*, RFC 9230, Jun. 2022. DOI: 10.17487/RFC9230. [Online]. Available: https://www.rfc-editor.org/info/rfc9230.

[15] *Icloud private relay overview*, Dec. 2021. [Online]. Available: https://www.apple.com/privacy/docs/iCloud_Private_Relay_Overview_Dec2021.PDF.

[16] APNIC Labs, *Use of dns resolvers for world*, https://stats.labs.apnic.net/rvrs, Accessed: 2023-01-23, 2023.

[17] D. Herrmann, C. Banse, and H. Federrath, "Behavior-based tracking: Exploiting characteristic patterns in dns traffic," *Computers & Security*, vol. 39, pp. 17–33, 2013.

[18] DNS-OARC, *OARC 39 Day 1 Part 1*, https://youtu.be/C7SvGT0k26w, Accessed: 2023-11-15, 2023.

[19] T. Mueller and E. Sy, "Enhanced performance and privacy via resolver-less dns," in *2021 International Conference on Information Networking (ICOIN)*, IEEE, 2021, pp. 243–248.

[20] APNIC Labs, *Cloudflare open recursive resolver dns query profile for australia*, https://stats.labs.apnic.net/edns/AU, Accessed: 24-Jan-2024, 2024.

[21] Microsoft Corporation, *What is iaas?* https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-iaas/, Accessed: 2023-xx-xx, Sep. 2023.

[22] "Google first page clicks: Over 25% of people click the first result." Accessed: 2024-01-22. (2020), [Online]. Available: https://www.searchenginejournal.com/google-first-page-clicks/374516/.

[23] Backlinko, *How people use google search (new user behavior study)*, https://backlinko.com/google-user-behavior, Accessed: March 8, 2024, 2022.

[24] P. E. Hoffman, *DNS Security Extensions (DNSSEC)*, RFC 9364, Feb. 2023. DOI: 10.17487/RFC9364. [Online]. Available: https://www.rfc-editor.org/info/rfc9364.

[25] M. Belshe, R. Peon, and M. Thomson, *Hypertext Transfer Protocol Version 2 (HTTP/2)*, RFC 7540, May 2015. DOI: 10.17487/RFC7540. [Online]. Available: https://www.rfc-editor.org/info/rfc7540.

Part III

Future Work and Conclusions

Chapter 7A

Future Work A: Identifier Systems in 5G Core Networks

7A.1 Purpose of this Chapter

In addition to the previous chapter's contribution to server-side computational DNS for URL resolution, this chapter explores promising future applications of the identifier system introduced in Chapter 5B. In addition, this work presents a collaboration with Intel on cloud-native service mesh infrastructure for 5G Core Network Functions (NFs). This resulted in an IEEE Access publication, titled *"Cloud-Native Service Mesh Readiness for 5G and Beyond"*, proposing identifier system requirements for inclusion in future 3GPP guidelines. This chapter selectively covers the aspects specific to this research discussion and demonstrates the continuation and application of this research as future work. For details of this publication, see Appendix D.

7A.2 Identifier Systems in Cloud-Native 5G

7A.2.1 Background and Challenges in Cloud-Native 5G

5G architecture represents a significant step forward in network technology. It takes advantage of the vast capacity of the cloud, the dynamic control of Software Defined Networking (SDN), the strategic resource management of Network Function Virtualisation (NFV), and the modular strategy intrinsic to microservices architecture [1]. Chapter 2 discussed the concept of a microservice – a self-contained service that fulfils a specific business function that can be independently developed, maintained and scaled [2]. The 5G architecture can offer scalable services and flexible deployment models by adopting this microservices-based approach combined with the aforementioned technologies.

Irrespective of the innovative design, the 5G Core Network Functions (NFs) faced unique challenges in the initial stages of 5G development, specifically in reliably and efficiently managing service requests. Direct communication between NFs led to significant communication overhead and increased latencies. Furthermore, the system grappled with a lack of load-balancing capabilities, the absence of a unified approach to observability and a complex operational landscape, amplifying the difficulties of achieving seamless network connectivity. The Service Communication Proxy (SCP), introduced as part of 3GPP's release 16 [3], [4], was created as a key component of the 5G Service Based Architecture (SBA) to address the challenges introduced

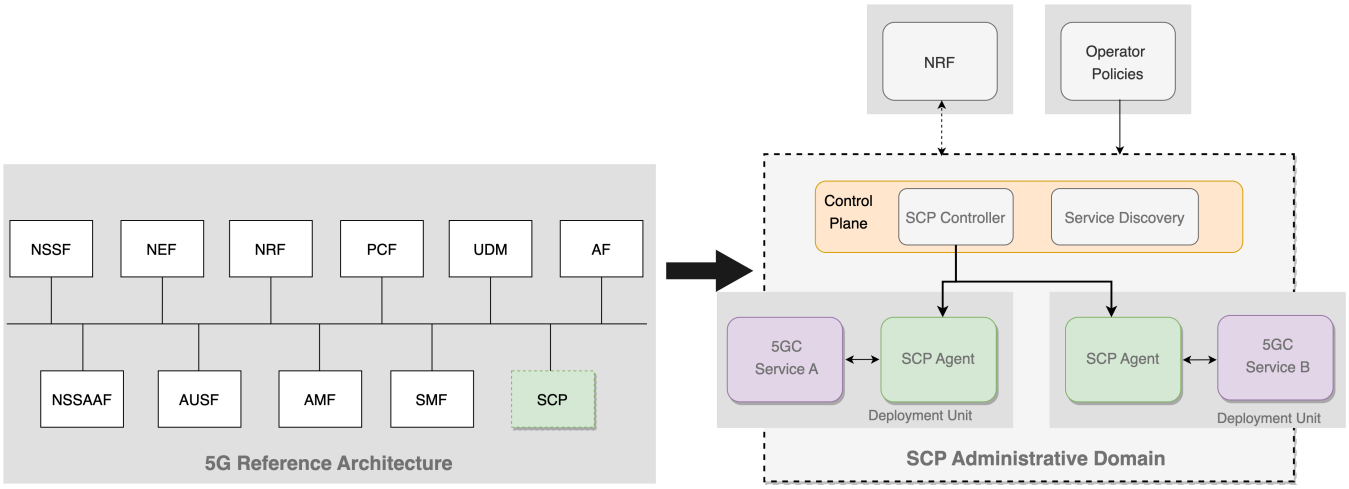


Figure 7A.1: 5G Control Plane Reference Architecture and the Inclusion of SCP.

by microservice architectures and provide reliable message delivery through efficient routing, load balancing and traffic monitoring between NFs.

As defined by the 3GPP standards, the functionality of the SCP primarily provides a guideline and outlines the protocol definition rather than specifying an implementation for service mesh deployments. Instead, vendors are tasked to develop and implement following the 3GPP specification. Service mesh, a promising technology, offers an avenue to subsume the SCP functionality outlined by 3GPP. By definition, a service mesh is a dedicated infrastructure layer for handling service-to-service communication [5]. It is responsible for the reliable delivery of requests through the complex topology of services that constitute a modern, cloud-native application. As described in Chapter 3, the deployment model of a service mesh is implemented as a collection of proxies deployed alongside an application to handle failure detection and streamline service-to-service communication. Furthermore, deploying 5G services within a service mesh enacting as SCP presents unique design considerations that require meticulous evaluation and standardisation. These include considerations around 5G signalling routing, service discovery, network programmability, traffic management, observability and load balancing.

The chapter aims to provide an industrial application of service mesh for efficiently executing 5G core signalling functions. This research acknowledges the need for advanced capabilities in service mesh architectures. It outlines the essential extensions within the service mesh proxy to integrate 5G semantics effectively. By exploring the adaptability of the Envoy proxy for 5G use cases, the study proposes to embed 5G routing requirements into the service data path, bridging the gap between traditional service mesh functions and the specific demands of 5G signalling.

Finally, this chapter explores identifier systems, which are central to future work to improve the latency of service communication through service discovery. This investigation contributes to identifier systems, a topic to be considered for inclusion in future standard guidelines.

7A.2.2 Session Communication Proxy and 5G Core Services

In 3GPP Release-15, the advent of 5GC introduced a Service-Based Architecture (SBA) [6], [7]. In this architecture, NFs interact with the Network Repository Function (NRF), a central repository for all 5G core NFs. This enables all 5G network NFs to discover other NFs and the

services and capabilities each exposes. Subsequent interactions between NFs are carried out directly using the HTTPS/HTTP2 protocol [8] via the Service-Based Interface (SBI).

Due to 5GC's microservices-based design, the communication traffic between NFs is anticipated to increase significantly compared to previous generations. Combined with millions of concurrent signalling messages, this architectural model is not resilient to large-scale real-time deployments, as the probability of partial failure grows exponentially with system size [9].

As described in the 3GPP Release-16, the SCP was introduced to address the challenges of extensive signalling traffic among the NFs. Designed to support large deployments, the SCP efficiently manages millions of concurrent transactions, prioritises signalling traffic, and ensures balanced load distribution across NF instances. Building on this foundation, the SCP architecture, as detailed in 3GPP's Release-16, comprises two primary components: the SCP Controller and the SCP Agent. The SCP Controller is the central control plane, responsible for managing and coordinating the SCP Agents. It also synchronises with the Network Repository Function (NRF) to obtain the current status of the network functions. The SCP Controller and SCP Agent communicate via an internal channel. The controller manages and distributes configurations to the agents and monitors their health through regular intervals, which involves extracting performance data and monitoring heartbeats from the SCP Agent. If the SCP Controller determines that an SCP Agent has failed, it initiates corrective remediation measures.

Functionally, the SCP Agent operates as a data plane, performing the role of HTTP intermediary between service consumers and service providers. The SCP acts as a mediator or proxy to facilitate communication between NFs. The SCP Agent's primary functions include acting as a load balancer and circuit breaker, encryption, and HTTP routing, directing traffic between the service consumer and provider. As illustrated in Figure 1, the SCP Agent interposes the communication path, determining the route for the consumer traffic. It selects an appropriate target based on the consumer's request and directs traffic to the targeted SCP Agent representing the provider. The SCP Controller orchestrates this routing and selection policy, utilising information sourced from the NRF, and subsequently disseminates routing and service selection directives to the SCP Agents.

7A.3 Considerations and Requirements

In a scaled 5GC environment, the challenges of direct communication between NFs include communication overhead and increased latencies. Mitigating this challenge in the 5G space requires delineating the responsibilities between the NF and SCP. This approach entails the NF focusing on the core network functions of the service while the SCP, considered an external service, manages the processing tasks previously handled by the NF. These tasks encompass encryption, load balancing and circuit breaking, all of which must be carefully considered.

In order to have the service mesh assume the role of SCP, modifications are required in the framework to support 5G core signalling use cases. Service mesh deployments primarily cater to traditional enterprise applications, not 3GPP deployments. This discrepancy highlights a gap in meeting specific needs, thus creating additional design considerations for service mesh. Attaining optimal scalability and seamless end-to-end performance necessitates specific requirements customised to 5G deployments within the service mesh infrastructure. The following section delineates these vital considerations:

1. **Optimised Low-Latency Communication:** Given the need for efficient communication between NFs in 5G deployments, designing and tailoring service mesh models explicitly for low-latency operations is essential.
2. **Scalability with Performance Maintenance:** The service mesh model should support the scalability of NFs without negatively impacting performance or increasing latency. This requirement is critical given the inherent communication complexities among 5G NFs.
3. **Efficient Service Discovery:** The service mesh infrastructure should offer scalable, low-latency service discovery tailored to the specific needs of 5G NFs, enhancing deployment scalability and reducing end-to-end communication latency.
4. **Comprehensive and Unified Observability:** The service mesh infrastructure should provide 5G-specific and comprehensive infrastructure-wide observability for monitoring and troubleshooting across the system.

Addressing the specified requirements and achieving the targeted Key Performance Indicators (KPIs) requires careful consideration and standardisation of various design choices. The following sections explore the high-level solution framework as well as examining these design considerations.

7A.3.1 Proposed Solution

The service mesh is a dedicated infrastructure to facilitate service-to-service communication within microservice architectures central to the open-source project Istio [10], [11]. This framework consists of two primary components: the control and data plane. The Envoy proxy is utilised as a reverse proxy in microservice architectures. Envoy operates predominantly as a sidecar proxy, mediating all network communication between microservices while managing service discovery, load-balancing, circuit-breaking, performing mutual TLS encryption and unified observability.

Figure 7A.2 provides an overview of Envoy's data plane operations, illustrating the filter chain where each filter contains a unique role. The listener operates at the socket level, tracking incoming connections. Attached to the listener are the network filters responsible for routing decisions. These filters consist of rules that map virtual hosts to clusters. The metadata within the request, such as the HTTP Host header, is essential to serve as a lookup key in the routing table and a namespace that distinguishes between distinct target instances. The entries in this table determine the destination service to which the request will be forwarded. The load balancer and associated clusters use specific policies to dictate the load-balancing algorithm and select hosts for inclusion in the load-balancing pool. These policies also include retries, timeouts, circuit breakers and other mechanisms for outlier detection related to service-level objectives.

In service mesh implementations, Istio was chosen for the solution for two primary reasons. Firstly, Istio's transition from its traditional role to embrace the new position of SCP is largely attributable to the extensibility of its data plane, particularly the inclusion of WebAssembly (WASM) [12]. WASM is a general-purpose virtual machine instruction set architecture that provides a bytecode that can be executed on native machine architectures. One of the engines that can interpret and run this bytecode is V8, a high-performance JavaScript and WebAssembly engine written in C++ [12], [13]. Envoy fundamentally embeds a subset of the V8 engine,

traditionally used in Chrome and Node.js. This extensibility enables developers to write filters as separate modules in their preferred languages. Once written, Envoy can load these filters, which are executed at runtime, influencing the proxy's behaviour. For instance, this WASM integration could facilitate the inspection and manipulation of traffic as it flows through the proxy [14]. This approach eliminates the need for developers to infuse this functionality directly into microservices or be dependent on specific libraries. Instead, the feature is introduced as an inherent part of the sidecar, marking a distinct advantage of embedding SCP functionality into the service mesh.

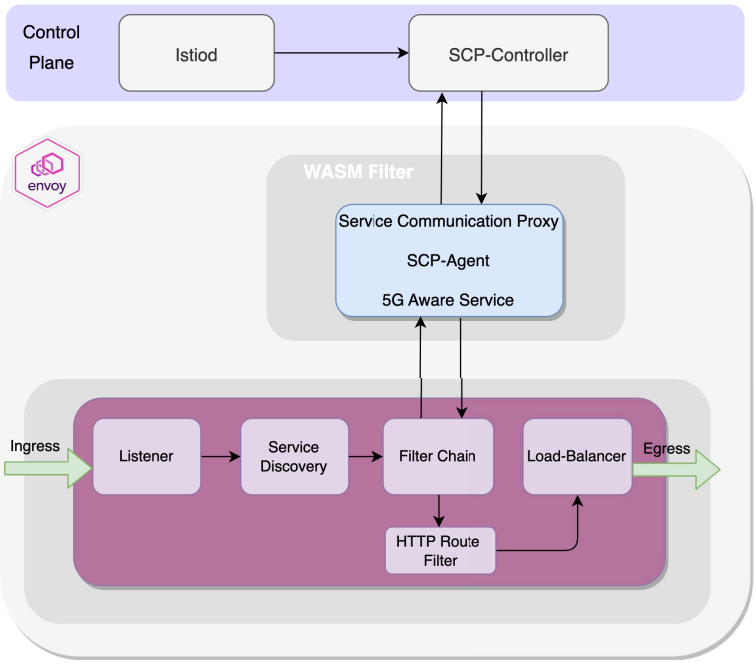


Figure 7A.2: Enhanced Envoy Architecture.

From the extensibility of Envoy's integration with WASM, two contributions to the service mesh control path architecture have emerged: the inclusion of the SCP Agent and the SCP Controller.

The SCP Agent operates within the Envoy's data plane and is implemented as a WASM filter. This SCP Agent offers three primary functionalities:

1. **Synchronisation with the SCP Controller**: The SCP Agent consistently syncs with the SCP Controller. This ensures that policy decisions and routing rules remain current with the latest network dynamics.

2. **Handling of NF Instances Discovery Requests**: When discovery requests are received, the SCP Agent communicates with the SCP Controller, which interacts with the NRF service. The NRF's response data is relayed to the SCP Agent via the SCP Controller. This response typically encompasses Uniform Resource Identifier (URI) [15], application protocol and service parameters.

3. **Integration with Envoy for Routing and Load Balancing**: Paired with Envoy's existing capabilities, the solution actively monitors downstream HTTP/2 messages. It then leverages 5G header data to determine the routing for requests, ensuring they reach the intended target provider. The pseudocode below illustrates the SCP Agent's request handling function for a detailed understanding of the overall process.

SCP Agent: Request Handling

```
for each incoming HTTP request to SCP Agent do
    Identify target Network Function (NF), e.g., AMF, SMF, etc.
    if request is for NF discovery and origin is free5gc then
        Redirect request to SCP agent for handling.
    else if request targets a specific NF (other than NRF), and origin is free5gc then
        Modify and append 3GPP protocol headers.
        Forward request to the designated target NF.
    end if
end for
return result
```

The SCP Controller is an integral component of the service mesh control plane and has been developed as an extension to Istio. Its responsibilities include:

- Registering with the NRF to receive updates about NF details and their operational status.
- Constructing network topologies and formulating dynamic 5G routing policies.
- Regularly updating the data plane with recent policy and routing decisions.
- Discovering available NF profiles from the NRF and registering its NF profile.
- Subscribing to notifications from the NRF.
- Conducting health checks and, upon startup or receipt of notifications from the NRF, disseminating available NF Profiles to the SCP Agent.

Deployment Scenario

As depicted in Figure 7A.3, NFs, such as the AMF (Access and Mobility Management Function) and SMF (Session Management Function), along with SCP Controllers, register with the NRF. Additionally, the SCP Controller subscribes to NRF events to keep the state and information about the NFs updated. A source NF, such as the AMF, sends an HTTP signalling message for communication. This message contains a header field named '3gpp-Sbi-Target-apiRoot' with the target URI value set to 'SMF', indicating the destination NF, which is the SMF. This communication is facilitated by Envoy proxies present on both the source and destination sides. The source SCP Agent, which operates as part of the AMF pod, interprets the incoming request. It then conducts routing discovery and forwards the request to the appropriate SMF instance. Moreover, the SCP Agent within the SMF Pod listens for incoming HTTP messages. Upon receiving a message, it parses the content, verifies the necessary 3GPP message semantics and then delivers the message to the SMF.

7A.4 Service Discovery as Identifier System

The benefit of this cloud-native service mesh approach is the localisation of discovery information, which the proxy provides to the client. The proxy's location to the client, along with prefetching the connection parameters required for establishing a connection between NFs, reduces latency by eliminating round trips. The systems managing these requests, known as

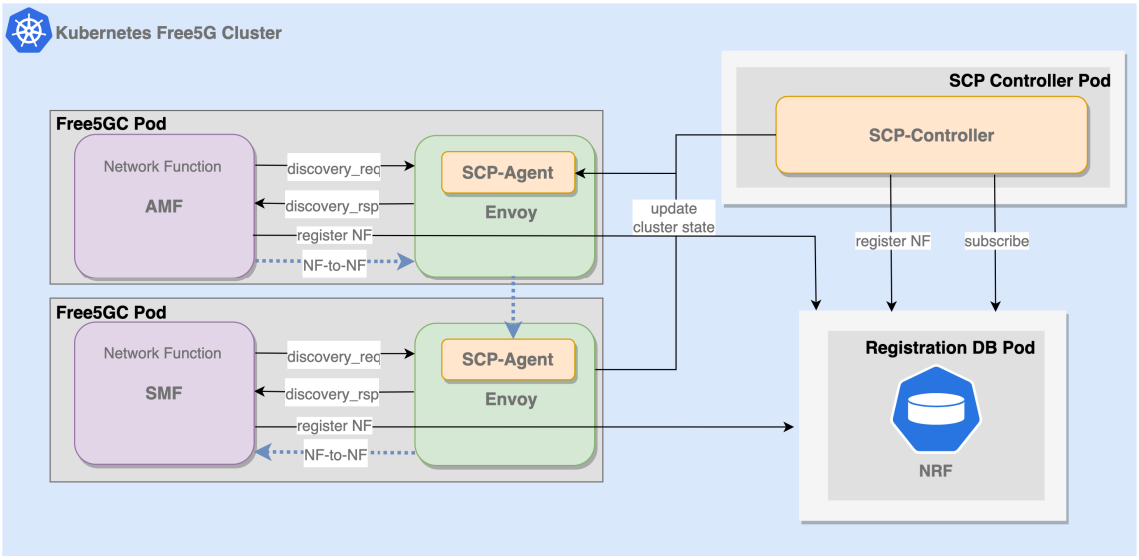


Figure 7A.3: Scenario-3: SCP as Distributed Envoy Proxy.

Service Discovery systems, act as identifier systems. This differs from the traditional discovery system, where the identifier resolves to an IP address.

As 5G core NFs evolve to mirror development patterns and methodologies of microservices, operators will implement strategies such as auto-scaling services for improved application response time. In addition, methodologies such as blue-green deployments to introduce functionality by version will cause an evolution of the original pattern.

From an identifier system perspective, such architectures will include clones of services, allowing a client service to select a target based on its capability or proximity. These identity systems are also responsible for ensuring clients access the correct connection profile, which requires customisation of responses. This customisation could be for security purposes, such as access-control. As discussed in Chapters 5A and 6A, flexibility, speed and core security are core parameters of the system. The following section proposes a set of requirements for future work to evolve the service discovery.

7A.5 Future Work: Shifting Roles in Service Discovery

The 5G cloud-native architecture is undergoing continuous evolution, and integrating an enhanced Envoy proxy with service discovery introduces the following key areas for investigation.

Domain-specific discovery semantics

As part of the discovery process, the Envoy SCP Agent must consider 5G-specific semantics in the discovery mechanisms. This includes rejecting consumer requests with appropriate 3GPP-specific error codes if no NF Service Producer supports the requested features. To address these needs, the conventional service discovery method must shift from providing a universal answer to answers tailored for each consumer. This entails customising responses for reasons such as access control, policy, or to optimise the connection between the consumer and provider. A strategy involves enhancing queries with meta-information, enabling clients to exchange their details for more accurate service information [9]. As illustrated, the service discovery resolution can then fine-tune its answers, offer improved estimations or even decline connections.

By enforcing identifiers to behave uniquely based on the querying entity, the resolution system could enable the implementation of access controls that adhere to domain-specific discovery semantics. In this context, service discovery is set to transition from a mere dictionary-based search to a system that enriches queries with added details, allowing personalised responses based on the querier's identity and metadata.

Low-latency and highly dynamic service discovery mechanism to support time-sensitive communication

One key challenge in any 5G deployment in the cloud is providing low-latency communication infrastructure to meet 5G KPI requirements [16]. The discovery mechanism must maintain a repository of parameters related to the destination NF reachability latencies and load.

Enhancing the service discovery system to provide more than just IP necessitates a shift from modelling hosts to modelling services. Services demand additional details, including port and protocol data, rather than just IP. This also involves understanding the precise needs and intentions of the service consumer's request. A client and service discovery broker exchanging information could streamline the search results, enhancing performance in establishing connections without additional steps to gather connection information. For instance, in the context of the upcoming 6G, the service discovery could supply connection profiles hinting at protocol support (e.g., HTTP/3 for 6G) alongside port details [17]. This ensures fast connections without latency during service negotiations.

Furthermore, application namespaces could be included as a parameter in the Service Discovery resolution. As NFs consolidate onto a shared platform, name-based routing distinguishes application sessions on a common host. Within this discovery process, the subscriber service may populate the name within the application protocol during the initial connection with the provider service. 3GPP's release 16 also suggests that a name-based method is another viable approach for discovery and routing [3], [4].

Additionally, service discovery systems often direct clients based on configured priorities and weights to enhance connection speeds and address time sensitivity. This method, aimed at evenly distributing traffic, might not always account for the querier's preference for faster response times. This mismatch can conflict between the server's choice of an 'optimal' service and the client's performance needs. Instead of providing a single answer, the service discovery system could offer multiple candidate answers, allowing the client to choose based on specific needs. For example, consider Figure 7A.4 where **_service.example.com** contains three possible answers, which are digital clones of each other. Rather than the identifier system steering the client to one of the answers, the system returns all three possible answers, allowing the client to determine the most appropriate candidate. Furthermore, the envoy proxy sends three synthetic transactions in parallel, choosing latency as a weight for its decision. In this instance, the client selected server B based on computing the lowest latency. Latency is just one example; many other parameters could influence a client's decision to select an appropriate resource. This adaptable method aligns closely with consumer interests, enabling a custom service selection and potential for further optimisation.

While client-side resolution computation is often less favourable, a possible implementation with Envoy could handle this task, ensuring no added burden on the NF. Envoy could facilitate the logic of identifying the best candidate through a series of synthetic transactions on behalf of the consumer to determine the optimal target. This approach represents another promising direction for service discovery evolution.

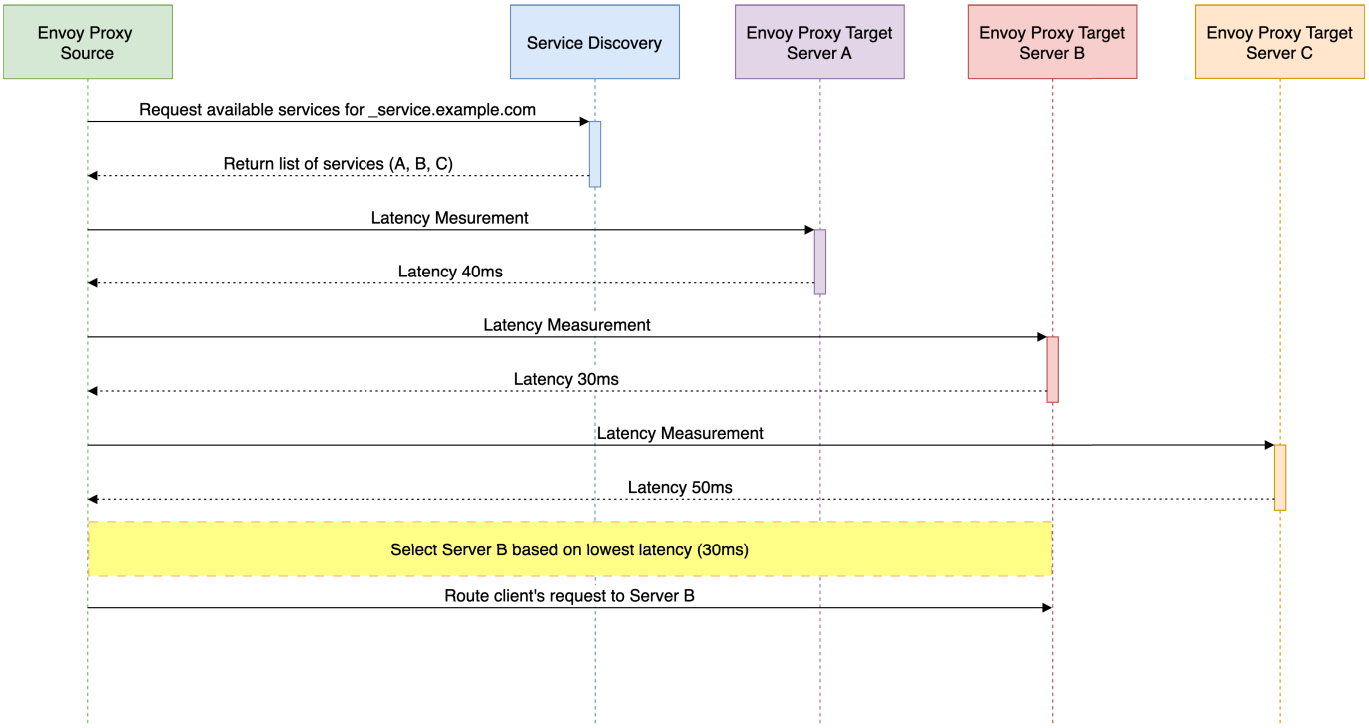


Figure 7A.4: Service Discovery: Client Side Candidate Selection.

7A.6 Conclusions

While 5G Service-Based Architecture (SBA) has enabled the cloudification of the 5G network and deployments, it has inherent challenges. Existing cloud infrastructure, designed for general applications, struggles to meet the stringent network response times demanded by 5G applications. This contribution proposes a service mesh enhancement to address these challenges. By implementing advanced traffic engineering, such as policy routing, load-balancing and circuit breaking, the proposed solution ensures application resilience on cloud platforms where the underlying infrastructure is out of the operator's control. Furthermore, the integration of SCP has enhanced performance by synchronising NF connection parameters across the proxy nodes within the cloud environment. Identity systems such as service discovery will be essential as NFs expand to mirror the microservices patterns. As outlined in future service discovery work, introducing the flexibility and security parameters for customised responses could improve latency and access control. However, the proposed requirements to enhance service discovery warrant further exploration. This contribution aims to trigger discussions on the future direction of identifier systems within cloud-native telecommunications.

Chapter 7B

Future Work B: Context-Aware DNS Query Handling

7B.1 Purpose of this Chapter

This Chapter 7B proposes a context-aware query handling model for the DNS, based on query qualification explored in Chapter 5B. This is another promising application of this study to develop context-awareness to enhance customised responses for queriers. This resulted in an issued U.S. patent titled *"Context-Aware Domain Name System (DNS) Query Handling"*, demonstrating this research's application and marking the context-awareness component as future work. See Appendix E for details. Disclaimer: The descriptions and explanations provided in this chapter are limited to the content explicitly detailed in Appendix E.

7B.2 Forward Looking Challenges in Network Security

Due to the shift towards remote working, enterprise network control has faced new challenges in the last four years. This transition accompanied networking without boundaries or 'borderless' networking, which eliminates the traditional perimeter-based security model. Traditional enterprise network topologies such as hub-and-spoke rely heavily on network-based middleware to facilitate security inspection. Through traffic engineering techniques, all traffic is inspected through centralised points of presence before reaching the destination, traversing firewalls, intrusion detection systems and VPN concentrators. In a borderless network model, rather than having traffic management forwarded to a central point – which proves inefficient due to suboptimal network paths – users can access their resources securely and directly over the Internet.

In this new networking paradigm, the focus shifts towards implementing security frameworks that extend the administrative domain across boundaries. One such framework is Zero Trust, which operates on the principle of 'never trust, always verify' [18]. Zero Trust addresses the core challenges by applying identity authentication and network segmentation, irrespective of the physical location of users and resources.

This section proposes the DNS as an additional approach within the existing security technologies framework that aims to mitigate threats from Internet users and machines residing in data centres and the cloud. While the DNS in this discussion is security-centric, the focus is the application of a computational DNS that provides custom responses based on various data

sources. The rationale for integrating data sources with DNS is the technology's ability to cross administrative boundaries, resolving user and machine requests to access the target entity.

This section proposes a novel method for enhancing the system's responsiveness and security through a technique referred to as context-aware DNS query handling, denoted as 'Semantic DNS', by adjusting the DNS resolution system to introduce context-awareness. The core of the techniques is handling the responses dynamically by customising DNS responses based on the data sources received by the DNS, which include the device and the application infrastructure posture. This results in proactively responding to security risks. This ties back to proof-by-argument of an adjustable namespace and resolution, which goes beyond the static translation of the DNS. While this thesis is not security-centric, it illustrates another application of DNS as a computational element.

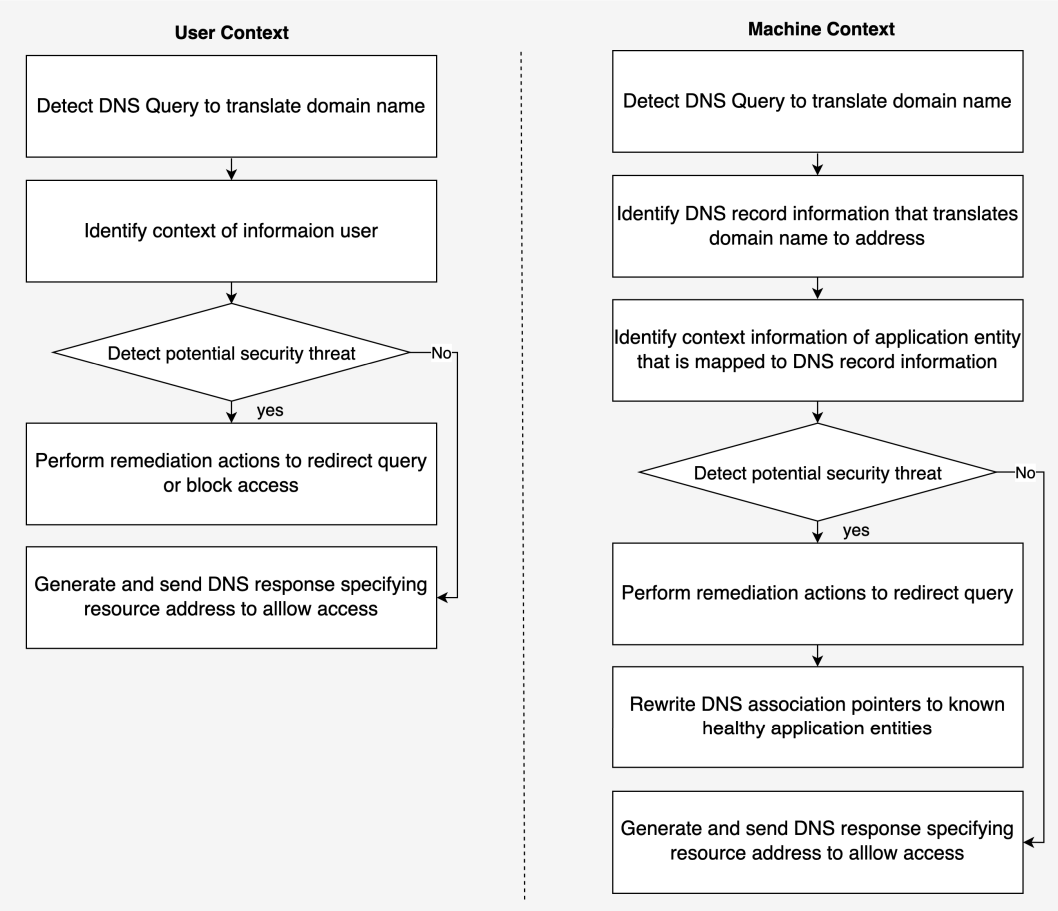


Figure 7B.1: User and Machine Process of Context-Aware Query Handling.

7B.3 Considerations and Requirements

The DNS is divided into a resolution protocol for handling requests and returning responses and a namespace for managing and allocating names to ensure uniqueness. Without this namespace, the communication system would be hindered. For threat mitigation strategies to apply within the DNS, adjustments are required in both the resolution and authoritative systems.

To enhance the DNS system's query handling for threat mitigation, adjustments to support the integration of various data sources are required for context-awareness. Furthermore, enhancing

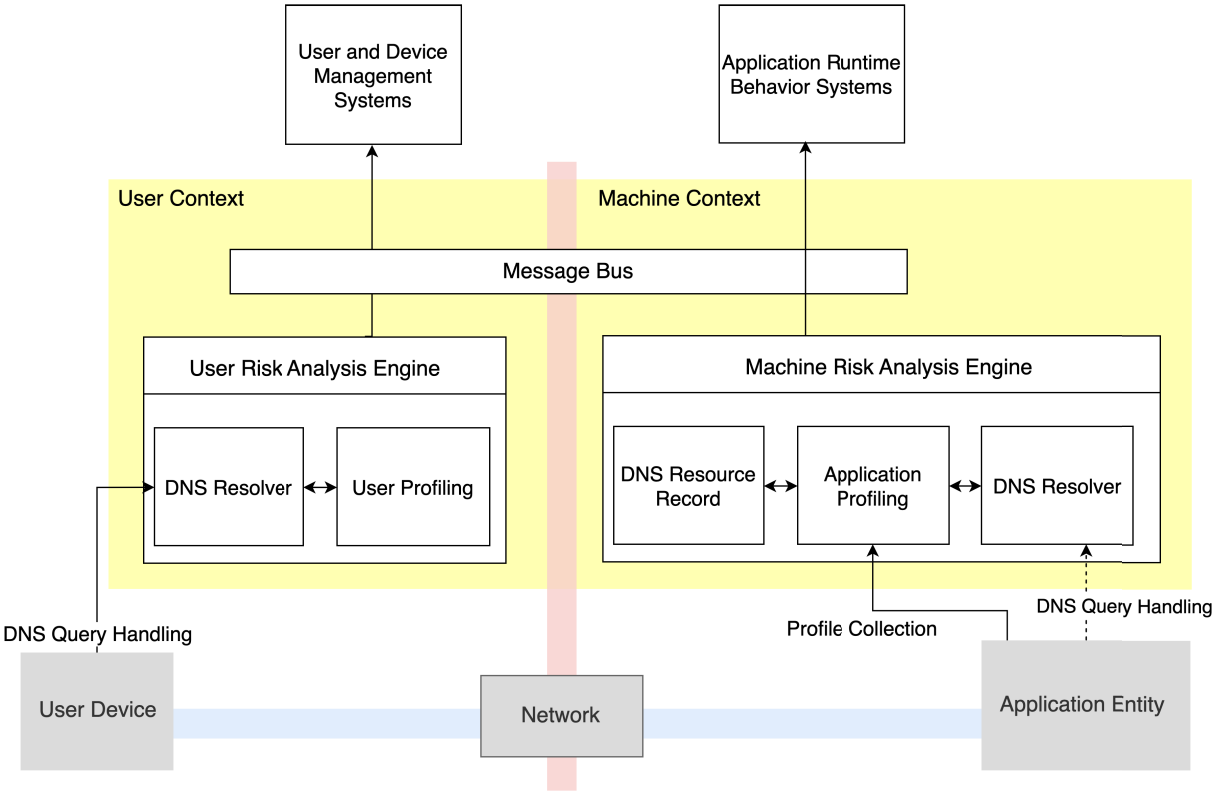


Figure 7B.2: Proposed Solution.

the authoritative DNS system requires enabling dynamic updates of resource records to reflect the latest security information for subsequent queries to that record.

The following sections detail the prerequisites and operational requirements:

- **Data Collection from user devices**: The DNS system must gather inventory data from client devices, including hardware and software profiles, geographical locations, and user behaviour, to assess the overall risk profile.
- **Data collection from applications**: Unlike user information, the DNS system must collect baseline data on active application processes. Runtime deviations from the baseline indicate a re-evaluation of the application risk posture.
- **Context-Aware DNS query handling (recursive resolver)**: The DNS system evaluates queries based on the collected data sources to decide how to handle each query. There are three modes:

  1. A normal name-to-IP resolution.
  2. Redirect the query for further examination before the decision.
  3. Block the query, resulting in no further resolution.

- **Custom DNS responses (authoritative DNS)**: The DNS modifies responses based on the application's security posture associated with the resource record. The DNS system must dynamically adjust the resource records.

7B.4 Proposed Solution

The proposed solution enhances the DNS query process by introducing a context-aware mechanism. Upon receiving a DNS query, the proposed solution undertakes the following steps, as depicted in Figure 7B.1, to customise responses based on the querier's context:

- **User Context:** The system detects a DNS query and identifies the corresponding DNS record information. If there is a deviation from the user's expected behaviour or an indication of elevated risk, the system either:
  - redirects the user to a third-party proxy for further evaluation or
  - blocks the query by not providing a resolution to the domain name.
- **Machine Context:** Upon detecting a DNS query, the system identifies the domain name being queried. If there is a deviation in the application entity's behaviour associated with this DNS record, the system:
  - redirects the query to a known healthy application instance, and
  - removes the DNS record's association with the compromised application entity to prevent subsequent access.

The proposed solution defines separate user and machine contexts, each with a context-aware DNS engine; however, it differs in functionality, as depicted in Figure 7B.2. In the user context, the recursive resolver handles the client queries, comparing them to past queries stored in the risk engine. Based on the information in the DNS packet, the risk engine examines the resolver IP address and query name and decides whether to resolve, redirect or block the query based on the client profile. In addition to DNS query data, the client profile is constructed from data collection of the user and device via the message bus. The data collection includes inventory management of user and device identity and management systems. The data includes software and hardware inventory, client/device geographic location, and behavioural patterns influencing DNS response decisions.

While having functionality similar to the user context, the machine context focuses on the authoritative domain namespace. It integrates a message bus with an application runtime and posture management system to collect data on application behaviour, including software processes, logs, and operating system information such as kernel behaviour and network statistics. This data constructs a profile for application entities paired with DNS resource records. When deviations from known baselines are detected, indicating a risk, the system adjusts the DNS record by removing the IP address of the affected application out of rotation, directing user traffic toward secure applications.

7B.5 Future Work: Towards a Semantic DNS

The proposed solution is another example of an evolutionary direction for DNS that shifts from traditional DNS, where names are resolved to IP addresses without understanding context, to a context-aware system capable of semantic interpretation. The examples described in this chapter demonstrate the significance of the client profile generating context to enable the resolution system to either provide standard responses, customise answers or withhold all answers. A potential advance would be for such a system to reveal multiple personas associated with a single identity, drawing parallels to human social interactions where the context of how

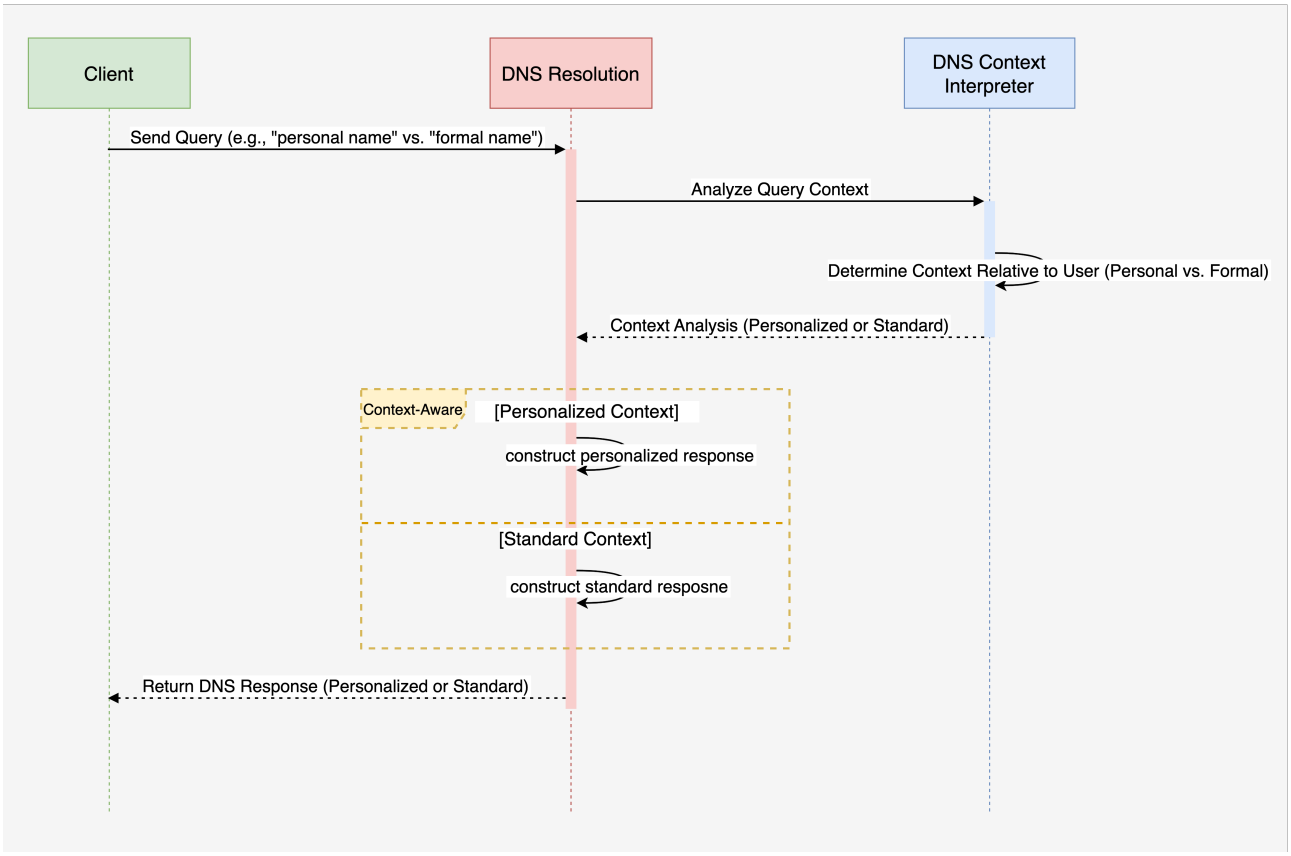


Figure 7B.3: Future State of Context-Aware DNS Query Handling.

an individual is referred and from whom matters greatly. For example, as depicted in Figure 7B.3, the distinction between formal and informal names, when queried, is an instruction to reveal a persona that is not available to anyone else. This nuanced response mimics the natural human interactions but introduces complexities. For example, traditional models of published directories do not meet the requirements of such dynamic identity systems.

The potential research direction could shift to a more intricate system that does not reveal any more about identity to third parties than necessary. It explores a directory system in which names are associated with elements that remain undisclosed unless the correct client initiates the conversation. This potential direction transforms the DNS from a simple dictionary lookup, where the answer is consistent irrespective of the querier, to a computational resource that provides a semantic interpretation of the name and querying client.

Efforts to implement aspects of this vision within the DNS have revealed constraints, as discussed in Chapter 6A, indicating that extending beyond current capabilities introduces trade-offs. However, the study demonstrates that adapting to meet the needs of digital identity management is not only feasible but also a valuable suggestion for a reasonable direction the DNS could take.

.

References for Chapters 7A and 7B

[1] B. Dab, I. Fajjari, M. Rohon, C. Auboin, and A. Diquélou, "Cloud-native service function chaining for 5g based on network service mesh," in *ICC 2020-2020 IEEE International Conference On Communications (ICC)*, IEEE, 2020, pp. 1–7.

[2] N. Dragoni, I. Lanese, S. T. Larsen, M. Mazzara, R. Mustafin, and L. Safina, "Microservices: How to make your application scale," in *Perspectives of System Informatics: 11th International Andrei P. Ershov Informatics Conference, PSI 2017, Moscow, Russia, June 27-29, 2017, Revised Selected Papers 11*, Springer, 2018, pp. 95–104.

[3] ETSI, "5g; 5g system; technical realization of service based architecture; stage 3," ETSI, Tech. Rep. 129 500, version 16.7.0, Apr. 2021. [Online]. Available: `https://www.etsi.org/deliver/etsi_ts/129500_129599/129500/16.07.00_60/ts_129500v160700p.pdf`.

[4] ETSI, "5g; procedures for the 5g system (5gs)," ETSI, Tech. Rep. 23 502, version 16.9.0, Jul. 2021. [Online]. Available: `https://www.etsi.org/deliver/etsi_ts/123500_123599/123502/16.09.00_60/ts_123502v160900p.pdf`.

[5] W. Li, Y. Lemieux, J. Gao, Z. Zhao, and Y. Han, "Service mesh: Challenges, state of the art, and future research opportunities," in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, IEEE, 2019, pp. 122–1225.

[6] A. Ghosh, A. Maeder, M. Baker, and D. Chandramouli, "5g evolution: A view on 5g cellular technology beyond 3gpp release 15," *IEEE access*, vol. 7, pp. 127 639–127 651, 2019.

[7] 3GPP, "System Architecture for the 5G System," Tech. Rep. TR 23.501, version 16.3.0, Dec. 2018. [Online]. Available: `https://www.3gpp.org/ftp/Specs/archive/23_series/23.501/23501-v16.3.0.pdf`.

[8] M. Thomson and C. Benfield, *HTTP/2*, RFC 9113, Jun. 2022. DOI: `10.17487/RFC9113`. [Online]. Available: `https://www.rfc-editor.org/info/rfc9113`.

[9] A. Babakian, P. Monclus, R. Braun, and J. Lipman, "A retrospective on workload identifiers: From data center to cloud-native networks," *IEEE Access*, vol. 10, pp. 105 518–105 527, 2022.

[10] *Istio*, Aug. 2016. [Online]. Available: `https://www.istio.io`.

[11] O. Sheikh, S. Dikaleh, D. Mistry, D. Pape, and C. Felix, "Modernize digital applications with microservices management using the istio service mesh," in *Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering*, 2018, pp. 359–360.

[12] A. Haas, A. Rossberg, D. L. Schuff, *et al.*, "Bringing the web up to speed with webassembly," in *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2017, pp. 185–200.

[13] J. Ménétrey, M. Pasin, P. Felber, and V. Schiavoni, "Webassembly as a common layer for the cloud-edge continuum," in *Proceedings of the 2nd Workshop on Flexible Resource and Application Management on the Edge*, 2022, pp. 3–8.

[14] S. M. Jain and S. M. Jain, "Extending istio with webassembly," *WebAssembly for Cloud: A Basic Guide for Wasm-Based Cloud Apps*, pp. 151–160, 2022.

[15] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform resource identifier (uri): Generic syntax," Tech. Rep., 2005.

[16] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Toward 6g networks: Use cases and technologies," *IEEE Communications Magazine*, vol. 58, no. 3, pp. 55–61, 2020.

[17] J. Zirngibl, P. Sattler, and G. Carle, "A first look at svcb and https dns resource records in the wild," in *International Workshop on Traffic Measurements for Cybersecurity 2023*, 2023.
[18] V. Stafford, "Zero trust architecture," *NIST special publication*, vol. 800, p. 207, 2020.

Chapter 8

Conclusions

The DNS stands as an exemplar of an identifier system, with alternative identifier systems that exist in parallel not drifting significantly from the root of the DNS, remaining fundamentally mappable. As outlined, the key lies in preserving the cohesiveness of the Internet, with trust anchored in the naming system. This necessity was made more acute by the uncertainties of the routing and addressing system, influenced in the latter case by IPv4 address exhaustion.

In today's Internet marketplace, a domain name's significance goes beyond the traditional role of treating identifiers as constant indices. It is not the identifier that determines the market; it is the complex relationship between the behaviour of the identifier and the system's ability to customise responses that determine the impact of the system as it addresses the user and market demands.

As examined in this thesis, one of the fundamental questions is whether these changes to support customisation and context-awareness are inherent flaws in the DNS or whether they are driven by the competitive nature of the marketplace, aiming to offer differentiated customisation solutions. In this study, it is the latter. The transformation occurring across all identifier systems is the behaviour of the name and its capacity for customisation beyond static translation.

The contribution of this research falls into four categories. Each category directly maps to the research objectives outlined in Chapter 1.3.

Contribution 1 - Systematic Literature Study as Research Method
This study has synthesised insights from a wide range of literature, spanning both Internet and library resources, to examine the behaviours of various systems in supporting identifiers, with a specific focus on their relevance to the DNS. The literature study uncovered two insights. First, it identified computational elements in successive generations of the DNS that support context-aware queries and embedding instructions within the query name. Second, it observed that parallel alternative namespaces deviating from the root of the DNS risk fragmenting the Internet. This discussion is crucial for future evaluation. The study suggests an approach for integrating existing namespaces with minimal disruption but requires a universally agreed approach and framework to introduce cohesion.

Contribution 2 - Proof by Argument towards Customisation and Context-Awareness
While not commonly recognised in systems research, the insights from the literature review and study emphasise the importance of a computational approach to address the two key findings

discussed in Chapter 5B. This research addresses whether the DNS is broken or whether it is evolving for competitive reasons, concluding with the latter. This study advocates for a shift from the traditional look-up system to dynamic computational resources to support flexibility in identifier responses. This proof by argument presents scenarios offering practical instances of such an evolution. Furthermore, it emphasises that no technology is immutable, and the DNS, the incumbent, is no exception. The emergence of parallel name systems represents a potential shift in research objectives in this field of study. The contribution argues for an evolutionary rather than revolutionary approach, suggesting ways to accommodate new modes of operation alongside the existing Internet infrastructure, thereby ensuring their coexistence.

Contribution 3 - Proving the Proposed DNS and URL Computation Model
Prior to lending weight to the proposition, the proof by argument chapter highlighted the case for context-awareness and customisation by introducing alternative paths the DNS could take. Chapter 6A offered a category-based DNS review, similar to the taxonomy outlining the five parameters of the DNS. This analysis is essential to identify which components of the DNS can be altered and the implications of such modifications. Emphasising that prioritising one parameter may result in compromises in others, this framework guides the proposed computational DNS model detailed in Chapter 6B.5. This model advocates for a computational approach that does not compromise flexibility while enhancing the system's speed and privacy. The effectiveness of this model is demonstrated through its performance in real-world network conditions across various geographies, showcasing its advantages over traditional DNS models.

If adopted for future implementation, the foundational principles of the proposed model will result in a clear shift in where effort is directed, where value emerges and where value declines. Augmentation of the computational aspect within the DNS is an emerging direction that will advantage agile players with sufficient scale and resources to perform this service quickly. Conversely, it will disadvantage the existing infrastructure through its distinction between name resolution and computational service. While it is unlikely to receive unanimous support, this development represents a new element of the ongoing disruption.

Contribution 4 - Applications of Research in Future Work
The approach to adaptive identifier systems is detailed in Chapters 7A and 7B. Chapter 7A outlines the transition of 5G core network functions to cloud-native platforms, requiring platform-independent identifiers. This implementation is one of the first such studies to be published, instantiating the model from 3GPP guidelines. The research also contributes by signifying the importance of future identifier systems. It describes the approach and technique, detailing the methodologies and techniques that will become essential as more applications shift to this operational paradigm. This analysis serves to inform future 3GPP guidelines, suggesting key elements that should be incorporated. Chapter 7B also contributes to another application of this research as future work. The study introduces context-awareness of DNS query handling to enhance the functionality of the DNS query and provide access control based on the client identifier and associated meta-data. This advancement builds upon the foundational 'proof by argument' research conducted earlier and introduces new paths for exploring the future direction of relative identities in networked systems.

Peering into the future, two key areas emerge for further exploration in this field. First, the ongoing discussion encompassing the unified namespace remains crucial. The network relies on a consistent set of identities for communication. This research proposed a method to enhance DNS functionality to ensure DNS remains the current unified system. However, as the need

for customisation and adaptation in parallel identity spaces outside the DNS grows, questions will arise about how network coherence can be preserved. Such parallel alternative systems risk fragmenting the Internet's namespaces, diminishing the value of a unified and cohesive communication network. This issue warrants active discussion. One approach could include a distinguishing label to refer to a namespace as described in Chapter 5A. Another might be to add a new top-level to the DNS, with the root zone of this augmented DNS being the distinguishing layer. If alternative namespaces are motivated to operate independently of the DNS, then a reliable mechanism must be created to signal which namespace to use to resolve a name. Alternatively, a passive strategy involves observing market trends to see which identifier system becomes dominant. This approach allows market demands to naturally determine the common namespace we adopt.

A deeper question will be whether identity systems must be uniquely rooted. The conventional wisdom of a single uniform identity system can be challenged; much like language, where we can shift alphabets and dictionaries yet maintain effective communication, identity systems may not require strict adherence to a uniform standard. By mutual consent, entities can operate within different identity frameworks without detrimental effects on their interaction capabilities. This idea parallels the human communication experience and suggests that while a universally recognised identity system, like DNS, offers a convenient rendezvous point, it may not always be rich or flexible enough to support the diverse and dynamic nature of digital interactions.

Finally, the concept of identity, with its relational and contextual nature, is crucial to an entity and other clients who need to refer to this entity. An entity's identifier becomes less significant than identifiers others use, especially when referring to third parties. In the client/server context, identity is more important to the client than to the server. It is possible to alter this dynamic; for instance, if a client suggests that it adopts a specific persona to the server, the server will adapt, revealing a persona that may not be disclosed to anyone else. In human social settings, these interactions are common. Individuals often portray themselves differently depending on the social circle in which they are engaging.

In the natural world, managing identity is seamless, contrasting with the digital realm's complexities. In an open environment that undertakes multiple roles, especially for those prioritising integration and privacy of communication, traditional models of open directories and searches need to be revised to embed relational and contextual behaviour into the system. Future systems may require a more nuanced approach, a slightly more convoluted system that does not compromise identity to any onlooker unless strictly necessary.

Exploring directory systems involves establishing a name and associated elements that should remain concealed unless interaction with the intended client occurs. This consists of a naming system that has evolved from a dictionary look-up, where uniform answers are given no matter the query, to a computation system, where the response may be substantially different depending on which client is inquiring. Managing relative identities and relative personas thus becomes another responsibility of identifier systems. Computers, historically, have lacked this level of subtlety. However, perhaps with the advent of artificial intelligence (AI), this is how alternative behaviours could be introduced and evolve to support relational and contextual identities that mirror the adaptive and subtle nature of human social interactions. This could redefine digital identity systems to be as intuitive as their natural counterparts.

Appendix A

Dynamic Virtual Machine Network Policy for Ingress Optimization

US011204791B2

(12) **United States Patent**
Babakian

(10) **Patent No.: US 11,204,791 B2**
(45) **Date of Patent: Dec. 21, 2021**

(54) **DYNAMIC VIRTUAL MACHINE NETWORK POLICY FOR INGRESS OPTIMIZATION**

(71) Applicant: **Nicira, Inc.**, Palo Alto, CA (US)

(72) Inventor: **Andrew Babakian**, Sydney (AU)

(73) Assignee: **NICIRA, INC.**, Palo Alto, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1450 days.

(21) Appl. No.: **14/788,573**

(22) Filed: **Jun. 30, 2015**

(65) **Prior Publication Data**

US 2017/0005923 A1 Jan. 5, 2017

(51) **Int. Cl.**
G06F 9/455 (2018.01)
H04L 29/08 (2006.01)
H04L 29/12 (2006.01)

(52) **U.S. Cl.**
CPC ***G06F 9/45558*** (2013.01); ***H04L 69/324*** (2013.01); ***H04L 69/325*** (2013.01); *H04L 61/2007* (2013.01); *H04L 61/6004* (2013.01)

(58) **Field of Classification Search**
CPC H04L 41/0893; H04L 45/24; H04L 45/44; H04L 45/74; H04L 45/586; H04L 47/20; H04L 47/125; H04L 61/103; H04L 61/2007; H04L 61/2061; H04L 69/324; H04L 69/325; G06F 9/455; G06F 9/45558; G06F 2009/45595
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2005/0129019 A1* 6/2005 Cheriton H04L 63/0272 370/392
2006/0184999 A1* 8/2006 Guichard H04L 45/00 726/3
2011/0310812 A1* 12/2011 Gage H04W 40/24 370/329
2014/0237100 A1* 8/2014 Cohn H04L 61/103 709/223

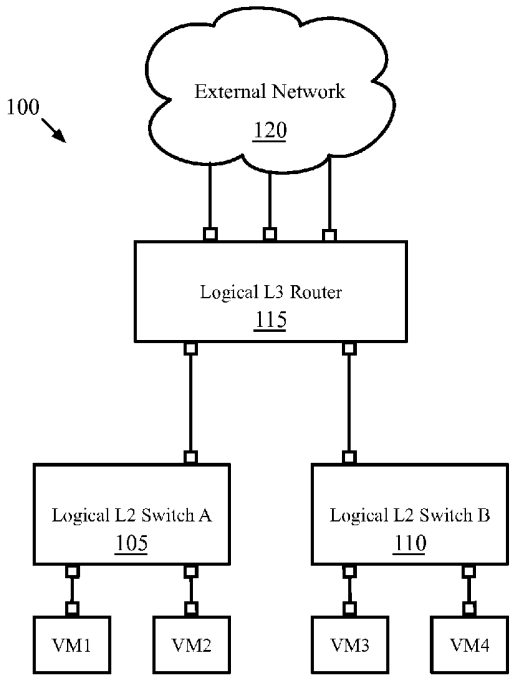
* cited by examiner

Primary Examiner — Marsha D Banks Harold
Assistant Examiner — Dharmesh J Patel
(74) *Attorney, Agent, or Firm* — Adeli LLP

(57) **ABSTRACT**

A method of performing ingress traffic optimization for active/active data centers. The method creates site-specific grouping constructs for virtual machines that run applications that are advertised to the external networks. The site specific grouping constructs provide an abstraction to decouple virtual machines from traditional networks for common ingress network policies. Each site-specific container includes a list of the virtual machines currently located at the site as well as a unique identifier of the site. Each virtual machine in a container is identified through the abstraction of metadata tag, logical data center objects, or the virtual machine's unique name. The IP address of each virtual machine is retrieved from the guest operating system and a network policy is generated to advertise the IP addresses of the virtual machines to the site's routing peer.

20 Claims, 11 Drawing Sheets



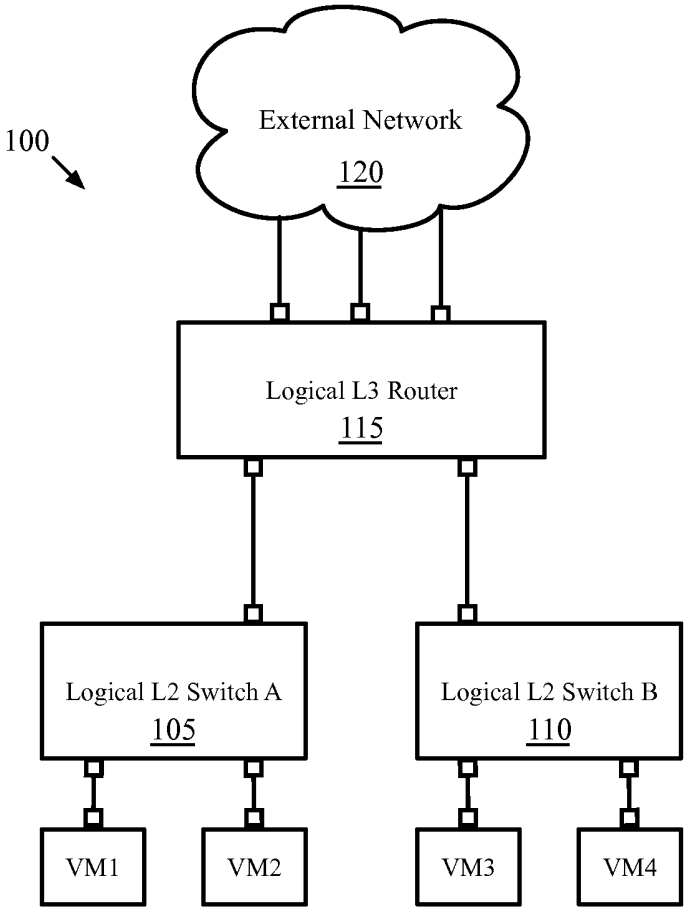

100
External Network
120
Logical L3 Router
115
Logical L2 Switch A
105
Logical L2 Switch B
110
VM1
VM2
VM3
VM4

Fig. 1

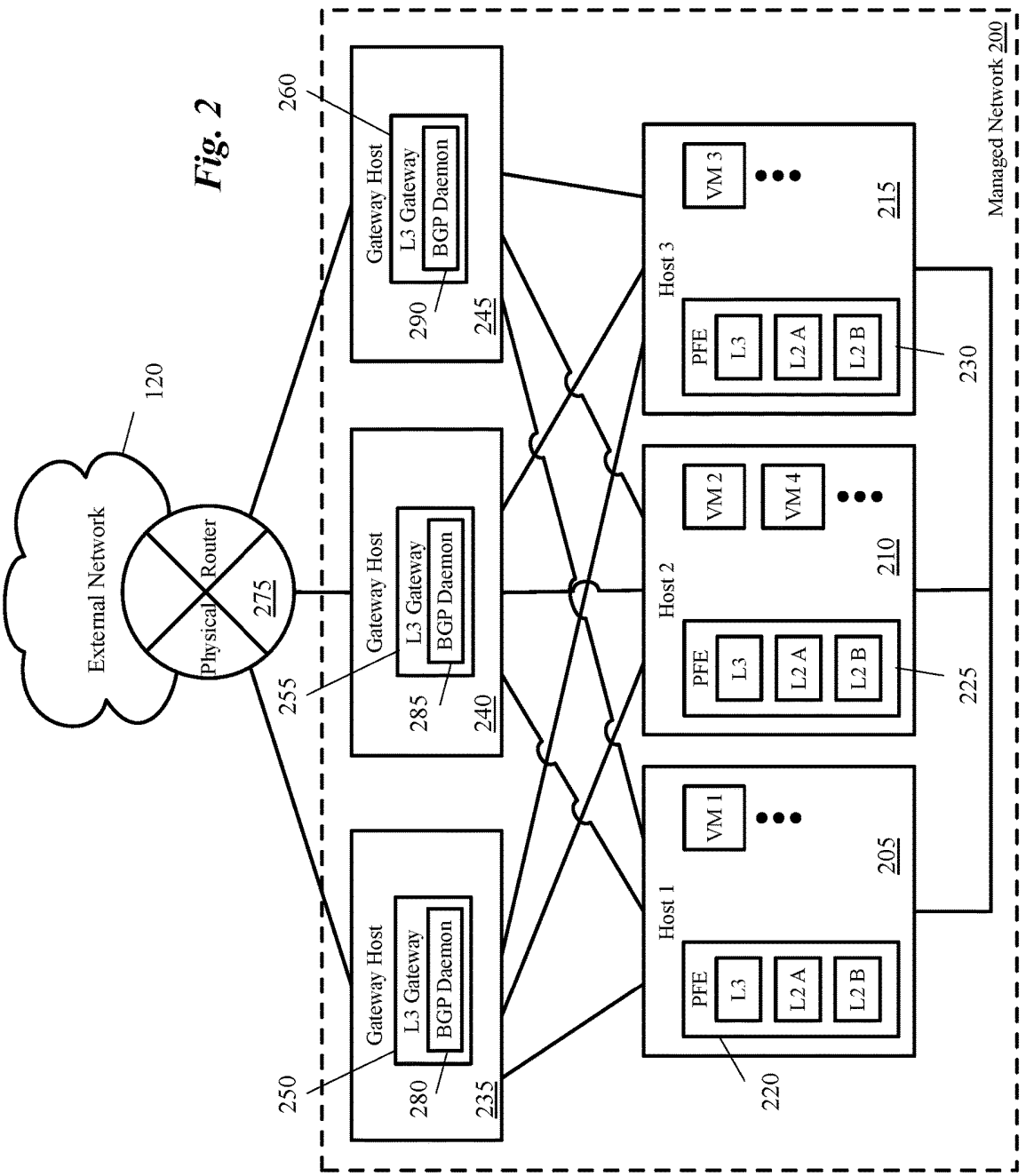

External Network
120
Physical Router
275
Managed Network 200
Gateway Host
L3 Gateway
BGP Daemon
290
245
260
285
240
255
280
235
250
Host 1
Host 2
Host 3
VM 1
VM 2
VM 3
VM 4
PFE
L3
L2 A
L2 B
205
210
215
220
225
230

Fig. 2

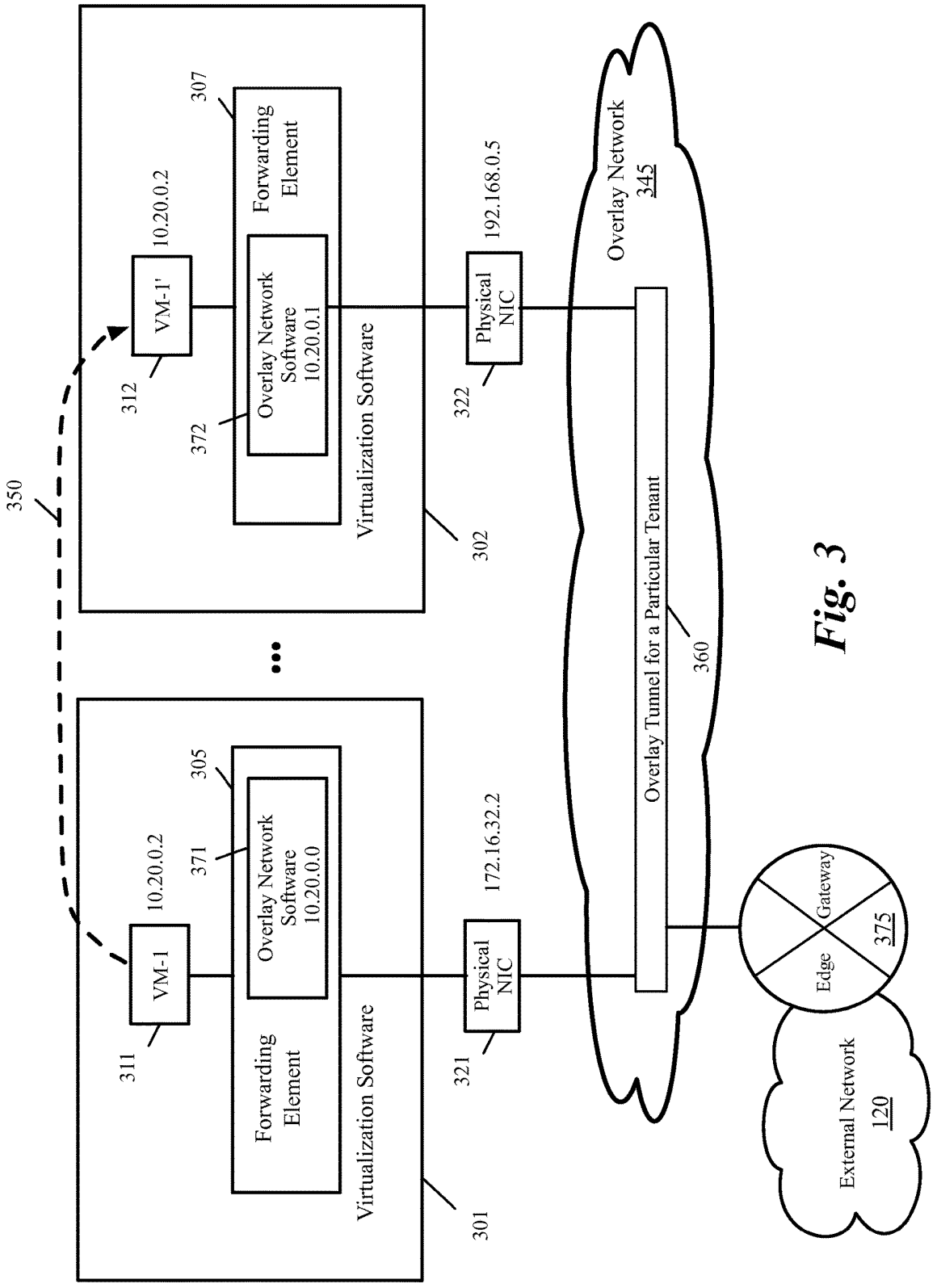

350
VM-1
10.20.0.2
311
Forwarding Element
305
Overlay Network Software 10.20.0.0
371
Virtualization Software
301
Physical NIC
172.16.32.2
321
VM-1'
10.20.0.2
312
Forwarding Element
307
Overlay Network Software 10.20.0.1
372
Virtualization Software
302
Physical NIC
192.168.0.5
322
Overlay Network 345
Overlay Tunnel for a Particular Tenant
360
Edge Gateway 375
External Network 120

Fig. 3

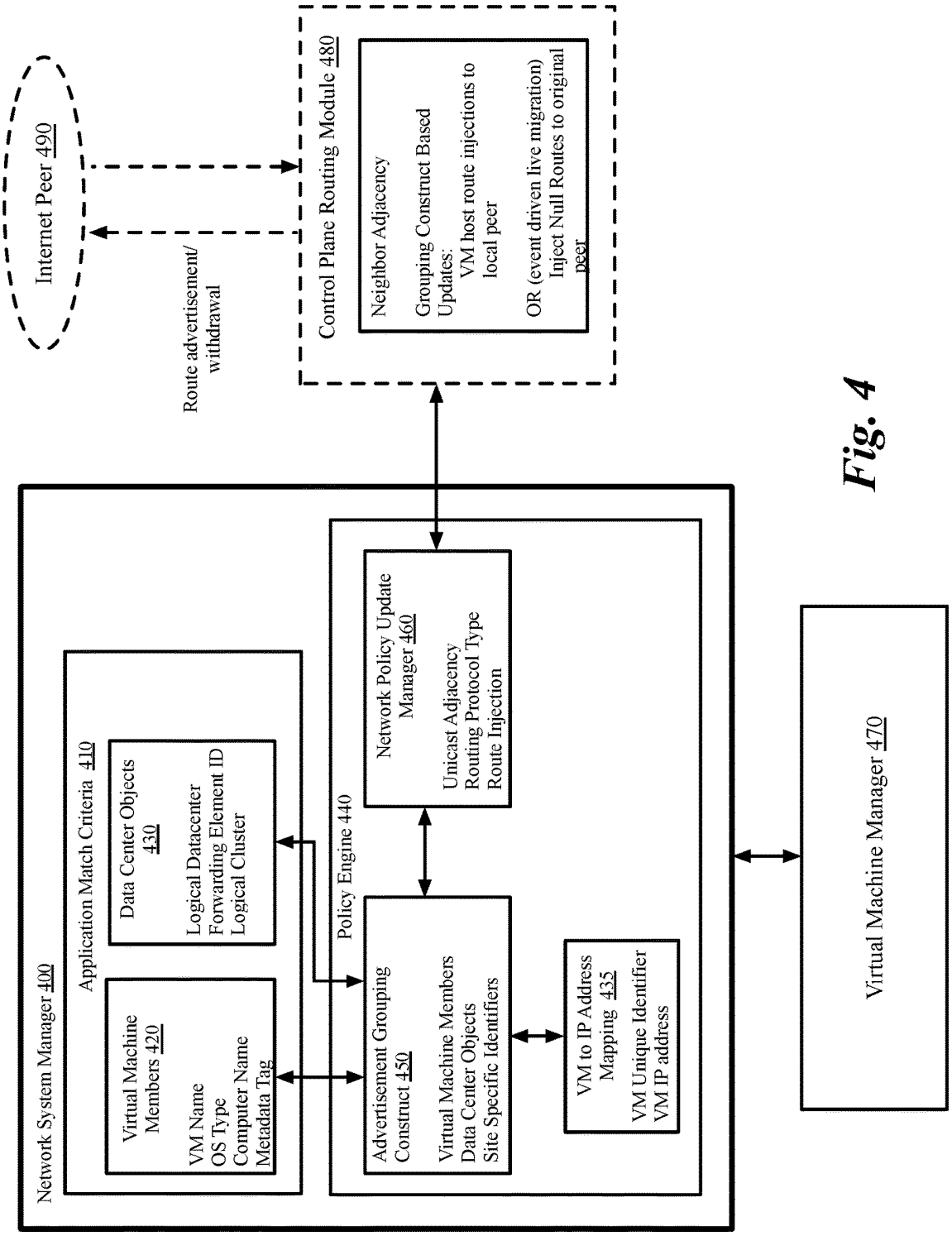

Internet Peer 490
Route advertisement/ withdrawal
Control Plane Routing Module 480
Neighbor Adjacency
Grouping Construct Based Updates:
VM host route injections to local peer
OR (event driven live migration)
Inject Null Routes to original peer
Network System Manager 400
Application Match Criteria 410
Virtual Machine Members 420
VM Name
OS Type
Computer Name
Metadata Tag
Data Center Objects 430
Logical Datacenter
Forwarding Element ID
Logical Cluster
Policy Engine 440
Network Policy Update Manager 460
Unicast Adjacency
Routing Protocol Type
Route Injection
Advertisement Grouping Construct 450
Virtual Machine Members
Data Center Objects
Site Specific Identifiers
VM to IP Address Mapping 435
VM Unique Identifier
VM IP address
Virtual Machine Manager 470

Fig. 4

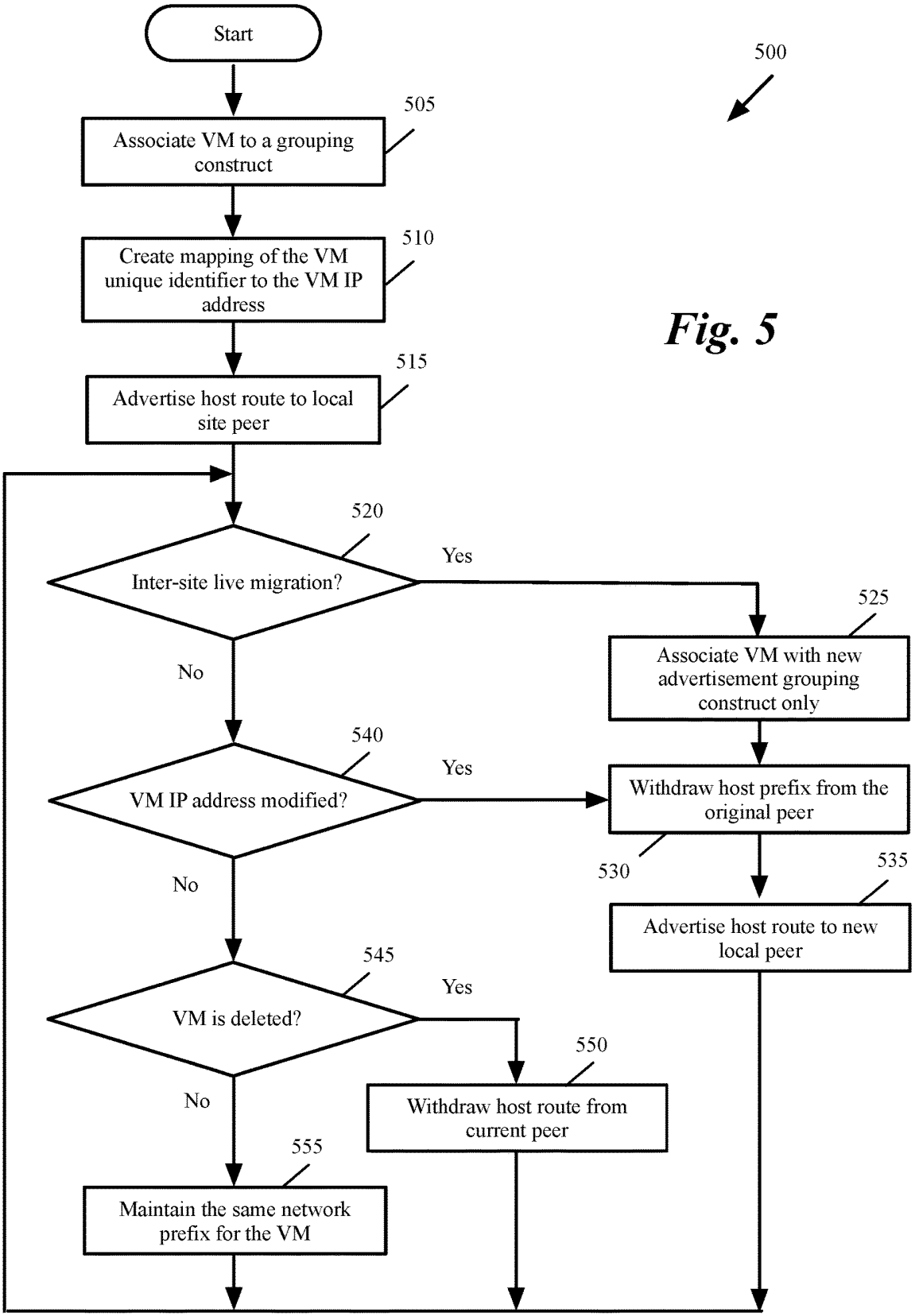

Start
500
505
Associate VM to a grouping construct
510
Create mapping of the VM unique identifier to the VM IP address
515
Advertise host route to local site peer
520
Inter-site live migration?
Yes
525
Associate VM with new advertisement grouping construct only
No
540
VM IP address modified?
Yes
Withdraw host prefix from the original peer
530
535
Advertise host route to new local peer
No
545
VM is deleted?
Yes
550
Withdraw host route from current peer
No
555
Maintain the same network prefix for the VM

Fig. 5

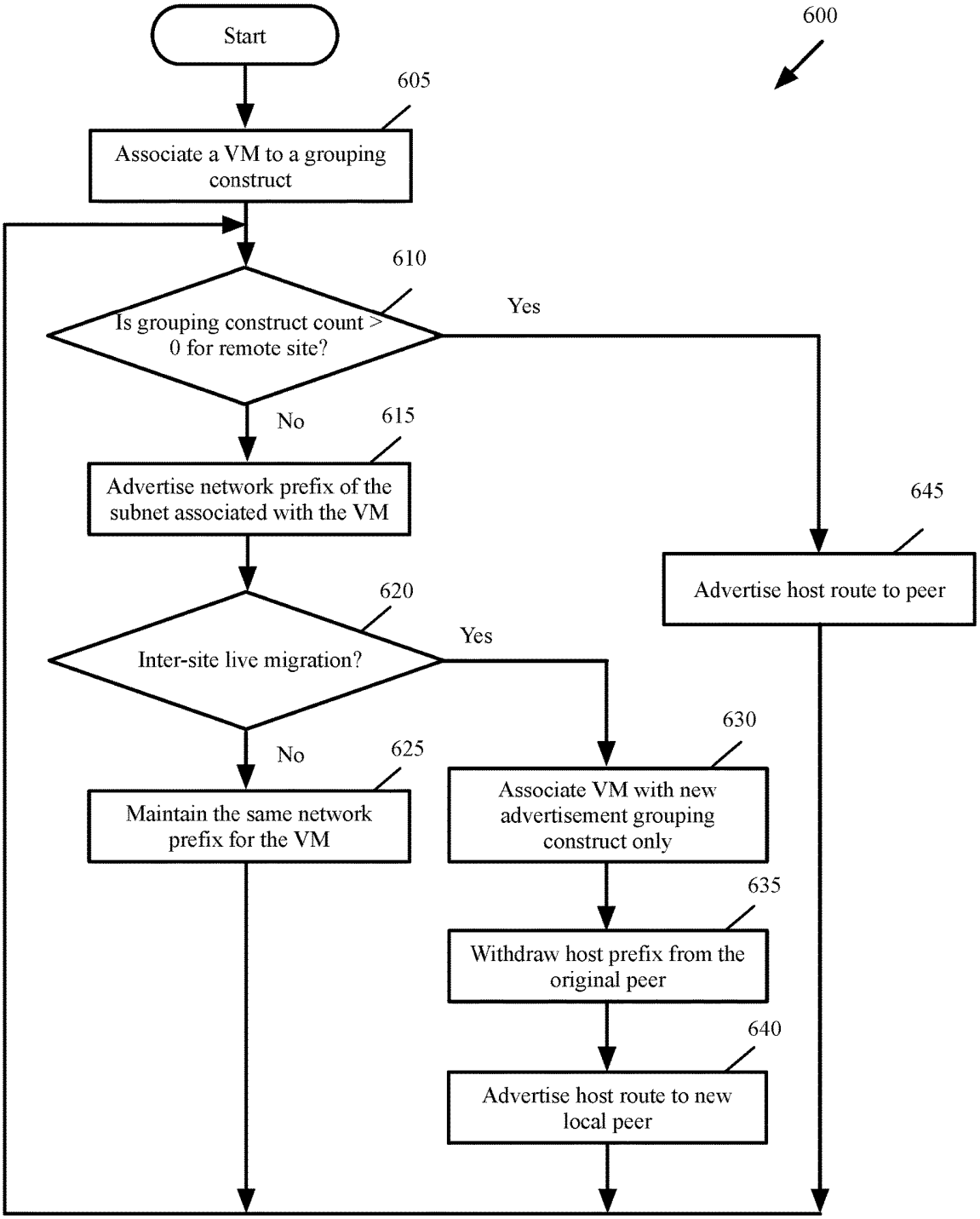

600
Start
605
Associate a VM to a grouping construct
610
Is grouping construct count > 0 for remote site?
Yes
No
615
Advertise network prefix of the subnet associated with the VM
645
Advertise host route to peer
620
Inter-site live migration?
Yes
No
625
Maintain the same network prefix for the VM
630
Associate VM with new advertisement grouping construct only
635
Withdraw host prefix from the original peer
640
Advertise host route to new local peer

Fig. 6

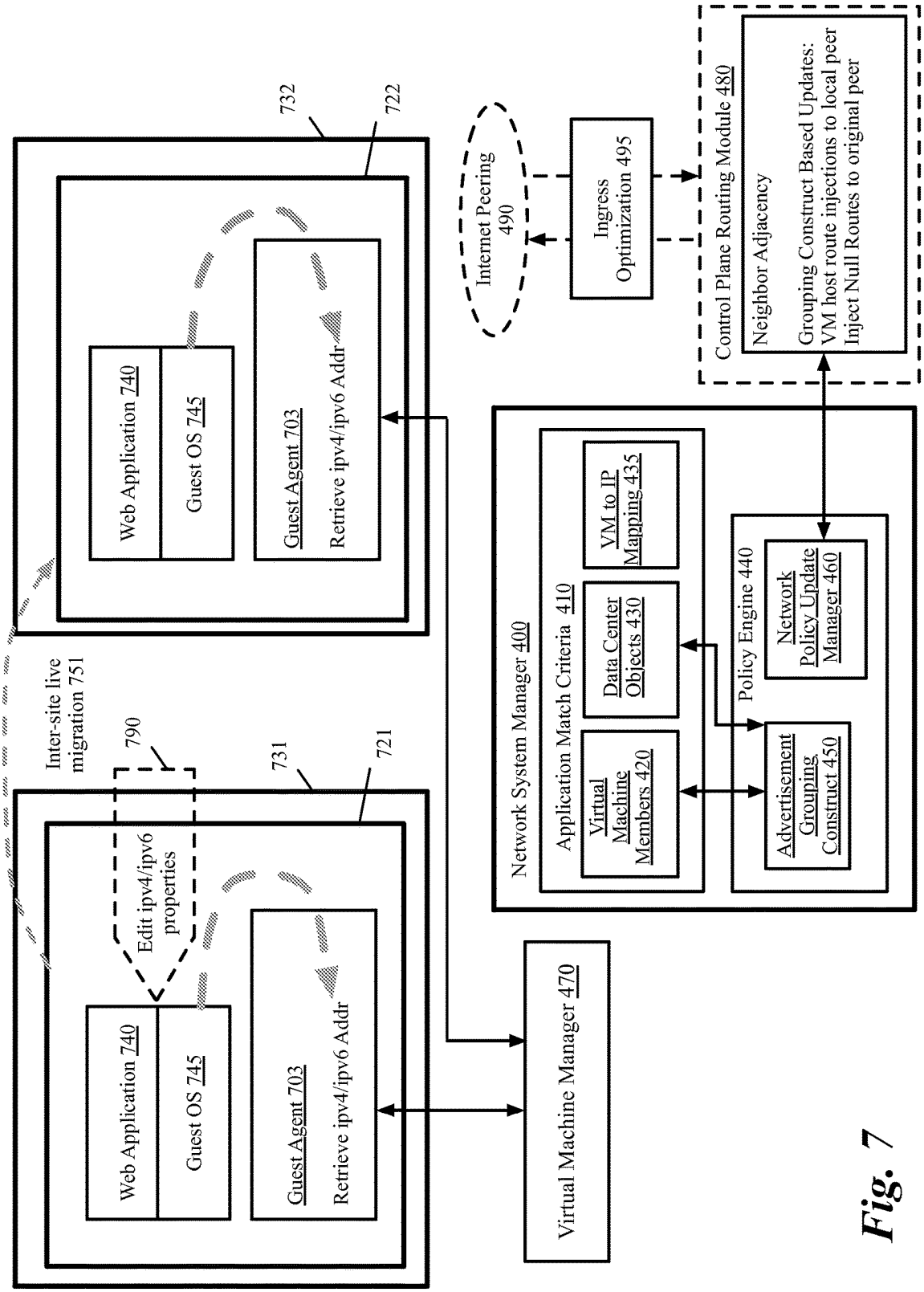

Control Plane Routing Module 480
Neighbor Adjacency
Grouping Construct Based Updates:
VM host route injections to local peer
Inject Null Routes to original peer
Internet Peering 490
Ingress Optimization 495
732
722
Web Application 740
Guest OS 745
Guest Agent 703
Retrieve ipv4/ipv6 Addr
Network System Manager 400
Application Match Criteria 410
Virtual Machine Members 420
Data Center Objects 430
VM to IP Mapping 435
Policy Engine 440
Advertisement Grouping Construct 450
Network Policy Update Manager 460
Inter-site live migration 751
790
731
721
Edit ipv4/ipv6 properties
Virtual Machine Manager 470

Fig. 7

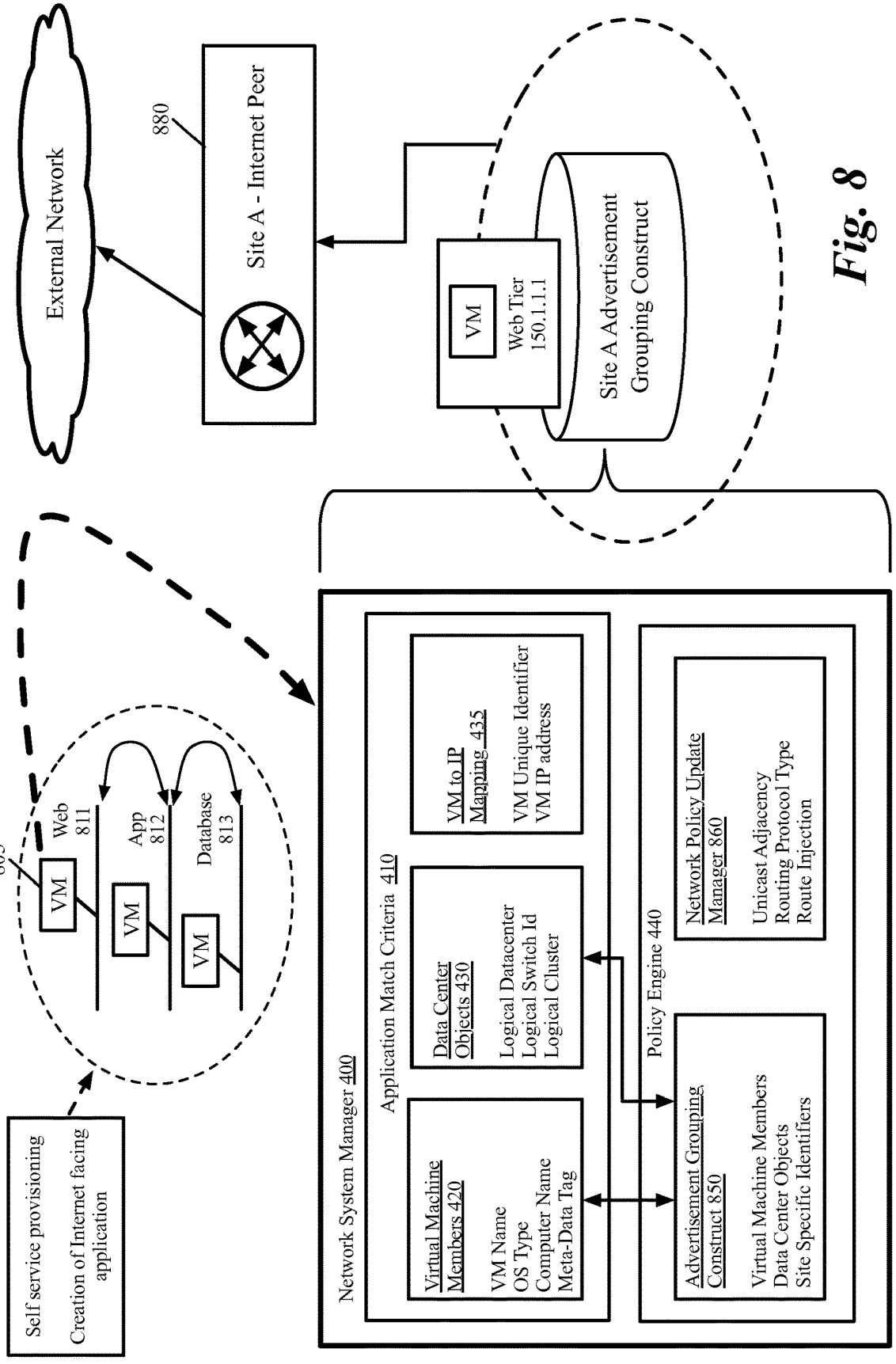

External Network
880
Site A - Internet Peer
VM
Web Tier
150.1.1.1
Site A Advertisement
Grouping Construct
805
VM
Web
811
VM
App
812
VM
Database
813
Self service provisioning
Creation of Internet facing application
Network System Manager 400
Application Match Criteria 410
Virtual Machine
Members 420
VM Name
OS Type
Computer Name
Meta-Data Tag
Data Center
Objects 430
Logical Datacenter
Logical Switch Id
Logical Cluster
VM to IP
Mapping 435
VM Unique Identifier
VM IP address
Policy Engine 440
Advertisement Grouping
Construct 850
Virtual Machine Members
Data Center Objects
Site Specific Identifiers
Network Policy Update
Manager 860
Unicast Adjacency
Routing Protocol Type
Route Injection

Fig. 8

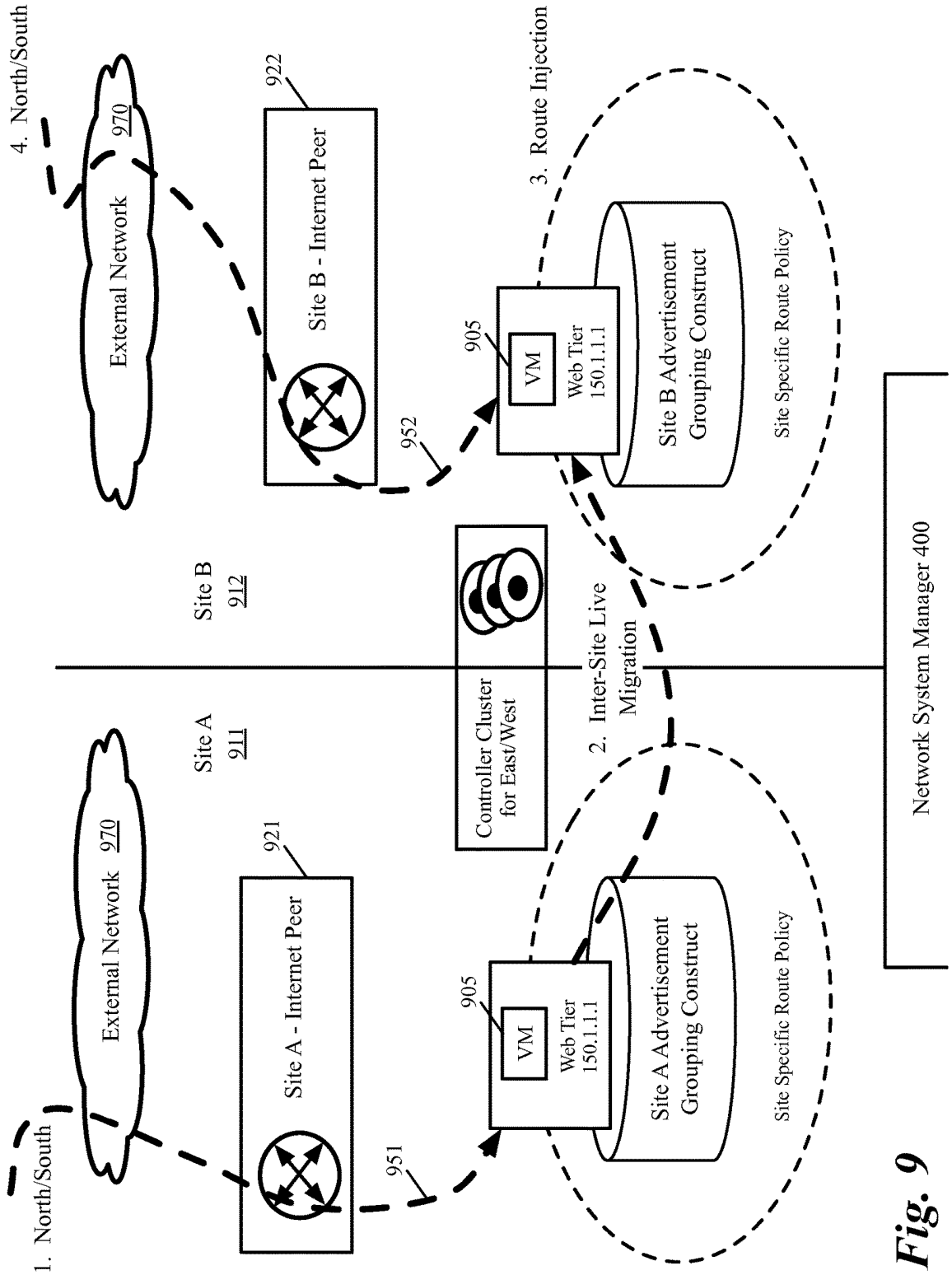

1. North/South
External Network 970
Site A - Internet Peer
921
951
Site A
911
905
VM
Web Tier
150.1.1.1
Site A Advertisement
Grouping Construct
Site Specific Route Policy
Controller Cluster
for East/West
2. Inter-Site Live
Migration
Network System Manager 400
Site B
912
905
VM
Web Tier
150.1.1.1
Site B Advertisement
Grouping Construct
Site Specific Route Policy
3. Route Injection
952
Site B - Internet Peer
922
External Network 970
4. North/South

Fig. 9

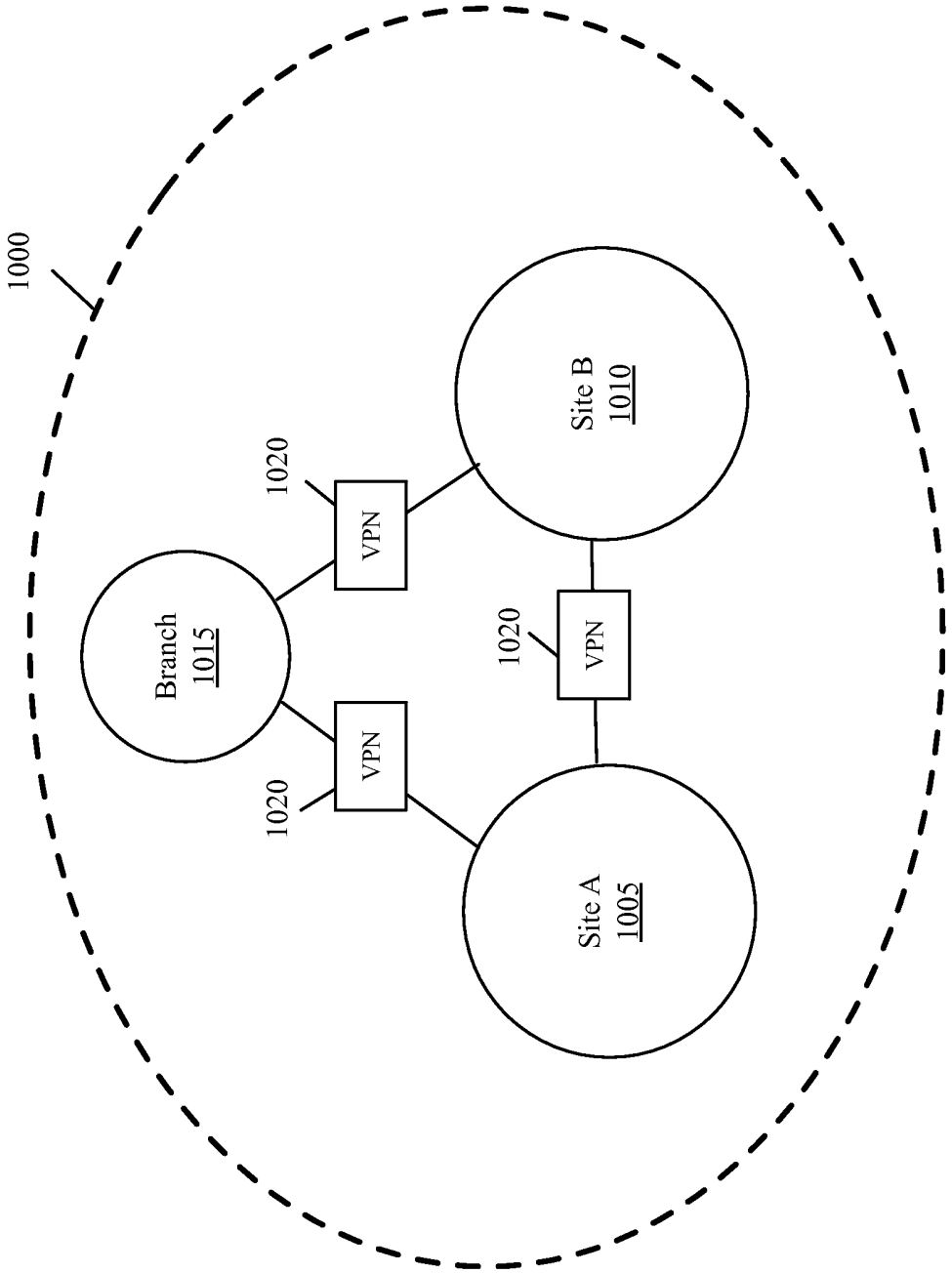

1000
Site B
1010
1020
VPN
Branch
1015
1020
VPN
1020
VPN
Site A
1005

Fig. 10

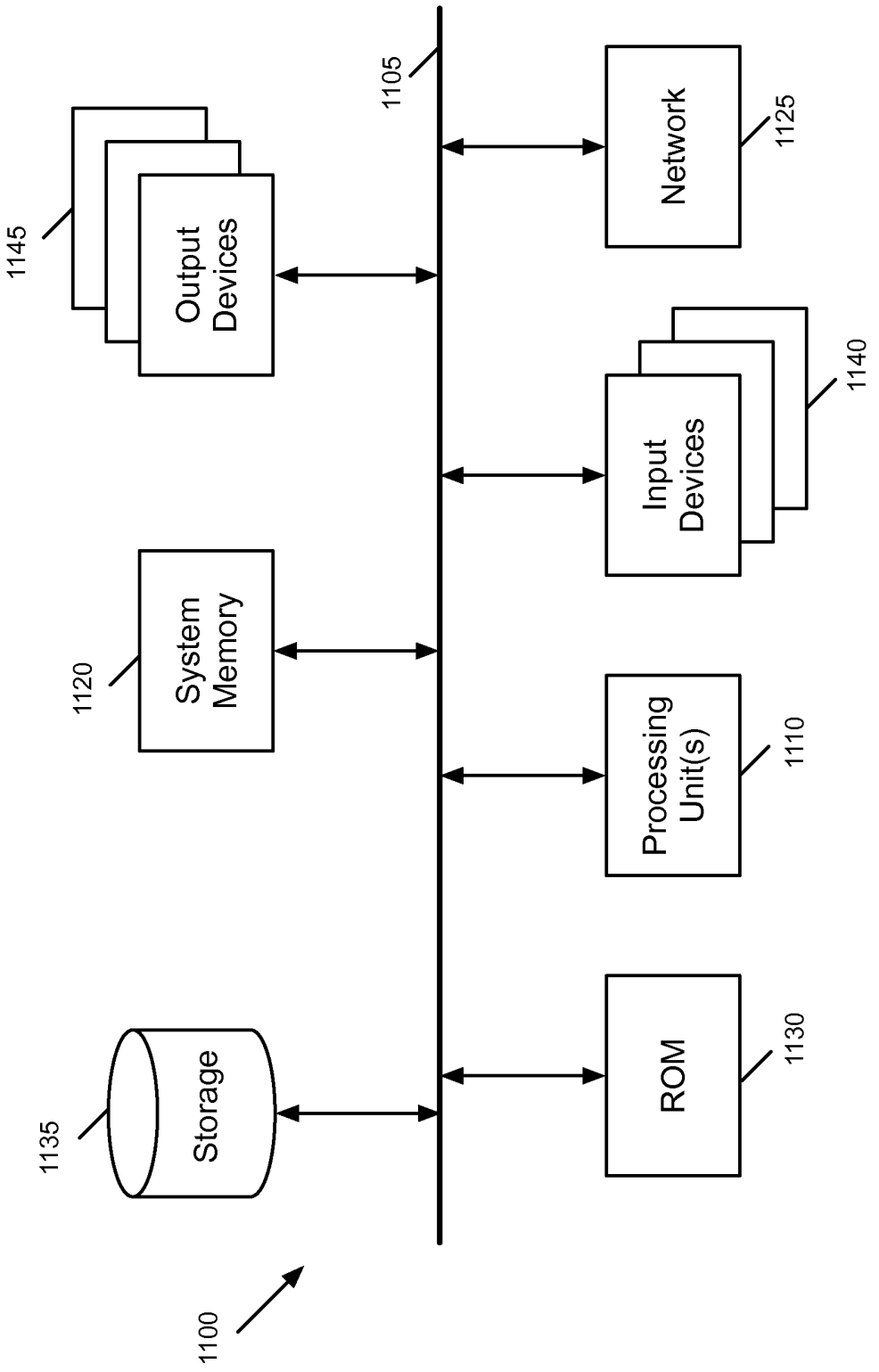

1100
1105
Network
1125
Output Devices
1145
Input Devices
1140
System Memory
1120
Processing Unit(s)
1110
Storage
1135
ROM
1130

Fig. 11

DYNAMIC VIRTUAL MACHINE NETWORK POLICY FOR INGRESS OPTIMIZATION

BACKGROUND

Software defined data centers (SDDCs) virtualize the physical network to provide isolated virtual networks for respective tenants. The innovations in virtual machine mobility live migrations from a networking perspective have allowed virtual machines to extend Open Systems Interconnection (OSI) model Layer 3 boundaries. Based on the specific use cases, the creation of active/active data centers allows inter-site reachability by emulating layer 2 technologies such as Overlay Transport Virtualization (OTV) or Virtual Extensible LAN (VXLAN) as well as the ability to dynamically migrate virtual machines between sites.

One challenge in supporting live migration is that active/active data centers have site-specific routing domains that form adjacency to the local Internet peer. This is a complication because specific virtual machine applications that are required to attract traffic externally could be subject to migration to a secondary site, which causes suboptimal ingress paths. Another challenge is that ingress tunnel techniques with existing solutions such as Locator/Identifier Specification Protocol (LISP) require specific hardware elements with proprietary line cards that are placed throughout routing domains to create location mappings.

BRIEF SUMMARY

Some embodiments provide a method of performing ingress traffic optimization for active/active data centers. The method creates site-specific grouping constructs for virtual machines that run applications that are advertised to the external networks. A grouping construct can include any type of network, compute, or security constructs. In some embodiments, one or more of the compute constructs, network constructs, and security constructs can be specified as dynamic grouping constructs that can have members dynamically added and/or removed from them. The virtual machines at each site are associated with the site-specific grouping constructs. Each site-specific grouping constructs includes a list of the virtual machines currently located at the site as well as a unique identifier of the site. Each virtual machine in a grouping construct is identified by a unique identifier such as a metadata tag or the virtual machine's unique name. The IP addresses of each virtual machine is retrieved from the guest operating system and a network policy is generated to advertise the IP addresses of the virtual machines to the site's routing peer.

Each site is associated with a network. A virtual machine that is provisioned at a first site is assigned an IP address in the network associated with the first site. When the virtual machine migrates from the first site to a second site that is associated with a different network, the virtual machine's unique identifier is removed from the first site's grouping construct and is added to the second site's grouping construct. The network policies for the first and second sites are updated such that the IP address of the virtual machine is removed from the first site's routing peer advertisement and is advertised along with the IP addresses of other virtual machines at the second site by the second site's routing peer. As a result, the ingress packets addressed to VM are directly received at the second site's routing peer despite the fact that the virtual machine's IP address belongs to a different network than the network at the second site.

The preceding Summary is intended to serve as a brief introduction to some embodiments of the invention. It is not meant to be an introduction or overview of all inventive subject matter disclosed in this document. The Detailed Description that follows and the Drawings that are referred to in the Detailed Description will further describe the embodiments described in the Summary as well as other embodiments. Accordingly, to understand all the embodiments described by this document, a full review of the Summary, Detailed Description and the Drawings is needed. Moreover, the claimed subject matters are not to be limited by the illustrative details in the Summary, Detailed Description and the Drawing.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features of the invention are set forth in the appended claims. However, for purposes of explanation, several embodiments of the invention are set forth in the following figures.

FIG. **1** conceptually illustrates a logical network architecture of some embodiments that includes a logical router.

FIG. **2** conceptually illustrates a physical implementation of the logical network of FIG. **1**.

FIG. **3** conceptually illustrates migration of a VM from a host to another in some embodiments when the two hosts are not connected to the same subnet.

FIG. **4** conceptually illustrates a system for classifying the site residencies of VMs and advertising the VMs routes through a dynamic routing policy in some embodiments of the invention.

FIG. **5** conceptually illustrates a process for route advertisement in some embodiments of the invention.

FIG. **6** conceptually illustrates an alternative process for route advertisement in some embodiments of the invention.

FIG. **7** conceptually illustrates a use case for live migration of a VM between two hosts located in two different subnets in some embodiments.

FIG. **8** conceptually illustrates a use case for providing network agility for the consumption of information technology services.

FIG. **9** conceptually illustrates a use case for an inter-site live migration of a VM in some embodiments of the invention.

FIG. **10** conceptually illustrates a multi-site datacenter in some embodiments.

FIG. **11** conceptually illustrates an electronic system with which some embodiments of the invention are implemented.

DETAILED DESCRIPTION

In the following detailed description of the invention, numerous details, examples, and embodiments of the invention are set forth and described. However, it will be clear and apparent to one skilled in the art that the invention is not limited to the embodiments set forth and that the invention may be practiced without some of the specific details and examples discussed.

Some embodiments provide a method of ingress traffic optimization in a network virtualization environment. Virtualization is the ability to simulate a hardware platform, such as a server, storage device or network resource, in software. A virtual machine (VM) is a software implementation of a machine such as a computer. Network virtualization provides a software-based view of the hardware and software networking resources such as logical switches, routers, firewalls, load balancers, and virtual private net-

works (VPNs) to connected workloads. The virtual networks are programmatically created, provisioned and managed, while the underlying physical networks are responsible for forwarding the packets. Network and security services are allocated to each VM according to its needs, and stay attached to the VM as the VM moves among hosts in the virtualized infrastructure domain.

FIG. **1** conceptually illustrates an example of a logical network architecture **100**. The logical network **100** includes two logical forwarding elements (LFEs) **105** and **110** and a logical router **115**. Each of the LFEs **105** and **110** connects several virtual machines. In this example, two virtual machines (VMs) are connected by each LFE and the logical router **115** connects the two LFE (i.e., logical layer 2 domains) together. In addition, the logical router connects the logical network to an external network **120**, via three logical ports. While in this example, the logical router **115** has several ports connecting to the external network (e.g., as uplink ports). In some embodiments the logical router only has a single port that connects to the external networks.

In some embodiments, the logical network is an abstract conception of a network generated by an administrator, and the logical network is implemented in a virtualized, distributed manner in a managed physical infrastructure (e.g., in a multi-tenant datacenter). That is, the virtual machines that connect to the LFEs may reside on various different host machines within the infrastructure, and physical forwarding elements (PFEs) (e.g., software virtual switches) operating on these host machines implement some or all of the logical forwarding elements (logical switches, logical routers, etc.). In the virtualization field, some refer to software switches as virtual switches as these are software elements. However, in this specification, the software forwarding elements are referred to as PFEs, in order to distinguish them LFEs, which are logical constructs that are not tied to the physical world. A PFE forwards packets in a physical network whether or not it is implemented in software while an LFE forwards packets in a logical network, which is logically decoupled or abstracted from the physical network. In other words, the software forwarding elements are referred to as PFEs because they exist and operate in the physical world, whereas an LFE is a logical representation of a forwarding element that is presented to a user when designing a logical network.

A logical router, such as logical router **115**, connects a set of LFEs to which VMs logically attach. Each LFE represents a particular set of IP addresses (i.e., a subnet), and is implemented in the managed network across a set of PFEs to which the VMs physically connect (e.g., through virtual interfaces). In some embodiments, the logical routers are implemented in a distributed fashion as well by the PFEs that connect to the virtual machines. However, when the logical router also connects to the external network via one or more ports, these connections to the external network are implemented through the use of one or more gateways. The gateways, in some embodiments, are responsible for both sending data traffic from the managed network to the external unmanaged physical network and processing traffic sent from the external network into the managed network.

FIG. **2** conceptually illustrates such a physical implementation of the logical network **100**. This figure illustrates a managed network **200** that includes three host machines **205-215** and three gateway host machines **235-245**. The VMs of the logical network **100** reside on the hosts **205-215**, implemented on top of virtualization software (e.g., a hypervisor, virtual machine monitor, etc.) that operates in the host. Additional virtual machines that connect to other logical networks may reside on some or all of these hosts, as well as additional hosts in the managed network that are not shown in this figure.

In addition to the virtual machines, each of the hosts **205-215** operates a physical forwarding element (PFE) **220-230**. In some embodiments, this PFE is a virtual switch that operates within the virtualization software of the host (e.g., Open VSwitch, or another software forwarding element). In the example illustrated in FIG. **2**, the PFEs **220-230** each implement both of the LFEs **105** and **110**, as well as the logical router **115**. This enables first-hop logical processing in some embodiments, in which all or most of the logical processing for a packet is performed at the first PFE that receives the packet. Thus, a packet sent from VM 1 to VM 4 would be processed, by the PFE **220**, through LFE **105** to logical router **115** and then to LFE **110**. The PFE **220** would identify the logical egress port of LFE **110** for the packet as the port to which VM 4 attaches, and map this egress port to a tunnel to the PFE **230** at host **210**.

In some embodiments, a network controller (or controller cluster) provisions the PFEs **220-230** by generating flow entries, or data tuples that the PFE converts into flow entries. These flow entries specify matching conditions (e.g., physical ingress port, logical ingress port, destination MAC or IP addresses, transport layer 5-tuples, etc.) and actions to take on a packet that matches the conditions (e.g., assign packet to a LFE, assign a logical egress port, write data to register, encapsulate in a particular tunnel, etc.). Thus, in order for the PFE to process a packet through the logical network, the PFE matches the packet to a first flow entry, performs the action (e.g., to modify the packet or store logical context data in a register for the packet), resubmits the packet in order to match another flow entry, etc.

The gateway host machines **235-245** of some embodiments host L3 gateways **250-260** for the logical network **100** that implement the connections between the external network **120** and the logical network **100** (specifically, the logical router **115**). When the physical router **275** receives a packet with a destination address that corresponds to one of the VMs of the logical network **100**, or a public IP shared by the VMs on a logical switch, the physical router **275** sends the packet to one of the gateway hosts **235-245**. The gateway hosts **235-245** also include PFEs, and in some embodiments these PFEs receive packets from the physical router **275** and hand off the packets to the L3 gateway in their respective host for processing.

In some embodiments, a user (e.g., an administrator) configures the logical network **100**. Upon receiving such a configuration with several logical router ports connecting to the external network, a network controller (or controller cluster) selects the set of gateway host machines **235-245** for implementing this connection. Specifically, some embodiments select a different gateway host machine for each of these logical router ports. In some embodiments, these gateways are spread across clusters of gateways in the network, such that each port is implemented in a different failure domain. The network controller calculates a routing table for the logical router, a portion of which is implemented at the gateway host machines and a portion of which is implemented by the PFEs (e.g., the PFEs **220-230** and those on the gateway host machines **235-245**).

The L3 gateways **250-260** implement the portion of the routing table of the logical router **115** for north-south traffic (i.e., traffic sent into and out of the managed network). Some embodiments only handle ingress traffic, with outgoing traffic sent through other means (e.g., by a direct connection between the PFEs in the host machines **220-230** and the

physical router **275** or other network elements in the external network **120**). In other embodiments, the L3 gateways handle both ingress and egress traffic.

As shown, the L3 gateways **250-260** each include a Border Gateway Protocol (BGP) daemon **280-290**. These daemons **280-290** peer with the external physical router **275** and advertise routes to this router for the logical router **115**. In some embodiments, the BGP daemons **280-290** operates in the same way as a traditional physical router in terms of its exchange of information with its neighbors. For example, these BGP daemons may open a BGP session with the physical router **275**, send keep-alive messages as specified by the protocol, and advertise its routes to the physical router **275** via BGP packets. In some embodiments, the BGP daemons also receive BGP packets sent by the physical router **275**, and uses these packets to identify routes. The BGP daemons of some embodiments either install the routes in their local routing tables (i.e., within the same namespaces), push the routes up to the network controller so that the network controller can calculate new routing tables for all of the L3 gateways implementing the logical router, or a combination thereof. In other embodiments, however, the BGP daemon only effectively works in one direction, sending out routes to its neighbors (to attract ingress traffic) but not installing routes received from those neighbors. That is, the BGP daemon neither pushes the received routes up to the network controllers nor installs the routes in the routing table at the local gateway.

In some embodiments, the peering of the L3 gateways with the physical router(s) is a user-specified property of the logical ports. In some embodiments, when the user (e.g., administrator) specifies for the logical router to peer with external routers, the controller generates routing protocol data based on the logical network configuration. For each port of the logical router that faces the external network, the controller identifies (i) the set of external routers with which the gateway implementing the port will peer (that is, its neighbors) and (ii) the set of routes that the gateway implementing the port will advertise. These routes may be simply the IP prefixes representing the logical switches that connect to the logical router, or may additionally include other routes input by the user or dynamically generated by processes that implement the logical router. In some embodiments, different ports of the logical router may advertise their routes to different external network routers. Once the network controller generates this data, along with the routing table data for the L3 gateway, the network controller distributes the data to the gateways (e.g., through a hierarchy of network controllers).

In some embodiments, the L3 gateways **250-260** are virtualized containers that have the ability to store a routing table, such as namespaces. In addition, the BGP daemons **280-290**, or other routing protocol applications, operate within these containers according to the data received from the controllers. One or more daemons may operate on the gateway host machine outside of the containers (e.g., in the virtualization software of the gateway) in order to receive data tuples from the controller that define both the routing tables and the BGP configuration for a particular namespace. This daemon or daemons operate to instantiate the namespace, provision the namespace with the routing table, and start the BGP daemon in the namespace. In addition, the daemon(s) generate a configuration file for the BGP daemon in some embodiments, and store the configuration file (e.g., in a file system of the host machine) for access by the routing protocol application. Upon installing its configuration file, the BGP daemons begins communication with its external router neighbors.

In the example shown in FIG. **2**, multiple gateways **235-245** that implement the same logical router **115** (e.g., implementing different ports) advertise the same routes (e.g., to reach VMs on the logical switches **105** and **110**) to the same edge network element such as external router **275**. The external router, in some embodiments, views these multiple L3 gateways as equal-cost next-hops for packets sent to the advertised addresses. As such, the external routers in some embodiments spread packets sent to those destinations across the various gateways that advertise the routes. The external routers may use any of a variety of different equal-cost multi-path (ECMP) techniques to determine to which gateway a packet should be sent.

In the above-described embodiments, the routing protocol application resides inline. That is, the application operates on the gateway, which is the location through which packets are sent. In some embodiments, however, the network controller(s) act as a route server for the gateways, and the routing protocol application resides on the controller. In this case, the routing protocol configuration is not distributed to the gateways by the controller, but instead used to instantiate the routing protocol application on the controller. The controller then advertises the routing information to the external routers (and potentially receives advertised routing information from the external routers). This advertised information informs the external routers as to which gateways to use for which routes. As in the inline case, the external routers may use ECMP techniques to distribute packets sent to the logical network between the several gateways.

One aspect of virtualization is the ability to live migrate VMs from one host computer to another without turning off the VM. Live (or stateful) migration enables dynamic load balancing among the physical host machines. In addition, live migration allows scheduled maintenance of hosts by migrating the VMs from a host before the host is taken offline for maintenance. FIG. **3** conceptually illustrates migration of a VM from a host to another in some embodiments when the two hosts are not connected to the same subnetwork (subnet). For simplicity, other VMs hosted by hosts **301-302** and some components of the virtualization software are not shown. The examples of embodiments of the invention are described below using Internet Protocol (IP) addresses. However, the same principles apply to any network protocol that uses packet routing.

An IP address includes two fields: a network prefix and a host identifier. The network prefix is the most significant portion of the IP address and identifies where a device is connected (e.g., identifies a router at the edge or the entry point of a network). The host identifier identifies who the host is (e.g., identifies an individual device or interface on a particular network). All devices in the same network (or subnet) have the same network prefix. The network prefix is expressed in Classless Inter-Domain Routing (CIDR) notation, which expresses the network prefix followed by a slash character ("/"), followed by the length of the prefix in bits. For instance, in Internet Protocol Version 4 (IPv4) the IP addresses include 32 bits and 172.16.0.1/20 indicates that 20 bits of the IP address are allocated for the subnet and the remaining 12 bits are used to identify individual hosts on the subnet. In the following discussions, IPv4 IP addresses are used in several examples. The discussions are similarly applicable to other IP protocols such as Internet Protocol version 6 (IPv6).

In the example of FIG. **3**, physical NICs **321** and **322** are connected to each other through an overlay network **345**. An overlay network is a network virtualization technology that achieves multi-tenancy in cloud computing environment. Examples of overlay networks include Virtual eXtensible LAN (VXLAN), Generic Network Virtualization Encapsulation (GENEVE), and Network Virtualization using Generic Routing Encapsulation (NVGRE). For instance, VXLAN is an OSI model Layer 2 (L2) overlay scheme over a Layer 3 (L3) network. VXLAN encapsulates an Ethernet L2 frame in IP (MAC-in-UDP encapsulation) and allows Virtual Machines (VM) to be a part of virtualized L2 subnets operating in separate physical L3 networks. Similarly, NVGRE uses Generic Routing Encapsulation (GRE) to tunnel L2 packets over L3 networks. For brevity, the rest of this specification uses the example of VXLAN as an overlay network that is a Layer 2 overlay scheme over a Layer 3 network. It should be understood that other overlay networks (which, e.g., may use other encapsulation protocols) be implemented instead of those specifically mentioned here.

VXLAN uses MAC Address-in-User Datagram Protocol (MAC-in-UDP) encapsulation to extend Layer 2 segments across a data center network. The MAC-in-UDP adds a VXLAN header to the original payload (i.e., the Layer 2 frame) and encapsulates them in a UDP-IP packet. The MAC-in-UDP encapsulation is then used to tunnel Layer 2 network over Layer 3 network.

As a part of the VXLAN configuration, a virtual tunnel endpoint (VTEP) is configured on every host. The VTEPs are used to map tenants' end devices to VXLAN segments and to perform VXLAN encapsulation and decapsulation. Each VXLAN network is assigned a unique identifier referred to as virtual network identifier (VNI). Each host VTEP is assigned a unique IP address. The VTEP uses this IP address to encapsulate Ethernet frames and transmit the encapsulated packets to the transport network through the IP interface.

In some embodiments, a host that participates in overlay network includes overlay network virtualization software that covers the configuration/control plane, data plane and overlay network packet processing functionality. The hosts include (e.g., as a part of the forwarding element **305-307** software) the overlay network virtualization software **371-372** that acts as a tunnel end point. In the example of VXLAN overlay network, the overlay network virtualization software is referred to as VXLAN Virtual VTEP software and the tunnel end point is known as VTEP. There is a unique identifier associated with each instance of an overlay network. The unique identifier associated with each VXLAN network is referred to as the VXLAN Network Identifier (VNI). The overlay network software **371-372** are responsible for maintaining a mapping table to include, e.g., VM media access control (MAC) to tunnel end point address and other address mappings. The overlay network traffic between a tenant's VMs that are hosted on separate hosts is passed through a tunnel **360** through the overlay network **345**.

The overlay network **345** is connected through an edge network element such as an edge gateway **375** to the external network **120**, which may include the Internet, local area networks (LANs), wide area networks (WANs), different hardware equipment such as cables, routers, switches, etc. Although the overlay network endpoints **371-372** are shown as a part of the corresponding forwarding elements **305-307** in the embodiment of FIG. **3**, in other embodiments the overlay network endpoint is implemented as a virtual bridge in the virtualization software **315**, an overlay aware VM application, or an overlay network capable switching hardware.

The VMs of a tenant (only one shown) form a logical network (also referred to as private network or virtual network), which links tenant VMs on host **301** with tenant VMs on host **302**. The logical network is an abstraction of a physical network and may provide a virtual Layer 2 (or data link layer) for services such as encapsulation and decapsulation of network layer data packets into frames, frame synchronization, medial access control, etc. The logical network may span one or more physical networks and be organized independent of the underlying physical topology and organization of the physical networks. The logical network is identified by a logical network identifier (also known as virtual network identifier or VNI). Each logical network is configured by a tenant.

As shown in FIG. **3**, VM-1 **311** that is hosted by host **301** is migrated (as shown by the arrow **350**) to host **302**. A copy, VM-1' **312**, of VM-1 is created on host **302**. The routing tables associated with the hosts are updated to indicate the presence of the VM's MAC address at the new site.

Once VM-1' starts running on host **302**, VM-1 is taken offline and live migration is complete. In the example of FIG. **3**, VM-1 **311**, Host **301**, and physical NIC **321** are on the same subnet, 172.16.32.0/20. On the other hand, host **302** and physical NIC **322** are on a different subnet, 192.168.0.0/24.

Live migration of VM-1 **311** from host **301** to host **302** requires that all network services be available to VM-1' **312** at the new host **302**. In order not to reconfigure the VM on the new host, the packets sent to the VM at address 172.16.32.0 have to be delivered to the VM at the new site. The overlay network tunnel endpoints **371** and **372** are on the same subnet 10.20.0.0/24. As a result, although VM-1' **312** with IP address 10.20.0.2 is hosted on host **315** that is on a different subnet, the packets addressed to 10.20.0.2 are delivered to the new location of the VM (i.e., VM-1') through the overlay network. VTEP **372** delivers the packets to forwarding element **207**, which in turn delivers the packets to VM-1' **312**.

Utilizing the system of FIG. **3** in an active/active data center causes several challenges. Active/active data centers provide disaster recovery by creating and storing real-time (live) copies of data. Active/active data centers, however, have site-specific routing domains that form adjacency to the local routing peer. The adjacency is from the edge network element to the site-specific physical network element, which then advertises the routes over IP wide area network (WAN) or other External networks. This is a challenge as VM live migration takes place, the advertisement of the route for the site-specific element will still attract traffic to the original VM, which has already been migrated causing sub-optimal path.

Network virtualization uses encapsulations as a transport mechanism between layer 3 boundaries. This mechanism creates reachability through the respective tunnel encapsulation between sites. As described below, some embodiments create a new method that is consumed through management application programming interfaces (APIs). The method automates provisioning of applications, which require to be advertised to site-specific WAN and Internet. Coupling this method with the existing functionality of network virtualization creates a novel function to perform ingress traffic optimization between active/active data centers over an agnostic transport network.

Another challenge is that ingress tunnel techniques with existing solutions such as Locator/Identifier Separation Protocol (LISP) require specific hardware device and proprietary line cards placed throughout routing domains to create location mapping. While the solution provided by LISP was effective in traditional networking, the manual aspects of creating LISP enabled nodes is not required in network virtualization as the functionality architecturally shifts the network decision making closer to applications. Furthermore, LISP is dependent on specific network hardware and line cards. Manual configuration of roles is also required to identify LISP sites with the respective namespaces/mappings, as well as non LISP sites (i.e., external networks). The LISP approach requires network elements in the data path to be LISP enabled which makes ingress traffic engineering solutions very challenging to be infrastructure agnostic.

Some embodiments provide a novel method of performing ingress traffic engineering for active/active data centers while utilizing a network virtualization solution. Some of these embodiments classify the site residencies of VMs (i.e., the physical site or data center where a VM resides) that run applications that are required to be advertised to external networks (or to internal/corporate networks) across private routing domains using VPN's or Layer 2 extensions to other sites (e.g., data center sites and branch office sites). The VMs are associated with a site-specific grouping construct that is advertised through a dynamic routing policy. The method simplifies ingress optimizations without the need for manual configuration within the network domains and provides abstractions through y placing homogeneous applications in grouping constructs irrespective of the corresponding subnet and location.

I. Dynamic Virtual Machine Network Policy for Ingress Optimization

FIG. **4** conceptually illustrates a system for classifying the site residencies of VMs and advertising the VMs' routes through a dynamic routing policy in some embodiments of the invention. As shown, the system includes a network system manager **400**. Network system manager **400** utilizes an application match criteria **410** and a policy engine **440**. For simplicity the following discussions assume there is a one-to-one correspondence between the VMs and the applications that run on the VMs and need to be advertised to external networks or internal/corporate networks (i.e., networks in a different subnet). However, when there is not a one-to-one mapping between these applications and VMs (i.e., multiple external facing applications are running on a VM), the TCP/IP protocol stack is utilized to identify the applications. Since each externally facing applications requires a different IP address, the IP address of each application is utilized to identify the application and satisfy the application match criteria. Some embodiments provide a guest agent (a process that runs inside the guest) that are utilized to retrieve the IP address of each application. The guest agent provides visibility within a VM's guest operating system to network information such as IP address, domain name system (DNS) name, etc.

As an example, in a classic three tier application (i.e., presentation tier (or web application), application tier (or business logic), and data tier (or database)), only the frontend presentation tier is externally facing. Once the frontend (or web) application is qualified by meeting the criterion of the application match criteria, the VM identification is passed to an advertisement grouping constructs (or advertisement group) for ingress optimization. A grouping construct is a construct that can include any type of network, compute, or security constructs. Each grouping construct has an identifier. The advertisement grouping construct in some embodiments includes VM members identifiers, data center objects, site specific identifiers, etc. In some embodiments, one or more of the compute constructs, network constructs, and security constructs can be specified as dynamic grouping constructs that can have members dynamically added and/or removed from them. In some embodiments, a member is added to a grouping construct by adding an identifier of the member to the grouping construct. In FIG. **4**, the unique identifier of VM is retrieved from the advertisement grouping construct and is sent to virtual machine manager **470**, which then retrieves the VM IP address through the guest agent (e.g., virtual machine manager **470** sends a REST call that includes the unique identification of the VM to retrieve the VM IP address). The VM is then mapped with its unique identifier and IP address and populated in a table **455** within the advertisement grouping construct.

As shown in FIG. **4**, application match criteria **410** includes a virtual machine members list **420** and a data center objects list **430**. Policy engine **440** includes one or more advertisement grouping constructs **440**, one or more network update policies managers **460**, and a VM to IP address mapping list **435**. As described below, the virtual machine members list **420** and data center objects list **430** are utilized to match against the advertisement grouping constructs **440**.

The virtual machine members list **420** includes information that is used to uniquely identify a VM. The information includes one or more of the VM name, fully qualified domain name (FQDN) of the corresponding application, guest operating system (DNS, computer name), and metadata tag. The data center objects list **430** includes a list of logical datacenters, forwarding elements identifications, and logical clusters. The matching is done based on the characteristics of the VMs (e.g., VM name), guest operating system, metadata tags, and any related objects that connect to VMs such as forwarding elements, logical data centers, logical cluster, etc. For instance, if the application running on a VM is a web tier application, then the guest operating system is searched (e.g., by using the aforementioned guest agent) and the FQDN of the application is retrieved. As another example, if there is a metadata tag associated with the VM, the metadata string is put into the corresponding grouping construct as the unique identifier of the VM. Once a VM the unique identifier of a VM is included in an advertisement grouping construct, the unique identifier of the VM (being mapped to the FQDN, VM name, metadata, related objects connected to the VM, etc.) is retrieved from the grouping construct and is used to obtain the VM IP address (e.g., using the guest agent or derived through inspecting source IP field within packets from the virtual Ethernet port). For instance, if the VM is associated with metadata tag "Web-DMZ" and the VM is at site A, the VM identifier, which is created by the virtual machine manager, is put in site A grouping construct. The unique identifier can then be used to retrieve the IP address of the VM.

The application match criteria **410** is used to filter the specific applications that require network policy update policy. Not every application in different sites needs to be filtered, as some applications may not require network policy update (e.g., the application are not external facing). Also, some applications may be static and never subjected to change, therefore do not need network policy update. Some embodiments subscribe to events (i.e., adding/deleting/modifying of VMs) in advertisement grouping constructs. The subscription to these events is used to detect whether there has been live migration of VMs between sites. An

event is created whenever a new VM is added to a grouping construct, the IP address of a VM is modified, or VM has been deleted from a grouping construct. These subscribed events are used to determine what network updates are needed per site. For instance, if the unique identification of a VM that was in Site A is added to the advertisement grouping construct of Site B, then this event will trigger the unique identification of the VM to be deleted from the advertisement grouping construct of Site A. The VM's routes are subsequently withdrawn from site A router and advertises into site B router.

The advertisement grouping construct **450** is a grouping construct that decouples applications from their physical network addressing. The advertisement grouping construct can be matched against the virtual machine members list **420** and/or data center objects list **430**. Once a VM has a match to an advertisement grouping construct, the virtual machine physical residence is queried against the network systems manager **400** to complete the match.

Depending on the system configuration, placement of VMs in an advertisement grouping construct may require matching against a site-specific identifier (SSI), to ensure host route advertisement is performed at the local site. The SSI is an identification that maps VMs to a specific data center site. The SSI can be the virtualization software unique identification (e.g., the universally unique identifier (UUID) of the virtualization software), applied to each physical host (virtualization software/hypervisor). The SSI can also be location specific identifier applied to each physical host (virtualization software/hypervisor) to provide logical groupings between sites. The SSI is used to associate the VM to a specific site.

As an example, consider a system with two sites: site A (i.e., physical data center A) and site B (i.e., physical data center B). The system only requires two grouping constructs (one per site). Initially (i.e., before the live migration of any VM), all VMs in site A are associated with Site A grouping construct, and all VMs in site B are associated with site B grouping construct. If a virtualization design did not organize the elements such that it is possible to identify which host is in site A and which host is in site B through clustering definitions, then the SSI of the host is associated with the grouping construct. If the virtualization design did define each compute cluster or logical datacenter as belonging to Site A or Site B then there is no need for an SSI and the cluster/logical datacenter objects are utilized as conditional expression to determine the VM's physical Location.

After the live migration of a VM, one or more of the following matching criteria are checked to determine whether the VM remains in the same grouping construct or is moved to another grouping construct. The VM's name, metadata, FQDN, associated forwarding element, logical cluster, logical datacenter, etc., are used to determine the VM's current site. If the VM used to be in Site A and the matching criteria indicates that the VM has moved to Site B, then the VM is added to Site B grouping construct and removed from Site A grouping construct. Otherwise, if the matching criteria indicate that the VM has not moved to a new site (e.g., the live migration was done to a host in the same data center site) the VM remains in the same grouping construct. As the example shows, a grouping construct can have VMs that are on different subnets, i.e., the placement of the VMs in a grouping construct depends on the current site residency as opposed to the subnet associated with the VMs. In other words, the embodiments of the invention decouple the virtual local area network (VLAN) and subnets.

Once the application/VM is associated with a grouping construct, the unique identifier of the VM is retrieved from the grouping construct and is mapped to the VM IP address. As shown in FIG. **4**, the VM to IP mapping list **435** is used to dynamically keep track of VM to IP address mapping per site. The VM to IP mapping is the key information needed to interact between the network policy update manager **460** and control plane routing module **480**. Some embodiments maintain a VM to IP address mapping table **435** for each site (e.g., one table for Site A and one table for Site B). Each table can be viewed as a dictionary and holds a mapping of the VM's unique identification to VM's IP Address for the particular site.

The network policy describes what is required to be done when a VM requires a change in routing advertisement (i.e., when a VM is first provisioned, a VM is deleted, or a VM is moved from one site to another). The policy includes specific instruction to update the site-specific network element with /32 network prefix of the VM. The policy carries this information in a data-structure, which is communicated to the control plane routing module **480**. The data structure has all the specific instructions such as forming unicast adjacency with site local border network element (which connects to the external/WAN network) as well as advertisement/withdraw messages.

In some embodiments a protocol such as border gateway protocol (BGP) is used, with autonomous system numbers, password, and other peering information. The network policy update manager **460** then interacts with control plane routing module **480** over the same API carrying BGP instructions to announce and withdraw network routes depending on the site residency of the VM. In other embodiments, the network policy update manager **460** communicates to the control plane routing module **480** using XML REST API to program the site-specific border network element.

In some embodiments, the network policy update manager **460** operates in four modes: (1) when new applications are provisioned, the network policy update manager **460** does the initial injection of routes, (2) if long distance live migration (i.e., VM migration from one subnet to another) takes place, the network policy update manager **460** withdraws routes from Site A and advertises in Site B, (3) if the VM IP address changes, the network policy update manager **460** re-advertises the VM address, and (4) if VM is deleted, the VM's prefix is also removed from the border router.

The advertisement grouping constructs group together the applications/VMs that require specific network advertisement to the site local peer irrespective of their traditional network segmentation/addressing. Route advertisement policy provided by the network policy update manager **460** is associated with the advertisement container **450** to create a list of VMs, which are required for network advertisement. The network policy update manager makes an API call to the control plane routing module **480** with the VMs' IP addresses which then advertises the route addition or withdrawal to the site-specific peer **490** through the preferred routing protocol (e.g., BGP). In an alternate embodiment, the network systems manager **400** has an API that has the data structure of the advertisement container, which then updates a hardware network device with the list of VMs that require advertisement to the Internet/WAN. The IP addresses of the VMs are retrieved, e.g., through the guest agent.

If an inter-site live migration occurs, the VM is queried against the SSI (or other unique identifiers such as data center logical cluster identification, etc.). If the SSI matches the alternate active site, the VM is included in the new site's

advertisement container only (a VM cannot be a part of two network grouping constructs at the same time). The network systems manager **400** sends an update to the control plane routing module **480** to withdraw route entry to the first site and inject /32 host route to the new site.

FIG. **5** conceptually illustrates a process **500** for route advertisement in some embodiments of the invention when a VM is live migrated to a new site, the VM IP address is modified, or the VM is deleted from the site. Process **500** in some embodiments is performed by a network system manager (e.g., by policy engine **440** of the network system manager **400** in FIG. **4**). As shown, the process associates (at **505**) a VM to a container. The process then creates (at **510**) a mapping of the VM's unique identifier to the VM IP address. The process then advertises (at **515**) host route to the local site peer.

The process then determines (at **520**) whether an inter-site live migration for the VM has occurred. In some embodiments, the process subscribes to the live migration events and is notified of live VM migrations. If the live migration has not occurred, the process maintains the process proceeds to **540**, which is described below.

Otherwise, the process associates (at **525**) the VM with the new advertisement container only (i.e., the process removes the VM from the previous site's container). The process then withdraws (at **530**) the host prefix from the original routing peer. The process then advertises (at **535**) the host route to the new local peer. The process then proceeds to **520**, which was described above.

When the process determines that inter-site live migration has not occurred, the process determines (at **540**) whether the VM IP address is modified. If yes, the process proceeds to **530**, which was described above. Otherwise, the process determines (at **545**) whether the VM is deleted. If yes, the process withdraws (at **550**) the host route from the current peer. The process then proceeds to **520**, which was described above. Otherwise, the process maintains (at **555**) the same network prefix for the VM. The process then proceeds to **520**, which was described above.

Different embodiments provide different options to extend the routing information to control plane routing module **480**. Some embodiments detect the VMs that have migrated to a new site and inject /32 routes into the global routing table of the border network elements which face WAN or external network. In other embodiments (when less entries in the global routing tables are preferred), summary subnets e.g., /24 are advertised and if live migration occurs then /32 is advertised at the new site.

FIG. **6** conceptually illustrates an alternative process **600** for route advertisement in some embodiments of the invention. Process **600** in some embodiments is performed by a network system manager (e.g., by policy engine **440** of the network system manager **400** in FIG. **4**). Process **600** is performed when VMs are added by default to a secondary site or for parsing the existing VMs, which happen to be at the secondary site. As shown, the process associates (at **605**) a VM to a container. The process then determines (at **610**) whether the container count (i.e., the VMs within the site container) for remote site is greater than zero. When the container count is greater than zero, the process advertises (at **645**) the host route (i.e., the /32 address) to the site peer. The process then proceeds to **610**, which is described above.

Otherwise, the process advertises (at **615**) the network prefix of the subnet associated with the VM. The process then determines (at **620**) whether an inter-site live migration for the VM has occurred. In some embodiments, the process subscribes to the live motion events and is notified of live VM migrations. If the live migration has not occurred, the process maintains the same network prefix for the VM (since the VM has not migrated to a different site). The process then proceeds to **610**, which was described above.

Otherwise, the process associates (at **630**) the VM with the new advertisement container only (i.e., the process removes the VM from the previous site's container). The process then injects (at **635**) a Null route to or withdraws network prefix from the original routing peer. The process then advertises (at **640**) the host route to the new local peer. The process then proceeds to **610**, which was described above.

FIG. **7** conceptually illustrates a use case for live migration of a VM between two hosts located in two different subnets in some embodiments. As shown, a VM **721** is migrated (as shown by the line **751**) from host **731** to host **732**. A copy **722** of the VM is created on host **732**. The VM includes a web application **740** and a guest operating system **745**.

Each VM includes a guest agent **703**. One advantage of the system described herein is the ability to identify VM IP addresses. Traditionally the virtual router or switch had to inspect packets and examine the source IP field within the IPv4 header to determine the VM IP address. The current invention however extracts a string data type through the guest agent to derive contextual data such as the VM IP address, DNS name, etc. VMs are then matched against the VMs' operating type, VM Name (start of string), custom metadata tags/labels, and different datacenter objects connected to the VMs including forwarding elements, clusters, data centers, etc. Once a container is defined that matches one or more of the above, a second query against the VMs that has matched the container is performed for the VMs' IP addresses through the guest agent. Another important aspect of the invention is maintaining site awareness. A site-specific identification can then match the container that could either be cluster identification or a specific host that is identified as a particular site. The route policy bound to this container is then injected by route advertisement. In another embodiment specific to route injection, an XML/REST based API with a potential physical networking element with decoupled control plane (that could be API driven) transports the defined application grouping constructs which allows to express routing injections.

Another advantage of the system described herein is the ability to update network advertisements in case the business user/administrator edits the IPv4 properties within the guest operating system. Prior art methods require manual network changes in the system configuration if the VM address is subject to change. As shown in FIG. **7**, the IPv4 address is changed (as shown by **790**) through the guest agent and due to its existing capability of matching against an abstracted format, the route policy then updates the physical elements on the /32 route.

FIG. **8** conceptually illustrates a use case for providing network agility for the consumption of information technology (IT) services. A three-tier application **805** is provisioned. The three-tier application includes a presentation tier or web application **811**, an application tier or business logic **812**, and a data tier or database **813**. Once the business user decides to use a dynamic network policy, VM **820** that runs the web application **811** is selected for dynamic network policy updates. The characteristics of the VM are used to classify the VM and place the VM in an advertisement container **850**. In some embodiments, a central management API framework provisions VM applications using the respective advertisement policies. The API framework can

be through Simple Object Access Protocol (SOAP) or Representational State Transfer (REST) to add, modify, or delete these grouping constructs.

Network policy update manager **860** then provides dynamic network policy updates (e.g., upon provisioning of the VM on a particular site or live migration of VM **805** to another site) for site advertisement to the Internet peer **880**. As shown, the disclosed system provides flexibility within the consumption of IT services without the complexity and provisioning delay.

FIG. **9** conceptually illustrates a use case for an inter-site live migration of a VM in some embodiments of the invention. In the prior art when an external facing VM (i.e., a VM that can be addressed by external networks) migrates from Site A to Site B, the incoming traffic directed to the VM arrives at Site A, which is then redirected to Site B. The redirection causes a suboptimal indirect path for delivering incoming packet traffic to the VM. As shown in FIG. **9**, when VM **905** is in Site A **911**, the incoming packet traffic is received at Site A Internet peer **921** and is directed to the VM **905** as shown by arrow **951**.

After the live migration of the VM to Site B **912**, the VM is advertised to Site B Internet peer **922** and a withdrawal message is sent to Site A Internet peer **921**. Site A and Site B Internet peers advertise the updated routing policy to the external network **970**. As a result, the ingress packets addressed to VM **905** is directly received at Site B Internet peer **921** as shown by the arrow **952**.

All of the above-mentioned embodiments are applicable for the applications that are Internet facing (i.e., applications that have an IP address that is accessible by the external networks such as the Internet) as well as the applications that are internally facing (i.e., applications that have a private IP address that is not accessible through the external networks such as the Internet). If an application is externally facing and has a public IP address programmed into the guest operating system IPv4 properties. In this instance the public address is advertised to the border element router, which then updates the external networks. When the application has a private IP address and a network address translation (NAT) router does address translation, then the NAT translation is carried to the new site.

The above-mentioned embodiments are also applicable to applications with private addresses that are facing internally (i.e., the application is only accessible through the associated private network). FIG. **10** conceptually illustrates a multi-site datacenter **1000** in some embodiments. As shown, the datacenter **1000** includes sites **1005** and **1010** and one branch site **1015** (e.g., a remote office outside the data centers), which are connected with VPN **1020** or other circuits such as dark fiber layer 2, multiprotocol label switching (MPLS), etc. This means the private addresses are routable within the datacenter **1000** and enterprise network.

The embodiments of the invention provide ingress traffic optimization between these sites while still using private IP on the VM. Although the VMs are not Internet accessible, the VMs are used to service branch offices within the private network. For instance, for a VM that has IP 192.168.1.2, the branch site can reach the VM via Site A **1005**. If the VM live migrates to Site B **1010**, the networks routing tables are updated and the branch site can now communicate to Site B through route techniques described herein. The multi-site datacenters such as datacenter **1000** with many internal branch sites can be part of a private IP range, which is reachable across the private network circuits.

II. Electronic System

Many of the above-described features and applications are implemented as software processes that are specified as a set of instructions recorded on a computer readable storage medium (also referred to as computer readable medium). When these instructions are executed by one or more processing unit(s) (e.g., one or more processors, cores of processors, or other processing units), they cause the processing unit(s) to perform the actions indicated in the instructions. Examples of computer readable media include, but are not limited to, CD-ROMs, flash drives, RAM chips, hard drives, EPROMs, etc. The computer readable media does not include carrier waves and electronic signals passing wirelessly or over wired connections.

In this specification, the term "software" is meant to include firmware residing in read-only memory or applications stored in magnetic storage, which can be read into memory for processing by a processor. Also, in some embodiments, multiple software inventions can be implemented as sub-parts of a larger program while remaining distinct software inventions. In some embodiments, multiple software inventions can also be implemented as separate programs. Finally, any combination of separate programs that together implement a software invention described here is within the scope of the invention. In some embodiments, the software programs, when installed to operate on one or more electronic systems, define one or more specific machine implementations that execute and perform the operations of the software programs.

FIG. **11** conceptually illustrates an electronic system **1100** with which some embodiments of the invention are implemented. The electronic system **1100** can be used to execute any of the control, virtualization, or operating system applications described above. The electronic system **1100** may be a computer (e.g., a desktop computer, personal computer, tablet computer, server computer, mainframe, a blade computer etc.), phone, PDA, or any other sort of electronic device. Such an electronic system includes various types of computer readable media and interfaces for various other types of computer readable media. Electronic system **1100** includes a bus **1105**, processing unit(s) **1110**, a system memory **1120**, a read-only memory (ROM) **1130**, a permanent storage device **1135**, input devices **1140**, and output devices **1145**.

The bus **1105** collectively represents all system, peripheral, and chipset buses that communicatively connect the numerous internal devices of the electronic system **1100**. For instance, the bus **1105** communicatively connects the processing unit(s) **1110** with the read-only memory **1130**, the system memory **1120**, and the permanent storage device **1135**.

From these various memory units, the processing unit(s) **1110** retrieve instructions to execute and data to process in order to execute the processes of the invention. The processing unit(s) may be a single processor or a multi-core processor in different embodiments.

The read-only-memory **1130** stores static data and instructions that are needed by the processing unit(s) **1110** and other modules of the electronic system. The permanent storage device **1135**, on the other hand, is a read-and-write memory device. This device is a non-volatile memory unit that stores instructions and data even when the electronic system **1100** is off. Some embodiments of the invention use a mass-storage device (such as a magnetic or optical disk and its corresponding disk drive) as the permanent storage device **1135**.

Other embodiments use a removable storage device (such as a floppy disk, flash drive, etc.) as the permanent storage

device. Like the permanent storage device **1135**, the system memory **1120** is a read-and-write memory device. However, unlike storage device **1135**, the system memory is a volatile read-and-write memory, such a random access memory. The system memory stores some of the instructions and data that the processor needs at runtime. In some embodiments, the invention's processes are stored in the system memory **1120**, the permanent storage device **1135**, and/or the read-only memory **1130**. From these various memory units, the processing unit(s) **1110** retrieve instructions to execute and data to process in order to execute the processes of some embodiments.

The bus **1105** also connects to the input and output devices **1140** and **1145**. The input devices enable the user to communicate information and select commands to the electronic system. The input devices **1140** include alphanumeric keyboards and pointing devices (also called "cursor control devices"). The output devices **1145** display images generated by the electronic system. The output devices include printers and display devices, such as cathode ray tubes (CRT) or liquid crystal displays (LCD). Some embodiments include devices such as a touchscreen that function as both input and output devices.

Finally, as shown in FIG. **11**, bus **1105** also couples electronic system **1100** to a network **1125** through a network adapter (not shown). In this manner, the computer can be a part of a network of computers (such as a local area network ("LAN"), a wide area network ("WAN"), or an Intranet, or a network of networks, such as the Internet. Any or all components of electronic system **1100** may be used in conjunction with the invention.

Some embodiments include electronic components, such as microprocessors, storage and memory that store computer program instructions in a machine-readable or computer-readable medium (alternatively referred to as computer-readable storage media, machine-readable media, or machine-readable storage media). Some examples of such computer-readable media include RAM, ROM, read-only compact discs (CD-ROM), recordable compact discs (CD-R), rewritable compact discs (CD-RW), read-only digital versatile discs (e.g., DVD-ROM, dual-layer DVD-ROM), a variety of recordable/rewritable DVDs (e.g., DVD-RAM, DVD-RW, DVD+RW, etc.), flash memory (e.g., SD cards, mini-SD cards, micro-SD cards, etc.), magnetic and/or solid state hard drives, read-only and recordable Blu-Ray® discs, ultra density optical discs, any other optical or magnetic media, and floppy disks. The computer-readable media may store a computer program that is executable by at least one processing unit and includes sets of instructions for performing various operations. Examples of computer programs or computer code include machine code, such as is produced by a compiler, and files including higher-level code that are executed by a computer, an electronic component, or a microprocessor using an interpreter.

While the above discussion primarily refers to microprocessor or multi-core processors that execute software, some embodiments are performed by one or more integrated circuits, such as application specific integrated circuits (ASICs) or field programmable gate arrays (FPGAs). In some embodiments, such integrated circuits execute instructions that are stored on the circuit itself.

As used in this specification, the terms "computer", "server", "processor", and "memory" all refer to electronic or other technological devices. These terms exclude people or groups of people. For the purposes of the specification, the terms display or displaying means displaying on an electronic device. As used in this specification, the terms "computer readable medium," "computer readable media," and "machine readable medium" are entirely restricted to tangible, physical objects that store information in a form that is readable by a computer. These terms exclude any wireless signals, wired download signals, and any other ephemeral or transitory signals.

While the invention has been described with reference to numerous specific details, one of ordinary skill in the art will recognize that the invention can be embodied in other specific forms without departing from the spirit of the invention. In addition, a number of the figures (including FIG. **5**) conceptually illustrate processes. The specific operations of these processes may not be performed in the exact order shown and described. The specific operations may not be performed in one continuous series of operations, and different specific operations may be performed in different embodiments. Furthermore, the process could be implemented using several sub-processes, or as part of a larger macro process.

This specification refers throughout to computational and network environments that include virtual machines (VMs). However, virtual machines are merely one example of data compute nodes (DCNs) or data compute end nodes, also referred to as addressable nodes. DCNs may include non-virtualized physical hosts, virtual machines, containers that run on top of a host operating system without the need for a hypervisor or separate operating system, and hypervisor kernel network interface modules.

VMs, in some embodiments, operate with their own guest operating systems on a host using resources of the host virtualized by virtualization software (e.g., a hypervisor, virtual machine monitor, etc.). The tenant (i.e., the owner of the VM) can choose which applications to operate on top of the guest operating system. Some containers, on the other hand, are constructs that run on top of a host operating system without the need for a hypervisor or separate guest operating system. In some embodiments, the host operating system uses name spaces to isolate the containers from each other and therefore provides operating-system level segregation of the different groups of applications that operate within different containers. This segregation is akin to the VM segregation that is offered in hypervisor-virtualized environments that virtualize system hardware, and thus can be viewed as a form of virtualization that isolates different groups of applications that operate in different containers. Such containers are more lightweight than VMs.

Hypervisor kernel network interface module, in some embodiments, is a non-VM DCN that includes a network stack with a hypervisor kernel network interface and receive/transmit threads. One example of a hypervisor kernel network interface module is the vmknic module that is part of the ESXi™ hypervisor of VMware, Inc.

One of ordinary skill in the art will recognize that while the specification refers to VMs, the examples given could be any type of DCNs, including physical hosts, VMs, non-VM containers, and hypervisor kernel network interface modules. In fact, the example networks could include combinations of different types of DCNs in some embodiments.

In view of the foregoing, one of ordinary skill in the art would understand that the invention is not to be limited by the foregoing illustrative details, but rather is to be defined by the appended claims.

What is claimed is:

1. A method of ingress traffic optimization in a datacenter comprising a plurality of sites, each site associated with a different network, the method comprising:

placing a unique identifier of each of a first set of data compute nodes (DCNs) located at a first site in a first route advertisement group associated with the first site;

retrieving Internet protocol (IP) addresses of each DCN in the first set using the unique identifier of the DCN, the IP address of a first DCN in the first set comprising a first network address corresponding to a first network associated with the first site;

advertising the IP addresses of the DCNs in the first route advertisement group to a routing peer associated with the first site;

determining that the first DCN has moved from the first site to a second site, the second site associated with a second network having a second network address different than the first network address;

in response to the determining, placing the unique identifier of the first DCN in a second route advertisement group associated with the second site; and

advertising the IP address of the first DCN along with IP addresses of a second set of DCNs in the second route advertisement group to a routing peer associated with the second site.

2. The method of claim **1**, wherein placing the unique identifier of a DCN in the first route advertisement group comprises placing the unique identifier of a DCN in the first route advertisement group only when the DCN is located at the first site and the DCN satisfies a set of criteria for being placed in a route advertisement group.

3. The method of claim **2**, wherein the set of criteria comprises at least one of the DCN being addressable from a network outside the first site and the DCN is not provisioned to remain at the first site.

4. The method of claim **1**, wherein retrieving the IP addresses of a DCN comprises querying a guest operating system associated with the DCN using the unique identifier of the DCN to retrieve the IP address of the DCN.

5. The method of claim **1** further comprising storing a mapping between the unique identifier and the retrieved IP address of each DCN at the first site, wherein advertising the IP addresses of the DCNs in the first route advertisement group comprises using the unique identifier of each DCN and the mapping to retrieve the IP address of each DCN in the first route advertisement group.

6. The method of claim **1** further comprising:

after determining that the first DCN has moved from the first site to the second site, removing the unique identifier of the first DCN from the first route advertisement group; and

withdrawing the IP address of the DCN from the advertisement to the routing peer associated with the first site.

7. The method of claim **1**, wherein the unique identifier of a DCN comprises one of a unique name associated with the DCN, a fully qualified domain name (FQDN) of an application running on the DCN, an identifier of a guest operating system associated with the DCN, and a metadata tag associated with the DCN.

8. The method of claim **1**, wherein advertising the IP address of the first DCN to the routing peer associated with the second site causes the ingress traffic directed to the first DCN to be delivered to the routing peer associated with the second site despite the IP address of the first DCN having a different network address than the second network connected to the second site.

9. The method of claim **1**, wherein determining that the first DCN has moved from the first site to a second site comprises:

receiving a notification that the first DCN has moved from a first host to a second host; and

determining that the second host is associated with a site with a different site-specific identifier than the first site.

10. The method of claim **1**, wherein the IP address of each DCN comprises a first field comprising said network address identifying a particular sub-network and a second field identifying a host connected to the particular sub-network.

11. A non-transitory machine readable medium storing a program for ingress traffic optimization in a datacenter comprising a plurality of sites, each site associated with a different network, the program comprising sets of instructions for:

placing a unique identifier of each of a first set of data compute nodes (DCNs) located at a first site in a first route advertisement group associated with the first site;

retrieving Internet protocol (IP) addresses of each DCN in the first set using the unique identifier of the DCN, the IP address of a first DCN in the first set comprising a first network address corresponding to a first network associated with the first site;

advertising the IP addresses of the DCNs in the first route advertisement group to a routing peer associated with the first site;

determining that the first DCN has moved from the first site to a second site, the second site associated with a second network having a second network address different than the first network address;

in response to the determining, placing the unique identifier of the first DCN in a second route advertisement group associated with the second site; and

advertising the IP address of the first DCN along with IP addresses of a second set of DCNs in the second route advertisement group to a routing peer associated with the second site.

12. The non-transitory machine readable medium of claim **11**, wherein the set of instructions for placing the unique identifier of a DCN in the first route advertisement group comprises a set of instructions for placing the unique identifier of a DCN in the first route advertisement group only when the DCN is located at the first site and the DCN satisfies a set of criteria for being placed in a route advertisement group.

13. The non-transitory machine readable medium of claim **12**, wherein the set of criteria comprises at least one of the DCN being addressable from a network outside the first site and the DCN is not provisioned to remain at the first site.

14. The non-transitory machine readable medium of claim **11**, wherein the set of instructions for retrieving the IP addresses of a DCN comprises a set of instructions for querying a guest operating system associated with the DCN using the unique identifier of the DCN to retrieve the IP address of the DCN.

15. The non-transitory machine readable medium of claim **11**, the program further comprising a set of instructions for storing a mapping between the unique identifier and the retrieved IP address of each DCN at the first site, wherein the set of instructions for advertising the IP addresses of the DCNs in the first route advertisement group comprises a set of instructions for using the unique identifier of each DCN and the mapping to retrieve the IP address of each DCN in the first route advertisement group.

16. The non-transitory machine readable medium of claim **11**, the program further comprising sets of instructions for:

removing, after determining that the first DCN has moved from the first site to the second site, the unique identifier of the first DCN from the first route advertisement group; and

withdrawing the IP address of the DCN from the advertisement to the routing peer associated with the first site.

17. The non-transitory machine readable medium of claim **11**, wherein the unique identifier of a DCN comprises one of a unique name associated with the DCN, a fully qualified domain name (FQDN) of an application running on the DCN, an identifier of a guest operating system associated with the DCN, and a metadata tag associated with the DCN.

18. The non-transitory machine readable medium of claim **11**, wherein advertising the IP address of the first DCN to the routing peer associated with the second site causes the ingress traffic directed to the first DCN to be delivered to the routing peer associated with the second site despite the IP address of the first DCN having a different network address than the second network connected to the second site.

19. The non-transitory machine readable medium of claim **11**, wherein the set of instructions for determining that the first DCN has moved from the first site to a second site comprises sets of instructions for:

receiving a notification that the first DCN has moved from a first host to a second host; and

determining that the second host is associated with a site with a different site-specific identifier than the first site.

20. The non-transitory machine readable medium of claim **11**, wherein the IP address of each DCN comprises a first field comprising said network address identifying a particular sub-network and a second field identifying a host connected to the particular sub-network.

* * * * *

Appendix B

A Retrospective on Workload Identifiers: From Data Center to Cloud-Native Identifiers

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 12 September 2022, accepted 26 September 2022, date of publication 3 October 2022,
date of current version 10 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3211293

TOPICAL REVIEW

A Retrospective on Workload Identifiers: From Data Center to Cloud-Native Networks

ANDREW BABAKIAN[1,2], PERE MONCLUS[1], ROBIN BRAUN[2], (Life Senior Member, IEEE), AND JUSTIN LIPMAN[2], (Senior Member, IEEE)

[1]VMware Inc., Palo Alto, CA 94304, USA
[2]School of Electrical and Data Engineering, University of Technology Sydney, Ultimo, NSW 2007, Australia

Corresponding author: Andrew Babakian (ababakian@vmware.com)

ABSTRACT As applications move to multiple clouds, the network has become a reactive element to support cloud consumption and application needs. Through each generation of network architectures, identifiers and the use of dynamic locators evolved in different levels of the protocol stack. The identifiers and locators type is defined by the isolation boundary and how the architecture considers semantic overload in the IP address. Each solution is an outcome of incrementalism, resulting in application delivery outgrowing the underlying network. This paper contributes an industrial retrospective of how the schemes and mechanisms for identification and location of network entities have evolved in traditional data centers and how they match cloud-native application requirements. Specifically, there is an evaluation of each application artifact that forced necessary changes in the identifiers and locators. Finally, the common themes are highlighted from observations to determine the investigation areas that may play an essential role in the future of cloud-native networking.

INDEX TERMS Workload identifiers, network locators, naming and addressing, data center, cloud-native.

I. INTRODUCTION

In today's dynamic environment, the network is under pressure to deliver connectivity and identity across multiple clouds in an agile manner. The architectural support for identifiers and locator split has solidified their role as an essential function to keep pace with application growth and flexible cloud computing models.

From a network access-control perspective, IP addresses could be viewed as identity labels for authorization in a network zone model and, in parallel, are routing instructions for networked elements to guide a packet to the destination. In each generation of network architectures, there is a realization that the IP space cannot serve two purposes of identity and location in virtualization and cloud computing environments. A technique to overcome the semantic overload of IP addresses was to split the IP space for persistent identity and dynamic location through encapsulation protocols as a point of indirection. However, as the new generation of cloud applications emerges, the IP space has become weak and mutable to maintain its role as a persistent identifier. For example, microservice applications heavily depend on network proxies, known as sidecars, to provide application availability and resiliency. In such network rendezvous architectures, the IP address has no lasting semantics [28]. An observation is made that the cloud-native architectures have shifted workload identifiers and created new namespaces to support service discovery, routing, and identification for applications to extend administrative boundaries.

This paper contributes to a discussion of the schemes and techniques for identifying and locating network entities that have evolved in traditional data centers and how they match cloud-native application requirements. The distinction this paper makes is that it surfaces how each application artifact forced a shift in workload identifiers required for discovery, routing, and identification, which is not so obvious in past papers from a comparison point of view. For example, the rationale behind identifiers, locators, and insertion

The associate editor coordinating the review of this manuscript and approving it for publication was Nitin Gupta.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

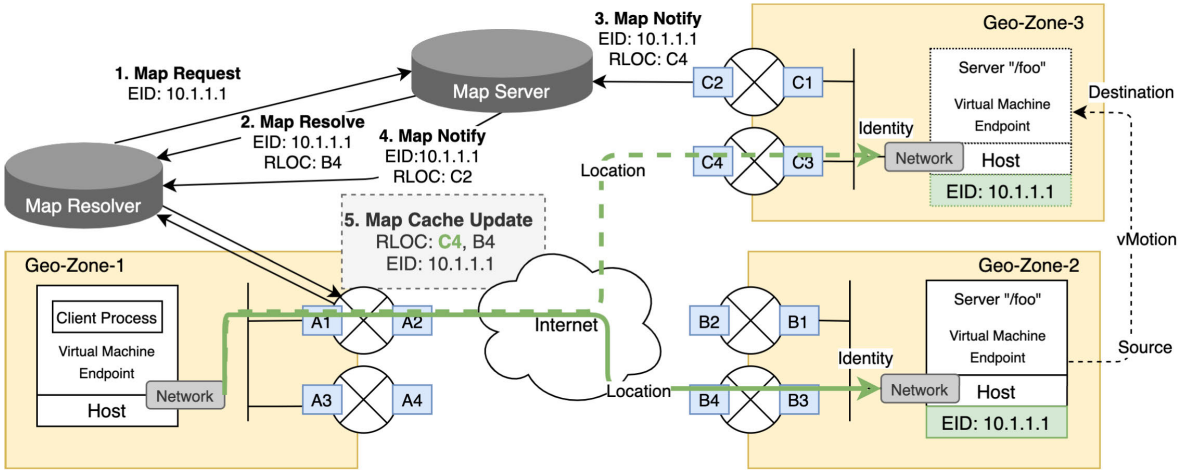


FIGURE 1. LISP Endpoint Identifier and Dynamic Locator.

points required for today's cloud-native networks is emphasized rather than focusing on identity and location split in its generality.

In addition, this paper contributes an increased awareness of evolved cloud application artifacts that lead to the functionality required for Service Mesh and its impact on the use of location-independent identifiers.

Finally, this paper closes by considering the discussed techniques, state-of-the-art opportunities, and directions for future investigation into the critical elements foundational to a cloud-native network.

II. LOCATION/ID SEPARATION PROTOCOL

A. SEMANTIC OVERLOAD IN THE IP ADDRESS

As background, the Internet Architecture Board (IAB) workshop [43], the concerns centered around the current routing information base (RIB) size of the border gateway protocol (BGP) [50]. One of the underlying causes of routing scalability was identifier/locator overload in the IP address. Enterprises were advertising provider-independent prefixes across sites to avoid server renumbering. From a firewall point of view, the server's IP address was its identity; therefore, IP renumbering would be an operational burden.

With the current IP architecture, it seemed impossible to have a single IP space that could serve two purposes, and a split seemed necessary to scale the routing system. This conclusion was the supporting material that led to the creation of the locator/ID separation protocol (LISP) [18]. There are several other proposals to separate identifiers and locators, such as HIP [45], Mobile IP [48] and RINA [29]. To get started, this section selected LISP that addressed the workload mobility challenges traditional in data centers.

The initial technique of LISP was to provide topological aggregation, leveraging a locator approach for the IP routing system. The network's edge contains more specific details needed to reach the endpoint. Provider-independent networks resided behind LISP networks and behaved similarly to network address translation (NAT) [17]. However, in contrast to NAT, the translation state in LISP was not held in a network device but stored in the original IP packet. In LISP, the original IP packet is encapsulated with an outer IP header that contains the locator information.

B. LISP FOR VIRTUALIZED DATA CENTERS AND THE VIRTUAL MACHINE ARTIFACT

While the initial motivation of LISP was to solve a route scalability issue, it became a tool to address mobility challenges created by virtualized data centers (VDCs) [4]. VDCs refer primarily to a hosted virtualization platform, where servers are decoupled from their underlying hardware resource. This disaggregation allowed the infrastructure layer to isolate memory, CPU, and storage to each virtual machine (VM) instance managed by the hypervisor [62]. Applications within the VMs artifact remained unaware of this abstraction. As VDCs became mainstream, VM portability between physical hosts became an appealing function for disaster recovery, operations maintenance, and a method to utilize capacity in active-active data centers (DC).

C. IDENTITY AND THE IP NAMESPACE

LISP naturally addressed the VDCs needs by allowing VM migration between data centers without burdening the routing system. The insertion point of LISP was at the edge of the data center core network, where the endpoint identifiers (EIDs), in this case, VMs, are mapped to routing locations (RLOCs) at the geozone level. The VMs are not globally routable and are topologically independent from the underlying network. For example, in Figure 1 VM "/foo" migrates from Geo-Zone-2 to Geo-Zone-3 without disrupting the existing transport-layer session from a client in Geo-Zone-1. Through LISP control-plane notifications, Geo-Zone-1 receives a Map-Notify to inform that "/foo's" IP address of "10.1.1.1"has migrated to Geo-Zone-3. The

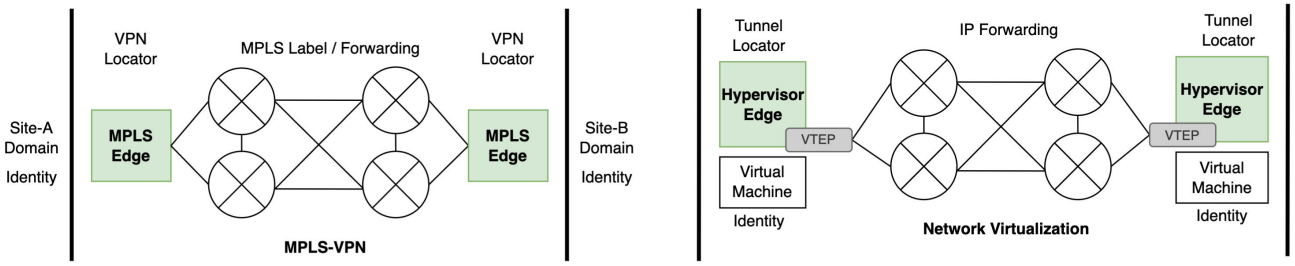


FIGURE 2. Core versus Edge Split: The Newly Revised Edge.

LISP-speaking router "A1" updates "10.1.1.1" entry to map to "C4" in Geo-Zone-3. This map and encapsulation with a dynamic approach allow the VM's identity namespace to remain preserved during the migration procedure.

In summary, LISP introduced a disjoint IP namespace, where IP is separated as an identity (i.e., EID) attribute and IP as used for location. This separated namespace technique in the data center network architecture allowed VMs to preserve their identity while taking advantage of the virtualization operational model, which proved to minimize service disruption based on its mobility usecase. LISP's mapping system introduced a new approach to providing network control at the edge of the data center. The arguments for separation of edge vs. core become an important approach for the IP routing system [30], and for providing a network insertion point to support ephemeral and mobile data center workloads.

III. NETWORK VIRTUALIZATION

A. CORE VS. EDGE SPLIT

In the section above, LISP's identifier mappings are only relevant at the edge (i.e., VMs), while the core of the network is forwarding IP packets to the correct Geo-Zone (i.e., location). This architecture is similar to Multi-protocol Label Switching (MPLS) [54]. In MPLS-VPN [53], the provider core (P) is solely focused on forwarding packets based on MPLS labels, while the provider edge (PE) has specific details of the neighbor domains (i.e., identifier). This separation allowed the PE to create multiple virtualized paths, which cater to overlapping address spaces over the same physical network.

B. NETWORK VIRTUALIZATION FOR CLOUD

In the quest for a purpose-built network for a cloud computing architecture, network virtualization (NV) [36] replicated the same separation of core and edge as demonstrated in the LISP and MPLS design and solved new challenges that cloud computing created.

In the technique of Network Virtualization (NV), the network becomes a reactive element to the application and cloud needs, promulgating the hypervisor vSwitch [49] as the newly revised edge, depicted in Figure 2. This is also known as the virtual network edge (NVE) [37]. Furthermore, cloud computing forced the network architecture to create a unifying abstraction across hypervisors. In contrast to LISP and MPLS-VPN, NV introduced the dynamic creation of virtual networks, each with independent address spaces and topologies that coexist over the same physical network.

In cloud computing, self-service consumption, rapid VM provisioning, and lifecycle management are subsumed in the NV architecture. The NVE allowed tenants to dynamically create network topologies to support the virtualized applications' or VM's lifecycle state. The virtual switch (vSwitch) within the hypervisor becomes the first network control point at the edge. In an NV model, the vSwitches are programmed by a logically centralized control plane responsible for synchronizing the forwarding state across all NVEs. The controller cluster is responsible for the overall state management, including the VM location. In addition, the control plane disseminates the VM identifiers to locator mappings efficiently using an advertisement-based model.

C. IDENTITY AND THE IP NAMESPACE

In the context of NVEs, vSwitches are responsible for providing connections over IP tunnels as paths without modifications to the physical network. While there are many variations of overlays [12], [24], the industry primarily adopted the Virtual eXtensible Local Area Network (VXLAN) [40] encapsulation protocol as the de facto standard in the data center and cloud. VXLAN embeds tenant-specific identifiers for isolation to ensure packets are distinguished between tenants that cohabit on a common platform. VXLAN preserves the original IP space as the VM's identity. The encapsulation contains an outer IP space to be used as a dynamic locator known as virtual-tunnel endpoints (VTEPs), which is a routable interface bound to the hypervisor.

In summary, NV adopted an application first principle, unlike LISP, where the NVE dynamically reacts to the consumption needs of self-service clouds. Rapid VM provisioning and management of lifecycle state requires the control plane to disseminate identifiers and locator mappings to hypervisors on a need-to-know basis. Like LISP, NV created a disjoint IPv4 namespace, separating the role of identifier and locator as independent addresses. Due to the prevalence of internal network-based middleware such as firewalls, the IP address must maintain its role as the core identifier for the VM.

IV. IDENTIFIER LOCATOR ADDRESSING

A. CONTAINER ARTIFACT AND THE ENTERPRISE CLOUD

There has been a great increase in the adoption of delivering applications in the format of containers [3]. It is appealing for cloud-native applications to be delivered in container-based virtulization [61] due to their efficient resource utilization and portability between cloud platforms. There are two new challenges that containers create for networking. Firstly, every container process requires a single IPv4 address, which leads to exhaustion in CIDR allocation due to the scale of processes compared to virtual machines. Secondly, the container's runtime layer is above the hypervisor, requiring a new networking control point.

B. IDENTIFIER LOCATOR ADDRESSING (ILA) NAMESPACE

To address these challenges, identifier locator addressing (ILA) [26] was proposed. Similar to NV described in the previous section, ILA is also an overlay network focused on separating core and edge networks. However, ILA differs from NV in two ways. ILA focuses on the layer above server virtualization, to which NV becomes transparent. In addition, ILA is an IPv6 overlay solution that takes advantage of a larger address space to modify the identifier, thus removing encapsulation overhead that VXLAN and LISP introduced. Its primary motivation is to support container virtualization use cases such as scheduling, task migration, and container orchestration [34], while preserving container identity.

The fundamental principles of ILA were borrowed from earlier work in the identifier-locator network protocol (ILNP) [2] and GSE (8+8) [46], which also addressed the challenges of overloaded semantics in the IP address. The ILNP approach demonstrated a way of using addressing by creating a distinction between the core routing and end systems.

While ILNP had the design properties to fit the dynamic behavior of cloud workloads, the implementation required modifications to the TCP stack to perform dynamic changing of locator bindings. ILA addressed this challenge by implementing an IPv6 [13] transformation technique that ensures the transport layer remains unmodified with an immutable 64-bit identifier and SIR prefix. There is an allocation of a /64 prefix per host representing the locator within the ILA scheme. As described in Figure 3, when a client/server session is initiated between two containers, the ILA host will perform an outside-of-process stateless IP address transformation on the higher-order 64 bits, the locator, to forward packets to the container's destination. On the receiving side, the ILA host rewrites the destination IP address from the locator prefix to the original identifier address and subsequently delivers the packet to the container application.

The control plane of this architecture disseminates identifiers and locator mappings to all ILA host routing elements. The ILA control plane is a distributed key-value store that manages mappings of identifiers to locators in a network. As tasks or containers are registered, the mappings are published to the control plane and disseminated to all ILA hosts through its push-based model, which differs from LISP pull-based model.

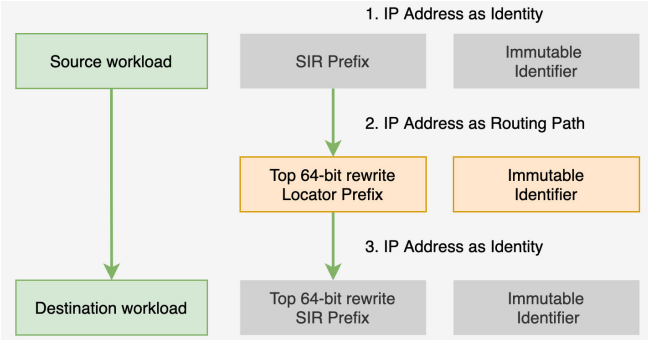


FIGURE 3. IPv6: Identifier and Locator Transformation.

In summary, while the ILA solution applies creative techniques through IPv6 transformations, the administrative boundaries of this architecture assume the control of the host stacks to allow an insertion strategy to preserve the original container identifiers and modify the locator instructions within the packet. In addition, it will continue to become more challenging to preserve the end-to-end identity that IPv6 advocates across heterogeneous cloud providers as NAT, Loadbalancers, and network-based middle-ware will force identifiers higher in the protocol stack.

V. CLOUD-NATIVE SERVICE MESH

A. MICROSERVICES AND SERVICE MESH

Previous sections discussed the networking challenges in cloud computing models that deployed N-Tier applications in VM or container artifacts. This section discusses the network requirements for a new generation of cloud-native applications called microservices [15]. A microservice is a collection of networked services, with each service performing a specific business function. Services are built independently from each other and delivered in the format of containers. The value is that developers can alter and upgrade the application without a full redeployment.

A single cloud-native application can be represented by thousands of services that are continually changing states. The probability of communication failure between services increases as cloud-native applications grows exponentially. Application resiliency leads to the need for a dedicated infrastructure layer handling the service-to-service communication. Service mesh [39], [41] addresses this challenge by injecting a proxy service, known as a sidecar, distributed within an application or microservice boundary. The service mesh ensures an optimal application-level response time by handling requests, retries, and timeouts, such as circuit breaking, load balancing, and traffic engineering. Each application request is routed optimally and is load-balanced without the knowledge of the application. The following sections describe the refinements of identifiers and locators, considering that an application residence is in multiple cloud platforms.

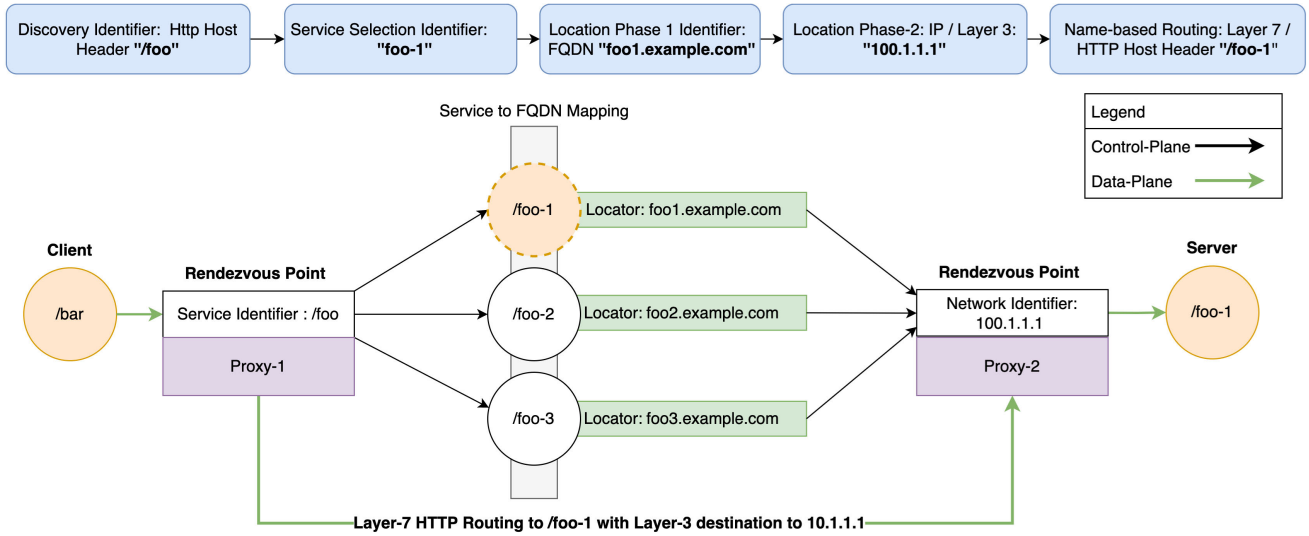


FIGURE 4. Cloud-Native Service Mesh: The Application-level Identifier for Name-based Routing.

B. SERVICE MESH NAMESPACE FOR DISCOVERY AND ROUTING

The service mesh is responsible for discovering, locating, and routing between proxies to establish application connections, as depicted in Figure 4. In this example, the client ''/bar'' service requests to communicate with the server ''/foo'' service. Firstly, the client sends an HTTP [20] request to query for foo, modifying the host header field to ''Host: foo''. The service name ''/foo'' is an anycast identifier [7] mapped to multiple instances or replicas of the service. The service mesh has a selection criterion to map the service identifier to a locator. In this example, the query for ''/foo'' is based on the proximity and latency between the client and server, and therefore ''foo-1'' is matched against the Fully Qualified Domain Name (FQDN) ''foo1.example.com ''. The resolution of this FQDN is IP address ''100.1.1.1'', which is the proxy rendezvous point. Furthermore, proxy-1 sets up a TCP connection with Proxy-2. Proxy-1 modifies the HTTP host header field to ''Host: foo-1'' to connect with the foo-1 service. In this example, the HTTP host header is a namespace that shifted the unique identifier from Layer-3 to Layer-7. The namespace to distinguish services is not limited to HTTP. With HTTP/S [51], the host header is encrypted, and the server name indicator (SNI), which is an extension of the TLS protocol [16] provides the namespace to distinguish services.

These name-based techniques were borrowed from earlier work in virtual-host networking [33], where web servers were consolidated to a common platform, overloading a single IP address with multiple service aliases [38]. A client would replicate the name previously learned from the Domain Name System (DNS) [44] into the HTTP application namespace, e.g., Host header field, to provide an application demuxer beyond the network layers. The HTTP/S protocol became the vehicle for carrying names and has been widely known as a name-based routing approach [35]. Content Delivery Networks (CDNs) also adopted the name-based virtual hosting to disentangle the conventional bindings between IP addresses, the applications they represent, and the servers to which they are assigned to [19].

C. SERVICE MESH IDENTITY AND THE URI NAMESPACE

Regarding identity, It is important to note that there are three distinct layers: the platform layer, the host layer, and the process/service layer. IP addresses have been the common identifier for modeling host-to-host communication. However, the scenarios where applications span multiple clouds present three challenges to model identity at the host layer:

- Workloads are ephemeral in nature, and the IP address is short-lived as an identifier.
- Workloads are deployed outside traditional administrative boundaries, such as the public cloud, which does not intersect with the internal enterprise network.
- Given that the service mesh proxies TCP connections and the network is riddled with network-based middleware boxes, the IP address becomes a weak and mutable identifier.

The objective of workload identity is to model service-to-service communication without preserving the IP address as the identifier. Applications that span administrative domains and cloud realms require identifiers to be uncoupled from the underlying infrastructure.

For cloud-native applications, the secure production identity framework for everyone (SPIFFE) addresses the workload identity challenge [58] using its naming schema independent of the discovery and routing described in this section. There are three components of SPIFFE: a specification and identifier used as a referral for a service (SPIFFE-ID), a SPIFFE Verifiable Identity Document (SVID) for embedding the SPIFFE-ID that is signed by a trusted authority, and the Workload API, which is an agent running within the

cloud that provides a method of obtaining the SVID. The SPIFFE-ID is a uniform resource identifier (URI) [5] that includes the scheme "spiffe://," a trust domain, and a path that specifies the name of the service. For example, the foo service is a billing system whose administrative domain is in the production environment. The identifier can be represented as the following:

spiffe://production.example.com/billing/foo

The SPIFFE ID is codified into an SVID, which is an extension of the x509 certificate [11], signed by an authority inside the trust domain. Within the specification, the SPIFFE ID is set to the URI type in the subject alternative name extension (SAN) [11]. Thus, each workload will have a SPIFFE bundle used for path validation against all the SVIDS that belongs to the trust domain. SVIDs allow clients and servers to gain mutual knowledge by asserting their provable cryptographic identity for mutual TLS authentication [14], [23].

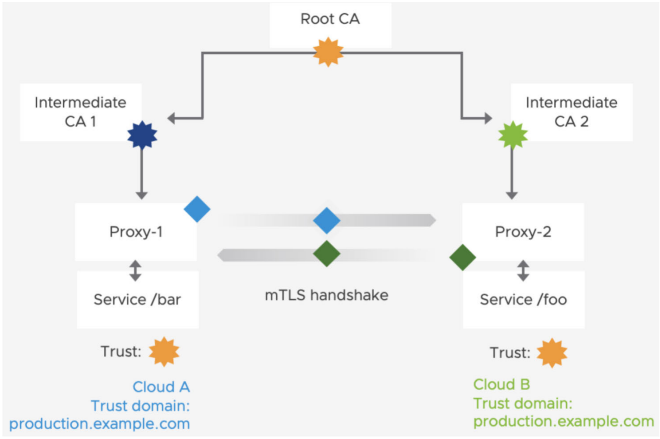


FIGURE 5. PKI: Mutual Authentication within a Trust Domain.

As depicted in Figure 5 an X509 certificate is issued to services "/bar" and "/foo". Both services are part of the trust domain of "production.example.com' and have a common root of trust in the PKI hierarchy [9]. Both services will undergo the verification process in this scenario and exchange their x509 certificate as their identity document for mutual TLS authentication. The trust domain within the SVID identifier allows the administrative domain the flexibility to expand outside its platform and across multiple clouds where control of identity at the IP level is harder to achieve.

In summary, in the cloud-native service mesh, IP addresses are no longer identifiers for applications, and the distinguishing factors are names. Names do not represent a host, VM, or container; they refer to services inside an application. Services are assigned a persistent name as the identifier with a flexible administrative scope, spanning multiple cloud environments and organizations. Therefore, it is essential to delineate naming systems' for identifiers and locators in the cloud-native service mesh. First, the service name is derived from the HTTP header for the discovery and locating phase, which provides a mapping to a concrete FQDN. The URI within the SVID provides the verification phase for digital identity.

VI. COMPARISONS AND FUTURE DIRECTIONS

This paper discussed how the iteration of each network architecture addressed a new set of challenges that were non-existent in the previous generation. The paper includes an observation that the application artifact and cloud consumption model forced changes in identifiers and locators at various layers of the protocol stack. Firstly, this section will summarize how the core vs. edge evolved based on the network control required to meet the cloud-native needs. After that, the paper discusses future directions and provides topical areas that may play an essential role in future cloud-native networking.

A. COMPARISON BETWEEN DISCUSSED TECHNIQUES

Table 1 summarizes the association between the generation of the application artifact, identity controller, identifier namespace, and the network architecture insertion point. LISP and NV were the generation architectures that intersected with server virtualization. The insertion point in both approaches assumes control of the network platform and therefore takes a bottom-up approach to identifiers and locators anchored to the IP layer. In contrast to LISP, NV shifted the edge to the hypervisor, projecting an application-first architecture supporting the dynamic characteristics of enterprise cloud computing. ILA adopted similar techniques to NV by adding a layer of indirection with IPv6 to preserve the container identifier. However, the ILA edge model shifted to a layer above NV by inserting control at the container OS network namespace level.

The public cloud created a new challenge not present in previous approaches. LISP, NV, and ILA are approaches where the insertion point can assert identity, independent of location, as long as there is network control of the platform. However, as the public cloud infrastructure layers are hidden from consumers, it negates the ability of the network-centric approaches to provide a persistent identity at the IP layer using overlay techniques. A new generation of applications is designed to be less dependent on the underlying platform. The artifact comprises ancillary services known as a sidecar, which provides reachability and resiliency. The cloud-native service mesh addressed this challenge by creating an insertion point as distributed proxies to be the new edge. The abstractions for identity evolved to be an attribute of the application. Therefore it did not serve the purpose of structuring the IP address plan for identity and location as demonstrated in predecessor architectures. The use of naming as a point of indirection demonstrated that network flows could be distinguished in the context of the application-level session.

SPIFFE was separated into its own category with the cloud-native workload identities framework. Many application instances could represent a service in a plurality of cloud platforms. SPIFFE addresses this challenge by disseminating identifiers in the form of the URI to genuine instances of a

TABLE 1. Supporting Network Architectures, Namespace, Idenitifers and Insertion Points.

Application Artifact	Identity Controller	Identity Namespace	Distinguished Identifier	Insertion Point
Single-Tier: Virtual Machine	Locator/Identifier Seperation Protocol	Disjoint IPv4	IPv4 Address	Data Center Core Network
Multi-Tier: Virtual Machines	Network Virtualization	Disjoint IPv4	IPv4 Address	Hypervisor Network
Multi-Tier: Containers	Identifier Locator Addressing	Separated IPv6	IPv6 Address	Container OS - Network Namespace
Microservice: Containers	Cloud-Native Service Mesh	HTTP/S Host Header	Service & Canonical Name	Container Sidecar Proxy
Microservice: Hyperscale	SPIFFE	X509	URI	PKI Certificate Issuance

service. The x509 document underpins the SPIFFE solution; therefore, cloud-native workloads must have a common root of trust to authenticate mutually. If two services are under different trust hierarchies, then a third-party broker is required to exchange trust bundles between authority domains for mutual authentication.

Based on the observations of the cloud-native network requirements, there are four distinct roles of naming used as identifiers:

1) **Service.** A public-like persona to discover and identify services needed to compose an application.
2) **Locator.** The core or canonicalized name and the locator identifier to resolve the IP address of the service instance.
3) **Routing.** An application-level identifier replicated in the application protocol namespace to distinguish network sessions over a common IP address target.
4) **Identification.** Verifiable identifier to provide assurance it is a genuine copy or application instance.

Using these identifiers is only possible with the strict governance of naming inside an isolation boundary. Therefore, a namespace is required to ensure the scope of discovery, identifying, and locating workloads is within the trust domain.

B. COMPARISONS WITH THE STATE-OF-THE-ART

In recent years, developers have shifted their focus to constructing and deploying microservices without considering the underlying infrastructure requirements [52]. Serverless computing is an example of this paradigm that allows developers to focus on the business logic of the application without the consideration of provisioning and scaling of the entire system. Serverless functions are seen as event-driven, which triggers a temporary execution environment only when required. Amazon Web Services (AWS) Lambda, Microsoft Azure Functions, Google Functions, and Cloudflare Workers have introduced serverless capabilities, known as Function-as-a-Service (FaaS) [8].

In the event the serverless function is called, a virtualized workload is provisioned with application files and dependencies required to execute the function and terminates after the operations are performed [25]. In most implementations, containers are used as the primary artifact for their lightweight, application process isolation, and resource provisioning [21], [31]. Although serverless was originally enabled for cloud environments, it gained traction to cement its position in edge computing [59] where computational resources are closer to the data source [1]. However, as the edge environments consist of resource-constrained devices and diverse CPU architectures, WebAssembly (Wasm) [55] has emerged as a promising artifact for future serverless support at the cloud-edge [21], [22], [25], [42].

The serverless model has created new challenges in cloud-native networking that were not present in previous sections of this paper. In comparison, cloud-native service mesh provides end-to-end network control through distributed proxies. Each proxy enacts as an ambassador for each service-to-service communication, using the above four identifiers. Unlike service mesh, serverless computing offered by cloud providers does not expose the naming and addressing of the infrastructure platform [32], which thwarts the ability to bind the identifiers and locators for network control.

Irrespective of the serverless artifact type, the cloud providers only expose a publicly accessible Universal Resource Locator (URL) [6] identifier, which is used in the HTTP protocol as an access mechanism to a serverless function. The URL namespace describes the path instructions to a location of the available service. For clarity, serverless applications do not reside in a single location but are replicated globally; therefore, the same URL is also required to access multiple instantiations or replicas.

In Figure 6, the domain name portion of the URL is referred to as anycast identifier [7], which is persistent and resolves to multiple locators or IP addresses. The domain name to IP address translation is coordinated by the cloud-providers DNS-based global traffic management system (GTM) [27]. The DNS-Based GTM enacts as a geographic site-selector

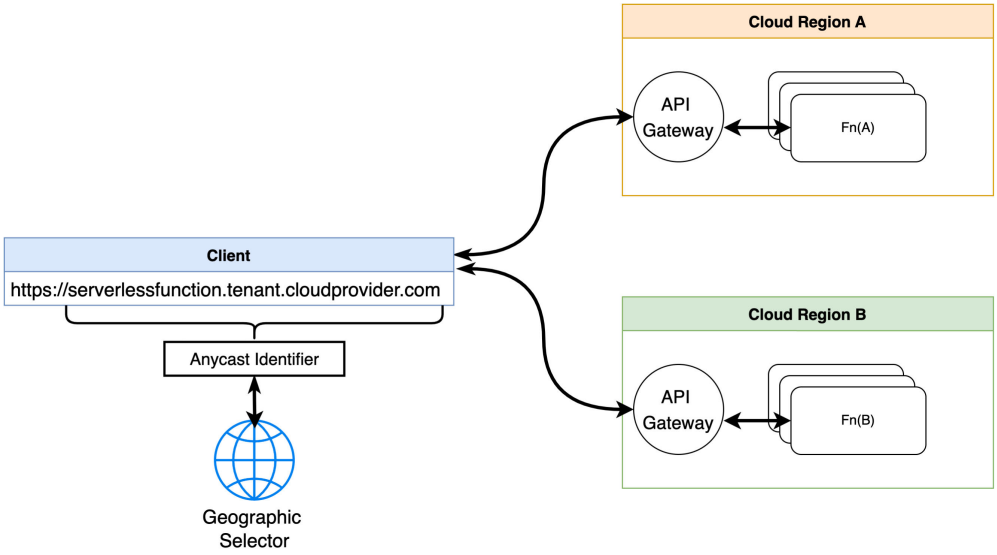


FIGURE 6. Persistent Identifier for Serverless in Hyperscaler Environments.

directing client connections to the closest serverless function [56]. This allows the URL to remain immutable, allowing the DNS to be the identity-like and public persona, returning customized answers based on the information from the client queries [10], [57].

C. FUTURE DIRECTIONS AND INVESTIGATIONS

A preview of future directions using Internet identifiers to support cloud-native services is appearing. Globally decentralized cloud-native applications rely on URLs to mimic and behave like URIs; To refer to the service independent of its location. This leads to three investigative areas to provide identifiers and the underlying infrastructure support that will play an essential role in cloud-native networks.

1) PERSISTENT IDENTIFIER

In the digital library space, the Digital Object Identifier (DOI) [47] is a persistent identifier that refers to a book or document residing in multiple locations at the same time. Analogous to a library, a cloud-native application also contains instances simultaneously running in multiple environments and locations. However, the complexity is that cloud-native applications today require four separate identifiers to be referred to and verified as the given instance. Cloud-native architectures need to adopt similar principles from the digital libraries; all application service instances should refer to as a single identifier. What an application is should be separated from where it is.

Furthermore, an investigation is also required to provide referential consistency that anchors application instances to a common identification scheme. This would comprise a directory service where the identifier could be passed between parties to provide locator mappings to cloud resources. In addition, the identifier is decorated with additional metadata such that the identifier would remain persistent. Unlike copies of books, applications evolve; therefore, providing additional metadata to adorn the query is a possible technique to obviate the need to create a new identifier.

2) IDENTITY RENDEZVOUS

The identity rendezvous provides two functions. Firstly, identity attestation to analyze credentials from all workloads to assure they are qualified application replicas or instances. Secondly, to connect and mutually authenticate between source and destination service. This approach is only possible when there is control over the workload deployment pattern, e.g., container workload and its ancillary service. When microservices are also constructed with serverless functions, it exposes new identity challenges that require the following two investigation areas. Firstly, as described in this section, cloud providers hide the interconnections of a serverless environment. It is possible to provide identity rendezvous as a sidecar or proxy as long as it is outside the providers' realm. The challenge it may create is additional network latency costs, which are worth exploring. Secondly, the execution of a serverless function has a temporary runtime environment which creates performance vs. security pressure to minimize application startup delays. In the context of the SPIFFE example, the framework assumes control of the underlying operating system of where the application is running. Since the serverless environment does not provide access, it is unclear how the attestation process will be completed to ensure the application is a genuine instance and the true owner of the identifier it passes before mutual authentication. This is use-case is not limited to serverless but future applications where only the API is exposed without access to the underlying infrastructure.

3) DISSEMINATION OF IDENTITIES

In order to construct complex applications between network and cloud realms, isolation boundaries can span multiple clouds and be created on-demand. This ensures that

application instances within an isolation boundary are uniquely rooted in a common trust model. If the industry wishes to continue using a SPIFFE-like model, where the certificate chain of trust underpins the identity framework, the challenge arises when applications are not within a common isolation boundary. While federation techniques are well understood in the industry with various approaches to exchanging bundles [60], the architecture lacks a third-party identity broker to manage and distribute trust bundles for general use and on a need-to-know basis.

VII. CONCLUSION

This paper highlighted how each generation of network architectures addressed a challenge created by evolved application artifacts. It highlights that cloud-native applications have shifted identifiers from the IP layer and have become an attribute of the application layer used to distinguish workloads and provide location-independent identification schemes. In addition, this paper reviewed the current state-of-the-art application deployment patterns and the future identifier challenges that may arise. This is an important topic that requires further investigation into three topical areas. Firstly, evaluate existing persistent identifiers in the digital space that could be applied for broader use. Regardless of location, application instantiations should be referred to as the single identifier. Secondly, the verification process, where credentials are analyzed to provide identity attestation and issuance of its imprimatur, should be consistent across any cloud and edge computing platform. Lastly, the delivery method to control and disseminate trust anchors to allow application communication between trust realms will underpin the success of a common identity system. It is hoped this retrospective of identifiers may influence the future design of identifiers and locators used in cloud-native architectures.

REFERENCES

[1] M. S. Aslanpour, A. N. Toosi, C. Cicconetti, B. Javadi, P. Sbarski, D. Taibi, M. Assuncao, S. S. Gill, R. Gaire, and S. Dustdar, "Serverless edge computing: Vision and challenges," in *Proc. Australas. Comput. Sci. Week Multiconference*, Feb. 2021, pp. 1–10.

[2] R. J. Atkinson and S. N. N Bhatti. *Identifier-Locator Network Protocol (ILNP) Architectural Description*, document RFC 6740, RFC Editor, Nov. 2012.

[3] G. Banga, P. Druschel, and J. C. Mogul, "Resource containers: A new facility for resource management in server systems," in *Proc. OSDI*, vol. 99, 1999, pp. 45–58.

[4] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, and M. G. Rabbani, "Data center network virtualization: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 909–928, 2nd Quart., 2013.

[5] T. Berners-Lee, R. T. Fielding, and L. Masinter. *Uniform Resource Identifiers (URI): Generic Syntax*, document RFC 2396, RFC Editor, Aug. 1998. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2396.txt

[6] T. Berners-Lee, L. Masinter, and M. McCahill. *Uniform Resource Locators (URL)*, document RFC 1738, RFC Editor, Dec. 1994. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1738.txt

[7] B. Carpenter and S. Brim.*Middleboxes: Taxonomy and Issues*, document RFC 3234, RFC Editor, Feb. 2002. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3234.txt

[8] P. Castro, V. Ishakian, V. Muthusamy, and A. Slominski, "The rise of serverless computing," *Commun. ACM*, vol. 62, no. 12, pp. 44–54, Nov. 2019.

[9] S. Chokhani *et al. Internet X. 509 Public Key Infrastructure Certificate Policy and Certification Practices Framework*, document RFC 3647, RFC Editor, Nov. 2003.

[10] C. Contavalli *et al. Client Subnet in DNS Queries*, document RFC 7871, RFC Editor, May 2016.

[11] D. Cooper *et al. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, document RFC 5280, RFC Editor, May 2008. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5280.txt

[12] B. Davie and J. Gross. *A Stateless Transport Tunneling Protocol for Network Virtualization (STT)*. document Internet-Draft draft-davie-stt-08, IETF Secretariat, Apr. 2016.

[13] S. E. Deering and R. M. Hinden. *Internet Protocol, Version 6 (IPv6) Specification*, document RFC 2460, RFC Editor, Dec. 1998. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2460.txt

[14] T. Dierks and C. Allen. *The TLS Protocol Version 1.0*, document RFC 2246, RFC Editor, Jan. 1999. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2246.txt

[15] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, "Microservices: Yesterday, today, and tomorrow," *Present and Ulterior Software Engineering*. Cham, Switzerland: Springer, 2017, pp. 195–216.

[16] D. Eastlake. *Transport Layer Security (TLS) Extensions: Extension Definitions*, document RFC 6066, RFC Editor, Jan. 2011. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6066.txt

[17] K. B. Egevang and P. Francis. *The IP Network Address Translator (NAT)*, document RFC 1631, RFC Editor, May 1994. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1631.txt

[18] D. Farinacci *et al. The Locator/ID Separation Protocol (LISP)*, document RFC 6830, RFC Editor, Jan. 2013. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6830.txt

[19] M. Fayed, L. Bauer, V. Giotsas, S. Kerola, M. Majkowski, P. Odintsov, J. Sitnicki, T. Chung, D. Levin, A. Mislove, C. A. Wood, and N. Sullivan, "The ties that un-bind: Decoupling IP from web services and sockets for robust addressing agility at CDN-scale," in *Proc. ACM SIGCOMM Conf.*, Aug. 2021, pp. 433–446.

[20] R. T. Fielding *et al. Hypertext Transfer Protocol—HTTP/1.1*, document RFC 2616, RFC Editor, Jun. 1999. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2616.txt

[21] P. K. Gadepalli, G. Peach, L. Cherkasova, R. Aitken, and G. Parmer, "Challenges and opportunities for efficient serverless computing at the edge," in *Proc. 38th Symp. Reliable Distrib. Syst. (SRDS)*, Oct. 2019, pp. 261–2615.

[22] P. K. Gadepalli, S. McBride, G. Peach, L. Cherkasova, and G. Parmer, "Sledge: A serverless-first, light-weight wasm runtime for the edge," in *Proc. 21st Int. Middleware Conf.*, 2020, pp. 265–279.

[23] A. Goel and B. Thangaraju, "Authenticating distributed systems using SPIRE over kubernetes cluster," in *Proc. IEEE Int. Conf. Electron., Comput. Commun. Technol. (CONECCT)*, Jul. 2022, pp. 1–6.

[24] J. Gross, I. Ganga, and T. Sridhar. *Geneve: Generic Network Virtualization Encapsulation*, document RFC 8926, RFC Editor, Nov. 2020.

[25] A. Hall and U. Ramachandran, "An execution model for serverless functions at the edge," in *Proc. Int. Conf. Internet Things Design Implement.*, Apr. 2019, pp. 225–236.

[26] T. Herbert and P. Lapukhov. *Identifier-Locator Addressing for IPv6*, document Internet-Draft draft-herbert-intarea-ila-01, IETF Secretariat, Mar. 2018.

[27] C. Huang, D. A. Maltz, J. Li, and A. Greenberg, "Public DNS system and global traffic management," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 2615–2623.

[28] G. Huston and C. S. Apnic, "In defence of NATs," *Internet Protoc. J.*, vol. 20, no. 3, pp. 25–33, 2017.

[29] V. Ishakian, J. Akinwumi, F. Esposito, and I. Matta, "On supporting mobility and multihoming in recursive internet architectures," *Comput. Commun.*, vol. 35, no. 13, pp. 1561–1573, Jul. 2012.

[30] D. Jen, M. Meisel, H. Yan, D. Massey, L. Wang, B. Zhang, and L. Zhang, "Towards a new internet routing architecture: Arguments for separating edges from transit core," in *Proc. HotNets*, 2008, pp. 103–108.

[31] E. Jonas, J. Schleier-Smith, V. Sreekanti, C.-C. Tsai, A. Khandelwal, Q. Pu, V. Shankar, J. Carreira, K. Krauth, N. Yadwadkar, J. E. Gonzalez, R. A. Popa, I. Stoica, and D. A. Patterson, "Cloud programming simplified: A Berkeley view on serverless computing," 2019, *arXiv:1902.03383*.

[32] K. Kaffes, N. J. Yadwadkar, and C. Kozyrakis, "Centralized core-granular scheduling for serverless functions," in *Proc. ACM Symp. Cloud Comput.*, Nov. 2019, pp. 158–164.

[33] C. Karayiannis, "The apache web server," in *Web-Based Projects that Rock Class*. Berlin, Germany: Springer, 2019, pp. 1–37.

[34] A. Khan, "Key characteristics of a container orchestration platform to enable a modern application," *IEEE Cloud Comput.*, vol. 4, no. 5, pp. 42–48, Sep./Oct. 2017.

[35] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun. (SIGCOMM)*, 2007, pp. 181–192.
[36] T. Koponen, K. Amidon, P. Balland, M. Casado, A. Chanda, B. Fulton, I. Ganichev, J. Gross, P. Ingram, E. Jackson, and A. Lambeth, "Network virtualization in multi-tenant datacenters," in *Proc. 11th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, 2014, pp. 203–216.
[37] M. Lasserre *et al. Framework for Data Center (DC) Network Virtualization*, document RFC 7365, RFC Editor, Oct. 2014.
[38] E. Lear and R. Droms. *What's in a name: Thoughts from the NSRG*, document Internet-Draft draft-irtf-nsrg-report-10, IETF Secretariat, Sep. 2003.
[39] W. Li, Y. Lemieux, J. Gao, Z. Zhao, and Y. Han, "Service Mesh: Challenges, state of the art, and future research opportunities," in *Proc. IEEE Int. Conf. Service-Oriented Syst. Eng. (SOSE)*, Apr. 2019, pp. 122–1225.
[40] M. Mahalingam *et al. Virtual Extensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks Over Layer 3 Networks*, document RFC 7348, RFC Editor, Aug. 2014. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7348.txt
[41] A. E. Malki and U. Zdun, "Guiding architectural decision making on service mesh based microservice architectures," in *Proc. Eur. Conf. Softw. Archit.* Cham, Switzerland: Springer, 2019, pp. 3–19.
[42] J. Ménétrey, M. Pasin, P. Felber, and V. Schiavoni, "WebAssembly as a common layer for the cloud-edge continuum," in *Proc. 2nd Workshop Flexible Resource Appl. Manag. Edge*, Jul. 2022, pp. 3–8.
[43] D. Meyer, L. Zhang, and K. Fall. *Report From the IAB Workshop on Routing and Addressing*, document RFC 4984, RFC Editor, Sep. 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4984.txt
[44] P. Mockapetris and K. J. Dunlap, "Development of the domain name system," in *Proc. Symp. Proc. Commun. Archit. Protocols (SIGCOMM)*, 1988, pp. 123–133.
[45] P. Nikander, A. Gurtov, and T. R. Henderson, "Host identity protocol (HIP): Connectivity, mobility, multi-homing, security, and privacy over IPv4 and IPv6 networks," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 2, pp. 186–204, 2nd Quart., 2010.
[46] M. D. O'Dell. *GSE—An Alternate Addressing Architecture for IPv6*, document Internet-Draft draft-ietf-ipngwg-gseaddr-00, IETF Secretariat, Feb. 1997.
[47] N. Paskin, "Digital object identifier (DOI) system," *Encyclopedia Library Inf. Sci.*, vol. 3, pp. 1586–1592, Dec. 2010.
[48] C. E. Perkins, "Mobile IP," *IEEE Commun. Mag.*, vol. 35, no. 5, pp. 84–99, May 1997.
[49] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, and K. Amidon, "The design and implementation of open vSwitch," in *Proc. 12th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, 2015, pp. 117–130.
[50] Y. Rekhter, T. Li, and S. Hares. *A Border Gateway Protocol 4 (BGP-4)*, document RFC 4271, RFC Editor, Jan. 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4271.txt
[51] E. Rescorla. *HTTP Over TLS*, document RFC 2818, RFC Editor, May 2000. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2818.txt
[52] M. Roberts and J. Chapin, *What is Serverless?*. Sebastopol, CA, USA: O'Reilly, 2017.
[53] E. Rosen and Y. Rekhter. *BGP/MPLS IP Virtual Private Networks (VPNs)*, document RFC 4364, RFC Editor, Feb. 2006.
[54] E. Rosen, A. Viswanathan, and R. Callon. *Multiprotocol Label Switching Architecture*, document RFC 3031, RFC Editor, Jan. 2001. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3031.txt
[55] A. Rossberg, B. L. Titzer, A. Haas, D. L. Schuff, D. Gohman, L. Wagner, A. Zakai, J. F. Bastien, and M. Holman, "Bringing the web up to speed with WebAssembly," *Commun. ACM*, vol. 61, no. 12, pp. 107–115, Nov. 2018.
[56] K. Schomp and R. Al-Dalky, "Partitioning the internet using anycast catchments," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 50, no. 4, pp. 3–9, Oct. 2020.
[57] K. Schomp, O. Bhardwaj, E. Kurdoglu, M. Muhaimen, and R. K. Sitaraman, "Akamai DNS: Providing authoritative answers to the world's queries," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun.*, Jul. 2020, pp. 465–478.
[58] *Secure Production Identity Framework for Everyone*. Accessed: May 8, 2021. [Online]. Available: https://www.spiffe.io
[59] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.
[60] M. S. L. D. Silva *et al. Integrating SPIFFE and SCONE to Enable Universal Identity Support for Confidential Workloads*, Universidade Federal de Campina Grande, 2021.
[61] A. Tosatto, P. Ruiu, and A. Attanasio, "Container-based orchestration in cloud: State of the art and challenges," in *Proc. 9th Int. Conf. Complex, Intell., Softw. Intensive Syst.*, Jul. 2015, pp. 70–75.
[62] B. Walters, "VMware virtual platform," *Linux J.*, vol. 1999, no. 63, p. 6, 1999.

ANDREW BABAKIAN is currently pursuing the Ph.D. degree in engineering with the University of Technology Sydney, Australia. He is also a Technologist and the Director at the Networking and Advanced Security Business Group, VMware, driving innovation efforts in software defined networking and identity technologies for cloud-native architectures. His current research interests include naming systems and the evolution of Internet identifiers.

PERE MONCLUS received the double M.S.E.E. degree from the Universitat Politecnica de Catalunya and the Politecnico di Torino. He is currently the CTO at the Networking and Advanced Security Business Group, VMware. He is responsible for defining strategy and leading an innovation team driving the evolution of software defined networking and cloud-native architectures. Previously, a Distinguished Engineer at Cisco, where he led multiple successful products in the area of security, load balancing, and data centers. He holds over 30 patents.

ROBIN BRAUN (Life Senior Member, IEEE) received the B.Sc. degree (Hons.) from Brighton University, Brighton, U.K., in 1980, and the M.Sc.(Eng.) and Ph.D. degrees from the University of Cape Town, Cape Town, South Africa, in 1982 and 1986, respectively. He started his academic career at the University of Cape Town, in 1986. In 1998, he moved to the University of Technology Sydney, Australia, where he occupied the Chair of Telecommunications Engineering. Prior to moving to academia, he spent ten years in industry, mostly with Philips and Plessey, where he worked on the design of precision electronic distance measuring equipment. He is currently an Honorary Professor with the School of Electrical and Data Engineering. His recent work has been in network protocols and the management of complex next-generation networks. He is very active in software-defined networks. He is also a Founder Member of Australia and New Zealand Software Defined Networking (ANZSDN).

JUSTIN LIPMAN (Senior Member, IEEE) received the Ph.D. degree in telecommunications engineering from the University of Wollongong, Australia, in 2004. He is an Industry Associate Professor at the University of Technology Sydney (UTS) and a Visiting Associate Professor at the Hokkaido University's Graduate School of Engineering. He is the Director of research translation with the Faculty of Engineering and IT and the Director of the RF Communications Technologies (RFCT) Laboratory, where he leads industry engagement in RF technologies, cybersecurity, privacy preserving technologies, the Internet of Things, and tactile Internet. He was previously the Deputy Chief Scientist of the Food Agility Cooperative Research Center. Prior to joining UTS, over a 12 year period, he held a number of senior management and technical leadership roles at Intel and Alcatel driving research and innovation, product development, architecture, and IP generation. His research interests include enabling "things" to be adaptive, connected, distributed, ubiquitous, and secure. He serves as a Committee Member at Standards Australia contributing to International IoT Standards and Digital Twins.

Appendix C

Internet Identifiers: Survey of History, Challenges, and Future Perspectives

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 February 2024, accepted 15 March 2024, date of publication 27 March 2024, date of current version 16 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3382115

SURVEY

Internet Identifiers: A Survey of History, Challenges, and Future Perspectives

ANDREW BABAKIAN[1], GEOFF HUSTON[2], ROBIN BRAUN[1], (Life Senior Member, IEEE), AND JUSTIN LIPMAN[1], (Senior Member, IEEE)

[1]School of Electrical and Data Engineering, University of Technology Sydney, Ultimo, NSW 2007, Australia
[2]Asia-Pacific Network Information Centre, Brisbane, QLD 4101, Australia

Corresponding author: Andrew Babakian (andrew.babakian@student.uts.edu.au)

ABSTRACT Identity systems are fundamental to any network, just as a language is shaped by a consistent set of symbols and their interpretation. A network is defined by a consistent set of identities and their usages. However, as pressures mount to customize and adapt to these identity spaces, one ponders how a network sustains its utility through coherence. This study investigates evolving Internet identifiers and their supporting systems. Owing to the multi-disciplinary nature of the topic, this paper draws perspectives from a wide array of sources that contribute to Internet systems and digital library systems. Initially, this paper highlights the dynamism and role of namespaces, focusing on the critical need to maintain coherence in the public domain. It also discusses the impact of mobility and digital cloning on identifiers and explores their influence on identity and location. In addition, key Internet identifiers are analyzed, contrasting them with methodologies adopted by digital library systems to provide deeper insights into various identity models. Furthermore, this study explores the evolution of the Domain Name System (DNS) as an identity system, examining the tensions and adaptations driven by customization demands. Finally, this paper explores alternative namespaces beyond the DNS, considering potential responses to these evolutionary pressures and future implications.

INDEX TERMS Identifier systems, naming systems, namespace, URI, URN, URL, DOI, IP address, DNS.

I. INTRODUCTION

Identifier systems serve as the backbone of the digital environment, facilitating interactions between diverse entities within the digital realm. The continuous adaptation of these identifiers reflects the need for efficient and flexible techniques to identify and interact with resources, services, and users. Analogous to a common language, the Internet relies on a common set of symbols for communication efficacy.

This study explores the evolution of Internet identifiers through a comprehensive survey of the relevant literature. It includes a historical overview, an analysis of existing studies, and discussions on current challenges and prospects in this field. Given the multi-disciplinary nature of the topic, this paper draws perspectives from a range of authoritative sources. These include the World Wide Web Consortium (W3C), Internet Architecture Board (IAB) Workshops, Request for Comments (RFCs), and academic literature. Additionally, it incorporates significant contributions from entities involved in digital library systems, such as the Library of Congress, and the development of key registry systems to support the Digital Object Identifiers (DOIs).

The emphasis is not on an exhaustive list of identifiers but on those critical to the Internet's infrastructure and influential in digital handle systems. The aim is to synthesize a broad range of literature concerning digital identifiers in the internet namespace and consider potential future directions influenced by evolutionary pressures. This synthesis seeks to enhance the understanding of the Internet's developmental trajectory.

Exploration begins by examining the fundamental roles of namespaces in identifying, locating, and classifying entities. Moreover, unique considerations on the Internet

The associate editor coordinating the review of this manuscript and approving it for publication was Diana Gratiela Berbecaru.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

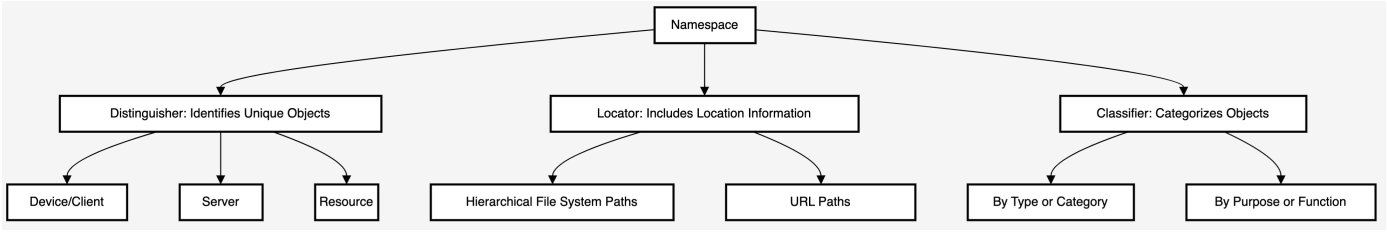


FIGURE 1. Namespace scope and roles.

are given attention, primarily the importance of maintaining coherence in the public namespace. Such coherence hinges on a consistent symbol interpretation, universal reachability, and referential integrity. Subsequently, the critical dimensions of identity and location within the Internet context are examined, exploring how their significance has been influenced by advancements such as mobility and digital cloning.

The narrative progresses to trace and emphasize the roles played by Universal Resource Identifiers (URIs), Universal Resource Locators (URLs), and Universal Resource Names (URNs) within the realm of identifiers and locators. To transcend the boundaries of networked systems, this study's exploration extends to identifier systems utilized in broader contexts, such as library systems. These include the International Standard Book Number (ISBN) used in publishing and Digital Object Identifiers (DOIs) for digital resources. The paper then juxtaposes these identity models with those at the Internet's heart, such as the URI versus URL and the IP Address namespace.

Subsequently, the study examines the thematic topic of search versus identification, exploring the evolution of directory lookup systems. This includes a chronological analysis, commencing with the earlier network namespaces, such as HOSTS.TXT and X.500. Examining X.500 Directory Naming Framework is important to compare its directory system approach with the Domain name system (DNS), particularly the system differences between searchability and uniqueness of resolving names. The study transitions to the emergence and importance of the DNS. The paper draws insights from the DNS predecessors to highlight its evolutionary journey. The successive generation of new resource records highlights this evolution, enhancing the system's flexibility to address emerging challenges.

Furthermore, the study analyzes the shift of DNS from a static lookup system to a dynamic, computation-oriented resource. In addition, this study further examines the evolutionary pressures that catalyzed the creation of other namespaces that operate outside the global scope of the DNS. These alternate namespaces, established for specific purposes, signify another facet of the Internet evolutionary process and highlight the necessity for a cohesive approach across varied namespaces to mitigate the risk of fragmentation.

In conclusion, this study offers insights into the progression of identifier systems in the Internet context, emphasizing the critical role of DNS in supporting Internet identifiers. It also highlights the ongoing transformations driven by evolving market needs and technological advancements. This discussion clarifies the complex interplay between identifier systems, the necessity of namespace cohesion, and the adaptation to evolutionary pressures.

II. BACKGROUND

Various factors contribute to the functionality and design of the complex field of Internet identifiers, making it essential to understand these elements thoroughly. This section establishes the foundation of the paper by tracing the developmental history of Internet identifiers, discussing namespaces and their role in structuring digital communication, and acknowledging their use across multiple disciplines. It clarifies the concept of a namespace, which is crucial for grasping the discussions that follow in this paper. Moreover, this section investigates the specific considerations and challenges within Internet identifiers, focusing on maintaining coherence and referential integrity. Finally, this section examines the interplay between identity and location, clarifying these concepts in the context of Internet infrastructure and library systems.

A. DEVELOPMENT AND HISTORY OF INTERNET IDENTIFIERS

The Internet's development was not in isolation; it was influenced by contemporaneous network protocols and the interplay between the Internet, DECnet and other proprietary network protocols. A common feature of these protocols was their limited scope. As a result, the inclination to associate natural language alias with network-connected hosts was not considered a significant issue. The prevailing solution was to associate a 'word' with a protocol address, assemble this information for all hosts, and distribute a copy to every connected host.

Two methods ensured uniqueness in constructing identifiers: one involved using a statistically unique random number, and the other was a managed identifier space through a hierarchical system, which is further discussed in section II-C. In the case of DECnet, scalability challenges were encountered early in its development, leading to the adoption of a hierarchical, two-level routing structure comprising of 'areas' and 'local networks' and a corresponding two-level name alias system. The system used a namespace format of two labels separated by a double

colon (AREA::NAME), tightly integrated with the network's connection topology [6].

In parallel, the rise of store/forward messaging systems introduced additional requirements. These systems, often not permanently connected, evolved without central coordination and rapidly expanded. Initial attempts to use single-level naming faced collision issues, prompting a shift to a generalized approach to the two-layer naming system using an n-layer hierarchy. The Internet adopted this hierarchical naming approach coinciding with the transition from the HOSTS.TXT file system to the Domain Name System (DNS), which is further detailed in sections IV-A and IV-C.

B. THE ROLE OF A NAMESPACE

Names categorize the world by assigning unique identifiers to concepts and entities. These names serve as a crucial means of conveying a shared understanding of a concept in communication, serving as essential components in maintaining referential integrity and enabling clear references. A namespace is the total set of allowable names within a broad class. Namespaces enable the establishment of defined and limited environments in which the uniqueness of names is ensured within a certain context. For instance, a person's first name may be sufficient to distinguish them within a close-knit community of friends, family, and coworkers. However, when the context expands to a larger population, such as a city, state, or country, additional information, such as the middle and last name, is necessary to guarantee the uniqueness of the name. In human use contexts, a significant level of ambiguity is often tolerated as opposed to automated systems.

Automated systems are no exception regarding the importance of the role and coordination of naming within namespaces. A namespace creates a boundary for the unique use of names and requires strict constraints to eliminate ambiguities. These namespaces are often formally specified to ensure consistency, ensuring that names are unique within a certain context. However, the conflation of multiple roles of names to indicate 'who,' 'where,' and 'what' in automated systems has led to an overload of their semantic role [13]. This study highlights the three roles played by a name in a namespace for automated systems. Please refer to Figure 1, which illustrates these multiple roles.

First, a name identifies and refers to a specific object, including a broad set of networked entities such as devices/clients, content, resources, or servers. They are used to distinguish distinct instances of objects within a broad class [25]. Techniques to disambiguate larger data sets concerning Internet considerations are discussed in Section II-D.

Second, the scope of identity in a namespace may include a locator. This is often observed in hierarchical file systems, where the prefix before the file name functions as a location path [3], [131]. However, it may not be associated with a location context, such as the IMEI in mobile networks, which has no location context. It is important to note that incorporating location into an identity name can assist in the uniqueness of the name, particularly for automated systems. However, this is a convenient shortcut for automated systems, whereas, for human systems, there is an implicit location context of 'here' and 'now' in everyday conversation. The significance of this distinction is discussed in Section II-E to compare and contrast the Internet challenges in terms of identity and location.

Finally, including a classifier in the namespace is optional and depends on the context of use. If the identity system is designed for a specific purpose, the context of its use is already defined. However, if the system is intended for multiple-use contexts, a classifier or 'what' parameter may be required. The classifier can be included as an attached qualifier or embedded in the identity token, depending on the system's requirements and efficiency. While embedding the classifier may be cheaper and more efficient in automated systems, it is not a mandatory feature of identity systems but rather a convenient shortcut in certain situations. The primary role of the classifier is to categorize objects across broad sets of classes using attributes within the namespace [68]. For example, library systems use classification techniques [17], [134] to logically categorize physical and electronic documents based on the subject and discipline, allowing users to browse and locate them.

C. FLAT AND HIERARCHIES

To ensure that identifiers are uniquely allocated to entities within a flat namespace, a fundamental operation involves enumerating each identifier across the namespace to prevent conflict. A conflict arises when multiple entities are assigned the same identifier within a namespace. When this occurs, it becomes impossible for the namespace to resolve the identifiers unambiguously, as there are no distinguishing factors between conflicting entities. Despite these limitations, certain systems have found ways to operate efficiently within flat namespaces. For instance, The Onion Router (TOR) [43] and Distributed Hash Table (DHT) [14] are examples of systems that have managed to maintain the uniqueness of identifiers even as the size of the namespace grows. However, as they become larger and more complex, the cost of administration and maintenance of identifiers and their bindings becomes non-trivial. Initially, Flat namespaces were not considered because of concerns about their ability to preserve uniqueness under scaling pressure. Hierarchies have emerged as a solution to the challenges posed by flat namespaces in maintaining the uniqueness of identifiers as the environment grows [78].

A hierarchical system is composed of interrelated subsystems, with each subsystem being hierarchic in structure until the lowest level of the elementary system is reached [120]. A divide-and-conquer approach was employed using a hierarchical tree to manage large namespaces. This approach allows the namespace to grow beyond the constraints of any single node. This contrasts with the finite size of flat namespace environments and effectively circumvents any limitation on the number of names that a single node can hold.

The identifiers in this context contain a discernible structure by grouping objects that represent their positions in the hierarchy [24]. Hierarchy has several advantages over flat spaces. First, the separation of concerns in large namespaces can be achieved by creating smaller administrative domains, each of which can allocate identifiers from a namespace that does not collide with others in the hierarchical tree [89]. This also enables the independent governance of the namespace subtrees for administrative and scalability purposes.

Second, while resolution through the hierarchical technique may sometimes be slow, it provides an efficient search for identifiers and their associated pointers in the namespace. This hierarchy allows for simple search functions that do not require the querier to assemble a local copy of the entire space for resolution. During a query, only the identity of the name server for the queried label must be locally stored.

Its tree-like structure allows navigation to traverse each parent and child node until the identifier is matched. Unlike hierarchical namespaces, flat namespaces inherently store identifiers sequentially. However, the task of searching in an unstructured flat space may not pose a significant challenge, given that an appropriate sorting mechanism is employed. The transformation of an unstructured data set into a structured or sorted set serves as the key to efficient search operations. The index can then be used for searches, thereby reducing the time required to find an identifier. By contrast, hierarchies provide an alternative approach that does not require the assembly of all data at a single point in time.

Finally, the reusability of the identifiers in a hierarchical structure is an important advantage. Each part of a hierarchical name is resolved relative to a different context of relatively small size. The same identifier can be used with different semantic meanings in different contexts to suit a variety of uses [24]. In the DNS context, it is important to note that there is a distinction between names and labels. A name is an ordered sequence of labels, where each label acts as a node in the name hierarchy. The equivalence of two names is established if they both comprise the same set of labels in the same sequence. This facilitates the utilization of a single label for multiple names without conflicts or ambiguity.

D. INTERNET CONSIDERATIONS

There are two essential preconditions for effective communication between the two entities. First, communication is defined by a common symbol space and its usage. Second, a common semantic interpretation of the composed symbols is essential for preserving the intended or original meaning of the message. Failure to satisfy these two preconditions results in communication failure [56], [116]. A communication network relies on a unique symbol space with consistent semantic interpretation to ensure that any entity can understand the intended meaning of a message received. These strict rules are crucial to prevent symbols from being interpreted differently, leading to the loss of the original meaning [56].

The uniqueness of a common symbol space, the namespace, is crucial for ensuring the unambiguous usage of Internet identifiers, allowing them to be passed on, referred to, and reused without semantic ambiguity. However, the growth and scaling of the Internet have led to the evolution of its name system mapping functions, allowing a name to refer to a constant digital object with a location relative to the querier rather than a single global locator. In addition, when a namespace avoids enclosing a unique location into its name value, issues related to the replication and distribution of digital objects become easier to reconcile. For example, if a digital object is assigned a unique identifier that is not location-specific, replication and distribution can occur without updating it.

Referential integrity in Internet namespaces ensures a reference, when transferred from one entity to another, consistently signifying the same object across the Internet. This objective eliminates the ambiguity in using these references and ensures a shared understanding of communicated terms as valid references. The system's capacity to accommodate Internet dynamism is of equal importance by regularly updating association pointers relevant to the location of the querying entity [36], [37].

For the Internet, as a public communications medium, to maintain its utility as a public service, its associated identifier namespace must be operated as a coherent public namespace [37]. In this context, fragmentation refers to the potential division of this unified namespace into smaller, disjointed parts. Such fragmentation poses a significant risk to the future of the Internet as it could lead to inconsistencies in symbol interpretation across different systems. Specifically, fragmentation can challenge the coherence of the Internet, which hinges on universal reachability, consistency in symbol interpretation, and referential integrity [53]. As described in Figure 2, consider the practice of using the '.local' top-level domain, a common strategy employed by Apple to create a variety of context-specific local namespaces. These namespaces are intentionally limited in visibility and cannot be accessed by standard DNS queries into the public namespace. This approach offers a unique method for creating application-specific namespaces yet concurrently introduces complexities in delineating the global public namespace of DNS from these local domains.

These complexities intensify when the constraints on these local namespaces fail or in scenarios contemplating the delegation of the '.local' domain as the global scope domain. Therefore, ensuring name coherence in local and global network environments is a substantial factor. The issue of namespace fragmentation highlights the significance of identifiers in a broader context. Name systems have traditionally existed in a realm associated with a common infrastructure, where any application can use this common identity system, irrespective of the context of the query into the system, the result the name system provides in response is the same. 'Who' is asking does not materially alter the response. However, adding context and qualification to a

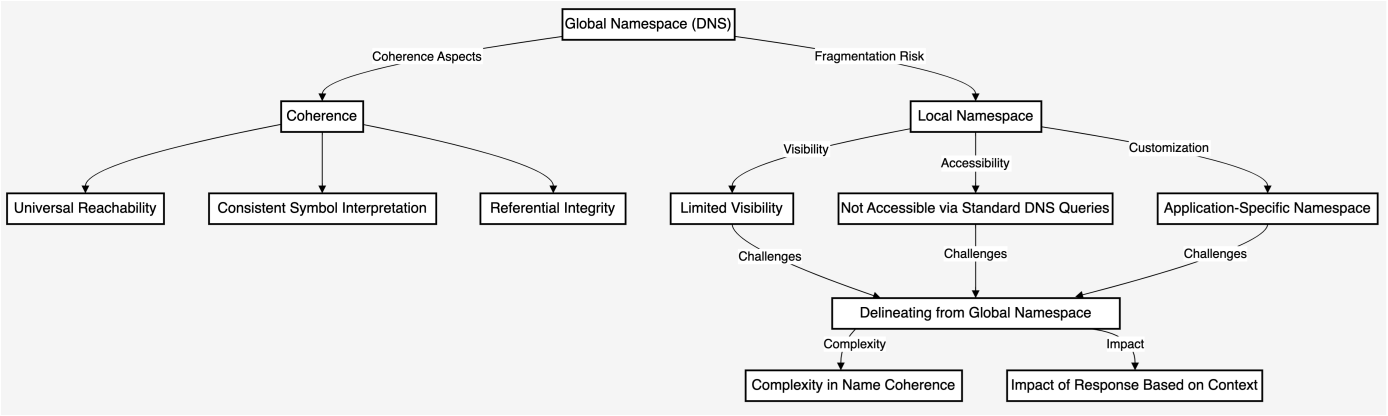


FIGURE 2. Global namespace coherence and fragmentation risks.

query in an identification system can affect the response received because the identity of the querier is a factor, similar to how the 'Client Subnet' in the DNS [23] can impact the response.

It is debatable whether a single infrastructure name system can effectively handle various needs and applications or if a customizable namespace is required. Customized identification systems can be more specific to the realm of use and provide more useful utility models for automated applications. However, the translation problem persists when attempting to transfer an identifier from one context to another.

E. TECHNOLOGICAL INFLUENCES ON IDENTITY AND LOCATION

Identity and location have gained increasing importance in the digital realm in contrast to their traditional usage. This can be attributed to two key factors, mobility and cloning. The early digital environment consisted of mainframe computers dispersed across various floors and buildings, with network connectivity predicated on a fixed model that firmly enmeshed identity and location. However, with the advent of portable computers, the Internet has needed to support an always-connected model that accommodates endpoint mobility. The evolution of mobile cellular services exposed an aspirational goal for endpoints to retain a constant service and context, even when transitioning between different network connections, each with a unique IP context.

This led to a challenge in which IP addresses were viewed interchangeably by different network elements, making them difficult to distinguish. For example, from the routing system perspective, an IP address serves as a locator, whereas from the Domain Name System (DNS) [79] perspective, names are isomorphic to IP addresses, and the IP address acts as an identity that is treated as such by applications [77]. In scenarios where mobility is a critical attribute of the network architecture, the care-of addresses serve as temporal locators only, as endpoints transition between locations and providers, necessitating a new IP address for the device. However, this frequent change in IP addresses disrupts the continuity of ongoing sessions, thereby introducing a unique challenge for mobile networks. Section III-D discusses this challenge and explores strategies to address frequent changes in IP addresses, considering the duality of identity and location inherent in mobile networks.

In contrast, when a file is copied between two systems, it generates two separate named duplicates. Each file's identification incorporates a distinct path namespace, leading to unique references despite the identical content. This raises challenges in identifying digital objects in a world where duplication and replication are necessary to optimize content delivery on the Internet.

In the digital world, the proliferation of cloning has raised challenges in identity and location. For instance, within a library system, there may be numerous physical or digital replicas of the same book, all of which are referred to as a single instance. Regardless of their location, these books are identified by a common identifier, often as a persistent URL. Library systems commonly employ key registry systems such as the International Standard Book Number (ISBN) and Digital Object Identifier (DOI).

Although the URL model is more intrinsically useful for locating the object being sought out, it is less useful when the content might exist at multiple locations. Content Delivery Networks (CDNs) address this challenge by replicating content in proximity to user requests. This involves employing aggressive caching, geo-replication, and traffic engineering [91]. The fundamental challenge in digital environments is to maintain multiple instances of the same entity at different locations. This issue is compounded when the aim is to steer clients optimally towards the instance of a digital object that delivers the best efficiency or speed for each individual client.

Two strategies have been developed to address this problem. The first strategy involves a selective response to DNS queries for the named digital object. The system returns the IP address corresponding to the 'optimal' instance of the object for the client. Akamai illustrated this method well,

using DNS as a geo-locator and modifying the response depending on the assumed identity of the queried [114]. In this context, IP addresses serve as geographical selectors.

On the other hand, Cloudflare's approach represents the second strategy. This involves using a small set of IP addresses announced into the routing system from multiple origination points. This strategy, known as anycast, allows the routing system to locate the optimal instance of a named object, leaving the DNS unaltered [15]. Both strategies enable URLs to mimic and behave like Uniform Resource Identifiers (URIs) [5]. This allows the establishment of a persistent identity while allowing the domain portion of the URL to serve as a dynamic locator.

In summary, the concepts of location and identity can often be interchanged using various mapping system functions. For example, in a mobility scenario, a device maintains its constant identity while its location changes. However, when digital objects are cloned or replicated, they simultaneously maintain a consistent identity across multiple locations. To support location and identity on the Internet, the burden shifts to different systems. For instance, in the case of mobility, updating the DNS to reflect the new IP address mapping when a device changes its location or informing the routing system of the device's new location helps to retain its IP address. In a scenario in which content entities are cloned or exist in multiple locations, certain strategies are necessary to maintain the URL as a persistent identifier. One strategy involves DNS to provide customized responses to clients' queries. This can include appending meta-data to the query resolution, such as using the source IP address to infer the querier's location. An alternative strategy is the anycast approach, in which the routing system independently guides the client to the optimal instance of the content entity. Each discussed approach for mobility and cloning has its own design trade-offs, making it suitable in certain contexts and not in others.

III. IDENTIFIERS VERSUS LOCATORS: PARALLEL INSIGHTS INTO LIBRARY AND INTERNET SYSTEMS

This section examines the roles of identifiers and locators in object identification, focusing on the differences between library and internet systems. Starting with an analysis of the International Standard Book Number (ISBN) and Digital Object Identifier (DOI), this discussion explores how these persistent identifiers operate outside the typical Internet namespace, drawing insights from digital libraries and publishing. The section then shifts to U-systems, including the URL, URI, URN, and IP namespace, to compare identity models across these varied contexts. A summary of distinct approaches to identifiers is presented in Table 1.

A. INTERNATIONAL STANDARD BOOK NUMBER

This section explores important registry key systems that publish persistent identifiers outside the Internet namespace to provide insight into information management and resource identification for digital and non-digital resources. The examination of registry key systems such as the International Standard Book Number (ISBN) and Digital Object Identifier (DOI) [88] in Section III-B are useful examples of the challenges they solve in the transformation of various media types.

ISBN is a separate model from the publishing industry to create a unique identifier independent of the media type, e.g., hardback, paper, and digital. The purpose of ISBN is to associate content with a unique identifier [44]. The International ISBN Agency manages the central registration of ISBNs, ensuring that each allocated ISBN is unique and corresponds accurately to its intended media. Importantly, the ISBN space is internally structured, facilitating the unique identification of the media.

When multiple copies and media types exist, identifying specific instances without a unique identifier can be challenging. For example, if two books have the same title, author, and publisher but different publication dates and editions, it may be difficult to discern which book is being referred to. The ISBN solves the distinguishing problem of assigning a unique key to a book. While numerous copies and media types may exist, the ISBN serves as a unique identifier for each book edition. However, the ISBN does not provide information on how to locate or retrieve a specific media instance. To find the media and its type, a separate catalogue search system or database containing a mapping function is required to determine the availability and retrieval options. In addition, to resolve the ISBN, one must use its unique identifier as an input to retrieve publishing information related to the identified entity [88].

Once an ISBN is assigned to a book or monograph, it identifies the exact entity to be referred to. This identifier remains static, and should the content be revised, a new ISBN should be issued. This contrasts sharply with the characteristics of a Uniform Resource Locator (URL) [11] on the World Wide Web (WWW). Unlike an ISBN, a URL does not specify the exact content retrieved but points to the location where it can be found. Moreover, the content associated with the URL is dynamic and can change over time. Content may also change locations, which poses a challenge for assigning persistent identifiers, as discussed in Section III-B.

B. DIGITAL OBJECT IDENTIFIER

The Digital Object Identifier (DOI) system assigns persistent identifiers to content entities, whether physical, digital, or abstract [88]. Although both ISBN and DOIs offer persistent identifiers for content entities, the DOI system was developed as an extension of the ISBN system to address challenges in the digital landscape. The handle system influenced the DOI system [124], created by the Corporation for National Research Initiatives (CNRI) as a framework for managing digital information. The handle system includes a global naming service that stores persistent identifiers, called handles, for digital resources and resolves those handles into the information needed to locate, access, and utilize the

resources. The handle remains a persistent identifier, whereas its associated values can change to reflect the current state of the identified entity.

Libraries and publishing communities were among the first to adopt Digital Object Architecture (DOA) [63], [64] because they recognized the need to uniquely and unambiguously identify content entities rather than reference them by location, which is an issue with existing identifiers such as URL. Although some URLs are stable, they lack version control mechanisms to track changes to specific versions that users might be interested in. Furthermore, when digital objects identified by URLs move between locations over time, these URLs can become invalid, causing additional challenges in updating the citation indices. This problem of maintaining a consistent identity with the associated metadata, exacerbated by the potential instability of URLs, is especially relevant for libraries responsible for digital preservation [48].

The Handle System was developed outside the DNS system, enabling the creation of location-agnostic qualification strings, thus forming a separate namespace [34]. However, extending the DNS as a general-purpose name service encountered two main challenges. First, DNS operates on a zone-level management structure that lacks administrative capabilities at the individual name level. This presents a considerable challenge regarding the fine-grained control and governance of digital identifiers, especially for entities such as publishers who require individual control over a multitude of digital objects. Second, administrative controls cater to network administrators, making them suitable for domain administration but less so for general-purpose administration [124], [125].

The DOIs are governed and administered by the International Digital Object Identifier Foundation (IDF), which publishers use. The DOI system builds on the Handle System, an integral component of the Digital Object Architecture (DOA) [48]. Like the Handle System, DOIs were initially mapped outside the DNS, requiring specialized client-side software to resolve handles [10]. However, as publishers and service providers were responsible for making DOIs resolvable to digital entities, the namespace was eventually integrated with the DNS using a URL scheme. By mapping DOIs to DNS, users can access and retrieve information through a common web browser without requiring client-side software, making the DOI system more accessible to a broader audience.

In summary, the identity representation of the ISBN and DOI is a persistent, unique identifier for a specific instance of a manifestation. All instances within the same identifier are considered complete and precise clones. The ISBN and DOI serve as identifiers that distinguish a member of a particular class (e.g., a book) from others. This system is location-independent, meaning that the location of an entity is not essential for identification. In this context, ISBN and DOI systems rely on a categorization-based identity that is exclusionary in nature because content entities without an ISBN or DOI are not considered part of the specified class.

C. URI, URL, AND URN

In networked systems, a resource identifier serves as a compact surrogate for a resource, enabling identification, retrieval, or both functions. Resource identifiers are represented as textual strings of characters that must be resolvable to yield the associated resource [62]. In the context of the World Wide Web (WWW), a Uniform Resource Identifier (URI) [10] was introduced to define a method for identifying digital object instances, such as documents, images, videos, web pages, or services that exist anywhere. Similar to ISO Object Identifiers (OIDs), URIs were primarily intended to serve as pure identifiers rather than retrieve, dereference, or locate a resource [2], [8], [10]. In addition to ISBN and DOI, a URI is considered a location-independent identifier, reinforcing the concept of a web object identity as a generic identifier [9].

URIs provide resource identification via a naming scheme or other resource attributes. Generic URI syntax specification does not imply the inclusion of names and addresses from various namespaces to be mapped in the URI strings [11]. URIs are general constructs in which the initial ‘‘scheme’’ part of the URI determines the structure and semantics of the remainder of the URI string. The following example demonstrates an abstract identifier consisting of a sequence of components referred to as the scheme, followed by a (‘‘:’’) and then a string (scheme-specific-part) [10].

scheme : scheme-specific-part

The URI syntax does not impose a specific structure or set of rules for the scheme-specific part of the URI. URIs, although relatively niche on the Internet, are not widely used. Their limited traction might be attributed to their amorphic and insubstantive nature, offering less immediate utility than URLs’ tangible benefits. Conversely, the URL, a subclass of the URI, has been widely adopted as it appeared easier to use. The URL identifies a resource by representing its primary access mechanism [74], thus establishing it as a functional form of identity that is location-dependent. The URL syntax enforces a specific structure to represent hierarchical relationships within a namespace. The following HTTP [40] URL example consists of a sequence of components referred to as the scheme, authority, path, query, and fragment.

https://example.com/articles/technology?view=summary

In this example, the scheme defines the Hypertext Transfer Protocol Secure (HTTPS) protocol [104], which is used to access the resource. The authority includes the domain name ‘example.com,’ which resolves the identifier to a specific host. The subsequent path, such as ‘/articles/technology,’ represents an internal path within the local filestore of the identified server. Finally, the query component pro-

TABLE 1. Identity models and comparisons.

Approach	Focus	Identifier Type	Advantages	Disadvantages
Location-Independent Identification	Describes 'what' the identity. Identity by categorization or classification. Identifiers that are exlusionary in nature.	Unique and Persistent identifiers (URI, URN DOI, ISBN)	Persistent identity, independent of location, effective for managing content across multiple instances	May not be suitable for resolving dynamic content, can be complex to implement and maintain, may provide multiple answers, including misleading ones, due to the vast amount of information associated with locators.
Location-Dependent Identification	Definition of 'where' the entity is. An Algorithmic-Based approach.	Locators with Specific Server-Side Instructions (URL)	Simple and efficient for one-to-one mapping of host to resource..	Not effective for content in multiple locations, can be vulnerable to changes in location. Requires CDN and DNS support for resolving URL to multiple locators.
Hybrid-Identification	Complementation of 'what' and 'where' the entity is.	IP Addresses, unique identifiers and locators	Duality expressed in the IP namespace is complementation between the two identity models.	Overloaded semantics, significant privacy concerns due to persistent identity tied to IP, potential for tracking and profiling.

vides additional information to the server. In this case, 'view=summary' indicates a request for a summary view of the technology article. In the example provided, the URL builds on the DNS System. The DNS name is contextualized as a reference point, while the rest of the URL have semantic significance and are subject to server-side translation. In the context of URLs, a domain name reliably identifies the designated host. However, changes in resource location pose challenges. The DNS has solutions for this, with Canonical Name (CNAME) records mapping an alias to a true domain name and Delegation Name (DNAME) records redirecting an entire domain subtree to another domain. These features ensure stability at the domain level despite changes in resource locations. DNS also incorporates advanced mechanisms such as Naming Authority Pointer (NAPTR) records. Rather than merely redirecting, a key role of NAPTR records is to phrase new queries through name transformation. Section IV-C provides more details on these DNS techniques.

In contrast to the challenges associated with the instability of URLs, Uniform Resource Names (URNs) are defined as a subclass of URIs that provide a persistent identifier for resources, independent of their location and access method [109], [123]. The resolution process for URNs is more complex than that of URLs. With URLs, the access mechanism is subsequently used to dereference the locator, retrieving a representation of the associated resource, such as a document [10], [109]. In contrast, URN resolution involves mapping a URN to one or more URLs, which still requires dereferencing the mapped locators to one or more resource representations. The key distinction in this identity system is persistence [109].

As a conceptual example, when a user interacts with a URN, the browser returns a set of candidate URLs, selecting one based on the location or access method to retrieve the resource while remaining entirely transparent to the user of the system [2]. However, similar to the challenges with URLs, if the locators change, such as the DNS name, the system must refresh and update the list to ensure that the URN remains resolvable and continues to yield the desired resource. The URN example identifier includes the scheme, namespace identifier (NID), and namespace-specific string (NSS):

urn:namespace-identifier:namespace-specific-string

In this example, the scheme identifies the type as URN. The URN includes a NID maintained by the Internet Assigned Numbers Authority (IANA) [126]. Examples of namespace identifiers include "NBN" for National Bibliography [46], 'ISBN' for International Standard Book Number [47] and 'ISSN' for International Serial Standard Number [108]. The NSS contains an opaque flat string within the NID that provides the format rules, such as a unique bibliographic identifier from the NBN.

There are several reasons that URNs have not been widely adopted. One reason is that retrofitting the Internet to include the complex mapping of URNs to dynamic locators, which was introduced relatively late in the development of the Internet, created maintenance challenges regarding how URLs would be updated and refreshed as resources changed locations or cloned [22]. Additionally, URNs attempt to perform too many roles at once without a clear focus on one purpose, such as providing a persistent URL, location independence, a resolution system, or a pure identifier [2].

D. INTERNET PROTOCOL ADDRESS

The previous sections discussing ISBN, DOI, and URI have revealed two distinct approaches to identity. The first approach involves assigning identifiers, such as ISBN or DOI, to each instance of a content entity that is an exact clone. In this model, identity is determined by the entity's membership within a specific class, unlike other existing classes. The second approach is an algorithmic-based method for identifying the location where the content entity can be found. In this model, identity is location-dependent. Aspects of identity, unique location and unique categorization are essential for defining an object. These two models of identity, 'what' and 'where,' complement each other in providing a useful namespace. The IP address [99]

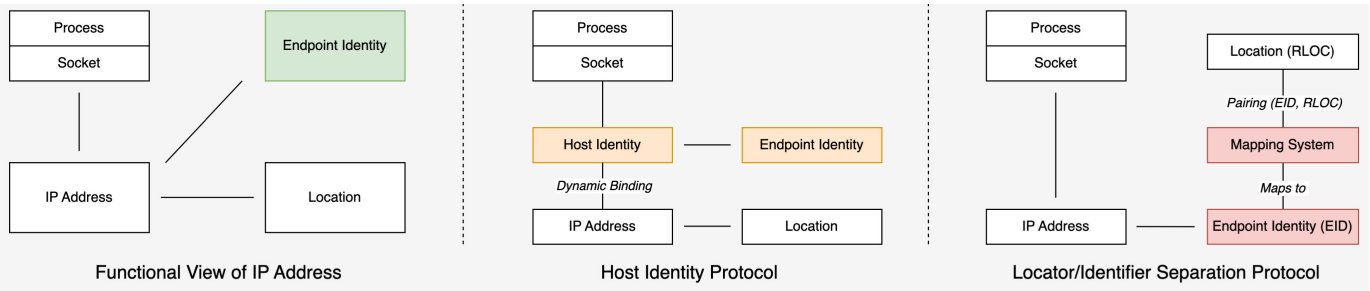


FIGURE 3. Role of an IP address: Example implementations for identity and location separation.

can be considered the ultimate complementation of these two identity models. However, IP addresses are heavily overloaded semantic because they attempt to serve multiple purposes simultaneously [16], [77].

Fundamentally, an IP address [99] is used to identify the point of attachment of a host to a network [117]. Unlike prior address spaces such as DECnet, where all interfaces share a common identifier to represent the host [97], an IP address does not equate to a host. A deliberate decision was made in IP to adopt a more functional view of the address and address numbering. The function of this identifier is to indicate how the network interface of a host is reached. For example, when the forwarding system matches a packet's destination address, it directs a packet to the network interface associated with the host [110]. As traditional networking views assume a fixed model, the identifier is spatially and temporally unique and identical to its locator [103].

Consequently, an IP address not only identifies the host's interface but also serves as a means to access it [37], [117]. The dual role of IP addressing, serving as an identifier and locator, is inherently advantageous but also introduces the potential for confusion. This duality has been widely embraced, largely owing to the static nature of hosts, which permits a level of permanence in address assignment.

Mobility was never considered an attribute that required attention until the realization that all endpoints - devices, services, and content might need to be able to frequently change network attachment [70], [127]. Including mobility as a network capability involves two approaches when an endpoint changes location. The first approach consists of updating the network's forwarding information to reflect the new network location, which maintains service continuity but burdens the routing system. Alternatively, the address of the attachment point could be adjusted to reflect its new location. This approach reduces the impact on the routing system but may cause service disruptions during location changes.

Although IP addresses are heavily overloaded semantic, as they attempt to serve multiple purposes simultaneously [16], [77], deliberate efforts have been made to develop various techniques for disambiguating location and identity to address mobility challenges. These approaches, extensively covered in the literatures [66], [101], [122], [130], and [137], share three key components in their design principles: a stable identifier for a mobile endpoint, a locator representing the IP address of the current location, and a mapping function to associate the identifier with the locator [130]. Refer to Figure 3 for a representation of these principles.

Within the context of architectural support for mobility, there are two main techniques: handling host relocation, which is transparent to the network using a level of indirection in the routing architecture, and transport-level approaches, which ensure that sessions are sustained in the event of an endpoint IP address change. Although numerous methods have been proposed, the following examples illustrate using the IP address as a network namespace.

Mobile IP [92], [93], [96], the Home Address (HA) is elevated from its role as an interface identifier to a constant endpoint identifier, independent of the host's point of attachment within the network domain [121]. The Care-of Address (CoA) serves as a temporary locator, changing as the endpoint moves between the network domains. The home agent maintains a binding between the HA and CoA, which the architecture utilizes when an endpoint transitions between the locations. The home agent is responsible for managing the mobility within the network, primarily by establishing tunnels [72], [95]. This facilitates seamless packet delivery to the relocated endpoint [94]. Importantly, this added layer of indirection localizes the implications of a location change, affecting only the devices engaged in communication with the mobile entity. This approach avoids burdening the entire routing system with an incremental load, thereby promoting efficiency and continuity in mobile networking environments.

The Host Identity Protocol (HIP) [80], [86], described in Figure 3, creates a new namespace between the IP layer and transport protocols. In this namespace, the host identifier is a public cryptographic key that allows hosts to authenticate to their peer hosts directly using their host identity [85]. The HIPs identity namespace replaces the traditional role of IP addresses in binding communication sessions, providing a stable connection if the endpoint's IP address changes [85]. The role of the IP address in the model is for routing purposes, creating a location/identity separation in the architecture.

The Locator/ID Separation Protocol (LISP) [38], described in Figure 3, creates a disjoint IP namespace where the Endpoint Identifier (EID) is decoupled from the locator

(RLOC) and is stored in its mapping system. LISP provides topological aggregation, leveraging a dynamic locator approach for IP routing, similar to the earlier effort of Pseudo-IP (PIP) [41]. LISP employs UDP encapsulation for its delivery mechanism, with the outer IP header containing the destination Routing Locator (RLOC). Based on this RLOC, the network core forwards packets to the destination's LISP-speaking gateway at the edge. The edge gateway is responsible for de-encapsulating the packet and subsequently forwarding it to the endpoint interface by utilizing the EID to determine the original destination.

Despite various proposals for separating endpoint identity from location, there are known disadvantages. For example, there is the added cost of administrative overhead, additional infrastructure, and the need for extra levels of indirection through several encapsulation techniques, each with inherent challenges in the Internet's routing architecture [71]. Another concern is privacy, as IP addresses visible in IP packet headers serve as identifiers and can be associated with a client's identity. A skilled attacker in the path between the client and destination can potentially infer the client's identity and activity on the network through passive monitoring [39]. To address this privacy issue, efforts have been made to create temporal identities by generating randomized identifiers, particularly in IPV6 [29], [82].

In a recent effort to address the mass consumer market to ensure privacy protection is more accessible to Internet users, Apple's iCloud Private Relay was designed to safeguard user privacy online. This service protects user privacy by concealing their online activities from network intermediaries and passive observers. The principle of a Private Relay is that IP addresses identifying users should be separated from the names of websites users access [57]. As described in Figure 4, the architecture consists of two tiers: ingress and egress. The responsibility of the ingress relay is for clients to establish a connection in which the original IP address is visible; however, the website names requested by the client are encrypted. Proxying traffic through the ingress relay shields the client's IP address from the egress relay, third-party intermediaries and destination server [111]. The technique employed by an ingress relay hides the client's IP address while exposing sufficient information for the egress relay to understand the client's location for address allocation purposes. The ingress relay converts the client's IP address into a 'geohash,' representing the user's location coordinates. Consequently, as the client communicates with the egress relay, the client's IP address is not forwarded. Refer to Figure 5 for a visual illustration of this sequence.

The egress relay performs three primary functions. Firstly, it receives the 'geohash' from private relay clients, then consults its database to identify and assign the nearest corresponding IP address to the client. Second, it decrypts the website name, requested by the client. Finally, it completes the connection between the client and the requested website. The egress relay is not privy to the user client's original IP address. The egress relay only receives location information

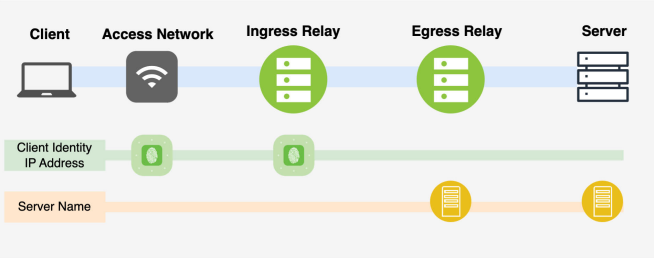


FIGURE 4. Apple relay: Anonymizing client's identity.

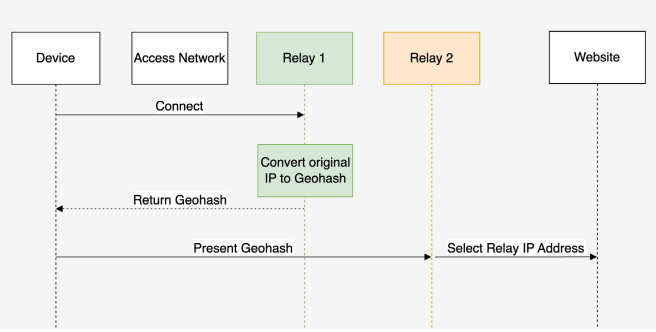


FIGURE 5. Ingress and egress relay: IP address identity, and location [57].

to assign the client the appropriate locator, which maps to the region from which the request originated.

Because the egress relay has no knowledge of the original IP address of the user, the Relay IP addresses rotate over time and between sessions, preventing their use as stable identifiers for the user. In addition, the iCloud Private Relay utilizes QUIC [60] to connect to the ingress relay. The tunnel to the egress employs a secure proxying protocol using the HTTP/3 [12], [90] proposed by the Multiplexed Application Substrate over QUIC Encryption (MASQUE) IETF working group [81]. Using QUIC as the transport protocol, MASQUE provides a secure connection that can combine multiple connections within a single proxy connection [111]. As part of this architecture, QUIC introduces a unique approach for sustaining transport-level connections, which is particularly useful in mobility scenarios for roaming clients. Unlike previous transport-level approaches [61], [121], QUIC provides a semantic distinction between the identity of a connection to an endpoint and the current IP address and port numbers, similar to HIP [1] and Serval [87]. QUICs connection endpoint identifier allows sessions to survive if the IP addresses and ports change [1]. For instance, if a client switches from a cellular to a Wi-Fi network, the QUIC connection ID allows the session to be recognized as an ongoing stream, irrespective of changes in the source IP address or UDP port numbers [1]. In summary, the fundamental changes brought about by mobility exert pressure on the IP namespace. To complement location and identity, distinct roles are assigned to IP addresses, requiring additional mapping systems to create disjoint namespaces, thus decoupling identity from location. In essence, names become isomorphic to IP addresses, and IP serves not only as a locator but also as an identity.

IV. SEARCH VERSUS IDENTIFICATION: COMPARING DIRECTORY AND DNS APPROACHES

This section reviews the transformation of HOSTS.TXT into a distributed database and provides an overview of the initial namespaces that are critical to the early stages of Internet development, such as HOSTS.TXT and X.500. The focus is on a thorough exploration of the theme of search versus identification, offering a comparison and contrast between the X.500 and the DNS. The section then examines the evolution of the DNS in lookup design, including the adoption of successive generations of resource records that meet evolving business needs for flexibility, currency, and speed.

A. HOSTS.TXT

In the early development of ARPAnet [50], the HOSTS.TXT file served as a directory to provide a human mnemonic for identifying resources, i.e., the printer [30], [65], [67]. Each line in the host table file contains a human-readable label, a network address, and protocol-specific information supported by the resource [46]. An identical copy of this host table exists in each host. The Network Information Center (NIC) at the Stanford Research Institute (SRI) was responsible for managing the registrations and distribution of the HOSTS.TXT file. As each host was registered to the network or changes were made to label mappings, the NIC would maintain that the host table is current and distribute it to every host on the network.

However, as ARPAnet grew in popularity, the increased frequency of host table updates and its distribution became burdensome and prone to failure. Consequently, the following challenges surfaced when using the HOSTS.TXT file: First, the host table approach assumes that changes occur infrequently and that the environment is constrained to a small number of hosts. The increased frequency of host table updates results from an increase in the number of hosts growing on the network. The increased frequency of updates posed a challenge for maintaining HOSTS.TXT so that network hosts could receive the latest version in a timely manner. Subsequently, the central server where HOSTS.TXT resided was overwhelmed by the number of download attempts, exacerbated by the HOSTS.TXT growing in size [25]. Second, the flat architecture of the HOSTS.TXT namespace introduces a significant challenge. Registrants who did not have the most up-to-date version of the host file could potentially attempt to register an already assigned name. Therefore, the registration process necessitates rigorous checks to maintain the uniqueness of entries in the host file [21].

While the NIC could uniquely assign addresses from the address space, it had no controls to prevent an administrator from erroneously entering a host with a conflicting name. Moreover, host table specifications were established to provide names with a discernible structure and promote ease of recall. The specifications include format rules, such as limiting names to no more than 24 characters composed of case-insensitive letters (A-Z), digits (0-9), and hyphens (‘‘-’’). Additionally, names were prohibited from starting or ending with a hyphen and containing spaces or any other special characters [49]. In summary, the growing success of ARPAnet illuminates the inefficiencies and limitations of the host table file approach, highlighting its potential points of failure. The need for a more efficient and timely system became apparent, shifting the focus toward developing an on-demand query resolution system using a distributed data model. This new direction aimed to replace the static distribution of a host file with a dynamic model, facilitating timely updates and improved overall efficiency. This represented a critical step in the search for a scalable, manageable, and globally available system that can handle the increasing demands of a rapidly expanding digital network [73].

B. X.500

The Open Systems Interconnection (OSI) reference model was standardized by the International Organization for Standardization (ISO) to create a vendor-neutral network architecture. The primary objective of this initiative was to devise an abstract network architecture that promotes inter-operation rather than fostering dependency on a particular vendor’s implementation.

Thereafter, the X.500 Directory Service was developed as an integral part of the OSI’s effort to manage network resources effectively. To address this issue, the X.500 directory service was created and standardized in ISO/IEC 9594 [102]. The X.500 directory is a logical database containing information about objects that model the physical world. The directory service manages names and their associated attributes and provides access to read and retrieve information in the database [98]. An important distinction of this directory system is that the returned answers are independent of the identity or location of the client that initiated the query [106]. Therefore, similar to a telephone directory, the same answer is provided irrespective of the querier.

Although X.500 shares similarities with a telephone directory, key differences and limitations exist when applying these models in the digital space. For instance, a telephone directory can only return a telephone number when the complete name and address of the provider are given [105]. Otherwise, the telephone directory cannot facilitate the query without providing these detailed attributes to disambiguate across all entries. Additionally, physical telephone directories are published no more than once a year, resulting in incorrect and outdated information that cannot be refreshed until the next publication. By contrast, X.500 addressed these challenges through its global namespace design, allowing a hierarchical search to facilitate incomplete queries based on its structured information model. If additional information is required, the attributes can be updated promptly to ensure the presence of distinguishing factors. For example, If two entries have the same first and last names, the supplementary information can be updated [98].

The implementation of the X.500 directory is based on the Directory Information Base (DIB) and Directory Information Tree (DIT). DIB is a structured information model composed of information that classifies objects called entries. Each entry consisted of one or more attributes. In addition, each attribute type contains one or more values that define the attribute characteristics [106]. DIT defines the relationships between entries to allow for a hierarchical search. The Distinguished Name (DN) uniquely identifies each entry in the Directory Information Tree (DIT) based on its location within the DIT hierarchy and the associated attributes, which are expressed through the namespace assigned to the object [83].

Each attribute associated with the entry is called a Relative Distinguished Name (RDN). To find a given entry's distinguished name (DN), the search starts at the tree root and traverses top-down until the entry is found. The DN concatenates each entry's relative distinguished names (RDNs) traversed [106]. For example:

countryName = Australia
organization = UTS
organizationalUnit = Faculty of Engineering & IT
commonName = Andrew Martin

As described in the above example, the upper levels of the hierarchy are based on the country, organization, and organizational unit. This approach was designed to describe the geographic location of an object. Country, organizational, and organizational units are helpful for hierarchical search, allowing users to locate candidate objects to choose from [105]. In contrast, other lookup systems, such as the DNS hierarchy, are based on naming conventions using a sequence of labels to construct an FQDN to identify objects that are not necessarily location-based. Additionally, the DNS is not a searchable database and only returns an answer based on the exact query provided. The fundamental difference between these approaches lies in the distinction between search and identification, as depicted in Figure 6. In a directory system, the emphasis is on strict search capabilities that enable users to perform targeted queries and retrieve specific information. Conversely, the DNS operates on a system of exact matches, where all queries and corresponding responses are exact [128].

In summary, the X.500 directory service has many attribute sets to describe its resources and users, which requires coordination to manage multiple points of authority at different levels of the hierarchy. Despite documented enhancements, deployments, and case studies [76], [119], [132], the complexity of the Directory Access Protocol (DAP) limits its widespread adoption. However, it is important to note that the development of a Lightweight Directory Access Protocol (LDAP), a simplified version derived from DAP, resulted in significant advancements and widespread adoption. With its lightweight and streamlined approach, LDAP has been successful in various systems, including the Microsoft ecosystem [51], [69].

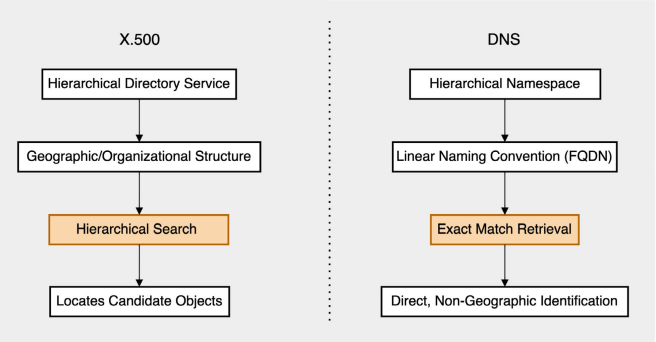


FIGURE 6. X.500 and DNS: Search versus identification.

C. THE DOMAIN NAME SYSTEM

The Internet employs multiple identities, each of which has distinct roles. As discussed in Section III-D, IP addresses illustrate a duality by functioning as identifiers for naming resources and services while also acting as locators essential for packet forwarding within the network. The DNS identity within URLs, detailed in Section III-C, maintains the identity of the service, requiring no modification even when the associated IP address changes. The domain name introduces an abstraction layer over the IP addresses. This allows URL identifiers to point to services without directly specifying their present location on the Internet [37].

The DNS can be described as method-independent [32], [33]. In essence, it assigns names to hosts regardless of the specific type of server or service in use. It functions as a naming system for naming computers or services, similar to how individuals are named in a telephone network, highlighting the notion of endpoint identity. The primary motivations of the DNS was to establish human-friendly identifiers for protocol addresses, while its hierarchical structure was designed to ensure scalability. The DNS established itself of critical importance to the Internet early on in its development owing to several important properties:

- The **Decentralized nature** of the distributed database was used to populate the DNS namespace, eliminating the need for a centralized organizational infrastructure to operate the entire namespace. Each entity operating a delegated part of the namespace can do this autonomously. This decentralization allows the DNS to scale without creating critical bottlenecks, operating with limited coordination with others, which aligns precisely with the broader operation of the Internet itself.
- The **DNS resolution function** operates deterministically, using an exact match between the queried names and DNS names. There is no intended scope within the DNS for ambiguity in the name resolution function, rendering it highly suitable for automation in a deterministic computation environment. This is important because it can support a single unique naming

space across the entire network where the same query generates the same response regardless of the location or identity of the querier. This determinism is the most important reason the DNS is critical to the Internet.

- **Resilience and Efficiency** in name resolution are the defining features of the DNS. The system avoids relying on any single point of failure. Caching results for a predetermined period using the TTL field reduces the load on the DNS resolver, thereby enhancing the efficiency of the system.
- The **Syntax** of DNS names involves concatenating individual labels to construct a DNS name and using a limited set of characters. This design permits the inclusion of natural, human-readable words within DNS labels, although it is constrained to only support ASCII characters in its standard format [32], [33].

The emergence of the DNS represented a significantly simpler form when compared to X.500 in Section IV-B; however, it drew insights from the latter. It also incorporates elements from the antecedents of the DECnet naming system that used a two-level hierarchy. As described in section II-A, each cluster in DECnet was assigned an area name, permitting the allocation of unique names within that specific area. An identical name could be reused in different areas due to its namespace 'AREA::NAME' hierarchy. In contrast to the two-level hierarchy, the DNS design sought to move beyond these constraints. Recognizing the limitations of a two-level hierarchy, DNS is deliberately designed to accommodate an arbitrary number of hierarchical levels. Each node in the hierarchical structure has an associated label. A unique domain name associated with a node is formed by assembling all labels from a specific node at the root of the tree. [79], [133]

In the predecessor HOSTS.TXT [78] approach, the host table file operated on a just-in-case principle, providing all possible host mappings as needed. The maintenance of the HOSTS.TXT file was challenging, particularly when hosts were added, removed, or had their IP addresses changed. This process required every file instance to be updated and distributed to each host, a procedure discussed in more detail in Section IV-A. DNS pioneered a just-in-time query protocol, enabling efficient retrieval of specific records when necessary. Users can query a particular resource and obtain the corresponding answer instead of retrieving all the mappings. This approach assumes a robust and highly available network that enables specific database record queries on demand. One such specific record is the Address (A) Record [33], which created a synthesis between the directory and protocol-level address where the namespace gained greater ascendancy. Addressing the deficiencies of HOSTS.TXT, the A record ensures that when a new host is added or an IP address changes, only the relevant record needs updating, eliminating the need to modify the entire host mapping file. As the Internet's complexity increased, so did the requirements for DNS identifiers. While the A record, which is fundamental to the system, continued to play its central role, it was complemented by successive generations of DNS identifiers. These new iterations introduced additional abstractions and enhanced functionality, meeting the evolving needs of the Internet landscape and, importantly, serving the demands of developing an identifier system. Among these needs is the necessity to accommodate changes in the location of resources while ensuring the permanence of names. With these evolving requirements and DNS advancements, the following subsections will explore three main themes: first, the shift in control of a name in its entirety through outsourcing an indirection to a third party; second, the introduction of a persistent identifier that allows publishers greater control over the resolution process; and lastly, DNS's role as a rendezvous control point, fostering service-level connectivity based on intent.

1) OUTSOURCE CONTROL: CANONICAL NAME RECORDS

As Internet outsourcing expanded, the necessity to relocate resources without altering their names became increasingly apparent. There grew a need for entities to map individualized names into the infrastructure managed by specialized service providers. Although delegating control of a DNS zone to a third party was possible, transferring control of the name proved challenging. This constituted a significant limitation in maintaining a name while letting a third party run the corresponding service.

To address the challenge of shifting control of a name to a third party, a Canonical Name (CNAME) was utilized [33], [35]. With a CNAME record, a name can point to another namespace without official delegation of the zone. This was instrumental in mapping names to a third-party infrastructure. Consider specialized service providers that host numerous entities. As described in Figure 7, these entities could utilize the CNAME record to establish an alias in the service provider's domain by outsourcing an indirection. This strategy enables entities to preserve their individual identities. Simultaneously, the naming system redirects queries to the resources managed by the hosting provider, which contributes to efficient query resolution. Importantly, this process operates in a manner that remains invisible to the end user.

However, the use of CNAMEs introduces unique considerations that have been the focus of industry discussion. Significantly, a CNAME alias controls the name in its entirety. This introduces the issue of handling the apex of a delegation, as CNAMEs cannot reside at the apex; they must be situated at least one level down. Moreover, it is imperative to distinguish that CNAMEs are affiliated with a distinct name, not a hierarchy. In contrast, the DNAME record [107] executes a broader role by redirecting an entire sub-tree of the DNS hierarchy to another domain. Effectively, a DNAME assimilates a name with all its descendants and maps it onto a disparate point within the namespace.

2) PUBLISHER CONTROL: POINTER RECORDS

Building on our earlier discussion of DNS operations, this section delves into the challenges related to the control

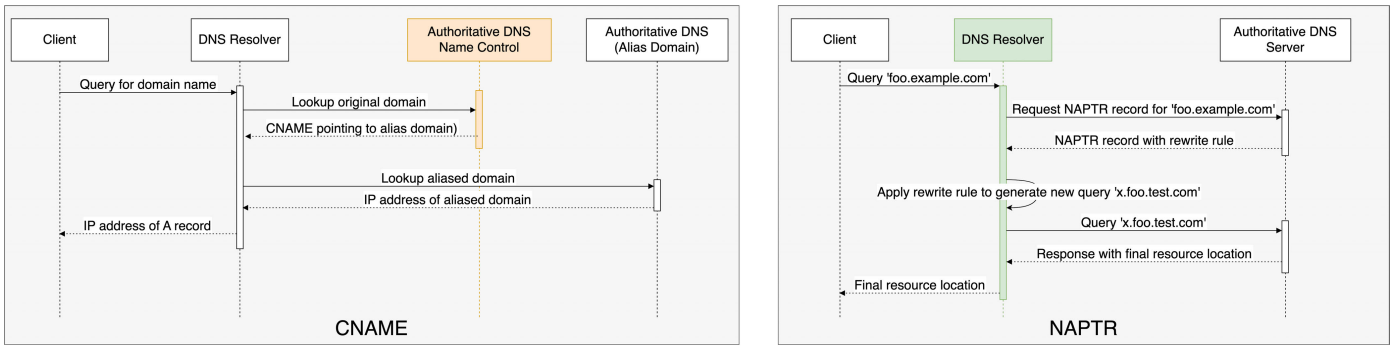


FIGURE 7. DNS lookup process: CNAME and NAPTR records.

of the resolution path, which is an aspect intrinsic to DNS as a lookup service. These challenges stem from the widespread caching of DNS information. When a DNS name propagates, its mapping achieves a certain level of persistence, which can complicate efforts to modify or customize the associated responses. Disseminating a DNS name involves a commitment to resolvability; however, this is not total or indefinite. Publishers retain control via the record's 'time to live' fields, which can regulate how long a specific DNS record remains cached.

As described in Section III-C, a key factor in maintaining a URL's stability is the enduring existence of its associated DNS name, which is a critical consideration when the resource changes locations. In this context, the functionality of the Naming Authority Pointer (NAPTR) [75], a distinct DNS record type, is apparent. The NAPTR record enables dynamic translation into designated target names. It establishes a separate layer that maintains independence between naming and the resolution system, along with its associated protocols and services. This ensures that changes in one system do not directly impact the other [28]. NAPTR records provide a pivotal mechanism to facilitate redirection or transformation, thus reinforcing the permanence of URLs. For example, in Figure 7, if a domain holder intends to relocate a resource from 'foo.example.com,' the NAPTR records serve to redirect incoming requests. Rather than directly resolving 'foo.example.com,' the DNS instructs the resolver to phrase a new query, such as 'x.foo.test.com.' This method ensures the integrity of the original URL, facilitating seamless redirection to the new resource location.

This NAPTR process involves the DNS server returning a regular expression pattern and rewrite rule to the resolver. The resolver then applies this pattern to the original query name and, following the rewrite rule, generates a completely new query. It is crucial to note that this translation lies within the control of the publisher and not the user. This arrangement allows the publisher to relocate content while ensuring a persistent identity for the domain name, thereby transforming the domain resolution process from a semi-permanent to a dynamic state. This technique provides considerable flexibility in terms of rewriting rules.

The Straight-forward-Naming Authority Pointer (S-NAPTR) was introduced as a simplified alternative to the NAPTR, which utilizes regular expressions for name transformations based on specific patterns. By contrast, S-NAPTR focuses on service discovery by working in conjunction with SRV records [45] to provide explicit service details [84]. NAPTR also differs in approach compared to CNAME and DNAME. While CNAME translates one name into another in an iterative process, and DNAME maps all names under one delegation point to another, NAPTR provides a more nuanced approach. The NAPTR offers an adaptable class of names that adheres to a specified pattern and maps them to target names. Although not identical to the original, these target names must conform to the provided pattern. This is in contrast to DNS records such as DNAME, where the query labels to the left of the DNAME point are fixed. In contrast, the NAPTR allows these labels to undergo transformation according to a defined set of rules.

Despite their potential, NAPTR records have not seen widespread adoption owing to their complexity and the perception of unnecessary indirection. Many users find it simpler to customize the namespace directly instead of complicating the resolution process. In the identity system space, the publisher bears the burden, not the user. Systems that impose significant costs or burdens on the user are likely to fail, given the availability of free alternatives that transfer the entire burden onto the publisher. This principle is why systems such as ISBN and DOI, which could have imposed costs on users, continue to exist because they do not impose additional name resolution burdens on the user.

3) RENDEZVOUS AS INTENT: SERVICE RECORDS

The advent of SVCB records [115] in DNS has marked a significant shift from previous DNS record types. Initially, service names were bound to domain names, meaning that all services sharing a logical name were grouped under a common domain name, offering limited flexibility. Services might share a common domain name for various human-use reasons, but there may be a need to disambiguate services and direct them differently, invisible to end users.

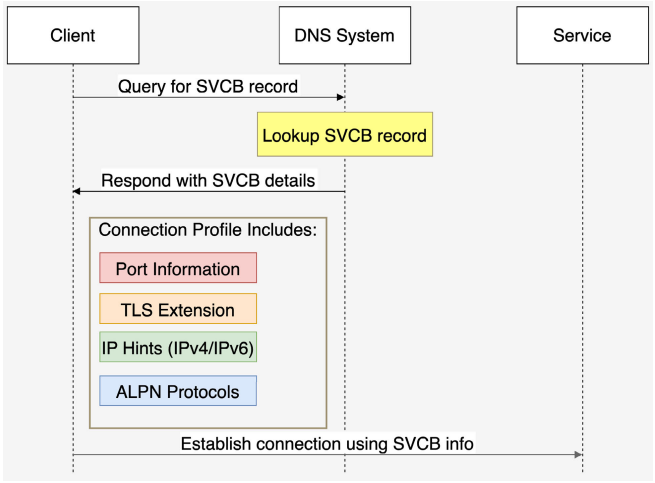


FIGURE 8. Service binding record.

The primary goal of the SVCB record is to allow a client to gain all necessary information to establish a connection to a service with a single DNS query. Moreover, SVCB records provide a DNS aliasing functionality, which is especially useful at the apex of a domain. This functionality allows for the separation of service identity from location, eliminating the need for additional CNAME records and reducing the round trips required for name resolution.

Although SVCB and SRV [45] records exhibit shared characteristics, significant differences exist. An SRV record is non-extensible and has a fixed number of fields. Any alterations to these fields require the creation of a new record type. By contrast, SVCB records can contain any number of key-value pairs, allowing for extensibility with new keys and values. Moreover, SRV records require an extra resolution step because they do not include elements comparable to the IPv4 or IPv6 hints found in the SVCB records. As described in Figure 8, these hints allow clients to initiate connections more rapidly by reducing the number of necessary DNS queries.

In the context of SVCB records, service binding requires encoding the service name, which is prepended in front of the domain name. This follows the Attrleaf naming pattern [26], [27]. For example, _service._port.example.com. With the introduction of SVCB records, DNS transitioned from being a simple locator to acting as a rendezvous broker, enabling more precise, user-intention-driven service discovery. Instead of translating a name to an IP address, DNS interprets the user's intent to determine the specific service they wish to access. This service is defined by a tuple of parameters such as port, protocol, and label, providing a precise description of the service point.

In this evolved model, when a query is made, DNS considers the user's location and the locations of all the potential service points. It then returns a connection profile with detailed information about the requested service, including aspects such as Application-Level Protocol Negotiation (ALPN) [55] and the necessary TLS parameters [42] for the subsequent connection [138].

In essence, the DNS evolved into a rendezvous broker, offering more than a simple IP address. It now provides a means of accessing specific services, thus significantly expanding its role. It is no longer just objects or locations anymore; it is about guiding users to the services they seek based on their intentions. This concept is an abstraction beyond the basic model, making DNS an arbitrator of service access within a namespace.

V. EVOLUTIONARY PRESSURES AND FUTURE PERSPECTIVES

Originally, the DNS modeled the telephone directory, a static system published periodically with invariant lookup keys. This telephone directory was non-customizable and uniform for all users, permitting anyone to use the same lookup key and retrieve identical information. This approach has held appealing properties of a digital identity system. One could share a key, and others could access the same database to retrieve identical results, making it possible to refer to entities via their keys. However, with the evolution of the Internet, the concept of identity systems also has evolved. The use of identifiers, especially within the DNS, began to shift. Instead of treating identifiers as constant indices in an invariant database, there was a growing need to customize these identifiers according to the needs of individual users. In the currently evolving DNS, queries are also based on the intention to generate a more comprehensive set of information, outlining the specific steps required to fulfill that intention. These intention-oriented queries consider a user's location and motivation, resulting in responses tailored to individual users. The extent to which intentions can be parameterized within these queries remains an active area of exploration.

Evolutionary pressures are driving the transformation of the DNS from a simple, efficient single lookup system into a more flexible computational resource that tailors answers to queries. In this context, query names no longer function merely as static lookup keys in a local database; instead, they can be regarded as microcode instructions, with the server operating as a computational engine. Furthermore, evolutionary pressures within the DNS, encompassing issues of uniqueness, time, performance, and evolving service models, are reshaping the definition of an identifier within a namespace. From a computing perspective, this creates a shift in which identifiers' behavior increasingly aligns with roles, services and clients' identities, rather than simply distinguishing one resource from another.

The first two subsections examine these changes in detail. First, Metadata and Query Qualification will explore increasingly sophisticated metadata usage to inform DNS queries. Second, as a Computational Resource, DNS observes query names as microcode instructions, a fundamental shift in how the DNS is utilized. These strategies, born from evolutionary pressures, carry distinct implications and properties, influencing various aspects of the DNS.

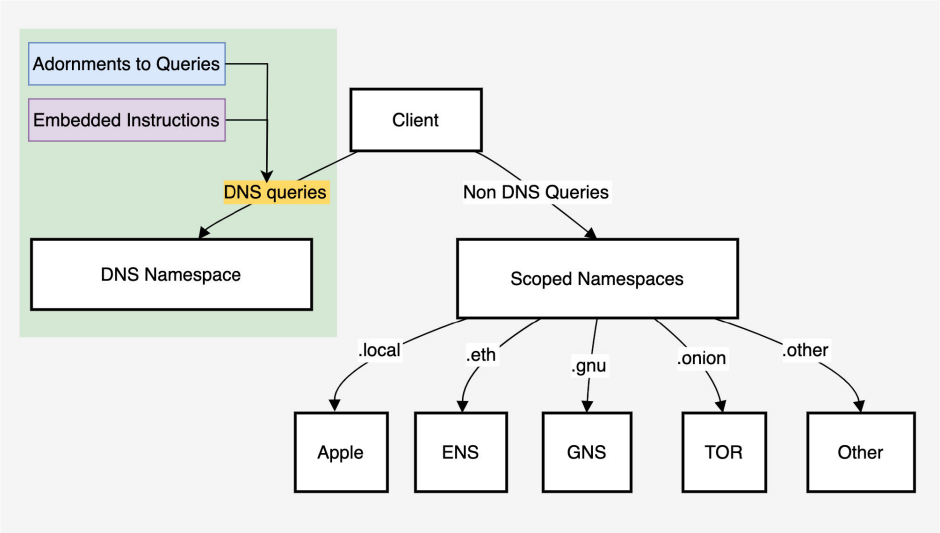


FIGURE 9. Evolutionary pressues and expanding namespaces.

Beyond these transformations within the DNS, the broader Internet landscape has been evolving outside the conventional DNS realm, as per Figure 9. These developments have led to the emergence of alternative identifier systems that can cater to specific needs and scenarios. Following the DNS-centric discussion, the scope expands to explore alternative identifier systems in their respective subsections. These include Apple's usage of the '.local' domain, TOR's '.onion' service, Ethereum's '.eth' approach, and the alternate GNU Name System. Each of these systems offers a unique perspective in addressing the Internet namespace challenges outside the DNS, providing insights into the potential evolutionary directions of digital identifier systems. Please refer to Table 2 for a summary of the subsections below.

A. METADATA AND QUERY QUALIFICATION

In conventional DNS operations, a client's identity typically remains undisclosed. However, with the advent of Extension Mechanisms for DNS (EDNS0) [118] and the inclusion of the Client Subnet (ECS), [23], a client's IP address prefix is exposed and utilized as a part of the retrieval process. A DNS query goes beyond simply requesting a name in an evolved model. It now encompasses the parameters dictated by EDNS0 extensions and the client's subnet (ECS) capability. This process necessitates a form of information trade, where the client provides its assumed identity, encapsulated in its subnet address, to receive an optimally proximate resource based on the server's estimation. Consequently, the previously invariant DNS response is now tailored to these additional query attributes. This progression introduces a distinct tension as the traditionally anonymous DNS unveils more information about the client. However, this transition also brings its own set of advantages. The precision of programmatic DNS responses can be enhanced by exposing the aspects previously concealed by the DNS, as illustrated in Figure 10.

With the advent of DNS over HTTPS (DoH), introducing query adornments could further customize the query by including various HTTP attributes. The objective is not to overload the label space with customization code but to integrate this customization into the query as adornments that can refine or limit the server's responses, ensuring each response is uniquely customized. This shift in the operational paradigm of the DNS suggests a dichotomy. On the one hand, the namespace is viewed as static, providing invariant responses. On the other hand, the DNS query is decorated with metadata that determines the identifier behavior. The server operates as a computational engine, and the DNS response is relative and customized to each query. This transformation underscores the tension between preserving client anonymity and the drive towards enhanced customization, a key challenge in the DNS and a possible evolutionary direction.

B. DNS AS COMPUTATIONAL RESOURCE

A proposal known as Oblivious DNS (DNS) [113] was introduced, although it has not been adopted by DoH or DNS over TLS (DOT) [52]. In the original DNS model, the query name undergoes encryption, yielding a new query name encoded in Base64. This encoded query name is directed not to the upper-level tree but to a specific domain name: **'obliviousdns.net.'**. This process involves a query string encrypted from the initial query. When the query reaches the oDNS server, acting as an authoritative server, it decrypts the encoded query name and retrieves the complete query name without prior knowledge of the query initiator. Essentially, the recursive resolver poses the question, not the original client. The authoritative server then switches its roles and becomes a stub resolver. It fully resolves the requested record, encrypts the result using the same key, sends it back to the recursive resolver, and subsequently delivers the answer to the client.

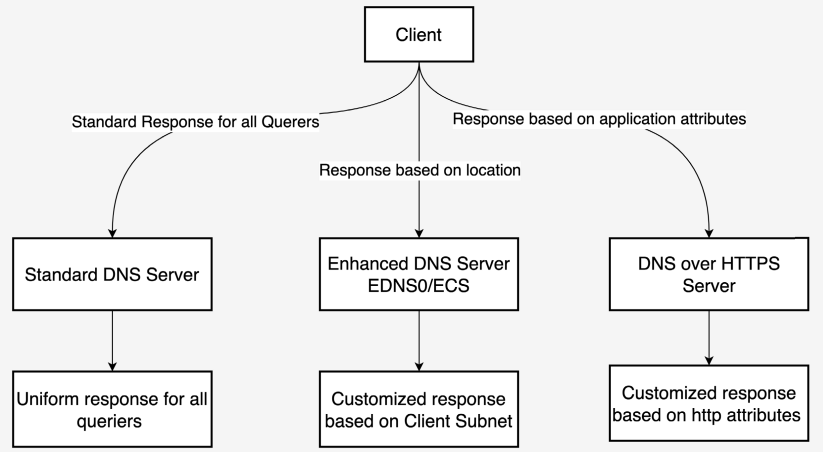


FIGURE 10. Evolution in customized responses.

The primary focus in this example is not the identity-hiding mechanism of DNS but rather the embedded information within the query name. The server assigned to that query name now performs computations on behalf of the client. This situation marks a departure from the traditional DNS lookup. It now involves the query name posing a computational task, which the server executes.

To extend this concept, a DNS server can dynamically construct the response from the context established in the query name. The query name could contain instructions that prompt a specific reaction from the server, effectively transforming the DNS server into a programmatic, systematic entity. This concept could steer the future direction of DNS. This technique allows names to become more elastic, representing a static lookup and a computational task. Consider the DNS as a microcode language, where the query effectively acts as a program for the server. In this scenario, the identifier namespace is not just a dictionary or directory but evolves into a distributed computational platform. From an evolutionary standpoint, shifting the dynamism of the namespace to convert DNS into a computational vehicle is a plausible future direction worth exploring.

C. MULTICAST DNS AND APPLES AND THE '.LOCAL' NAMESPACE

With Zero Configuration Networking (Zeroconf), Apple aimed to make technology more user-friendly by automating the discovery and communication process between devices and eliminating manual configuration. This vision was actualized through Bonjour, Apple's proprietary implementation of Zeroconf [20], [58].

Central to Bonjour's operation is the '.local' pseudo-domain, a namespace intentionally designed to streamline local networking environments. This namespace, primarily employed within smaller networks, is deliberately designed with restricted visibility to foster seamless device-to-device communication. Multicast DNS (mDNS), a protocol adept at providing name resolution services in localized networks where conventional DNS would be inefficient, was utilized to achieve this [18]. The mDNS executes multicast queries with a time-to-live (TTL) value of 1, effectively confining network traffic within its originating subnet [20]. Within this namespace, user-friendly devices such as printers can broadcast their capabilities, enhancing their discoverability by other network clients.

Unlike public DNS domains, which ensure global uniqueness, '.local' domains are designed to be link-local and not globally unique [59]. Hostnames within the '.local' domain are managed cooperatively by mDNS responders present within the local network, preserving the local naming scope and allowing the reuse of the same name in different contexts [7], [20]. In the event of a name conflict within the local network, Bonjour's design ensures automatic reassignment to a new, unique name while maintaining the integrity and consistency of network references.

However, this approach presents the Internet not as a single coherent namespace but as a plethora of local namespaces. These are intentionally limited and inaccessible through standard DNS queries in the public namespace. This unique strategy for creating application-specific namespaces brings forth challenges in distinguishing between the global namespace of DNS and these local domains. The '.local' domain is considered a special-use domain, not intended for use on the public Internet [19]. If this domain were to be delegated into the public space, it would disrupt the intentional constraints that Apple has implemented. While Apple's implementation with the '.local' domain is a noteworthy example, it is important to highlight that this strategy of localized, scoped visibility is part of a broader trend and an evolutionary pressure pushing towards a need for systems that limit the visibility of names to specific, defined sub-ranges rather than making them universally visible, as is the case with traditional DNS.

D. TOR NETWORK AND THE '.ONION' NAMESPACE

The motivation for creating The Onion Router (TOR) network was to establish an Internet environment that safeguards user privacy. The TOR architecture employs

TABLE 2. Application domain of summarized evolutionary responses.

Evolutionary Response	Namespace	Application Domain	Usecase
Metadata Enhanced DNS Queries	DNS	Content Steering, Internet Infrastructure	Optimize content delivery.
DNS as Computation Resource	DNS	Dynamic Content Delivery, Internet Infrastructure	Compute the answer, rather than the location of the answer.
Apple mDNS	.local	Local Network Services	Discovery and communication on local networks.
TOR Network	.onion	Privacy and Secure Communication	Anonymous access to TOR services
Blockchain-based Naming	.eth, .bit, .crypto, etc.	Blockchain, Decentralized Web	Provide readable names to wallets and websites.
GNS Name System	.gnu	Censorship Resistant Name Service	Alternate name system peer-to-peer network for storing and resolving names.

multilayered encryption to anonymize the end user. Each layer of encryption ensures that the user's profile remains confidential and cannot be exposed [31].

The TOR network uses public DNS infrastructure to handle regular DNS queries. However, it employs a technique to protect the identities of authoritative servers responsible for resolving popular domains on the public Internet and the identities of clients accessing the network. A name within the '.onion' namespace is derived from the public key of the asymmetric key pair [31]. When an onion service is set up, it automatically generates a private key and a corresponding public key. The public key is then hashed and truncated. The resulting hash is encoded using Base32, which yields a character string. This string is then suffixed with ''.onion' to create the fully qualified domain name of the onion service, indicating that it is accessible only over the Tor network.

To enhance the recognizability of an onion address, vanity domains provide a means of generating key pairs until the hash of the public key aligns with the desired string, thereby making the domain name reflective of the desired content [31]. It is important to note that the '.onion' domain does not rely on traditional DNS for resolution. Instead, queries for the '.onion' namespace are handled by the TOR protocol, which maps names to their corresponding network addresses within the TOR network.

The TOR was not originally intended to extend or augment the DNS. The primary goal was to respond to the pressures of providing anonymous communications. Similarly to Apple's '.local' namespace method, the '.onion' namespace is a special-use Top-Level Domain (TLD) that serves a distinct purpose [4]. It is deliberately designed to avoid resolution through the traditional DNS infrastructure. This deliberate separation ensures that the hidden TOR services remain concealed within the TOR network and are not exposed to the global public DNS namespace. Consequently, the Special-Use Domain Name Registry maintains a registry for these types of domain names reserved for special purposes, ensuring they are not utilized for general purposes.

E. ETHEREUM NAME SERVICE AND THE '.ETH' NAMESPACE

Enforcement of uniqueness is a key aspect of online identities and transactions. This is typically maintained via centralized systems, where names are recorded on a ledger, ensuring that another cannot claim it once a name is used. However, alternatives exist to this standard model, one of which is Blockchain-based systems. For instance, the Ethereum Name Service (ENS) leverages smart contracts to manage the decentralized registry of '.eth' domain names across the Ethereum network [135].

The ENS manages domains through three smart contracts: the Registry, Registrar, and Resolver. The ENS Registry contains three key pieces of information: the owner of the ENS domain, the Resolver associated with the domain, and the time-to-live TTL for the domain records. The Registrar contract owns and allocates domains and subdomains under pre-defined rules in its smart contract. Finally, the Resolver synthesizes the domain names into associate pointers, such as cryptocurrency addresses [100].

The ENS system ensures the uniqueness of each domain name through an implicit collision mechanism, which is activated when there is an attempt to register a claimed name. This decentralized approach mirrors the functionality of a central entity coordinating the uniqueness of domain names [136]. However, it is important to note that while the ENS operates on a decentralized framework, decisions regarding the registration, renewal and governance of top-level domains (TLDs) such as '.eth.' are centralized.

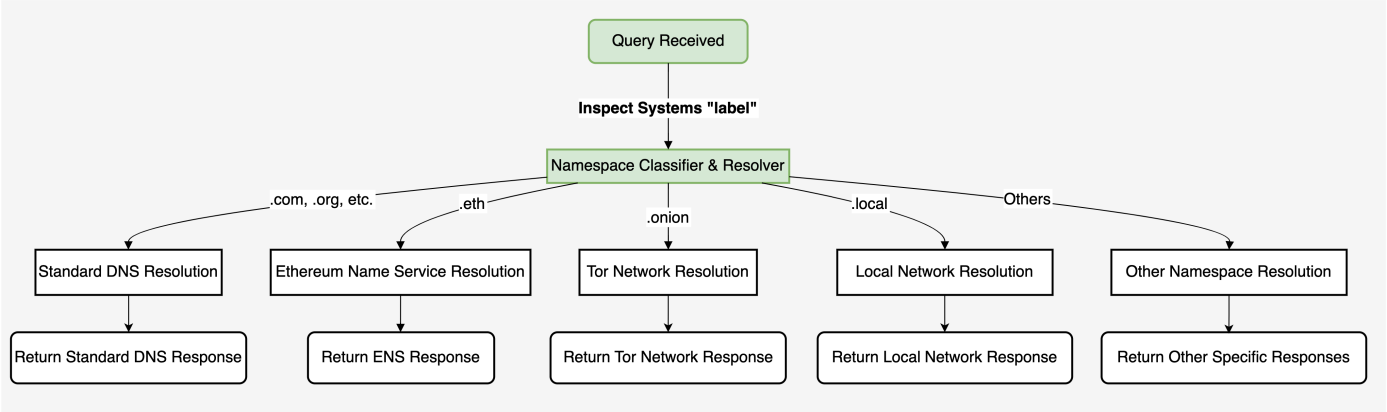


FIGURE 11. Classifier for multi namespace forwarding.

This distinctive architecture presents an intriguing set of evolutionary pressures with advantages and disadvantages. Decentralized systems, such as ENS, offer benefits such as increased user pseudonymity. However, the inherent transparency of these systems can become challenging in public communication settings, distinct from private ones, where the lack of complete anonymity can pose significant issues. Striking a balance between pseudonymity, where users have a persistent but not directly identifiable presence, and full anonymity, analogous to sending mail without a return address, often leads to societal dilemmas. Consequently, a degree of residual accountability, such as a return address for mailed letters, is often seen as a positive attribute in public systems.

F. GNU NAME SYSTEM AND THE '.GNU' NAMESPACE

GNS uses '.gnu,' a pseudo-top-level domain (TLD), to distinguish names resolved in the GNS namespace from those resolved in the DNS namespace [112]. This unique structure leads to a distinctive approach in name resolution, where, unlike hierarchical DNS, every user's zone is treated as an independent root. This results in all zones becoming equal starting points for the resolution process [129]. The resolution process in the GNS is transitive, following delegations from one zone to another based on public keys. These keys, pivotal for navigating different zones, are located using a Distributed Hash Table (DHT), which is a decentralized data structure that efficiently maps keys to values [129].

In contrast, DNS uses an iterative resolution process, which begins at a globally recognized root zone and continues through a sequence of authoritative servers from the TLD to the subdomain to resolve the IP address of the requested resource. Although both systems use delegation, their resolution methodologies are fundamentally different. Like DNS, GNS allows owners complete control over their namespaces. Additionally, users can delegate a part of their namespace, requiring any label within that domain to map to another identity. This feature mirrors DNS, easing the transition to this alternative architecture [129].

The emergence of alternative naming systems outside the traditional DNS has amplified evolutionary tensions toward the interoperability of various Internet namespaces. For instance, the '.gnu' TLD is a distinguishing label that prompts a GNS-aware resolver to transition from the traditional DNS resolution process to the GNS protocol. This mechanism allows names intended for resolution in the GNS namespace to be handled correctly, thus mitigating conflicts with DNS and other name systems.

G. INNOVATION VERSUS INTEGRATION: A DISCUSSION

The use of names within references determines how resources are accessed. A name passed between two entities should resolve to the same digital resource. This is only achievable if names are allocated from a unified namespace and the resolution of the name into network addresses is consistent and coherent. Market competition dynamics for digital presence naturally encourage all participants to be bound within this unified namespace residing in the DNS. Moreover, for the Internet to continue functioning as a public communications system, it is essential to remain within the same namespace to maintain coherence. This study outlines examples of alternate namespaces and notes the caveat that the parallel use of such alternate name systems risks fragmenting the Internet's namespace, eroding the value of a unified and cohesive communication network.

In response to support cohesion across diverse namespaces, one approach involves augmenting name resolvers with classifier functionality for various name suffixes such as '.eth,' '.bit,' '.onion,' '.gnu,' and '.local.'. As described in Figure 11, this method introduces a distinguisher label to specify the namespace that is used. From an architectural point of view, this approach may provide a potential avenue toward interoperability among multiple namespaces, necessitating minimal changes to the existing global DNS infrastructure [54]. However, the challenge lies in determining who decides which distinguisher labels are used by each of these namespaces. This will only be an effective approach with a coherent and universally accepted method for organizing such distinguishing labels.

VI. CONCLUSION

The evolutionary pressures that shaped the alternative namespace have not strayed far from the root of the DNS and remain fundamentally mappable to the DNS; the key challenge lies in preserving cohesiveness.

The foundation of trust in Internet interactions primarily stems from confidence in the naming system, a necessity made more acute by the indeterminism of the routing and addressing system, influenced in the latter case by IPv4 address exhaustion. In this context, the trust placed in Internet transactions relies on the exchange of credentials between the engaging parties. This crucial process enables the receiving party to verify that the responses they receive are from an entity that controls the domain name of the accessed service.

Navigating the evolution of Internet identifiers is intricate, as it is critical to accommodate various aspects of technical innovation in a gradual and incremental manner while simultaneously preserving the coherence of the identifier space. These may appear to be conflicting goals, as they often stand in opposition.

REFERENCES

[1] (Jul. 2017). *A Look at Quic Use*. [Online]. Available: https://www.potaroo.net/ispcol/2022-07/quic.html
[2] (2007). *About URIs*. [Online]. Available: https://www.loc.gov/standards/uri/about.html
[3] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer, "FARSITE: Federated, available, and reliable storage for an incompletely trusted environment," *ACM SIGOPS Operating Syst. Rev.*, vol. 36, no. SI, pp. 1–14, 2002.
[4] J. Appelbaum and A. Muffett, *The 'Onion' Special-Use Domain Name*, document RFC 7686, Oct. 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7686
[5] A. Babakian, P. Monclus, R. Braun, and J. Lipman, "A retrospective on workload identifiers: From data center to cloud-native networks," *IEEE Access*, vol. 10, pp. 105518–105527, 2022.
[6] T. C. Bartee, *ISDN, DECnet, and SNA Communications*. USA: H. W. Sams, 1989, p. 414.
[7] M. Bartosh and R. Faas, *Essential Mac OS X Panther Server Administration: Integrating Mac OS X Server Into Heterogeneous Networks*. USA: O'Reilly Media, 2009.
[8] T. Berners-Lee, *Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as Used in the World-Wide Web*, document RFC 1630, Internet Requests for Comments, Jun. 1994.
[9] T. Berners-Lee. (Aug. 1996). *The World Wide Web: Past, Present and Future*. [Online]. Available: https://www.w3.org/People/Berners-Lee/1996/ppf.html
[10] T. Berners-Lee, R. T. Fielding, and L. Masinter, *Uniform Resource Identifier (URI): Generic Syntax*, document STD 66, RFC3986, Internet Requests for Comments, RFC Editor, Jan. 2005. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3986.txt. http://www.rfc-editor.org/rfc/rfc3986.txt
[11] T. Berners-Lee, L. Masinter, and M. McCahill, *Uniform Resource Locators (URL)*, document RFC 1738, Internet Requests for Comments, RFC Editor, Dec. 1994. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1738.txt
[12] M. Bishop, *HTTP/3*, document RFC 9114, Jun. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9114
[13] M. Bowman, S. K. Debray, and L. L. Peterson, "Reasoning about naming systems," *ACM Trans. Program. Lang. Syst.*, vol. 15, no. 5, pp. 795–825, 1993.
[14] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, and S. Shenker, "ROFL: Routing on flat labels," in *Proc. Conf. Appl., Technol., Architectures, Protocols Comput. Commun.*, 2006, pp. 363–374.
[15] M. Calder, A. Flavel, E. Katz-Bassett, R. Mahajan, and J. Padhye, "Analyzing the performance of an anycast CDN," in *Proc. Internet Meas. Conf.*, 2015, pp. 531–537.
[16] B. E. Carpenter and B. E. Carpenter. (Jun. 2007). *General Identifier-Locator Mapping Considerations*. Internet Eng. Task Force, Internet-Draft. [Online]. Available: https://datatracker.ietf.org/doc/draft-carpenter-idloc-map-cons/01/
[17] L. M. Chan, S. S. Intner, and J. Weihs, *Guide To the Library of Congress Classification*. USA: ABC-CLIO, 2016.
[18] S. Cheshire and M. Krochmal, *Multicast DNS*, document RFC 6762, Feb. 2013. [Online]. Available: https://www.rfc-editor.org/info/rfc6762
[19] S. Cheshire and M. Krochmal, *Special-Use Domain Names*, document RFC 6761, Feb. 2013. [Online]. Available: https://www.rfc-editor.org/info/rfc6761
[20] S. Cheshire and D. Steinberg, *Zero Configuration Networking: The Definitive Guide*. U.K.: O'Reilly Media, 2006.
[21] D. E. Comer and L. L. Peterson, "Issues in using DARPA domain names for computer mail," in *Proc. 9th Symp. Data Commun.*, 1985, pp. 158–164.
[22] *The Relationship Between URNs, Handles, and Purls*, Library of Congress, Washington, DC, USA, 1997.
[23] C. Contavalli, W. van der Gaast, D. C. Lawrence, and W. A. Kumari, *Client Subnet in DNS Queries*, document RFC 7871, May 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7871
[24] G. Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems: Concepts and Design*, 2nd ed. Boston, MA, USA: Addison-Wesley, 2001, p. 779.
[25] National Research Council, *Signposts in Cyberspace: The Domain Name System and Internet Navigation*. USA: National Academies Press, 2005.
[26] D. Crocker, *DNS Attrleaf Changes: Fixing Specifications That Use Underscored Node Names*, document RFC 8553, Mar. 2019. [Online]. Available: https://www.rfc-editor.org/info/rfc8553
[27] D. Crocker, *Scoped Interpretation of DNS Resource Records Through 'Underscored' Naming of Attribute Leaves*, document RFC 8552, Mar. 2019. [Online]. Available: https://www.rfc-editor.org/info/rfc8552
[28] D. R. Daniel and M. H. Mealling, *Resolution of Uniform Resource Identifiers Using the Domain Name System*, document RFC 2168, Jun. 1997. [Online]. Available: https://www.rfc-editor.org/info/rfc2168
[29] D. S. E. Deering and B. Hinden, *Internet Protocol, Version 6 (IPv6) Specification*, document RFC 8200, Jul. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8200
[30] L. Deutsch, *Host Names On-Line*, document RFC 606, Internet Requests for Comments, RFC Editor, Dec. 1973.
[31] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proc. 13th USENIX Secur. Symp. (SSYM)*, Aug. 2004, pp. 303–320.
[32] *Domain Names—Concepts and Facilities*, document RFC 1034, Nov. 1987. [Online]. Available: https://www.rfc-editor.org/info/rfc1034
[33] *Domain Names—Implementation and Specification*, document RFC 1035, Nov. 1987. [Online]. Available: https://www.rfc-editor.org/info/rfc1035
[34] A. Durand, "Digital object architecture and the handle system," ICANN| Office CTO, Los Angeles, CA, USA, 2019. Accessed: Dec. 7, 2023. [Online]. Available: https://www.icann.org/en/system/files/files/octo-002-14oct19-en.pdf
[35] R. Elz and R. Bush, *Clarifications to the DNS Specification*, document RFC 2181, Jul. 1997. [Online]. Available: https://www.rfc-editor.org/info/rfc2181
[36] M. P. Evans and S. Furnell, "The resource locator service: Fixing a flaw in the web," *Comput. Netw.*, vol. 37, nos. 3–4, pp. 307–330, 2001.
[37] P. Fältström and G. Huston. (Dec. 2004). *A Survey of Internet Identities*. Internet Engineering Task Force, Internet-Draft. [Online]. Available: https://datatracker.ietf.org/doc/draft-iab-identities/02/
[38] D. Farinacci, V. Fuller, D. Meyer, D. Lewis, and A. Cabellos-Aparicio, *The Locator/ID Separation Protocol (LISP)*, document RFC 9300, Oct. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9300
[39] S. Farrell and H. Tschofenig, *Pervasive Monitoring is an Attack*, document RFC 7258, May 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7258
[40] R. T. Fielding, J. Gettys, J. C. Mogul, H. F. Nielsen, L. Masinter, P. J. Leach, and T. Berners-Lee, *Hypertext Transfer Protocol—HTTP/1.1*, document RFC 2616, Internet Requests for Comments, RFC Editor, Jun. 1999. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2616.txt

[41] P. Francis, *Pip Near-Term Architecture*, document RFC 1621, May 1994. [Online]. Available: https://www.rfc-editor.org/info/rfc1621

[42] S. Friedl, A. Popov, A. Langley, and S. Emile, *Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension*, document RFC 7301, Jul. 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7301

[43] D. Goldschlag, M. Reed, and P. Syverson, "Onion routing," *Commun. ACM*, vol. 42, no. 2, pp. 39–41, 1999.

[44] S. Griffiths, "ISBN: A history," *Inf. Standards Quart.*, vol. 27, nos. 2–3, pp. 46–48, 2015.

[45] A. Gulbrandsen and D. L. Esibov, *A DNS RR for Specifying the Location of Services (DNS SRV)*, document RFC 2782, Feb. 2000. [Online]. Available: https://www.rfc-editor.org/info/rfc2782

[46] J. Hakala, *Using National Bibliography Numbers As Uniform Resource Names*, document RFC 8458, Internet Requests for Comments, RFC Editor, Oct. 2018.

[47] J. Hakala and H. Walravens, *Using International Standard Book Numbers As Uniform Resource Names*, document RFC 3187, Internet Requests for Comments, RFC Editor, Oct. 2001.

[48] J. Hakala, "Persistent identifiers—An overview," Nat. Library Finland, Helsinki, Finland, 2010.

[49] K. Harrenstien, M. Stahl, and E. Feinler, *DoD Internet Host Table Specification*, document RFC 952, Oct. 1985.

[50] M. Hauben, *History of Arpanet*, vol. 17. Porto, Portugal: L'Instituto Superior de Engenharia do Porto, 2007, pp. 1–20.

[51] T. Howes, M. Smith, and G. S. Good, *Understanding and Deploying LDAP Directory Services*. Reading, MA, USA: Addison-Wesley, 2003.

[52] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. E. Hoffman, *Specification for DNS Over Transport Layer Security (TLS)*, document RFC 7858, May 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7858

[53] G. Huston, "DNS evolution: Innovation or fragmentation?" 2022. Accessed: Nov. 27, 2023. [Online]. Available: https://circleid.com/posts/20221001-dns-evolutioninnovation- or-fragmentation

[54] G. Huston, P. Koch, A. Durand, and W. A. Kumari. (Sep. 2016). *Problem Statement for the Reservation of Special-Use Domain Names Using RFC6761*. Internet Engineering Task Force, Internet-Draft. [Online]. Available: https://datatracker.ietf.org/doc/draft-adpkja-dnsop-special-names-problem/06/

[55] A. Hutton, J. Uberti, and M. Thomson, *The ALPN HTTP Header Field*, document RFC 7639, Aug. 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7639

[56] *IAB Technical Comment on the Unique DNS Root*, document RFC 2826, Internet Requests for Comments, RFC Editor, May 2000.

[57] (Dec. 2021). *iCloud Private Relay Overview*. [Online]. Available: https://www.apple.com/privacy/docs/iCloudPrivateRelayOverviewDec2021.PDF

[58] Apple Inc. (2013). *Net Services Programming Guide*. Accessed: Jul. 24, 2023. [Online]. Available: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/NetServices/Introduction.html

[59] Apple Inc. (2013). *Resolving DNS Domain Names*. Accessed: Jul. 24, 2023. [Online]. Available: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/NetServices/Articles/domainnames.html

[60] J. Iyengar and M. Thomson, *QUIC: A UDP-Based Multiplexed and Secure Transport*, document RFC 9000, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9000

[61] J. Iyengar, C. Raiciu, S. Barre, M. J. Handley, and A. Ford, *Architectural Guidelines for Multipath TCP Development*, document RFC 6182, Mar. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6182

[62] G. Janée, "Resource identifier," in *Encyclopedia of Database Systems*. Santa Barbara, CA, USA: Institute for Computational Earth System Science, Univ. of California at Santa Barbara, 2009.

[63] R. Kahn and R. Wilensky, "A framework for distributed digital object services," *Int. J. Digit. Libraries*, vol. 6, no. 2, pp. 115–123, Apr. 2006.

[64] R. Kahn and R. Wilensky, "A framework for distributed digital object services," *Int. J. Digit. Libraries*, vol. 6, no. 2, pp. 115–123, 2006.

[65] P. Karp, *Standardization of Host Mnemonics*, document RFC 226, Internet Requests for Comments, RFC Editor, Sep. 1971.

[66] M. Komu, M. Sethi, and N. Beijar, "A survey of identifier–locator split addressing architectures," *Comput. Sci. Rev.*, vol. 17, pp. 25–42, Aug. 2015.

[67] M. Kudlick, *Host Names On-line*, document RFC 608, Internet Requests for Comments, RFC Editor, Jan. 1974.

[68] R. Kumbhar, *Library Classification Trends in the 21st Century*. Amsterdam, The Netherlands: Elsevier, 2011.

[69] G. Lavender and M. Wahl, "Internet directory services using the lightweight directory access protocol," in *Practical Handbook Internet Comput.*, Sep. 2004, doi: 10.1201/9780203507223.pt4.

[70] E. Lear and R. Droms. (2003). *What's In A Name: Thoughts From the NSRG*. Internet Engineering Task Force, Internet-Draft. [Online]. Available: https://datatracker.ietf.org/doc/draft-irtf-nsrg-report/10/

[71] T. Li, *Recommendation for a Routing Architecture*, document RFC 6115, Feb. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6115

[72] T. Li, D. Farinacci, S. P. Hanks, D. Meyer, and P. S. Traina, *Generic Routing Encapsulation (GRE)*, document RFC 2784, Mar. 2000. [Online]. Available: https://www.rfc-editor.org/info/rfc2784

[73] C. Liu and P. Albitz, *DNS Bind*. USA: O'Reilly Media, 2006.

[74] M. Mealling and R. Denenberg, *Report From the Joint W3C/IETF URI Planning Interest Group: Uniform Resource Identifiers (URIs), URLs, and Uniform Resource Names (URNs): Clarifications and Recommendations*, document RFC 3305, Internet Requests for Comments, RFC Editor, Aug. 2002.

[75] M. H. Mealling and D. R. Daniel, *The Naming Authority Pointer (NAPTR) DNS Resource Record*, document RFC 2915, Sep. 2000. [Online]. Available: https://www.rfc-editor.org/info/rfc2915

[76] N. Meulemans, "A yellow pages service based on X.500," *Comput. Netw. ISDN Syst.*, vol. 28, no. 14, pp. 1939–1946, Nov. 1996.

[77] D. Meyer, L. Zhang, and K. Fall, *Report From the IAB Workshop on Routing and Addressing*, document RFC 4984, Internet Requests for Comments, RFC Editor, Sep. 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4984.txt

[78] D. Mills, *Internet Name Domains*, document RFC 799, Internet Requests for Comments, RFC Editor, Sep. 1981.

[79] P. Mockapetris and K. J. Dunlap, "Development of the domain name system," in *Proc. Symp. Commun. Architectures Protocols*, 1988, pp. 123–133.

[80] R. Moskowitz, P. Jokela, T. Henderson, and P. Nikander, *Host Identity Protocol*, document RFC 5201, Apr. 2008. [Online]. Available: https://www.rfc-editor.org/info/rfc5201

[81] (May 2020). *Multiplexed Application Substrate Over Quic Encryption*. [Online]. Available: https://datatracker.ietf.org/doc/charter-ietf-masque/

[82] D. T. Narten, R. P. Draves, and S. Krishnan, *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*, document RFC 4941, Sep. 2007. [Online]. Available: https://www.rfc-editor.org/info/rfc4941

[83] G. W. Neufeld, "Descriptive names in X.500," in *Proc. Symp. Commun. Architectures Protocols*, Aug. 1989, pp. 64–71.

[84] A. Newton and L. Daigle, *Domain-Based Application Service Location Using SRV RRs and the Dynamic Delegation Discovery Service (DDDS)*, document RFC 3958, Jan. 2005. [Online]. Available: https://www.rfc-editor.org/info/rfc3958

[85] P. Nikander, A. Gurtov, and T. R. Henderson, "Host identity protocol (HIP): Connectivity, mobility, multi-homing, security, and privacy over IPv4 and IPv6 networks," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 2, pp. 186–204, 2nd Quart., 2010.

[86] P. Nikander and R. Moskowitz. (Dec. 2009). *Host Identity Protocol Architecture*. Internet Engineering Task Force, Internet-Draft. [Online]. Available: https://datatracker.ietf.org/doc/draft-moskowitz-rfc4423-bis/00/

[87] E. Nordström, D. Shue, P. Gopalan, R. Kiefer, M. Arye, S. Y. Ko, J. Rexford, and M. J. Freedman, "Serval: An end-host stack for service-centric networking," in *Proc. 9th USENIX Symp. Networked Syst. Des. Implement. (NSDI)*, 2012, pp. 85–98.

[88] N. Paskin, "Digital object identifier (DOIő) system," in *Encyclopedia of Library and Information Sciences*, vol. 3. U.K.: Taylor & Francis, 2010, pp. 1586–1592.

[89] S. Paul, J. Pan, and R. Jain, "A survey of naming systems: Classification and analysis of the current schemes using a new naming reference model," Citeseer, WUSTL Tech. Rep., 2009.

[90] T. Pauly, E. Rosenberg, and D. Schinazi. (Mar. 2023). *QUIC-Aware Proxying Using HTTP*. Internet Engineering Task Force, Internet-Draft. [Online]. Available: https://datatracker.ietf.org/doc/draft-pauly-masque-quic-proxy/06/

[91] G. Peng, "CDN: Content distribution network," 2004, *arXiv:cs/0411069*.

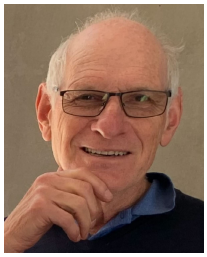
[92] C. Perkins, *IP Mobility Support for IPv4*, document RFC 3344, Internet Requests for Comments, RFC Editor, Aug. 2002. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3344.txt

[93] C. Perkins, D. Johnson, and J. Arkko, *Mobility Support in IPv6*, document RFC 6275, Internet Requests for Comments, RFC Editor, Jul. 2011. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6275.txt

[94] C. E. Perkins, "Mobile IP," *IEEE Commun. Mag.*, vol. 35, no. 5, pp. 84–99, May 1997.
[95] C. E. Perkins, *IP Encapsulation Within IP*, document RFC 2003, Oct. 1996. [Online]. Available: https://www.rfc-editor.org/info/rfc2003
[96] C. E. Perkins, *IP Mobility Support*, document RFC 2002, Oct. 1996. [Online]. Available: https://www.rfc-editor.org/info/rfc2002
[97] R. Perlman, *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols*. Reading, MA, USA: Addison-Wesley, 2000.
[98] D. M. Piscitello and A. L. Chapin, *Open Systems Networking: TCP/IP and OSI*. Reading, MA, USA: Addison-Wesley, 1993.
[99] J. Postel, *Internet Protocol*, document STD 5, Internet Requests for Comments, RFC Editor, Sep. 1981. [Online]. Available: http://www.rfc-editor.org/rfc/rfc791.txt
[100] B. Rajendran, G. Palaniappan, R. Dijesh, and S. D. Sudarsan, "A universal domain name resolution service—Need and challenges–Study on blockchain based naming services," in *Proc. IEEE Region 10 Symp. (TENSYMP)*, Jul. 2022, pp. 1–6.
[101] W. Ramirez, X. Masip-Bruin, M. Yannuzzi, R. Serral-Gracia, A. Martinez, and M. S. Siddiqui, "A survey and taxonomy of ID/Locator split architectures," *Comput. Netw.*, vol. 60, pp. 13–33, Feb. 2014.
[102] *X. 509| ISO/IEC 9594-8*, ITUT Rec Inf. Technology-Open Syst. Interconnection, The Directory: Authentication Framework, 1993.
[103] Y. Rekhter, J. Crowcroft, and B. E. Carpenter, *IPv4 Address Behaviour Today*, document RFC 2101, Feb. 1997. [Online]. Available: https://www.rfc-editor.org/info/rfc2101
[104] E. Rescorla, *HTTP Over TLS*, document RFC 2818, Internet Requests for Comments, RFC Editor, May 2000. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2818.txt
[105] J. K. Reynolds, S. Heker, and C. Weider, *Technical Overview of Directory Services Using the X.500 Protocol*, document RFC 1309, Mar. 1992. [Online]. Available: https://www.rfc-editor.org/info/rfc1309
[106] M. T. Rose, *The Open Book: A Practical Perspective on OSI*. Upper Saddle River, NJ, USA: Prentice-Hall, 1990.
[107] S. Rose and W. Wijngaards, *DNAME Redirection in the DNS*, document RFC 6672, Jun. 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6672
[108] S. Rozenfeld, *Using the ISSN (international Serial Standard Number) As URN (uniform Resource Names) Within an ISSN-URN Namespace*, document RFC 3044, Internet Requests for Comments, RFC Editor, Jan. 2001.
[109] P. Saint-Andre and J. Klensin, *Uniform Resource Names (URNs)*, document RFC 8141, Internet Requests for Comments, RFC Editor, Apr. 2017.
[110] J. H. Saltzer, *On the Naming and Binding of Network Destinations*, document RFC 1498, Aug. 1993. [Online]. Available: https://www.rfc-editor.org/info/rfc1498
[111] P. Sattler, J. Aulbach, J. Zirngibl, and G. Carle, "Towards a tectonic traffic shift: Investigating Apple's new relay network," in *Proc. 22nd ACM Internet Meas. Conf.*, Oct. 2022, pp. 449–457.
[112] M. Schanzenbach, C. Grothoff, and B. Fix, "The GNU name system," RFC Editor, Tech. Rep. RFC 9498, p. 74, Nov. 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9498
[113] P. Schmitt, A. Edmundson, A. Mankin, and N. Feamster, "Oblivious DNS: Practical privacy for DNS queries: Published in PoPETS 2019," in *Proc. Appl. Netw. Res. Workshop*, Jul. 2019, pp. 17–19.
[114] K. Schomp, O. Bhardwaj, E. Kurdoglu, M. Muhaimen, and R. K. Sitaraman, "Akamai DNS: Providing authoritative answers to the World's queries," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Architectures, Protocols Comput. Commun.*, Jul. 2020, pp. 465–478.
[115] B. M. Schwartz, M. Bishop, and E. Nygren. (Mar. 2023). *Service Binding and Parameter Specification via the DNS (DNS SVCB and HTTPS RRs)*. Internet Engineering Task Force, Internet-Draft. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-dnsop-svcb-https/12/
[116] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jul. 1948.
[117] J. Shoch, "Inter-network naming, addressing, and routing," in *Proc. IEEE COMPCON Fall*, 1978, pp. 72–79.
[118] J. da Silva Damas, M. Graff, and P. A. Vixie, *Extension Mechanisms for DNS (EDNS(0))*, document RFC 6891, Apr. 2013. [Online]. Available: https://www.rfc-editor.org/info/rfc6891
[119] D. L. Silver, J. W. Hong, and M. A. Bauer, "X.500 directory schema management," in *Proc. IEEE 10th Int. Conf. Data Eng.*, Feb. 1994, pp. 393–400.
[120] H. A. Simon, "The architecture of complexity," *Proc. Amer. Philos. Soc.*, vol. 106, no. 6, pp. 467–482, 1962.
[121] A. C. Snoeren and H. Balakrishnan, "An end-to-end approach to host mobility," in *Proc. 6th Annu. Int. Conf. Mobile Comput. Netw.*, Aug. 2000, pp. 155–166.
[122] A. C. Snoeren, H. Balakrishnan, and M. F. Kaashoek, "Reconsidering Internet mobility," in *Proc. 8th Workshop Hot Topics Operating Syst.*, 2001, pp. 41–46.
[123] K. Sollins, *Architectural Principles of Uniform Resource Name Resolution*, document RFC 2276, Internet Requests for Comments, RFC Editor, Jan. 1998.
[124] S. Sun, L. Lannom, and B. Boesch, *Handle System Overview*, document RFC 3650, Internet Requests for Comments, RFC Editor, Nov. 2003.
[125] S. Sun, S. Reilly, and L. Lannom, *Handle System Namespace and Service Definition*, document RFC 3651, Internet Requests for Comments, RFC Editor, Nov. 2003.
[126] (2023). *Uniform Resource Names (URN) Namespaces*. [Online]. Available: https://www.iana.org/assignments/urn-namespaces/urn-namespaces.xhtml
[127] A. Venkataramani, J. F. Kurose, D. Raychaudhuri, K. Nagaraja, M. Mao, and S. Banerjee, "MobilityFirst: A mobility-centric and trustworthy Internet architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 74–80, 2014.
[128] P. Vixie, "What DNS is not," *Commun. ACM*, vol. 52, no. 12, pp. 43–47, Nov. 2009.
[129] M. Wachs, M. Schanzenbach, and C. Grothoff, "A censorship-resistant, privacy-enhancing and fully decentralized name system," in *Proc. 13th Int. Conf. Cryptol. Netw. Security (CANS)*, col. 13. Crete, Greece: Springer, Oct. 2014, pp. 127–142.
[130] R. Wakikawa, Z. Zhu, and L. Zhang, *A Survey of Mobility Support in the Internet*, document RFC 6301, Jul. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6301
[131] B. Walker, G. Popek, R. English, C. Kline, and G. Thiel, "The LOCUS distributed operating system," *ACM SIGOPS Operating Syst. Rev.*, vol. 17, no. 5, pp. 49–70, Dec. 1983.
[132] C. Weider and R. Wright, "A survey of advanced usages of X.500," RFC Editor, Request Comments, Tech. Rep. RFC 1491, p. 18, Jul. 1993. [Online]. Available: https://www.rfc-editor.org/info/rfc1491
[133] *Naming and Name Management Systems: A Survey of the State of the Art*, Dept. Comput. Sci., Univ. Western Ontario, London, U.K., 1989.
[134] W. A. Wiegand, "The 'Amherst method': The origins of the Dewey decimal classification scheme," *Libraries Culture*, vol. 33, no. 2, pp. 175–194, 1998.
[135] P. Xia, H. Wang, Z. Yu, X. Liu, X. Luo, G. Xu, and G. Tyson, "Challenges in decentralized name management: The case of ENS," in *Proc. 22nd ACM Internet Meas. Conf.*, Oct. 2022, pp. 65–82.
[136] P. Xia, H. Wang, Z. Yu, X. Liu, X. Luo, and G. Xu, "Ethereum name service: The good, the bad, and the ugly," 2021, *arXiv:2104.05185*.
[137] L. Zhang, A. J. Mankin, J. W. Stewart III, D. T. Narten, and D. M. Crawford. (Oct. 1999). *Separating Identifiers and Locators in Addresses: An Analysis of the GSE Proposal for IPv6*. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-ipngwg-esd-analysis/05/
[138] J. Zirngibl, P. Sattler, and G. Carle, "A first look at SVCB and HTTPS DNS resource records in the wild," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops (EuroS&PW)*, 2023, pp. 470–474.

ANDREW BABAKIAN is currently pursuing the Ph.D. degree in engineering with the University of Technology Sydney, Australia. He has held numerous technical leadership positions driving innovation efforts in software-defined networking and identity technologies for cloud-native architectures. His current research interests include naming systems and the evolution of Internet identifiers.

GEOFF HUSTON received the A.M., B.SC., and M.Sc. degrees. He was an Internet Researcher and an ISP Systems Architect with National Telco and a network operator at various times. He is currently the Chief Scientist with Asia-Pacific Network Information Centre (APNIC), the Regional Internet Registry that serves the Asia–Pacific Region. He has been closely involved with the technical evolution of the internet for many years, particularly within Australia, where he was responsible for building the internet within Australian Academic and Research Sector, in 1990. He is the author of several internet-related books. He was a member of the Internet Architecture Board, from 1999 to 2005. He served on the Board of Trustees of the Internet Society, from 1992 to 2001, and chairing several IETF Working Groups.



ROBIN BRAUN (Life Senior Member, IEEE) received the B.Sc. degree (Hons.) from Brighton University, Brighton, U.K., in 1980, and the M.Sc. (Eng.) and Ph.D. degrees from the University of Cape Town, Cape Town, South Africa, in 1982 and 1986, respectively. In 1986, he started his academic career at the University of Cape Town. In 1998, he moved to the University of Technology Sydney, Australia, where he occupied the Chair of Telecommunications Engineering. Prior to moving to academia, he spent ten years in industry, mostly with Philips and Plessey, where he worked on the design of precision electronic distance measuring equipment. He is currently an Emeritus Professor with the School of Electrical and Data Engineering. His recent work has been in network protocols and the management of complex next-generation networks. He is very active in software-defined networks. He is a Founder Member of Australia and New Zealand Software Defined Networking (ANZSDN).

JUSTIN LIPMAN (Senior Member, IEEE) received the Ph.D. degree in telecommunications engineering from the University of Wollongong, Australia, in 2004. He is currently an Industry Associate Professor with the University of Technology Sydney (UTS) and a Visiting Associate Professor with the Graduate School of Engineering, Hokkaido University. He is also the Director of Research Translation with the Faculty of Engineering and IT and the Director of the RF Communications Technologies (RFCT) Laboratory, where he leads industry engagement in RF technologies, cybersecurity, privacy-preserving technologies, the Internet of Things, and tactile internet. He serves as a Committee Member in Standards Australia contributing to the International IoT Standards and Digital Twins. Previously, he was the Deputy Chief Scientist of the Food Agility Cooperative Research Center. Prior to joining UTS, over a 12 year period, he held several senior management and technical leadership roles at Intel and Alcatel, driving research and innovation, product development, architecture, and IP generation. His research interests include enabling ''things'' to be adaptive, connected, distributed, ubiquitous, and secure.

Appendix D

Cloud-Native Service Mesh Readiness for 5G and Beyond

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 15 November 2023, accepted 20 November 2023, date of publication 23 November 2023,
date of current version 29 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3335994

RESEARCH ARTICLE

Cloud-Native Service Mesh Readiness for 5G and Beyond

SAIDULU ALDAS[1] **AND ANDREW BABAKIAN**[2]
[1]Intel Corporation, Santa Clara, CA 95054, USA
[2]VMware Inc., Palo Alto, CA 94304, USA

Corresponding author: Andrew Babakian (ababakian@vmware.com)

ABSTRACT The 3GPP has defined the 5G network infrastructure differently from earlier generations, utilizing a Service-Based Architecture (SBA) to accommodate scalability and ensure cost-effectiveness in a cloud and microservices-friendly design. Each 5G Network Function (NF) operates as a service, seamlessly scaling to support an on-demand load and latency requirements. However, this introduces significant complexity to the infrastructure due to an influx of messages across 5G services, making request handling and resiliency challenging. While the 3GPP has provided guidelines and recommendations to address this complexity, it is left to the vendors to develop a working model and design choices within the cloud-native framework. This paper explores the intricate challenges that 5G encounters within the current cloud-native deployment landscapes, with a focus on service-to-service communication, service discovery, and network programmability. It highlights the essential role of Service Mesh in fulfilling 3GPP's vision, proposing enhancements tailored for 5G deployment scenarios. In addition, this work provides architectural guidance for enhancing Service Mesh frameworks to support 5G Core Signalling, extending the Envoy proxy to meet 5G's unique semantic requirements. Finally, this paper sheds light on the future requirements for developing optimal Service Mesh infrastructure that minimizes performance implications while maintaining seamless scalability for cloud-native environments.

INDEX TERMS Cloud-native, 3GPP, 5G core, network functions virtualization, service communication proxy, service mesh, service discovery, identifier system.

I. INTRODUCTION

5G architecture represents a significant step forward in network technology. It intelligently harnesses the vast capacity of the cloud, the dynamic control of Software Defined Networking (SDN), the strategic resource management of Network Function Virtualization (NFV), and the modular strategy intrinsic to microservices architecture [4]. Central to this modular strategy is the concept of a microservice: a self-contained service that fulfills a specific business function that can be independently developed, maintained, and scaled [5]. The 5G architecture can offer scalable services and flexible deployment models by adopting this microservices-based approach combined with the aforementioned technologies.

The associate editor coordinating the review of this manuscript and approving it for publication was Tiago Cruz.

Irrespective of the innovative design, the 5G Core Network Functions (NFs) faced unique challenges in the initial stages of 5G Development, specifically in reliably and efficiently managing service requests. Direct communication between NFs led to significant communication overhead and increased latencies. Furthermore, the system grappled with a lack of load-balancing capabilities, no unified approach to observability, and a complex operational landscape, amplifying the difficulties in achieving seamless network connectivity. The Service Communication Proxy (SCP), introduced as part of 3GPP's release 16 [6], [7], was created as a key component of the 5G Service Based Architecture (SBA) to address these challenges introduced by microservice architectures to provide reliable message delivery through efficient routing, load balancing, and traffic monitoring between NFs.

As defined by the 3GPP standards, the functionality of the SCP primarily provides a guideline and outlines the

© 2023 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

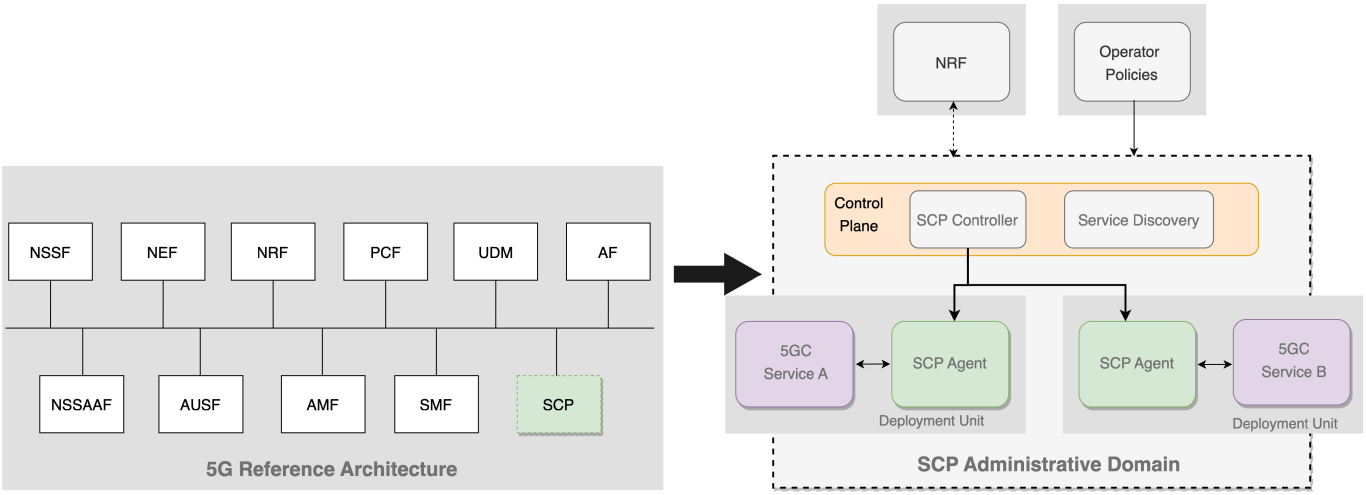


FIGURE 1. 5G control plane reference architecture and the inclusion of SCP.

protocol definition rather than specifying an implementation for Service Mesh deployments. Instead, vendors are tasked to develop and implement following the 3GPP specification. Service Mesh, a promising technology, offers an avenue to subsume the SCP functionality outlined by 3GPP. By definition, a Service Mesh is a dedicated infrastructure layer for handling service-to-service communication [14]. It is responsible for the reliable delivery of requests through the complex topology of services that comprise a modern, cloud-native application. The deployment model of a Service Mesh is implemented as a collection of proxies deployed alongside an application to handle failure detection and streamline service-to-service communication. Furthermore, deploying 5G services within a Service Mesh enacting as SCP presents unique design considerations that necessitate meticulous evaluation and standardization. These include considerations around 5G signaling routing, service discovery, network programmability, traffic management, observability, and load balancing.

The paper aims to provide architectural guidance on enhancing Service Mesh environments for the efficient execution of 5G Core Signaling functions. This research acknowledges the need for advanced capabilities in service mesh architectures. It outlines the essential extensions within the Service Mesh proxy and control path, particularly in the Envoy, to integrate 5G semantics effectively. By exploring the adaptability of the Envoy proxy for 5G use cases, the study proposes solution scenarios designed to embed 5G routing requirements into the service data path, effectively bridging the gap between traditional Service Mesh functions and the specific demands of 5G signaling. Furthermore, this paper presents a comprehensive outlook on deployment strategies within a Service Mesh infrastructure, offering substantial improvements, facilitating a discussion on various design scenarios, and evaluating the considerations pertinent to each.

Finally, this paper explores the future work integral to the Service Mesh to improve the latency of service communication through Service Discovery and 5G custom message handling. This investigation contributes a fresh perspective to the ongoing discourse on optimizing 5G deployment within Service Mesh infrastructures, thereby setting the stage for future exploration in service-to-service communication in complex 5G architectures.

II. CONSIDERATIONS AND REQUIREMENTS

In a scaled 5GC environment, the challenges with direct communication between NFs include communication overhead and increased latencies. Mitigating this challenge in the 5G space requires delineating the responsibilities between the NF and SCP. This approach entails the NF focusing on the core network functions of the service while the SCP, considered an external service, manages the processing tasks previously handled by the NF. These tasks encompass encryption, load balancing, and circuit breaking, all of which must be carefully considered.

In order to have the Service Mesh assume the role of SCP, modifications are required in the framework to support 5G Core Signaling use cases. Service Mesh deployments primarily cater to traditional enterprise applications, not 3GPP deployments. This discrepancy highlights a gap in meeting specific needs, thus creating additional design considerations for Service Mesh. Attaining optimal scalability and seamless end-to-end performance necessitates specific requirements customized to 5G deployments within the Service Mesh infrastructure. The following section delineates these vital considerations:

1) Optimized Low-Latency Communication: Given the necessity for efficient communication between NFs in 5G deployments, designing and tailoring Service Mesh models explicitly for low-latency operations is essential.
2) Scalability with Performance Maintenance: The Service Mesh model should support the scalability of NFs without negatively impacting performance or increasing latency. This requirement is critical given the inherent communication complexities among 5G NFs.

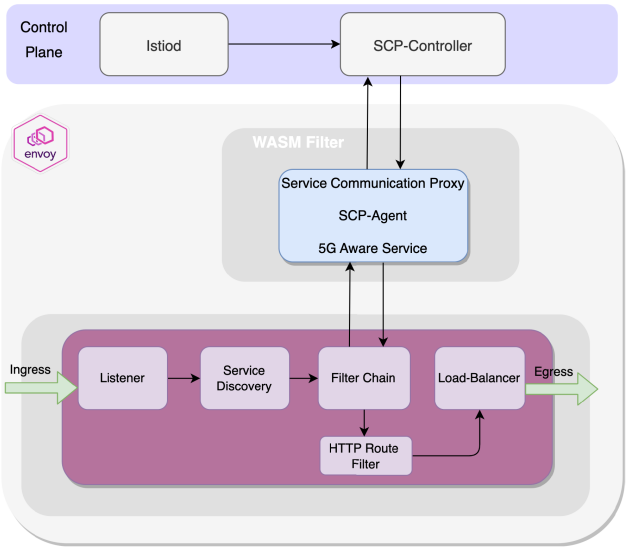


FIGURE 2. Enhanced envoy architecture.

3) Efficient Service Discovery: The Service Mesh infrastructure should offer scalable, low-latency service discovery, tailored to the specific needs of 5G NFs, enhancing deployment scalability and reducing end-to-end communication latency.
4) Comprehensive and Unified Observability: The Service Mesh infrastructure should provide 5G-specific and comprehensive infrastructure-wide observability for monitoring and troubleshooting across the system.

Addressing the specified requirements and achieving the targeted Key Performance Indicators (KPIs) requires careful consideration and standardization of various design choices. The following sections explore the high-level solution framework as well as examining these design considerations. Central to this exploration is the enhancement of the Service Mesh architecture, incorporating SCP adaptations to address 5G-specific semantics effectively. Concurrently, these enhancements blend the existing functionalities integral to the Service Mesh. This includes maintaining optimized routing, effective load-balancing, and mutual TLS encryption, thereby ensuring that the core advantages of the Service Mesh are fully leveraged in the 5G context.

III. HIGH-LEVEL SOLUTION

In 3GPP Release-15, the advent of 5GC introduced a Service-Based Architecture (SBA) [1], [9]. In this architecture, NFs interact with the Network Repository Function (NRF), a central repository for all 5G core NFs. This enables all 5G Network NFs to discover other NFs and the services and capabilities each exposes. Subsequent interactions between NFs are carried out directly using the HTTPS/HTTP2 protocol [19] via the Service-Based Interface (SBI).

Due to 5GC's microservices-based design, the communication volume between NFs is anticipated to increase significantly compared to previous generations. Combined with millions of concurrent signaling messages, this architectural model is not conducive to large-scale real-time deployments, as the probability of partial failure grows exponentially with system size [2].

As described in the 3GPP Release-16, the SCP was introduced to address the challenges of extensive signaling traffic among the NFs. Designed to support large deployments, the SCP efficiently manages millions of concurrent transactions, prioritizes signaling traffic, and ensures balanced load distribution across NF instances. Building on this foundation, the SCP architecture, as detailed in 3GPP's Release-16, is composed of two primary components: the SCP Controller and the SCP Agent. Firstly, the SCP Controller is the central control plane, responsible for managing and coordinating the SCP Agents. It also synchronizes with the Network Repository Function (NRF) to obtain the current status of the network functions. The SCP Controller and SCP Agent communicate via an internal channel. The controller manages and distributes configurations to the agents and monitors their health through regular intervals, which involves extracting performance data and monitoring heartbeats from the SCP Agent. If the SCP Controller determines that an SCP Agent has failed, it initiates corrective remediation measures.

Functionally, the SCP Agent operates as a data plane, performing the role of HTTP intermediary between service consumers and service providers. The SCP acts as a mediator or proxy to facilitate communication between NFs. The SCP Agent's primary functions include a load balancer, circuit breaker, encryption, and HTTP router, directing traffic between the service consumer and provider. As illustrated in Figure 1, the SCP Agent interposes the communication path, determining the route for the consumer traffic. It selects an appropriate target based on the consumer's request and directs traffic to the targeted SCP Agent representing the provider. The SCP Controller orchestrates this routing and selection policy, utilizing information sourced from the NRF, and subsequently disseminates routing and service selection directives to the SCP Agents.

The following section will outline the implementation of these SCP guidelines within the Service Mesh architecture to cater to 5G requirements.

A. EXTENDING SERVICE MESH: A CONTRIBUTION TO 5G

The Service Mesh is a dedicated infrastructure to facilitate service-to-service communication within microservice architectures central to the open-source project Istio [12], [18]. This framework consists of two primary components: the control and data plane. The Envoy proxy is central to the data plane and is utilized as a reverse proxy in microservice architectures. In Kubernetes environments, Envoy operates predominantly as a sidecar proxy, mediating all network communication between microservices while managing tasks such as load-balancing, circuit-breaking, performing mutual TLS encryption, and unified observability.

Figure 2 provides an overview of Envoy's data plane operations, illustrating the filter chain where each filter

contains a unique role. The listener operates at the socket level, tracking incoming connections. Attached to the listener are the network filters responsible for routing decisions. These filters consist of rules that map virtual hosts to clusters. The metadata within the request, such as the HTTP Host header, is essential to serve as a lookup key in the routing table and a namespace that distinguishes between distinct target instances. The entries in this table determine the destination service to which the request will be forwarded. The load balancer and associated clusters use specific policies to dictate the load-balancing algorithm and select hosts for inclusion in the load-balancing pool. These policies also include retries, timeouts, circuit breakers, and other mechanisms for outlier detection related to service-level objectives.

In Service Mesh implementations, Istio was chosen for this solution for two primary reasons. Firstly, Istio's transition from its traditional role to embrace the new position of SCP is majorly attributed to the extensibility of its data plane, particularly the inclusion of WebAssembly (WASM) [11]. WASM is a general-purpose virtual machine instruction set architecture that provides a bytecode that can be executed on native machine architectures. One of the engines that can interpret and run this bytecode is V8, a high-performance JavaScript and WebAssembly engine written in C++ [11], [15]. Envoy fundamentally embeds a subset of the V8 engine, traditionally used in Chrome and Node.js. This extensibility enables developers to write filters as separate modules in their preferred languages. Once written, Envoy can load these filters and are executed at runtime, which influences the proxy's behavior. For instance, this WASM integration could facilitate the inspection and manipulation of traffic as it flows through the proxy [13]. This approach negates the need for developers to infuse this functionality directly into microservices or be dependent on specific libraries. Instead, the feature is introduced as an inherent part of the sidecar, marking a distinct advantage of embedding SCP functionality into the Service Mesh.

1) INTEGRATION OF SCP WITH ENVOY

From the extensibility of Envoy's integration with WASM, two contributions to the Service Mesh control path architecture have emerged: The inclusion of the SCP Agent and the SCP Controller.

The SCP Agent operates within the Envoy's data plane and is implemented as a WASM filter. This SCP Agent offers three primary functionalities:

1) **Synchronization with the SCP Controller**: The SCP Agent consistently syncs with the SCP Controller. This ensures that policy decisions and routing rules remain current with the latest network dynamics.
2) **Handling of NF Instances Discovery Requests**: When discovery requests are received, the SCP Agent communicates with the SCP Controller, which interacts with the NRF service. The NRF's response data is relayed to the SCP Agent via the SCP Controller. This response typically encompasses Uniform Resource Identifier (URI) [3], application protocol, and service parameters.
3) **Integration with Envoy for Routing and Load Balancing**: Paired with Envoy's existing capabilities, the solution actively monitors downstream HTTP/2 messages. It then leverages 5G header data to determine the routing for requests, ensuring they reach the intended target provider. The pseudocode below illustrates the SCP Agent's request handling function for a detailed understanding of the overall process.

SCP Agent: Request Handling

```
for each incoming HTTP request to SCP Agent do
    Identify target NF, e.g., AMF, SMF, etc.
    if request is for NF discovery and origin is free5gc then
        Redirect request to SCP agent for handling
    else if request targets a specific NF (other than NRF) and origin is free5gc then
        Modify and append 3GPP protocol headers and forward request to the designated target NF
    end if
end for
```

The SCP Controller is an integral component of the Service Mesh control plane and has been developed as an extension to Istio. Its responsibilities include:

- Registering with the NRF to receive updates about NF details and their operational status.
- Constructing network topologies and formulating dynamic 5G routing policies.
- Regularly updating the data plane with recent policy and routing decisions.
- Discovering available NF Profiles from the NRF and registering its NF Profile.
- Subscribing to notifications from the NRF.
- Conducting health checks and, upon startup or receipt of notifications from the NRF, disseminating available NF Profiles to the SCP Agent.

2) MULTI-CLOUD SUPPORT

In the context of multi-cluster and multi-cloud deployments, the imperative to broaden administrative domains across heterogeneous cloud environments emerges as a critical factor for ensuring secure connectivity among distributed services. VMware's adaptation of Istio addresses this need by enabling the extension of administrative boundaries by implementing a Global Namespace (GNS) [20], [21]. As described in Figure 3, a GNS establishes a unified framework for defining service boundaries spanning multiple cloud environments.

This adaptation to Istio maps Kubernetes namespaces from multiple clusters into a single, unified namespace within the Service Mesh. This abstraction streamlines operations

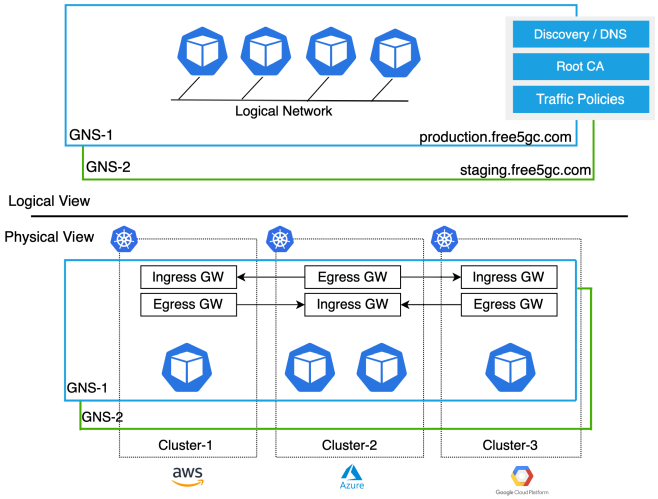


FIGURE 3. Global namespace overview.

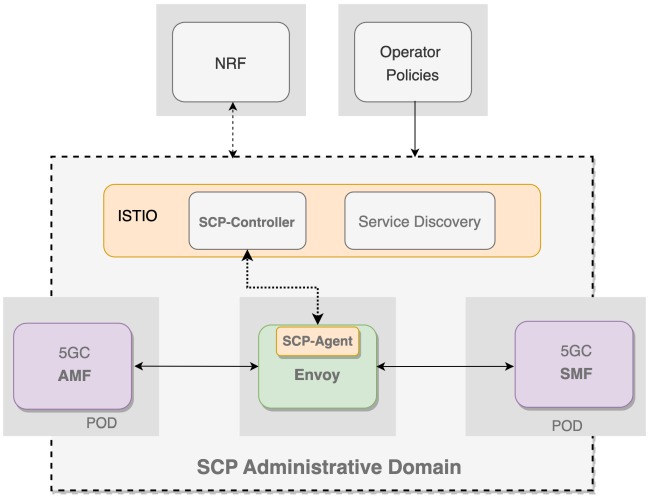


FIGURE 4. Scenario-2: SCP as Standalone.

by eliminating the need to manage individual namespaces per cluster, simplifying cross-cluster interactions and policy enforcement.

Within the GNS, each cloud-specific Kubernetes cluster is rendered interchangeable, providing a flexible approach to bind services at the global level and to include or remove Kubernetes clusters as needed. The key features of this configuration likely pertain to the benefits or capabilities introduced by utilizing the GNS, including:

- Certificate Authority Integration: Within a multi-cloud framework, all workloads subscribing to a GNS inherently share a common trust chain, ensuring consistent identity verification and secure inter-service communication via mutual TLS (mTLS), irrespective of their cloud platform.
- Service Discovery and Domain Name System: Within a GNS, scoped identifiers are essential for service discovery and the Domain Name System (DNS) resolution. Services within the GNS are assigned unique identifiers, which ensure that any service discovery requests or DNS queries are strictly contained within the GNS. This confinement of identifiers to the GNS ensures precise service resolution and maintains operational integrity. Furthermore, scoped identifiers within the GNS play an essential role in maintaining isolation boundaries for services spanning diverse cloud platforms.

These combined efforts position Envoy as a valuable complement to the additional features brought by SCP. Detailed elaborations of these enhancements are provided in the design scenarios discussed in Sections III and V. With an observation on the evolving requirements of 5G, Section VI shifts the focus to future work. It details advanced enhancements for Envoy within the Service Mesh model, including the further extension of Envoy filters and service discovery mechanisms. Additionally, this section explores the application of GNS in control-plane slicing, highlighting key areas for ongoing and future research.

IV. INTEGRATING THE ENHANCED ENVOY IN 5G DEPLOYMENT MODELS

Establishing a robust baseline architecture and deployment model is crucial for achieving the requirements and design goals. This section explores different architectural and deployment models that aim to efficiently and scalably meet these requirements. Different SCP deployment and interworking design choices with Service Mesh are discussed to balance scalability and communication latencies in the overall deployment of 5G services. Three distinct scenarios are considered: first, direct communication without the use of SCP; second, discovery through a standalone SCP; and third, employing SCP as an integrated component within a distributed Envoy proxy setup.

Scenario-1: This implies no involvement of proxy or SCP, and all discovery requests are served directly by the NRF. While this model minimizes proxy-induced latency, it introduces substantial communication overhead and fails to leverage the benefits of Service Mesh infrastructure.

Scenario-2: Within the standalone SCP model, each NF is deployed as a microservice within the Service Mesh environment. In this setup, the Envoy filter encapsulates the SCP capability, as shown in Figure 4. Its main functions include managing outbound traffic for NFs and overseeing the service discovery process. In this setup, the control plane extension of the SCP Controller is tasked with maintaining an up-to-date representation of the core network's state. It provides essential routing and discovery information to the SCP Agent via the eXtensible Discovery Service (XDS) API, a protocol Envoy uses for dynamic service discovery and configuration.

All NFs, including the SCP Agent, are registered with the NRF. Uniquely, the SCP Agent subscribes to notifications about the status alterations of NFs, including events such as registration, updates, and deregistration. This mechanism allows the SCP to oversee all discovery and routing requests. Upon receiving a request from an NF consumer, the SCP's role is to conduct service discovery and direct the request to the appropriate consumer.

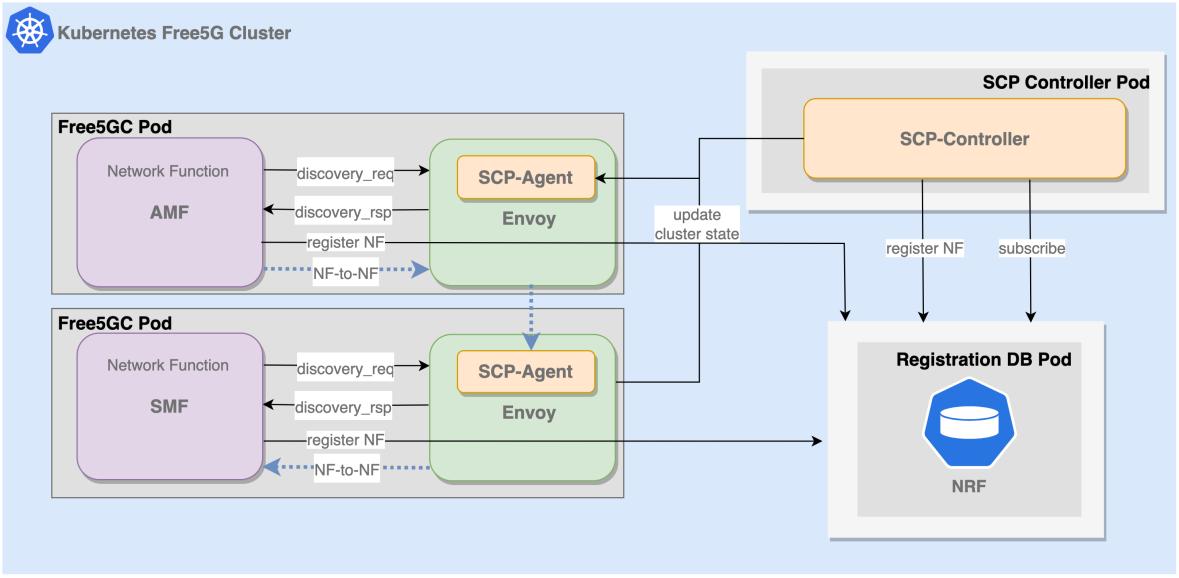


FIGURE 5. Scenario-3: SCP as distributed envoy proxy.

The embedding of the SCP as an Envoy filter within the Service Mesh infrastructure brings many advantages forward. This includes secure inter-service communication and observability without necessitating additional implementation in the SCP or NFs. Realizing the standalone SCP model leverages two pivotal components: a control plane extension that tracks the network state and a data plane or Envoy extension responsible for executing 5G-specific service discovery, routing, and visibility.

Scenario-3: In the SCP as a distributed Envoy proxy model, SCP is deployed as a sidecar within the same pod as the NFs, facilitating in-pod service discovery and routing within the microservice pod. As illustrated in Figure 5, NFs, such as the AMF (Access and Mobility Management Function) and SMF (Session Management Function), along with SCP Controllers, register with the NRF. Additionally, the SCP Controller subscribes to NRF events to keep the state and information about the NFs updated. A source NF, such as the AMF, sends an HTTP signaling message for communication. This message contains a header field named '3gpp-Sbi-Target-apiRoot' with the target URI value set to 'SMF,' indicating the destination NF, which is the SMF. This communication is facilitated by Envoy proxies present on both the source and destination sides. The source SCP Agent, which operates as part of the AMF pod, interprets the incoming request. It then conducts routing discovery and forwards the request to the appropriate SMF instance. Conversely, the SCP Agent within the SMF Pod listens for incoming HTTP messages. Upon receiving a message, it parses the content, verifies the necessary 3GPP message semantics, and then delivers the message to the SMF.

This architecture's core is the SCP WASM filter, as detailed in Section III. It is implemented as an essential Envoy filter, working with the SCP Controller. The SCP Controller is critical in this architecture, as it continually monitors the state of the network and updates the SCP Agent about any changes through the XDS API. By integrating the SCP's functionalities into the Envoy sidecar, the SCP Agent can effectively manage outbound traffic from the NF and efficiently handle the service discovery process. This approach offers a scalable solution that significantly reduces additional latency, thus enhancing communication efficiency.

V. DESIGN CHOICES AND PERFORMANCE CHARACTERIZATION

Each of the design scenarios mentioned earlier presents its own set of advantages and disadvantages:

1) **Scenario-1: Direct Communication Without SCP** - This deployment model is straightforward but lacks the advantages of a Service Mesh. It is characterized by significant communication overhead and limited scalability. In this scenario, each request from the source NF involves a multi-step process for service discovery. Firstly, the source NF sends a request to the NRF to obtain discovery information. Upon receiving the discovery response from the NRF, the source NF sends the communication request to the destination NF. This process introduces multiple additional requests due to the interaction with the NRF, which is hosted on a separate machine in most deployments. This additional layer of communication complicates the process and contributes to the increased overhead and reduced scalability.
2) **Scenario-2: Standalone SCP** - This model benefits from Service Mesh infrastructure; the deployment model is scalable. The SCP is deployed as a separate microservice in the Service Mesh environment with a centralized proxy, i.e., Envoy. The SCP functionality is implemented as a virtual container service that controls outbound traffic and handles discovery requests. In this

scenario, the source NF requests the standalone SCP. The SCP then performs service discovery and forwards the request to the appropriate destination NF instance. Notably, in this configuration, the SCP may be running on a separate machine, which could have implications for the routing and latency of network communications.

3) **Scenario-3: SCP as Distributed Envoy Proxy** - This model also benefits from Service Mesh infrastructure and is scalable. The SCP is deployed as a WASM filter within the Envoy sidecar, sharing a pod with the NF. The SCP handles both discovery requests and communication. In this scenario, the communication request from the source to the destination NF is intercepted and processed by the SCP Agent. The SCP Agent interprets the service discovery information and directs the request to the appropriate destination NF. The key distinction between scenarios two and three lies in the location and mechanism of discovery realization: in Scenario-3, all discovery processes are executed within the pod itself, resulting in minimal latency. In contrast, Scenario-2 involves the request traversing the network to a separate physical machine hosting the SCP agent, potentially requiring one or more network hops, which could significantly increase latencies compared to Scenario-3.

Systematic experiments were conducted to evaluate the deployment models' effectiveness. These aim to measure the discovery and communication latencies across NFs in the three scenarios. The experimental setup involved the following:

- **Workload:**
  Free5GC [8] was deployed to emulate a comprehensive suite of 5G core network functions (NFs). This included the Access and Mobility Management Function (AMF), Session Management Function (SMF), Policy Control Function (PCF), Authentication Server Function (AUSF), Network Slice Selection Function (NSSF), User Plane Function (UPF), Unified Data Management (UDM), Unified Data Repository (UDR), and the supporting MongoDB database. As an open-source project compliant with 3GPP Release 15 and beyond, Free5GC served as an effective solution for the core network in our simulations. It accurately mirrored end-to-end core NF communication scenarios. The choice of Free5GC for the experiments was driven by its comprehensive capabilities as an open-source solution for simulating 5G core workloads, filling the gap in the absence of commercial options.
- **Simulation:** UERANSIM, acting as a RAN and UE simulator, was employed as a load generator to benchmark the latencies of control path intercommunication within the 5G core network. Deployed on one of the servers, UERANSIM was instrumental in simulating network activities such as registration, deregistration, and PDN session establishment. Specifically, the deregistration procedure was initiated using the UERANSIM to generate SCTP messages towards the Access and Mobility Function (AMF), which in turn communicated with other NFs over the Service-Based Interface (SBI) using HTTP/2 protocols, thereby facilitating a comprehensive end-to-end procedural test.
- **Service Mesh infrastructure:** Istio and Envoy were integral in establishing the Service Mesh environment. Istio acted as the control plane, while Envoy served as the data plane, equipped with 5G-specific extensions described in Section III and IV to manage the unique communication requirements of the 5G Core network. The experimental framework utilized a single Kubernetes (k8s) cluster to orchestrate the deployment of network functions, with Docker serving as the container runtime. Each NF was housed in its container, with an Envoy proxy container co-existing within the same pod to enable Service Mesh communication features specifically for Scenario-3.
- **Hardware Configuration:** The tests ran on a hardware setup that included two servers powered by 3rd Gen Intel® Xeon® Scalable Processors, connected through a 100G networking fabric to enable the high-speed data transfer required for 5G operations.

Benchmark data:

The benchmarking process involved triggering the deregistration procedure using UERANSIM, which generated SCTP messages directed towards the AMF. Subsequently, AMF communicated with other NFs, as described in the workload section, over the Service-Based Interface (SBI) using HTTP/2 protocols. This setup facilitated a comprehensive end-to-end procedural test. The focus of latency measurements was between two specific network functions: the UDM and UDR. These NFs were selected because their interactions with the database are anticipated to be more latency-oriented in the overall test scenario, given the nature of data management and repository functions.

TABLE 1. Comparisons of deployment scenarios.

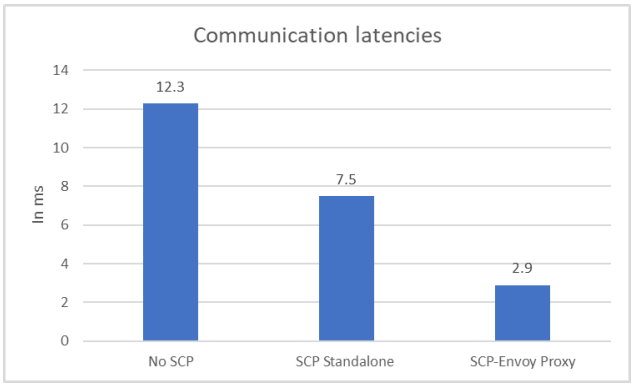


Table 1 illustrates the communication latencies observed across three deployment models: without SCP, with standalone SCP, and with SCP as a distributed Envoy proxy. The analysis reveals that Scenario-2, which employs a

centralized SCP, exhibits higher latencies than Scenario-3, where SCP is embedded within the NF pod, directly facilitating discovery and communication. Both scenarios demonstrate their potential for scalable and low-latency communication within a 5G context.

In Scenario-2, although scalability is emphasized, it necessitates augmented allocation of resources, specifically in terms of memory and computation. However, this scenario fails to attain the reduced latency levels observed in Scenario-3, where the SCP is seamlessly integrated within each NF pod.

Refinements in the experimental approach have led to notable latency improvements. Particularly, the integration of the SCP with the distributed Envoy proxy, as detailed in Scenario-3 of Section IV, has been instrumental in achieving these latency reductions. Embedding the SCP within the NF pod and utilizing Envoy's advanced routing capabilities, Scenario-3 meets the sub-millisecond latency benchmarks crucial for high data rate 5G communications. This performance is consistent with the stringent requirements specified in 3GPP TS 22.261 Section 6.4.2.1 and Annex A [17], reflecting the standard's emphasis on resource-use efficiency for both control and user planes and supporting the targeted latencies for 5G Core Signaling networks.

VI. FUTURE PERSPECTIVES AND NECESSARY DEVELOPMENTS

The 5G cloud-native architecture is undergoing continuous evolution, and integrating an enhanced Envoy proxy brings several key areas into the spotlight for future investigation. Among these, service discovery is integral to achieving efficient routing, effective load balancing, and low latency. The following subsections will focus on exploring avenues for future work. This includes addressing the challenges associated with service discovery, emphasizing the integration of custom 5G message handling, and limiting environments specifically to the control-plane network slice.

A Service Mesh solution maintains records and meta-information about the services and endpoints in a cluster to handle routing, policy management, and load balancing. For instance, Istio utilizes its service registry and discovery system, which populates from the platform discovery system. Furthermore, the Envoy proxy, responsible for routing decisions, communicates with the Istio control plane through the XDS API to acquire discovery and routing details. The Envoy service discovery obtains DNS information on the endpoints and services. However, to support 5G-specific requirements, the discovery and routing mechanisms must be enhanced to accommodate 5G routing and load balancing scenarios adequately. For instance, consider the following scenarios, which offer limited literature examples related to 5G service discovery:

Network Slicing-Specific Discovery: Network slicing enables optimal use of network resources by employing virtualization technologies on shared physical infrastructure [16]. The Envoy SCP architecture must consider the service discovery semantics of singling messages related to specific network slices (NSSAI) to satisfy the spec requirements and achieve optimal routing of messages to the destination/producer NF.

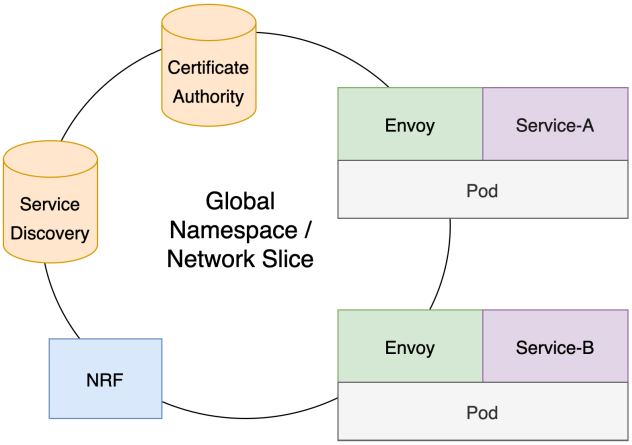


FIGURE 6. Global namespace per network slice.

The Global Namespace (GNS), as illustrated in Figure 6, could enhance the network in governing connectivity in VMware's adaptation of Istio [20]. As described in Section III, Global Namespace (GNS) is a logical abstraction that establishes an application boundary spanning multiple environments and clouds. Within the Service Mesh framework, once Envoy becomes a member of a particular GNS, it is confined to the scope of that GNS, essentially its administrative boundary.

In the examined scenarios, where the control plane is segmented, service discovery query names and certificate authorities are confined within the boundaries of specific Global Namespaces. This arrangement ensures authenticated and secure communication within each network slice. While deploying a dedicated NRF for each slice is feasible, particularly in situations involving distinct tenancy requirements, such as those between independent operators or in distinguishing staging from production environments, a more practical approach is identified. This approach involves the utilization of a common NRF that serves multiple slices. The NRF functions as an area border function in such a framework, intersecting with the GNS boundaries. This intersection enables the NRF to discriminate access based on subscriptions effectively. This method aligns with maintaining efficient and secure network slicing, providing a controlled yet flexible environment within the 5G architecture.

Domain-Specific Discovery Semantics: As part of the discovery process, the Envoy SCP Agent must consider 5G-specific semantics in the discovery mechanisms. This includes rejecting consumer requests with appropriate 3GPP-specific error codes if no NF Service Producer supports the requested features. To address these needs, the conventional service discovery method must shift from providing a universal answer to one tailored for each consumer. This entails customizing responses for reasons such as access control, policy, or to optimize the connection between

the consumer and provider. A strategy involves enhancing queries with meta-information, enabling clients to exchange their details for more accurate service information [2]. The service discovery resolution can then fine-tune its answers, offer improved estimations, or even decline connections, as illustrated.

By enforcing identifiers to behave uniquely based on the querying entity, the resolution system could enable the implementation of access controls that adhere to domain-specific discovery semantics. In this context, service discovery is set to transition from a mere dictionary-based search to a system that enriches queries with added details, allowing personalized responses based on the queriers' identity and metadata.

Low-Latency and Highly Dynamic Service Discovery Mechanism to Support Time-Sensitive Communication: One key challenge in any 5G deployment in the cloud is providing low-latency communication infrastructure to meet 5G KPI requirements [10]. The discovery mechanism must maintain a repository of parameters related to the destination NF reachability latencies and load.

Enhancing the service discovery system to provide more than just IP necessitates a shift from modeling hosts to services. Services demand additional details than just IP, including port and protocol data. This also involves understanding the precise needs and intentions of the service consumer's request. A client and service discovery broker exchanging information could streamline the search results, enhancing performance in establishing connections without additional steps to gather connection information. For instance, in the context of the upcoming 6G, the service discovery could supply connection profiles hinting at protocol support (e.g., HTTP/3 for 6G) alongside port details [22]. This ensures fast connections without latency during service negotiations.

Furthermore, application namespaces could be included as a parameter in the Service Discovery resolution. As NFs consolidate onto a shared platform, name-based routing distinguishes application sessions on a common host. Within this discovery process, the subscriber service may populate the name within the application protocol during the initial connection with the provider service. 3GPP's release 16 also suggests that a name-based method is another viable approach for discovery and routing [6], [7].

Furthermore, Service Discovery systems often direct clients based on configured priorities and weights to enhance connection speeds and address time sensitivity. This method, aimed at evenly distributing traffic, might not always account for the queriers' preference for faster response times. This mismatch can conflict between the server's choice of an 'optimal' service and the client's performance needs. Instead of providing a single answer, the service discovery system could offer multiple candidate answers, allowing the client to choose based on specific needs. This adaptable method aligns closely with consumer interests, enabling a tailored service selection and potential for further optimization. While client-side resolution computation is often less favorable, a possible implementation with Envoy could handle this task, ensuring no added burden on the NF. Envoy could facilitate the logic to identify the best candidate through a series of synthetic transactions on behalf of the consumer to determine the optimal target. This approach represents another promising direction for service discovery evolution.

Custom 5G Message Handling:

In this framework designed for the SCP architecture, provisions have been made for future extensions in custom message handling. A key focus is integrating the desired 3GPP functionalities, leveraging custom headers. This enhancement empowers Envoy to efficiently process 5G signaling messages encapsulated in HTTP headers and apply transformation rules accordingly. Such advanced processing is crucial in guiding informed routing decisions. The framework's advanced capabilities are crucial in making informed choices about data routing. Moreover, they ensure that vital information is reliably sent to the main controlling entity, which helps maintain a consistent state for all the NFs across the system.

For 5G aware load balancing, immediate enhancements warrant consideration. The framework has considered the support of custom messages, notably the **3gpp-Sbi-Oci**. This message is critical for indicating overload control notifications and receiving responses from consumers and providers. The load balancer can harness this data as a component of its algorithm, potentially selecting alternate service candidates for request routing. Additionally, the framework accommodates the **3gpp-Sbi-Message-Priority**, a custom message vital for signaling when a specific NF necessitates traffic prioritization over other NFs. This can be used to provide congestion control. By synergistically leveraging these messages, there is an opportunity to elevate performance and reduce latency within the Service Mesh design.

These concepts improve how we perceive service discovery, routing, and load balancing. As the paper outlines, the Service Mesh is the foundational infrastructure layer, facilitating a programmable data plane. This not only establishes a foundation for a system that's more agile and nuanced but also sets the stage for future explorations. In addition to the contributions mentioned, efforts will persist in improving and refining the platform, ensuring it is primed and 5G-ready. This enhanced system is uniquely positioned to cater to the needs of both providers and consumers in cloud-native 5G environments.

VII. CONCLUSION

In conclusion, the 5G Service-Based Architecture (SBA) has facilitated the cloudification of the 5G network and deployments, but not without its inherent challenges. The existing cloud infrastructure, designed predominantly to run general applications, faces difficulties when hosting specific 5G workloads, such as 5G core NFs. This situation leads to sub-optimal utilization of cloud resources, fragmented deployment models, and intensive resource consumption to fulfill the rigorous SLAs and KPIs.

The enhancement of the Envoy proxy to become 5G-aware offers a tailored solution, augmenting the service communication proxy mediating between NFs, as evidenced by the deployment strategies delineated in this article. The ongoing evolution of the Service Mesh infrastructure uncovers a promising path toward addressing the distinctive requirements of 5G as the industry advances towards 6G. It shows a future marked by more flexible, scalable, and dependable network architectures. In the near term, service discovery support will be essential for discoverability and ensuring the integrity of the communication system, especially when addressing the multifaceted challenges of the identifier system and transactional integrity across administrative domains.

With the potential extensibility of Envoy, there is an opportunity to weave innovations such as AI/ML, network efficiency, and cost reduction that align with the outlines of 6G specifications. These advancements in Envoy could play a pivotal role in shaping the next generation of network architectures. This progression also positions the Service Mesh to serve as a conduit for a seamless transition from 5G to 6G networks.

The hope is that these insights contribute to the ongoing development within the telecommunications industry, offering a perspective that may assist in realizing the significant potential of next-generation networking technologies. This work aspires to set the stage for a future characterized by enhanced connectivity, adaptability, and efficiency.

ACKNOWLEDGMENT

The authors would like to thank Intel and VMware for their partnership in the Go Big Multi-Cloud Networking Initiative. This joint venture has been central to the research and development process, leading to notable advancements in the service mesh project for 5G technologies. This collaboration fostered innovation and shared expertise in shaping the future of telecommunications. They also appreciate the engineers, researchers, and support staff involved in this project.

(Saidulu Aldas and Andrew Babakian contributed equally to this work.)

REFERENCES

[1] 3GPP, "System architecture for the 5G system," 3rd Gener. Partnership Project (3GPP), Sophia Antipolis, France, Tech. Rep. TR 23.501, Dec. 2018. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.06.00_60/ts_123501v160600p.pdf

[2] A. Babakian, P. Monclus, R. Braun, and J. Lipman, "A retrospective on workload identifiers: From data center to cloud-native networks," *IEEE Access*, vol. 10, pp. 105518–105527, 2022.

[3] T. Berners-Lee, R. T. Fielding, and L. M. Masinter, "Uniform resource identifier (URI): Generic syntax," RFC Editor, RFC 3986, Jan. 2005, p. 61. [Online]. Available: https://www.rfc-editor.org/info/rfc3986, doi: 10.17487/RFC3986.

[4] B. Dab, I. Fajjari, M. Rohon, C. Auboin, and A. Diquélou, "Cloud-native service function chaining for 5G based on network service mesh," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–7.

[5] N. Dragoni, I. Lanese, S. T. Larsen, M. Mazzara, R. Mustafin, and L. Safina, "Microservices: How to make your application scale," in *Perspectives of System Informatics: 11th International Andrei P. Ershov Informatics Conference, PSI 2017*, Moscow, Russia, Cham, Switzerland: Springer, Jun. 2018, pp. 95–104.

[6] ETSI, "5G; 5G system; technical realization of service based architecture; stage 3," ETSI, Sophia Antipolis, France, Tech. Rep. 129 500, Apr. 2021. [Online]. Available: https://www.etsi.org/deliver/etsits/129500129599/129500/16.07.0060/ts129500v160700p.pdf

[7] ETSI, "5G; procedures for the 5G system (5Gs)," ETSI, Sophia Antipolis, France, Tech. Rep. 23 502, Jul. 2021. [Online]. Available: https://www.etsi.org/deliver/etsits/123500123599/123502/16.09.0060/ts123502v160900p.pdf

[8] (2019). *Free5GC*. [Online]. Available: https://free5gc.org/

[9] A. Ghosh, A. Maeder, M. Baker, and D. Chandramouli, "5G evolution: A view on 5G cellular technology beyond 3GPP release 15," *IEEE Access*, vol. 7, pp. 127639–127651, 2019.

[10] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Toward 6G networks: Use cases and technologies," *IEEE Commun. Mag.*, vol. 58, no. 3, pp. 55–61, Mar. 2020.

[11] A. Haas, A. Rossberg, D. L. Schuff, B. L. Titzer, M. Holman, D. Gohman, L. Wagner, A. Zakai, and J. Bastien, "Bringing the web up to speed with webassembly," in *Proc. 38th ACM SIGPLAN Conf. Program. Lang. Design Implement.*, 2017, pp. 185–200.

[12] (Aug. 2016). *ISTIO*. [Online]. Available: https://www.istio.io

[13] S. M. Jain and S. M. Jain, "Extending ISTIO with webassembly," in *WebAssembly for Cloud: A Basic Guide for Wasm-Based Cloud Apps*. Berlin, Germany: Springer, 2022, pp. 151–160.

[14] W. Li, Y. Lemieux, J. Gao, Z. Zhao, and Y. Han, "Service mesh: Challenges, state of the art, and future research opportunities," in *Proc. IEEE Int. Conf. Service-Oriented Syst. Eng. (SOSE)*, Apr. 2019, pp. 122–1225.

[15] J. Ménétrey, M. Pasin, P. Felber, and V. Schiavoni, "Webassembly as a common layer for the cloud-edge continuum," in *Proc. 2nd Workshop Flexible Resource Appl. Manage. Edge*, 2022, pp. 3–8.

[16] K. Samdanis, A. Prasad, M. Chen, and K. Hwang, "Enabling 5G verticals and services through network softwarization and slicing," *IEEE Commun. Standards Mag.*, vol. 2, no. 1, pp. 20–21, Mar. 2018.

[17] 3GPP, "Service requirements for the 5G system; stage 1 (release 17)," 3rd Generation Partnership Project (3GPP), Sophia Antipolis, France, Tech. Specification 22.261, Sep. 2022.

[18] O. Sheikh, S. Dikaleh, D. Mistry, D. Pape, and C. Felix, "Modernize digital applications with microservices management using the istio service mesh," in *Proc. 28th Annu. Int. Conf. Comput. Sci. Softw. Eng.*, 2018, pp. 359–360.

[19] M. Thomson and C. Benfield, *HTTP/2*, document RFC 9113, Jun. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9113

[20] VMware. *Vmware Tanzu Service Mesh Documentation—Global Namespaces*. Accessed: Nov. 13, 2023. [Online]. Available: https://docs.vmware.com/en/VMware-Tanzu-Service-Mesh/services/concepts-guide/GUID-9E3F1F90-4310-415B-98C8-C06E59B8A5EE.html

[21] *Vmware Tanzu Service Mesh Usage Guide*. Accessed: Nov. 13, 2023. [Online]. Available: https://docs.vmware.com/en/VMware-Tanzu-Service-Mesh/services/using-tanzu-service-mesh-guide/GUID-8D483355-6F58-4AAD-9EAF-3B8E0A87B474.html

[22] J. Zirngibl, P. Sattler, and G. Carle, "A first look at SVCB and HTTPS DNS resource records in the wild," in *Proc. Int. Workshop Traffic Meas. Cybersecurity*, 2023, pp. 470–474.

SAIDULU ALDAS is currently a Principal Engineer and a Lead Software Architect with the Software and Advanced Technology Group (SATG), Intel Corporation. He is a more than 20 years veteran in wireless, cloud networking, and network virtualization technologies. He has a deep understanding of various technologies and strong business acumen to lead and transform innovation into customer-winning products.

ANDREW BABAKIAN is currently pursuing the Ph.D. degree in engineering with the University of Technology Sydney, Australia. He is also a Technologist and the Senior Director of the Modern Application Platform Business Unit, VMware, driving innovation efforts in software-defined networking and identity technologies for cloud-native architectures. His current research interests include naming systems and the evolution of internet identifiers.

Appendix E

Context-Aware Domain Name System (DNS) Query Handling

US011799899B2

(12) **United States Patent**
Babakian et al.

(10) **Patent No.: US 11,799,899 B2**
(45) **Date of Patent: Oct. 24, 2023**

(54) **CONTEXT-AWARE DOMAIN NAME SYSTEM (DNS) QUERY HANDLING**

(71) Applicant: **VMware, Inc.**, Palo Alto, CA (US)

(72) Inventors: **Andrew Babakian**, San Francisco, CA (US); **Emad Benjamin**, San Jose, CA (US); **Pere Monclus**, Morgan Hill, CA (US)

(73) Assignee: **VMWARE, INC.**, Palo Alto, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 671 days.

(21) Appl. No.: **16/516,649**

(22) Filed: **Jul. 19, 2019**

(65) **Prior Publication Data**

US 2021/0021634 A1 Jan. 21, 2021

(51) **Int. Cl.**

H04L 29/06	(2006.01)
H04L 9/40	(2022.01)
G06F 9/455	(2018.01)
H04L 61/2567	(2022.01)
H04L 61/4511	(2022.01)

(52) **U.S. Cl.**

CPC ***H04L 63/1441*** (2013.01); ***G06F 9/45558*** (2013.01); ***H04L 61/2567*** (2013.01); ***H04L 61/4511*** (2022.05); ***H04L 63/1416*** (2013.01); ***H04L 63/1425*** (2013.01); *G06F 2009/45587* (2013.01); *G06F 2009/45595* (2013.01)

(58) **Field of Classification Search**

CPC H04L 63/1416; H04L 63/1425; H04L 63/1441; H04L 61/1511; H04L 61/2567; G06F 9/45558

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2014/0245439 A1* 8/2014 Day G06F 21/577 726/23
2020/0314067 A1* 10/2020 Rudnik H04L 63/0236

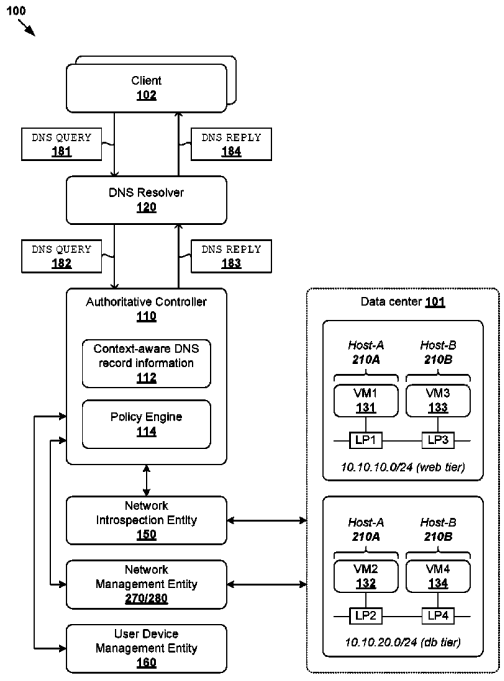
* cited by examiner

Primary Examiner — Jeffrey C Pwu
Assistant Examiner — Thong P Truong
(74) *Attorney, Agent, or Firm* — SU IP CONSULTING

(57) **ABSTRACT**

Example methods are provided a computer system to perform context-aware domain name system (DNS) query handling in a software-defined networking (SDN) environment. One example method may comprise detecting a DNS query to translate a domain name; identifying DNS record information that translates the domain name to a network address assigned to a virtualized computing instance; and identifying context information that is associated with the virtualized computing instance and mapped to the DNS record information. The method may also comprise: in response to detecting a potential security threat based on the context information, performing a remediation action to block access to the virtualized computing instance; but otherwise, generating and sending a DNS reply specifying the network address assigned to allow access to the virtualized computing instance.

21 Claims, 8 Drawing Sheets



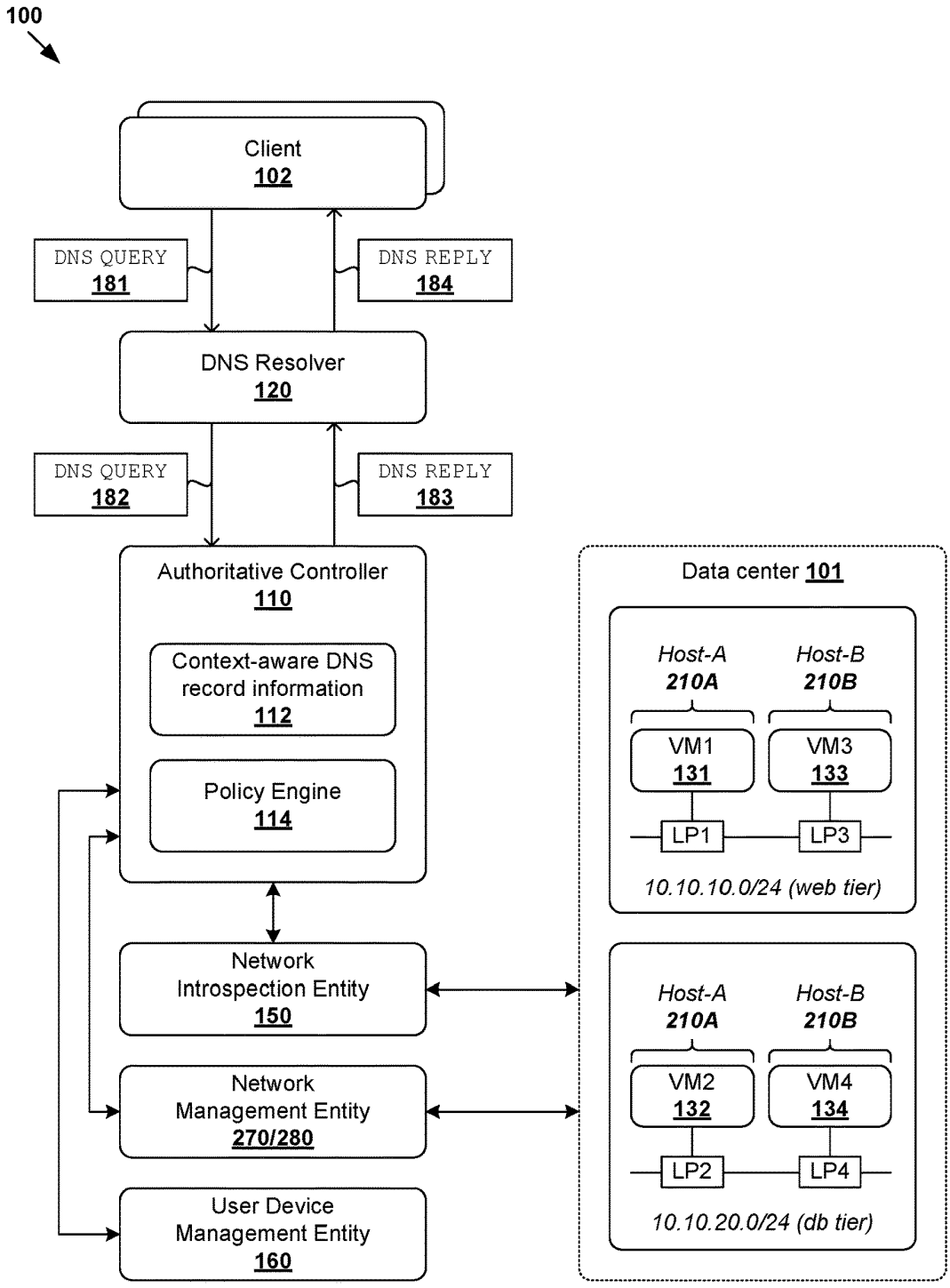

100
Client
102
DNS QUERY
181
DNS REPLY
184
DNS Resolver
120
DNS QUERY
182
DNS REPLY
183
Authoritative Controller
110
Context-aware DNS record information
112
Policy Engine
114
Network Introspection Entity
150
Network Management Entity
270/280
User Device Management Entity
160
Data center 101
Host-A
210A
Host-B
210B
VM1
131
VM3
133
LP1
LP3
10.10.10.0/24 (web tier)
Host-A
210A
Host-B
210B
VM2
132
VM4
134
LP2
LP4
10.10.20.0/24 (db tier)

Fig. 1

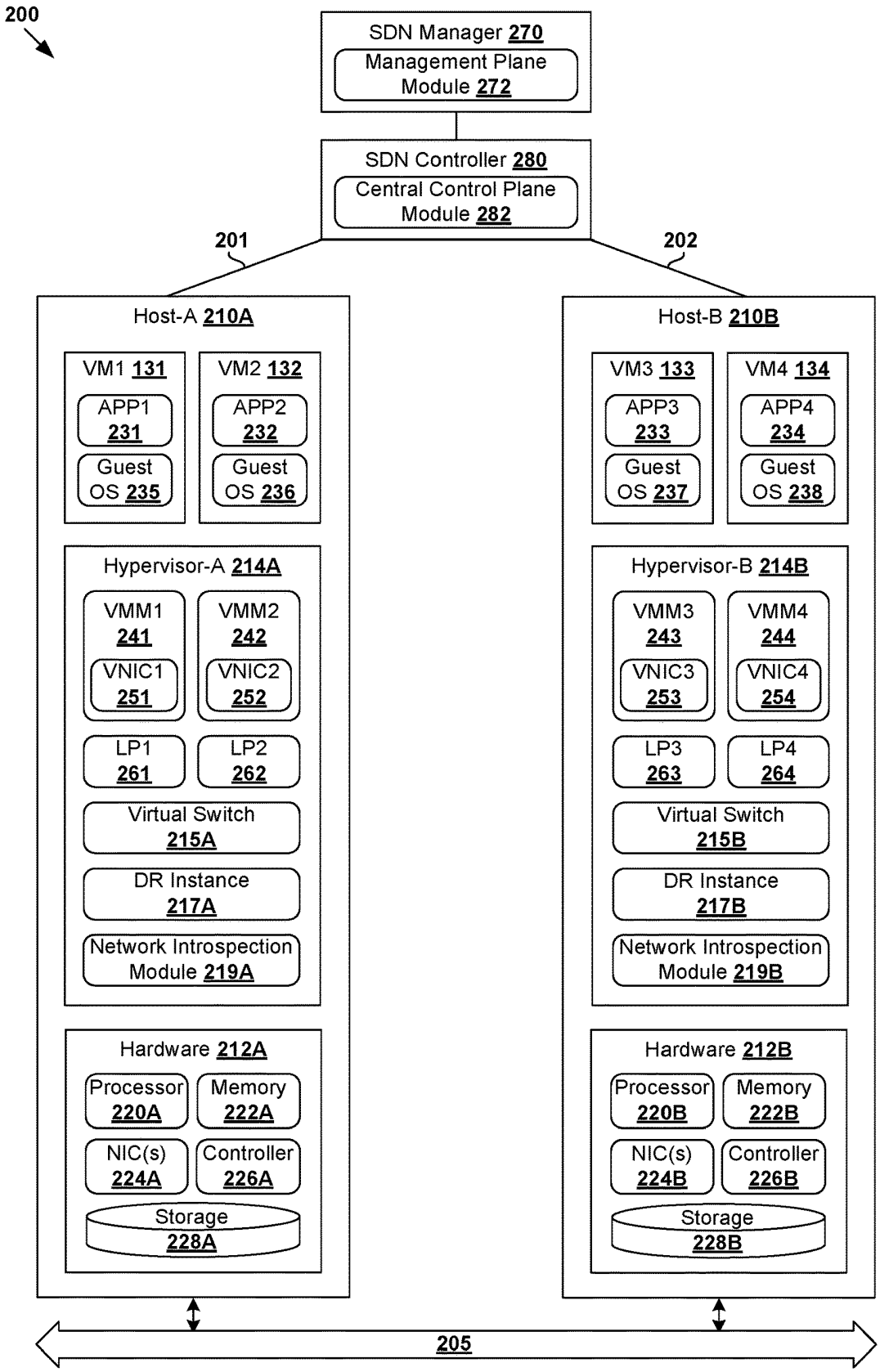

200
SDN Manager 270
Management Plane Module 272
SDN Controller 280
Central Control Plane Module 282
201
202
Host-A 210A
VM1 131
APP1 231
Guest OS 235
VM2 132
APP2 232
Guest OS 236
Hypervisor-A 214A
VMM1 241
VNIC1 251
VMM2 242
VNIC2 252
LP1 261
LP2 262
Virtual Switch 215A
DR Instance 217A
Network Introspection Module 219A
Hardware 212A
Processor 220A
Memory 222A
NIC(s) 224A
Controller 226A
Storage 228A
Host-B 210B
VM3 133
APP3 233
Guest OS 237
VM4 134
APP4 234
Guest OS 238
Hypervisor-B 214B
VMM3 243
VNIC3 253
VMM4 244
VNIC4 254
LP3 263
LP4 264
Virtual Switch 215B
DR Instance 217B
Network Introspection Module 219B
Hardware 212B
Processor 220B
Memory 222B
NIC(s) 224B
Controller 226B
Storage 228B
205

Fig. 2

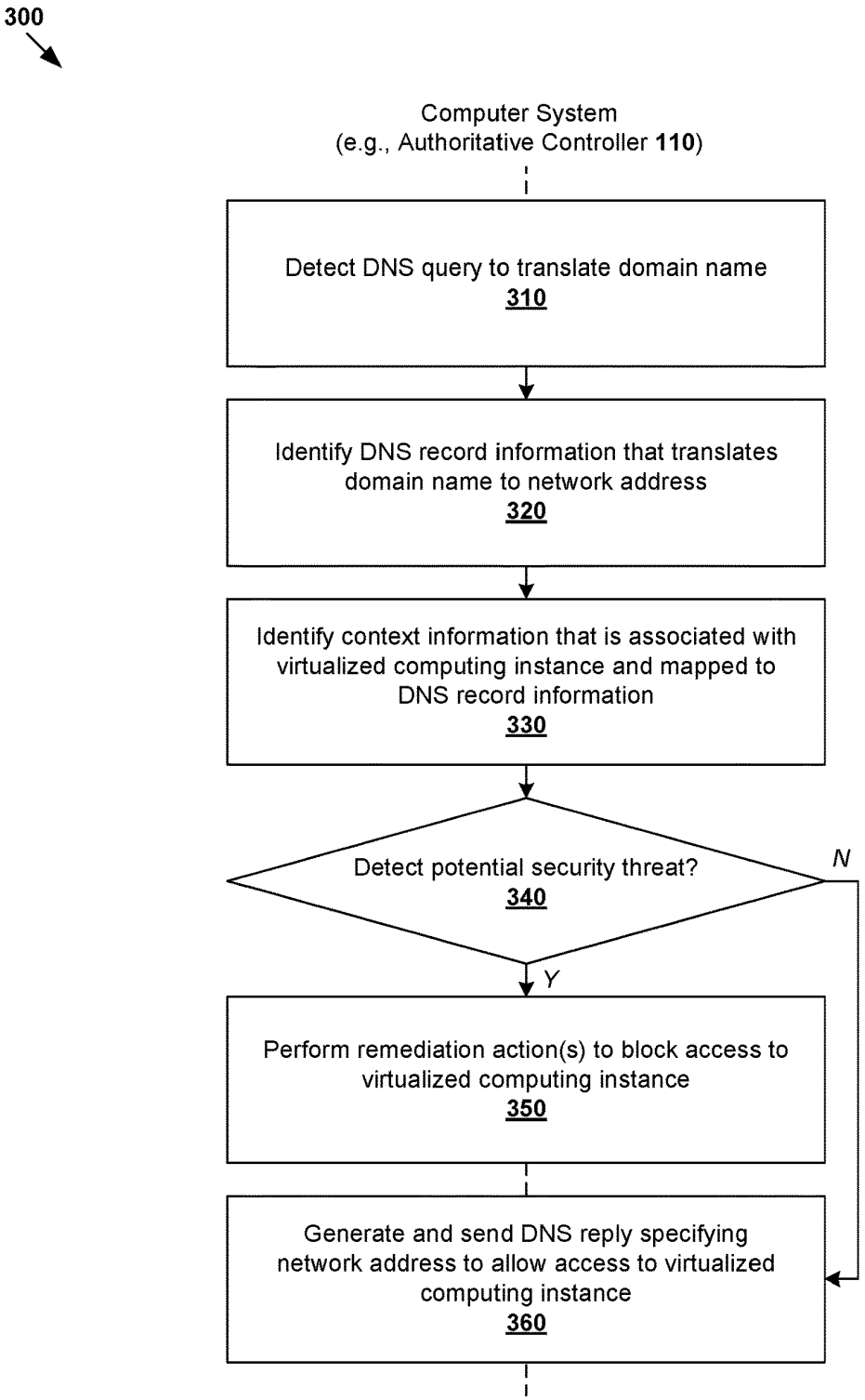

300
Computer System
(e.g., Authoritative Controller 110)
Detect DNS query to translate domain name
310
Identify DNS record information that translates domain name to network address
320
Identify context information that is associated with virtualized computing instance and mapped to DNS record information
330
Detect potential security threat?
340
N
Y
Perform remediation action(s) to block access to virtualized computing instance
350
Generate and send DNS reply specifying network address to allow access to virtualized computing instance
360

Fig. 3

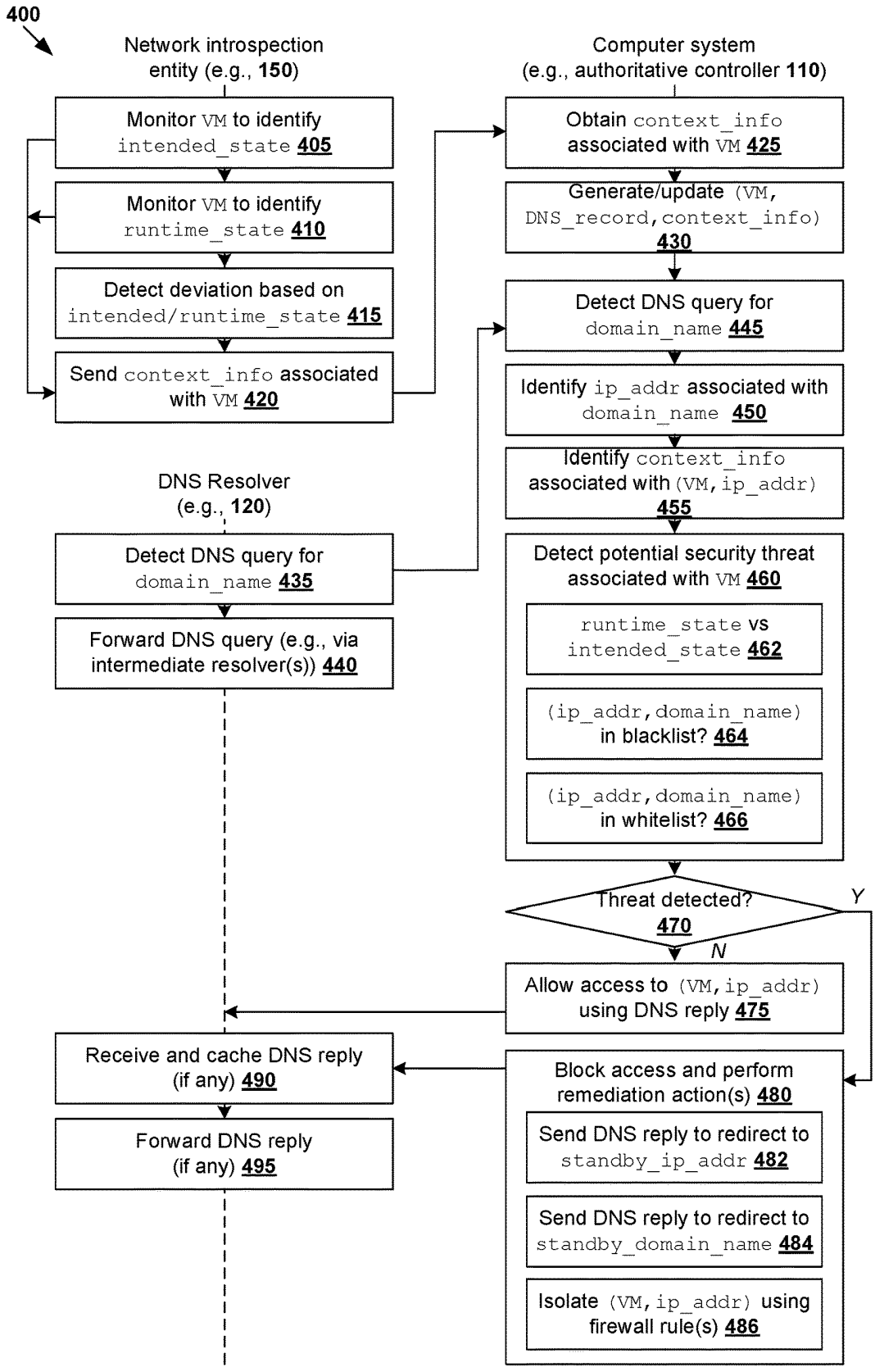
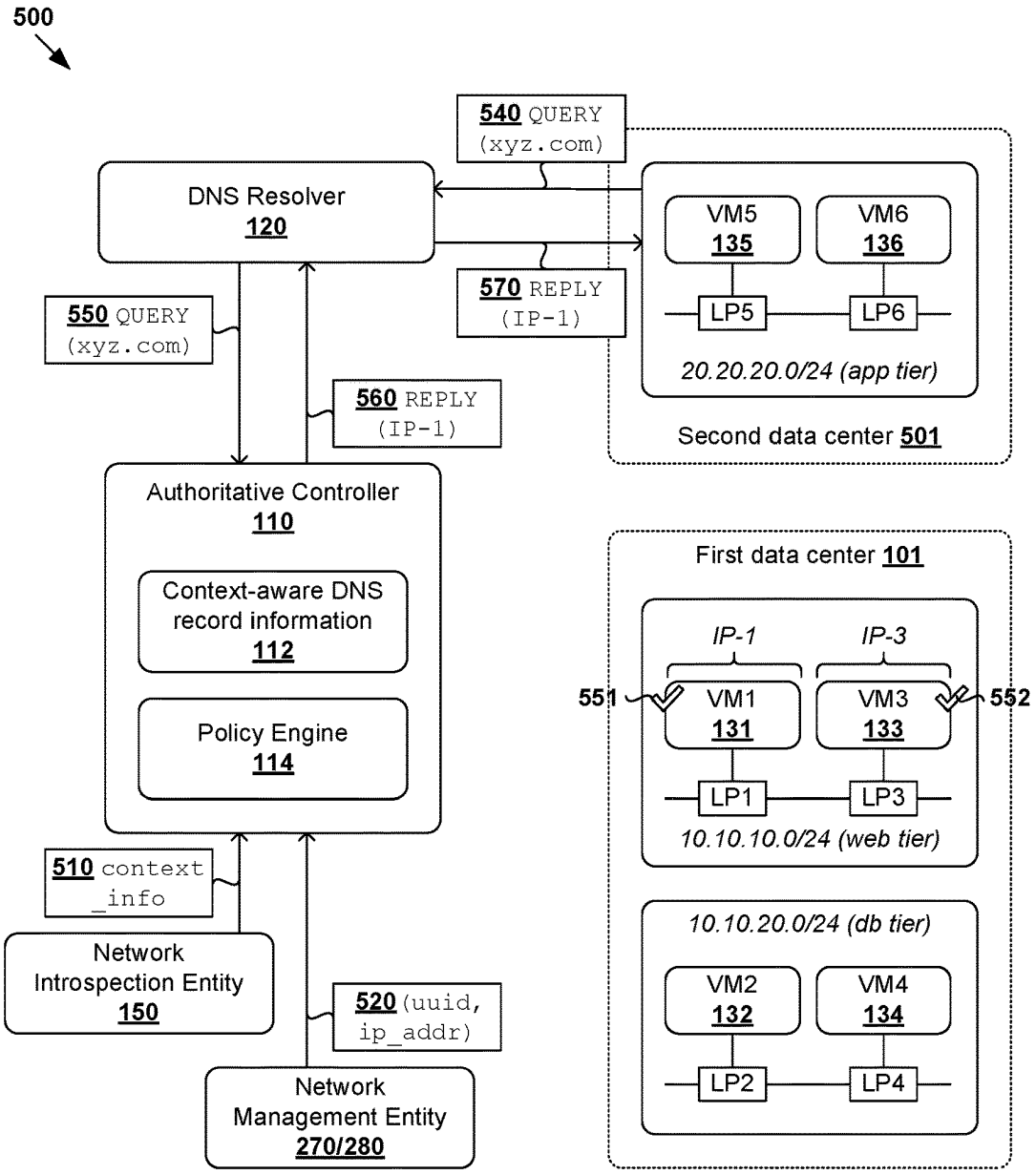

400
Network introspection entity (e.g., 150)
Monitor VM to identify intended_state 405
Monitor VM to identify runtime_state 410
Detect deviation based on intended/runtime_state 415
Send context_info associated with VM 420
DNS Resolver (e.g., 120)
Detect DNS query for domain_name 435
Forward DNS query (e.g., via intermediate resolver(s)) 440
Receive and cache DNS reply (if any) 490
Forward DNS reply (if any) 495
Computer system (e.g., authoritative controller 110)
Obtain context_info associated with VM 425
Generate/update (VM, DNS_record, context_info) 430
Detect DNS query for domain_name 445
Identify ip_addr associated with domain_name 450
Identify context_info associated with (VM, ip_addr) 455
Detect potential security threat associated with VM 460
runtime_state vs intended_state 462
(ip_addr, domain_name) in blacklist? 464
(ip_addr, domain_name) in whitelist? 466
Threat detected? 470
Y
N
Allow access to (VM, ip_addr) using DNS reply 475
Block access and perform remediation action(s) 480
Send DNS reply to redirect to standby_ip_addr 482
Send DNS reply to redirect to standby_domain_name 484
Isolate (VM, ip_addr) using firewall rule(s) 486

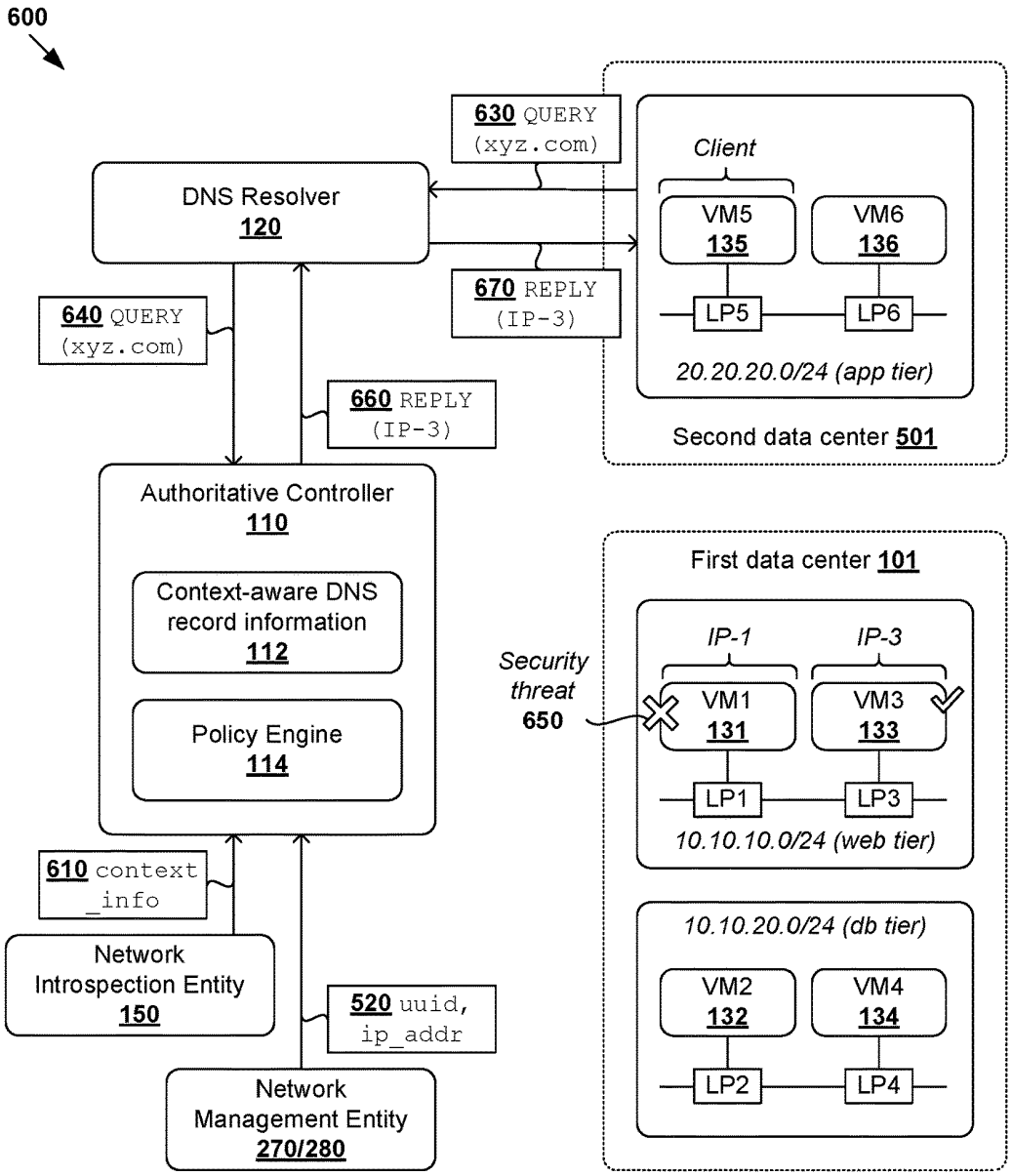
Fig. 4



Context-aware DNS record information **530/112**

name	record_type	ip_addr	uuid	context_info	
xyz.com	A	IP-1	ID-VM1	context-1 (healthy)	531
xyz.com	A	IP-3	ID-VM3	context-3 (healthy)	532

Fig. 5



Context-aware DNS record information **530/112**

name	record_type	ip_addr	uuid	context_info	
xyz.com	A	IP-1	ID-VM1	context-1* (healthy→unhealthy)	~620
xyz.com	A	IP-3	ID-VM3	context-3 (healthy)	~532

Fig. 6

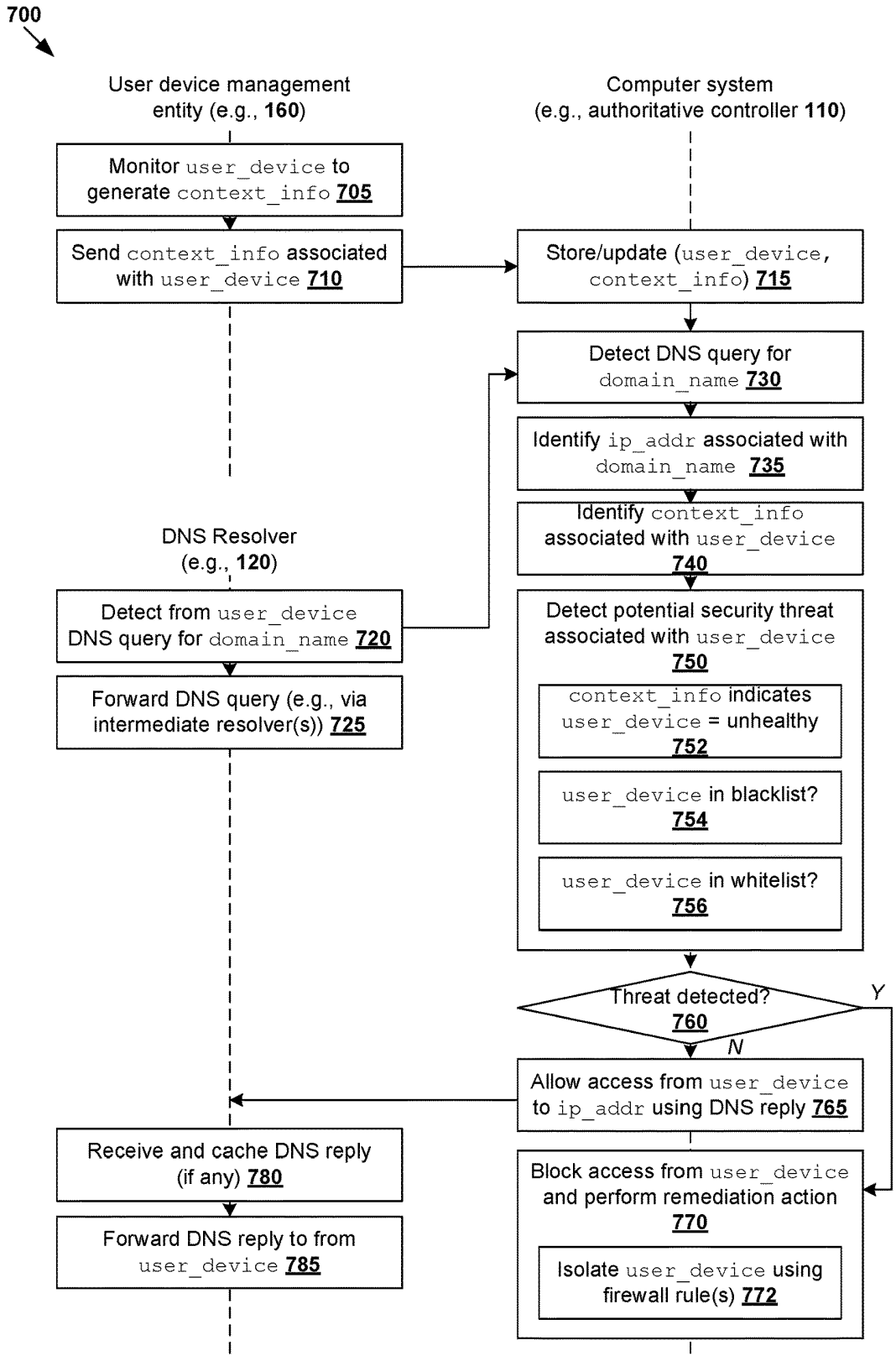
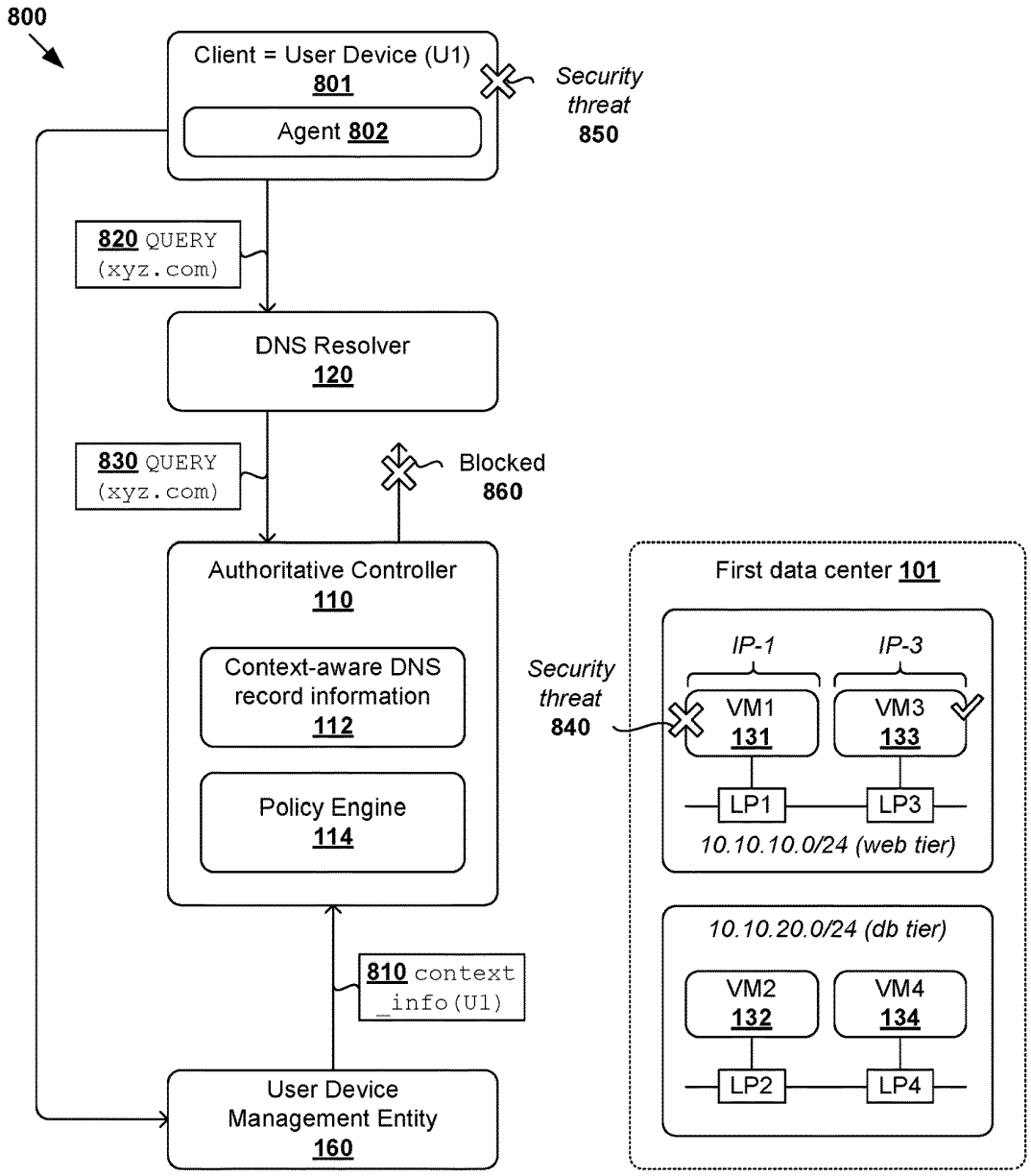

700
User device management entity (e.g., 160)
Computer system (e.g., authoritative controller 110)
Monitor user_device to generate context_info 705
Send context_info associated with user_device 710
Store/update (user_device, context_info) 715
Detect DNS query for domain_name 730
Identify ip_addr associated with domain_name 735
Identify context_info associated with user_device 740
DNS Resolver (e.g., 120)
Detect from user_device DNS query for domain_name 720
Detect potential security threat associated with user_device 750
context_info indicates user_device = unhealthy 752
user_device in blacklist? 754
user_device in whitelist? 756
Threat detected? 760
Y
N
Forward DNS query (e.g., via intermediate resolver(s)) 725
Allow access from user_device to ip_addr using DNS reply 765
Receive and cache DNS reply (if any) 780
Block access from user_device and perform remediation action 770
Isolate user_device using firewall rule(s) 772
Forward DNS reply to from user_device 785

Fig. 7



Context-aware DNS record information **530/112**

name	record_type	ip_addr	uuid	context_info	
xyz.com	A	IP-1	ID-VM1	context (unhealthy)	531
xyz.com	A	IP-3	ID-VM3	context (healthy)	532

Client context information (user device) **815**

ID,ip_addr	sw_version	sw_list	location	jailbreak_status
U1, IP-U1	1.0.3	(s1, s2)	Sydney	Yes (unhealthy)

Fig. 8

CONTEXT-AWARE DOMAIN NAME SYSTEM (DNS) QUERY HANDLING

BACKGROUND

Virtualization allows the abstraction and pooling of hardware resources to support virtual machines in a software-defined data center (SDDC). For example, through server virtualization, virtualized computing instances such as virtual machines (VMs) running different operating systems may be supported by the same physical machine (e.g., referred to as a "host"). Each VM is generally provisioned with virtual resources to run a guest operating system and applications. The virtual resources may include central processing unit (CPU) resources, memory resources, storage resources, network resources, etc. In practice, Domain Name System (DNS) allows a client (e.g., user device, VM) to use domain names instead of numerical addresses to connect with a server. When the client needs to access the server (e.g., VM) by its domain name, the client needs to have the domain name translated to an actual routable address through DNS querying. However, there is a risk that the client and/or server have gone rogue, which may expose other entities in the SDDC to malicious attacks.

BRIEF DESCRIPTION OF DRAWINGS

FIG. **1** is a schematic diagram illustrating an example software-defined networking (SDN) environment in which context-aware domain name system (DNS) query handling may be performed;

FIG. **2** is a schematic diagram illustrating a physical implementation view of hosts in the SDN environment in FIG. **1**;

FIG. **3** is a flowchart of an example process for a computer system to perform context-aware DNS query handling in an SDN environment;

FIG. **4** is a flowchart of an example detailed process for context-aware DNS query handling in an SDN environment according to a server-side approach;

FIG. **5** is a schematic diagram illustrating a first example of context-aware DNS query handling in an SDN environment according to a server-side approach;

FIG. **6** is a schematic diagram illustrating a second example of context-aware DNS query handling in an SDN environment according to a server-side approach;

FIG. **7** is a flowchart of an example detailed process for context-aware DNS query handling in an SDN environment according to a client-side approach; and

FIG. **8** is a schematic diagram illustrating an example of context-aware DNS query handling in an SDN environment according to a client-side approach.

DETAILED DESCRIPTION

In the following detailed description, reference is made to the accompanying drawings, which form a part hereof. In the drawings, similar symbols typically identify similar components, unless context dictates otherwise. The illustrative embodiments described in the detailed description, drawings, and claims are not meant to be limiting. Other embodiments may be utilized, and other changes may be made, without departing from the spirit or scope of the subject matter presented here. It will be readily understood that the aspects of the present disclosure, as generally described herein, and illustrated in the drawings, can be

arranged, substituted, combined, and designed in a wide variety of different configurations, all of which are explicitly contemplated herein.

Challenges relating to data center security and domain name system (DNS) will now be explained in more detail using FIG. **1**, which is a schematic diagram illustrating example software-defined networking (SDN) environment **100** in which context-aware DNS query handling may be performed. It should be understood that, depending on the desired implementation, SDN environment **100** may include additional and/or alternative components than that shown in FIG. **1**. Although the terms "first" and "second" are used to describe various elements, these elements should not be limited by these terms. These terms are used to distinguish one element from another. For example, a first element may be referred to as a second element, and vice versa.

In the example in FIG. **1**, SDN environment **100** includes multiple hosts **210**A-B that support various virtualized computing instances in the form of virtual machines (VMs) **131-134**. For example, host-A **210**A supports VMs **131-132**, and host-B **210**B supports VMs **133-134**. In practice, SDN environment **100** may include any number of hosts (also known as a "host computers", "host devices", "physical servers", "server systems", "transport nodes," etc.), where each host may be supporting tens or hundreds of virtual machines (VMs).

VMs **131-134** may be deployed as network nodes in data center **101** to implement a multi-node application whose functionality is distributed over multiple network nodes. For example, VM**1** **131** and VM**3** **133** may be web servers that belong to a "web tier" of the multi-node application. In another example, VM**2** **132** and VM**4** **134** may be database servers that belong to a "database tier" of the multi-node application. Depending on the desired implementation, multiple network nodes may implement the same functionality to improve performance and fault tolerance, in which case a load balancer (not shown) may be deployed to distribute traffic among those network nodes using any suitable algorithm (e.g., round robin, least connection, chained failover, source IP address hash, etc.).

Hosts **210**A-B will be explained in more detail using FIG. **2**, which is a schematic diagram illustrating physical implementation view **200** of example SDN environment **100** in FIG. **1**. Each host **210**A/**210**B may include suitable hardware **212**A/**212**B and virtualization software (e.g., hypervisor-A **214**A, hypervisor-B **214**B) to support respective VMs **131-134**. Hypervisor **214**A/**214**B maintains a mapping between underlying hardware **212**A/**212**B and virtual resources allocated to respective VMs. Hardware **212**A/**212**B includes suitable physical components, such as central processing unit(s) (CPU(s)) or processor(s) **220**A/**220**B; memory **222**A/**222**B; physical network interface controllers (NICs) **224**A/**224**B; storage controller(s) **226**A/**226**B; and storage disk(s) **228**A/**228**B, etc.

Virtual resources are allocated to VMs **131-134** to support respective application(s) **231-234** and guest operating systems **235-238**. For example, the virtual resources may include virtual CPU, guest physical memory, virtual disk, virtual network interface controller (VNIC), etc. Hardware resources may be emulated using virtual machine monitors (VMMs). For example in FIG. **2**, VNICs **251-254** are virtual network adapters for VMs **131-134**, respectively, and are emulated by corresponding VMMs **241-244** instantiated by their respective hypervisors **214**A-B. The VMMs may be considered as part of respective VMs, or alternatively, separated from the VMs. Although one-to-one relationships

are shown, one VM may be associated with multiple VNICs (each VNIC having its own network address).

Although examples of the present disclosure refer to VMs, it should be understood that a "virtual machine" running on a host is merely one example of a "virtualized computing instance" or "workload." A virtualized computing instance may represent an addressable data compute node (DCN) or isolated user space instance. In practice, any suitable technology may be used to provide isolated user space instances, not just hardware virtualization. Other virtualized computing instances may include containers (e.g., running within a VM or on top of a host operating system without the need for a hypervisor or separate operating system or implemented as an operating system level virtualization), virtual private servers, client computers, etc. Such container technology is available from, among others, Docker, Inc. The VMs may also be complete computational environments, containing virtual equivalents of the hardware and software components of a physical computing system.

The term "hypervisor" may refer generally to a software layer or component that supports the execution of multiple virtualized computing instances, including system-level software in guest VMs that supports namespace containers such as Docker, etc. Hypervisors **214**A-B may each implement any suitable virtualization technology, such as VMware ESX® or ESXi™ (available from VMware, Inc.), Kernel-based Virtual Machine (KVM), etc. The term "packet" may refer generally to a group of bits that can be transported together, and may be in another form, such as "frame," "message," "segment," etc. The term "traffic" or "flow" may refer generally to multiple packets. The term "layer-2" may refer generally to a link layer or media access control (MAC) layer; "layer-3" to a network or Internet Protocol (IP) layer; and "layer-4" to a transport layer (e.g., using Transmission Control Protocol (TCP), User Datagram Protocol (UDP), etc.), in the Open System Interconnection (OSI) model, although the concepts described herein may be used with other networking models.

Through virtualization of networking services in SDN environment **100**, logical networks (also referred to as overlay networks or logical overlay networks) may be provisioned, changed, stored, deleted and restored programmatically without having to reconfigure the underlying physical hardware architecture. A logical network may be formed using any suitable tunneling protocol, such as Virtual eXtensible Local Area Network (VXLAN), Stateless Transport Tunneling (STT), Generic Network Virtualization Encapsulation (GENEVE), etc. For example, VXLAN is a layer-2 overlay scheme on a layer-3 network that uses tunnel encapsulation to extend layer-2 segments across multiple hosts which may reside on different layer 2 physical networks. In the example in FIG. **1**, VM**1** **131** and VM**3** **133** (e.g., web tier) may be deployed in one logical network and connected via a first logical switch. VM**2** **132** and VM**4** **134** (e.g., database tier) may be deployed in another logical network and connected via a second logical switch.

Hypervisor **214**A/**214**B implements virtual switch **215**A/**215**B and logical distributed router (DR) instance **217**A/**217**B to handle egress packets from, and ingress packets to, corresponding VMs. In SDN environment **100**, logical switches and logical DRs may be implemented in a distributed manner and can span multiple hosts. For example, logical switches that provide logical layer-2 connectivity, i.e., an overlay network, may be implemented collectively by virtual switches **215**A-B and represented internally using forwarding tables (not shown) at respective virtual switches **215**A-B. The forwarding tables may each include entries that collectively implement the respective logical switches. Further, logical DRs that provide logical layer-3 connectivity may be implemented collectively by DR instances **217**A-B and represented internally using routing tables (not shown) at respective DR instances **217**A-B. The routing tables may each include entries that collectively implement the respective logical DRs.

Packets may be received from, or sent to, each VM via an associated logical port. For example, logical switch ports **261-264** are associated with respective VMs **131-134**. Here, the term "logical port" or "logical switch port" may refer generally to a port on a logical switch to which a virtualized computing instance is connected. A "logical switch" may refer generally to an SDN construct that is collectively implemented by virtual switches **215**A-B in FIG. **2**, whereas a "virtual switch" may refer generally to a software switch or software implementation of a physical switch. In practice, there is usually a one-to-one mapping between a logical port on a logical switch and a virtual port on virtual switch **215**A/**215**B. However, the mapping may change in some scenarios, such as when the logical port is mapped to a different virtual port on a different virtual switch after migration of the corresponding virtualized computing instance (e.g., when the source host and destination host do not have a distributed virtual switch spanning them).

Hosts **210**A-B may maintain data-plane connectivity with each other via physical network **205**, and control-plane connectivity with network management entities **270-280** via respective control-plane channels **201-202**. One example of SDN controller **280** is the NSX controller component of VMware NSX® (available from VMware, Inc.) that operates on a central control plane. SDN controller **280** may be a member of a controller cluster (not shown for simplicity) that is configurable using SDN manager **270** operating on a management plane. Network management entity **270**/**280** may be implemented using physical machine(s), VM(s), or both.

Referring to FIG. **1** again, DNS allows some servers to be accessed by client **102** using domain names instead of numerical addresses. In practice, the term "client" may refer generally to any suitable computer system that may require access to a server. Client **102** may be a VM (see FIGS. **5-6**), a container, a user device (see FIGS. **7-8**), or any other network node. Consider a scenario where client **102** wishes to access VM**1** **131** (e.g., to access functionality implemented by APP**1** **231**). Prior to client-server connection establishment, client **102** may send a DNS query to DNS resolver **120**. The DNS query is to request for a translation of a domain name (e.g., www.xyz.com) to a routable network address (e.g., IP address=IP-1) assigned to VM**1** **131**.

DNS resolver **120** may then forward the DNS query to authoritative controller **110**, which may then respond with a DNS reply. Authoritative controller **110** may represent an authoritative name server that maintains DNS records and responds to DNS queries for domain names for a particular DNS zone it is responsible for. In practice, DNS resolver **120** may be a "recursive resolver" (also known as a "DNS recursor") that operates as the first stop for a DNS query. After receiving a DNS query from a client, a recursive resolver will either respond with cached data, or execute a sequence of iterative queries in the case of a cache miss. In the latter case, DNS resolver **120** may send respective queries to a root name server (not shown), a top-level domain (TLD) name server (not shown), and finally to authoritative controller **110**. After receiving a DNS reply from authoritative controller **110**, DNS resolver **120** may

send a DNS reply containing the requested IP address (e.g., IP-1) and a time to live (TTL) parameter to client **102**. This way, client **102** may start communicating with a server (e.g., VM1 **131**) using the requested IP address.

One of the challenges in SDN environment **100** is improving the overall network security. For example in FIG. **1**, servers (e.g., VMs **131-134**) and/or client **102** are susceptible to malicious attacks such as malware infection. In practice, the term "malware" is used as an umbrella term to cover various forms of hostile or intrusive software, such as viruses, worms, Trojan horse programs, spyware, phishing, adware, riskware, rookits, spams, scareware, ransomware, a combination thereof, etc. Conventionally, authoritative controller **110** and DNS resolver **120** may not be aware of whether servers and/or clients have been compromised. In this case, the DNS query handling process may inadvertently facilitate access to rogue entities, which may in turn expose more entities to malicious attacks.

Context-Aware DNS Query Handling

According to examples of the present disclosure, a "context-aware" approach for DNS query handling may be implemented to improve security threat mitigation in SDN environment **100**. For example, DNS record information associated with a server (e.g., VM1 **131**) may be mapped to its context information, which may be used to detect any potential security threats during DNS query handling. This way, DNS query handling may be used as a form of access control to improve network security.

In more detail, FIG. **3** is a flowchart of example process **300** for a computer system to perform context-aware DNS query handling in SDN environment **100**. Example process **300** may include one or more operations, functions, or actions illustrated by one or more blocks, such as **310** to **360**. The various blocks may be combined into fewer blocks, divided into additional blocks, and/or eliminated depending on the desired implementation. Examples of the present disclosure may be implemented by any suitable "computer system," such as authoritative controller **110**, DNS resolver **120**, a combination of both, etc. In the following, various examples will be discussed using VMs **131-132** as example "virtualized computing instances" or "servers" that is accessible using a domain name.

At **310** in FIG. **3**, authoritative controller **110** may detect a DNS query to translate a domain name. In the example in FIG. **1**, the DNS query (see **181-182**) may be initiated by client **102** requiring access to a domain name (e.g., www.xyz.com), and forwarded to authoritative controller **110** via DNS resolver **120**.

At **320** and **330** in FIG. **3**, authoritative controller **110** may identify DNS record information and context information that is mapped to the DNS record information (see **112** in FIG. **1**). The DNS record information translates the domain name to a network address (e.g., IP-1) assigned to VM1 **131** supporting APP1 **231**.

At **340** and **350**, in response to detecting a potential security threat based on the context information, authoritative controller **110** may perform remediation action(s) to block access to VM1 **131**. Otherwise, at **360**, a DNS reply (see **183-184**) specifying the network address (e.g., IP-1) assigned to VM1 **131** may be generated and sent to allow access to VM1 **131**.

As will be discussed using FIGS. **4-6**, any suitable "remediation action" may be performed to block access to VM1 **131**. In a first example, VM3 **133** may be a "standby virtualized computing instance" to which traffic may be directed. In this case, the remediation action may involve redirecting client **102** to a standby network address (e.g., IP-3) by generating and sending a DNS reply specifying the standby network address assigned to VM3 **133**. In a second example, the remediation action may involve translating the domain name (www.xyz.com) to another (standby) domain name (e.g., abcd_patched_version_app.xyz.com). In a third example, a notification may be generated and sent to network management entity **270**/**280** to cause isolation of VM1 **131** using firewall rule(s). In a fourth example, the remediation action may involve returning nothing back to client **102**, thereby blocking access.

As will be discussed using FIGS. **4-6**, any suitable "context information" may be used for detecting potential security threat(s), such as information relating to an intended state associated with VM1 **131**, a runtime state associated with VM1 **131**, etc. In this case, based on that the context information, a deviation between the intended state and runtime state may be identified to detect the potential security threat. In practice, the context information associated with VM1 **131** may be obtained from various entities in SDN environment, such as network introspection entity **150**, network management entity **270**/**280**, etc.

Additionally, a "client-side approach" may be implemented to detect any potential security threat associated with client **102** to protect servers (e.g., VM1 **131**) against rogue client **102**. As will be discussed using FIGS. **7-8**, "client context information" may be obtained from user device management entity **160** to detect a potential security threat associated with client **102**. For example, client **102** may be a user device (e.g., mobile device, laptop or tablet computer, desktop computer, etc.). In this case, the client context information may include one or more of the following: hardware profile information (e.g., device type), software profile information (e.g., software version, list of frequently-used applications, jailbreak status, etc.), geographical location information (e.g., location=Sydney), user profile information (e.g., user's role and permission), etc.

According to examples of the present disclosure, context information associated with servers (e.g., VM1 **131**) and/or clients (e.g., **102**) may be used to detect potential security threats. Using a server-side approach, healthy client **102** may be protected from an "unhealthy" server (e.g., VM1 **131**) and redirected to an alternative "healthy" server (e.g., VM3 **133**) instead. Using a client-side approach, a healthy server (e.g., VM1 **131**) may be protected from unhealthy client **102**. In both cases, context-aware DNS query handling may be used for security threat mitigation to reduce the attack surface in SDN environment **100**. Examples of the present disclosure may be implemented to invoke security controls and provide an indirect actuation in SDN environment **100**, irrespective of where clients and servers are deployed. Various examples will be explained below.

Server-Side Example

FIG. **4** is a flowchart of example detailed process **400** for context-aware DNS query handling in SDN environment **100** according to a server-side approach. Example process **400** may include one or more operations, functions, or actions illustrated at **405** to **495**. The various operations, functions or actions may be combined into fewer blocks, divided into additional blocks, and/or eliminated depending on the desired implementation. Various examples will be explained using FIG. **5** and FIG. **6**. FIG. **5** is a schematic diagram illustrating first example **500** of context-aware DNS query handling in SDN environment **100** according to a server-side approach. FIG. **6** is a schematic diagram illustrating second example **600** of context-aware DNS query handling in SDN environment **100** according to a server-side approach.

(a) Context Information

At **405** in FIG. **4**, network introspection entity **150** may monitor VMs **131-134** to identify or establish their normal operational behavior (i.e., intended state). Any suitable approach may be used for network monitoring, such as using VMware's AppDefense™, etc. For example, network introspection entity **150** may interact with hypervisor **214**A/**214**B (see network introspection module **219**A/**219**B in FIG. **2**) and/or guest OS **235-238** (e.g., via an agent executed by each guest OS). To establish their intended state, network introspection entity **150** may monitor VMs **131-134** and respective applications **231-234** for a period of time.

At **410-415** in FIG. **4**, network introspection entity **150** may monitor VMs **131-134** to identify or establish their runtime operational behavior (i.e., runtime state), and compare the runtime state against their corresponding intended state. This way, any deviation between the intended state and the runtime state may be identified. In practice, some deviations or anomalies may be identified to be legitimate changes due to system updates or software patching, whereas others could potentially real threats.

At **420** in FIG. **4**, network introspection entity **150** may generate and send context information associated with VMs **131-134** to authoritative controller **110**. Although not shown in FIG. **4** for simplicity, network introspection entity **150** may also perform any suitable action, such as suspending or shutting down a VM, generating a snapshot of the VM, alerting an entity (e.g., program or user) responsible for investigating threats, etc.

Depending on the desired implementation, the context information may specify any of the following fingerprint information associated with a workload (e.g., VM or container): OS kernel behavior information (e.g., to monitor OS kernel integrity based on an expectation of module(s) being executed); process behavior information (e.g., process network attestation in terms of all egress/ingress processes that have established network connections); process ID information (e.g., historical visibility of all processes that are running inside the workload); network flow information (e.g., ingress/egress network flows to monitor network-level communication), UUID and/or IP address information for identifying the workload, etc. In practice, process ID monitoring may also provide visibility as to how a process ID might be exploited by a new process (e.g., malware or command control).

(b) Context-Aware DNS Record Information

At **425** in FIG. **4**, authoritative controller **110** may obtain context information (see **510** in FIG. **5**) associated with server=VM **131/133** from network introspection entity **150**. The context information may specify the intended state of VM **131/133**, runtime state of VM **131/133**, deviation(s) between the intended state and runtime state, or any combination thereof. Additionally, authoritative controller **110** may obtain identification information (see **520** in FIG. **5**) associated with VM **131/133** from SDN manager **270** and/or SDN controller **280**, such as their UUID, IP address, MAC address, logical network information, etc.

At **430** in FIG. **4**, authoritative controller **110** (e.g., using policy engine **114**) may generate context-aware DNS record information (see **530** in FIG. **5**) by mapping DNS record information associated with VM **131/133** to their context information. The context-aware DNS record information (see **112**) provides a one-to-one mapping between DNS information record(s) associated with VM **131/133**, and context information associated with VM **131/133**. In practice, context-aware DNS record information may be updated dynamically at block **430** based on any updated context information.

In practice, the term "DNS record information" may refer to resource record(s) that provide information about a specific resource. For example, an address (A, AAAA) record specifies an IP version 4 (IPv4) address or IP version 6 (IPv6) address that is mapped to a domain name. A CNAME record specifies redirection to another domain name. A reverse-lookup pointer (PTR) record specifies a domain name associated to an IP address. A host information (HINFO) record specifies general information about a host (e.g., OS information). A name server (NS) record specifies an authoritative name server for a given host. An integrated services digital network (ISDN) record specifies an ISDN address for a host. A service (SRV) record specifies a hostname and a port number of a particular service, etc.

In the example in FIG. **5**, domain name=www.xyz.com may be mapped to both VM**1** **131** and VM**3** **133**. At **531**, context-aware DNS record information is generated by mapping a DNS record (domain name=www.xyz.com, IP-1) to context information (labelled "context-1") associated with VM**1** **131**. At **532**, context-aware DNS record information is generated by mapping another DNS record (domain name=www.xyz.com, IP-3) to context information (labelled "context-3") associated with VM**3** **133**. The context information (e.g., context-1 and context-3) may be stored in any suitable form, such as a unique ID that is generated based on (OS kernel behavior information, process behavior information, process ID information, network flow information) discussed at block **420**.

The context information (e.g., context-1 and context-3 in FIG. **5**) may be used to derive a health status associated with a workload (e.g., VM, container). In the case of semantic DNS, a "semantic name" may refer generally to a name that has been decorated with additional metadata that implies its intended behavior. The metadata represents an abstraction of various resources, such as application processes, virtualization-based processes, network communication processes, or any other functionality that may be associated with the workload, etc. For example, a semantic CNAME may include (CNAME, fingerprint metadata), where the "fingerprint" (or DNA) represents metadata associated with what is running of the workload. In practice, semantic DNS may be implemented for other types of records, such as A, AAAA, etc.

To implement semantic DNS, there may be two top-level steps: (1) generate a fingerprint associated with a workload and (2) add semantic to the fingerprint's metadata. In relation to the first step, the fingerprint may be generated through recursive traversal of what is running on the workload, such as a long string representing all directory structures and associated processes. As explained at block **420**, the fingerprint information may include OS kernel behavior information, process behavior information, process ID information, network flow information, UUID and/or IP address information, or any combination thereof, etc.

In relation to the second step, the semantic connection or interpretation may involve inquiring an existing cache of predefined tokens, token cache (i.e., tokens learnt over time by the system and authenticated by a rinsing system) for specific words that exists in the fingerprint information. For example, if particular word or token (e.g., "commons-codec-1.11") appears in a partial copy of a fingerprint, then a semantic CNAME decoration process may involve querying the token cache to determine whether the token has been flagged as suspicious in the past, or it has the right to access,

and any other suitable security-based properties defined by the user. This way, a health status associated with the workload may be derived, such as healthy (see **531**/**532** in FIG. **2**) or unhealthy (see **620** in FIG. **6** to be discussed below).

(c) Context-Aware DNS Query Handling

Context-aware DNS record information **530** may be used during DNS query handling as a form of security control. In the following, consider a scenario where client=VM**5** **135** wishes to access server=VM**1** **131** using domain name=www.xyz.com. In the example in FIG. **5**, VM**5** **135** and VM**6** **136** may be application servers (i.e., application tier) that are deployed on second data center **502**. Note that data center **101**/**502** may be deployed on a private cloud platform (e.g., on-premise SDDC), a public cloud platform, or a combination of both. A public cloud platform may be implemented using any suitable cloud technology, such as Amazon Web Services® (AWS) and Amazon Virtual Private Clouds (VPCs); VMware Cloud™ (VMC) on AWS; Microsoft Azure®; Google Cloud Platform™, IBM Cloud™; a combination thereof, etc. Amazon VPC and Amazon AWS are registered trademarks of Amazon Technologies, Inc.

Referring to FIG. **4** again, at **435**-**440** in FIG. **4**, in response to detecting a DNS query for domain name=www.xyz.com initiated by client=VM**5** **135**, DNS resolver **120** may forward the DNS query to authoritative controller **110**. In practice, DNS resolver **120** may first check whether the DNS query may be satisfied using cached information. In not (i.e., cache miss), the DNS query will be forwarded to authoritative controller **110**. See DNS query **540**-**550** in FIG. **5**.

At **450**-**455** in FIG. **4**, in response to detecting the DNS query from DNS resolver **120**, authoritative controller **110** may identify context-aware DNS record information **531**, which maps DNS record information (www.xyz.com, IP-1) with context information (i.e., context-1) associated with VM**1** **131**. At **460**-**470** in FIG. **4**, based on the context information, authoritative controller **110** may detect whether there is a potential security threat associated with VM**1** **131**. If yes (see **480**), access to VM**1** **131** is blocked, but otherwise (see **470**), access is allowed.

Depending on the desired implementation, block **460** may involve determining whether the context information indicates any deviation between an intended state and a runtime state (see **462**). Block **460** may also involve checking whether domain name=www.xyz.com and/or IP-1 are included in a blacklist (see **464**) or a whitelist (see **466**). If found in the blacklist, access will be automatically blocked. If not found in the whitelist, access will also be automatically blocked.

At **470** (yes) and **480** in FIG. **4**, in response to detecting a potential security threat based on the context information, authoritative controller **110** may block access to VM**1** **131** and perform remediation action(s). Otherwise, at **475**, access is allowed by generating and sending a DNS reply to client=VM**5** **135** via DNS resolver **120**.

In the example in FIG. **5**, both VM**1** **131** and VM**3** **133** are determined to be "healthy" (see **551**-**552**) based on their context information. For example, VM **131**/**133** may be considered "healthy" when behaving normally or as expected during runtime. In contrast, VM **131**/**133** may be considered "unhealthy" (or compromised) when deviation or anomaly is detected during runtime. In this case, a DNS reply (see **560**-**570**) specifying the requested IP address=IP-1 is sent, thereby allowing application server VM**5** **135** to access web server VM**1** **131**. DNS resolver **120** may then cache the DNS reply for future DNS queries for the same domain name, and forward the DNS reply to VM**5** **135** according to blocks **490**-**495**.

(d) Blocking Access

In another example in FIG. **6**, VM**1** **131** is determined to be "unhealthy" based on updated context information (see **610**) from network introspection entity **150**. In this case, context-aware DNS record information **531**/**620** associated with VM**1** **131** may be updated according to block **430**. In response to detecting a DNS query (see **630**-**640**) initiated by client=VM**5** **135**, authoritative controller **110** may detect a potential security threat (see **650**) based on updated context-aware DNS record information **620**, which indicates that VM**1** **131** is now "unhealthy."

According to block **480**, authoritative controller **110** may block access to VM**1** **131** and perform remediation action(s). In one example (see **482**), authoritative controller **110** may redirect to a standby IP address (e.g., IP-3) associated with standby web server=VM**3** **133**, which is "healthy" based on context-aware DNS record information **532**. See **660**-**670** in FIG. **6**. In another example (see **484**), a redirection to a standby or specific domain name (e.g., abcd_patched_version_app.xyz.com) may be initiated. Using semantic DNS explained above using blocks **425**-**430**, additional DNS record information may be configured to have the necessary syntax and semantic of names to ensure redirection to healthy servers. Depending on the desired implementation, the remediation action may involve returning nothing back to VM**5** **135**, thereby blocking access to VM**1** **131**.

Further (see **486**), a notification may be generated and sent to network management entity **270**/**280** to cause isolation of VM**1** **131** using firewall rule(s). For example, hypervisor **214**A on host-A **210**A may implement a distributed firewall (DFW) engine to filter packets to and from VM**1** **131**. SDN controller **160** may be used to configure firewall rules that are enforceable by the DFW engine on host-A **110**A. Firewall rules may be defined using five tuples to match a specific packet flow, such as source IP address, source port number (PN), destination IP address, destination PN, and protocol, in addition to an action (e.g., allow or block).

Client-Side Example

FIG. **7** is a flowchart of example detailed process **700** for context-aware DNS query handling in SDN environment **100** according to a client-side approach. Example process **700** may include one or more operations, functions, or actions illustrated at **705** to **785**. The various operations, functions or actions may be combined into fewer blocks, divided into additional blocks, and/or eliminated depending on the desired implementation. Various examples will be explained using FIG. **8**, which is a schematic diagram illustrating example **800** of context-aware DNS query handling in SDN environment **100** according to a client-side approach.

In the example in FIG. **8**, client=user device **801** operated by a user wishes to access VM**1** **131** using domain name=www.xyz.com. In practice, a bring-your-own-device (BYOD) approach has allowed corporate users to have freedom of choice, such as smartphones, tablets and laptop computers. This way, users may choose to work from the office, or remotely. However, remote user devices or endpoints may be more challenging to control, which affect how they are authenticated and connected securely. Using a context-aware approach, access by user device **801** may be allowed or blocked based on client context information associated with user device **801**.

(a) Context Information

At **810** in FIG. **8**, user device management entity **160** may monitor various user devices including "U1" **801**, and send client context information to authoritative controller **110**. In practice, user device management entity **160** may be a mobile device management (MDM) entity that is responsible for managing, monitoring and securing mobile devices accessing resources in SDN environment **100**. At **815** in FIG. **8**, authoritative controller **110** may store or update the client context information in association with any identification information (e.g., device ID) associated with user device **801**. See corresponding **705-715** in FIG. **7**.

In the example in FIG. **8**, client context information **815** may include one or more of the following (not all shown for simplicity): hardware profile information (e.g., device type=smartphone, manufacturer, etc.), software profile information (e.g., software version=1.0.3, software list=(s1, s2) specifying frequently-used applications, jailbreak status=yes, etc.), geographical location information (e.g., location=Sydney), user profile information (e.g., user's role and permission), etc. In one example, user device management entity **160** may interact with agent **802** running on user device **801** to collect client context information **815**.

(b) Context-Aware DNS Query Handling

At **820-830** in FIG. **8**, in response to detecting a DNS query initiated by user device **801** via DNS resolver **120**, authoritative controller **110** may identify context-aware DNS record information **531-532** specifying domain name=www.xyz.com. At **840** in FIG. **4**, authoritative controller **110** may detect a potential security threat associated with VM1 **131** according to the example in FIG. **4**. In this case, a remediation action is performed to redirect to standby IP address=IP-3 associated with "healthy" VM3 **133** instead of "unhealthy" VM1 **131**.

Before allowing user device **801** to access VM3 **133**, authoritative controller **110** may identify client context information **815** to detect any potential security threat associated with user device **801**. See corresponding blocks **730-760** in FIG. **7**. To identify user device **801**, one approach is to map a source IP address (e.g., IP-U1) in DNS query **820/830** to client context information **815**. If any potential security threat is detected, access will be blocked according to block **770**. Otherwise, access is allowed by generating and sending a DNS reply to user device **801** via DNS resolver **120** according to blocks **765** and **780-785**.

Potential security threats may be detected in various scenarios. For example, the number of cache misses at DNS resolver **110** may be greater than a predetermined threshold based on DNS queries from user device **801**. In another example, user device **801** may behave abnormally at runtime, such as by querying for domain names supported by external platforms (i.e., does not satisfy its intended state). Also, geographical location information associated with user device **801** may indicate that user device **801** is operating outside of its expected location. Further, a potential security threat may be detected based on software profile information (e.g., software version, software list, jailbreak status). The detection may also involve checking whether domain name=www.xyz.com and/or IP-U1 are included in a blacklist (see **754**) or a whitelist (see **756**). If found in the blacklist, access will be automatically blocked. If not found in the whitelist, access will also be automatically blocked.

At **850** in FIG. **8**, a potential security threat associated with user device **801** is detected, such as based on jailbreak status=yes. In this case, access to standby VM3 **133** will be blocked (see **860**) and no DNS reply will be forwarded to DNS resolver **120**. Additionally, (see **772**), a notification may be generated and sent to user device management entity **160** to cause isolation of user device **801** using firewall rule(s).

Container Implementation

Although explained using VMs, it should be understood that public cloud environment **100** may include other virtual workloads, such as containers, etc. As used herein, the term "container" (also known as "container instance") is used generally to describe an application that is encapsulated with all its dependencies (e.g., binaries, libraries, etc.). In the examples in FIG. **1** to FIG. **8**, container technologies may be used to run various containers inside respective VMs **131-136**. Containers are "OS-less", meaning that they do not include any OS that could weigh 10s of Gigabytes (GB). This makes containers more lightweight, portable, efficient and suitable for delivery into an isolated OS environment. Running containers inside a VM (known as "containers-on-virtual-machine" approach) not only leverages the benefits of container technologies but also that of virtualization technologies. The containers may be executed as isolated processes inside respective VMs.

Computer System

The above examples can be implemented by hardware (including hardware logic circuitry), software or firmware or a combination thereof. The above examples may be implemented by any suitable computing device, computer system, etc. The computer system may include processor(s), memory unit(s) and physical NIC(s) that may communicate with each other via a communication bus, etc. The computer system may include a non-transitory computer-readable medium having stored thereon instructions or program code that, when executed by the processor, cause the processor to perform process(es) described herein with reference to FIG. **1** to FIG. **6**. For example, the instructions or program code, when executed by the processor of the computer system, may cause the processor to implement a "network device" to perform connectivity check according to examples of the present disclosure.

The techniques introduced above can be implemented in special-purpose hardwired circuitry, in software and/or firmware in conjunction with programmable circuitry, or in a combination thereof. Special-purpose hardwired circuitry may be in the form of, for example, one or more application-specific integrated circuits (ASICs), programmable logic devices (PLDs), field-programmable gate arrays (FPGAs), and others. The term 'processor' is to be interpreted broadly to include a processing unit, ASIC, logic unit, or programmable gate array etc.

The foregoing detailed description has set forth various embodiments of the devices and/or processes via the use of block diagrams, flowcharts, and/or examples. Insofar as such block diagrams, flowcharts, and/or examples contain one or more functions and/or operations, it will be understood by those within the art that each function and/or operation within such block diagrams, flowcharts, or examples can be implemented, individually and/or collectively, by a wide range of hardware, software, firmware, or any combination thereof.

Those skilled in the art will recognize that some aspects of the embodiments disclosed herein, in whole or in part, can be equivalently implemented in integrated circuits, as one or more computer programs running on one or more computers (e.g., as one or more programs running on one or more computing systems), as one or more programs running on one or more processors (e.g., as one or more programs running on one or more microprocessors), as firmware, or as virtually any combination thereof, and that designing the

circuitry and/or writing the code for the software and or firmware would be well within the skill of one of skill in the art in light of this disclosure.

Software and/or to implement the techniques introduced here may be stored on a non-transitory computer-readable storage medium and may be executed by one or more general-purpose or special-purpose programmable microprocessors. A "computer-readable storage medium", as the term is used herein, includes any mechanism that provides (i.e., stores and/or transmits) information in a form accessible by a machine (e.g., a computer, network device, personal digital assistant (PDA), mobile device, manufacturing tool, any device with a set of one or more processors, etc.). A computer-readable storage medium may include recordable/non recordable media (e.g., read-only memory (ROM), random access memory (RAM), magnetic disk or optical storage media, flash memory devices, etc.).

The drawings are only illustrations of an example, wherein the units or procedure shown in the drawings are not necessarily essential for implementing the present disclosure. Those skilled in the art will understand that the units in the device in the examples can be arranged in the device in the examples as described, or can be alternatively located in one or more devices different from that in the examples. The units in the examples described can be combined into one module or further divided into a plurality of sub-units.

We claim:

1. A method for a computer system to perform context-aware domain name system (DNS) query handling, wherein the method comprises:

detecting, by the computer system, a DNS query to translate a domain name, wherein the DNS query is initiated by a client requiring access to the domain name;

identifying, by the computer system, DNS record information that translates the domain name to a network address assigned to a virtualized computing instance, wherein the virtualized computing instance is deployed external to the client;

identifying, by the computer system, context information of the virtualized computing instance and mapped to the DNS record information, wherein the context information includes at least one of a health status associated with the virtualized computing instance and a runtime state and an intended state of the virtualized computing instance; and

in response to detecting a potential security threat based on the context information, performing, by the computer system, a remediation action to block access to the virtualized computing instance;

otherwise, generating and sending, by the computer system, a DNS reply specifying the network address assigned to allow access to the virtualized computing instance.

2. The method of claim **1**, wherein detecting the potential security threat comprises:

determining that the context information indicates a deviation between the runtime state and the intended state of the virtualized computing instance.

3. The method of claim **1**, wherein detecting the potential security threat comprises:

determining that the context information indicates the health status associated with the virtualized computing instance, wherein the health status is derivable based on one or more of the following: operating system (OS) kernel behavior information, process behavior information, process identifier (ID) information, network flow information, universally unique ID (UUID) information and network address information associated with the virtualized computing instance.

4. The method of claim **1**, wherein performing the remediation action comprises:

redirecting the client to a standby network address by generating and sending a DNS reply specifying the standby network address assigned to a standby virtualized computing instance.

5. The method of claim **4**, wherein performing the remediation action comprises:

translating the domain name to a standby domain name that is associated with the standby network address.

6. The method of claim **1**, wherein the method further comprises:

identifying client context information associated with the client; and

in response to detecting a potential security threat associated with the client based on the client context information, blocking the client from accessing the virtualized computing instance.

7. The method of claim **6**, wherein identifying the client context information comprises:

identifying the client context information associated with the client in the form of a user device, wherein the client context information includes one or more of the following: hardware profile information, software profile information, geographical location information and user profile information.

8. A non-transitory computer-readable storage medium that includes a set of instructions which, in response to execution by a processor of a computer system, cause the processor to perform a method of context-aware domain name system (DNS) query handling, wherein the method comprises:

detecting a DNS query to translate a domain name, wherein the DNS query is initiated by a client requiring access to the domain name;

identifying DNS record information that translates the domain name to a network address assigned to a virtualized computing instance, wherein the virtualized computing instance is deployed external to the client;

identifying context information of the virtualized computing instance and mapped to the DNS record information, wherein the context information includes at least one of a health status associated with the virtualized computing instance and a runtime state and an intended state of the virtualized computing instance; and

in response to detecting a potential security threat based on the context information, performing a remediation action to block access to the virtualized computing instance;

otherwise, generating and sending a DNS reply specifying the network address assigned to allow access to the virtualized computing instance.

9. The non-transitory computer-readable storage medium of claim **8**, wherein detecting the potential security threat comprises:

determining that the context information indicates a deviation between the runtime state and the intended state of the virtualized computing instance.

10. The non-transitory computer-readable storage medium of claim **8**, wherein detecting the potential security threat comprises:

determining that the context information indicates the health status associated with the virtualized computing

instance, wherein the health status is derivable based on one or more of the following: operating system (OS) kernel behavior information, process behavior information, process identifier (ID) information, network flow information, universally unique ID (UUID) information and network address information associated with the virtualized computing instance.

11. The non-transitory computer-readable storage medium of claim **8**, wherein performing the remediation action comprises:

redirecting the client to a standby network address by generating and sending a DNS reply specifying the standby network address assigned to a standby virtualized computing instance.

12. The non-transitory computer-readable storage medium of claim **11**, wherein performing the remediation action comprises:

translating the domain name to a standby domain name that is associated with the standby network address.

13. The non-transitory computer-readable storage medium of claim **8**, wherein the method further comprises:

identifying client context information associated with the client; and

in response to detecting a potential security threat associated with the client based on the client context information, blocking the client from accessing the virtualized computing instance.

14. The non-transitory computer-readable storage medium of claim **13**, wherein identifying the client context information comprises:

identifying the client context information associated with the client in the form of a user device, wherein the client context information includes one or more of the following: hardware profile information, software profile information, geographical location information and user profile information.

15. A computer system configured to perform context-aware domain name system (DNS) query handling, wherein the computer system comprises:

a processor; and

a non-transitory computer-readable medium having stored thereon instructions that, when executed by the processor, cause the processor to:

detect a DNS query to translate a domain name, wherein the DNS query is initiated by a client requiring access to the domain name;

identify DNS record information that translates the domain name to a network address assigned to a virtualized computing instance, wherein the virtualized computing instance is deployed external to the client;

identify context information of the virtualized computing instance and mapped to the DNS record information, wherein the context information includes at least one of a health status associated with the virtualized computing instance and a runtime state and an intended state of the virtualized computing instance; and

in response to detecting a potential security threat based on the context information, perform a remediation action to block access to the virtualized computing instance;

otherwise, generate and send a DNS reply specifying the network address assigned to allow access to the virtualized computing instance.

16. The computer system of claim **15**, wherein the instructions for detecting the potential security threat cause the processor to:

determine that the context information indicates a deviation between the runtime state and the intended state of the virtualized computing instance.

17. The computer system of claim **15**, wherein the instructions for detecting the potential security threat cause the processor to:

determine that the context information indicates the health status associated with the virtualized computing instance, wherein the health status is derivable based on one or more of the following: operating system (OS) kernel behavior information, process behavior information, process identifier (ID) information, network flow information, universally unique ID (UUID) information and network address information associated with the virtualized computing instance.

18. The computer system of claim **15**, wherein the instructions for performing the remediation action cause the processor to:

redirect the client to a standby network address by generating and sending a DNS reply specifying the standby network address assigned to a standby virtualized computing instance.

19. The computer system of claim **18**, wherein the instructions for performing the remediation action cause the processor to:

translate the domain name to a standby domain name that is associated with the standby network address.

20. The computer system of claim **15**, wherein the instructions further cause the processor to:

identify client context information associated with the client; and

in response to detecting a potential security threat associated with the client based on the client context information, block the client from accessing the virtualized computing instance.

21. The computer system of claim **20**, wherein the instructions for identifying the client context information cause the processor to:

identify the client context information associated with the client in the form of a user device, wherein the client context information includes one or more of the following: hardware profile information, software profile information, geographical location information and user profile information.

* * * * *