Software-Defined Networks: Architecture for Extended SDN Applications and Resource Optimization in Cloud Data Centers

Minh Pham

Supervisor

Professor Doan B. Hoang

A thesis submitted to Faculty of Engineering and Information Technology

University of Technology Sydney

in fulfillment of the requirements for the degree of Doctor of Philosophy

04/2020

DEDICATION

I dedicated this thesis to my parents, Bay Tan Pham and Dao Thi Nguyen, who always encourage me to study to have a better life, to my brothers Thanh Minh Pham, Xuan Tu Pham and Minh Nhut Pham for their support and encouragement me to complete my study.

ACKNOWLEDGEMENT

I sincerely express my deepest gratitude to my principal supervisor, Professor Doan B. Hoang, for his supervision and continuous encouragement, enthusiasm, technical comments, and constructive criticism throughout my whole PhD study. His full support, guidance, wisdom, and enthusiasm have made me both more mature as a person and more confident to be a good researcher. He has been outstanding in providing insightful feedback and guidance in each step of the creating of the research engagement. I admire his encouragement and determination to help me conquer all difficulties to achieve all results and to complete the thesis. Without his guidance, I would be in struggle in the research career. I feel so fortunate to have him as my supervisor for the past four and half years.

I thank Dr. Zenon Chaczko to co-supervise me. I am very thankful to him for his valuable feedback in research papers and the thesis. He is always there to help me when I need it, it does not matter it is a technical issue or other research-related issues. I believe that he is the most helpful person I ever known.

I thank the Australian Government to offer me the position of an HDR student and for all supports to help me in the study process.

I thank UTS Librarians to be so helpful in providing the books that I need during my time of studying. They are always there to help with the best books and journals when required. I always feel confident every time I go to UTS Library as I always can get the materials that I need.

I thank the SEDE School, SEDE Research and the Faculty Engineering and IT in supporting me in the research journey in the past four years. I realize that I can study in every corner of UTS as long as I feel comfortable. And UTS is a huge place where students can study to gain knowledge.

I thank Ms. Sarah Lok and the Counsellors at Student Services for their support. They are always so helpful with precious advices to support my study all the way.

I thank Dr. Marie Manidis, my GRS tutor since 2016 until now even when she did not work at UTS anymore. She is always there, very helpful in helping me out any questions that I have in my research journey.

I thank Prof. Robert Steele in giving the seeds to become a researcher to contribute to the Research Communities.

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Minh Nguyet Thi Pham, declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Electrical and Data Engineering Faculty of Engineering and IT at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise reference 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.

Signature of student: ____

Date: ____

THE AUTHOR'S PUBLICATIONS

International Conferences Publications and Proceedings

Hoang, Doan & Pham, Minh, 2015, "On software-defined networking and the design of SDN controllers", presented at International conference of Network of the Future 2015, Montreal, Sep 30 – Oct 2, 2015, as a Poster

Pham, Minh & Doan, Hoang, 2016, "SDN applications - the intent- based Northbound interface realisation for extended applications", presented at IEEE conference on Network Softwarization 2016, Seoul, June 6 - June10, 2016

Pham, Minh, Hoang, Doan & Chaczko, Zenon, 2019, "Realization of Congestion-aware Energy-aware virtual link embedding", presented at International Telecommunications Networks and Applications Conference (ITNAC), Nov 27th, 2019, Auckland, New Zealand

International Peer Review Journals

Pham, Minh, Hoang, Doan & Chaczko, Zenon, 2019, "Congestion- aware energy-aware virtual network embedding", IEEE/ACM Transactions on Networking, Vol. 28, Issue 1, Feb. 2020, pp 210-223

Pham, Minh, Hoang, Doan & Chaczko, Zenon, 2019, "Resource allocation optimization of latency aware network slicing in 5G core network", IEEE Transactions on Service and Network Management, submitted after peer reviewed

ABSTRACT

Virtualization is the main mechanism to share resources to many customers by creating virtual resources on the common physical resources. The challenge is to search for an optimal resource allocation mechanism that maximizes the capacity of the virtual resources. Network virtualization needs a new virtual network embedding (VNE) mechanism that focuses concurrently on control congestion, cost saving, energy saving; a link embedding mechanism needs to select actively based on multiple objectives the physical link resources, network slicing requires a new resource allocation mechanism that satisfies latency constraints of 5G mobile system. This research investigated and developed solutions for resource request delivery, and optimal resource allocation in network virtualization and 5G core network slicing applying SDN technology.

In the research, firstly, the three-tier architecture applying micro-service architecture for extended SDN application is presented to facilitate the flexibility, in which new services are created or composed, existing services are reused. The evaluation is the prototype of the Dynamic resource allocation using the proposed architecture.

Secondly, the multiple-objective VNE that focuses on congestion avoidance, energy saving and cost saving (CEVNE) is presented. The novelty lies in the CEVNE mathematical model for multiple-objective optimization problems, and its nodes and link embedding algorithms. The evaluation showed that CEVNE outperformed The-State-Of-The-Art in acceptance ratio in the challenged, near-congestion scenarios.

Thirdly, the architecture to realize virtual link mapping in CEVNE is presented. The novelty is in the SDN-based heuristic algorithm, and the applying of the architecture for extended SDN applications. The research results in the realization of the active virtual link embedding process that focuses on multi-objective concurrently. The evaluation showed that the solution outperformed the traditional link mapping in all three objectives.

Fourthly, the mathematical model of the resource allocation optimization in latency-aware 5G core network slicing is presented. The novelties lie in the satisfaction of different latency requirements of 5G applications: eMBB, uRLLC, and mMTC, and the solution strategy to linearize, convex-relax and decompose the program into sub-problems. The evaluation shows that the solution outperformed the The-State-Of-The-Art in resource allocation, execution time, latency satisfaction and the arrival rates.

In this thesis, the resource optimization problem and the architecture for extended SDN applications have been studied comprehensively. The results of this thesis can readily be applied to 5G vertical applications where resource optimization and network routing problems exist naturally in multiple domains and require software defined networking logically centralized control architecture for efficient and dynamic solutions.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
ARP	Address resolution protocol
ARS	Approximate resource scarcity
ASR	Architecturally significant requirements
BPM	Business process management
C2X	Car to everything
CAM/ TCAM	Content addressable memory / Ternary content addressable memory
CAPEX	Capital expenditure
CEVNE	Congestion-aware energy-aware virtual network embedding
CEVNE NoM	Congestion-aware energy-aware virtual network embedding node embedding
CEVNE LiM	Congestion-aware energy-aware virtual network embedding link embedding
CL	Cloud computing
CLI	Command line interface
COS	Commercial off the shelf
CP	Control plane
CTMC	Continuous time Markov Chain
CUPS	Control plane user plane separation
DC	Data center
DDD	Domain driven design
DHCP	Dynamic host control protocol
E2E	End-to-end
ECMP	Equal cost multi-path
eMBB	Enhanced mobile broadband
eNB	The base station of the mobile system.
EPC	Evolved packet core
ICMP	Internet control message protocol
INT	In-band network telemetry
IoT	Internet of thing
IP	Internet protocol

ITU	International Telecommunication Unit
KPI	Key performance Indicators
LAN	Local area network
LiM	Link mapping
LP	Linear program
MILP	Mixed-integer linear program
MINLP	Mixed-integer non-linear program
MIQCP	mixed-integer quadratic constraint program
ML	Machine learning
MMPP	Markov-modulated Poisson process
mMTC	Massive machine type communication
MP	Mathematical program
MPLS	Multiprotocol label switching
MSA	Micro-service architecture
NBI	Northbound application interface
NEF	Network exposure function
NFV	Network function virtualization
NGPaaS	Next generation platform as a service
NIC	Network interface cards
NLP	Non-linear program
NoM	Node mapping
NOS	Network operating system
NRF	Network repository function
NS	Network slice
NSR	Network slice request
NSSF	Network slice selection function
NV	Network virtualization
OF	Openflow protocol
OPEX	Operational expenditure
PaaS	Platform as a service
PCE	Path computation element
PGW	Package data gateway
QoE	Quality of experience
QoS	Quality of service

RAN	Radio access network
RDB	Resource database
REST	Representational state transfer
RFB	Reusable function block
RPC	Remote procedure call
SBA	Service-based architecture
SBI	Southbound application interface
SGW	Service gateway
SID	Segment identifier
SLA	Service level agreement
SN	Substrate network
SOA	Service-Oriented Architecture
SOAP	Service-Oriented Architecture protocol
SPG	Shortest path graph
SR	Segment routing
SRA	Segment routing application
SRH	Segment routing header
TCP	Transport control protocol
TDF	Traffic detection function
TE	Traffic engineering
UDM	Unified Data Management
UDP	User Datagram protocol
UDSF	Unstructured data storage function
UP	User (data) plane
uRLLC	Ultra-reliable low latency communication
V2X	Vehicle to everything
VM	Virtual machine
VN	Virtual network
VNE	Virtual network embedding
VNF	Virtual network function
VNR	Virtual network request

CHAPTER 1 INTRODUCTION

The Introduction chapter roadmap consists of the thesis motivation, research issues, the aims, the research questions, research objectives and scope, research contributions, research significance, the methodology overview, and the outlines of the thesis.

1.1 Motivation and Research Issues

The emerging Cloud computing paradigm facilitates computing resources to be leased to the public in the economies of scale (Jennings & Stadler 2014). Computing resources become a commodity such as electricity, gas, water, which is enabled by virtualization technology (Marinescu 2018). Similarly, network virtualization (NV) leases networking resources, which are switches, routers, physical links and bandwidth as isolated virtual networks (VN) to network service providers, academia researchers, and end users (Y.3011 Jan 2012). Users send requests for network resources to service providers; their requests are provisioned as a network application on top of the Software-defined networks (SDN). Currently, network requests are provisioned in an ad-hoc, self-explored way; they do not add value for reuse, which leads to 'reinventing the wheel'. The key research issue is the lack of a mechanism that allows network developers to concentrate efforts on application development in a systematic way, that encourages modular designs, and facilitates software reuse. As SDN introduced programmability to network applications, the mechanism needs to be based on sound architecture principles and design patterns so it can support service composition, service creation, service decomposition in the process of provisioning a user's request.

NV has been shown to be significant, while it resolves critical issues of traditional virtual networks (VPN, virtual LAN), NV also allows the coexistence of multiple VNs that have different topologies and protocols, and facilitates evolving as researchers can test new protocols on the same physical networks (Y.3011 Jan 2012). NV consists of two processes: node embedding and link embedding. The-State-of-the-Art Virtual Network Embedding (VNE) almost focuses on a single objective[1],

[1] Please see section 2.2.4 for related work that are multi-objective VNE

which may have negative impacts on other desired optimal network operations such as congestion control, energy consumption, quality of service (QoS). To maintain optimal performance, operators regularly carry out network reconfigurations, which are expensive, and interfere with activities of all VNs. The key issue is the lack of VNE focusing on multiple objectives: congestion aware, energy aware, and cost saving, which consider for both VNE and optimal operation in the substrate network to avoid regular reconfiguration.

The VNE problem is modeled as a mathematical program (MP) that is solved by an optimization solver. The results by solving the MPs are used to embed virtual nodes on to substrate nodes; the link embedding process is completely passive, in traditional networks, virtual links are embedded on the shortest paths connecting embedded substrate nodes (Fischer et al. 2013). This leads to the over-utilization of substrate links and paths in the shortest paths, but under-utilization elsewhere, which results in both network congestion, and waste of link resources. Current VNE could not detect network congestion in real-time caused by elephant flows that require large bandwidth for a long duration (Vahdat 2015), (Moorthy 2016), or hash collision which leads to network congestion (Katta et al. 2017). The challenge in the virtual link embedding is the lack of an active link embedding process that can focus on multiple objectives: congestion aware, energy aware, and cost saving as the node embedding process does, can detect elephant flows, can steer network traffic at the flow level so that it ensures both efficient virtual link embedding and optimal network operation.

Network slicing is the new 5G mechanism that provisions multiple logical E2E, self-contained virtual networks to deliver mobile services to customers concurrently. 5G promises a revolution in customer services with emerging vertical, distributed applications in eHealth, vehicle-to-anything, natural disaster, robotics and factories (Dohler 2019). 5G technology needs to combine the challenge of accurate response time (1ms), accurate location in mobile technology with the specific requirements of vertical-industry related functions, the control of the remote robotic surgery application in eHealth application, the speedy interaction between vehicles and objects in real-time (with vehicle speed up to 500km/h) in vehicle-to-everything (V2X) application (Dohler 2019), to maintain the communication of mobile services in natural disasters (earthquakes, fire storms, tsunamis, floods, hurricanes) to facilitate basic voice, text messages so users are able to signal their presence and their location in lifeline communication, the large number of mobile connections, large data volume, high data speed in crowded events (TR_22.891 Sep 2016). The comparison of 5G key performance indicators (KPIs) to 4G shows that connection density is 1000 times more, mobile data volume is 1000 times more, data rates are 10 to 20 times more, latency is one tenth, energy consumption is one tenth. 5G KPIs facilitate the above emerging services.

Network slicing (NS) is designed specifically in 5G to implement each service. NS enabling technologies are SDN, NFV and cloud computing. NS is an emerging research area that attracts attention of academia and industry, and profound solutions have been found in project NECOS (2019), 5GEx (2020). Nevertheless, the challenge of NS is the lack of a new resource allocation mechanism that satisfies constraints of the 5G stringent latency requirement based on three types of applications: eMBB, uRLLC and mMTC, constraints of the resource optimization in NV, constraints of the resource optimization and placement in NFV. Therefore, the issues of existing approaches are summarised as follows.

- ✓ Not sufficient to provision network resource requests in an SDN network as it lacks a mechanism for network applications to be developed in a systematic way; a new approach allowing developers to focus on the network application development is needed,
- ✓ Not sufficient to embed virtual networks for optimal production operation as it lacks a VNE that satisfies several production constraints in network operation; a new VNE is needed that concentrates on multiple objectives in both node embedding and link embedding processes,
- ✓ Not sufficient in network slicing as it lacks a mechanism that focuses on both resource optimization and latency satisfaction caused by VNFs processing delay; a new network slicing mechanism is needed that focuses on optimizing the resource allocation and satisfies different latency requirements of 5G applications.

Hence, this research is aiming to investigate and propose an architecture for extended SDN applications, a VNE solution based on SDN for optimal network operation, and a 5G network slicing research for optimal resource and latency satisfaction.

1.2 Research aims and Research questions

1.2.1 Research aims

The primary focus of this research is to investigate and to develop solutions for resource request provisioning, resource optimization in network virtualization and 5G core network slicing applying SDN technology.

1.2.2 Research questions

The research questions of this thesis are as follows:

Question 1: Given users requesting network resources, what is needed in an SDN-based network for the service providers to provision users' networking requests in a systematic way?

Question 2: Given the network virtualization problem, what would it take to provide an SDN-based solution in optimized resource allocation?

Question 3:

Given the network virtualization problem in cloud data centers with the two separate processes: node embedding and link embedding, can SDN technology itself be used to provide heuristic solutions for the virtual link embedding and implement the solutions dynamically and efficiently?

Question 4:

Given 5G core services come in various shapes and sizes with different stringent requirements, how can one model the network slicing concept and what would be the solution for resource optimization and latency constraint satisfaction?

Question 5:

Given the network virtualization problem in SDN networks, and the network services in 5G core systems, do the multiple-objective optimization technique and the decomposition technique of large, structure systems contribute to efficient solution approaches?

1.3 Research objectives and scope

The research objectives (RO) are identified to address the above research questions and are presented as follows.

1.3.1 Research objective 1

RO1 is to investigate and to propose an architecture for extended SDN applications that forms the basis to build composite services on top of the SDN controller. The architecture is based on three-tier architecture, micro-service architecture, micro-service design patterns: service registration, service discovery, and service composition; it takes advantage of the service model for service provider in SDN architecture (TR-521 2016) to ensure the orchestration and integration in SDN vertical applications, SDN business processes, and workflows.

In this research objective, there are subtasks as follows.

- To investigate the requirements of the architecture for extended SDN applications.
- To investigate and propose the technologies that the architecture will apply to meet the above requirements.
- To present the technologies and their main characteristics of the proposed architecture.
- To investigate the effect of the resource and service views of SDN architecture (TR-521 2016) on the architecture of extended SDN applications.
- To investigate how the controller NBI facilitates the design of the proposed architecture.
- To investigate the effect of the SDN controller platform on the proposed architecture.
- To prototype the proposed architecture in an SDN extended application.

1.3.2 Research objective 2

RO2 is to investigate and to propose a mathematical model for the virtual network embedding (VNE) in network virtualization that focuses on multiple objectives: saving cost, saving energy and avoiding congestion concurrently. As solving the VNE is NP-Hard, the heuristic algorithms will be designed for the node embedding and link embedding processes to find close-to-optimal results.

In this research objective, there are subtasks as follows.

- To investigate the main characteristics of the VNE focusing concurrently on multiple objectives: cost saving, energy saving and congestion avoidance (CEVNE).
- To investigate the mechanism to implement each objective, and to formulate each objective.
- To investigate the theory of multi-objective programming in Operation Research.
- To build a CEVNE mathematical model using the weighting and constraint methods of the multi-objective optimization technique, and to obey the rules to generate pareto-optimal solutions for all objectives.
- To investigate and identify the main characteristics of the mathematical program (MP), to investigate the solution strategy to resolve MP.
- To propose an optimization solver to resolve the above MP in small scale and large scale.
- To investigate and propose a mechanism for large scale networks.
- To evaluate the performance of the mechanism in the scarce resource, challenged scenarios, and normal scenarios with The-State-of-the-Art.

1.3.3 Research objective 3

RO3 is to investigate and to propose a virtual link embedding process, which will be active, be able to focus on multiple objectives: cost saving, energy saving and congestion avoidance as the node embedding process does. The mechanism is based on SDN and SR technologies. The realization of the mechanism is based on the architecture of the RO1 as an extended SDN application that integrates services implementing each objective, uses a monitoring function in real-time, is deployed on the SDN substrate network that runs SR as the source routing mechanism to forward packets.

In this research objective, there are subtasks as follows.

- To investigate the mechanism that will be used to search for an optimal solution in the virtual link embedding optimization problem focusing on multi-objectives.
- To identify the motivation for a heuristic virtual link embedding algorithm.
- To investigate the mechanism to realize the solution of the multi-objective virtual link embedding.
- To investigate the mechanism to implement each objective: cost saving, energy saving and congestion avoidance, and to integrate these objectives to get the result.
- To investigate the mechanism that monitors the states of the networks in real-time to ensure the proposed solution provides optimal network operation.
- To realize the CEVNE link embedding process as an SDN application on the SDN leaf-spine fabric on ONOS controller running the SR application.
- To evaluate the CEVNE link embedding realization with The-State-of-the-Art.

1.3.4 Research objective 4

RO4 is to investigate and to propose a mathematical model for resource optimization in network slicing satisfying latency requirement of different 5G application types: eMBB, uRLLC and mMTC, applying SDN, NFV technologies, and cloud computing. We only investigate the 5G core network slicing. The mechanism focuses on optimizing the resource allocation and satisfying the critical latency requirement of different types of 5G service. The research builds the resource model and performance model of network slicing in order to build its mathematical model. The network slicing problem focuses on two objectives: resource optimization and latency satisfaction; its program is considered as a structure system. The program is non-convex, non-linear, mixed-integer, and NP-Hard. The solution strategy focuses on the convex relaxation, linearization, and the decomposition technique for structure systems in optimization problem. The MP is decomposed into the master model and two sub-models: VNF embedding and virtual link embedding, which are solved using the IBM OPL flow control mechanism and scripts.

In this research objective, there are subtasks as follows.

- To investigate and identify different models to progress from network slicing theory concepts to a comprehensive solution based on scientific evidence.
- To investigate and propose a performance model for network slicing with its main characteristics that will lead to a concrete formulation.
- To investigate and define the network slicing problem and to formulate the problem.
- To investigate and identify the main characteristics of the mathematical model.
- To investigate solution strategies that can be applied to the mathematical model that leads to a solvable program.
- To investigate the decomposition techniques in the optimization of structure systems using the master model and sub-models.
- To investigate the optimization solver that will be applied to solvable program; specifically, the flow control mechanism in IBM OLP as the decomposition mechanism.
- To apply the optimization solver to the network slicing problem to get the optimal solution.
- To evaluate the mechanism with The-State-of-the-Art.

1.4 Contributions of Thesis

The research contributions are mainly in innovative solutions as follows:

A novel architecture to realize extended SDN applications on top of the SDN controllers. It is based on Micro-service architecture (MSA), and MS design patterns: service registration, service discovery, service composition, service creation and service reuse. It is a three-tier architecture with the database tier, business tier and web tier. The database tier ensures that the extended application can have its own database. The business tier consists of micro-services (MS) applying MS design patterns. The web tier ensures the application can be accessed via the browser. The architecture focuses on the composability attribute of extended applications. It allows the creation of new services, the reuse of existing services, and the composition of a service based on new and existing services. It is based on the domain driven design (DDD) principle, in which the main problem is divided into sub-problem domains, and the solution is resolved in each sub-domain. The architecture ensures SDN applications are integrated into vertical applications, workflows or business processes based on the programmability of the SDN paradigm. The prototype showed that the proposed architecture is appropriate, effective, realizable, and practical for emerging applications.

A novel mathematical model and its algorithms for the VNE that focuses on multiple objectives: saving cost, saving energy and avoiding congestion in network virtualization in cloud data centers (CEVNE). The cost saving objective tries to ensure minimal substrate resource allocation to virtual networks; the energy saving objective ensures a minimal number of active nodes, which are substrate nodes that the virtual nodes are embedded on; the congestion avoidance objective spreads traffic into different paths to minimize the maximum link utilization. The mathematical program (MP) applies the weighting methods for the cost saving and energy saving with positive weighs, and the normalization to build the CEVNE objective; the constraint method is applied to the congestion avoidance using binding constraints; these will generate the pareto-optimal solutions and the best compromised solutions for all three objectives. CEVNE mathematical model is a mixed-integer linear program (MILP), which is relaxed into a linear program, and solved by GLPK. As solving VNE is NP-Hard, heuristic algorithms are designed for CEVNE with node embedding, link embedding and integration algorithms. The node embedding algorithm is designed to convert the linear results back to the approximation of integer variables of the sub-optimal solution. The evaluation showed that the CEVNE outperformed The-State-of-the-Art in the challenging scenarios where virtual networks have very high CPU and bandwidth demands that may cause congestion to the substrate networks. In overall evaluation, the results show that CEVNE succeeds in delivering concurrently multiple objectives: cost saving, energy saving and congestion avoidance.

An SDN-based heuristic algorithm for the CEVNE link embedding process is proposed on an SDN substrate network that focuses on three objectives of saving cost, saving energy and avoiding congestion concurrently. The heuristic solution is realized as a composite SDN application using the architecture for extended SDN application that integrates ONOS core services, SR application, monitoring application, and new MS that are designed for each objective. The application is based on the three-tier architecture, MSA, MS design patterns and DDD principles. It allows the bandwidth resources to be allocated dynamically to virtual links, which can be further exploited to the area of bandwidth allocation on demand. The CEVNE LiM application invokes the path selection service to choose the substrate path that satisfies multiple objectives. The substrate network is an SDN leaf-spine fabric running SR on an SDN controller. The evaluation showed that the CEVNE LiM application outperformed the k-shortest path in cost saving, energy saving and congestion avoidance; it is an active process and is flexible in delivering results.

As network slicing is a new and difficult concept, a novel solution approach is presented for the network slicing in 5G core networks. Different models are built to understand how network slicing operates. Resource model is used to allocate resources to VNFs and to virtual links connecting VNFs. Performance model using queueing theory is used to predict network slicing end-to-end latency. The VNF is modeled as M_t/G/1 using queueing theory with the Markov-modulated Poisson process (MMPP) is applied to the input queue of VNF. The linear function of the processing latency on CPU resource allocation is applied to ensure that the allocated resources satisfy the latency threshold. The network slicing problem is modelled using mathematical modelling into an MP. An MP focuses on two objectives: to minimize resource allocation and to satisfy critical latency thresholds of 5G applications in different types: eMBB, uRLLC and mMTC. The MP is non-convex, non-linear, mixed-integer and NP-Hard. The novelty of the solution is also in the solution strategy, which focuses on convex relaxation using linearization and decomposition. The MP is decomposed into the master model and two sub-models: VNF embedding and virtual link embedding algorithms; the separation will generate the solution for the original network slicing problem. The master model and its sub-models are solved using IBM OPL scripts and the flow control mechanism. The evaluation showed that the algorithms outperformed The-State-of-the-Art in cost saving, acceptance ratio, execution time, latency satisfaction.

1.5 The significance of research

The significance of the research is presented as follows.

The new architecture based on MSA, MS design patterns can be applied to custom 5G core services or vertical applications as 5G core architecture duplicates SDN architecture as follows: 5G network exposure function (NEF) plays the role of SDN NBI, 5G network repository function (NRF) provides service registration and discovery; and 5G core services are required by its vertical applications in the same way that SDN core services are required by SDN applications.

The new mathematical model of multi-objective CEVNE and the SDN-based heuristic link embedding solutions can be applied to any networking technologies that require resource optimization and network routing via dynamically allocated bandwidth resources such as NFV, SFC, and network slicing. In these applications, the main problems are the VNF resource optimization and placement and the routing between VNFs; they may operate in the distributed

SDN architecture, hence, the satisfaction of the requirement of the multi-domain environment can be considered as an additional objective.

For structure systems, the novel mathematical model and the innovative solution approaches using decomposition can be applied to the multi-objective optimization problem as it has common attributes with the structure systems. As the structure and multi-objective optimization models result in better decision-making than the single objective model, but they become more complex to solve, the decomposition solution will decrease the complexity, simplify the system structures and reduce the time to solve for the solutions. All the structure and multi-objective systems and decomposition technique can be applied to any networking optimization problems.

The realization of CEVNE virtual link embedding using the architecture for SDN applications can be applied in the network applications that are completely different from NV such as in the bandwidth on demand, the aggregation of bandwidth, or the bandwidth on demand and schedule.

All the solutions in the thesis can be applied to 5G custom services and 5G vertical applications as they are based on 5G core services; therefore, they can reuse the models of multi-objective and structure systems, the decomposition technique and the architecture for extended SDN applications as long as 5G core architecture duplicates the SDN architecture.

1.6 Methodology overview

The main steps in the methodology development consists of initialization, methodology development, performance evaluation and conclusion, which are presented in Figure 1.2. In the initialization step, the research topic is decided via the research motivation and key issues. The hypothesis, research questions and the research objectives are developed. Methodology development consists of the methodology investigation, problem definition, problem statement, problem formulation, solution approach design and solution realization. Performance evaluation consists of the evaluation setup, test scenario design and result analysis discussion. Conclusion includes the discussion and future work. The methodology development and performance evaluation are working in a loop, the evaluation result is used to refine the methodology development until the research objectives are satisfied.

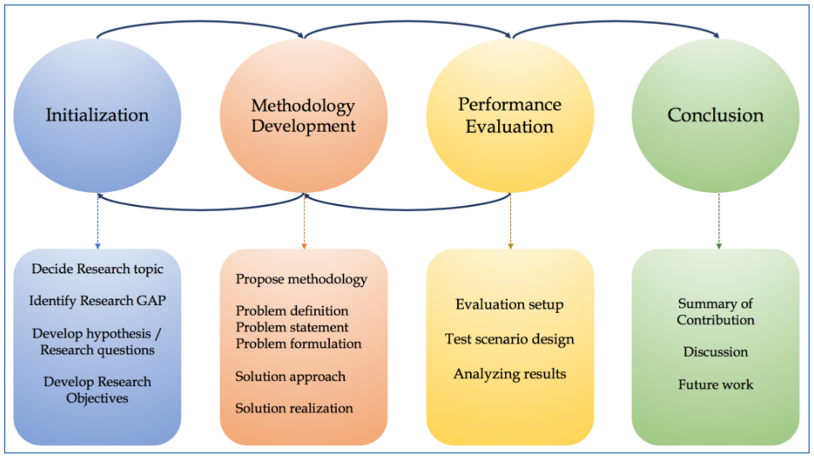


Figure 1.1: Steps in the development of Research methodology

The methodology in Figure 1.1 fits into the two blocks "*Basic Research*" and "Concept formulation" of the research model presented in Figure 1.2 (Bernardo 2012), in which the research gap is identified, the methodology is proposed, the evaluation shows the contribution is significant, and the resulting paper is presented to a conference, or sent to an international journal. In later steps, it is checked for the usable capacity. It is sent for outsider usage before it is prepared to be commercialized as presented in Figure 1.2.

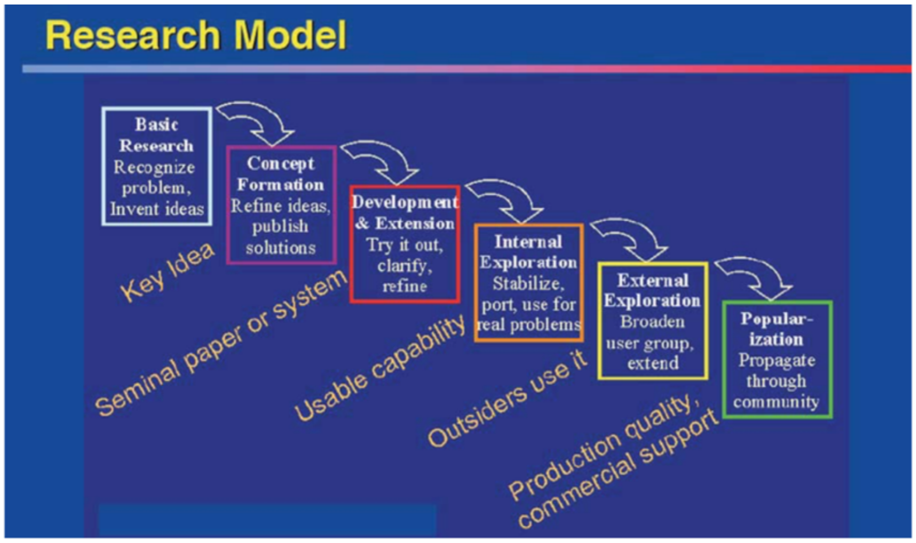

Research Model
Basic Research Recognize problem, Invent ideas
Concept Formation Refine ideas, publish solutions
Development & Extension Try it out, clarify, refine
Internal Exploration Stabilize, port, use for real problems
External Exploration Broaden user group, extend
Popular-ization Propagate through community
Key Idea
Seminal paper or system
Usable capability
Outsiders use it
Production quality, commercial support

Figure 1.2: Research model adapted (Bernardo 2012)

1.7 Thesis outlines

The thesis consists of seven chapters, which is presented in Figure 1.3. Each chapter is summarized as follows.

Chapter 1 - Introduction.

Chapter 1 outlines the context of the research area with the emerging SDN technology, the issues in the network virtualization and network slicing in 5G core networks. The motivation and key research issues are identified. The research aims, research questions, research objectives and scopes are presented. The significance of the research and the research contributions are presented. The methodology overview is explained in detail. The research outline of the thesis is depicted with seven chapters.

Chapter 2 – Background theory and related work

This chapter presents the background theory of the SDN networking paradigm, micro-services, micro-service architecture and design patterns, segment routing applied in SDN networking, the theory about network virtualization, service function chains, network slicing, mathematical modeling, queueing theory, P4 and in-band network telemetry. Related work to the research works in the thesis is also presented.

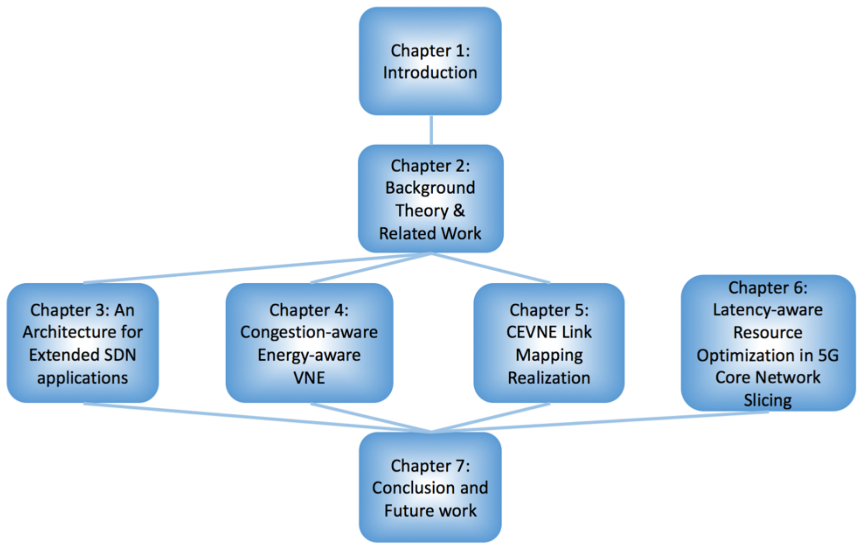

Chapter 1: Introduction
Chapter 2: Background Theory & Related Work
Chapter 3: An Architecture for Extended SDN applications
Chapter 4: Congestion-aware Energy-aware VNE
Chapter 5: CEVNE Link Mapping Realization
Chapter 6: Latency-aware Resource Optimization in 5G Core Network Slicing
Chapter 7: Conclusion and Future work

Figure 1.3: Thesis outline

Chapter 3 – An architecture for extended SDN applications

The chapter proposes an architecture for extended SDN applications and services. The investigation identifies the requirements of the architecture that include the composability, the creation of new services, the reuse of existing services and the integration of new and existing services in the composite service. The proposed architecture applies micro-service architecture (MSA) and MS design patterns, three-tier application architecture, and domain driven design principles. The novelty of the solution lies in the three-tier architecture applying MSA and MS design patterns.

Chapter 4 – Congestion-aware Energy-aware Virtual Network Embedding

The chapter proposes a multiple-objective virtual network embedding (VNE) problem called congestion-aware energy-aware VNE (CEVNE) that focuses on saving cost, saving energy and avoiding congestion. The investigation defines the CEVNE problem in the substrate networks, virtual networks, virtual network requests, the VNE problem with two processes: node embedding and link embedding, and the augmented substrate network. CEVNE is modelled using the weighting and constraint methods to combine three objectives into the CEVNE program obeying the rules to generate pareto-optimal solutions for involved objectives. CEVNE is a MILP; solving MILP is computationally intractable, hence, CEVNE is relaxed to LP. As solving the VNE is NP-

Hard, the heuristic node embedding and link embedding algorithms are designed to solve CEVNE in polynomial time. The node embedding algorithm applies the rounding function to find the approximation of the integer variables in the sub-optimal solutions. The novelty of the solution lies in the mathematical model, the application of the rules to generate pareto-optimal solutions for involved objectives, and the heuristic algorithms for the node and link embedding processes.

Chapter 5 – Virtual link embedding process in CEVNE

The chapter proposes an SDN-based heuristic algorithm for the virtual link embedding process that is active, and focuses on multi-objectives: cost saving, energy saving and congestion avoidance as the node embedding process does. The realization of the heuristic algorithm applies the architecture for extended SDN applications, which supports MSA and MS design patterns, and the three-tier architecture for SDN composite applications. Each objective of cost saving, energy saving, and congestion avoidance is realized as micro-services. The path selection algorithm is proposed as a new service, which selects the substrate path based on the results of services implementing each objective. The novelty of the solution lies in the SDN based heuristic algorithm, and its realization as an SDN application based on a three-tier architecture on the SDN leaf-spine fabric substrate network running segment routing.

Chapter 6 – Latency-aware resource optimization for 5G core network slicing

The chapter proposes the solution for the resource optimization in the latency-aware network slicing problem in a 5G core network. The network slicing problem focuses on two objectives: resource optimization and latency satisfaction for different types of 5G applications: eMBB, uRLLC and mMTC. The novelty of the solution lies in the building of the operational model, resource model and performance model in order to formulate the problem using mathematical programming. The network slicing program is considered a structure system. The program is NP-Hard, non-convex, mixed-integer and nonlinear (MINLP). The novelty of the solution lies in the convex relaxation, linearization and decomposition of the MP into the master model and two sub-models: VNF embedding and virtual link embedding, which are solved using IBM OPL flow control mechanism and scripts. The decomposition will generate the sub-optimal solutions for the original network slicing problem.

Chapter 7 – Conclusions and Future work

The chapter summarizes the research aims, research outcomes, research contributions and research significance through the results of the solutions in Chapter 3, Chapter 4, Chapter 5 and Chapter 6.

Future works are identified based on the research so far in different research areas: architecture for extended SDN applications, network virtualization, network function virtualization and network slicing. The chapter concludes that the thesis has achieved the aims and objectives that are presented in chapter 1 Introduction.

1.8 Summary

This chapter identifies the research motivation and key issues, research aims, research questions, research objectives and scope. The chapter describes the research significance with its contributions to the field of SDN applications, network virtualization, network slicing in 5G core networks. The thesis outline is presented. Chapter 2 - Literature review and related work is presented next.

CHAPTER 2 LITERATURE REVIEW AND RELATED WORK

In the previous chapter, we introduced the thesis, and presented the research motivation, key research issues, research objectives, research aims, research significance and research contributions.

This chapter presents the background theory and related work that are related to our solutions in the research areas of interest. The related work sections are titled starting with "Related work" and are included in the related background theory section. Firstly, the theory background is presented with details of SDN, SDN architecture, SDN controller, SDN switches, SDN applications, segment routing V6, network virtualization (NV), network function virtualization (NFV), 5G and network slicing (NS), mathematical model to solve optimization problems in operations research, queueing theory for performance prediction, and P4 language, and the In-band network monitoring application (INT). Secondly, we review current solutions in the research areas of interests as follows.

- The multiple-objective VNE that has been investigated in the research community,
- The active virtual link embedding in NV,
- 5G network slicing in mobile core systems that have been investigated in the research community utilizing utilities theory, game theory and modelling techniques.

The roadmap of chapter 2 is presented in Figure 2.1 that consists of the background theory and related work.

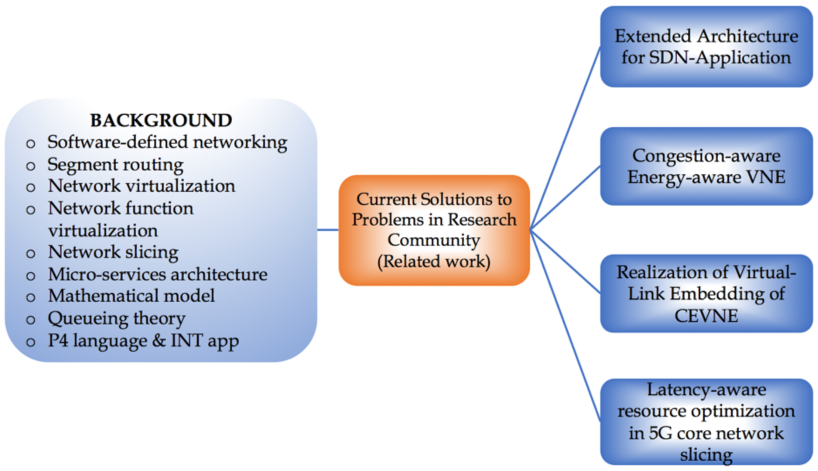

BACKGROUND
Software-defined networking
Segment routing
Network virtualization
Network function virtualization
Network slicing
Micro-services architecture
Mathematical model
Queueing theory
P4 language & INT app
Current Solutions to Problems in Research Community (Related work)
Extended Architecture for SDN-Application
Congestion-aware Energy-aware VNE
Realization of Virtual-Link Embedding of CEVNE
Latency-aware resource optimization in 5G core network slicing

Figure 2.1: Chapter 2 Literature review and roadmap

2.1 Software-defined networking

SDN paradigm defines a new way of networking in computer systems. SDN technology consists of many related concepts, only concepts that are related to the thesis are presented as shown in Figure 2.2. They consist of SDN network paradigm definition, SDN main components comprising SDN architecture, SDN controller, SDN switches, and SDN applications. These form the basis theory to comprehend SDN technology, chapter 3 named Architecture for extended SDN applications, chapter 4 named Congestion-aware Energy-aware VNE, chapter 5 named Realization of CEVNE virtual link mapping, and chapter 6 named Latency-aware resource optimization for 5G core network slicing.

Chapter 3 extends the SDN application layer into a three-tier architecture applying MS architecture and design patterns for extended SDN applications. The theory of SDN architecture is important to understand how SDN applications can manipulate the traffic flow in network devices via Northbound API and Southbound API. The SDN programmability, and SDN ability to steer network flows at the coarse-grained and fine-grained levels are important for the contribution of the listed chapter 4, chapter 5, and chapter 6. Mininet is an SDN simulation tool to evaluation solutions in the thesis.

2.1.1 What is SDN paradigm?

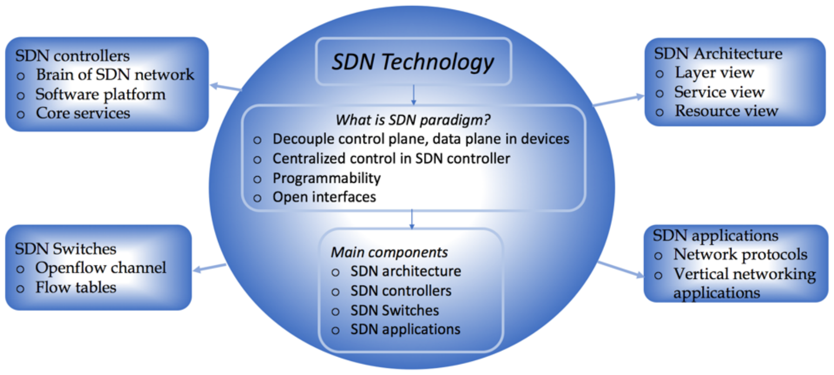


Figure 2.2: SDN technology roadmap

Software-Defined Network (SDN) is a network technology that separates the control plane from the data plane in network devices, centralizes control in the controller, adds programmability to network operations, and opens interfaces so everyone can contribute. This leads to major benefits to network providers, network consumers, cloud data centers, network operations, and the design of new network protocols, new network hardware as follows.

- Simplified network devices as they only forward traffic according to their look-up flow table,
- Programmability as developers can program the behaviour of the network by loading policies on network devices,
- Autonomous device configuration using commands on the controller,
- Network virtualisation and the offer of networks on demand based on central control and programmability,
- Network services and applications that can be supported through their interfaces, southbound interface (SBI) for devices and northbound interface (NBI) for applications (Hoang & Pham 2015).

SDN network paradigm is a new technology (Hoang 2015) as SDN is beneficial for researchers, network operators, service providers, and has triggered a wave of changes in most areas of computer networking (Duan & Toy 2017), (Kreutz et al. 2015). The changes are significant in the direction of saving capital expenditure (CAPEX) via encouraging commercial-off the shelf (COS) devices; saving operation expenditure (OPEX) via the flexibility, automation to daily network operations (Goransson, Black & Culver 2016), simplifying network operation by reducing time and effort via programmability, central control, and central view.

He et al. (2019) conducted a survey about flexibility in dynamic network using SDN technology. The flows and resources are elements that SDN offers flexibility in communication networks. A flow is defined with a source node, a destination node and data rates; resources, which consist of computation power, memory and bandwidth, are mandatory in network operations. SDN can steer network flows, offer bandwidth on demand, and monitor the network in real-time. Chapter 4, and chapter 5 of the thesis show the flexibility that SDN brings to dynamic networks. SDN also contributes to the automation of network configuration using YANG, NetConf and RestConf (Claise, Clarke & Lindblad 2019). The proposed architecture in chapter 3 touches on the automation process to provision SDN network request.

2.1.2 SDN architecture

2.1.2.1 Layer model

SDN architecture is depicted with three layers: the infrastructure layer, the control layer, and the application layer.

The infrastructure layer consists of SDN devices, which are physical / virtual switches, or routers. They operate based on Openflow (OF) protocol. Their main tasks are packet forwarding. Each SDN device has an interface to connect to the controller southbound interface (SBI), several OF flow tables that contain flow entries, and the hardware component to process packets.

The controller layer consists of one or more controllers operating in the cluster mode. The controllers are the SDN network brain. Each controller provides core services to manage devices, flows, topology… of the SDN network underneath. Controllers connect to devices via SBI, connect to applications via NBI, connect to other controllers in the same cluster using the master and slave mechanism (ONOS 2019b). Controllers provide a central view of all SDN devices to the applications via core network services.

The application layer consists of network applications that are built on top of SDN controllers. They connect to the controllers via the NBI. They operate on the underlying network of SDN devices using the core services provided by the control layer. The list of applications is unprecedented as it expands to different areas of networking technologies. Figure 2.3 presents the SDN high level architecture in the layer views: the infrastructure layer, the control layer, and the application layer.

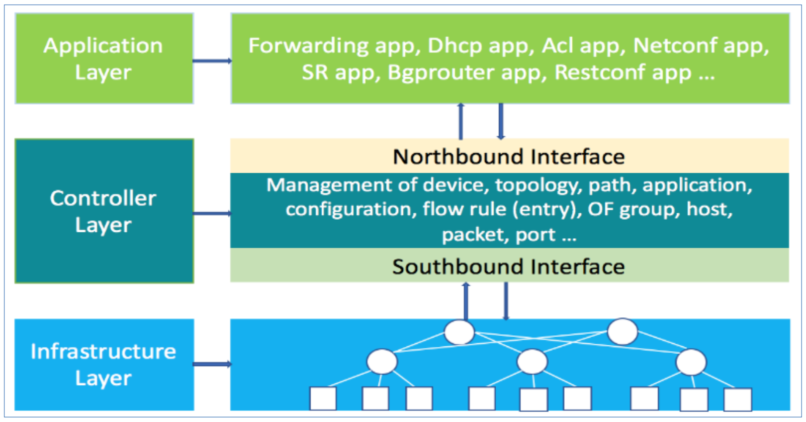

Application Layer
Forwarding app, Dhcp app, Acl app, Netconf app, SR app, Bgprouter app, Restconf app ...
Controller Layer
Northbound Interface
Management of device, topology, path, application, configuration, flow rule (entry), OF group, host, packet, port ...
Southbound Interface
Infrastructure Layer

Figure 2.3: SDN architecture in layer view

2.1.2.2 Service model

TR-521 (2016) presented the service model of SDN architecture. The service model is the top-down view of an SDN customer that sends requests to SDN controller. In the architecture service model, customers in the application layer are the service consumer (client), SDN controllers are service providers. Customers send requests to the controller for some applications of network operations on the SDN networks. At receiving the clients' requests, SDN controller checks them against its policies and its resources. If requests are feasible, the controller provision them based on an optimization model. A request is rejected if it is infeasible; otherwise the controller will send the response to the client. The service view is presented in Figure 2.4.

2.1.2.3 Resource model

The architecture resource-oriented view is all about the devices in the infrastructure layer, it is the bottom-up view of SDN devices according to an SDN network operator. The resource view is used to explain the provisioning of requests in the service view. A network resource may be physical or virtual, active or passive, may be created, scaled or removed by operators or by customer requests.

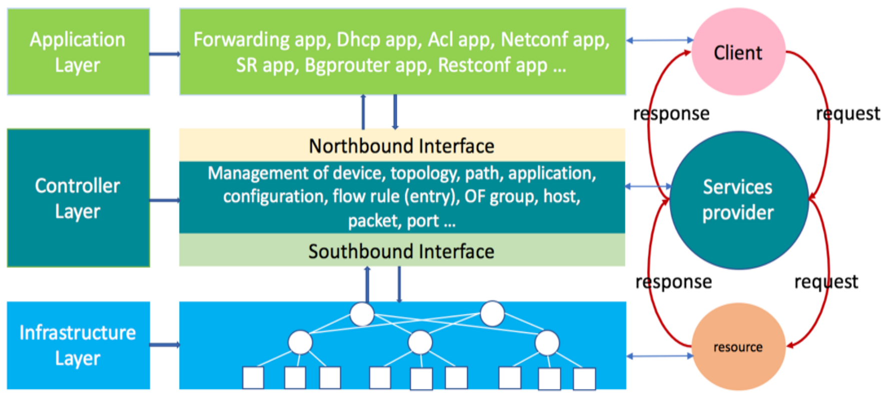

Application Layer
Forwarding app, Dhcp app, Acl app, Netconf app, SR app, Bgprouter app, Restconf app ...
Client
response
request
Controller Layer
Northbound Interface
Management of device, topology, path, application, configuration, flow rule (entry), OF group, host, packet, port ...
Southbound Interface
Services provider
response
request
Infrastructure Layer
resource

Figure 2.4: SDN architecture in layer, service and resource views

A network resource may be occupied since it is used in a customer request, or it may be available for incoming requests. Similar network devices are grouped together into a resource group. The resource database (RDB) is a local repository that stores all information about resources to be survived in SDN controller replacement. SDN controller core services or SDN applications should have adequate rights to update resource information in the RDB. Operations on network devices are managed via the event management in SDN controller, so devices states are up to date for incoming requests. The SDN architecture is presented in the layer, service and resource views as in Figure 2.4.

2.1.3 SDN controller

2.1.3.1 Main characteristics

SDN controller is the brain in the SDN architecture, it plays the role of a network operation system (NOS) to provide core services to manage the SDN network underneath. SDN controller is an intelligent entity that provides service to applications in the application layer. The controller has the central view of the network; it keeps the view up to date in real-time by catching up any event occurring to devices, flows, topology, hence, the controller is called the feedback node. The SDN controller connects to the devices via SBI, connects to the applications via the NBI. Figure 2.4 presents SDN controller in the control layer, in a service to client’s requests.

In the SBI, each device provides its driver to the controller if it is a proprietary SDN switch. For example, the CISCO driver specifies how NETCONF protocol is implemented, how device discovery, link discovery, port discovery and statistics work with CISCO devices; HP drivers describe how HP devices implement the handshake process, how the pipeline works in different

HP groups of devices (V1, V2, V3, V3500) (ONOS 2019b). The drivers work with the OF core service to manage the SDN devices.

In the NBI, ONOS controller provides the Intent framework (ONOS 2019a) as an abstraction for an application to add/update/delete flow rules onto the network devices. Other SDN controllers have similar mechanism in the NBI. ONOS applications consider the intent framework as a critical component as it can update the flow rules on devices without going to low level details.

Internally, each intent has life cycles with different states as follows: starting, complied, deployed, and deleted that are related to the moment when an intent is created, compiled to flow rules, deployed on to the devices, and deleted. Each intent is classified according to the network connection type, such as the point-to-point intent is used for wired networks, the optical connection intent is for optical networks, the multi-point to multi-point intent is for wireless networks. Figure 2.5 presents ONOS intent life cycle.

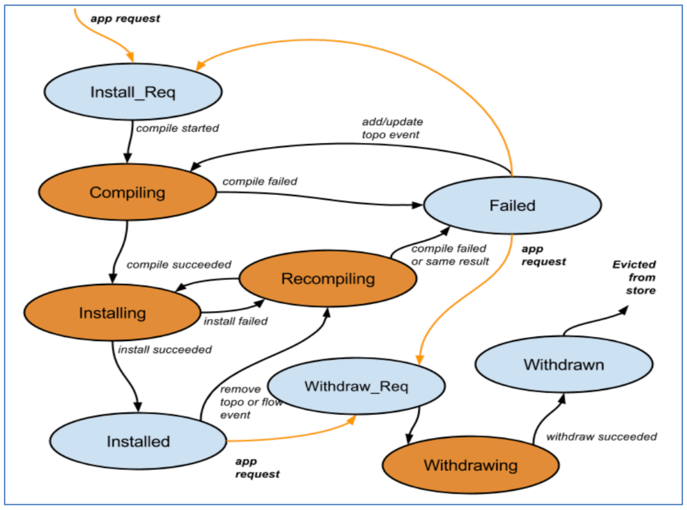


Figure 2.5: ONOS intent state transition diagram (ONOS 2019a)

ONOS provides flow rule API and flow objective API, which are the abstractions as the intent framework, in the NBI to facilitate the application development. Figure 2.6 presents ONOS controller in the SDN architecture.

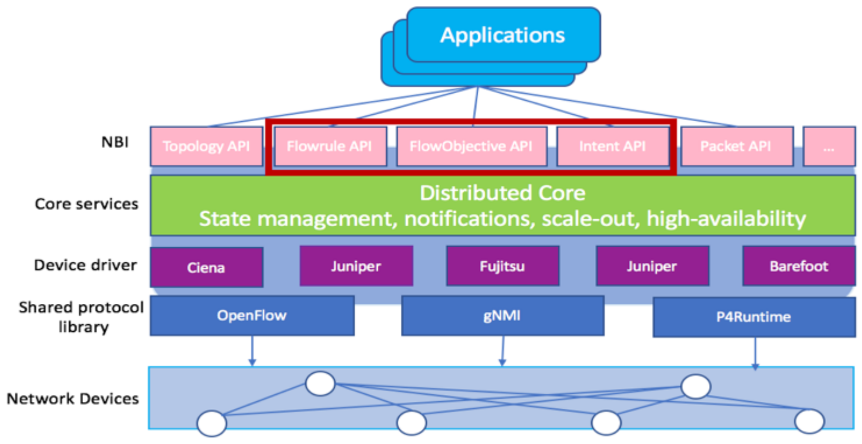


Figure 2.6: ONOS controller (ONF 2019a)

2.1.3.2 Core services

SDN controller provides core services to manage states of SDN network via Rest API. Device management, topology management, path management, application management, configuration management, flow rule (entry) management, OF group management, host management, packet management, port management are fundamental services in an SDN controller. These services are available to be accessed publicly via the NBI. Each SDN controller (open source or proprietary) may have additional core services based on their design. Internally, the services access the SDN devices via the device driver in SBI to create/update/read/delete flows, ports, paths, or packets. Openflow protocol is presented as a core service inside the controller to manage the operation of the controllers and devices according to OF protocol. The OF service handles the handshake process, creates the communication channel, and handles messages between the controller and the device. Although the OF core service is critical for the SDN controller operation, OF service is hidden from external SDN applications. Via programmability, the core services are accessed via the service API, the REST API, and the CLI.

Listed above are common characteristics of an SDN controller (Goransson, Black & Culver 2016), (Kreutz et al. 2015), (Hoang & Pham 2015), (Jarraya, Madi & Debbabi 2014). According to Tootoonchian et al. (2012), the performance of NOX controller was measured, then the design was updated to handle multi-threaded, I/O batching, asynchronous I/O that resulted in the performance was improved by 33 times. The experience from NOX controller is applied to the design of Beacon controller (Erickson 2013) and all controllers (ONOS, OpenDayLight, Ryu, etc.) developed lately.

2.1.4 SDN applications

SDN applications are in the application layer in SDN architecture, they access the resources in network devices via the NBI. To keep the controller small, all network protocols are implemented as SDN applications, which may be included in the controller on the necessity basis. Segment routing (SR) protocol is implemented in the SR application (SRA). SRA is a network application that can be applied to an SDN network without further enhancement. SRA consists of the configuration services for devices and applications, implements fundamental protocols ARP, DHCP, ICMP, implements pseudo wire as a service for layer 2 tunnel using virtual LAN or MPLS label, layer 3 in Internet protocol (IP), Equal Cost multiple path (ECMP) of existing paths in the fabric (Das 2015). The vertical network applications are integrated into the reusable function block (RFB) (Mimidis-Kentis et al. 2019).

2.1.5 SDN switches

SDN devices operate currently under the OF protocol in an SDN networks, hence they are called OF devices (switches / routers). An OF switch consists of an interface to connect to the SDN controller, flow tables, the hardware component, which is CAM or TCAM to forward packets. An OF switch connects to the SDN controller via an OF communication channel, which is a remote procedure call (RPC) session. An OF switch can connect to multiple controllers using multiple channels. An OF switch has OF ports that are used to receive input packets (ingress port) and forward output packets (egress port). An SDN switch has physical ports, OF ports and virtual ports, which can be OF port, as presented in Figure 2.7.

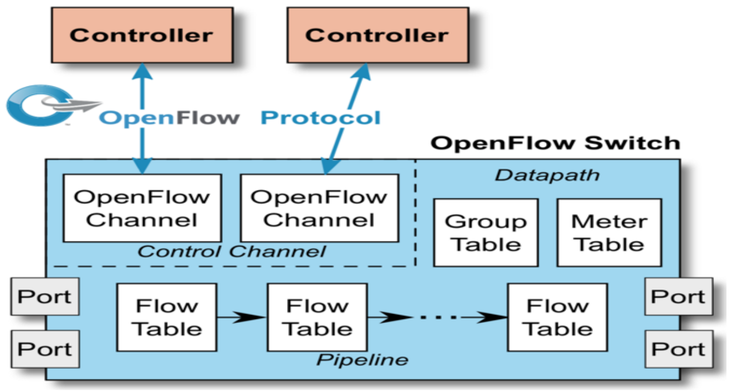
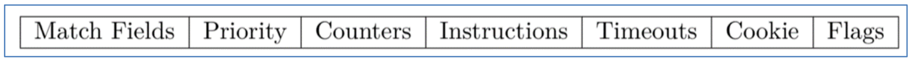


Figure 2.7: Main components of an OF switch (ONF 2014)

Each flow table contains a set of flow entries. Each table entry consists of main components as follows: match fields, priority, counters, timeouts, cookie and flags as presented in Figure 2.8. The entries in the flow tables are added in the proactive mode, which is added in advance based on the topology and protocol of the SDN networks, or in reactive mode, which is added in an ad-hoc manner by the controller.



Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Figure 2.8: Main components of a table entry (ONF 2014)

When an SDN switch receives a packet at an ingress port, it extracts the packet header to lookup in the flow tables to find the matched flow entry, and implements the instructions, which may be forwarding to an egress port, or go to the next table. An entry called table-miss entry with default actions for packets that are not matched any entries (ONF 2014). The implementation of SDN in the cloud data centers with thousands of servers is expensive as it requires many TCAM for possible paths. One approach to reduce the TCAM usage is to apply source routing (SR protocol or MPLS) in SDN network.

2.1.6 Mininet

Mininet is developed as a simulation tool for SDN technology (Lantz 2014). It provides a testbed to test SDN applications in different network topologies. Mininet used open vswitch as SDN network switches. SDN controller is specified whether as a default local controller or a remote controller with the IP address. Mininet is built in Linux environment, it does not have a web interface, the commands are entered in the CLI. Mininet offered default network topologies for test networks; it accepts custom topologies that are developed in Python; it accepts any Linux network commands to retrieve the OF tables, to dump flow entries… There are other simulation environments to test SDN, such as GNS3 (GNS3 2020), but Mininet seems the best fit to evaluation SDN applications on SDN networks.

2.1.7 Segment routing

In SDN networks, Segment routing (SR) is a source routing protocol to steer traffic instead of using TCAM table entries in network devices to optimize cost. Abdullah, Ahmad & Hussain (2018) conducted a survey about the leveraging of SR in SDN networks, and they found that SR source routing is beneficial for the traffic engineering purpose to minimize congestion, minimize end-to-end delay, minimize energy consumption, optimize resource utilization, recovery from failures and traffic duplication for protection. In chapter 4 named Congestion-aware Energy-aware VNE and chapter 5 named Realization of virtual link embedding, the research focuses on the SDN substrate network running SR protocol to provide alternative paths, which are different from shortest paths.

2.1.8 Segment routing architecture

SR allows a router to steer packets through the network using a list of segments (labels). Each segment has an identifier called segment ID (SID). There are three types of segments as follows:

- Node SID is a global SID that is associated with a node in the network,
- Adjacent SID is a local SID that is related to nearby nodes,
- Service SID is a global SID that is associated with a service deployed on a node (Filsfils et al. 2015).

The SR architecture includes the control plane and the data plane. The data plane defines how to encode the list of segments to be applied to the packets. The control plane defines how network devices are instructed to apply a list of segments. SR supports two data plane technologies: MPLS and IPv6. SR applies IPv6 is called SRv6, which is the popular data plane currently.

2.1.9 Segment routing with IPv6

In segment routing with IPv6, every segment is an IPv6 address. SRv6 segment routing header (SRH) is considered as the extension of the IPv6 address header. SRH contains a list of segments; the field “segment left” is decreased 1 every time a segment address is reached in the path. Figure 2.9 presents an extension header in SRv6.

When SR is deployed in an SDN network, SR replaces flow rules in TCAM memory with the list of segment IDs in the packet’s header; hence, costs are minimized. SR also simplifies the maintaining of routing data, hence, it speeds up the process of changing routes or adding new routes to the network (Lee & Sheu 2016). Trellis (Das & Peterson 2019) is an SDN fabric implemented with SRv6.

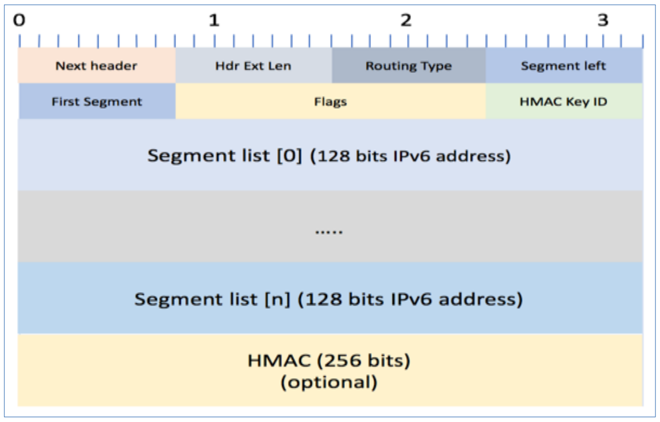


Figure 2.9: Segment routing IPv6 extension header (Lebrun et al. 2018).

2.2 Network virtualization

2.2.1 What is network virtualization

Network virtualization (NV) is the process of creating virtual networks on top of substrate networks. The main problem of NV is the allocation resources for multiple virtual networks on the same substrate network, which is called virtual network embedding (VNE). VNE is an optimization problem focusing on objectives such as to minimize the resource usage, to maximize revenues. An example of VNE is presented in Figure 2.10, the virtual network (left) is embedded on to the physical network (right).

He et al. (2019) conducted a survey about the flexibility that NV brings to the dynamic networks, and they identified that NV ensures virtual network topology adaptation, multi-tenancy and isolation. Other main targets of NV are the parameter configuration, the placement of virtual nodes and virtual links, and resource scaling. Multiple-objective VNE is explored in chapter 4 named CEVNE, chapter 5 named Realization CEVNE virtual link embedding, and chapter 6 named Latency-aware resource optimization for 5G core network slicing.

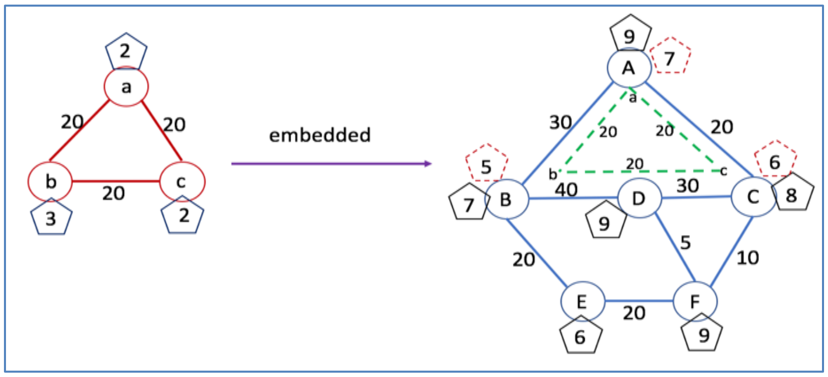


Figure 2.10: An example of virtual network embedding

2.2.2 Resource optimization

The substrate networks include switches / routers and substrate links. The virtual network requests include the CPU, bandwidth demands and the topology of the virtual networks. There are two processes in VNE: the node embedding and the link embedding. The metrics such as acceptance ratio, resource consumption, execution time, energy saving, revenue maximal are used to evaluate the CEVNE research work in chapter 4.

Based on how the node mapping and link mapping cooperation, VNE solutions are classified as follows:

- ✓ Uncoordinated VNE is the process in which the node embedding, and link embedding are solved in their own way without awareness of the other process,
- ✓ Coordinated VNE is the process in which node embedding, and link embedding are working in such a way for the optimal results; this group is divided into one stage and two stages; one-stage coordinated VNE is the process where both processes are embedded at the same time in a coordinated way; two-stage coordinated VNE is the process where the nodes are embedded first so that it makes the link embedding process better off (Fischer et al. 2013).

As VNE is an optimization problem, the VNE is modeled using mathematical programming into a mathematical program (MP). The MP has an objective function to specify the resource minimization, revenue maximization or energy consumption minimization, and the constraints to describe the limits of the substrate capacities, and the conditions of the embedding such as satisfying the resource demands of virtual nodes and virtual links. Solving VNE problem is NP-Hard (Amaldi et al. 2016) as it relates to the multi-way separator problem. Three different solution approaches are investigated so far:

- ✓ Exact solutions are used in small size problems and provide the base line for heuristic algorithms,
- ✓ Heuristic-based solutions try to find good solution, but not optimal, in a low execution time,
- ✓ Meta-heuristic solutions such as simulated annealing, genetic algorithm, ant colony optimization, particle swarm optimization, or tabu search if the VNE problem is such a combinatorial problem that it is hard to use other optimization strategies (Fischer et al. 2013).

Although heuristic algorithms result in non-optimal solutions, but they provide the solution in the affordable time.

2.2.3 Current issues in network virtualization

The main issues of network virtualization in traditional networks are summarized as follows. The node embedding is the main process that determines the success of an VNE algorithm, then based on the node embedding results, the link embedding process is implemented to embed the virtual links on to the substrate network using the current network protocol. The link embedding process is passive, and always return the shortest path (Fischer et al. 2013). This leads to congestion on the shortest path and under-utilized of other paths.

There were other critical issues in NV in cloud data centers (DC). NV plays the main role in DC in provisioning users' requests, which comprised of the VM, and the virtual networks connecting VMs. The cloud service providers must ensure the isolation between tenants, it means isolation of VM resources, and isolation of their virtual networks. Traditional networks in DC implemented tunnels as virtual networks for tenants, which added a burden in the network management, especially in the endpoints of the tunnels. The solution was not flexible in reconfiguration that occurs most often.

2.2.4 Related works of chapter 4 Congestion aware, Energy-aware VNE

On the VNE problem, Zhu & Ammar (2006), Chowdhury, Rahman & Boutaba (2012), Yu et al. (2008), and Cheng et al. (2011) have made profound contributions to VNE solutions.

Zhu & Ammar (2006) investigated the challenge in network virtualization, which is assigning the substrate resources to the virtual networks (VNs) efficiently and on-demand. The authors proposed algorithms to embed VNs without reconfiguration (VNA-I) and with reconfiguration (VNA-II).

Chowdhury, Rahman & Boutaba (2012) proposed ViNEYARD, which consists of a set of online virtual network embedding algorithms for embedding nodes and links. ViNEYARD algorithms consist of the deterministic virtual network algorithm (D-ViNE), randomized VN algorithm (R-ViNE), D-ViNE load balancing (LB) algorithm, R-ViNE-LB algorithm and window-based (Win) VN embedding algorithm. The authors introduced the concept of the augmented substrate graph in their problem formulation. Yu et al. (2008) investigated the design of a substrate network to simplify the VN embedding process. The proposed approaches allow the substrate network to split a virtual link into multiple substrate paths and exploit the path migration to re-optimize the resource allocation.

Cheng et al. (2011) proposed a model using the Markov random walk (MRW) to rank a node based on its resource and topological attributes. The node embedding algorithm is designed to map the virtual nodes on substrate nodes based on the ranking. The link embedding is implemented on shortest paths, using the breadth-first search method with backtracking.

Fischer et al. (2013) surveyed VNE algorithms and classified them using different criteria. They were classified as centralized or distributed using the embedding algorithm. They were classified based on objectives such as the optimal QoS in terms of bandwidth, delay, optimal infrastructure costs, survivable VNEs. Some VNEs have their performance improved via the coordination of node and link mapping processes; these are called coordinated VNEs. In contrast, uncoordinated VNEs have the node embedding and link embedding implemented in the uncoordinated way.

Fischer, Beck & De Meer (2013) proposed an energy-efficient virtual network embedding based on minimizing the number of active substrate nodes. They consider a node is active if one or more virtual nodes are embedded to it. The authors set up the link weight for each link based on whether it is connected to an inactive node and choose paths with minimum weight. They showed that their algorithm was comparable with other algorithms but used significantly fewer active nodes. Özbek et al. (2016) proposed an energy-aware routing algorithm based on the same concept of active nodes in the SDN networks.

Elias et al. (2014) proposed an optimization model to mitigate the congestion in the virtual network functions (VNF) in the cloud environment. The congestion control functions include minimizing the total congestion cost in the link usage and minimizing the congestion ratio. The performance evaluation shows that the investigation is efficient in controlling congestion in the virtual networks. Recently, the authors have extended their work to the distributed cloud environments (Elias et al. 2017). However, it is a single-objective optimization problem.

Zhang et al. (2015) put forward a solution to the VNE problem that aims to satisfy two objectives concurrently: minimization of the energy consumption and maximization of the revenue in a cloud data center. However, they did not consider the congestion avoidance objective at all.

Houidi, Louati & Zeghlache (2015) investigated a three-objective VNE problem that minimizes the energy consumption, maximizes the availability, and the load balance of virtual machines. The proposal applied different algorithms to optimize these objectives, even promoted reconfiguration of embedded nodes and embedded links. Thus, the solution did not consider the disruption in the substrate network and the additional cost that were resulted in continuous reconfiguration.

With the advance of software-defined technology, researchers have renewed their interest in the VNE problem for the software-defined networks. Haghani, Bakhshi & Capone (2018) proposed the VNE algorithms for the SDN substrate networks, aiming to maximize business profiles and minimize delays in the embedded virtual networks (VN). The authors decomposed the problem into two stages. The first stage focuses on maximizing the profit. The result of the first stage is used in the second stage that focuses on minimizing the delay in the embedded VNs. However, they did not consider the ternary content addressable memory (TCAM) problem of OpenFlow (OF) switches.

Li et al. (2018) proposed a self-adaptive VNE (SA-VNE) in the SDN substrate networks, focusing on three types of virtual network requests (VNR): high bandwidth, low latency, and both high bandwidth and low latency. An SDN-based VNE mapping framework is proposed to centralize the allocation of the network resources. The evaluation showed that the SA-VNE could make full use of the resources, improve the revenue, and handle multi-demands effectively. However, the investigation did not take advantage of SDN centralized network view to avoid congestion.

Bays et al. (2016) investigated VNE in the SDN networks. To manage the TCAM memory usage efficiently, each VNR is required to specify a usage profile for its requested VN (the network policies, traffic patterns). Based on this profile, resources (CPU, TCAM memory, bandwidth) are allocated accordingly. The evaluation showed that the scheme is satisfactory in embedding VNs into SDN networks.

Blenk & Kellerer (2019) investigated the SDN virtualization processes that allow multiple virtual SDN networks (vSDNs) to be constructed and share the same underlying infrastructure. A virtualization layer was designed to manage the vSDNs using the network hypervisors. The proposed design, however, did not consider the traffic engineering aspects in these vSDNs. Dahir, Alizadeh & Gözüpek (2019) proposed an energy efficient VNE solution in multi-domain SDNs. The optimization problem was formulated as a mixed-integer linear program (MILP). A heuristic algorithm was designed to solve the problem in polynomial time. The heuristic exploited the hierarchy of the SDN controllers to allocate the cross-domain resources to the VNRs. The solution, however, did not consider the network congestion issue.

Rout, Patra & Sahoo (2018) proposed an energy-aware routing algorithm in an SDN network using the SDN controller. Depending on the incoming traffic, the controller determined a proper link rate for each link. As considerable amount of energy can be saved when the links operate at low rates, the overall network energy consumption was significantly reduced. However, the problem was not formulated as an optimization problem.

To the best of our knowledge, there are no multi-objective VNE solutions that focus on congestion avoidance, energy saving and cost saving which are implemented in an SDN substrate network running SR.

2.3 Network function virtualization

Network function virtualization (NFV) allows network functions to be virtualized and placed optimally on physical networks. He et al. (2019) conducted a survey about the flexibility that NFV brings to dynamic networks, and they identified that the NFV main contributions are the function configuration, parameter configuration, function placement and chaining, and resource and function scaling. NFV is one of the enabling technologies of the NS in 5G core system, which is investigated in chapter 6 named Latency-aware resource optimization for 5G core network slicing.

2.3.1 NFV definition

Network function virtualization (NFV) is a technology that replaces physical network devices implementing network functions with one or more software programs executing the same network function while running on commercial off the shelf (COS) hardware; the software programs are called virtual network functions (VNFs). ETSI proposed NFV for the telecommunication industry in 2014 to apply software into network function in a similar approach to programmability in SDN technology (Pei et al. 2017).

To realize VNF, ETSI proposed a reference architecture called MANO (NFV-MAN 2014), in which there are three main components: the NFV infrastructure (NFVI) layer includes the network substrate resources, software of network functions, and virtualized resources; the virtualized network functions (VNF) layer holds the VNF instances; NFV management block that manages the VNFs life cycles; the operational and orchestration layer allows existing operational systems to support the VNFs that replaced physical devices to continue to work with the whole system (Chayapathi, Hassan & Shah 2016).

2.3.2 Service function chaining

Traffic in a network may need to go through a sequence of network functions that are linked together in a chain, so they constitute a network service through their combined effects. The design of arranging the sequence of network functions is called the service function chaining (SFC) or service chaining (Chayapathi, Hassan & Shah 2016). The main issue of SFC is the filtering and routing of packets to each of the VNFs in the chain. Hantouti et al. (2018) conducted a survey about traffic steering in SFC, and showed that SDN, NFV technologies are used to steer the traffic in three main methods: header-based method, tag-based method and programmable switch-based method. The authors also concentrate on SDN-based methods to steer network traffic as the role of SDN in network configuration and its ability in updating the network flows.

2.3.3 Current issues with NFV and SFC

The main issues of VNFs and SFCs are the efficient allocation of substrate resources (CPU, memory, storage) in the NFVI to activate the VNFs in SFCs so they can process data packets. The SFC main tasks are the placement of the VNFs on to substrate nodes or virtual machines (VMs) that satisfy their computing demands, the placement of link resources onto the virtual links connecting VNFs, and the steering of network flows from VNFs until the flows get to the end of the chain. This is called the placement and chaining of VNF (PC_VNF).

NVF and SFC are designed with SDN technology in mind and the integration between SDN and NFV, SFC has been explored in the traffic filtering and steering. The PC_VNF is the extension of VNE, which is a resource optimization problem, but it is more difficult than the VNE problem as the order of VNFs is considered, which introduces more constraints in their mathematical programs. The role of SDN in PC_VNF is to select actively the substrate paths that meet the bandwidth requirements to embed the virtual links; this is additional to the task of traffic filtering and steering.

2.3.4 Related work of chapter 5 Realization of CEVNE virtual link embedding

Yu et al. (2008) proposed the design of substrate networks that simplifies the virtual network embedding and utilizes efficiently the network resources. The design allows path splitting of a virtual link onto multiple substrate paths if no single substrate path has the required bandwidth to accommodate the virtual link. Each substrate node is kept track with both its CPU resource and the total bandwidth resources of its links:

$$H(n_s) = CPU(n_s) + \sum_{l_s \in L(n_s)} bw(l_s)$$

The formula is used to find the substrate node, where the virtual link can be split, based on the total bandwidth of its links. The path splitting algorithm increases the number of VNRs that can be embedded as well as the acceptance ratio. However, the research did not consider multiple objectives: energy saving, and congestion avoidance.

Zahavi et al. (2016) proposed Links as a Service (LaaS) in data centers as a mechanism for isolating tenants' virtual networks in a cost effective and implementable way. The LaaS architecture is built on top of the cloud architecture, which includes a client front-end, a scheduler and a network controller: i) the scheduler includes functions for allocating virtual links, ii) the network controller employs routing rules to enforce link allocation. However, the research did not consider the energy saving, and congestion avoidance objectives.

Khan et al. (2016) proposed a survivability VNE based on the multi-path link embedding called Survivability in Multi-Path Link Embedding (SiMPLE). SiMPLE focuses on the survivability of link embedding since the link failures occur more often than the node failures, which can be modelled as multiple link failures. SiMPLE operates in both proactive and reactive modes. However, the research did not consider cost saving, energy saving and congestion avoidance objectives.

Marotta et al. (2017) proposed an energy efficient and robust Virtual Network Function (VNF) placement. The energy consumption of servers and switches is considered separately. Each server consumes an amount of energy in the idle period, which is limited by a maximum power consumption. In between is the power consumption that is based on the server's workload. Each switch has two energy consumption values: a static value based on its electronic circuits and chassis, and a dynamic value based on the interface operation. However, the solution did not consider the cost saving and congestion avoidance objectives.

The Green Robust VNF Placements (GRVP) is a heuristic solution with three separate steps as follows: the VNF Chain (VNFC) placements, the robust heuristics, and the latency constraint flow routing. The first step is to allocate VMs resources onto different servers considering the average resource demand is satisfied. This step is implemented based on a mixed integer linear (MIP) problem in a first fit strategy. The second step is called the greedy heuristic for robust VNFC placement. The last step is called the latency constraint flow routing: the traffic latency is calculated based on an optimal model of the least power consumption, and the constraints of the flow latency. It showed that the solution took less than 1 second to place a flow in a large mobile network. However, the solution did not consider cost saving and congestion avoidance objectives.

Chowdhury et al. (2017) proposed the Reallocation of Virtual Network Embedding (ReVINE) algorithm to eliminate the bottlenecks in the VNE. The contribution is presented as follows: (i) the MIP formulation for the optimal ReVINE to balance between the bottleneck of substrate links and the total bandwidth consumption of the virtual networks, (ii) a heuristic algorithm based on simulated annealing, (iii) the comparison of the heuristic algorithm with other annealing based heuristic algorithms. The performance evaluation showed that ReVINE mitigates the substrate link bottlenecks. However, the solution did not consider the cost saving and energy saving objectives.

Many cloud data centers (such as Facebook, Google) deploy the leaf-spine fabric as their networking infrastructure (Moorthy 2016), (Vahdat 2015). Leaf-spine fabric brings major benefits: redundant bandwidth, and easy to scale to allocate to VMs. The main issue with the leaf-spine fabric is the imbalance of the traffic caused by hashing collision in ECMP. Researchers have proposed different methods to load balance internal network traffic in the leaf-spine fabric. These investigations have a common solution, that is to divide elephant flows into smaller flows called flowlets or flowcells. Each investigation applied different methods to steer these flowlets to different paths in the fabric. Alizadeh et al. (2014) proposed Congestion aware (CONGA) using custom ASIC for both leaf and spine switches to identify different paths for flowlets. He et al. (2015) proposed Presto at the edge of cloud data centers using open vswitch (openvswitch.org 2016). Presto also used an SDN controller to have a central view, central control of the substrate network. Katta, Hira, Kim, et al. (2016) proposed an architecture called Hop-by-hop Utilization-aware Load-balancing Architecture (HULA) using P4 language. At the heart of HULA are the processing probes to identify best paths, and the algorithm to forward flowlets to best paths. Katta, Hira, Ghag, et al. (2016) proposed a load balancer called Congestion-aware LOad balancing from the Virtual Edge (CLOVE) using entirely virtual switches of hypervisors. CLOVE identifies congested paths using either P4 programs explicit congestion notification or In-band Network Telemetry (INT). It then updates the weighted round robin on every path for the load balancing. CLOVE is refined (Katta et al. 2017) to work on any off-the-shelf switches. CLOVE combined the stateless transport tunnel (STT) encapsulation, Paris trace routes and open vswitch to identify network states. However, these solutions did not consider cost saving and energy saving objectives.

Li, Lung & Majumdar (2015) and (Li, Lung & Majumdar 2016) proposed an energy aware management technique for the leaf-spine fabric in cloud data centers. The solution keeps track of dynamic workload, especially on spine switches to enable switches that are necessary for the current workload. However, the research did not consider cost saving and congestion avoidance objectives.

All the above link mapping algorithms are designed to work on a single objective; however, a multi-objective virtual link mapping is required.

2.4 Network slicing in 5G networks

The theory of network slicing covers latency requirement of 5G eMBB, uRLLC and mMTC applications, network services, NFV, and SFC; they are explored in this research in chapter 6 named Latency-aware resource optimization for 5G core network slicing. Chandramouli, Liebhart & Pirskanen (2019) investigated the application of 5G mobile networks, and they emphasized that 5G is different from its previous generations, which are 4G, 3G, 2G, which focus on voice and text messages, the new characteristics of 5G are to offer connection to everyone, everything, everywhere at any time with different requirements in latency, bandwidth, and device density.

2.4.1 5G core networks

5G theory provides the basis for network slicing as follows.

2.4.1.1 The control plane (CP) user plane (UP) separation (CUPS)

3GPP proposed the logical separation between the signaling network and data transport network. 5G access-network and core-network functions are separated from the transport network functions (Penttinen 2019). The separation reduces the latency on application service, supports increased data traffic, allows the updating, locating and scaling independently of CP and UP resource, and enables SDN to deliver user plane data more efficiently (Schmitt, Landais & YongYang 2018).

Figure 2.11 presents the CUPS interfaces of UP and CP of Service gateway (SGW), packet data gateway (PGW), and traffic detection function (TDF).

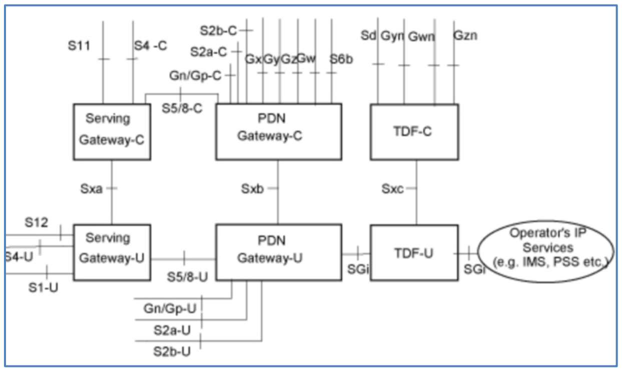


Figure 2.11: CUPS interfaces (Schmitt, Landais & YongYang 2018)

2.4.1.2 The service-based architecture of 5G core network

The network functions in 5G core system are REST service based. 5G core system forms a service-based architecture, they use APIs to communicate with each other. 5G has a new group of customers called verticals, which consist of 5G applications in various areas in research and industries; verticals access 5G network functions via 5G core service APIs. 5G core services are accessed via the internet with HTTP/2 (Mayer 2018).

2.4.1.3 Three main types of 5G applications

5G applications are classified into three main types: enhanced mobile broadband (eMBB), ultra-reliable low latency communication (uRLLC), and massive machine type communication (mMTC). 5G application types are classified according to their focus either on human centric, device centric, or every device; their vertical sector, and their key performance indicators (KPI). They are presented briefly as follows (Marsch et al. 2018), (ITU-R_M.2083 2015), (Foukas et al. 2017).

- ❖ eMBB focuses on the human-centric access to multi-media content, services, and internet, which have been provided by 4G. 5G facilitates creating new applications, improving performance and user experience in existing services. The new applications are either wide-area coverage or hotspot. eMBB focuses on the KPIs: spectrum efficiency, traffic density, power efficiency, mobility, and data rate.

- uRLLC is related to services with stringent requirements in latency and availability. It also focuses on the reliability, mobility, and data rate KPIs. This is the new 5G application type, which is utilized in new 5G vertical applications such as autonomous driving, next generation factories, virtual reality applications.
- mMTC is related to applications having very large number of devices transmitting non-delay sensitive data in low volume. mMTC focuses on the connection density KPI.

In Figure 2.12, the mobile broadband (green color) represents the eMBB, the critical communication (yellow color) represents the uRLLC, and the machine-to-machine (dark blue color) represents the mMTC.

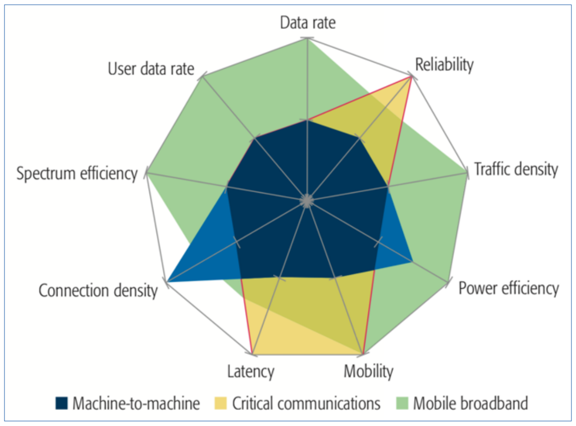


Figure 2.12: Application types and their key performance indicators (Foukas et al. 2017)

In 5G architecture, network slicing (NS) is proposed as a virtual mobile system to implement services belonging to any group; hence, it plays an important role in delivering 5G services; NS is explored in detail in the next subsection.

2.4.2 Network slicing in 5G

2.4.2.1 5G service

5G system architecture is defined as service-based architecture (SBA) (TS_23.501 Jun. 2019) that applies the RESTful design paradigm to expose its core services to vertical industry such as IoT, Mobile Vehicle, or Factories according to Mayer (2018). This is a 5G distinctive feature compared to previous generation such that 5G mobile services can be embedded into different vertical applications. 5G core network functions are accessed via the service-based interfaces (SBI). The

5G core network functions are divided into main groups: network access control, registration and connection management, session management, QoS, security, charging, policy control, network slicing selection (TS_23.501 Jun. 2019). Each main group, as its name suggests, provides network functions in that area. In each group, there are main component services that are constructed by atomic services. Services in different groups can call each other using its REST APIs; this means each network service can play the role of a service consumer or service provider. A network service may be made up of several component services, which are composed of several atomic services. 5G SBA promotes software reuse via service composition to deliver the services to different customers. 5G promotes network slicing as its new design paradigm that leverages the virtualization techniques at both the hardware and software levels to deliver services simultaneous to customers having quite different latency requirements. Figure 2.13 presents 5G core services and their interfaces.

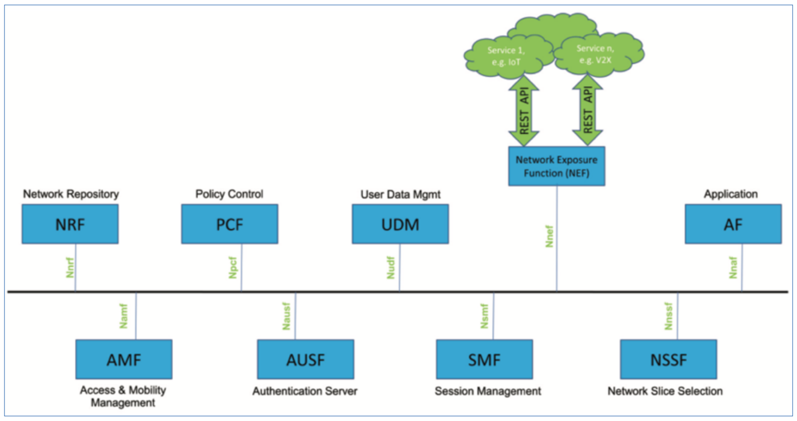


Figure 2.13: 5G core service: RESTFul API of the service-based interface and the northbound communication (Mayer 2018)

2.4.2.2 5G network slicing

Network slicing is a mechanism to enable the deployment of multiple end-to-end logical, self-contained networks on a common infrastructure platform so different mobile applications for different users can run concurrently (TR-526 2016). The end-to-end logical networks are used to deliver mobile services that users request, the network slice (NS) is created to deliver the required service. As a mobile service is constituted by components, and atomic services, a NS must include the runtime of these services to process data packet. A NS needs to deploy the service instance; a

NS needs CPU resources, memory resources, storage resources on servers, virtual machines (VM) or containers to run the services; it needs bandwidth resources to carry packets from one service to another at the required speed so the whole mobile service is delivered during the timeframe. The example in Figure 2.14 consists of four network slices, which are running different mobile service concurrently on the same infrastructure. NS 1 belongs to service 1 (*eMBB* type) from the Telstra telecom company, NS 2 belongs to service 2 in ambulance eHealth (*uRLLC* type), NS 3 belongs to service 3 in the vehicle-to-vehicle application (*uRLLC* type), and NS 4 belongs to service 4 for users at a stadium in a sport event (*eMBB* type). The infrastructure layer provides the substrate resources for all four network slices.

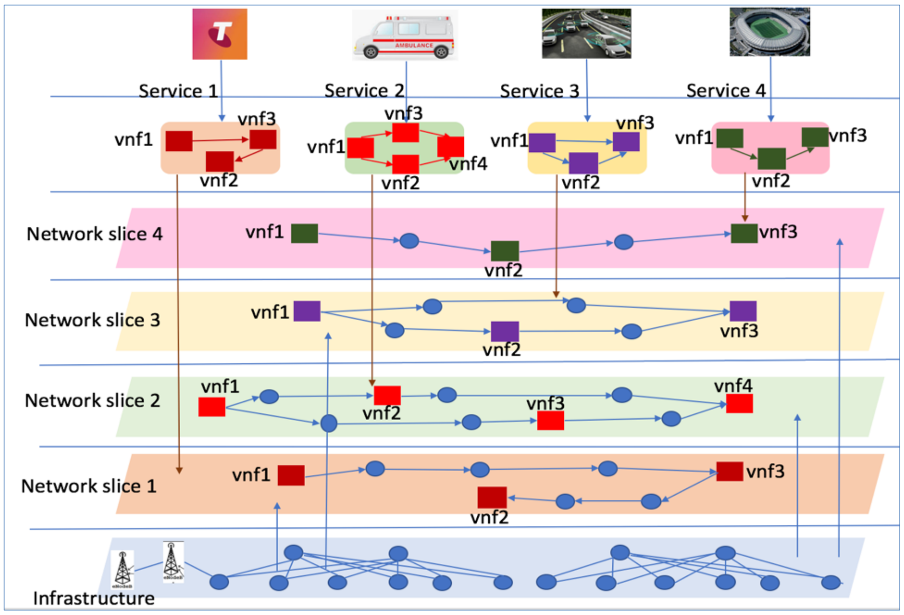


Figure 2.14: 5G network slice examples, adapted from (Perez 2017)

Technologies that enable network slicing, are cloud computing (CL) or container technology, which provide virtual machines or containers on the shared substrate servers; network virtualization technology, which creates virtual networks on the shared substrate network infrastructure; network function virtualization (NFV), which creates virtual network functions to place them in the virtual networks. On top of these technologies is software-defined networking (SDN), which plays the controller role in the whole infrastructure to route the traffic from end-to-end of the NS. In 5G mobile network, a NS comprises Radio access network (RAN) slice, Core network slice and Transport network slice (Sunay 2017). In this thesis, we concentrate only on core NS.

Network slicing is a 5G distinct feature: 4G core network functions are implemented on the physical core networks, while in 5G, core services are implemented in network slices that are provisioned for the service and the customer only, and the slice is deleted when the service is completed.

A network slice includes VNFs in SFCs to implement a mobile service requested by external customers. The network slicing problem involves the placement and chaining of VNFs in SFCs; to implement the mobile service, the network slice needs to process all packets to meet the latency threshold, so it satisfies the SLA of the customers. As there are different mobile services in three different types of applications: eMBB, uRLLC, and mMTC, each has the latency thresholds 10 to 100 times different from other types, so the differences in latency thresholds play an important role in the provisioning of network slices. The network slicing problem is an optimization problem in resource allocation with the strict latency constraints based on the application type of the service.

As 5G mobile networks are in the process of standardization, different standards have been designed by different technical groups that will not be finalized until 2020 by the International Telecommunication Union (ITU). The 3GPP is the group that proposes significant changes to the 5G network in architecture, use cases, access networks, core networks, and transport networks; their research has been supported by other technical groups.

2.4.3 Related work of chapter 6 named Latency-aware resource optimization for 5G core network slicing

Afolabi et al. (2018) is a comprehensive survey focuses on the principal concept and the enabling technologies that allow the end-to-end network slicing. Network slicing is based on seven concepts as follows: automation, isolation, customization, elasticity, programmability, end-to-end and hierarchy abstraction. The business roles involved with 5G network slicing are infrastructure providers, cloud providers, virtual network operators, service brokers, application providers and verticals. Network sharing supports network slicing in the business cases: multiple core networks share a common RAN; multiple RANs share a common CORE network. Technologies enabling network slicing are hypervisors, virtual machines and containers, SDN, NFV, cloud and edge computing. It concentrates on all aspects of network slicing: access, core and transport slicing. It identifies future research challenges: network slicing architecture and APIs, UE slicing, slice optimality / pareto optimality, and slice security.

Foukas et al. (2017) focuses on the architecture for network slicing. The infrastructure layer spans both the RAN and Core networks. It includes the allocation of resources to slices and the management of resources. The infrastructure will be virtualized to serve customers' requests. The network function layer focuses on the configuration and management of network functions. The granularity of network functions considers course-grained and fine-grained functions. The coarse-grained functions are simple to be deployed but the slice is less flexible and less adaptive to change. But the fine granularity may cause issues in interfacing and chaining as more network functions exist. The service layer is linked to the service model of the network slice. The service description should include the network functions involved so the placement process is simplified.

Agarwal et al. (2018) proposed a solution for VNF placement and CPU allocation using queueing theory. The solution modeled VNFs as M/M/1 queues, all VNFs are connected in arbitrary graphs, multiple CPU allocation decisions are available for a VNF, and the effect is measured about the allocation decisions on system performance. A heuristic algorithm is designed on joint VNF placement and CPU allocation. The evaluation showed that the performance is consistent and close to optimum for any of the types of VNF graphs. However, the solution did not consider how the allocated resources contribute to satisfy the latency threshold of all VNFs.

Vassilaras et al. (2017) investigated the factors that support network slicing in 5G in different online problems: the online minimum cost multi-commodity, the online network embedding, the online VNF placement, the online packing, and the online facility location. Although a thorough investigation about network slicing is conducted, no actual problem formulation and no algorithms are investigated to satisfy the strict requirements on latency of 5G.

Paschos, Abdullah & Vassilaras (2018) proposed a solution to network slicing with splittable flows in 5G networks. The virtual network embedding (VNE) splittable flow is based on the formation of a multipartite graph and is modelled to capture the combinatorial structure of different embedding options. Solving the VNE problem is NP-Hard, therefore, a poly-time heuristic is proposed. The evaluation showed that the heuristic provides good performance. However, the solution did not consider the latency satisfaction in 5G networks.

Leconte et al. (2018) proposed an optimization resource allocation framework for robust-and-efficient resource provisioning for network slicing. The framework is modelled based on the utility function of network bandwidth and cloud capacities. The network designer can drive the system to different operational trade-offs between traffic-fairness and computing fairness. An algorithm is designed to resolve their MP using the Alternating direction method of multipliers (ADMM). However, it did not consider the satisfaction of latency requirement in 5G.

Ye et al. (2018) modeled an end-to-end packet delay for multiple traffic flows passing through an embedded virtual network function chain in 5G. The model is built on the queueing theory, in which a tandem queue is established for packets going through CPU processing and link transmission. The dominant resource generalized processing sharing (DR-GPS) is deployed so that either computing or bandwidth is dominant, the other resource is allocated to ensure high utilization. However, the contribution is mainly to propose a VNF queue model for the processing delay.

Dietrich et al. (2017) proposed a solution to the VNF placement problem in the virtualized cellular cores (4G). The solution consists of the mixed-integer linear program (MILP) to retrieve optimal embedding solutions, a scalable LP with a rounding algorithm for near-optimal solutions in poly-time, and a greedy heuristic as a base line. The evaluation showed that the LP outperformed the MILP in load balancing, acceptance ratio, resource allocation, and runtime. However, the LP imposed a small penalty in inter-data center link load balancing.

Sciancalepore et al. (2017) proposed a solution for traffic analysis and prediction per network slice, admission control decisions for network slice requests, and adaptive corrections of forecast load based on deviations. The evaluation showed the solution achieved substantial results in resource utilization. However, the solution did not use the traffic forecasting to improve the latency of network slicing.

Fendt et al. (2018) proposed a resource allocation process in network slicing for 5G networks that gives feedback to clients and operators about the feasibility to provision the network slice based on the current resources in the system. They analyzed the strength, weakness of their solution focusing on the radio access network. However, the investigation presented the analysis model without proceeding to the mathematical model.

Caballero et al. (2018) proposed a network slicing framework using Games theory called NES that combined the admission control, resource allocation and user dropping. They proved that if an admission control ensured slices satisfying the rate requirement, the game is a Nash equilibrium. The evaluation showed that the research achieved the same or better utility than static resource allocation. However, the solution considered user dropping out as one of the conditions to balance the framework.

Gouareb, Friderikos & Aghvami (2018) proposed a solution to the problem of VNF routing and placement to minimize the edge-cloud latency. The problem is modelled as an integer non-linear program (MINLP) that subsequently applied linearization techniques. The evaluation showed that the solution reduced network delay by 18% in the single feature request scenario. However, the first drawback is the assumption that all VNFs of the network slice have already been admitted, and the second drawback is the assumption of unchanged arrival rate.

Prados-Garzon et al. (2019) proposed a holistic Long-Term Evolution (LTE) mathematical framework for Evolved Packet Core (4G) network slice planning for eMBB applications. The research comprises LTE workload generation for both control plane (CP), data plane (DP), the analysis model to predict CP, DP performance (response time and packet loss) using queueing theory. It is modelled as a multi-objective optimization problem that consists of minimal workload imbalance between edge clouds, maximal resource utilization, and quality of experience (QoE) to users. However, the solution did not consider 5G core networks.

Addad et al. (2019) proposed a holistic optimization model for cross-domain network slicing for 5G core networks devoted to the ultra-fast uRLLC (cars and drones) applications. The proposed architecture comprises the infrastructure layer, the VNF layer and the network-slice layer over multiple domains. The MP consists of the objective function and constraints in the VNF placement, resource, link arrangement, latency aware, and bandwidth aware. The MP is transformed into an MIP that is solved using Gurobi. The heuristic is presented to reduce computational costs. The evaluation showed that the heuristic has polynomial runtime. However, the solution did not consider the relation between allocated resources and processing latency in VNFs.

Alleg et al. (2017) proposed an adaptable VNF placement and chaining that satisfies the network latency, throughput, and error rate. The research assumed the virtue of linear proportion between the computing resource and the VNF processing speed that is presented as the Flexible resource allocation model (FRAM). The FRAM mathematical model is a mixed-integer quadratic constraint program (MIQCP) that is resolved using AIMMS (AIMMS 2020) version 4.3. To demonstrate the robustness of FRAM, the Strict resource allocation model (SRAM) is introduced. SRAM is modeled as a mixed-integer linear program (MILP) that allocated resources at the maximal demand. The evaluation showed that FRAM outperforms SRAM in resource saving, admission rate, and execution time. However, the solution has not considered the satisfaction of a wide range of latency requirements in 5G.

Wang et al. (June 2019) proposed the SliceNet framework based on the network slicing technology in 5G. The research focuses on the migration of eHealth Telemedicine to 5G network. SliceNet consists of several layers as follows: (i) the SFC enables best effort slicing and the network function forwarding graph enables network services for a slice, (ii) QoS-aware slicing attempts to realize the network slice with guaranteed network layer QoS, (iii) the control plane can customize the slice based on vertical application requirements, (iv) The layer enables E2E network slicing and multi-domains in the future, and (v) the cognitive management is used to achieve the QoE experience. However, the solution is a not a network slicing optimization problem.

Kammoun et al. (May 2018) proposed the heuristic algorithms for network slicing focusing on three objectives: reliability, availability, and latency. The problem is formulated as a mathematical model. Solving the model is NP-Hard, hence, the authors propose two algorithms: (i) the first algorithm selects the paths having sufficient resources using a SDN controller, (ii) the second algorithm selects the best path from the results of algorithm 1. The evaluation is conducted on each objective separately. However, the solution did not apply the composition theory to get the results that consider all objectives at once.

The NECOS project (NECOS 2019) extends network slicing further to Cloud network slicing that supports a vertical industry or service. A cloud network slice is a set of infrastructures (network, cloud, data center), components / network functions, infrastructure resources (managed connectivity, compute, storage resource) and service functions that have attributes designed specifically to meet the needs of a vertical industry or service. Therefore, a cloud network slice is a managed group of resources, network functions, network virtualized functions at different planes in the architecture as follows: data plane, control plane, management/orchestrator plane, and service plane. The behaviors of a cloud network slice are realized via network slice instance. The NECOS architecture is designed to provide slices as a service.

To the best of our knowledge, there are no network slicing investigations for 5G core network that focus on minimizing resource allocation to satisfy a wide range of network latency for different service types: *eMBB, uRLLC* and *mMTC.*

2.5 Micro-service architecture and patterns

Micro-service architecture and patterns are explored in Chapter 3 named Architecture for extended SDN applications. The three-tier architecture is proposed with UI tier, business tier and database tier. The proposed architecture is based on micro-service design principles; it promotes the service composition, service reuse in exploiting service discovery pattern, service compose and decompose pattern, service registry pattern.

2.5.1 Micro-service architecture

What is a micro-service? A micro-service (MS) is an independent, standalone process that can perform a function on all its clients. It may have its persistent data store. What makes it unique is each MS is developed, deployed independently from others (Kocher 2018).

What is micro-service architecture? Micro-service architecture (MSA) is an application architecture pattern (Richardson 2018), in which an application is constructed as a set of loosely coupled micro-services, so that they enable a small team to rapidly develop, test, deploy, and maintain their applications independently from other applications. In an MSA, services communicate with each other using HTTP/REST or asynchronous protocols. MSA is popular in SDN technology as open source SDN controllers such as ONOS, OpenDayLight are MSA applications. The SDN controllers are developed in Java using OSGi technology, which promotes service reuse, so it goes well with MSA.

Micro-service architecture (MSA) is part of the service-oriented architecture that promotes the service-oriented patterns: service provider, service consumer, service discovery (Fowler 2014), (Lewis 2012), (Richardson 2019), and allows the flexibility in managing applications. The strength of MSA is its service composability, allowing the construction of fully functioning applications by integrating atomic services; each provides a function, based on some business patterns. The service composition can be accomplished through an automation process with a service integration engine, or it can be done on an individual service. In an SDN controller, new services are composed of the core network services and existing services for SDN applications. The MSA main characteristics facilitate the development of robust, modular, and reusable applications. The characteristics are presented as follows:

- ✓ Data decentralised principle: in our design, each application has its own database management system to ensure application independence from the data perspective.

- ✓ Componentised application: the application comprises web service components to promote reuse of existing services.
- ✓ Application design robustness: to ensure the application is robust, our design considers both successful and failed scenarios.
- ✓ Distributed principle: each application can run in their own process so they will not interfere with each other within the controller.

In Figure 2.15, the data decentralized attribute, the composition attribute of MSA are visualized.

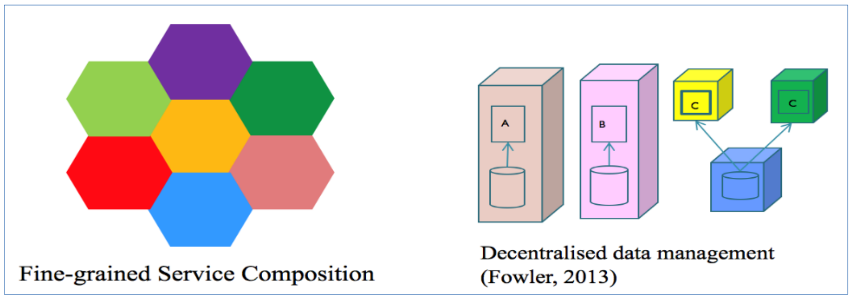

Fine-grained Service Composition
A
B
C
C
Decentralised data management
(Fowler, 2013)

Figure 2.15: the characteristics of Micro-services (Fowler 2014)

2.5.2 Micro-service Patterns

Micro-service patterns become more and more important as they promote reusing existing services, and cost and time saving from ‘reinventing the wheel’. One of our solutions is about the architecture of extended SDN applications that are applying MSA, MS design patterns. More and more micro-services patterns have been discovered (Richardson 2018); they are classified into different groups based on the problem that is solved by the pattern, such as application architecture group, service discovery group, data management group. The focus will be on the application architecture patterns, service discovery pattern, and data management pattern.

2.5.2.1 Decomposition pattern

MS presents an approach to escape a monolithic application (mono app) using decomposition pattern (Decom), which breaks the mono application into a set of micro-services such that they retain their functionalities. According to Richardson (2018), there are two methods: decompose by business capacity, and decompose by subdomain. While “decompose by business capacity” focuses on different steps of the business values in enterprises, “decompose by subdomain” may be suitable for the engineering discipline. “Domain driven design” (DDD) is a design methodology that supports the decomposition into services (Millett 2015); DDD principles may be applied to Decom.

2.5.2.2 Discovery pattern

In MSA, the discovery of existing micro-services is important. There are several service discovery patterns such as client-side discovery, server-side discovery, service registry, soft registration, and 3rd party registration, which are presented as follows (Richardson 2019)

- Client-side discovery discovers MS on the client side,
- Server-side discovery discovers MS on the server side,
- Service registry registers each MS to the service repository when it is in operation, removing it from the repository when it is deleted,
- Self- registration is the process that allows an MS registers itself when it is in operation, and it is removed from the registry when it is deleted,
- 3rd party registration is like other service registration processes, except it is implemented by a third-party organization, which is not the service provider or service consumer.

2.5.2.3 Database patterns

MS is very beneficial in flexible development, deployment, especially with data management as each MS may have its own database. Richardson (2019) identified MS patterns that consist of database per service in data management as described as follows.

Database per service ensures the persistent data for each service is available internally to it, users may call the service through its API. There are several options to ensure private database per service as follows: separate table per service: each service owns a set of tables that can access / update only by the service, schema per service: each service has its own schema, and database server per service: each service has its own server (Richardson 2019).

2.5.2.4 Integration into an application

The most important pattern of MS is the integration of MS into an application. According to Wolff (2017) MS can be integrated at different levels:

- MS can be integrated at UI level,
- MS can be integrated at the logic level,
- MS can be integrated at the database level.

MS integration at the logic level is most common using different technologies: REST, Coding and Messaging. Each MS has the REST API, using different data formats such as XML, JSON, or Protocol Buffer. Coding for integration calls the MS via their APIs in the registry in the implementation code using Python, Java, .Net. MS can be integrated using Messaging; each MS sends messages to another MS as a request and receives responses.

2.6 Software architecture patterns: module pattern

Bass (2013) identified different software architecture patterns; in each pattern he identified the context, the problem and the solution that the pattern resolves. In chapter 3, the solution leverages the modular design pattern, which is based on the module pattern as follows.

1. Context

In software architecture, the module pattern is designed for the context of large and complex systems, there are always needs to develop or evolve some portions independently. Hence, developers need a clear separation of concerns so that independent modules are developed and maintained separately.

2. Problem

The problem is that the software needs to be segmented into different modules so that they can be developed and evolved independently with little interaction between the modules, supporting modifiability and reuse.

3. Solution

To achieve the separation of concerns, the layered pattern divides the software into units called layers. Each layer consists of a set of modules that deliver cohesive services. Between adjacent layers, there is a constraint called "allowed-to-use", which is a kind of dependent relationship and unidirectional, in which only the above layer is "allowed-to-use" the lower layer.

4. Constraints

Each module must be allocated to exactly one layer. There are at least two layers (but usually there are three layers or more). The allow-to-use relation should not be bidirectional: the lower layer cannot use the above layer.

5. Weakness

The layers add up-front costs and complexity to the system. Layers might contribute to the performance penalty.

In this chapter, Figure 2.3, and Figure 2.4 present the SDN architecture, which is an example that applies the module architecture pattern.

2.6.1 Reusable function blocks and Next generation PaaS

The reusable function block and next generation platform as a service are applied in chapter 5 named Realization of congestion-aware energy-aware virtual link embedding.

2.6.1.1 Reusable function block

Reusable function block (RFB) consists of both SDN applications and network functions in NFV. The main different between RFB and NVFs is as follows: RFB can be decomposed recursively into other RFBs. RFB can be mapped to both software components (virtual networks) and hardware components (network infrastructure). RFB can include their metadata in the composition resulting in its simple configuration process. Figure 2.16 presents the RFB concept, in which an RFB is constituted as micro-services and consists of the IaaS that it is deployed in it; RFBs are composed into a parent RFB.

Each industry has its main business that differentiates it from other vertical industries. The business in each industry also has its own advantages compared to other companies in the same industry. The RFB represents a specific service in an industry; it needs to be customized to work for a company in the industry. This shows the two main characteristics of RFB:

- It represents common functions in the industry,
- It needs to be customized to provide the advantages for companies to attract customers.

RFB is the main component of the next generation platform as a service (NGPaaS).

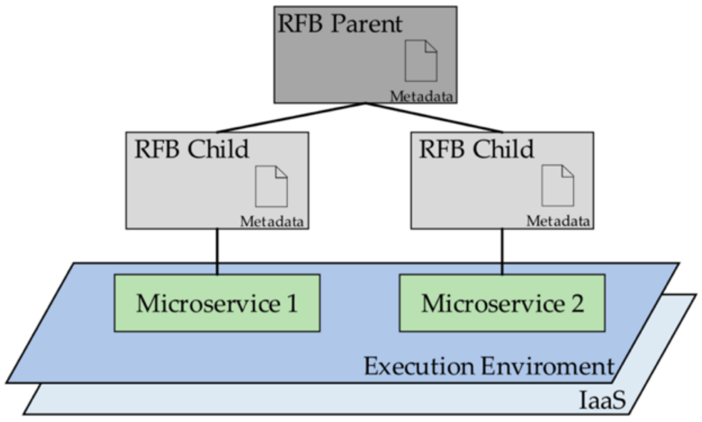


Figure 2.16: RFB concept (Mimidis-Kentis et al. 2019)

2.6.1.2 Next generation platform as a service

Next generation platform as a service (NGPaaS) arises in the context of 5G core systems, in which services are very diverse, hence, there are no one-size-fits-all infrastructure that can support all 5G core services (Van Rossem et al. 2018). NGPaaS is the concept of a platform of platforms, which is based on micro-services, modularity and build-to-order. NGPaaS is based on the workflows that allow the building of an integration platform, which is constituted of component platforms, each of which is a deployed RFB (Mimidis-Kentis et al. 2019). CORD is an example of reusable function block (RFB) (Peterson 2019).

2.7 Optimization and Mathematical modelling

Optimization and mathematical modelling are applied in the problem formulation of chapter 4 named Congestion-aware Energy-aware Virtual Network Embedding, and chapter 6 named Latency-aware resource optimization for 5G core network slicing. While chapter 4 focuses on a mixed-integer linear program, chapter 6 explores non-convex, mixed-integer, non-linear program, they are both NP-Hard. Both chapters proposed heuristic algorithms to find close-to-optimal solutions using different solution approaches and optimization solvers.

Optimization problems in network virtualization, network function virtualization and network slicing in 5G are part of the Operational Research (OR) that classifies and investigates solutions to optimization problems (Rardin 2017). OR is an important research area because of its contribution to the economics by reducing cost, increasing products and profits. In this section, the mathematical programming (MP) models are applied to solve the resource optimization problems. This section presents the methods using mathematical programming (MP) to solve the multi-objective problems, the large system and structure system based on the theory investigated by Cohon (1978), Lasdon (1970); Williams (2013). We acknowledge the Genetic and Evolution Algorithms methods (Coello, Lamont & Van Veldhuizen 2007) to solve the multi-objective problem, but our solutions are investigated based on the MP, which is more suitable for our problem domains.

2.7.1 Mathematical models

Optimization problems are modelled using mathematical models. A mathematical model is an abstract model that shows the interconnection between objects using algebraic symbolism. The interconnection between objects is described using a set of mathematical relationships such as equations, logical dependencies, equalities and inequalities (Williams 2013). The motives to build a mathematical model are as follows: (i) it reveals connections that are not obvious to see, (ii) the model is used to analyze for the course of actions, and (iii) one can conduct experiments with models by varying the parameters in the mathematical equations, inequalities.

2.7.2 Solving mathematical programs

A mathematical model of an optimization problem is called its mathematical program (MP). MP includes decision variables, the objective function, which is to minimize resources, maximize benefits, and the constraints, which are based on the limit of resources, are the operation conditions. MP needs to be solved by specialized software called optimization solvers, to find the optimal values for the objective function and decision variables.

Depending on the algebraic expression and equations in the MP, MPs are classified as linear programs (LP) if their objective functions, constraints are linear functions; MPs are integer programs (IP) if the decision variables only have integer values; MPs are mixed integer linear programs (MILP) if some decision variables accept integer values only while others accept real values. LP are most popular MPs, their solvers can be in gnu license such as GLPK, or proprietary such as AIMMS, IBM CPLEX. Alevin framework is built with GLPK solver to automate the solving LP, also provide the metrics to evaluate the MPs.

Solving MILP models is computational intractable (Fischer et al. 2013), so one solution approach is to transform them into LPs by relaxing the integer constraints; then, in the sub-optimal solutions, these linear constraints are converted back to the approximation of the integer variables using the rounding techniques. In non-linear programs (NLP), the objective function and constraints consist of quadratic terms. There is also a mixed integer non-linear program (MINLP), in which some decision variables accept integer values only, and the MP is non-linear. NLPs are also classified as convex or non-convex (Rardin 2017). For MINLP, as they are non-convex, the solving process includes convex relaxation by replacing non-convex terms with a different variable (Bliek1ú, Bonami & Lodi Oct. 2014), and adding lower, upper binding conditions to it. After this process, it becomes nonlinear, convex to be solved by nonlinear solvers.

2.7.3 Multi-objective programming

This subsection presents the solution to the multi-objective problems using mathematical programming as specified in section 2.7. The multi-objective problems focus on the optimization of two or more objectives. Multi-objective problems are more challenging than single-objective problems as the optimal solutions of one objective may not be the optimal solutions of the other objectives. Hence, the optimal solutions need to be considered for all involved objectives. Cohon (1978) introduced the concept of noninferiority solutions that is related to the optimal solutions of the multi-objective problems.

The noninferior solutions mark the set of values of each objective, and the values of decision variables; any change for the better in one objective must come with a cost of being worse off in another objective. They are also called the pareto-optimal solutions. In contrast, there exist inferior solution sets, in which the solutions could be improved for all involved objectives. The noninferior solution that is selected among all the alternatives is called the best compromise solution (Cohon 1978). Preferences among objectives or graphical representation of alternative noninferior sets are often used to select the best-compromise solution. Solving of the multi-objective problems is the searching for the noninferior solutions, then among the noninferior solutions, searching for the best compromise solution. The rules to generate noninferior solutions are very important for the multi-objective problems.

Several well-established modelling techniques and solution strategies have been presented for the multi-objective problems in Williams (2013), Cohon (1978). Solving each objective independently is one of the approaches. We summarize three relevant approaches by Williams (2013) for solving the multi-objective problem and the rules to generate pareto-optimal solutions in each approach by Cohon (1978).

1. The first approach is to solve the model with each objective, compare the results of each objective to search for a satisfactory result or further investigation. Obviously, trying each solution is the only way to search for the pareto-optimal solutions in this approach.
2. The second approach is to interchange between objectives and constraints after the model has been built. This is called the constraint method (Cohon 1978). Preferences among objectives or graphical representation of alternative noninferior sets are often used. Experiments will be then conducted by considering all but one objective as constraints, varying the objective to be optimized and the right-hand value of constraints. In this approach, (Cohon 1978) showed that the rule to generate the noninferior solutions is all constraints on the objectives are binding.
3. The third approach is to take a suitable linear combination of all the objective functions, each objective with its weight and optimize the whole problem. Experiment is required to find possible solutions. This is called the weighting method by Cohon (1978). Each of the objective terms is normalised to have the same unit. In general, the normalizing factor for each term is considered a base weight; variable weights can be incorporated into the base weights to express the priority or preference on each of the objectives. In this approach, Cohon (1978) showed that the rule to generate the noninferior solutions is all the weights are strictly positive.

2.7.4 Large systems, structure systems

Lasdon (1970) observed that the real systems usually are large as they comprise many factors in a real production environment. Large systems always have special structure; hence, they are also called structure systems. Large programming models arise through the combining of smaller models as observed by Williams (2013). Structure models are more powerful decision-making tools than small models as a structure model considers every requirement of small constituent model. The structure model matrix representation is depicted in Figure 2.17.

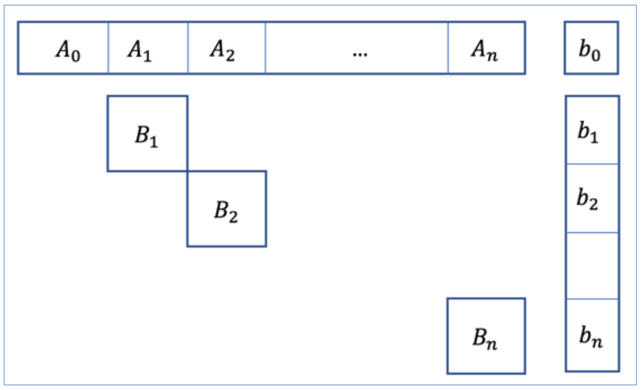


Figure 2.17: The general structure of the main model and sub-models using matrix (Williams 2013)

In Figure 2.17, line $A_0, A_1, A_2, .., A_n = b_0$ is the common block of the large and small models. The objective of the large model is in the common block. The number of common constraints in the common block is a measure of how the optimal solution of the large models is close to the sum of optimal solutions of sub-models; it is the case of no or fewer common constraints. Solution approaches for structure systems are direct methods or decomposition / partitioning methods. If a large system is solved using the direct method, it will be very costly as there are a huge number of decision variables, rows and columns. We concentrate on the decomposition approach. As the sub-problems interact via the common constraints, solving only the subsystems will not yield correct solutions. A control level called the master problem of the sub-problems is required so the solutions of the original problem are maintained. The decomposition approach is depicted in Figure 2.18. The master model connects to the sub-models using messages, etc.

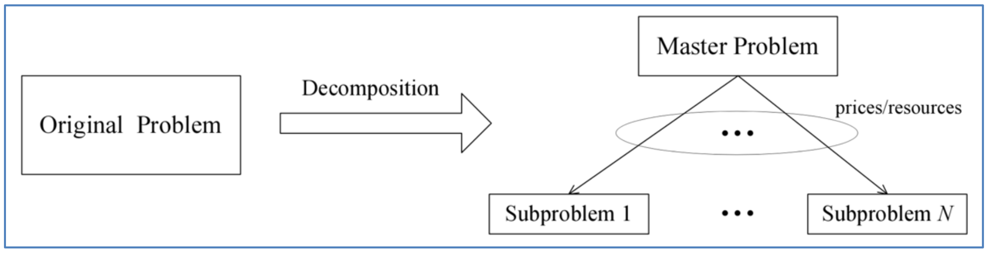


Figure 2.18: The process of decomposing a system into sub-systems (Lasdon 1970), (Palomar & Chiang 2006)

The IBM OPL flow control mechanism is designed to implement the decomposition in large optimization programs. It allows to decompose a model into smaller, more manageable models and solve them with the control of the master model (Arkalgud & Rux 2018); therefore, it will generate the solution of the original model. The flow control uses OPL script to define the master problem and sub-problems. The master problem transfers results between sub-problems using the OPL script pre-processing and post-processing capacities. Our network slicing problem in chapter 6 named Latency-aware resource optimization for 5G core network slicing will use the IBM OPL flow control as a decomposition mechanism.

2.8 Queueing theory and performance modelling

Queueing theory and its application in performance modelling are leveraged to build the performance model of network slicing in chapter 6 named Latency-aware resource optimization for 5G core network slicing. The road map to explore queueing theory is presented in Figure 2.19 with the probability model and overview, Markov chains, M_t/M/1queue and M_t/G/1 queue.

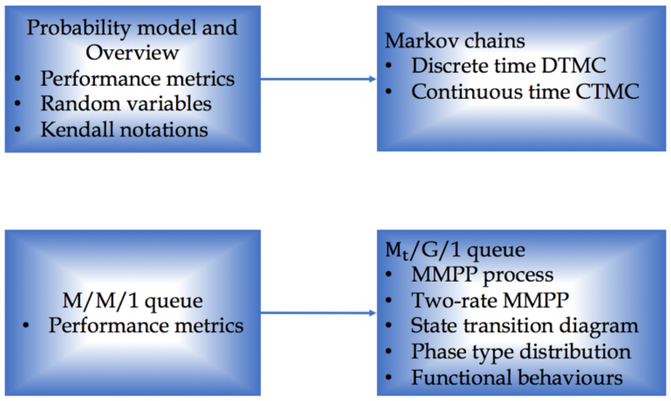


Figure 2.19: the road map to explore queueing theory

2.8.1 Probability models and overview

Queueing theory investigates the queueing problem when there are many jobs, scarce resources, hence, delays and long waiting for services. It is the theory answering questions about what makes queues appear, and how to make queues disappear. “Queueing theory is the study of queueing behaviors in networks and systems. A queueing network is made up of servers” (Harchol-Balter 2013).

2.8.1.1 Performance metrics

A simplest queueing network consists of one server; arriving jobs are waiting to be served by the server, then exit. The performance metrics in single server systems are the job response time, job waiting time, number of jobs in the system and number of jobs in the queue as follows.

2.8.1.2 Job response time

Job response time T is the defined as

$$T = t_{depart} - t_{arrive}$$

t_{arrive} is when the job arrives at the system, t_{depart} is when the job leaves the system after being served. E(T) is the mean response time.

2.8.1.3 Job waiting time

Job waiting time T_Q is the time the job is waiting in the queue, not being served. $E(T_Q)$ is the mean waiting time. The mean response time is the total of the mean waiting time $E(T_Q)$ and mean service time $E(S)$ that the server serves the job:

$$E(T) = E(T_Q) + E(S)$$

2.8.1.4 Other metrics

- ***Number of jobs in the system (N)*** includes the jobs waiting in the queue and the jobs are being served
- ***Number of jobs in queue N_Q*** is jobs waiting in the queue
- In the simple queue network, the ***job arrival rate λ*** jobs / second, and the ***job service rate μ*** jobs / second, ***the assumption that $\mu > \lambda$*** is important to avoid congestion and job loss.

The arrival queue and service queue are modelled based on probability models of discrete and continuous random variables. The probability theory is referenced from (Harchol-Balter 2013).

2.8.1.5 Random variables

A random variable (r.v) is a function of the outcome of an experiment in real value. An r.v can be discrete or continuous. *A discrete r.v can take countable number of values, each with a probability.* The ***probability mass function*** (p.m.f) p_X of r.v X is defined as follows.

$$p_X(a) = P\{X = a\}, \qquad where \sum_x p_X(x) = 1$$

The ***cumulative distribution function*** (c.d.f) of r.v X is defined as follows.

$$F_X(a) = P\{X \leq a\} = \sum_{x \leq a} p_X(x)$$

It is also written using the opposite condition ($X > a$):

$$F_X^1(a) = P\{X > a\} = \sum_{x > a} p_X(x) = 1 - F_X(a)$$

$Poisson(\lambda)$ is a very common discrete distribution in a computer system. If X is a $Poisson(\lambda)$, its p.m.f p_X is defined as follows.

$$p_X(i) = \frac{e^{-\lambda}\lambda^i}{i!}\ with\ i = 0,1,2 \ldots$$

Continuous r.v can take on an uncountable number of values. The range of a continuous r.v is a collection of intervals on the real line. The probability of a continuous r.v is defined via a ***probability density function*** (p.d.f), which is a non-negative function f_X, as follows.

$$P\{a \leq X \leq b\} = \int_a^b f_X(x)dx \quad and \quad \int_{-\infty}^{\infty} f_X(x)dx = 1$$

The ***cumulative distribution function*** (c.d.f) F_X of a continuous r.v X is defined based on the p.d.f, as follows.

$$F_X(a) = P\{-\infty \leq X \leq a\} = \int_{-\infty}^{a} f_X(x)dx$$

$$F_X^1(a) = P\{X > a\} = 1 - F_X(a)$$

$Exp(\lambda)$ is an exponential distribution with rate λ, if X is $Exp(\lambda)$; its c.d.f is calculated as follows.

$$f_X(x) = \begin{cases} \lambda e^{-\lambda x}, x \geq 0 \\ 0, \quad x < 0 \end{cases}$$

$$F_X(x) = \int_{-\infty}^{x} f_X(x)dx = \begin{cases} 1 - e^{-\lambda x}, x \geq 0 \\ 0, x < 0 \end{cases}$$

The discrete $Poisson(\lambda)$ and the continuous $Exp(\lambda)$ are applied to the Poisson process that is described in the M/M/1 queue.

2.8.1.6 Kendall notation

Kendall notation (Bolch et al. 2006) is used in the queueing modeling to identify the behaviors of the arrival queue, service queue and number of servers, as follows.

A/B/m – queueing discipline

A is the distribution of inter-arrival time, B is the distribution of service time, $m \geq 1$ is the number of servers in the computer system, queueing discipline is how jobs are served at the server. The symbols which are used for A and B are as follows:

- M for exponential distribution (memoryless).
- G for General distribution
- GI for General distribution with independent inter-arrival times

For example, M/M/1 is a simple single server system, in which the arrival and the service pattern are Markovian. M_t/G/1, in which M_t denotes the Markov modulated Poisson process (MMPP), models the time varying arrival rates based on the m states of an independent Markov chain (Fischer & Meier-Hellstern 1993).

2.8.2 Markov chains

Markov chains are classified into Discrete-time Markov Chain (DTMC) and Continuous-time Markov Chain (CTMC). The Markov chain theory that consists of definition and theorem are referenced in (Harchol-Balter 2013).

2.8.2.1 Discrete-Time Markov Chain (DTMC)

A stochastic process is a sequence of r.v. DTMC is a stochastic process $\{X_n, n = 0,1,2 \ldots\}$ where X_n is the state at time step $n, \forall n > 0, \forall i, j, \forall i_{n-1}, .. i_0$,

$$P\{X_{n+1} = j \mid X_n = i, X_{n-1} = i_{n-1} \ldots, X_0 = i_0\} = P\{X_{n+1} = j \mid X_N = i\} = P_{ij}$$

where P_{ij} is independent of time steps $X_0, X_1, .., X_{n-1}$.

Markovian property states that the conditional distribution of future state X_{n+1}, is independent of past states $(X_0, X_1, .., X_{n-1})$, and is only dependent on the present state X_n.

The transition probability matrix associated with a DTMC is a matrix P, whose (i,j)th entry $P_{i,j}$ is the probability of moving to state j in the next transition from the current state i.

Limiting probability: let $\pi_j = \lim_{n\to\infty} P_{ij}^n$

***π_j* is the limiting probability** that the chain is in state j (independent of starting state i). For an M-state DTMC, with states 0,1, ..,M-1,

$$\boldsymbol{\pi} = (\boldsymbol{\pi_0}, \boldsymbol{\pi_1}, .., \boldsymbol{\pi_{M-1}}) \;\; where \sum_{i=0}^{M-1} \pi_i = 1$$

represents the limiting distribution of being in each state.

A probability distribution $\pi = (\pi_0, \pi_1, .., \pi_{M-1})$ is said to be ***stationary for a DTMC*** if

$$\pi P = \pi \;\; and \sum_{i=0}^{M-1} \pi_i = 1$$

Theorem: stationary distribution =limiting distribution

Given a finite state DTMC with M states, let $\pi_j = \lim_{n\to\infty} P_{ij}^n > 0$

be a limiting probability of being in state j and let

$$\boldsymbol{\pi} = (\boldsymbol{\pi_0}, \boldsymbol{\pi_1}, .., \boldsymbol{\pi_{M-1}}) \;\; where \sum_{i=0}^{M-1} \pi_i = 1$$

be the limiting distribution. Assuming the limiting distribution exists, then π is also the stationary distribution and no other stationary distribution exists.

A Markov chain, in which the limiting probabilities exist, is said to be stationary if the initial state is chosen according to the limiting probabilities.

A periodic of state j is the greatest common divisor (GCD) of the set of integers n, *such that*

$$P_{jj}^n > 0$$

A state is aperiodic if it has period 1. A chain is aperiodic if all its states are aperiodic.

State j is accessible from state i if $P_{ij}^n > 0$ with some $n > 0$. States i and j are communicate if i is accessible from j and vice versa.

A Markov chain is irreducible if all its states communicate to each other.

Theorem: Given an aperiodic, irreducible, finite state DTMC, with transition matrix P, as $n \to \infty$, $P^n \to L$, L is the limiting matrix π, the vector π has all positive components, summing to 1.

2.8.2.2 Continuous-Time Markov Chain (CTMC)

CTMC has all the properties of DTMC except that the transition between states can happen at any times, only the present state matters, and transition probabilities are stationary and independent of time step.

A CTMC is a stochastic process that every time it enters state i, the amount of time that the process spends in state i is exponentially distributed with some rate called ν_i. When the process leaves state i to enter state j with probability p_{ij}, which is independent of the time spent in state i.

Theorem of CTMC:

Given an irreducible CTMC, suppose $\exists \pi_i, \forall j$:

$$\pi_j \nu_j = \sum_i \pi_i q_{ij} \quad and \; \sum_i \pi_i = 1$$

π_i are the limiting probabilities for the CTMC, and the CTMV is ergodic.

2.8.3 M/M/1 queue

M/M/1 is the simplest queueing model with a single server when the service distribution is an exponential distribution with rate μ, the arrival rate is an exponential distribution with rate λ, to have a stable system: $\lambda < \mu$. The number of customers in the system form a CTMC.

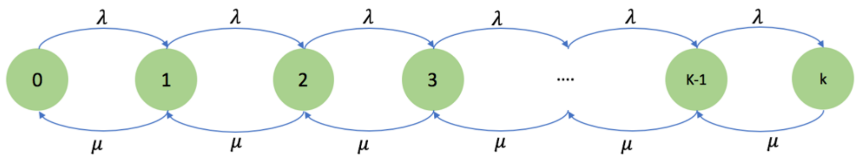


Figure 2.20: CTMC of the customer number in M/M/1 queue, adapted (Harchol-Balter 2013)

The structure in Figure 2.20 is called the birth-death process. The performance metrics of M/M/1 queue are presented as follows.

- ***The server utilization*** is calculated based on arrival rate λ and service rate μ:
$$\rho = \frac{\lambda}{\mu}$$
- ***The mean number of customers in the system***:
$$E(N) = \frac{\rho}{1-\rho}$$
- ***The mean response time***:

$$E(T) = \frac{E(N)}{\lambda} = \frac{1}{\mu - \lambda}$$

- ***The mean waiting time***:

$$E(T_Q) = E(T) - \frac{1}{\mu} = \frac{\rho}{\mu - \lambda}$$

2.8.4 M_t/G/1 queue

2.8.4.1 Definition

M_t/G/1 queue denotes a queueing system with an arrival queue that is a non-Markov arrival process that arrival rate varies randomly over time, a single server that the service distribution is general distribution. Markov-modulated Poisson process (MMPP) has been used to model the M_t arrival queue. MMPP is the doubly stochastic Poisson process, in which the arrival rates of $\lambda_0, \lambda_1,,, \lambda_{m-1}$ vary based on the states of the underlying continuous time Markov Chain (CTMC). MMPP is bivariate with $[J(t), N(t): t \in T], N(t)$ is the number of arrivals within a time interval $(0, t]$ $t \in T$, and $J(t)$ with $0 < J(t) < m$ is the state of CTMC. The MMPP can be reconstructed by changing the arrival rate of a Poisson process according to the m-state of a CTMC, which is independent of the arrival process (Fischer & Meier-Hellstern 1993).

MMPP is parameterized by the m-state CTMC with its generator matrix Q, and the m arrival rates $\lambda_0, \lambda_1,,, \lambda_{m-1}$. In matrix Q, $\alpha_{ij}, i \neq j$ denotes the rate that the Markov process changes from state i to state j,

$$\alpha_{i,j} = \lim_{\tau \to 0} (\frac{P[X(t+\tau) = j] | P[X(t) = i)]}{\tau}), i \neq j$$

Matrix Q has the format:

$$Q = \begin{bmatrix} -\alpha_0 & \alpha_{01} & \alpha_{0m} \\ \alpha_{10} & -\alpha_1 & \alpha_{1m} \\ \alpha_{m0} & \alpha_{m1} & -\alpha_m \end{bmatrix}$$

$$\alpha_i = \sum_{j=0, j \neq i}^{m} \alpha_{ij} \,;\ \alpha_{ii} = -\alpha_i$$

And α_i, the element in diagonal, is equal to the total rates. The holding time of state i is exponential distributed with rate α_i (Nelson 2013). Q is calculated using the steady-state vector π of the Markov chain: $\pi.Q = 0 \ \ and \ \pi.e = 1 \ , \ with \ e = (1,1,..,1)^T$ is an m-element vector.

$\wedge$ is a diagonal matrix of $(\lambda_0, \lambda_1, .., \lambda_m)$, and λ is the transpose vector of $(\lambda_0, \lambda_1, .., \lambda_m)$

$$\wedge = diag(\lambda_0, \lambda_1, .., \lambda_m)$$

$$\lambda = (\lambda_0, \lambda_1, .., \lambda_m)^T$$

2.8.4.2 Two-rate MMPP

We apply a two-rate MMPP as the arrival model of the arrival queue: λ_0 has the holding time value $Exp(\alpha_0)$, and λ_1 has the holding time value $Exp(\alpha_1)$, the arrival rates vary between λ_0 and λ_1, but we can add more rates if we need more accuracy. Figure 2.21 presented the arrival queue arrival process as an MMPP process with arrival rates λ_0, λ_1; MMPP stays at rate λ_0 for a period of $exp(\alpha_0)$ and changes to λ_1 and stays there for $exp(\alpha_1)$ before it changes back to λ_0.

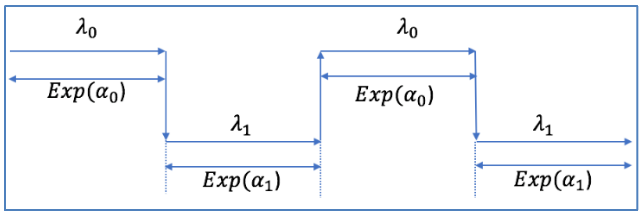


Figure 2.21: The arrival process is a Markov-modulated Poisson process with 2 states, adapted (Harchol-Balter 2013)

2.8.4.3 State transition diagram of MMPP process

The MMPP process with two arrival rates λ_0, λ_1 starts at state 00, when packet arrives with rate λ_0 it transfers to state 01, and then 02, …; when the server processes the packet with service rate μ, it moves back one state to the previous state. The state transition of the process is in the similar manner with arrival rate λ_1 with state 10, 11, 12, ... In the meantime, α_{01} triggers the transition from arrival rate λ_0 to arrival λ_1; α_{10} triggers the transition from arrival rate λ_1 to arrival λ_0; the service rate μ triggers the state transition of the process within the same arrival rate. Figure 2.22 shows a state transition diagram of a $M_t/G/1$ queue system with arrival rates λ_0, λ_1.

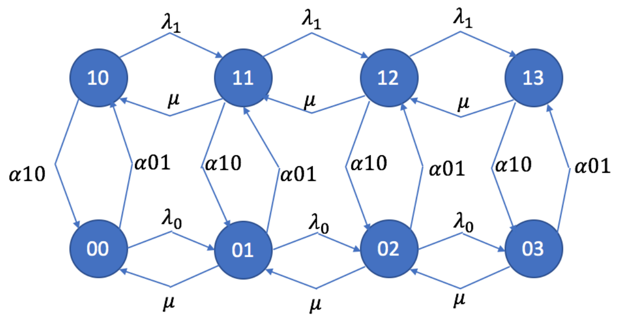


Figure 2.22: State transition diagram of an MMPP model with two states (Bolch et al. 2006)

2.8.4.4 Phase-type distribution and the matrix analytic method

Harchol-Balter (2013) proposed a technique called the method of phases, in which all distributions can be represented accurately by a mixture of exponential distributions called phase-type distribution (PH). Firstly, the varying arrival rates are modeled as a set of exponential distributions with different rates λ_0, λ_1. Then, the matrix analytic method is applied to find the transition matrix of the underlying CTMC, which is repeated and grows unboundedly in one direction. As in Figure 2.23, in the arrival process there are two rates λ_0, and λ_1; at each level i, there will be two states i_0 and i_1, the limiting distribution $\overline{\pi_\iota}$ will be based on π_i^0 for rate λ_0, and π_i^1 for rate λ_1:

$$\overline{\pi_\iota} = (\pi_i^0, \pi_i^1)$$

$\overline{\pi_\iota}$ is recursively expressed in term of π_{i-1} using the $\mathcal{R}$, such that

$$\overline{\pi_\iota} = \overline{\pi_{\iota-1}}.\mathcal{R}$$

$$\overline{\pi_\iota} = \overline{\pi_0}.\mathcal{R}^i$$

the limit distribution can be expressed in term of its level components:

$$\bar{\pi} = (\pi_0^0, \pi_0^1, \pi_1^0, \pi_1^1, \pi_2^0, \pi_2^1, \dots)$$

The generator matrix Q is defined in the matrix:

$$\vec{\pi}.Q = \vec{0} \quad in\, which \quad \vec{\pi}.\vec{1} = \vec{\pi}$$

Q is expressed in the matrix in the M/M/1 queue with arrival rate λ and service rate μ as follows.

$$Q = \begin{bmatrix} -\lambda & \lambda & \\ \mu & -(\mu+\lambda) & \lambda \\ & \mu & -(\mu+\lambda) \end{bmatrix}$$

Q expressed for M_t/M/1 queue is a matrix with multiple blocks, each block is like the Q matrix in M/M/1, for an arrival rate λ_0, λ_1, α_{01}, α_{10}, and μ. By recursively calculating $\pi_0^0, \pi_0^1, \pi_1^0, \pi_1^1, \pi_2^0, \pi_2^1, \ldots$ with Q to receive the limit distribution $\bar{\pi}$. The MMPP performance metrics are calculated based on Q as well. The matrix analytic method is used for calculation and not for problem formulation.

2.8.4.5 Functional behaviors

Gupta et al. (June 2006) proposed an emerging approach to analyze and explore the functional behaviors of the fluctuating system in terms of the input parameter: the rate of varying between high load and low load, how other parameters: arrival rate, service rate affect the performance metrics and which parameters are important. This approach is very suitable for the network slicing problem formulation in chapter 6.

2.9 P4 language and SDN 2.0

P4 language and its application In-band network telemetry are explored as a monitoring tool in the SDN substrate network running segment routing in chapter 5 named Realization of CEVNE virtual link embedding.

2.9.1 What is P4 language

P4 is the short form of Programming Protocol-Independent Packet Processors. P4 was designed with three main goals (Bosshart et al. 2014):

- ✓ the ability for software or hardware switches to change how they process packets,
- ✓ switches are not tied to any specific protocol,
- ✓ the ability to describe independently the functions of packet processing, free from hardware.

P4 language includes these main components as follows: headers, parsers, tables, actions, and control programs. P4 headers specify a series of field values in the packet headers. P4 parsers decide how valid header fields are identified. P4 tables describe how header fields are matched and the consequent actions; there is only one action for each match, there are multiple tables that are processed in a pipeline. The table pipeline specifies the orders to process packet headers as each table is designed for a specific header of the packet. P4 de-parser function is in reverse of the parser function, in which packet headers are merged together to send to egress nodes. According to ONF, P4 language is one of the main components in SDN 2.0

2.9.2 The in-band network telemetry

One of the most popular application of P4 language is the in-band network telemetry (INT). INT is a framework to capture network states in the data plane. There are three types of components in the INT framework: the INT source, INT sink and the transit nodes. The INT source can be any node, device, application or event in the network; it gives instructions to collect metadata for the transit nodes along the forwarding path. INT sink collects the metadata to process and remove them from the packet headers. INT is designed to handle network trouble shooting, congestion control, routing and data plane verification. INT is designed for the data plane state encapsulation for different network protocols such as VXLAN, GENEVE, NSH, TCP, UDP. Figure 2.23 presents an INT application example. INT application works on the network path from host A to host Z via virtual switch 1, hardware switch L, spine switch 0, hardware switch R, and virtual switch 2. The virtual switch 1 is setup as an INT source to give instructions to collect metadata (Kim et al. 2016), which is the latency field. The virtual switch 2 is setup as the INT sink that sends metadata to the analytics engine. The switches along the paths are INT transition hops, which add their latency into the packet headers. At the virtual switch 2 (INT sink), all latency fields in the header are sent to the analytics engine, also removed from the packet (Kim et al. 2016).

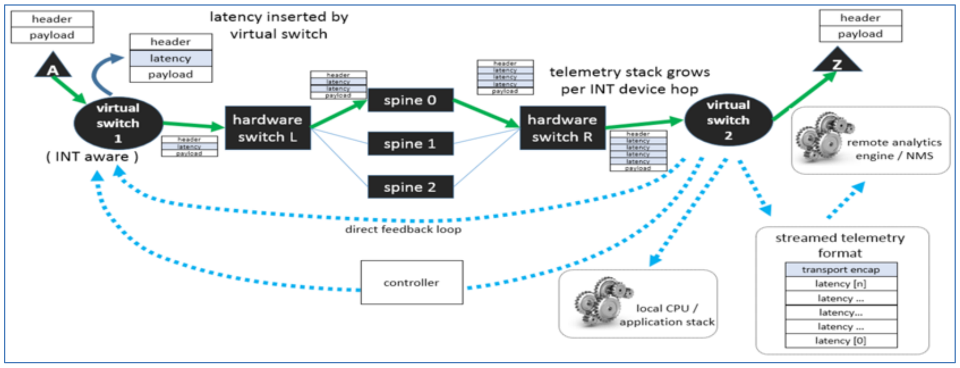


Figure 2.23: INT use case to collect latency in details (Hira & Wobker 2015)

2.10 Summary

In this chapter, the literature review is conducted. SDN principles, SDN architecture, SDN controllers, SDN switches, SDN applications, and Mininet as an SDN network simulation have been described. The review in NV, NFV, network slicing in 5G has been conducted. The background of optimization, mathematical modelling, MSA, queueing theory, and P4 language are presented. In the next chapter, the architecture of extended SDN applications will be investigated.

CHAPTER 3 AN INTENT SERVICE-BASED (ISB) ARCHITECTURE FOR SDN APPLICATIONS

3.1 Introduction

In the previous chapter, a literature review has been conducted focusing on the background theory of SDN, network virtualization, NFV, SFC, network slicing, queueing theory, and 5G core mobile networks. This chapter research objective is to investigate and develop a new architecture to realize extended network applications via the connections to NBI in SDN controllers; to investigate and develop a new architecture to realize 5G custom core network services.

In the chapter, section 2 explores the connection between SDN NBI, and the applications in the SDN architecture; section 3 presents the novel architecture in detail; section 4 focuses on the novel architecture for custom 5G core services; section 5 is the prototype of the Dynamic resource allocation (DRM) application that is realized using the proposed architecture, and section 6 is the summary. The roadmap of the chapter is described in Figure 3.1.

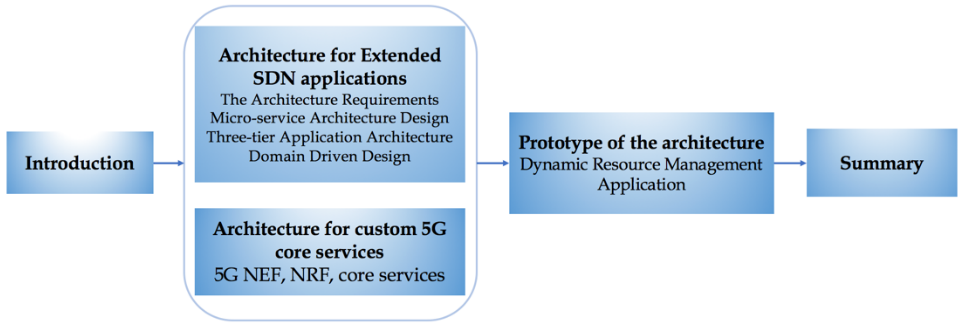

Introduction
Architecture for Extended SDN applications
The Architecture Requirements
Micro-service Architecture Design
Three-tier Application Architecture
Domain Driven Design
Architecture for custom 5G core services
5G NEF, NRF, core services
Prototype of the architecture
Dynamic Resource Management Application
Summary

Figure 3.1: The roadmap of the Chapter 3

SDN adoption has become a norm in the computer networking. Central Office re-architected to Data center (CORD) is a large-scale project at Open Networking Foundation (ONF) to transform telecom central office into cloud service centers. In CORD, the optical layer and the packet-switched layer are converged in the topology using ONOS controller, such innovation makes it easy to calculate the network path, and create new services for customers (Peterson 2019). PEN is a proposed application at Telstra that combines SDN applications with service provisioning so users can request network connections between two locations, and obtain billing information at the same time (ODL 2019).

These business applications represent a class of emerging applications that require network support to some degree, the real task is to provision required services through the management of its underlying network resources. Currently, ad-hoc and self-explored approaches are not able to support the emerging group of applications because of the lack of functionality (Paul et al. 2014) including service partitioning, network and service-context checking, service-composition functionalities, etc. Therefore, this chapter proposes an architecture that not only supports traditional and emerging applications, but also facilitate service creation, and service composition to provision the user requests in a systematic way, which encourages modular designs, and facilitates software reuse. The proposed architecture for extended SDN applications has been presented at the 2016 International Network Softwarization (2016 Netsoft) conference in Seoul, South Korea.

What is new in the research?

The architecture for extended SDN applications promotes a systematic way to provision user request. The architecture promotes service reuse, service composition, service creation and modular design in the three-tier architecture: the database tier, business tier and user interface tier. To discover existing services, the architecture applies the MS design patterns: service discovery and service self-registration. The domain driven design (DDD) is preferred to other design methodologies as it applies the ‘divide and conquer’ strategy that is suitable to MS design. The main business domain is divided into sub-domains to search for the solutions. These solutions are developed as micro-services, which are then composed into the main service to provide the required function. The prototype showed that the solution is effective, realizable and practical for emerging applications and it allows developers to concentrate on developing applications.

3.2 NBI and application layer in SDN architecture

3.2.1 NBI: Intent, Flow rules, Flow objectives

SDN controller offers the NBI to applications to utilize network resources in the SDN networks.

In ONOS controller, three abstractions: Intent API, flow rules API and flow objective API (in the NBI) are used to facilitate the development of SDN applications (ONF 2019a). Each of them is used in its specific scenarios as presented as follow.

- ✓ The flow rule API connects to devices via Openflow, Netconf, and server in the SBI,
- ✓ The flow objective API is working with the device pipelines (single table pipeline, OF-DPA pipeline),
- ✓ the Intent framework provides models for different types of network connections including wired, wireless and optical.

Each of the abstractions is a sub-system that translates into the Openflow (OF) flow rules (OF switches) or P4 executable program (P4 programmable switches) and installed on the SDN hardware switches (ONF 2019a). The translation of abstractions is encapsulated inside the SDN controller as it involves the critical handshake between the controller and the SDN devices that is specified by the OF protocol. These abstractions are in their hierarchy of abstracts, in which the intent framework is in the highest level of abstract, next is the flow objective framework, and last comes the flow rule framework. Figure 3.2 presents the three abstractions according to their level of abstraction.

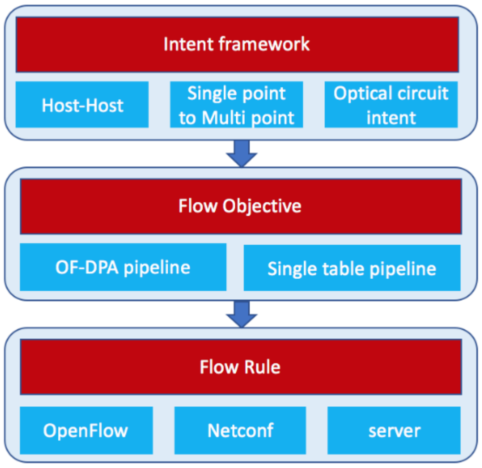


Figure 3.2: Intent, flow objective, and flow rule in NBI, adapted from (ONF 2019a)

Figure 3.3 presents the three abstraction in the ONOS controller NBI. The ONOS controller is also structured into layers including the NBI (with the above abstractions), distributed core services, device drivers, shared protocol libraries (OF, gNMI, P4Runtime), and network devices.

In each SDN application, the invoke of either the intent, the flow objective or the flow rule is the required step so that the application can utilize network resources of the SDN devices.

The intent framework is designed with the purpose to allow applications specifying networks as polices instead of low-level rules. Each intent has its own life cycle that is presented in chapter 2.

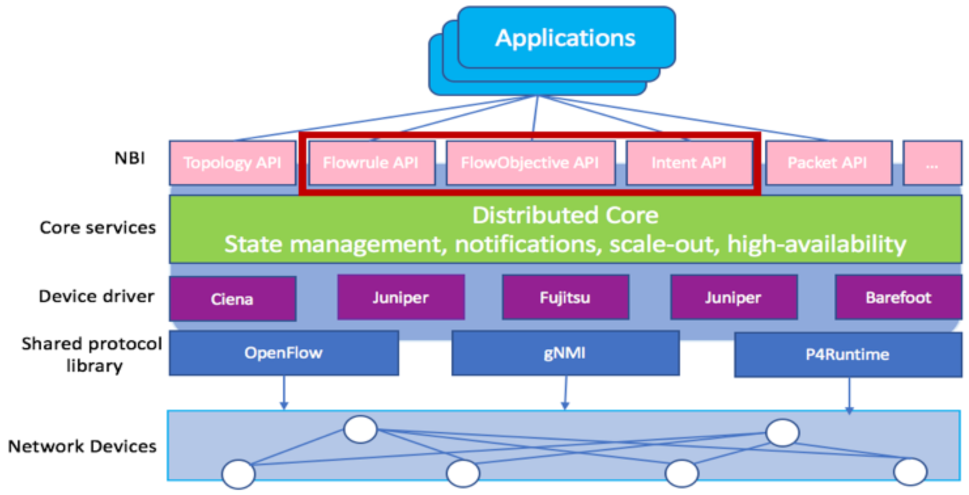


Figure 3.3: ONOS controller and the NBI with the abstractions (ONF 2019a)

The intent framework is used to demonstrate the internal translation mechanism. It is important to select the right intent for the network connections in the application. The compilation process could be failed if new policies conflict with existing ones on the devices, resulting in developers need to update the intent to resolve the conflict (ONOS 2019b) . Figure 3.4 summarizes the three main use cases “create intent”, “update intent” and “withdraw intent” of the intent framework in ONOS.

- ✓ In the “create intent” use case, an intent is created and submitted via the NBI, the intent framework then compiles, stores and installs flow rules on devices via the SBI.
- ✓ The “update intent” use case is raised by changes in the network environment themselves, the monitoring system identified the change and triggered an event to recompile the intent.
- ✓ In the “withdraw intent” use case, an intent is withdrawn. The user needs to provide the intent id and the application id to identify the intent; the intent framework will remove the intent together with its flow rules from the devices, and from the store.

The internal process presented (below the NBI) in Figure 3.4 is invisible to users as ONOS provides these functions via the intent framework. The flow rule API and flow objective API operate in the similar way of creating, updating and withdrawing flow rules (flow objectives) using its compilation as the intent API.

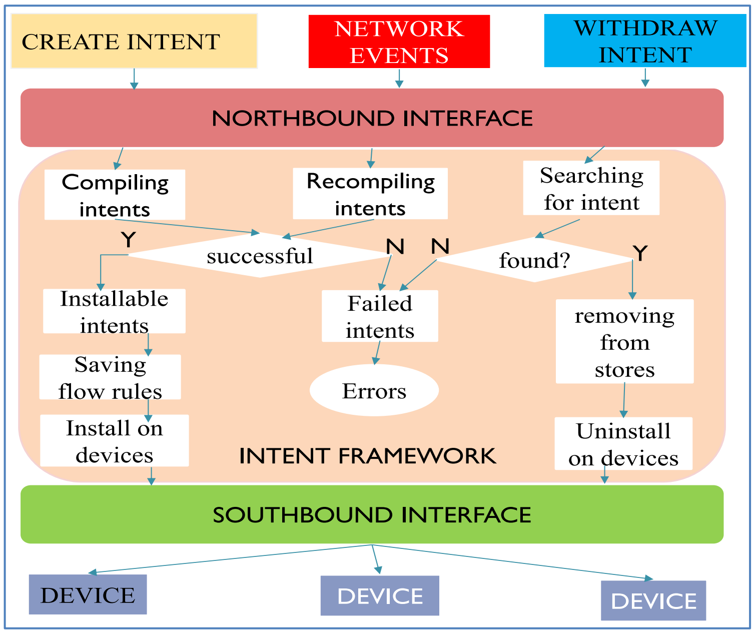


Figure 3.4: Use cases of the ONOS intent framework

3.2.2 The application layer

SDN applications in the application layer leverage the NBI services and the core services to utilize network resources as they need (Figure 3.3). NBI hides the complexity in the process that translates the application into OF rules. The SBI hides the process of choosing the proper device driver for the SDN switch, and the handshake process between the controller and the device to write / update the OF rules into OF tables in the switches. The SDN applications algorithm is in the developers' thoughts, hence, it is ad-hoc, and every application is not fully aware of reuse existing services in their domain.

The application intent framework, ODL NBI or Floodlight NBI, and others facilitate application development somewhat, but they are inadequate for constructing extended, emerging applications and services because they lack functionality for service partitioning, service composition, service context checking. Therefore, the support to handle complex business logic is required on top of the SDN controller. In the next section, the proposed solution architecture is presented.

3.3 The proposed solution architecture

The proposed solution must satisfy all requirements of the architecture for extended SDN applications that are presented as follows.

3.3.1 The requirements of the architecture

Our proposed solution for an architecture is to satisfy all requirements as follows.

- ✓ The extended applications and services need to respond to dynamic changes in the application context.
- ✓ The architecture should ensure the separation of the application environment from the controller environment so that applications will not interfere with the controller.
- ✓ It must be able to handle the composability attribute via the service composition.
- ✓ In all scenarios, the architecture should allow the creation of new services, reuse of existing services, and composition of (new and existing) services into new applications.
- ✓ The extended applications and services can implement complex service composition based on policy, configuration or performance requirements.
- ✓ An application built with the proposed architecture should be user-friendly with different user interfaces: Command Client Interface (CLI), Rest API or service interface.
- ✓ As a production system, it needs to ensure availability, scalability and modifiability.
- ✓ It can integrate with a workflow application to become part of a business process management (BPM) system, or a vertical-stack integration with other applications.

For the architecture, the main architecturally significant requirements (ASR) (Bass 2013), is the composability as it emphasizes the divide-and-conquer solution approach, enables modular design, and reuse of components to compose applications. Figure 3.5 visualizes the architecture from the user requirement view. Some requirements are linked together to handle complex business rules, they allow new-service creation, service reuse, and service composition or service composition engine.

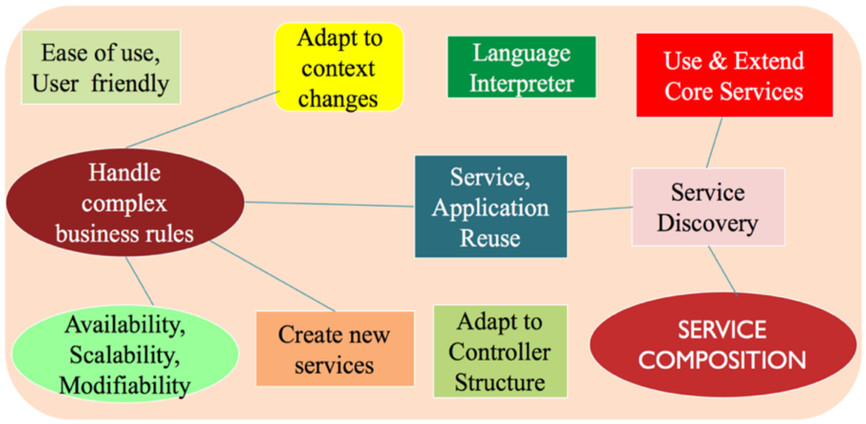

Ease of use, User friendly
Adapt to context changes
Language Interpreter
Use & Extend Core Services
Handle complex business rules
Service, Application Reuse
Service Discovery
Availability, Scalability, Modifiability
Create new services
Adapt to Controller Structure
SERVICE COMPOSITION

Figure 3.5: Visualized the requirements of the architecture

3.3.2 The proposed solution architecture

In this section, the mechanism that satisfies all the above requirements is investigated and the results are presented as follows.

3.3.2.1 Micro service architecture (MSA) design

MSA is part of the service-oriented architecture (SOA) that promotes the service-oriented patterns: service provider, service consumer, service discovery (Fowler 2014), (Richardson 2018); and allows the flexibility in managing applications. In chapter 2 named Related works, section 2.5, the MSA, and MS patterns are presented.

The main strength of MSA is its service composability, allowing the construction of fully functioning applications by integrating atomic services; each service provides a function, based on some business patterns. Besides it, other important MS design principles, which are leveraged to facilitate the development of robust, modular, and reusable applications in the proposed architecture, are presented as follows.

- ✓ Data decentralised principle: in the architecture, each application has its own database management system, to ensure application independence from data perspective.
- ✓ Componentized application: the application comprises micro-service components to promote reuse of existing services and applications.
- ✓ Application design robustness: to ensure the application is robustness, the research considers both successful and failed scenarios.

- ✓ Distributed principle: each application can run in their own process so they will not interfere with each other and with the controller.

Emerging SDN controllers are built in MSA so their environment offers services that cover the MS design pattern. In Figure 3.6, the MSA composition attribute and decentralized data management attribute are visualized.

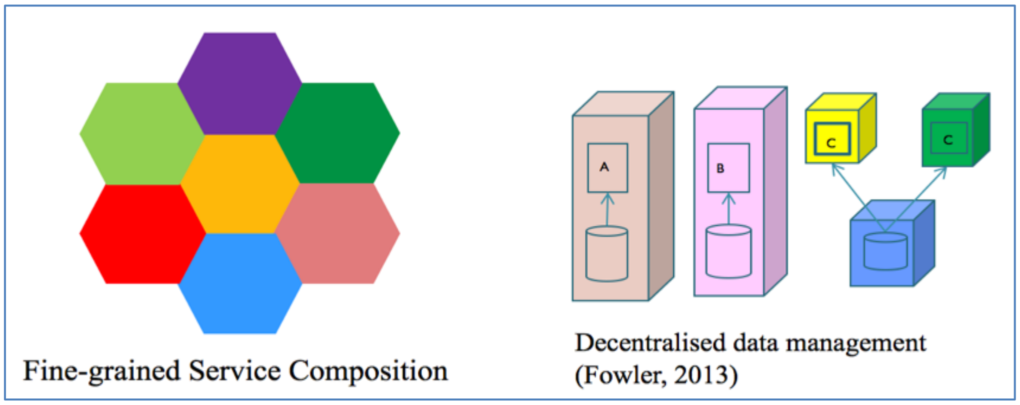

Fine-grained Service Composition
A
B
C
C
Decentralised data management
(Fowler, 2013)

Figure 3.6: The characteristics of Micro-services (Fowler 2014)

3.3.2.2 The three-tier application architecture

The three-tier architecture is proposed for extended SDN network applications to promote modular design, and flexibility as services in each tier are independent to update their components without affecting other tiers; concurrently they work cohesively to deliver results. The three-tier architecture, which is applied the module architecture pattern as presented in subsection 2.6, consists of the database tier, the business logic tier and the presentation tier. The services accommodated in the three-tier architecture address the requirements and serve the development of rich commercial applications. Each tier is presented in detail as follows.

- ✓ The database tier persists application data. It can be a relational database, or a NoSQL database such as a Mongo DB. If the database tier requires significant computing resources, one option is to deploy it in a separate process so it will not affect the SDN application performance. The database MS design pattern, which is presented in subsection 2.5.2.3, can be leveraged in this tier.
- ✓ The business tier handles complex-business-logic requirements via service creation, service splitting and service composition. The atomic services are the NBI services (intent API, flow objectives API, flow rules API) and SDN core services; they are also utility services as they

offer basic functions for the application to control SDN devices, and they form the basic layer for service orchestration. Depending on the problem domains, new services will be created as base services or composite services for commercial or networking purposes. The composite service element (presented in Figure 3.7) integrates new, existing services, and existing applications based on users' requirements. For commercial services, the integration of services emphasizes on transaction integrity. For networking services, the emphasis is on the correctness of protocols and the satisfaction of QoS requirements.

- ✓ The presentation tier accepts input from users and provides different interfaces of the application: Command Line Interface (CLI), the REST API, and the service API. REST API is related to the web interfaces of the application. CLI is familiar with network operators and related to the features of the controller platform.

Figure 3.7 depicts the three-tier architecture and its components for extended SDN applications on top of the SDN architecture. The topmost is the user interface tier with a Rest API, a service programming API, and a Command Line Interface (CLI). The business tier is in the middle with new services, existing network services, existing applications, and a service registry. Below the business tier is the database tier. Each service component may compose of several micro services as required by its functionality. The business tier is the most important tier as it is where new services and composite services are created. With the proposed three-tier architecture, developers can plug in their service components to the appropriate tier to construct their applications.

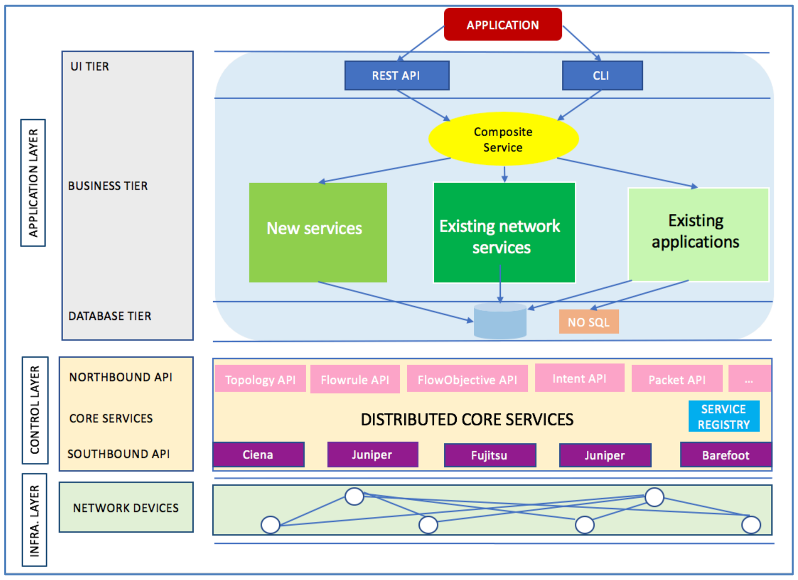


Figure 3.7: Component view of the three-tier architecture for SDN application

3.3.2.3 Domain driven design

In the analysis process, the proposed architecture selects domain driven design (DDD) principle over other software designs as DDD is suitable to the composition attribute in MSA principles (Richardson 2019). Applying DDD, the requirements are decomposed into smallest problem sub-domains (SD), and then the solutions of each sub-domain are built. The solutions may be core services in the controller or existing services; they may also be NBI-enabled new services for the application (Millett 2015).

DDD is about systematic modularization and business-context aware. Figure 3.8 exemplifies the DDD process of the DRM application, the problem domain is divided into three sub-domains: SD1 is about the resource database, SD2 is about the GEANT network, SD3 is about network virtualization. Using the skill and experience of the domain expert, each solution domain is designed in different business contexts that are represented by the database context, GEANT topology, VNE in network virtualization, ONOS controller, and residual resources. The expert and development team provide solutions in each business context.

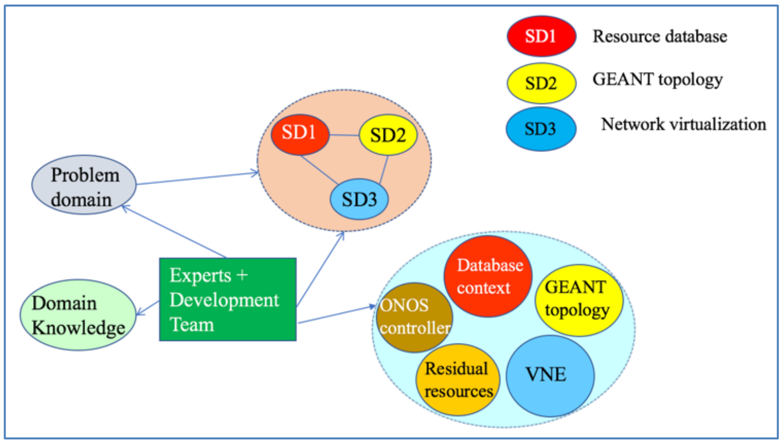


Figure 3.8: Domain driven design example

3.3.2.4 The controller platform

SDN controller offers many services to manage network devices, and process network flows (Hoang & Pham 2015). To support different classes of applications, it is expected that a controller platform to be extensible to support emerging applications and possesses properties as follows.

- ✓ It should manage the life cycle of a service autonomously by introducing, registering, restarting, and terminating the service without restarting the server.
- ✓ It should have its own service registry, service discovery, and built-in functions to create service-based applications.
- ✓ It allows applications to be deployed flexibly in containers or in embedded devices.
- ✓ It should promote performance, modularity, service reuse, and maintainability.
- ✓ It should handle dependencies at compilation time, so developers never have missing classes or objects at runtime.

Popular controllers written in Java like Beacon, ODL Floodlight, and ONOS are built on the OSGi platform. The core architecture of OSGi technology is a dynamic system written in Java (OSGi_Alliance 2016) with different layers as presented in Figure 3.9.

Developers apply the DDD principles to divide the application into logical components called bundles, and develop the bundles using core java language. Each bundle has a manifest file to describe it. In each bundle, the bundle context is the only interface to connect to the OSGi framework. To resolve the dependencies between bundles, OSGi framework allows bundles to export and import for reference. A bundle can be registered as a service via the service registry. A bundle retrieves its service from the registry, and then invokes its services. Life cycle is used to install, start, stop, update and uninstall of bundles and services. Module defines how bundles can import and export code for code dependencies.

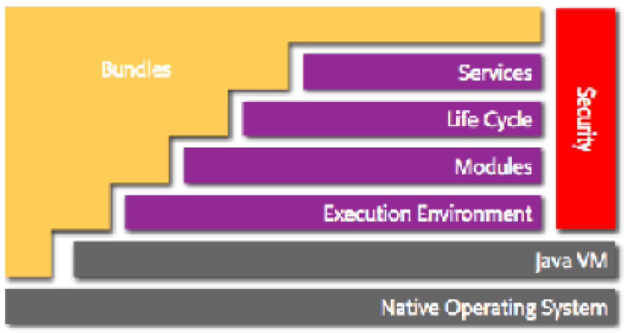


Figure 3.9: OSGi layered architecture (OSGi_Alliance 2016)

Both ONOS and ODL controllers are deployed in Apache Karaf, an OSGi runtime container that provides a rich set of CLI commands to interact with applications and services. This set of commands are extended to create CLI commands for SDN applications. The web GUI is also the extension of the Apache Karaf features.

Choosing the controller that are architected based on the MSA and MS design patterns means that the extended SDN application can leverage these features without building them by itself.

3.4 Proposed architecture for applications in service-based architecture

In this section, the architecture for extended SDN applications is considered for the service-based architecture environment, in which services and applications are composed of micro-services and are exposed via their service APIs. In this environment, the service registry is used to register newly created services, and to lookup existing services for reuse. A mechanism that plays the similar role to the SDN NBI is useful if the environment provides it. Additionally, one important condition dictates that the service composition or decomposition should be based on the programmability of the network infrastructure underneath. This condition is satisfied thoroughly as SDN paradigm has been adopted in almost areas of computer networking; SDN brings programmability, open API, commercial-off-the-shelf (COS) hardware, and reduces vendor lock-in.

Our proposed architecture is applicable to 5G core services as 3GPP facilitates their 5G application architecture to be similar to SDN architecture; 5G core services rely on its enabling technologies: SDN, NFV and cloud computing. We will discuss the application of our proposed architecture in the next section.

3.4.1 Requirements for the 5G custom core services

The architecture of 5G core mobile system is designed as a service-based architecture (TS_23.501 Jun. 2019) with the requirements as follows: service flexibility, easy to add new services, service access control list for each group of users, easy to remove a service, unified policy control, automated network deployment, and modular extendable architecture (Chandramouli, Liebhart & Pirskanen 2019).

3.4.2 Proposed architecture for 5G custom core services

As the main requirements of 5G core architecture emphasizes on service composition and new service creation, our proposed architecture can satisfy these requirements as it is designed based on MSA. The proposed architecture for 5G custom core services are presented as follows.

3.4.2.1 MSA, MS design pattern, and three-tier architecture

MSA and MS design patterns of service composition, service decomposition, service discovery, and service registration are presented in chapter 2 section 2.5.

The three-tier architecture with UI tier, business tier, and database tier are leveraged for the development of custom 5G core services based on the module architecture design pattern (Bass 2013) offering modularity and flexibility; services in each tier are updated independently without affecting to services in other tiers. Each tier is presented as follows.

- ➢ UI tier is used to present the REST interface, service API as it is specified in the architecture of 5G core services (TS_23.501 Jun. 2019).
- ➢ Business tier is where composite services are integrated, and new services are created. The atomic services for 5G custom core services are the 5G core services: authentication and mobility, session management, policy and charging, user plane function, network exposure function, network function repository, unified data management, authentication server function, application function, network slice selection, location management (TS_23.501 Jun. 2019).
- ➢ The database tier is for the two functional groups UDM and UDSF (TS_23.501 Jun. 2019).

5G custom core services will be built on top of 5G core architecture (Chandramouli, Liebhart & Pirskanen 2019). The infrastructure layer consists of 5G core central cloud, and all edge clouds where 5G core services are deployed. 5G core service layer consists of all 5G core services and their functions; the network exposure function (NEF) is the NBI, and the network repository (NRF) is the registry that keeps a record of all services (Penttinen 2019).

At the high level, 5G mobile system consists of the infrastructure layer, which comprises base stations (gNBs), 5G radio access network (RAN), 5G core network (CN) and the transport network (TN) that connects 5G RAN to 5G CN. 5G applications are built on top of 5G core architecture. 5G separates the data plane from the control plane. The custom core services are developed for control plane functions only. The NEF, as the northbound API, provides details of 5G core services to external services and applications. Via NEF, 5G system restricts specific fields of each service to be exposed, so the services encapsulation attribute is guaranteed (TS_23.502 2019). The NRF is used to register services and to retrieve them. The custom 5G core services are the main functional components of the RFB*,* in the emerging NGPaaS, which are presented in chapter 2, section 2.6. Figure 3.10 presents the architecture for custom 5G core services.

When the proposed architecture is applied for custom 5G core services, there are differences that are presented as follows.

- The atomic service and utility services are 5G core services.
- The NEF plays the role of the NBI, the NRF plays the role of the service registry. The NEF needs to consider two scenarios in the home domain and the visiting domain (Penttinen 2019). How NEF and NRF work is completely different from the SDN architecture.
- The infrastructure of the 5G core system is the core cloud and all edge clouds, which are heterogeneous.
- There are no concepts of southbound API as the smallest component in the architecture is an RFB that is deployed in its own infrastructure.
- The custom composite service is a part of the RFB.

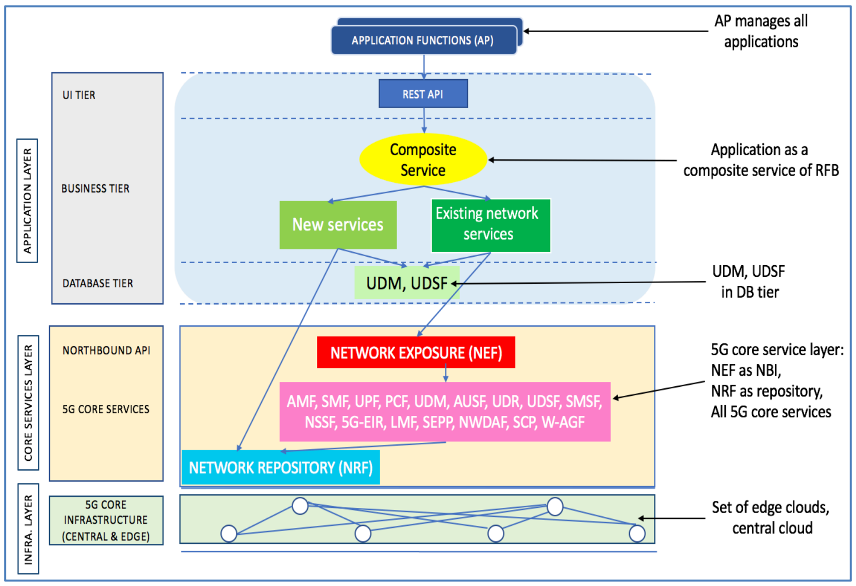

APPLICATION FUNCTIONS (AP)
AP manages all applications
UI TIER
BUSINESS TIER
DATABASE TIER
APPLICATION LAYER
REST API
Composite Service
Application as a composite service of RFB
New services
Existing network services
UDM, UDSF
UDM, UDSF in DB tier
NORTHBOUND API
5G CORE SERVICES
CORE SERVICES LAYER
NETWORK EXPOSURE (NEF)
AMF, SMF, UPF, PCF, UDM, AUSF, UDR, UDSF, SMSF, NSSF, 5G-EIR, LMF, SEPP, NWDAF, SCP, W-AGF
NETWORK REPOSITORY (NRF)
5G core service layer: NEF as NBI, NRF as repository, All 5G core services
INFRA. LAYER
5G CORE INFRASTRUCTURE (CENTRAL & EDGE)
Set of edge clouds, central cloud

Figure 3.10: The proposed architecture for 5G custom core service

In the prototype, to realize the custom 5G core service, a custom mobility service for 5G core system was intended to be implemented as a demo paper.

3.5 The realization of the architecture for extended SDN application

The realization of the proposed architecture consists of the realization of each components, which consist of MSA, MS design patterns, the three-tier architecture, DDD principles, and SDN controller platform, in such a way that all positive attributes of the research are exploited to bring best benefits of the application development.

3.5.1 MSA and MS design patterns

The main purpose of the proposed architecture is to create composite services. An MSA composite service invokes functions of its component services with input parameters. The business requirements determine the context of invoking the component services. The composite service also implements its own algorithms. Although different technologies such as Remote Procedure Call (RPC), SOAP and REST support service composition, REST is selected for the proposed architecture. Services can be synchronous, asynchronous or events based (Newman 2015). When service composition becomes a major activity in the SDN application, a service composition engine is utilized. The service composition engine operates in a lightweight manner (ONOS 2019b) to minimize the associated overheads.

The proposed architecture encourages extendable and modular design. Each MS is designed in its own business domain, to implement its functions, and connect with other services in the same or related domains. Based on REST API, each service needs to provide its interface so other services can invoke its functions. In an MSA design, each service provides a unique functionality so developers can update or replace it independently without affecting other services.

Another important feature of MSA is software reuse. Composite services are designed in such a way that they maximize the reuse of component services. This leads to the creation of different service layers based on their functionalities. The composite service search for the required component service using the service registry, and service lookup activity. If the required service is found, the composite service accesses its functions via the API.

3.5.2 Three-tier architecture

Based on the classification, whether to persist states in the database systems, the service is in the database tier, to implement a business logic, the service is in the business tier, or to gather inputs and present outputs, the service is in UI tier. The front end can be CLI, GUI, and REST services. While a REST API is user-friendly, a CLI is simple to develop based on the interface provided by the OSGi runtime containers. Services in each tier (UI tier, business tier or database tier) need to follow the design framework in their tier. Usually the controller platform provides the design framework for each tier as SDN applications are considered as a part of the SDN architecture.

Figure 3.11 visualizes the process to realize the dynamic resource management application (DRM) using the proposed architecture with REST API and CLI. New services to calculate the residual resources are built. They use the database to store resource capacity of network devices (switches/routers and links), hence the database services are designed. DRM service is the composite service of residual resource services, path intent, and device service.

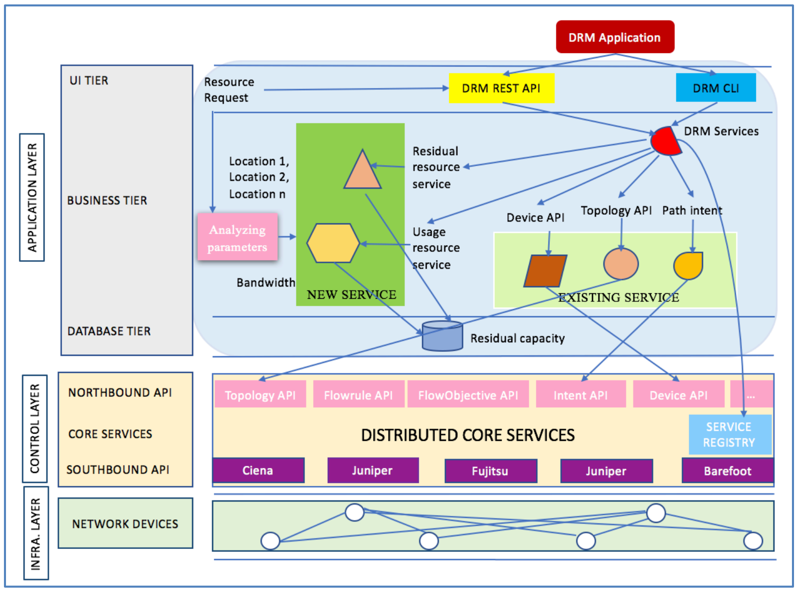


Figure 3.11: The visualization of the process to realize the proposed architecture.

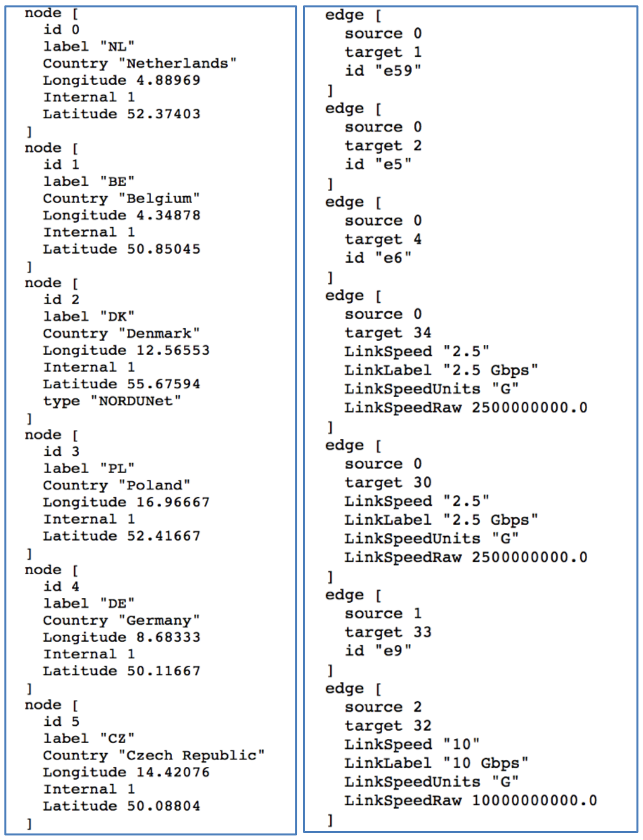
3.6 Prototypes

3.6.1 DRM application prototype

This section describes the implementation of the DRM application on the ONOS controller. It includes the setup of the test bed, which is a GEANT network in Mininet, the setup of the DRM, and the results of DRM application.

3.6.1.1 Test bed setup

The test bed is the GEANT network, which is provided in .gml format (Knight et al. 2011), (University_of_Adelaide 2012) . Each node is denoted with an id, and a label. Each link is denoted with an id, and the source and destination nodes. The node and edge are coded as presented in Figure 3.12 in the GEANT.gml file.



```
node [
  id 0
  label "NL"
  Country "Netherlands"
  Longitude 4.88969
  Internal 1
  Latitude 52.37403
]
node [
  id 1
  label "BE"
  Country "Belgium"
  Longitude 4.34878
  Internal 1
  Latitude 50.85045
]
node [
  id 2
  label "DK"
  Country "Denmark"
  Longitude 12.56553
  Internal 1
  Latitude 55.67594
  type "NORDUNet"
]
node [
  id 3
  label "PL"
  Country "Poland"
  Longitude 16.96667
  Internal 1
  Latitude 52.41667
]
node [
  id 4
  label "DE"
  Country "Germany"
  Longitude 8.68333
  Internal 1
  Latitude 50.11667
]
node [
  id 5
  label "CZ"
  Country "Czech Republic"
  Longitude 14.42076
  Internal 1
  Latitude 50.08804
]
```

```
edge [
  source 0
  target 1
  id "e59"
]
edge [
  source 0
  target 2
  id "e5"
]
edge [
  source 0
  target 4
  id "e6"
]
edge [
  source 0
  target 34
  LinkSpeed "2.5"
  LinkLabel "2.5 Gbps"
  LinkSpeedUnits "G"
  LinkSpeedRaw 2500000000.0
]
edge [
  source 0
  target 30
  LinkSpeed "2.5"
  LinkLabel "2.5 Gbps"
  LinkSpeedUnits "G"
  LinkSpeedRaw 2500000000.0
]
edge [
  source 1
  target 33
  id "e9"
]
edge [
  source 2
  target 32
  LinkSpeed "10"
  LinkLabel "10 Gbps"
  LinkSpeedUnits "G"
  LinkSpeedRaw 10000000000.0
]
```

Figure 3.12: the sample of node and edge in GEANT.gml file

Based on the data in GEANT.gml format, the switches and links are extracted and recreated in python file as a custom topology, which is then uploaded into Mininet code libraries. Figure 3.13 presents the GEANT network in Mininet that has been run in ONOS controller.

In the resource database of GEANT network, switches and links are created in HBase database system. Each switch is denoted by the id, which is used in the GEANT.gml file. So is each link between the source and destination nodes. Some nodes have the capacity of 50 MHz, others have 20 MHz, some links have the bandwidth capacity of 100 Mb/s, others have 250 Mb/s. Figure 3.14 presents the image captured from the resource database.

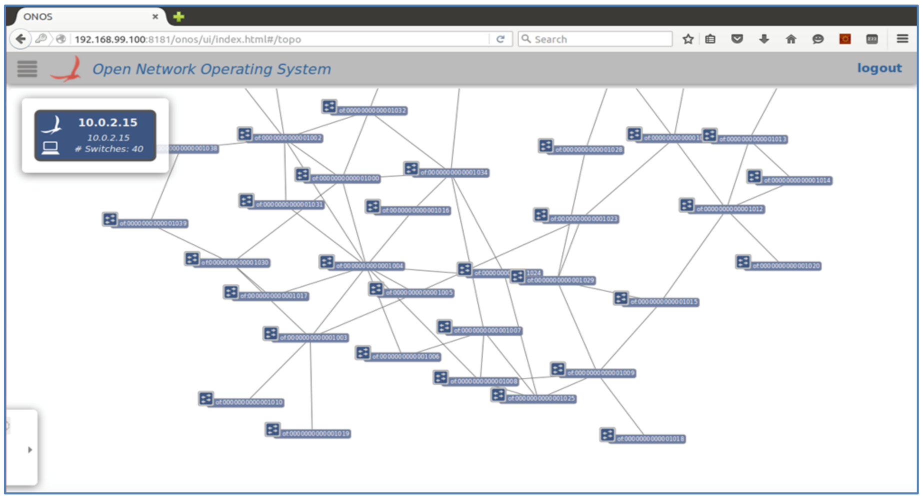
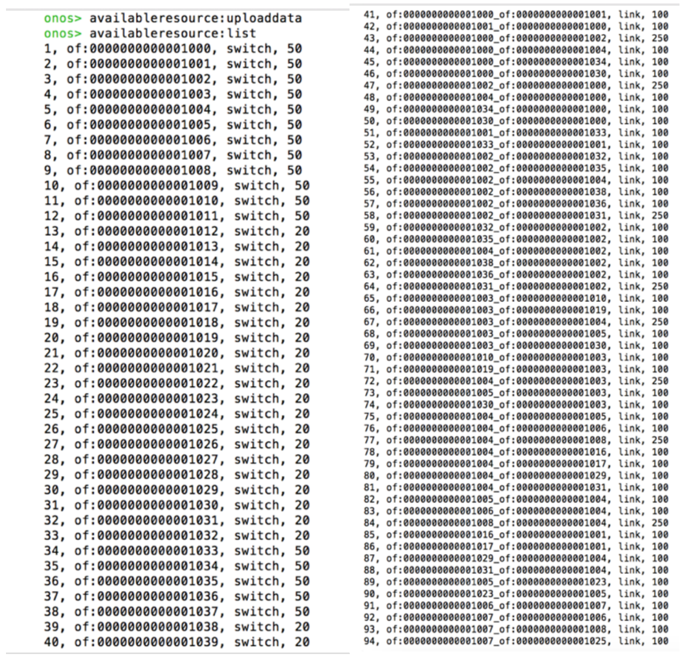

ONOS
192.168.99.100:8181/onos/ui/index.html#/topo
Open Network Operating System
logout
10.0.2.15
10.0.2.15
# Switches: 40

Figure 3.13: DRM running on GEANT topology (Mininet)



```
onos> availableresource:uploaddata
onos> availableresource:list
1, of:0000000000001000, switch, 50
2, of:0000000000001001, switch, 50
3, of:0000000000001002, switch, 50
4, of:0000000000001003, switch, 50
5, of:0000000000001004, switch, 50
6, of:0000000000001005, switch, 50
7, of:0000000000001006, switch, 50
8, of:0000000000001007, switch, 50
9, of:0000000000001008, switch, 50
10, of:0000000000001009, switch, 50
11, of:0000000000001010, switch, 50
12, of:0000000000001011, switch, 50
13, of:0000000000001012, switch, 20
14, of:0000000000001013, switch, 20
15, of:0000000000001014, switch, 20
16, of:0000000000001015, switch, 20
17, of:0000000000001016, switch, 20
18, of:0000000000001017, switch, 20
19, of:0000000000001018, switch, 20
20, of:0000000000001019, switch, 20
21, of:0000000000001020, switch, 20
22, of:0000000000001021, switch, 20
23, of:0000000000001022, switch, 20
24, of:0000000000001023, switch, 20
25, of:0000000000001024, switch, 20
26, of:0000000000001025, switch, 20
27, of:0000000000001026, switch, 20
28, of:0000000000001027, switch, 20
29, of:0000000000001028, switch, 20
30, of:0000000000001029, switch, 20
31, of:0000000000001030, switch, 20
32, of:0000000000001031, switch, 20
33, of:0000000000001032, switch, 20
34, of:0000000000001033, switch, 50
35, of:0000000000001034, switch, 50
36, of:0000000000001035, switch, 50
37, of:0000000000001036, switch, 50
38, of:0000000000001037, switch, 50
39, of:0000000000001038, switch, 20
40, of:0000000000001039, switch, 20
```

```
41, of:0000000000001000_of:0000000000001001, link, 100
42, of:0000000000001001_of:0000000000001000, link, 100
43, of:0000000000001000_of:0000000000001002, link, 250
44, of:0000000000001000_of:0000000000001004, link, 100
45, of:0000000000001000_of:0000000000001034, link, 100
46, of:0000000000001000_of:0000000000001030, link, 100
47, of:0000000000001002_of:0000000000001000, link, 250
48, of:0000000000001004_of:0000000000001000, link, 100
49, of:0000000000001034_of:0000000000001000, link, 100
50, of:0000000000001030_of:0000000000001000, link, 100
51, of:0000000000001001_of:0000000000001033, link, 100
52, of:0000000000001033_of:0000000000001001, link, 100
53, of:0000000000001002_of:0000000000001032, link, 100
54, of:0000000000001002_of:0000000000001035, link, 100
55, of:0000000000001002_of:0000000000001004, link, 100
56, of:0000000000001002_of:0000000000001038, link, 100
57, of:0000000000001002_of:0000000000001036, link, 100
58, of:0000000000001002_of:0000000000001031, link, 250
59, of:0000000000001032_of:0000000000001002, link, 100
60, of:0000000000001035_of:0000000000001002, link, 100
61, of:0000000000001004_of:0000000000001002, link, 100
62, of:0000000000001038_of:0000000000001002, link, 100
63, of:0000000000001036_of:0000000000001002, link, 100
64, of:0000000000001031_of:0000000000001002, link, 250
65, of:0000000000001003_of:0000000000001010, link, 100
66, of:0000000000001003_of:0000000000001019, link, 100
67, of:0000000000001003_of:0000000000001004, link, 250
68, of:0000000000001003_of:0000000000001005, link, 100
69, of:0000000000001003_of:0000000000001030, link, 100
70, of:0000000000001010_of:0000000000001003, link, 100
71, of:0000000000001019_of:0000000000001003, link, 100
72, of:0000000000001004_of:0000000000001003, link, 250
73, of:0000000000001005_of:0000000000001003, link, 100
74, of:0000000000001030_of:0000000000001003, link, 100
75, of:0000000000001004_of:0000000000001005, link, 100
76, of:0000000000001004_of:0000000000001006, link, 100
77, of:0000000000001004_of:0000000000001008, link, 250
78, of:0000000000001004_of:0000000000001016, link, 100
79, of:0000000000001004_of:0000000000001017, link, 100
80, of:0000000000001004_of:0000000000001029, link, 100
81, of:0000000000001004_of:0000000000001031, link, 100
82, of:0000000000001005_of:0000000000001004, link, 100
83, of:0000000000001006_of:0000000000001004, link, 100
84, of:0000000000001008_of:0000000000001004, link, 250
85, of:0000000000001016_of:0000000000001001, link, 100
86, of:0000000000001017_of:0000000000001001, link, 100
87, of:0000000000001029_of:0000000000001004, link, 100
88, of:0000000000001031_of:0000000000001004, link, 100
89, of:0000000000001005_of:0000000000001023, link, 100
90, of:0000000000001023_of:0000000000001005, link, 100
91, of:0000000000001006_of:0000000000001007, link, 100
92, of:0000000000001007_of:0000000000001006, link, 100
93, of:0000000000001007_of:0000000000001008, link, 100
94, of:0000000000001007_of:0000000000001025, link, 100
```

Figure 3.14: The data in the DB table

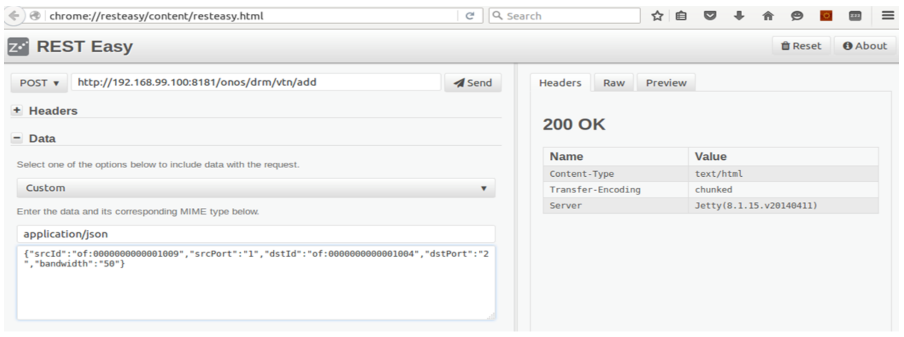
3.6.1.2 DRM application setup

Mijumbi et al. (2014) proposed a solution to network virtualization that emphasized the efficient management of network resources. In a data center scenario, users' requests for virtual networks (VN) came randomly, and the mapping to the substrate network created the required virtual networks. This gradually depleted switches and links resources. If resources were not well managed, they would be too fragmented that further users' request would be rejected; this led to revenue loss. (Mijumbi et al. 2014) used Floodlight SDN controller to provision virtual networks. It used a resource database that records the available resources of the substrate networks, and the VN resource usage. Each VN was provisioned based on the least cost path calculation as follows: of all available paths, the one with least usage ratio, which is the ratio of the usage resource over the available resource, was chosen. The authors showed that the DRM solution improved the request acceptance ratio by 40%.

The proposed architecture and DDD design principles are applied to implement the DRM application. As presented in Figure 3.8 about the domain driven design, the DRM requirements were decomposed into the networking sub-domain and resource management sub-domain. The networking requirement is realized using the Path Intent and the topology service. The resource management requirements are realized using a resource database, which is used to store the available resources of the substrate network as presented in the test bed setup, and to store usage resources of virtual networks. The residual resources are stored in the cache during the virtual mapping process and they are updated to the database when the VNR is completed. OSGi µservices and built-in features, and OSGi containers are used for the service registry and service discovery. ONOS core services are built using OSGi µservices so they are always registered and are discovered in the registry. DRM has REST API and CLI as interfaces in the UI tier. Using DRM Rest API, input parameters were entered in Json format as in Figure 3.15.

The two endpoints were of: 1009 and of: 1004, the required bandwidth was 50 units. The DRM application returned the least cost path between the above two endpoints via the third switch of: 1008 as specified in Figure 3.16, in which the intent was created.

The resource of the substrate network was used as variables for the testing: the virtual network created between two switches was varied when resources of network elements along its path were changed.


chrome://resteasy/content/resteasy.html

POST http://192.168.99.100:8181/onos/drm/vtn/add

Select one of the options below to include data with the request.

Custom

Enter the data and its corresponding MIME type below.

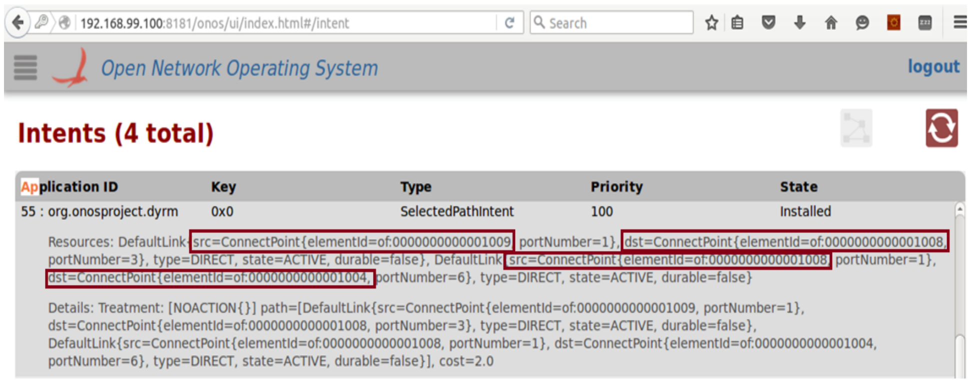
application/json

```
{"srcId":"of:0000000000001009","srcPort":"1","dstId":"of:0000000000001004","dstPort":"2
","bandwidth":"50"}
```

200 OK

Name	Value
Content-Type	text/html
Transfer-Encoding	chunked
Server	Jetty(8.1.15.v20140411)

Figure 3.15: DRM REST interface


192.168.99.100:8181/onos/ui/index.html#/intent

Intents (4 total)

Application ID	Key	Type	Priority	State
55 : org.onosproject.dyrm	0x0	SelectedPathIntent	100	Installed

```
Resources: DefaultLink{src=ConnectPoint{elementId=of:0000000000001009, portNumber=1}, dst=ConnectPoint{elementId=of:0000000000001008,
portNumber=3}, type=DIRECT, state=ACTIVE, durable=false}, DefaultLink{src=ConnectPoint{elementId=of:0000000000001008, portNumber=1},
dst=ConnectPoint{elementId=of:0000000000001004, portNumber=6}, type=DIRECT, state=ACTIVE, durable=false}

Details: Treatment: [NOACTION{}] path=[DefaultLink{src=ConnectPoint{elementId=of:0000000000001009, portNumber=1},
dst=ConnectPoint{elementId=of:0000000000001008, portNumber=3}, type=DIRECT, state=ACTIVE, durable=false},
DefaultLink{src=ConnectPoint{elementId=of:0000000000001008, portNumber=1}, dst=ConnectPoint{elementId=of:0000000000001004,
portNumber=6}, type=DIRECT, state=ACTIVE, durable=false}], cost=2.0
```

Figure 3.16: The intent view

3.6.1.3 Results

The results are presented in two categories as follows: the applying of the proposed architecture and the result of the running DRM on the Geant network.

3.6.1.4 Applying the proposed architecture

Both prototypes and their results showed that it is feasible to implement the proposed architecture to realize extended SDN applications. The architecture promotes modularity via the three-tier architecture pattern, decentralized data management and componentized application thus enhances the modifiability of the architecture. It made the realization of extended application as straightforward as other business applications. It promotes the economic values of service-based architecture via the maximum reuse of existing services that are available in the controller and existing applications. The three-tier architecture promotes a neat architecture as it ensures that applications have their components in their own modules and will not interfere with controller modules. With any type of requirements, the steps in DDD in the architecture can be applied to separate requirements in each sub-domain module, to search for each solution, and to integrate them into the final solution. The application is also robust because the design always concerns with both success and failed scenarios. Finally, using the controller core service as the base layer, it ensures the realization process always can access fully the topology abstraction provided in the controller.

3.6.1.5 Running DRM on ONOS controller

When the DRM application was running on the ONOS controller, the available resources were uploaded into the database. Available resources include the available and usage resources for network elements based on Geant network. They are organized into two separate tables: the available resource table and the usage resource table. REST client was used to run the test cases on the browser, and in ONOS UI to check whether the intent was created. The log data was useful to follow the trace of the program. The actual results were matched with the manual calculations in the expected results and were recorded in the Table 3.1 as follows.

Table 3.1: Test results of running DRM on ONOS

Test case 1 details	Source: switch 09, Destination: switch 04, Bandwidth: 50
Resource setup	ONOS returns two paths between 09–04 Path 1: 09-08-04: average usage / availability ratio: 102/600, Path 2: 09-29-04: average usage / availability ratio: 102/270
Expected result	Path 1 with the least average ratio
Actual result	An intent was created for Path 1
Test case 2 details	Source: switch 00, Destination: switch 31, Bandwidth: 50
Resource setup	ONOS returns two paths between 00–31 Path 1: 00-04-31: average usage / availability ratio: 102/200 Path 2: 00-02-31: average usage / availability ratio: 102/600
Expected result	Path 2 with the least average ratio
Actual result	An intent was created for Path 2
Test case 3 details	Source: switch 04, Destination: switch 12, Bandwidth: 50
Resource setup	ONOS returns one path between04-12 Path 1: 04-29-15-12, all links are >=100

Expected result	Path 1 should be returned
Actual result	An intent was created for Path 1

It should be emphasized that the approach that implemented DRM in the Floodlight controller was ad-hoc, self-explored, and did not add value for reuse. Our proposed solution is systematic, it allows developers to concentrate efforts on the application development, it encourages modular designs, and facilitates software reuse. The selection of the DRM application is specifically to demonstrate the generality of the proposed platform.

3.6.2 Prototype of CEVNE LiM application

The realization of CEVNE LiM is presented in detail in chapter 5 named Realization of CEVNE virtual link embedding, which consists of the heuristic algorithm for the virtual link embedding process, and the realization of the heuristic algorithm using the proposed architecture presented in section 3.3 of this chapter. Please refer to chapter 5 for further detail.

3.7 Summary

In this chapter, the research objective is to investigate and to propose a mechanism to provision users' network requests in a systematic and modular way. The proposed architecture for extended SDN applications and custom 5G core services are presented, which is based on MSA, MS design patterns, and the three-tier architecture. For extended SDN applications, the proposed architecture is based on SDN architecture, SDN controller core services, and the SDN controller platform. The prototype, which is the realization of the DRM application, showed that the proposed architecture is appropriate, effective, realizable and practical for extended, business-like network applications for SDN networks. For custom 5G core services, the research is based on 5G core networks and infrastructure, and 5G core services.

CHAPTER 4 CONGESTION-AWARE ENERGY-AWARE VIRTUAL NETWORK EMBEDDING

The previous chapter objective was to investigate and develop a novel architecture to realize extended SDN applications, which is based on MSA and the three-tier architecture. The proposed three-tier architecture for extended SDN applications is built on top of SDN controllers using emerging MSA and MS design patterns, domain driven design, and service composition on the OSGi platform, which is used to deploy SDN controller.

This chapter objective is to investigate and to develop novel mechanisms for resource optimization in NV in cloud data centers. The research aim is to investigate and to build a novel mathematical program for the VNE that focuses concurrently on multiple objectives of saving cost, saving energy and congestion avoidance, which is called CEVNE. The concurrent objectives are selected as they are among the main factors of an optimally operational environment in substrate networks; the testbed is based on the leaf-spine fabric for cloud data centers. The chapter roadmap is presented in Figure 4.1.

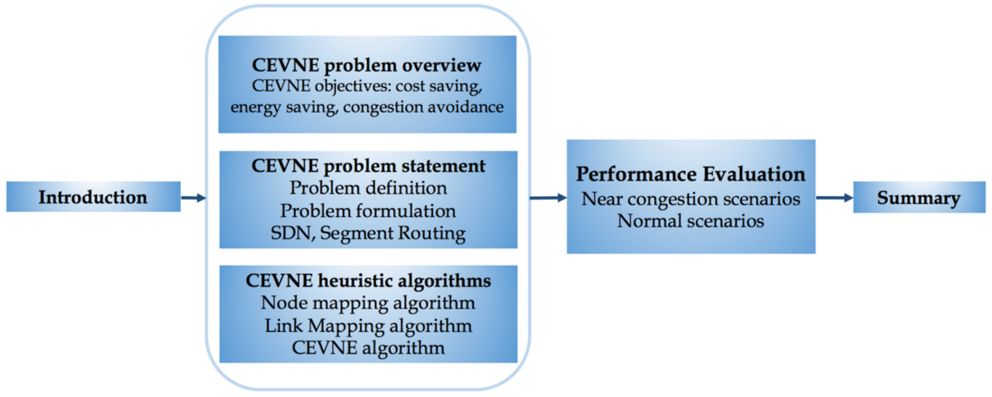


Figure 4.1: The roadmap of chapter 4

In this chapter, section 1 presents the overview of the novel CEVNE; section 2 presents the CEVNE problem statement that consists of problem definition, problem formulation with multiple objectives, constraints, and enabling technologies of SDN and SR; section 3 presents novel CEVNE heuristic solutions with node embedding, link embedding, and the integrated algorithms; section 4 presents the performance evaluation: CEVNE congestion control objective is beneficial in scarce-resource, near-congestion networks; and CEVNE node embedding performance results;

and section 5 summarizes the chapter. The CEVNE research work has been published in the IEEE/ACM Transaction on Networking, Vol. 28, Issue 1, Feb. 2020, pp 210-223.

What is new in the solution?

SDN brings the changes to network virtualization in almost three areas as follows: the substrate network is an SDN network, the SDN network paradigm will enhance to the node and link embedding processes, and the network operation facilitates the node and link embedding processes with the flexibility that SDN offers.

In this chapter, we concentrate on all three aspects although the focus is on the resource allocation optimization. The solution focuses concurrently on three objectives: cost saving, energy saving and congestion avoidance. The research investigates the multi-objective optimization problem and applies the rules to generate the pareto-optimal solutions for multi-objectives (Cohon (1978) and Williams (2013)). The problem formulation of the CEVNE problem is as follows: formulating each objective, formulating the combined objective of the problem, in which the weighting method with positive weights is applied to the cost saving and energy saving objectives, and the constraint method with binding constraint is applied to the congestion control objective. The novelties also lie in the nodes and link embedding algorithms, and the applying the congestion ratio using the hose traffic demand model (Oki 2012). In the evaluation, the results show that the congestion avoidance objective is beneficial in the embeddings of near-congestion scenarios. In normal scenarios, the CEVNE NoM outperforms in runtime by 50% and in energy saving by 20% compared to the State-Of-The-Art.

4.1 CEVNE Overview

Table 4.1 presents all the notations used in section 4.1 except the sub-section about the max of congestion ratio.

Table 4.1: Notations and their explanations

$\boldsymbol{G_S = (N_S, E_S)}$	the substrate network modelled as an undirected graph G_S with substrate nodes N_S and substrate links E_S
$\boldsymbol{G_V = (N_V, E_V)}$	the virtual network request modelled as an undirected graph G_V with virtual nodes N_V and virtual links E_V
$\boldsymbol{n_s}$	a substrate node, $n_s \in N_S$
$\boldsymbol{e_s}$	a substrate link, $e_s \in E_S$
$\boldsymbol{c(n_s)}$	the CPU capacity of $n_s \in N_S$

$\boldsymbol{bw(e_s)}$	the bandwidth capacity of $e_s \in E_S$
$\boldsymbol{n_v}$	a virtual node, $n_v \in N_V$
$\boldsymbol{e_v}$	a virtual link, $e_v \in E_V$
$\boldsymbol{b(e_v)}$	the bandwidth demand of $e_v \in E_V$
$\boldsymbol{c(n_v)}$	the CPU demand of virtual node $n_v \in N_V$
$\boldsymbol{bw(e_v)}$	the bandwidth demand of $e_v \in E_V$
$\boldsymbol{R_N(n_s)}$	the CPU residual capacity of $n_s \in N_S$ after n_s embeds virtual nodes of several VNRs
$\boldsymbol{P_S}$	all substrate paths in G_S
$\boldsymbol{p_s(s,t)}$	all substrate paths between nodes s and t; $s,t \in N_S$, $p_s(s,t) \in P_S$
$\boldsymbol{p_s(e_s)}$	all substrate paths passing substrate link $e_s \in E_S$; $p_s(e_s) \in P_S$
$\boldsymbol{M: G_V \to G_S}$	the embedding of the VNR onto the substrate network
$\boldsymbol{M = (M_N, M_L)}$	the embedding process consists of node embedding M_N and link embedding M_L
$\boldsymbol{R_{rev}(G_V)}$	The revenue received when the VNR G_V is embedded.
$\boldsymbol{R_E(e_s)}$	the residual bandwidth of substrate link e_s after e_s embeds virtual links of several VNRs.
$\boldsymbol{R_E(p_s(e_s))}$	the bandwidth residual of substrate path p_s is equal the minimal residual bandwidth of substrate links connecting into the path $p_s(e_s)$: $R_E(p_s(e_s)) = \min\limits_{e_s \in p_s} R_E(e_s)$
$\boldsymbol{bw(e_v, e_s)}$	the bandwidth of $e_s \in E_S$ allocated to $e_v \in E_V$
$\boldsymbol{i}$	the index of virtual links, $1 \le i \le \lvert E_V \rvert$
$\boldsymbol{bw(e_v^i)}$	The bandwidth demand of virtual link ith e_v^i
$\boldsymbol{f_{uv}^i}$	the flow of virtual link i on the substrate link $(uv) \in E_S$
$\boldsymbol{s^i, s_i}$	The ith virtual flow's source node
$\boldsymbol{t^i, t_i}$	The ith virtual flow's destination node
$\boldsymbol{c_w(m)}$	the CPU resource of substrate node $m \in N_S$ allocated to virtual node $w \in E_V$
$\boldsymbol{r}$	the network congestion ratio
$\boldsymbol{R}$	the maximal congestion ratio, which is the upper bound of r
$\boldsymbol{x_{uv}}$	a binary variable equal to 1 if $\sum_i (f_{uv}^i + f_{vu}^i) > 0$, otherwise 0
$\boldsymbol{y_n^{uv}}$	a binary variable equal to 1 if $\sum_i (f_{un}^i + f_{nu}^i) > 0$ or $\sum_i (f_{vn}^i + f_{nv}^i) > 0$, otherwise 0
$\boldsymbol{\delta}$	a very small positive number to prevent dividing by 0
$\boldsymbol{\mu(n_v)}$	meta node of virtual node $n_v \in N_V$ has unlimited CPU resources
$\boldsymbol{\Omega(n_v)}$	the cluster of virtual node $n_v \in N_V$
$\boldsymbol{N_{S1}}$	the set of substrate nodes and meta nodes

E_{S1}	the set of substrate links and meta links

4.1.1 The Virtual network embedding problem

4.1.1.1 Substrate networks and virtual networks

A substrate network (SN) is modelled as an undirected graph $G_S = (N_S, E_S)$. Each substrate node $n_s \in N_S$ has a CPU capacity $c(n_s)$; each substrate link $e_s \in E_S$ has a bandwidth capacity $bw(e_s)$.

Users send a virtual network request (VNR) to a service provider to acquire a virtual network (VN). A VN is modelled as $G_V = (N_V, E_V)$. Each virtual node $n_v \in N_V$ has a CPU demand $c(n_v)$; each virtual link $e_v \in E_V$ has a bandwidth demand $bw(e_v)$. Figure 4.2 presents an example of the VNE process (Fischer et al. 2013). Two VNs: VN1 and VN2 are embedded on the same SN.

4.1.1.2 The virtual network embedding

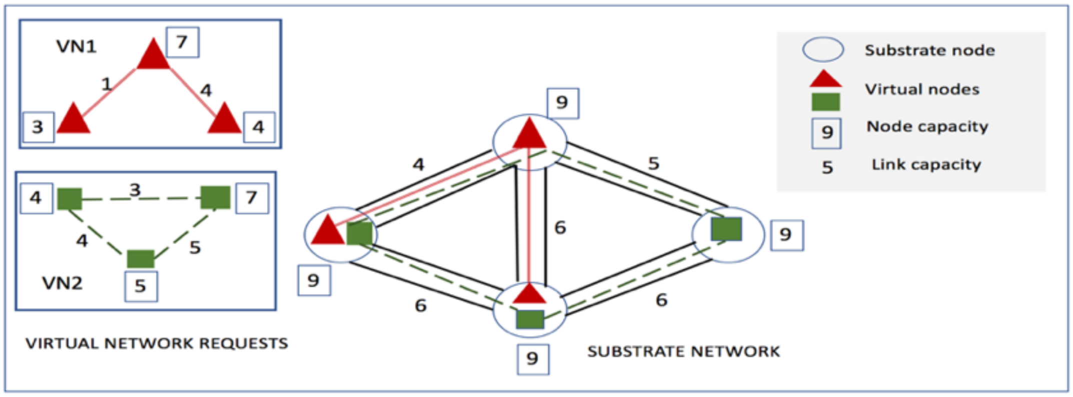


Figure 4.2: An example of the VNE process (Fischer et al. 2013)

The VNE is the embedding of a VNR onto the SN:

$$M: G_V \rightarrow G_S \tag{1}$$

There are two embedding processes, the node embedding M_N and the link embedding M_L: $M = (M_N, M_L)$. Within a VNR, each virtual node n_v can be embedded to only one substrate node n_s as the virtual link e_v connecting the virtual nodes has the bandwidth demand $bw(e_v)$. A substrate node can embed multiple virtual nodes belonging to multiple VNRs if it satisfies the CPU demands. The condition to embed successfully a virtual node n_v on a substrate node n_s is as follows.

$$R_N(n_s) \geq c(n_v) \tag{2}$$

Similarly, a virtual link e_v is embedded successfully onto the substrate path $p_s(e_s)$ based on the condition as follows.

$$R_E\big(p_s(e_s)\big) \geq bw(e_v) \tag{3}$$

According to (Yu et al. 2008), the revenue of a VNR after it has been embedded is defined as follows.

$$R_{rev}(G_v) = \sum_{n_v \in N_v} c(n_v) + \sum_{e_v \in E_v} b(e_v) \tag{4}$$

The revenue of a VNR is the total of substrate resources that have been used to embed its virtual nodes and virtual links. If the VNE supports path splittable mappings, a virtual link can be embedded to multiple substrate paths that have their total demand bandwidth (Yu et al. 2008).

4.1.2 Overview of CEVNE multi-objectives

CEVNE is an online, two-stage coordinated VNE that focuses on the objectives: saving cost, saving energy and avoiding congestion. As an online VNE, CEVNE embeds VNRs in real-time when they arrive using its node and link embedding processes. Each objective is presented as follows.

4.1.2.1 Cost saving objective

CEVNE aims for a least cost solution in terms of SN resources that are used in the embedding VNRs. CEVNE applies the load-balancing resource consumption that is proposed by Chowdhury, Rahman & Boutaba (2012), and is expressed as follows.

$$\min \Big(\sum_{(u,v) \in E_S} \frac{\sum_i \mathrm{f}^{\mathrm{i}}_{\mathrm{uv}}}{\mathrm{R_E(u,v)} + \delta} + \sum_{w \in N_S} \sum_{m \in N_V} \frac{\mathrm{c_w(m)}}{\mathrm{R_N}(w) + \delta} \Big) \tag{5}$$

In the objective function (5), the first term denotes the virtual link demand load balancing over the substrate links residual resources, the second term denotes the virtual node demand load balancing over substrate nodes residual resources. The cost saving objective focuses on minimizing the resource cost (5). Mijumbi et al. (2014) showed that the load balancing approach can reduce the fragmentation in the substrate networks. Minimizing the resource cost is equivalent to maximizing the number of VNRs embedded onto the same SN, hence, the revenue is increased. Although ViNE-LB algorithms support splittable path mappings, CEVNE supports un-splittable path mappings in our SDN-based link mapping algorithm, and the splittable ones are for future work.

4.1.2.2 Energy saving objective

CEVNE focuses on saving energy consumption in the VNE process. This objective is implemented by minimizing the number of active substrate nodes that embed virtual nodes in VNRs. (Fischer, Beck & De Meer 2013), (Rodriguez et al. 2015) showed that the routers consume the major part of the energy in the data center. Inactive nodes will operate in the sleep mode or in the low power idle (LPI) mode (Reviriego et al. 2012) based on IEEE 802.3az. A router power consumption is modeled as a base with no traffic load, and a dynamic component that is dependent on its active interfaces (ports) and traffic loads. CEVNE energy saving objective is expressed using y_n^{uv} as follows.

$$\min \sum_{uv\in E_S} \sum_{n_s\in N_S} y_{n_s}^{uv} \tag{6}$$

4.1.2.3 Congestion avoidance objective

Network congestions imply the conditions that the amount of traffic injected into the network approaches the capacity limits of the network handling resources. They affect severely the performance of the substrate networks and the embedded virtual networks in terms of throughput, response time, and services completion. Different methods are investigated to control the network congestion. The congestion-ratio minimization approach is preferred over the congestion handling based on the link cost (Fortz & Thorup 2000) as CEVNE handles traffic on a flow basis. CEVNE aims to avoid the network congestion by minimizing *r*, the congestion ratio (Medhi 2017), (Ng & Hoang 1987).

$$r = \max_{uv\epsilon E_S} \left(\frac{\sum_i f_{uv}^i + f_{vu}^i}{R_E(uv)}\right) \tag{7}$$

The congestion avoidance objective is to minimize the congestion ratio as follows:

$$\min \quad r \tag{8}$$

subject to:

$$\sum_i f_{uv}^i + f_{vu}^i \le r * R_E(uv) \quad \forall (uv \in E_S) \tag{9}$$

$$f_{uv}^i - f_{vu}^i = \begin{cases} bw(e_v^i) & if\ u = s^i \\ 0 \ \ if \ \ u \neq s^i, u \neq t^i \\ -bw(e_v^i) & if\ u = t^i \end{cases} \tag{10}$$

The constraint (9) is derived from the congestion ratio (7). In constraint (9), the bound of link resources is updated to apply the congestion ratio r times the link capacity. Constraint (10) is the flow conservation as follows: for an intermediate node that is not a source node or a destination node, the total flow into the node is equal to the total flow out of that node.

4.1.2.4 The maximum congestion ratio

The notations presented in Table 4.2 are used only in this sub-section.

Table 4.2: Notations and their explanation related to max congestion ratio

$\boldsymbol{G(V,E)}$	directed network with V nodes and E links
$\boldsymbol{Q \subseteq V}$	the set of edge nodes in G
$\boldsymbol{c_{ij}}$	the link capacity of substrate link $(i,j) \in E$
$\boldsymbol{y_{ij}}$	the total load on substrate link $(i,j) \in E$
$\boldsymbol{t_{pq}}$	the traffic demand from edge node p to edge node q; $p,q \in Q$
$\boldsymbol{r}$	the congestion ratio is calculated as the max link utilization of all links $r = \max_{(ij \in E)} \left\{\frac{y_{ij}}{c_{ij}}\right\}$
$\boldsymbol{T}$	the traffic demand matrix, $\{t_{pq}\}$, $\forall p,q \in Q$ $\quad T = \{t_{pq}\}$
$\boldsymbol{\alpha_p}$	the total of out-going traffic at edge node p for all t_{pq}
$\boldsymbol{\beta_q}$	the total of in-coming traffic at edge node q for all t_{pq}

In the direct network $G(V,E)$, the optimal routing problem seeks to optimize the routing of the traffic between edge nodes $(p,q) \in Q$ that maximizes the link utilization but avoids network congestion. The network demand is given by the traffic demand matrix T. The total traffic of incoming and outgoing at the edge node $p,q \in Q$ is constrained by α_p and β_q respectively.

$$\sum_q t_{pq} \le \alpha_p,\ p \in Q;\ \sum_p t_{pq} \le \beta_q\,, q \in Q$$

The optimal congestion ratio, if it exists, will be used as the benchmark to set the value for R, which denotes the maximum congestion ratio in our VNE problem. We use the hose demand model as the traffic demand in CEVNE is almost unknown except that the total of all virtual network traffic is limited by the substrate link capacities.

4.1.2.5 Modelling techniques and solution approaches

The CEVNE programming model applies the weighting methods on the cost saving and energy saving objectives, and the constraint method on the congestion avoidance objective as presented in subsection 2.7.3. The CEVNE program is a MILP. Solving MILP is computationally intractable (Amaldi et al. 2016); hence, the MILP program is relaxed to a LP, and solved using the GLPK solver to find sub-optimal solutions for the involved objectives. The rounding algorithm in (Chowdhury, Rahman & Boutaba 2012) is used to find the approximation values of the integer values x and y. Our modelling techniques and solution approaches are summarized in Figure 4.3 as follows.

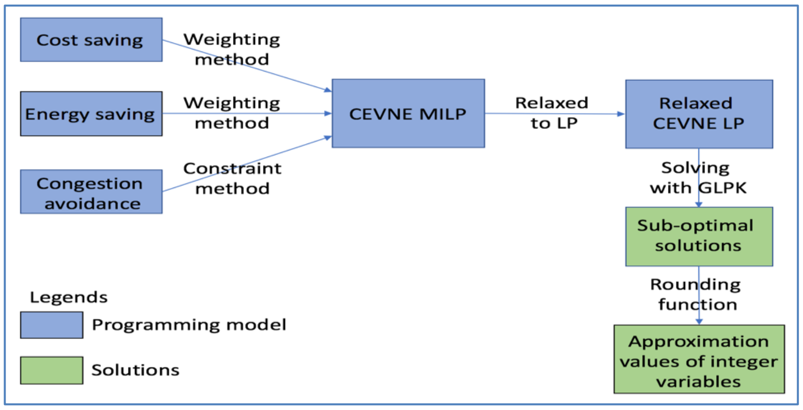


Figure 4.3: The summary of CEVNE modelling techniques and solution approaches

CEVNE is a multi-objective VNE problem with cost saving, energy saving and congestion avoidance objectives. In CEVNE modelling techniques, the weighting method with strictly positive weights, and the constraint method with the binding constraint are applied, so the CEVNE combined programming model will generate the pareto-optimal solutions for all involved objectives (Cohon (1978) and Williams (2013)).

4.1.3 The augmented substrate networks

CEVNE uses the augmented substrate network for the coordination of node and link mapping that is proposed by Chowdhury, Rahman & Boutaba (2012). Each virtual node n_v is used to create a cluster $\Omega(\mathrm{n_v})$ of the substrate nodes that satisfy its CPU demand. A meta node m is created for each virtual node n_v. Meta links are created between the meta node m and all substrate nodes n_s in the cluster.

In the augmented substrate network, the total number of nodes N_{S1} include the meta nodes m and the substrate nodes N_S .

$$N_{S1} = N_S \cup \{m = \mu(n_v) | n_v \in N_V\}$$

The total number of links E_{S1} include the meta links and substrate links E_S.

$$E_{S1} = E_S \cup \{(\mu(n_v), n_s), n_s \in \Omega(\mathrm{n_v})\}$$

4.2 CEVNE Problem formulation

4.2.1 The problem statements

It is given the substrate network (SN) with a set of N_S substrate nodes, E_S of substrate links, and the set of substrate paths P_S:

$$G_S = \{N_S, E_S,\ P_S\}$$

Each substrate node $n_s \in N_S$ has a CPU resource. Each substrate link $(u, v) \in E_S$ has a bandwidth resource.

It is given a virtual network request (VNR) demands N_V virtual nodes and E_V virtual links

$$G_V = \{N_V, E_V\}$$

Each virtual node $n_v \in N_V$ has a CPU demand. Each virtual link $(u, v) \in E_V$ has a bandwidth demand.

It is given that the VNE process consists of two processes of the node embedding and link embedding processes. An augmented substrate network is built from the SN with N_{S1} nodes, and E_{S1} links.

4.2.2 The problem formulation

The CEVNE programming model applies the weighting methods on the cost saving and energy saving objectives, and the constraint method on the congestion avoidance objective (Cohon (1978) and Williams (2013)). The cost saving and the energy saving objectives are both related to the embedded substrate nodes; the weighting method is applied to create a combined program for

CEVNE. Each of the objective terms is normalized to have the same ratio unit and has the base weight applied. These two objectives form the CEVNE objective as follows.

Objective function:

$$\text{minimize} \left(\sum_{uv \in E_S} \left(\frac{1}{R_E(u,v)+\delta}\right) \sum_i f_{uv}^i + \sum_{w \in N_S} \left(\frac{1}{R_N(w)+\delta}\right) \sum_{m \in N_{S1} \setminus N_S} x_{mw} c(m) + \sum_{uv \in E_S} \left(\frac{1}{card(N_S)}\right) \sum_{n_s \in N_S} y_{n_s}^{uv} \right) \quad (11)$$

In the objective function (11), the first two terms include the costs of the virtual links and virtual nodes that are specified as the ratios to load balance. The model selects the links and nodes with the maximum residuals of bandwidth, and CPU resources. The last term tries to limit the ratio of active nodes to the total number of substrate nodes to save energy, $card(N_S)$ is the total number of substrate nodes in the SN. The first two terms have been specified in formulas (5) in simple format without considering the augmented substrate network.

For the congestion avoidance objective, to integrate with the combined model and to form the overall multi-objective formulation, the constraint method is applied. As a constraint, the objective does not appear in the expression of the objective function (11). The congestion avoidance objective is transformed into a constraint by setting an upper limit R on the congestion ratio r. This is a binding constraint if the solution exists. Therefore, the CEVNE programming model will generate the noninferior solutions (Cohon 1978) if they exist.

Subject to:

$$\sum_i (f_{uv}^i + f_{vu}^i) \leq r * R_E(uv) \quad \forall (uv) \in E_S \quad (12)$$

$$0 \leq r \leq R \quad (13)$$

$$0 \leq R \leq 1 \quad (14)$$

In constraint (12), the bound of the link resources is updated to apply the congestion ratio r times the link capacity (Medhi 2017). Constraint (13) denotes the limit of variable r that is R as presented in sub-section 4.1.2.4. Constraint (14) is the limit of R.

Other constraints:

$$R_N(w) \geq x_{mw}\, c(m) \quad \forall m \in N_{S1} \setminus N_S, \forall w \in N_S \quad (15)$$

$$\sum_{w\in N_{S_1}} f^i_{uw} - \sum_{w\in N_{S_1}} f^i_{wu} = \ 0 \ \forall (i,u) \in N_{S_1}\backslash\{s_i,t_i\} \tag{16}$$

$$\sum_{w\in N_{S1}} f^i_{s_iw} - \sum_{w\in N_{S1}} f^i_{ws_i} = \ bw(e^i_v) \quad \forall i \tag{17}$$

$$\sum_{w\in N_{S1}} f^i_{t_iw} - \sum_{w\in N_{S1}} f^i_{wt_i} = -bw(e^i_v) \ \ \forall i \tag{18}$$

$$\sum_{w\in\Omega(m)} x_{mw} = 1 \ \ \forall m \in N_{S1}\backslash N_S \tag{19}$$

$$\sum_{m\in N_{S1}\backslash N_S} x_{mw} \le 1 \ \ \forall w \in N_S \tag{20}$$

$$f^i_{uv} \ \ge 0 \ \ \forall u,v \ \in N_{S1} \tag{21}$$

$$x_{uv} \in \{0,1\} \ \forall u,v \in N_{S1} \tag{22}$$

$$y^{uv}_{n_s} \ \in \{0,1\} \ \ \forall u,v \in N_S \tag{23}$$

Constraint (15) ensures the substrate node CPU residual resource must satisfy the virtual node CPU demand for the virtual node embedding (Chowdhury, Rahman & Boutaba 2012).

Constraints (16), (17) and (18) are about the flow conservation. Constraint (16) ensures that for any internal nodes u, w, which are not the source node s_i or the destination node t_i, the net flow at these nodes is 0. Constraint (17) ensures that the total flow at the source node s_i is equal to the required bandwidth of the virtual link. Constraint (18) ensures that the total flow at the destination node t_i is equal to the required bandwidth of the virtual link in the negative sign.

Constraint (19) and (20) ensure that in the augmented substrate network, only one substrate node w is selected for the meta node m (Chowdhury, Rahman & Boutaba 2012).

Constraint (21) is the limit of the flow f^i_{uv}. Constraints (22) and (23) are the integer constraints of the variable x_{uv} and variable $y^{uv}_{n_s}$ accordingly.

Solving the MIP is computationally intractable. Hence, the MIP is relaxed to a LP so it can be solved in polynomial time. The MIP relaxation is carried out by changing the integer constraints into the linear constraints. Clearly the solutions to the relaxed problem, if they exist, are only sub-optimal solutions. The integer constraints (22) and (23) have been relaxed into the linear constraints as in formulas (24) and (25).

$$0 \leq x_{uv} \leq 1 \quad \forall u, v \in N_{S1} \tag{24}$$

$$0 \leq y_{n_s}^{uv} \leq 1 \ \forall u, v \in N_S \tag{25}$$

Solving the relaxed LP will give sub-optimal solutions if they exist. A heuristic algorithm is required to find the approximation for the integer variables of the original MIP.

4.2.3 CEVNE's enabling technologies

CEVNE's enabling technologies are Software-defined networks (SDN) and Segment Routing (SR) that are presented in chapter 2, section 2.1. Both SDN and SR technologies facilitate the routing on the substrate network at both coarse-grained and fine-grained levels, thus, empower the embedding process.

4.2.3.1 Software-defined networks

The CEVNE investigates the VNE problem in SDN networks. The SDN architecture includes three layers as follows: the data layer consists of Openflow (OF) network devices, the control layer includes SDN controllers, and the application layer consists of applications built on top of the controller. Recently SDN has been applied to traffic engineering (TE) because of its ability to steer the network traffic at both coarse-grained and fine-grained levels (Lee & Sheu 2016).

CEVNE exploits SDN TE solutions in addressing the congestion control objective. Based on the SDN programmability, the virtual link embedding process is realized as a composite service on top of the SDN controller. For routing, OF switches store flow rules in their TCAM. As TE routing algorithms would result in a large number of flow rules in SDN switches, a huge amount of expensive TCAM memory is required. Segment routing protocol is deployed to reduce the memory requirement.

4.2.3.2 Segment routing

Segment routing (SR) is a source routing protocol that allows the packets to be forwarded with minimal retaining of the network states. In SR, a source node specifies the traversal path for a packet in its header using a list of labels or segment identifiers (SID). In SR, routers along the path do not need to store routing rules in the routing tables, they only need to process the labels using simple pop, push or continue operations. SR allows alternative paths between the two nodes. In an

SDN network, SR replaces the flow rules in the TCAM memory with the list of SIDs in the packet header, leading to a reduction in TCAM storage costs. SR also speeds up the process of changing routes or adding new routes to the network (Lee & Sheu 2016). CEVNE exploits the flexibility, agility, programmability of SDN and SR in addressing the congestion control issue in our virtual link embedding solution.

4.3 CEVNE Algorithms

4.3.1 CEVNE node embedding algorithm

The CEVNE node embedding algorithm (CEVNE NoM) solves the relaxed LP to find the sub-optimal solutions (SOPS) in polynomial time if they exist. The relaxed LP is solved with the GLPK in the Alevin framework. In the SOPS, the variables may take on linear values as their integer constraints have been relaxed. The D-ViNE rounding algorithm (Chowdhury, Rahman & Boutaba 2012) is used to convert back these linear values to integer values, approximating the integer variables of the original MIP problem. As the CEVNE has been formulated based on the rules to generate noninferior solutions, the SOPS and their approximations reflect the trade-offs between three objectives.

The input to the CEVNE NoM, which is presented in Figure 4.4, is the CEVNE mathematical model that combines the three objectives, then is relaxed to a LP; the augmented substrate network as presented in section 4.1.3, and the value of R (line 2 - 4). Each virtual node is used to create a cluster of substrate nodes based on CPU demands (line 9 - 11) (Chowdhury, Rahman & Boutaba 2012). GLPK is invoked to solve the CEVNE relaxed formulation (line 12). GLPK returns the sub-optimal solutions, if they exist, with values of (linear and relaxed) variables, and the sub-optimal values of each objective. The approximation for the integer variable x is calculated using the rounding function (line 14), which is in D-ViNE algorithm (Chowdhury, Rahman & Boutaba 2012). In the rounding function, the virtual node's cluster is used to find the substrate node with the highest weight. The weight is calculated as a product of the value of x and the total flow passing the meta link in both directions. The CEVNE NoM returns the embedded substrate nodes of virtual nodes in the VNR.

Figure 4.4: CEVNE node embedding algorithm

```
Input
    CEVNE objective, is relaxed
    The augmented substrate network
    The value of R
Output
    The embedded substrate nodes

Begin
    For virtual node $n_v$ in $N_V$
        Create a cluster for virtual node $n_v$
    End for
    Solve the CEVNE formulation using GLPK
    // the rounding function
    Applying only the rounding function in D-VINE
    Return the embedded substrate nodes
End
```

Figure 4.5: CEVNE link embedding algorithm

```
Input
    Virtual link $(i, j) \in E_V$, its bandwidth demands
    Embedded nodes $(u, v) \in E_S$, the bandwidth capacities
Output
    Embedded substrate paths $p_{u,v} \in P_S$ for virtual link $(i, j)$

Begin
    Retrieve shortest path between $(u, v)$ assign to $P_{u,v}$;
    path = select_path($P_{u,v}$);
    If path is not empty
        Return path;
    End if
    Retrieve other paths between $(u, v)$,
        sort them by length, assign to $P_{u,v}$;
    path = select_path ($P_{u,v}$);
    If path is not empty
```

```
                Return path;
            Else alert “all paths are congested”
                Exit with failure
            End if
        End
        Function select_path ($P_{u,v}$)
            For each $p_{u,v} \in P_{u,v}$
                If $p_{u,v}$ is not congested
                    Return $p_{u,v}$
                End if
            End for
        End function
```

Figure 4.6: CEVNE algorithm

```
Input:
    substrate network topology
    virtual network topology
    virtual node CPU demands
    substrate node CPU capacities
    virtual link bandwidth demands
    substrate link bandwidth capacities
Output:
    A virtual network for a VNR
Begin
  invoke CEVNE node embedding algorithm
  if VNR is rejected, exit
  invoke CEVNE link embedding algorithm
  if VNR is rejected, exit
  register active nodes into the active-node registry
  for substrate node $n \in N_S$
      $\text{residual}_{cpu}(n) = \text{resource}_{cpu}(n) - \text{demand}_{cpu}(n)$
  end for
 for substrate link $l \in E_S$
```

```
20          $\text{residual}_{\text{bw}}(l) = \text{resource}_{\text{bw}}(l) - \text{demand}_{\text{bw}}(l)$
21      end for
22      set inactive substrate nodes to sleep mode
23    End
```

4.3.2 CEVNE Link embedding algorithm

The CEVNE link-mapping algorithm (CEVNE LiM) embeds the virtual links on top of the substrate paths that satisfy the demands and the three objectives. CEVNE LiM utilizes the path services in the SDN controller, and the monitor application to actively select the best path for each virtual link in the VNR. CEVNE LiM is presented in Figure 4.5.

The inputs of CEVNE link embedding algorithm are the virtual links in the VNR, and the embedded substrate nodes (line 2 –3). At first, the path service is invoked to retrieve the shortest paths between nodes (u, v) (line 8). The select path function is called to return the first non-congested path among the shortest paths (line 11), the CEVNE LiM returns with a success status. Otherwise, the path service is invoked with a different weight to retrieve another set of paths between nodes (u, v) (line 13-14). They are sorted by the path length. The select path function is called to return the first non-congested path (line 15) among the second set of paths, the CEVNE LiM returns with a success status (line 17). Otherwise, an alert is sent to the operator, and the function exits with a failure status (line 18 - 19).

This process is repeated for every virtual link in the VNR. If the embedding has failed on a virtual link, the VNR is rejected. The select path function reflects the monitor application checking for congestion on a substrate path (line 22 – 28).

4.3.3 CEVNE embedding algorithm

The CEVNE algorithm includes the CEVNE NoM algorithm and CEVNE LiM algorithm as presented in Figure 4.6. A registry is used to keep track of the active nodes of each VNR. Inactive nodes are filtered using the SDN topology service and are put in sleep mode. First, CEVNE NoM is invoked (line 11). If it is failed, the VNR is rejected (line 12). Otherwise, CEVNE LiM is invoked (line 13); if it is failed, the VNR is rejected (line 14). Otherwise, after each VNR is embedded successfully, the registry keeps track of the active nodes of both the NoM and LiM processes (line 15). The CEVNE algorithm operates in a virtualization platform that will keep track of VNRs that have been processed. The residual resources of substrate nodes are calculated (line 16 - 18), so are the residual resources of substrate links (line 19 - 21). Finally, inactive nodes are set to the sleep mode (line 22).

4.4 Performance evaluation of CEVNE node embedding

In this section, we focus on the evaluation of the CEVNE NoM process. Firstly, the evaluation setup is presented; secondly, the near-congestion scenarios are presented, in which resources are scarce and CEVNE congestion control is beneficial to the VNR embedding; and lastly, the simulation and test scenarios used to evaluate the CEVNE NoM process is presented.

4.4.1 CEVNE evaluation setup

To set up the evaluation for CEVNE, the Alevin framework (Beck et al. 2014) is used as a simulation tool to generate network topologies for both SNs and VNs; and generate network resources and demands for these network elements.

4.4.1.1 Topology generation

Waxman method is used to create SN and VN topology. First, it creates nodes and their coordination in the topology graph. Whether a network link exists between any two nodes is determined by their distance, the values of α and β in the probability P that is calculated as follows (Hesselbach et al. 2016).

$$P(u,v) = \alpha * \exp^{\left(-\frac{d}{\beta * d_{\max}}\right)} \tag{26}$$

$P(u,v)$ is the probability that a connection of two nodes u, v is established, d is the Euclidian distance between u, v; the $d_{\max}$ is the max Euclidian distance between any two nodes. All SN and VN are directional.

For the evaluation, two SNs are set up, a large 50 nodes, and a small 25 nodes. Three types of topologies are designed for each SN based on values of α and β: simple (S), moderate (M) and complex (L); they are applied for large and small SNs. Both SNs and VNs use the same values of α and β. Table 4.3 and Table 4.4 present the parameters of the 50-node network and the 25-node network and their VNs.

Table 4.3: Three topologies of the 50-node network

	Substrate networks		Virtual networks			
50-node	**α, β**	**Links**	**α, β**	**Nodes**	**Layers**	**Links**
Simple (S)	0.5 0.5	1000-1100	0.5 0.5	9	4-6	30-40
Moderate (M)	0.75 0.75	1300-1400	0.75 0.75	9	4-6	50-60
Complex (L)	1.0 0.5	1500-1600	1.0 0.5	9	4-6	65-75

Table 4.4: Three topologies of the 25-node network

	Substrate networks		Virtual networks			
25-node	**α, β**	**Links**	**α, β**	**Nodes**	**Layers**	**Links**
Simple (S)	0.5 0.5	245-275	0.5 0.5	5	4-6	5-9
Moderate (M)	0.75 0.75	320-340	0.75 0.75	5	4-6	10-14
Complex (L)	1.0 0.5	420-450	1.0 0.5	5	4-6	16-20

The size of the substrate networks is selected 25 nodes and 50 nodes; the size of the virtual networks is selected as 5 virtual nodes and 9 virtual nodes. These figures are selected after several pre-tests to ensure that virtual networks can be created in the not nearly congested cases to facilitate the measurements of metrics that are related to three objectives: cost saving, energy saving and congestion avoidance.

4.4.1.2 Resource and demand generation

Alevin framework generates resources and demands randomly within a specified range of the minimum and maximum values. The load ρ is calculated based on the max resource and max demand (Hesselbach et al. 2016) as follows.

$$D_{\max} \quad = \rho * R_{\max} \quad * \left(\frac{V}{k * V_k}\right) \tag{27}$$

$D_{\max}$ is the max demand value, $R_{\max}$ is the max resource value, V is the number of substrate nodes in the SN, k is the number of VNs in a VNR, which is specified as the layers in the range [4-6], and V_k is the number of virtual nodes in each VN; it is assumed that V_k is the same in all VNs in a VNR. To examine the effects of the load on the CEVNE NoM performance, different values of ρ (from light to heavy load) are used in different test cases.

Three types of load are designed based on virtual links bandwidth demands: light load (S), medium load (M) and overloaded (L). The rest of the parameters, CPU resources and CPU demands are the same for all load types. Table 4.5 presents each type of load based on resources and demands; the unit of CPU resource is GHz; the unit of bandwidth resource is Gbps.

Table 4.5: Resource and demands

	Substrate network Resources		**Virtual networks Demands**	
	CPU	**BW**	**CPU**	**BW**
	min/max	**min/max**	**min/max**	**min/max**
Simple (S)	10 / 20	60 / 70	1 / 5	5 / 10
Moderate (M)	10 / 20	60 / 70	1 / 5	10 / 15
Overloaded (L)	10 / 20	60 / 70	1 / 5	15 / 20

4.4.1.3 The congestion ratio parameter

The evaluation networks are varied randomly to ensure the fairness and completeness of the evaluation, so R is calculated using a test network provided in (Oki 2012). It is a dense network (6 nodes and 12 links) with α_p and β_p are set to the value: $\sum_q t_{pq} = \alpha_p; \sum_p t_{pq} = \beta_q$; the traffic demand matrix (on the left) and the traffic capacity matrix (on the right) are as follows.

$$\begin{array}{cccccc} 0 & 35 & 35 & 35 & 35 & 35 \\ 35 & 0 & 35 & 35 & 35 & 35 \\ 35 & 35 & 0 & 35 & 35 & 35 \\ 35 & 35 & 35 & 0 & 35 & 35 \\ 35 & 35 & 35 & 35 & 0 & 35 \\ 35 & 35 & 35 & 35 & 35 & 0 \end{array} \qquad \begin{array}{cccccc} 0 & 100 & 100 & 100 & 0 & 100 \\ 100 & 0 & 100 & 100 & 100 & 100 \\ 100 & 100 & 0 & 0 & 100 & 0 \\ 100 & 100 & 0 & 0 & 100 & 100 \\ 0 & 100 & 100 & 100 & 0 & 100 \\ 0 & 100 & 0 & 100 & 100 & 0 \end{array}$$

The result of the optimal routing ratio in the above traffic and capacity matrices is 0.875; it is used as a benchmark to estimate the value of R such that $0 \leq R \leq 1$. The selected value R is close to its upper bound, so it does not affect the acceptance ratio. We select R=0.945, so it does not violate the bandwidth capacity of the edge; as such, our test results are non-biased. As R is the optimal value of the congestion ratio, we ensure it is never the case that $R > 1$.

4.4.1.4 The simulation environment and scenario generators

Alevin framework is used to generate test scenarios. Each test scenario includes the SN parameters, VN parameters, resource and demand parameters, and the node and link mapping algorithms. Each test scenario is designed as one experiment in Alevin framework with 9 different test cases (three seed values and three-layer values).

4.4.2 Challenging and near congestion experiments

4.4.2.1 The design of near-congestion or challenging scenarios

The network congestion is caused by node congestion, link congestion and both node congestion and link congestion. Node congestion occurs where the node demand is close to the node capacity. Similarly, link congestion occurs when the link demand is close to the link capacity. In VNE, network congestion is caused by incoming VNRs with virtual node demands, virtual link demands and virtual network topologies. Therefore, the challenging and near congestion experiments are designed so that a node congestion can occur when the condition is met as follows.

total CPU demand=total CPU resource;

a link congestion can occur when the condition is met as follows.

total bandwidth demand = total bandwidth resource;

and in both node and link congestion with the VN topology having many links, the SN topology having fewer links. Hence, challenging and near-congestion experiments are designed with VNRs, which are very difficult to be embedded successfully onto the SN.

4.4.2.2 Introduction of Approximate resource scarcity concept

ARS is introduced as a quantitative measurement to show how close is a VN demand compared to a SN resource (Fischer 2016). ARS consists of node scarcity and link scarcity as follows.

For a mapping $(M, (N_i)_{i=1,..,n})$ with each VN: $N_i = (V_i, E_i, w)$: V_i are virtual nodes, E_i are virtual links, w is the demand; and SN: $M = (U, F, w)$, U are substrate nodes, F are substrate links, and w is the resource. ARS of an experiment is defined by a vector (x, y) that are the ratios of total demands over total resources: x is node scarcity, y is link scarcity; (x, y) is presented as the coordination in the scarcity map (Fischer 2016):

$$x = \sum_{i=1}^{n} \frac{w(V_i)}{w(U)} \qquad y = \sum_{i=1}^{n} \frac{w(E_i)}{w(F)}$$

4.4.2.3 Network parameters in challenging scenarios

We transfer the ARS into parameters in the Alevin framework to set up near-congestion experiments for both algorithms CEVNE and D-ViNE-LB (Chowdhury, Rahman & Boutaba 2012). In Table 4.6, challenging and near-congestion experiments are set up on 25-node networks, the VN topologies are set up with $\alpha = 0.75$, β in [0.70 - 0.85]. The algorithms are CEVNE NoM and k-shortest path, and D-ViNE-LB and k-shortest path. Two sub-categories are identified in near-congestion scenarios as follows.:

(i) Most-scarce resource (MSR) category with a simple-topology SN $\alpha = 0.5$ and $\beta = 0.3$; a greedy topology VN $\alpha = 0.75$; β = 0.70, 0.75;

(ii) Less-scarce resource (LSR) with average-complexity SN $\alpha = 0.5$ and $\beta = 0.6$, greedy and complex topology VN $\alpha = 0.75$ and β = 0.80; 0.85. Although VNs are more complex than the MSR, the SN topology is also more complex: we must increase β from 0.3 to 0.6 in the SN, otherwise, it is too hard to complete successfully any VNRs.

Table 4.6: The design of near congestion scenarios

Substrate network: 25 nodes, α = 0.5, β = 0.3 →160-170 links; β =0.6 → 270-280 links, bidirectional.
Virtual networks: 7 nodes, 5 layers, α = .75, β in [0.70-0.85] → [28-34] links/VN, bidirectional.
Substrate node: CPU resource in [10-20]
Virtual node: CPU demand in [5-20]
Substrate link: bandwidth resource in [60-70]
Virtual link: bandwidth demand in [20-35]
Embedding algorithms:
Node embedding: CEVNE NoM algorithm; link embedding: k-shortest path

Node embedding: ViNEYARD algorithm; link embedding: k-shortest path

Experiments in the MSR and LSR categories are tested repeatedly in the pre-defined time window, and only successful ones, which have acceptance ratio > 0, are recorded; their ARSs are shown in the ARS map (Fischer 2016) for both algorithms, CEVNE and D-ViNE-LB.

In the test results, CEVNE successful embeddings in each category are always double to three times D-ViNE-LB successful embeddings. The number of CEVNE successful embeddings in the MSR category is about 60% of the total number of CEVNE successful embeddings in LSR category. This holds true for D-ViNE-LB as well although its number of successful embeddings is far less than CEVNE. The results also show the effects on VNE of the network topology, and the random allocation of resource and demands that can only be measured on a case-by-case basis.

The results of experiments for two categories and their algorithms are shown in Figure 4.7 with each experiment ARS vector (x, y) calculated as in (27). Figure 4.8 displays the results in the whole ARS map for a global view.

4.4.2.4 Evaluation results of challenging scenarios

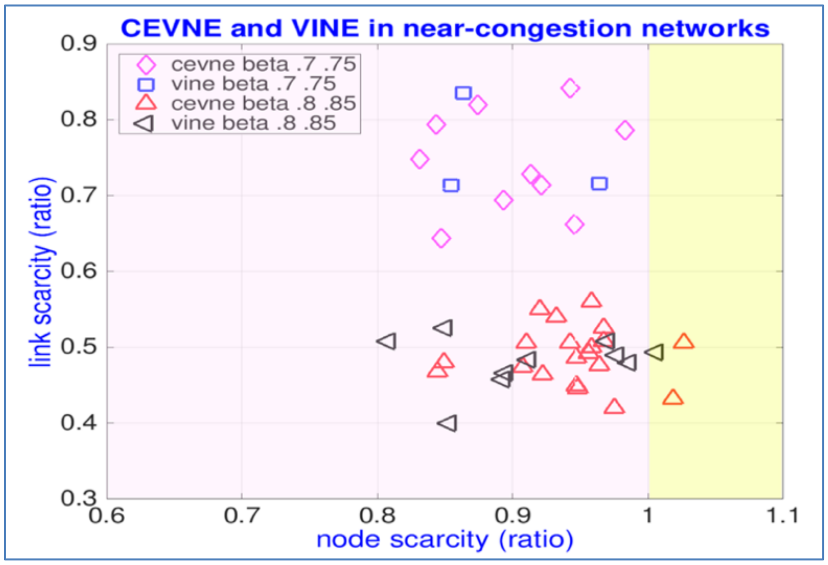


Figure 4.7: Node and link scarcity of near congestion experiments (Fischer 2016)

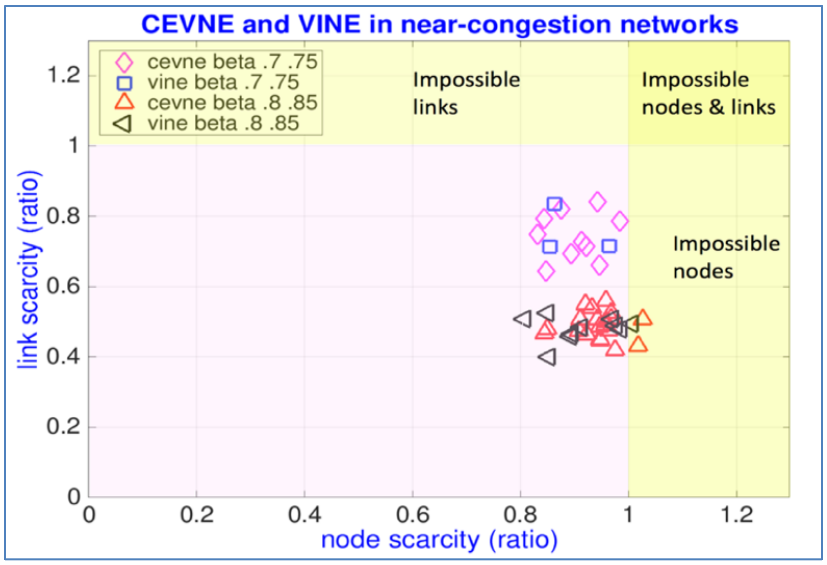


Figure 4.8: Showing experiment results in ARS map (Fischer 2016)

(i)The MSR embeddings (including blue and magenta symbols) have their node scarcity in the range 0.8 to 1, and the link scarcity in the range 0.3 to 0.45. SN is directional so the actual link scarcity is from 0.6 to 0.9. This is very close with the "hard" area in (Fischer 2016).

(ii) The LSR embeddings (including red and black symbols) have their node scarcity in the range 0.8 to 1.05 and link scarcity in the range 0.2 to 0.28; as the SN is directional, it is actually from 0.4 to 0.56, which is matched with the survey that D-ViNE-LB link scarcity is 0.4-0.5 in (Fischer 2016). The result showed CEVNE NoM algorithm outperforms D-ViNE-LB algorithm in nearly-congestion, scarce-resource networks.

4.4.3 The node embedding evaluation results

CEVNE NoM algorithm is evaluated against D-ViNE-LB algorithm by running experiments, recording and analysing results in metrics. As both are node-mapping algorithms, a common link-mapping algorithm is used for the evaluation purposes. The Eppstein algorithm (Eppstein 1998) for k-shortest path is selected as it matches with the testing scenario of 50-node or less SNs. Results of the evaluation are collected in metrics that are related to the CEVNE objectives as follows:

(i) the runtime and cost / revenue metrics are used to justify the cost saving objective,

(ii) the average active-node stress metric is used to justify the congestion avoidance objective,

(iii) the running nodes metric is used to justify the energy saving objective; and acceptance ratio is used for overall performance.

Each metric is collected for both 50-node and 25-node SNs, three types of loads S, M and L. The result of a test scenario is presented in a boxplot. In the boxplot, the central mark is the median, two top edges are the 25th and 75th percentiles, outliers are displayed separately. Each boxplot is named by its topology (the first letter) and load (the second letter). For example, LL specifies the large topology (50 node) and overloaded.

Figure 4.9 presents an example of the two boxplots (one for CEVNE, another for D-ViNE-LB) for runtime of 50-node, overloaded network.

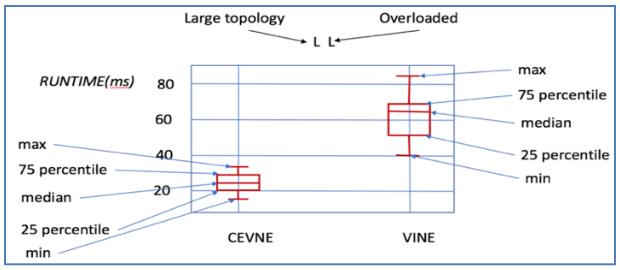


Figure 4.9: an example to explain runtime result of CEVNE and VINE algorithms

4.4.3.1 Runtime

The runtime results in Figure 4.10 and 4.11 show that CEVNE NoM converged faster than D-ViNE-LB algorithm: CEVNE's total runtime is improved by 50% compared to D-ViNE-LB's. This is explained using the Simplex method as follows: CEVNE introduces more explicit constraints, so the search space is more confined, and hence it found solutions quicker. Figure 4.10 shows the runtime of 50-node network in all topologies and all workloads.

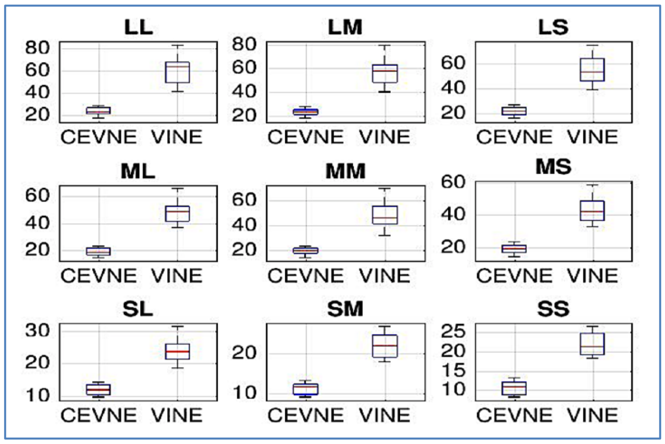


Figure 4.10: Total runtime of 50-node network

Figure 4.11 shows the results of the runtime metric of 25-node network in all topologies and all workloads.

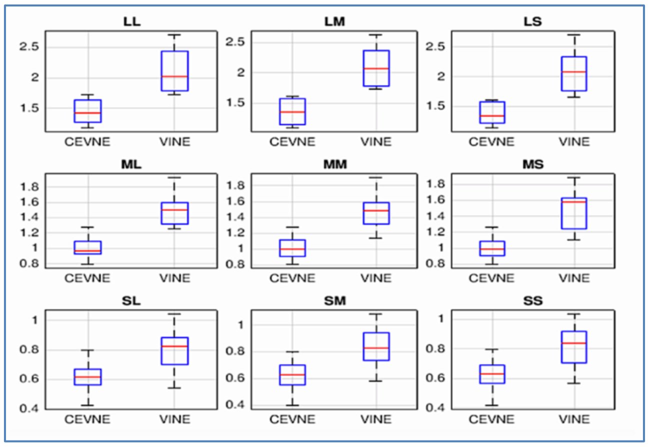


Figure 4.11: Total runtime of 25-node network

4.4.3.2 Average active node stress

The average active node stress is calculated by dividing the total number of embedded virtual nodes by the number of active substrate nodes. CEVNE objective function puts a limit on the bandwidth usage to avoid congestion: virtual flows will only use R value multiplied by the max bandwidth; this implies that virtual node mappings are arranged tightly for virtual flows to meet the constraints on the limit bandwidth resources. Therefore, CEVNE average active node stress will be higher as in Figure 4.12 that shows the results of 50-node network in all topologies and all workloads. Figure 4.13 shows the average active node stress of 25-node network in all topologies and all workloads.

4.4.4 Energy consumption

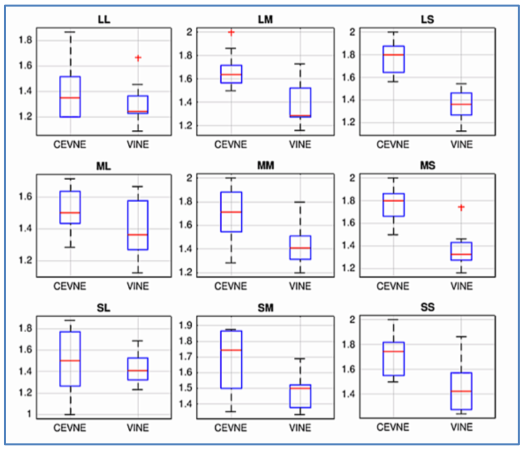


Figure 4.12: Average active node stress of 50-node network

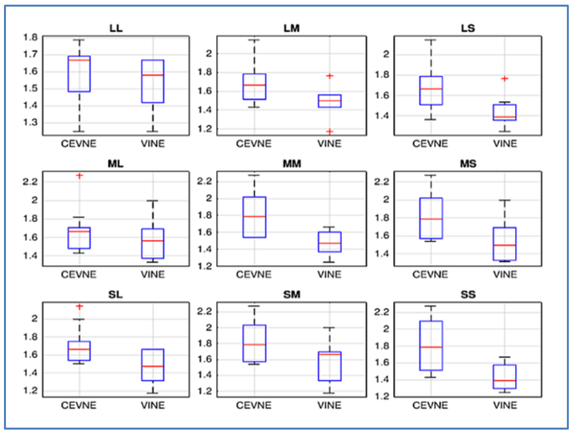


Figure 4.13: Average active node stress of 25-node network

4.4.4.1 Energy consumption

The running nodes metric is used to evaluate the energy consumption of CEVNE against D-ViNE-LB: the algorithm with less running nodes will consume less energy. The test results in Figure 4.14 show that the number of running nodes in CEVNE is always less than in D-ViNE-LB. In 50-node network, CEVNE always has 10 running nodes less than D-ViNE-LB; this means CEVNE consumes 20% less energy. Figure 4.14 shows the results in 50-node networks with all topologies and all workloads.

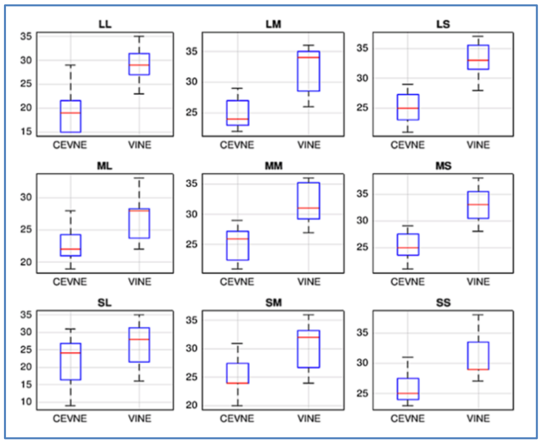


Figure 4.14: Running nodes in 50-node network

Figure 4.15 shows the results of the running nodes in 25-node networks with all topologies and all workloads.

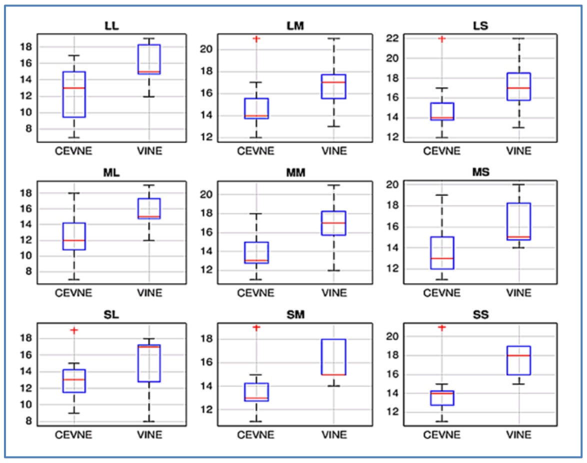


Figure 4.15: Running nodes in 25-node network

4.4.4.2 Acceptance ratio

The acceptance ratio is the ratio of virtual networks that have been embedded successfully on the substrate network. The acceptance ratios of both algorithms are very similar in both 50-node and 25-node networks.

4.4.4.3 Total costs

The total cost is calculated based on the substrate resources that are used for the VNE; it does not include the energy cost. The total costs of both algorithms are very similar in both 50-node and 25-node networks. Through the analysis of the above metrics: runtime, average active node stress, running nodes, total cost and acceptance ratio, CEVNE NoM prevails over D-ViNE-LB in all three objectives: congestion-aware, energy-aware and cost saving.

4.5 Summary

In this chapter, the research objective is to investigate and to propose a novel mathematical model for a multi-objective VNE. A congestion-aware energy-aware VNE (CEVNE) is proposed with the purpose to minimize network resource costs, avoid network congestion, and minimize energy usage. Its substrate network is an SDN network running SR. The congestion avoidance is implemented by setting the upper limit R for the congestion ratio, in which R is calculated using the hose model of the routing and traffic demand models (Oki 2012). As a coordinated node and link embedding VNE, CEVNE, is formulated as an MIP program, then it is relaxed to an LP, and solved by the GLPK solver. The CEVNE node-embedding algorithm applies the rounding function to find the approximation of the integer variables for the sub-optimal solutions. The evaluation on near-congestion, scarce-resource networks shows that the CEVNE congestion control objective is useful: the number of its successful cases is double the number of D-ViNE-LB. The overall evaluation of the node embedding process shows that CEVNE delivers concurrently the saving-cost, the congestion-aware and the energy-aware objectives. The result of this chapter will be extended to the CEVNE link embedding process in the next chapter.

CHAPTER 5 REALIZATION OF CEVNE VIRTUAL LINK EMBEDDING

The objective of previous chapter was to investigate and propose a novel mathematical model for the virtual network embedding that focuses concurrently on multiple objectives: saving cost, saving energy and avoiding congestion, called CEVNE. As solving the VNE is NP-Hard (Rost & Schmid 2018), the embedding algorithms are designed as a heuristic approach for the node embedding and link embedding processes to find the sub-optimal solutions in polynomial time.

This chapter objective is to investigate and propose the novel mechanism to realize CEVNE link embedding process, which will be an active process that focuses on multiple objectives: cost saving, energy saving and congestion avoidance, as the node embedding process does. The realization of the novel link embedding algorithm is an extended SDN application that integrates each objective as an application / service and uses a monitoring function in real-time. The realization application is deployed on the SDN substrate network running SR. The roadmap of the chapter is presented in Figure 5.1

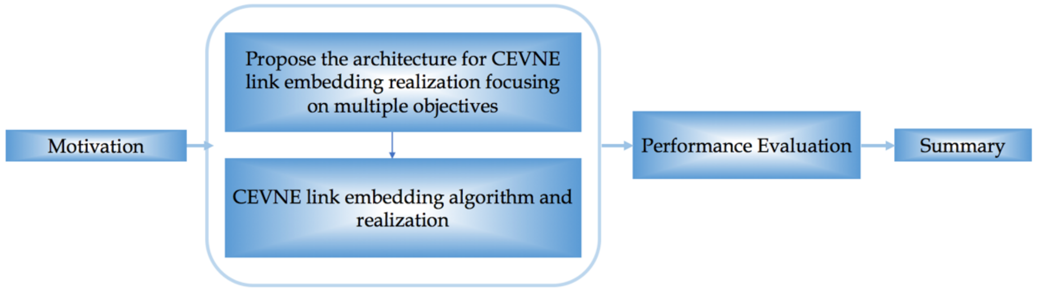

Motivation
Propose the architecture for CEVNE link embedding realization focusing on multiple objectives
CEVNE link embedding algorithm and realization
Performance Evaluation
Summary

Figure 5.1: The roadmap of Chapter 5

In the chapter, section 5.1 presents the motivation for a novel heuristic solution with a short revision of CEVNE problem formulation in chapter 4 named Congestion-aware energy-aware virtual network embedding, section 5.2 presents the novel architecture to realize the CEVNE LiM process, section 5.3 presents the novel CEVNE LiM algorithm and its realization on the SDN controller, section 5.4 presents the performance evaluation, and section 5.5 summarizes the chapter.

The proof of concept of the research with title "Realization of congestion-aware energy-aware virtual network embedding" has been presented at the International Telecommunication Networks and Applications Conference (ITNAC) 2019 in Auckland NZ on November 27th, 2019

What is new in this solution?

The novelties of the solution are summarised as follows.

- ✓ the SDN-based heuristic algorithm.
- ✓ the active virtual link embedding heuristic algorithm that can focus on multiple objectives: cost saving, energy saving and congestion avoidance as the node embedding algorithm does.
- ✓ The realization of the virtual link embedding using the three-tier architecture, micro-service architecture and micro-service design patterns: service creation, service registration and service discovery
- ✓ The realization on an SDN substrate networks, which is a leaf-spine fabric running segment routing with an in-memory routing protocol called equal cost multi path (ECMP) shortest path graph (SPG).

5.1 Motivation for a heuristic algorithm

5.1.1 The review of CEVNE problem and solution approach

The CEVNE is a VNE problem that focuses concurrently on multiple objectives: cost saving, energy saving and congestion avoidance. The CEVNE applies the rules of multi-objective optimization to generate the pareto-optimal solutions using the positive weights and the binding constraints to combine its objectives. Solving the VNE problem is NP-hard as it relates to the multi-way separator problem (Amaldi et al. 2016), a heuristic solution is proposed involving a two-stage coordinated CEVNE, a node embedding process and a link embedding process. The CEVNE is modelled as a MILP. Solving MILP is computationally intractable; hence, the MILP is relaxed to a LP to be solved in polynomial time using the GLPK in the Alevin framework. The results are sub-optimal solutions as the program has been relaxed to linear program. The node embedding algorithm applies the D-VINE rounding function (Chowdhury, Rahman & Boutaba 2012) to find the approximation of integer variables. The link embedding algorithm has been provided in section 4.3.2. In this chapter, the CEVNE LiM will be presented in more details.

5.1.2 Motivation for a heuristic virtual link embedding

The purpose of the virtual link embedding process in CEVNE is to embed virtual links in the VNR onto the substrate links of the SN. Solving the virtual link embedding is NP-Hard as it is reduced to the un-split-table flow problem (Fischer et al. 2013); therefore, a heuristic algorithm is required to find the close-to-optimal solutions in reasonable time. The shortest path algorithm is utilized in the virtual link embedding process (Fischer et al. 2013). This leads to network congestion on the shortest paths and under-utilization elsewhere. Even the greedy approach could not provide the solution that can integrate with network monitoring in real-time for the optimal operation requirement. Nevertheless, applying SDN paradigm may allow the integration with any monitoring applications, and the focus on multiple objectives based on SDN attributes: programmability, central control, and central view.

In this chapter, the virtual link embedding heuristic algorithm is further enhanced from the virtual link embedding algorithm in chapter 4, by applying SDN attributes and SR source routing on a leaf-spine fabric substrate network. The technologies will empower it as an active process that focuses on multiple objectives; each individual objective is designed as a service that can apply the control and calculation functions of the SDN programming attribute. Additionally, SDN-based approach will allow the link embedding algorithm to select the appropriate substrate paths based on SDN ability to steer the network flows at fine-grained and coarse-grained levels. Therefore, the link embedding algorithm will be realized as an SDN application on top of SDN controller.

5.2 Proposed architecture to realize CEVNE LiM

The proposed architecture to realize CEVNE LiM is based on the architecture for extended SDN application (Pham & Hoang 2016) with the rational as follows:

- ✓ As the substrate network is an SDN network, the CEVNE LiM process will be realized as an application built on top of SDN controller such that CEVNE LiM can access the controller core services to select the substrate paths for the LiM process.
- ✓ As presenting in Chapter 3, the architecture for extended SDN application facilitates new service creation and service composition. Each objective of CEVNE LiM is designed as one or multiple MS as presented in chapter 2. The CEVNE LiM service is the composition of new services, existing SDN core services, and the applications: the path calculation element (PCE) and monitoring applications, as presented in the CEVNE link embedding algorithm in Figure 4.5.

- ✓ The realization of the CEVNE LiM will be based on a three-tier architecture with UI tier, business tier, and database tier. The UI tier allows one to input CEVNE LiM parameters, which are virtual link demands, and the results of CEVNE NoM process; and view CEVNE LiM results on the web browser. The business tier encapsulates the creation and composition of services as presented in chapter 3. The databased tier allows one to store the link residual capacities.

The architecture to realize CEVNE LiM process is depicted in Figure 5.2. The three-tier architecture of CEVNE LiM stays in the application layer of the SDN controller, in which the UI tier consists of REST API and CLI, the business tier consists of new services for cost saving, energy saving and congestion avoidance, existing SDN core services to be composed are intent API, topology API, path management API, existing applications to be composed are SR, PCE, monitoring application; the database tier consists of a database used to store link residual capacities. CEVNE LiM application consists of the CEVNE LiM service, which is the integration of all services in the business tier, multiple interfaces: CEVNE LiM Rest API and CEVNE LiM CLI, and the database to persist data.

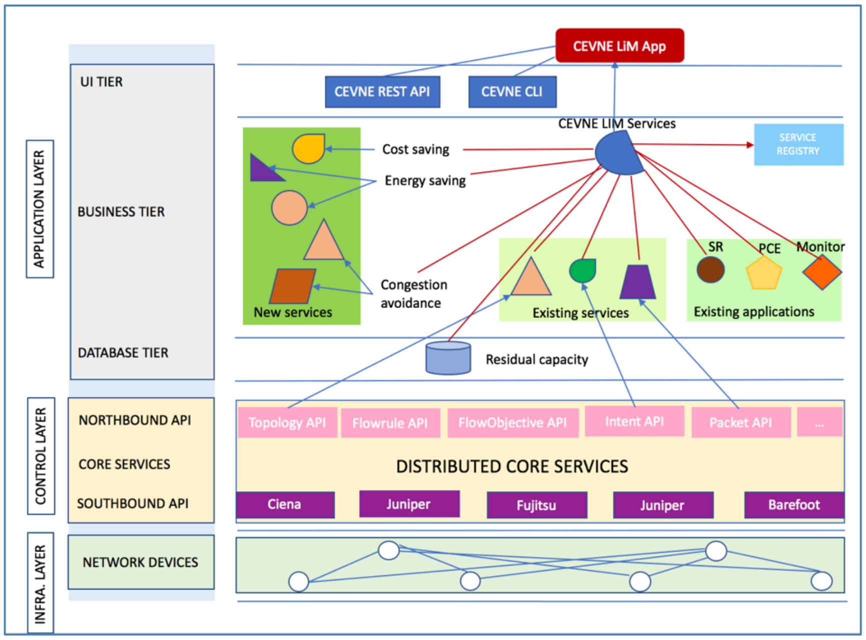


Figure 5.2: The architecture to realize CEVNE LiM.

5.2.1 Services to handle multi-objective link embedding process

In this section, micro-services are proposed for each CEVNE LiM objective: cost saving, energy saving and congestion avoidance. The monitoring application provides real-time status to exclude substrate paths, which are alerted as congested. The path selection algorithm selects the substrate path satisfying the bandwidth demand of the virtual link, and the three objectives to embed the virtual link.

5.2.1.1 Cost saving

The path satisfying the cost saving objective is the shortest path between two endpoints satisfying the bandwidth demand. The shortest paths can be retrieved using the path management API in SDN core service, or the function equal cost multi path (ECMP) shortest path graph (SPG) in SR application, the paths satisfy the bandwidth demand are selected. Figure 5.3 presents an example of the path satisfying the cost saving objective, in which the shortest paths between two endpoints A and B are A-S1-B or A-S2-B that satisfy the virtual link bandwidth demand. The other paths between A and B are A-S1-C-S2-B, A-S2-D-S1-A, which satisfy the virtual link bandwidth demand and are not shortest paths.

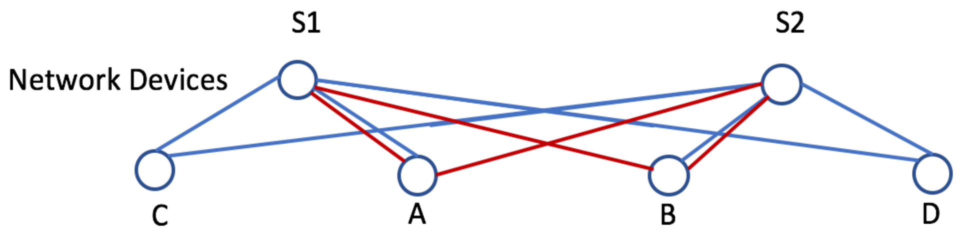


Figure 5.3: the example of the cost saving between A and B

5.2.1.2 Energy saving

In the simplest way, the energy saving objective is implemented based on the resilience design of the leaf-spine fabric to steer traffic flows to the fall-back spine in case of network failure. There are links from reserved spines to leaf nodes. To save energy, the reserved spines and their links are put into sleep mode, they will be turned to active mode in case of emergency. The results of energy saving objectives are substrate paths that satisfy the bandwidth demand and not in sleep mode.

Figure 5.4 presents an example of a leaf-spine fabric with a resilience design, spine S3 is the reserved one, S3 and all its links S3-A, S3-B, S3-C, S3-D, S3-E are put in sleep mode to save energy.

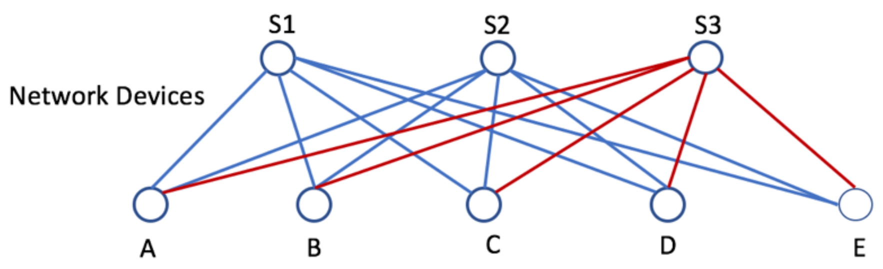


Figure 5.4: an example of energy saving of leaf-spine fabric

In normal scenarios, the energy-saving algorithm looks for the least utilization spine switches and turns them into sleep mode (Li, Lung & Majumdar 2015) that is presented in Figure 5.5 as follows.

Figure 5.5: The algorithm to inactivate a spine node

```
1    Begin
2        If number of inactive spine switches >= limit of inactive spine
3            keep current number of inactive spine switches and return, otherwise
4        For each spine switch, check all its link usage
5            If all link usage < the low thresh-hold,
6            Turn the spine into inactive mode, and migrate its traffic to other active spines
7    End
```

In the algorithm in Figure 5.5, the leaf-spine fabric has a limit on the number of inactive spine switches to ensure the overall performance of the datacenter. If the current number of inactive spine switches is at the limit, the algorithm just returns (line 2 – 3). Otherwise, each spine switch is checked for all its link usage against the low threshold (line 5). If all the link usage is lower than the threshold, the spine is set to inactive mode, and all its traffic is migrated to other active neighbor spine switches (line 6). In reverse, when the fabric has high traffic demand, the inactive spines will be activated to prevent congestion as presented in the algorithm in Figure 5.6.

Figure 5.6: Algorithm to activate an inactive spine node

```
1   Begin
2   If number of inactive spine switch <=0
3     then return, otherwise
4   For each spine switch, check all its link usage
5   If all spines have link usage > the high thresh-hold
6     Change an inactive spine to the active mode, and migrate its spine neighbours
    traffic to it
7   End
```

In the algorithm in Figure 5.6, when the number of inactive spine switches is 0, the algorithm returns (line 2 - 3). The administrator can record how often this situation occurs as it indicates high utilization, thus more spine switches should be added to the fabric. Otherwise, for each spine switch, all its link usages are checked against a high threshold (line 5). If all spine switches are utilized higher than the high threshold, then an inactive spine switch is turned to active mode. The traffic of its active neighbor spines is migrated to it (line 6). Both algorithms in Figure 5.5 and Figure 5.6 are working continuously to ensure the fabric performance, saving energy, and preventing congestion.

5.2.1.3 Congestion avoidance

The congestion-aware algorithm spreads selected substrate paths that satisfy the bandwidth demands of the virtual links on all active spine switches. It uses an internal structure to keep track of which active spines it has allocated for virtual links to avoid using these spines again. If it has used all active spines, then it resets the list of active spines to start the next round of embedding. The congestion-aware algorithm might include the dynamic resource allocation algorithm to balance the link usage and prevent fragmentation (Mijumbi et al. 2014). Initially, all substrate resources are equal, and the fabric is symmetric. After provisioning some VNRs, the resources are allocated to virtual nodes and links, their residual resources are different for each device. The hypervisors and substrate links residual resources can be stored in the database. The congestion-aware algorithm retrieves the fabric residual resources to select switches that satisfy the bandwidth demand and have the largest resource in the active spine list. Figure 5.7 presents an example of the congestion avoidance objective.

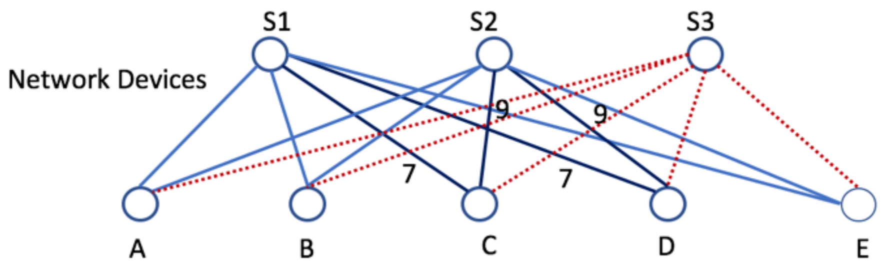


Figure 5.7: An example of congestion avoidance in leaf-spine fabric

In Figure 5.7, the leaf-spine fabric has 3 spine nodes and 5 leaf nodes, the spine S3 and its links (from S3 to the leaf nodes) are in sleep mode. The paths between C and D are C-S2-D and C-S1-D, both satisfy the bandwidth demand, to avoid congestion, path C-S2-D is selected because of the highest residual capacity.

5.2.2 Monitoring application

Integration with monitoring application is one of the strengths of CEVNE heuristic algorithms. Several options are available as presented as follows.

- ✓ Monitoring applications are built with P4 language. The In-band network telemetry (Kim et al. 2016) captures states in the data plane and collects them at the sink nodes; INT monitors real-time network operation. The Multi-route inspection application uses a field called Explicit congestion notification (ECN) in the IP options header to record the switch congestion state.
- ✓ OpenNetMon network monitoring method (Tsai et al. 2018) provides an end-to-end measurement of throughput, packet loss, latency and link utilization in the SDN network,
- ✓ The CPMan monitoring application (Li, Koshibe & Prete 2019) exploits the Openflow built-in statistics to monitor the link usage.

The monitor application is used to watch for highly utilized substrate paths (nearly congestion), which will be excluded from future virtual link embedding. The results of congestion avoidance and the monitoring application will complement each other.

Figure 5.8 captured the UI of the INT application with the traffic delays of the leaf and spine switches at time T running on a leaf-spine fabric with two leaf nodes and two spine nodes. The higher the bar, the longer is the delay of the data packet at the switch. The longest traffic delay occurred at leaf 1 (the violet stacks). The second longest is recorded at leaf 2 (the orange stack). The traffic at two spine switches is not significant. If we need to select the leaf switch with less traffic, then leaf 2 is selected. The INT application is presented in section 2.9.2 in chapter 2.

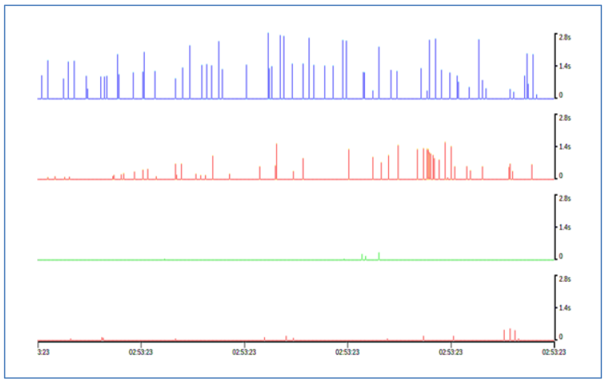


Figure 5.8: Result of INT monitor application at time T

Multi-route inspection, which is a P4 monitoring application, utilizes the field called Explicit congestion notification (ECN) in the IP options header to record the switch congestion state. The application used the queue depth of the switch to estimate the congestion by comparing it with a threshold. If the queue depth is larger than the threshold, then ECN is set to value 11 to mark congestion. Every switch will record its switch id and its queue depth into a stack in the IP options. The host at the destination will extract the data and finds out the switch with the large queue depth, which is potential in congestion (Tahmasbi 2017).

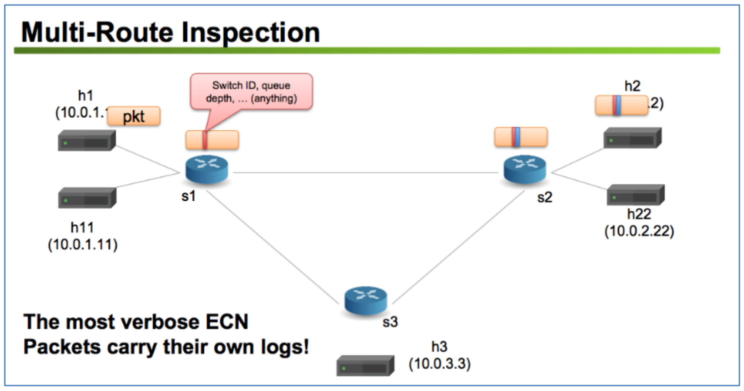


Figure 5.9: Multi-route inspection application: each packet carries its own logs (Tahmasbi 2017).

5.2.3 The initialization processes

CEVNE LiM configures the SDN substrate network to apply SR technology. It is assumed that ONOS controller has been started and SR application has been activated in advance before the CEVNE LiM initialisation. The initialization process is described in Figure 5.10.

Figure 5.10: Initialization process of CEVNE LiM

```
1   Begin
2       Configuring the SDN substrate network to use Segment Routing
3       SR application builds ECMP shortest path graphs (SPG) for the SN based on
    Breadth First Search (BFS)
4       Running the monitor application to find out the long delay paths in the SN
5       Running the inactive algorithms for spine nodes for energy saving objective
6       Activate the DRM application to record residual resources.
7   End
```

The initialization process is described (line 1 – 7) in the algorithm. First, the substrate network is configured using the SR application to implement the SR source routing. The SN configuration is translated into flow entries for specific flow tables and group tables, based on the SDN and SR technologies (Broadcom 2019). Each switch is a white box switch that requires 'ofdpa-ovs' driver, its IP address, its ports, its segment id, and whether it is a leaf or a spine. Each host has its mac address and its network location (Das 2019).

CEVNE LiM uses the SR application routing service for the link embedding process. Internally, SR application uses ONOS link service to generate ECMP shortest path graph (SPG) for each switch or router in the SN (ONOS 2019c) (line 3). Each ECMP SPG has a root node, and the paths connecting it to other switches in the fabric are set up based on the minimum number of hops. From the root node, SR application searches in its egress links for the shortest path for each of the destination nodes and implements its own routing services based on the ECMP SPGs. The routing service will recalculate the ECMP SPGs where there are link failures or node failures based on the device event handler, and the link event handler in the ONOS core services (ONOS 2019c). The SR application uses the ECMP SPGs and the routing services in the role of the path calculation element. CEVNE LiM uses the INT or the OF statistic-based monitor application to report the real-time delays at switches to exclude the heavy traffic path in the delay constraint (line 4). The algorithms to activate / inactivate spine nodes are started to setup inactive spines and turn them into sleep mode (line 5). The DRM function is setup to record the residual resources (line 6).

5.2.4 Path selection algorithm

The path selection algorithm (PSA) specifies how to select a substrate path to embed to a virtual link, which satisfies the bandwidth demand and the three objectives: cost saving, energy saving and congestion avoidance, hence, it is a critical component in CEVNE LiM. PSA consists of two steps: the preparation step and the normal running status. The preparation step consists of the initial inactivating of spine nodes for the energy saving objective, and the setup of DRM function for load balancing. For each objective, there may be more than one algorithm, the preparation step will specify the algorithm to be used. The preparation step is included in the initialization process. After the preparation step, the PSA is in the normal running status, and it will select the substrate paths using the algorithms of three objectives.

The details of the PSA are presented as follows. It receives the two endpoints and the virtual link bandwidth demand as input; it provokes the services to implement cost saving, energy saving and congestion avoidance algorithms to filter the substrate paths that satisfy three objectives. The PSA selects the substrate path using the steps as follows. First, the substrate paths that are the results of the energy saving objective will satisfy the bandwidth demand and the cost saving objective based on the energy saving algorithm. These substrate paths will be checked for their residual bandwidth resources in the congestion avoidance objective, and the one with highest residual bandwidth is selected. The results of the congestion control objective and the monitoring application should match as the link having lowest residual link will be alerted as a potential congestion in the monitoring application. If the PSA gives no result, the congestion control service will be invoked on different paths for alternative results, which will be selected, as the congestion avoidance objective is ranked high priority. If the PSA gives multiple results, one is selected randomly. The PSA ensures that the selected substrate path satisfies the bandwidth demand and three objectives, and it needs to respond quickly in a high-speed fabric.

To illustrate how the PSA works, Figure 5.11 presents an example of the leaf-spine fabric of four spine nodes and seven leaf nodes. The two endpoints are C and E, the cost saving objective results in substrate paths C-S1-E, C-S2-E, C-S3-E, and C-S4-E; the energy saving objective results in substrate paths C-S1-E, C-S2-E, C-S3-E that satisfy the bandwidth demand; the congestion avoidance objective results in the descending order based on the residual link capacity C-S2-E, C-S3-E, and C-S1-E; as the substrate path C-S1-E is alerted as a congested path by the monitoring application, the substrate path for the virtual link embedding between node C and E is C-S2-E.

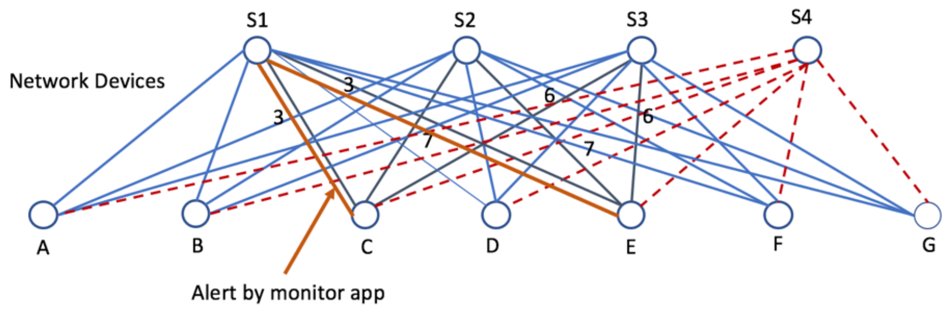


Figure 5.11: Example of PSA in a leaf-spine fabric

5.3 CEVNE LiM algorithm and realization

5.3.1 CEVNE LiM algorithm

The inputs for CEVNE LiM algorithm are the VNR, the substrate node endpoints that virtual nodes are embedded on, and the bandwidth demands of virtual links. LiM algorithm will call the PSA to select the substrate paths for each virtual link, which satisfy the three objectives, and not being alerted as in congestion state according to the monitoring application. LiM repeats the process until all virtual links are embedded, or an error occurs, and the whole VNR is terminated. It is depicted in Figure 5.12 as follows.

Figure 5.12: Virtual link embedding process of CEVNE LiM

```
1   Input
2       CEVNE Node embedding results
3       Virtual links and their bandwidth demand in the VNR
4   Output
5       Embedded substrate paths of virtual links
6   Begin
7       For each virtual link $l = (i, j) \in E_V$
```

```
8              Invoke the PSA(i,j) to search the substrate paths satisfying three objectives
9              If the PSA(i,j) returns the substrate path
10                 Creating a flow rule for the substrate path endpoints
11                 Creating a VLAN tunnel for the substrate path
12                 Return the substrate path
13             Else
14                 Remove all interim flow rules and VLAN tunnels
15                 Exit with failure
16         End For
17     End
```

CEVNE embeds each virtual link in the VNR using the virtual link embedding process (line 1 to line 17) in the algorithm in Figure 5.12. The PSA has inputs that are the embedded substrate nodes as the results of the CEVNE NoM. The virtual links with their bandwidth demands are also the PSA input. The embedding steps (line 7 – 16) are repeated for each virtual link. CEVNE LiM application accepts the two endpoints i and j resulted from the CEVNE NoM. The PSA is invoked to select the substrate path between node i and j (line 8) to embed the virtual link using its algorithm. If the substrate path that satisfies the bandwidth demand and three objectives, are returned, a flow rule is created for the selected path (line 10). A VLAN tunnel is created for the selected substrate path (line 11). Otherwise, all interim flow rules and VLAN tunnels are removed (line 14) and the algorithm exit with failure (line 15). The steps in the virtual link embedding process are repeated in the order as specified in the algorithm. The loop is ended when all the virtual links are embedded successfully (line 16).

Figure 5.13 summarized the flow diagram of the CEVNE LiM algorithm with the main steps: the initialization process, and the CEVNE virtual link embedding process. The initialization process consists of the configuration of the SR and SN, the activation of the SR application, monitoring application, the inactivate algorithm of spine nodes for energy saving objective, and the DRM application for the link load balancing purpose. The CEVNE virtual link embedding is repeated for each virtual link to search for the substrate path that satisfies the bandwidth demand and the three objectives. If no substrate path is found for the virtual link, or if the intent that is created for the substrate path is not installed, the algorithm exits with failure. Otherwise, it exits with success if all virtual links are embedded successfully.

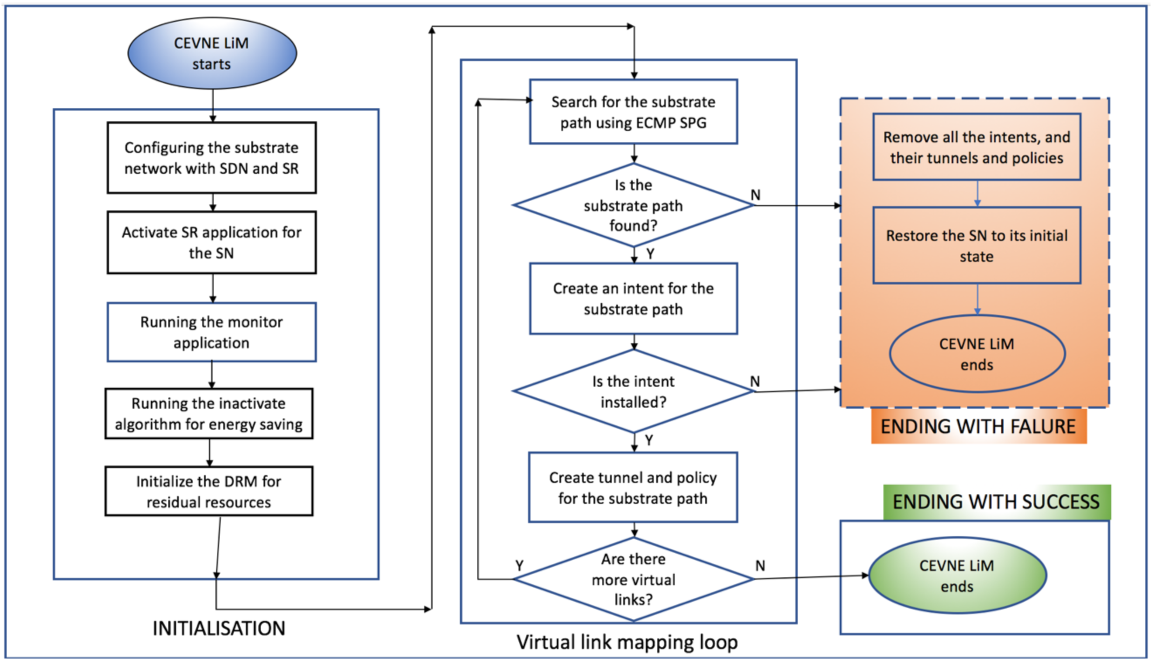

CEVNE LiM starts
Configuring the substrate network with SDN and SR
Activate SR application for the SN
Running the monitor application
Running the inactivate algorithm for energy saving
Initialize the DRM for residual resources
INITIALISATION
Search for the substrate path using ECMP SPG
Is the substrate path found?
Y
N
Create an intent for the substrate path
Is the intent installed?
Y
N
Create tunnel and policy for the substrate path
Are there more virtual links?
Y
N
Virtual link mapping loop
Remove all the intents, and their tunnels and policies
Restore the SN to its initial state
CEVNE LiM ends
ENDING WITH FALURE
ENDING WITH SUCCESS
CEVNE LiM ends

Figure 5.13: CEVNE LiM algorithm

5.3.2 Operational diagram of CEVNE LiM

The purpose of the operational diagram is to examine closely the parameters exchanged between the CEVNE LiM and the CEVNE NoM internally. The diagram presents both processes when a VNR arrives. The VNR has its virtual nodes, virtual links and the topology. The CEVNE NoM embeds the virtual nodes onto the substrate hosts and sends the results to the CEVNE LiM. The CEVNE LiM embeds the virtual links onto the substrate paths using the CEVNE LiM algorithm. When all the virtual links of the VNR are embedded, the VNR is completely provisioned.

In Figure 5.14, the substrate network is an SDN network that is configured into a leaf-spine fabric using the Segment Routing (SR) application. A VNR is input into the CEVNE NoM process to embed its virtual nodes onto the substrate nodes. Virtual nodes A, B, C, D and E are mapped to host 1, 2, 3, 4 and 5 accordingly as these hosts meet their CPU demands. The results of the CEVNE NoM, the VNR topology, are input to the CEVNE LiM process to map the virtual links onto the substrate paths. The INT or Openflow statistic-based monitoring application is used to provide the real-time delay when the packets travel through the fabric. The Path computation element (PCE) could be the ONOS path service that computes different types of paths between two endpoints. Virtual links AB, BC, AE and ED are embedded onto the substrate paths of the fabric.

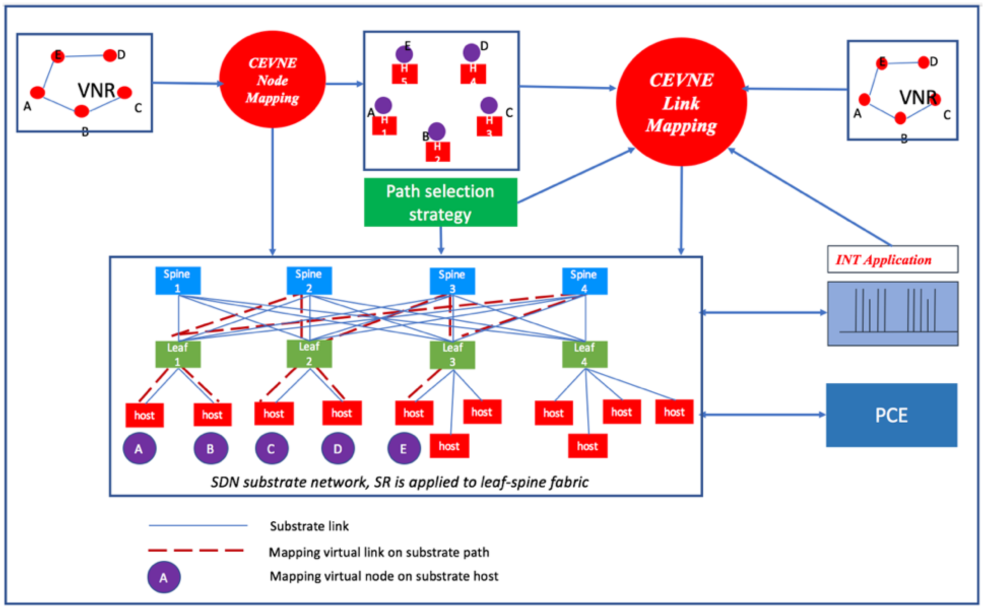


Figure 5.14: CEVNE node and link mapping processes on SDN fabric network

5.3.3 Realization of CEVNE LiM application

CEVNE LiM algorithm is developed into an SDN composite application (Pham & Hoang 2016) called CEVNE LiM application that is deployed on top of the ONOS controller in a leaf-spine fabric SDN substrate network running the SR application. The leaf-spine fabric is configured in ONOS controller and is presented in the ONOS UI as in Figure 5.15.

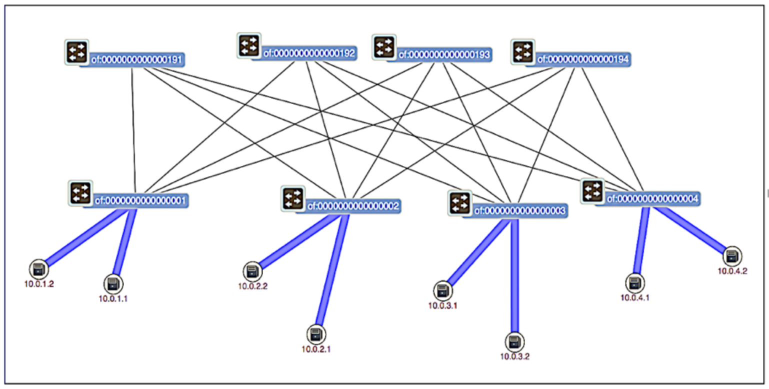


Figure 5.15: Substrate network configured as leaf-spine fabric in ONOS UI

5.4 Performance evaluation

The CEVNE LiM performance evaluation consists of the steps: the test scenario design, performing CEVNE test scenarios, and analyzing results. The next subsection describes the test scenario setup.

5.4.1 Test scenarios

The test scenario includes a substrate network and multiple virtual networks that will be virtualized and provisioned over the substrate network. The substrate network is a leaf-spine fabric with four leaf switches, four spine switches, eight hosts, and links connecting switches and hosts as in Figure 5.15. The leaf and spine switches, which are white boxes running open-source software, are configured to run SR protocol on ONOS controller as specified in Atrium (ONF 2019b). The substrate network and their capacities are specified in Table 5.1.

Table 5.1: Substrate network capacities

	Total number	CPU capacity (GHz)	BW capacity (Mbps)
Leaf switches	4	1	
Spine switches	4	1	
Hosts	8	100	
Link leaf-spine	16		10
Link leaf-host	8		1

In Table 5.1, hosts have large CPU capacities, they are ready to have virtual nodes embedded on them. Links between leaf and spine switches have high bandwidth capacity, they are ready for virtual links to be embedded.

Different test cases are designed with various VNRs requirements to test the performance of CEVNE LiM in terms of runtime, average path length, average node stress, average link stress, energy consumption, acceptance ratio and total costs / revenue. VNRs in test cases are different in topologies and resource demands as specified in Table 5.2.

Table 5.2: The design of test scenarios

Test cases	No of virtual nodes	CPU demands (GHz)	No of virtual links	BW demands (Mbps)
1	3	10	2	3
2	5	10	4	3
3	4	10	3	3
4	5	10	4	3
5	4	10	3	3

In Table 5.2, there are 5 different VNRs, with the number of requested virtual nodes ranges from 3 to 5, and the number of requested virtual links ranges from 2 to 4.

5.4.2 CEVNE LiM application execution platform

The CEVNE LiM application is compiled, deployed and activated on the ONOS controller as CEVNE virtual link service. CEVNE LiM application implements PSA energy-saving algorithm, cost-saving algorithm, and congestion-aware algorithm. The CEVNE LiM is invoked via the CLI or the REST API. The input for each virtual link consists of a tuple of the source and destination host locations, and the bandwidth demand. The result of the CEVNE LiM application runs on the ONOS controller is shown in the Figure 5.16.

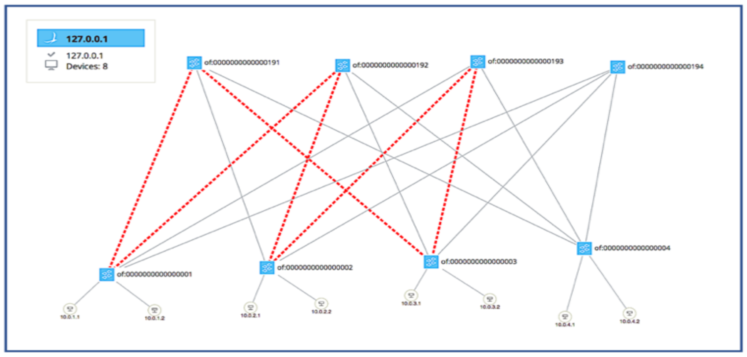


Figure 5.16: CEVNE LiM application execution in ONOS UI

In Figure 5.16, the red-dashed lines are the selected substrate paths, where the virtual links are embedded on. On the substrate paths, the flow rules are installed on leaf and spine switches to establish the route. They are highlighted as the result of the flow rule creation for each substrate path. The CEVNE LiM execution in the ONOS CLI for the VNR in Figure 5.16 is summarized in TABLE 5.3.

Table 5.3: Results of CEVNE LiM when running a test case

	Virtual links	Node mapping results	CEVNE LiM results
1	AB	host 1, host 2	The hosts are connected to leaf 01.
2	BC	host 2, host 3	onos:cevne-get-mapped-paths of:0000000000000001 of:0000000000000002 5000000 DefaultPath{src=of:0000000000000192/2, dst=of:0000000000000192/1, type=INDIRECT, state=ACTIVE, expected=false, links=[DefaultLink{src=of:0000000000000192/2, dst=of:0000000000000002/3, type=DIRECT, state=ACTIVE, expected=false}, DefaultLink{src=of:0000000000000001/3, dst=of:0000000000000192/1, type=DIRECT, state=ACTIVE, expected=false}], cost=ScalarWeight{value=0.0}}
3	AE	host 1, host 5	onos:cevne-get-mapped-paths of:0000000000000001 of:0000000000000003 5000000 DefaultPath{src=of:0000000000000191/3, dst=of:0000000000000191/1, type=INDIRECT, state=ACTIVE, expected=false, links=[DefaultLink{src=of:0000000000000191/3, dst=of:0000000000000003/4, type=DIRECT, state=ACTIVE, expected=false}, DefaultLink{src=of:0000000000000001/4, dst=of:0000000000000191/1, type=DIRECT, state=ACTIVE, expected=false}], cost=ScalarWeight{value=0.0}}
4	ED	host 5, host 4	onos:cevne-get-mapped-paths of:0000000000000003 of:0000000000000002 5000000 DefaultPath{src=of:0000000000000193/2, dst=of:0000000000000193/3, type=INDIRECT, state=ACTIVE, expected=false, links=[DefaultLink{src=of:0000000000000193/2, dst=of:0000000000000002/2, type=DIRECT, state=ACTIVE, expected=false}, DefaultLink{src=of:0000000000000003/2, dst=of:0000000000000193/3, type=DIRECT, state=ACTIVE, expected=false}], cost=ScalarWeight{value=0.0}}

In Table 5.7, the CLI command cevne-get-mapped-paths invokes the virtual link mapping for virtual links AB, BC, AE, and ED.

Row 1: The virtual link AB is the link between host 1 and host 2, which is within the leaf 01, they are connected in the VXLAN connecting the ToR switch (leaf 1) and the hosts.

Row 2: The virtual link BC is between host 2 and host 3, which is the path between leaf 01 (host 2 location) and leaf 02 (host 3 location). The path between leaf 01, and spine 02 (port 1), spine 03 (port 2) and leaf 02 is selected. Data in row 3 and row 4 are explained similarly as in row 2.

5.4.3 Evaluation results

The CEVNE LiM performance is evaluated by analyzing its implementation results and comparing them with those of the k-shortest path link embedding algorithm (Beck et al. 2014). The metrics used to compare the behavior of the two algorithms are the total runtime, acceptance ratio, average path length, average node stress, average link stress, and energy consumption. These metrics are selected because they reflect the three CEVNE objectives: cost saving, energy saving and congestion avoidance. The following subsections will examine each of the results.

5.4.3.1 Runtime

The runtime results show that CEVNE LiM converged faster than the k-shortest path algorithm. This explains the simple path selection algorithm of CEVNE LiM. In the Alevin framework, the runtime includes the configuration time for each test run. In ONOS controller, the configuration is run once, and its runtime is calculated separately. It takes the CEVNE LiM application about 3ms to embed each virtual link; and the total runtime is the multiplication of the number of virtual links in the VNR. Figure 5.17 shows that k-shortest path runtime is in the range of 60–70ms for the five test cases, while CEVNE LiM runtime is in the range of 10–30ms; 30ms is when the CEVNE LiM application is first loaded. This shows that the CEVNE LiM achieves it cost saving objective with the quick response time.

5.4.3.2 Average path length

The average path length results show that the CEVNE LiM approach always have the shorter substrate path length compared to the k-shortest path algorithm. This is the result of the CEVNE LiM application searching for the shortest paths between two endpoints. The k-shortest path approach configures the fabric as a normal bi-directed graph, and searches in the whole graph for each virtual link. This affects both the path length and the runtime.

In Figure 5.17, the CEVNE LiM algorithm has 2.0 as the average path length for any substrate paths that the virtual link is embedded on. The k-shortest path algorithm has the average path length in the range of 3.0–4.0. This shows that the CEVNE LiM achieves its cost saving objective, and its energy saving objective because the longer the path, the more active substrate nodes and more energy consumption.

Figure 5.17 presents the runtime and the average path length results of the two approaches: k-shortest path algorithm and CEVNE LiM application.

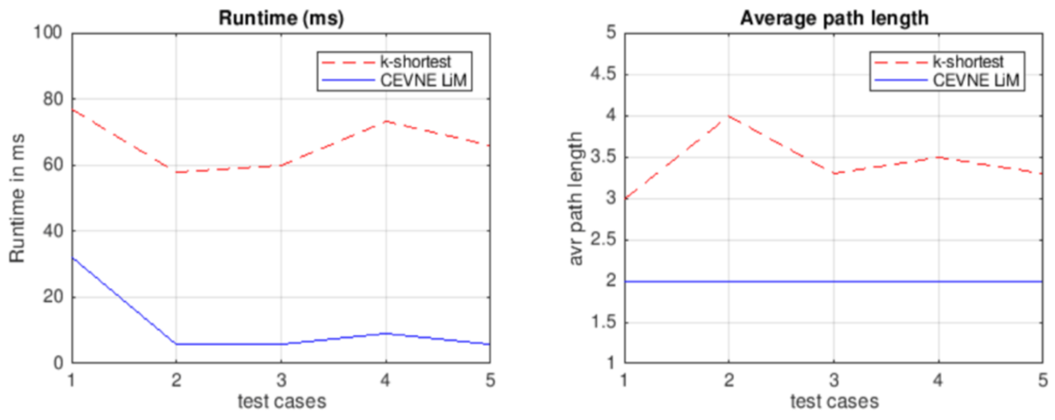


Figure 5.17: Results of runtime and average path length

5.4.3.3 Average node stress and average link stress

In the Alevin framework, the average node stress is the ratio of the total number of all node stress over the total number of substrate nodes. Similarly, the average link stress is the ratio of the total number of all link stress over the total number of substrate links. These are similar to the average node utilization and average link utilization introduced in (Chowdhury, Rahman & Boutaba 2012).

In CEVNE, the average node stress and average link stress are calculated for leaf and spine switches on the fabric. If each virtual node of a virtual link is connected to a different leaf switch, the average node stress involves three nodes: two leaf switches and one spine switch. The average link stress involves two substrate links: links between each leaf and spine switches. For congestion control purposes (Medhi 2017), the link mapping is spread out to different leaf and spine switches as implemented in the path selection algorithm. The average node stress and the average link stress results show that the CEVNE LiM algorithm distributes the link mapping onto more substrate nodes and links than the k-shortest path algorithm.

In Figure 5.18, the k-shortest path algorithm has the average node stress in the range of 0.2–0.3 while the CEVNE LiM average node stress is in the range 0.38–0.55. The k-shortest path algorithm has the average link stress in the range of 0.18–0.3 while the CEVNE LiM average link stress is in the range 0.25–0.38. The CEVNE LiM has on average 0.2 (20%) more node stress than the k-shortest path approach. The CEVNE LiM has on average 0.1 (10%) more link stress than the k-shortest path approach.

5.4.3.4 Energy consumption

The energy consumption results show that CEVNE LiM algorithm achieves its energy saving objective as specified in the average path length metric. As CEVNE LiM has shorter average path length than the k-shortest path algorithm, CEVNE LiM has a smaller number of active nodes along the paths, resulting in less power consumption. In Figure 5.18, the energy consumption of CEVNE LiM is 10 units lower on average compared to the k-shortest path algorithm.

5.4.3.5 Acceptance ratio

The acceptance ratios of both algorithms are the same, and at the 100% in all test cases. The results of runtime, average path length, average node stress, average link stress, and energy consumption show that CEVNE LiM algorithm achieves the three objectives and prevails over the k-shortest path algorithm in these objectives.

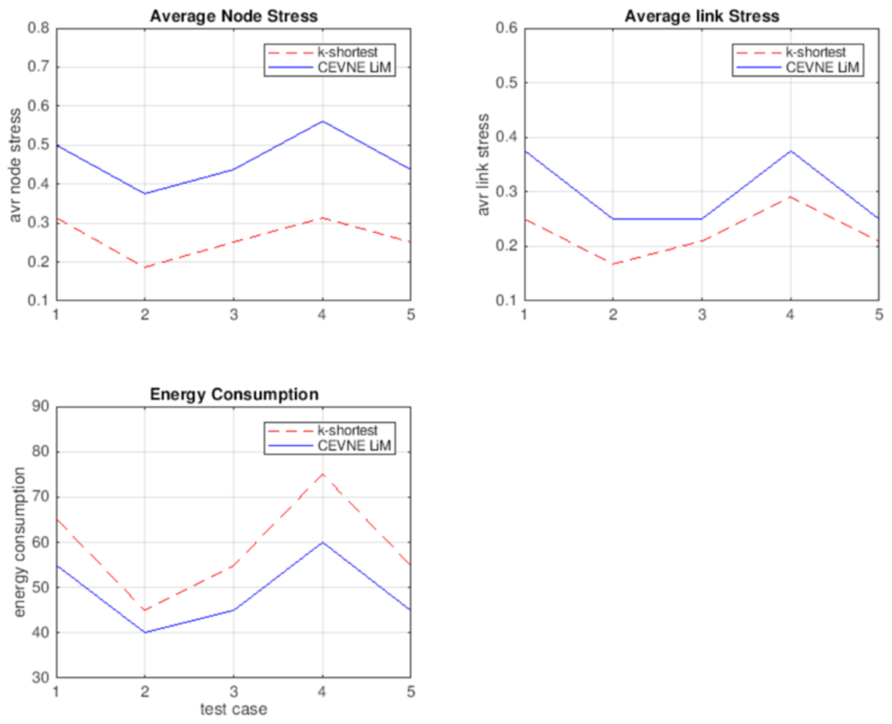


Figure 5.18: Results of average node stress, average link stress, and energy consumption

5.5 Summary

In this chapter, the research objective is to investigate and propose a mechanism to realize CEVNE link embedding process. CEVNE heuristic algorithm is a novel SDN-based algorithm, which is active, and can focus on multi-objectives: cost saving, energy saving and congestion avoidance as the node embedding process does. The novel realization applied the architecture for extended SDN application. CEVNE SDN-based link heuristic algorithm is realized as an SDN application on top of a leaf-spine fabric substrate network running SR. Each objective of the virtual link embedding is proposed as a micro-service; a path selection algorithm is proposed as a composite service that integrates all three objective services to choose the optimal path. The evaluation results show that CEVNE LiM application outperforms the k-shortest path algorithm in runtime, energy consumption, path length, average node stress and average link stress.

Our solution is not only restricted to use the leaf spine fabric, but it can be applied to other network topologies in the cloud data centres. The condition to apply our solution in different network topologies is presented as follows. We need to specialize the topology service into a sub-class of topology service that serves only the specific topology of the cloud data center. Further, we provide the in-memory routing of the topology so it can be accessed quickly by other services when we run our multi-objective link embedding algorithm.

CHAPTER 6 LATENCY-AWARE RESOURCE OPTIMIZATION FOR 5G CORE NETWORK SLICING

The previous chapter objective was to investigate and to propose the link mapping process, which is an active process as the node mapping process, and can focus on multi-objectives: cost saving, energy saving and congestion avoidance. The mechanism is based on SDN and SR technologies. The realization of the mechanism is an extended SDN application that integrates each objective as an application / service and uses a monitoring function; the application is deployed on the SDN substrate network running SR as a source routing mechanism to forward packets.

This chapter objective is to investigate and to propose a novel mathematical model for resource allocation optimization in network slicing satisfying latency requirement, applying SDN, NFV technologies, and cloud computing. We only investigate the 5G core network slicing. The mechanism is modelled as a mathematical program that optimizes resource allocation and satisfies the critical latency requirement of different 5G service types: eMBB, uRLLC and mMTC. The research outcome is the network slicing resource model, performance model, and mathematical model in 5G core networks, and the solution approach using the decomposition technique in large or structure mathematical programs.

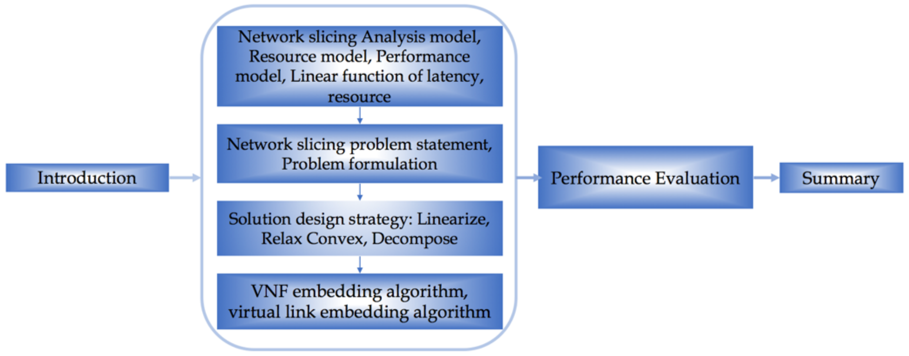

Introduction
Network slicing Analysis model, Resource model, Performance model, Linear function of latency, resource
Network slicing problem statement, Problem formulation
Solution design strategy: Linearize, Relax Convex, Decompose
VNF embedding algorithm, virtual link embedding algorithm
Performance Evaluation
Summary

Figure 6.1: Chapter 6 roadmap

In this chapter, section 1 presents the novel models of the 5G core network slicing problem, section 2 formulates the network slicing problem, section 3 presents the novel solution approach and the solution, section 4 presents the performance evaluation of the solution, and section 5 summarizes the chapter. This chapter has been submitted to the IEEE Transaction on Networks and Service Management with title "Latency-aware Resource Optimization in 5G core network slicing", after the peer reviewed process.

What is new in the solution?

The novelties of the network slicing solution lie in the areas as follows.

- ✓ The analysis, resource models of network slicing are as follows: using 5G theory, the analysis identifies how network slicing works to deliver the 5G core service. The resource model presents the allocation of CPU resource to VNFs to process packets, and bandwidth resource to virtual links connecting VNFs to deliver packets E2E of the network slice instance.
- ✓ The performance model predicts the VNF total delay using queueing theory as follows: The VNF is modelled as an $\mathtt{M_t/G/1}$ queueing system with the arrival queue using Markov modulated Poisson Process (MMPP).
- ✓ The mathematical model of the network slicing problem optimizes the resource allocation and satisfies different latency requirement of eMBB, uRLLC and mMTC applications.
- ✓ The mathematical program (MP) is non-linear, non-convex, mixed-integer, and NP-Hard. The novelty lies in the solution approach that applies linearization, convex-relaxation and decomposition of the MP into the master process and two sub-processes and solve them using the IBM Optimization Programming Language (OPL) flow control mechanism.

6.1 Modelling network slicing for mobile services in 5G core networks

6.1.1 Application classification in 5G

5G classifies applications into three main types: enhanced mobile broadband (eMBB), ultra-reliable low latency communication (uRLLC), and massive machine type communication (mMTC). 5G application types are classified according to their focus either on human centric, device centric, or every device, their vertical sector, and their key performance indicators (KPI). They are presented briefly as follows (Marsch et al. 2018), (ITU-R_M.2083 2015), (Foukas et al. 2017).

- eMBB focuses on the human-centric access to multi-media content, services, and internet, which have been provided by 4G. 5G facilitates creating new applications, improving performance and user experience in existing services. The new applications are either wide-area coverage or hotspot. eMBB focuses on the KPIs: spectrum efficiency, traffic density, power efficiency, mobility, and data rate.
- uRLLC is related to services with stringent requirements in latency and availability. It also focuses on the reliability, mobility, and data rate KPIs. This is the new type of 5G application, and it is utilized in new 5G vertical applications such as autonomous driving, next generation factories, virtual reality applications.
- mMTC is related to applications having a very large number of devices transmitting non-delay sensitive data in low volume. mMTC focuses on the connection density KPI.

6.1.2 Control plane data plane separation (CUPS)

Another characteristic of the 5G mobile network is the control plane (CP) user plane (UP) separation (CUPS). 3GPP proposed the logical separation between the signaling network and data network. 5G access-network and core-network functions are separated from the transport network functions (Penttinen 2019). The separation reduces the latency on the applications or services, supports increased data traffic, allows the updating, locating and scaling of CP and UP resource independently, and enables SDN to deliver user plane data more efficiently (Schmitt, Landais & YongYang 2018).

An E2E NS consists of access network slice, core network slice, and transport network slice. In this chapter, we focus on 5G core network slicing. We assume the network slicing infrastructure is at the edge cloud, which is situated very close to the base station (eNodeB); therefore, the latency is improved because of shorter transmission distance.

6.1.3 Network slicing and related concepts

In chapter 2, section 2.4, the overview about 5G mobile system, network services, network slicing and related concepts have been presented. From these 5G concepts, the analysis model of network slicing is built with VNF and SFC components constituting a 5G core network slicing; each VNF has a resource demand, and a processing latency threshold; each virtual link connecting VNFs has a transmission latency threshold; the whole NSR has a latency threshold that it must satisfy. Network slicing is designed to implement services of different types: *eMBB*, *uRLLC or mMTC,* which have very different latency thresholds. The example in Figure 6.2 consists of multiple network slices; each is embedded in their vertical application. The infrastructure layer provides the substrate resources for network slicing as presented in the example. The access networks and core networks are on top of the (hidden) transport networks. NS 1 belongs to service 1 (*eMBB* type) from Telstra, an Australian Telecommunication company, NS 2 belongs to service 2 in the ambulance eHealth (*uRLLC* type), NS 3 belongs to service 3 in the vehicle-to-vehicle service (*uRLLC* type), and NS 4 belongs to service 4 for users at a stadium in a sporting event (*eMBB* type).

A network slice instance has a recursive structure, it is constituted of several network slice subnets (subnet), each subnet is constituted of other subnets, a subnet is constituted of network functions (core and access network function) (TR_28.801 Jan. 2018).

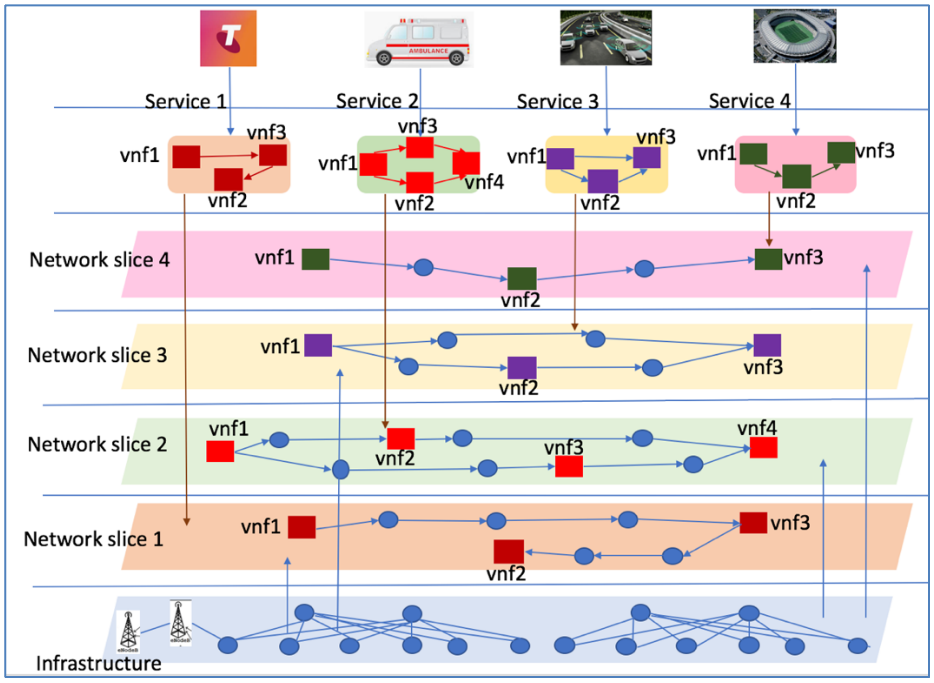


Figure 6.2: 5G network slice example - adapted from (Perez 2017)

6.1.4 Resource model of network slicing

A NS needs CPU, memory and storage resources to process packets, and bandwidth resource to propagate packets. A NS is constituted of an ordered set of VNFs; SFC place holders are ignored without effecting the NS functionalities. As it is designed in the forwarding graph (FG), a VNF can be invoked many times in an NSR, but the VNF needs to be allocated CPU resource at initialization. Similarly, the bandwidth resources are allocated to virtual links. The resources are released when the service is completed. The resource model is utilized to allocate resource efficiently and to satisfy the VNF delay with the contribution of the NS performance model that is described below.

6.1.5 Network slicing performance model using queueing theory

Table 6.1 defines the notations and their explanation that are used from here on.

Table 6.1: Notations and their explanation

$\mathbf{G_S = (V_S, E_S)}$	The directed graph represents the physical network with V_S substrate nodes, which include servers and switches / routers, and E_S substrate links
$\mathbf{V_{serv} \subset V_S}$	The set of servers in the physical network
$\mathbf{G_F = (V_F, E_F)}$	The directed graph represents the network slice request with V_F of VNFs and E_F of virtual links connecting between VNFs.
NSR	NSR includes all VNFs $\{f_0, f_1, .., f_{MH-1}\}$ the total number of VNFs is denoted as MH
$\mathbf{M_t/G/1}$	The Kendall notation of a queuing system that the arrival model is MMPP, the service model is general distribution, and 1 server only
$\boldsymbol{\lambda}$	The arrival rate in the M/M/1 queue
$\boldsymbol{\lambda^H}$	The high arrival rate in an MMPP process
$\boldsymbol{\lambda^L}$	The low arrival rate in an MMPP process
$\boldsymbol{\lambda^A}$	The arrival rate is calculated based on λ^H and λ^L in the queueing system $M_t/G/1$
$\boldsymbol{\mu^L - \mu^H}$	The processing rate of the VNF is in the range $[\mu^L - \mu^H]$
E(S)	The expected value of the service time
E(T)	The expected total delay in the system
E(W)	The expected waiting time in the system
$\boldsymbol{\rho}$	The utilization of the server
$\boldsymbol{\mu}$	The expected service rate
k	The number of equal parts of the processing rate between $[\mu^L - \mu^H]$
x	The variable denotes the rate between $[\mu^L - \mu^H]$, which is divided into k rates, $0 \leq x \leq k - 1$

$\mathbf{p_x}$	The probability that the service rate is x
$\mathbf{inc_{\mu}}$	The portion of service rate when $[\mu^L - \mu^H]$ is divided into k rates, $0 \leq x \leq k-1$
$\mathbf{E(S^2)}$	The expected value of the square of service time
$\mathbf{\overline{T}}$	The expected processing time
$\mathbf{c(m)}$	The max computing capacity of server $m \in V_{serv}$
$\mathbf{i}$	The index of the VNF in the NSR: $0 \leq i \leq (MH-1)$
$\mathbf{T_i}$	The total delay of VNF ith
$\mathbf{\overline{T}_i}$	The expected processing delay of VNF ith
$\mathbf{T_{tran}(f_i)}$	The transmission delay of VNF ith
$\mathbf{R_N(m)}$	The residual computing capacity of server $m \in V_{serv}$
$\mathbf{T_{NSR}}$	The total delay of the NSR
$\mathbf{bw(m,n)}$	The max bandwidth capacity of substrate link $(m,n) \in E_S$
$\mathbf{R_E(m,n)}$	The residual bandwidth capacity of substrate link $(m,n) \in E_S$
$\boldsymbol{\mathcal{P}_S}$	The set of all substrate paths in substrate network G_S
$\mathbf{p_{m,n}}$	The substrate path $p_{m,n} \in \mathcal{P}_S$ between two substrate nodes $(m,n) \in E_S$
$\mathbf{c(f_{(i)})}$	Computing demand of VNF $f_{(i)} \in NSR$
$\mathbf{bw(f_i, f_{(i+1)})}$	Bandwidth demand of virtual link $(f_{(i)}, f_{(i+1)})$
$\mathbf{a^{f(i)}}$	The parameter of the function f_i that denotes the linear function of processing delay on the allocated CPU resources
$\mathbf{b^{f(i)}}$	The parameter of the function f_i that denotes the linear function of processing delay on the allocated CPU resources
$\mathbf{D_n^{f(i)}}$	The processing latency for VNF $f_{(i)}$ when it is deployed on node $n \in E_S$
$\mathbf{D_{n,Max}^{f(i)}}$	The threshold processing latency for VNF $f_{(i)}$ when it is deployed on node $n \in E_S$ $D_n^{f(i)} \leq D_{n,Max}^{f(i)}$
$\mathbf{D_{n,Min}^{f(i)}}$	The minimal processing latency of VNF $f_{(i)}$ when it is deployed on node $n \in E_S$ $D_n^{f(i)} \geq D_{n,Min}^{f(i)}$
$\mathbf{D_{m,n}^{(f_{(i)},f_{(i+1)})}}$	The transmission delay of virtual link $(f_{(i)}, f_{(i+1)})$ on substrate link (m,n)
$\mathbf{D_{tran}^{(f_{(i)},f_{(i+1)})}}$	The threshold of transmission delay of virtual link $(f_{(i)}, f_{(i+1)})$
$\mathbf{D_{Max}^{NSR}}$	Threshold of max delay of NSR
$\mathbf{P_{Max}^{NSR}}$	Threshold of processing delay of NSR
$\mathbf{D_{tran}^{NSR}}$	Threshold of transmission delay of NSR

$\mathbf{k_n^{f(i)}}$	$k_n^{f(i)} \in \mathbb{R}$: a portion of CPU resource of node $n \in E_S$ is allocated to VNF f_i. It is a variable
$\mathbf{k_{n,Max}^{f(i)}}$	The max CPU resource to be allocated to VNF f_i
$\mathbf{k_{n,Min}^{f(i)}}$	The min CPU resource to be allocated to VNF f_i
$\mathbf{x_n^{f(i)}}$	$x_n^{f(i)} \in \{0,1\}$ is a binary variable, $x_n^{f(i)} = 1$ if VNF f_i is mapped on to substrate node $n \in V_{serv}$; otherwise $x_n^{f(i)} = 0$.
$\mathbf{h_{m,n}^{f(i),f(i+1)}}$	$h_{m,n}^{f(i),f(i+1)} \in \mathbb{R}$ A portion of bandwidth of link $(m, n) \in E_S$ allocated to virtual link $(f_{(i)}, f_{(i+1)})$
$\mathbf{R_{res}(f_i)}$	The total allocated resources for VNF ith includes CPU and bandwidth resources
$\mathbf{R_{res}(NSR)}$	The total allocated resources for the NSR includes CPU and bandwidth resources
$Z_n^{f(i)}$	the new variable is created to linearize the formulation $Z_n^{f(i)} = x_n^{f(i)} * k_n^{f(i)}$

6.1.5.1 Markov-modulated Poisson Process

In queueing theory, Markov-modulated Poisson process (MMPP) is the doubly stochastic Poisson process, in which the arrival rates $\lambda = (\lambda_0, \lambda_1, , , \lambda_{m-1})$ vary based on the states of the underlying Continuous Time Markov Chain (CTMC). MMPP is bivariate with $[J(t), N(t): t \in T], N(t)$ is the number of arrivals within a time interval $(0, t]$ $t \in T$, and $J(t)$ with $0 < J(t) < m$ is the state of CTMC. The MMPP can be reconstructed by changing the arrival rate of a Poisson process according to the m-state of a CTMC, which is independent of the arrival process. MMPP has been used to model processes whose arrival rates vary randomly over time (Fischer & Meier-Hellstern 1993), so it is applied to the network slicing performance model.

The VNF model is assumed two arrival rates: λ_0 for a duration $Exp(\alpha_0)$, and λ_1 for a duration $Exp(\alpha_1)$. Figure 6.3 presents the arrival process of MMPP, in which arrival rate varies starting at λ_0 then it is changed to λ_1 and back to λ_0. The MMPP process's state transition diagram with two arrival rates λ_0, λ_1 are depicted in Figure 6.4

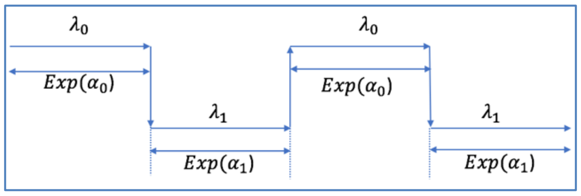


Figure 6.3: The arrival process of MMPP (Harchol-Balter 2013)

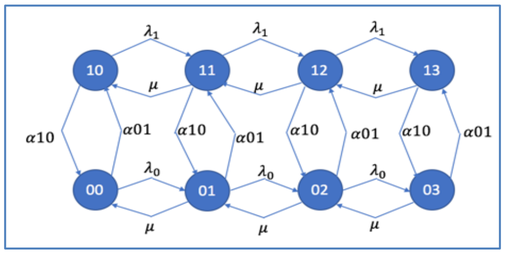


Figure 6.4: State transition diagram of a two-state MMPP model (Bolch et al. 2006)

6.1.5.2 VNF performance model

A VNF is modelled as a tandem server with two queues: (i) the NIC queue, and (ii) the central queue. The NIC queue is modelled as $M_t/M/1$ having the arrival queue as a Markov-modulated Poisson process (MMPP). Zinner et al. (2017) observed that the interrupt mechanism has been simplified to handle IO in less CPU time, therefore, the VNF central queue is simplified. Figure 6.5 presents the packet processing by X86 Linux kernel using the simplified interrupt mechanism.

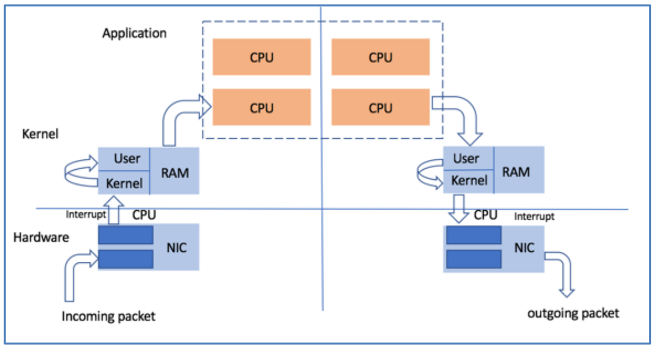


Figure 6.5: Packet processing by X86 Linux Kernel (Zinner et al. 2017)

The central queue is the VNF itself, which is modelled as a $M_t/G/1$ queue with the service queue is general distribution and 1 server, the order to the service is FCFS. The arrival queue accepts two rates λ^H and λ^L. Figure 6.6 presents the tandem server of the VNF queueing model.

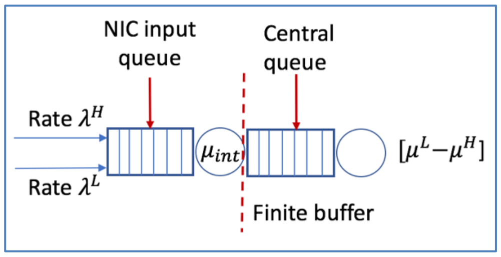


Figure 6.6: Tandem server with the NIC queue and central queue

Gupta et al. (June 2006) explored the performance model of the $M_t/M/1$ queueing system with varying-arrival rate as follows: λ^H in duration $\exp(\alpha^H)$, λ^L in duration $\exp(\alpha^L)$, and then back to λ^H in duration $\exp(\alpha^H)$ again. When both α^L, α^H are very high, the fluctuations are very rapid, the system converges to a single M/M/1 queueing system with the approximation of arrival rate λ^A is calculated based on $\lambda^H, \lambda^L, \alpha^H$ *and* α^L as follows:

$$\lambda^A = \frac{\frac{\lambda^H}{\alpha^H} + \frac{\lambda^L}{\alpha^L}}{\frac{1}{\alpha^H} + \frac{1}{\alpha^L}} \qquad (1)$$

This rapid fluctuation scenario is applied to the arrival pattern of the VNF in our 5G core network slicing as it reflects the quick changing of the arrival rates in 5G systems. The VNF Mt/G/1 queueing system is emerging to the $M/G/1$ system with the arrival rate of λ^A in equation (1) when the fluctuation is rapid.

The Pollaczek-Khinchine formulas for $M/G/1$ is used with the arrival rate λ, mean service rate μ, the mean service time $E(S)$, the expected delay in the system $E(T)$ is calculated as follows (Hariganesh 2018).

$$E(T) = E(S) + \left(\lambda. \frac{E(S^2)}{2(1-\rho)}\right) \qquad (2)$$

The right-hand side of equation (2) consists of two terms, the first term is the expected service time, the second term is the expected queueing delay $E(W)$. Applying equation (2) to the converged $\mathrm{M/G/1}$queue in rapid fluctuations, the expected system delay is calculated based on λ^A in equation (3) as follows.

$$E(T) = E(S) + \left(\lambda^A . \frac{E(S^2)}{2(1-\rho)}\right) \tag{3}$$

Figure 6.7: Visualization of the k rates between $\mu^H - \mu^L$

In the $M_t/G/1$ queue, the service rate is assumed to be discrete distribution into k rates ($k > 1$) between μ^L and μ^H with probabilities of $p_0, p_2, .., p_{k-1}$; the expected service rate is calculated based on all component rates as follows.

$$\mu = \sum_{x=0}^{k-1} p_x * \left(x * (\frac{\mu^H - \mu^L}{k-1}) + \mu^L\right) \tag{4}$$

Using the new constant inc_μ such that:

$$inc_\mu = \frac{\mu^H - \mu^L}{k-1} \tag{5}$$

$$\mu = \sum_{x=0}^{k-1} p_x * \left(x * inc_\mu + \mu^L\right) \tag{6}$$

The utilization ρ of the server is calculated using the arrival rate λ^A and the mean service time $E(S)$ as follows.

$$\rho = \lambda^A . E(S) \tag{7}$$

Applying equation (7) into equation (3), the mean system time E(T) calculated as follows.

$$E(T) = E(S) + \left(\lambda^A . \frac{E(S^2)}{2(1 - \lambda^A . E(S))}\right) \tag{8}$$

In equation (8), E(S) is the reciprocal of μ, which is calculated in equation (6); the expected system delay $E(T)$ is calculated based on the expected service time $E(S), E(S^2)$, and the approximation of arrival rates λ^A. As one or more NSIs are provisioned for a user service, the network traffic for each NSI is not many. When λ^A is small, the VNFs process packets at their arrival, so the waiting queue almost does not exist, the mean system time $E(T)$ is almost the mean service time $E(S)$ of VNF, which is a software application, that is presented in the subsection 6.1.7. The next subsection presents the VNF total delay in the E2E NSR.

6.1.6 VNF total delay

The total delay of a VNF f_i, which is denoted T_i, comprised the expected system time $\overline{\mathrm{T}}_i$ processed at the VNF, and the transmission delay $T_{tran}(f_i, f_{i+1})$ to the next VNF, as being expressed as follows.

$$T_i = \overline{\mathrm{T}}_i + T_{tran}(f_i, f_{i+1}) \tag{9}$$

The right-hand side of equation (9) consists the expected processing delay $\overline{\mathrm{T}}_i$, which is calculated based on equation (8) and the transmission delay $T_{tran}(f_i, f_{i+1})$. Equation (9) is applied to each VNF in the NSR.

Per the NS topology, simple chain, lightly-meshed and fully-meshed topologies are considered (Agarwal et al. 2018), which are depicted in Figure 6.8 as follows.

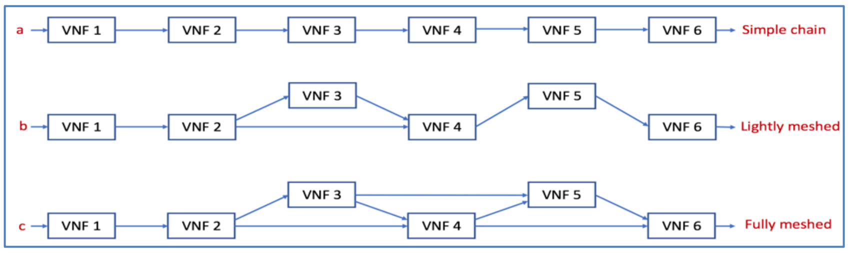


Figure 6.8: three NSR topologies (Agarwal et al. 2018)

In the simple chain (SC), VNFs are invoked in sequence, the total processing time T_{NSR} of the NS is the sum of processing time of all VNFs. In the lightly-meshed (LM) or fully-meshed (FM) topologies, the branching occurs at the VNF invocation based on runtime conditions, such as when VNF2 is completed, one of VNF 3 or VNF 4 can be invoked but not both; the total delay T_{NSR} is equal to the total delay of the longest branch that consists all VNFs. Of the presented topologies, the NSR total delay is calculated as follows.

$$T_{NSR} = \sum_{i=0}^{MH-1} T_i \leq D_{Max}^{NSR} \tag{10}$$

In equation (10), the NSR consists of (MH) VNFs, and the total delay T_{NSR} of the NS is limited by the D_{Max}^{NSR}. However, the dependence between the resource model and the performance model is important, which is presented as the dependence service speed on the allocated CPU resource, is presented next.

6.1.7 Linear function of delay on allocated resources

Alleg et al. (2017) assumed that the VNF processing delay is affected by its allocated resource; when a VNF is allocated sufficient resource $k_{n,min}^{f_i}$, it starts to operate with speed $D_{n,max}^{f_i}$ (point A). After that, the more resource is allocated to the VNF, the faster its processing speed is. The VNF reaches its highest processing speed $D_{n,Min}^{f_i}$ with the allocated resource $k_{n,Max}^{f_i}$, (point B), the processing speed will not increase although the allocated resource is increased. Figure 6.9 depicts this dependency.

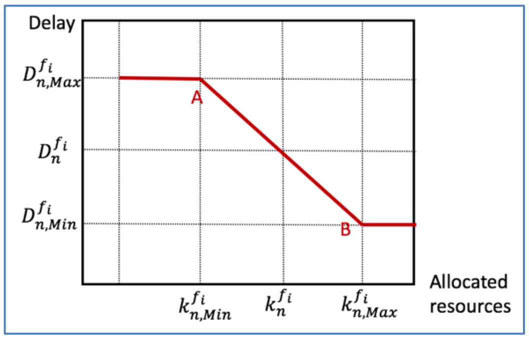


Figure 6.9: Processing delay is a linear function of allocated resources (Alleg et al. 2017)

$$k_{n,Max}^{f(i)} \geq k_{n}^{f(i)} \geq k_{n,Min}^{f(i)} \tag{11}$$

$$D_{n,Min}^{f(i)} \geq D_{n}^{f(i)} \geq D_{n,Max}^{f(i)} \tag{12}$$

The function of the processing speed to the allocated CPU resource is linear as denoted as follows:

$$D_{n}^{f(i)} = a^{f(i)} * k_{n}^{f(i)} + b^{f(i)} \tag{13}$$

The linearly proportion expressed in the parameters $a^{f(i)}$ and $b^{f(i)}$ are calculated as follows (Alleg et al. 2017):

$$a^{f(i)} = \frac{D_{n,Max}^{f(i)} - D_{n,Min}^{f(i)}}{k_{n,Min}^{f(i)} - k_{n,Max}^{f(i)}} \tag{14}$$

$$b^{f(i)} = \frac{D_{n,Min}^{f(i)} * k_{n,Min}^{f(i)} - D_{n,Max}^{f(i)} * k_{n,Max}^{f(i)}}{k_{n,Min}^{f(i)} - k_{n,Max}^{f(i)}} \tag{15}$$

One can leverage equations (13), (14), and (15) to allocate resources to a VNF and estimate its processing delay.

The transmission delay of VNF f_i, which is specified as $T_{tran}(f_i, f_{i+1})$ in equation (9), is calculated using the allocated bandwidth on each substrate link $(uv) \in p_{mn}$, where m, n are the substrate nodes that two VNFs are embedded on.

$$T_{tran}(f_i, f_{i+1}) = \sum_{(u,v)\in p_{m,n}} \frac{1}{h_{u,v}^{f(i),f(i+1)}} \tag{16}$$

The more bandwidth is assigned to the virtual link, the faster is the transmission delay $D_{m,n}^{(f_{(i)},f_{(i+1)})}$ to satisfy the transmission threshold $D_{tran}^{(f_{(i)},f_{(i+1)})}$ in the VNF f_i profile as follows.

$$D_{m,n}^{(f_{(i)},f_{(i+1)})} \leq D_{tran}^{(f_{(i)},f_{(i+1)})} \tag{17}$$

Each processing latency and transmission latency has its own threshold, one can normalize the unit of the CPU resource and the unit of the bandwidth resource into the common resource unit without affecting to the optimal result. Hence, the resource allocation of VNF f_i to fulfill the delay threshold is expressed in term of the allocated resources as follows:

$$R_{res}(f_i) = k_n^{f(i)} + \sum_{(u,v)\in p_{m,n}} h_{u,v}^{f(i),f(i+1)} \tag{18}$$

With $k_n^{f(i)}$ is bound to equation (11) and equation (12). Applying equation (18), the total allocated resources for NSR is the sum of the allocated resources of each VNF as follows.

$$R_{res}(NSR) = \sum_{i=0}^{MH-1} R_{res}(f_i) \tag{19}$$

$$R_{res}(NSR) = \sum_{n\in V_S} \sum_{i=0}^{MH-1} k_n^{f(i)} + \sum_{p_{m,n}\in \mathcal{P}_S} \sum_{i=0}^{MH-2} \sum_{(u,v)\in p_{m,n}} h_{u,v}^{f(i),f(i+1)} \tag{20}$$

The number of VNFs in the NSR is assumed to be MH as denoted in equations (19-20). Hence, the results in this subsection can be leveraged to formulate the postulated network slicing problem.

6.1.8 Solution approach

The solution approach of the network slicing problem consists of several steps to reach the solution. The solution approach is depicted in Figure 6.10. Several models were created to formulate the network slicing resource allocation problem. First is the establishment of the analysis model (block 1), which is input into the resource model (block 2) and the performance model (block 3). Block 2 and block 3 have connections between them; they are inputs to the numerical problem formulation (block 4). The network slicing problem is modelled using mathematical programming with objective function and constraints that will be transformed and solved using an optimization solver (block 5). Network slicing algorithms are evaluated in block 6.

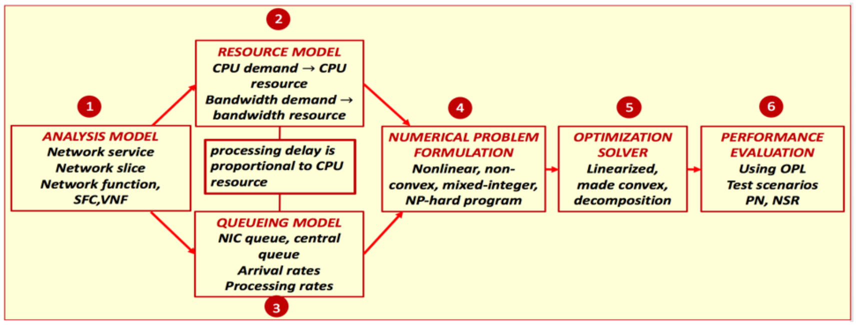

1
ANALYSIS MODEL
Network service
Network slice
Network function, SFC,VNF
2
RESOURCE MODEL
CPU demand → CPU resource
Bandwidth demand → bandwidth resource
processing delay is proportional to CPU resource
3
QUEUEING MODEL
NIC queue, central queue
Arrival rates
Processing rates
4
NUMERICAL PROBLEM FORMULATION
Nonlinear, non-convex, mixed-integer, NP-hard program
5
OPTIMIZATION SOLVER
Linearized, made convex, decomposition
6
PERFORMANCE EVALUATION
Using OPL
Test scenarios
PN, NSR

Figure 6.10: The solution roadmap for our network slicing problem

6.2 Problem statement

The notations and their explanation are presented in Table 6.1.

6.2.1 Network slicing problem statement

It is given a 5G network service that a user requires. The requested service belongs to one of three types of 5G applications: *eMBB*, *uRLLC*, or *mMTC*; each type of applications has the latency requirement about 10 to 100 times difference in magnitude (Dohler 2019). The requested network service must be implemented in a satisfactory manner. The network slice selection function (NSSF) (TR_28.801 Jan. 2018) translates the service request into a complete logical network that comprises an ordered set of VNFs, based on the user subscription.

It is given a physical network G_S with V_S substrate nodes, E_S substrate links, and $\mathcal{P}_S$ substrate paths, each substrate node has a CPU resource. Each substrate link has a bandwidth resource. Each substrate path has the bandwidth resource that is equals to the minimal of bandwidth resource of its constituent substrate links. The physical network is at an edge cloud close to the base station.

6.2.2 Network slicing: objectives

The network slicing optimization problem consists of three main objectives:

- To satisfy the network latency threshold when packets are processed by VNFs in NSR for the requested service, which can be one of the three types of applications: eMBB, uRLLC or mMTC in 5G core mobile networks.
- To minimize the CPU resource that is allocated to VNFs so they can process user flows.
- To minimize the bandwidth resource that is allocated along the substrate path connecting substrate nodes that two consecutives VNFs are embedded on.

With input data sets reflecting:

- The topology of the substrate network with nodes, links, node CPU capacities, link bandwidth capacities
- The topology of the NSR that constitutes of VNFs, their CPU demands, and bandwidth demands, the processing delay thresholds, and the transmission delay thresholds when packets are processed in the VNFs, and in the NSR for three types of 5G applications: eMBB, uRLLC, and mMTC.
- If a service can be translated into different NSRs, there may be a set of resource demands and thresholds for each NSR of the service (TR_28.801 Jan. 2018).

6.2.3 Problem formulation

In the problem domain, the processing latency and the transmission latency, each has its own threshold, therefore, they can be processed separately, and we can normalize the unit of CPU resource and bandwidth resource as the resource unit without affecting to the optimal result. We leave the cost of resources in monetary value to the Policy and Charging Unit of the mobile service. The objective is formulated as follows:

$$\min\left(\sum_{i=0}^{MH-1}\sum_{n\in V_S} k_n^{f(i)} * x_n^{f(i)} + \sum_{p_{m,n}}\sum_{i=0}^{MH-2}\sum_{(u,v)\in p_{m,n}} h_{u,v}^{f(i),f(i+1)}\right) \tag{21}$$

Subject to:

$$k_n^{f(i)} \geq 0, \forall i \leq (MH-1) \tag{22}$$

$$\sum_{i=0}^{MH-1} x_n^{f(i)} * k_n^{f(i)} \leq c(n), \forall n \in V_S \tag{23}$$

$$h_{u,v}^{f(i),f(i+1)} \geq bw_{u,v}^{f(i),f(i+1)}, \forall (u,v) \in E_S, \forall i \leq (MH-2) \tag{24}$$

$$\sum_{(u,v)\in p_{m,n}} \frac{1}{h_{u,v}^{f(i),f(i+1)}} \leq D_{tran}^{\left(f_{(i)},f_{(i+1)}\right)}, \qquad \forall i \leq (MH-2), \forall p_{m,n} \in \mathcal{P}_S \tag{25}$$

$$\sum_{n\in V_S} x_n^{f(i)} = 1, \qquad \forall i \leq (MH-1) \tag{26}$$

$$x_n^{f(i)} \in \{0,1\}, \qquad \forall i \leq (MH-1) \tag{27}$$

$$h_{m,n}^{f(i),f(i+1)} > 0 \; \boldsymbol{if} \left(x_m^{f(i)} * x_n^{f(i+1)}\right) = 1, \forall i \leq (MH-2) \tag{28}$$

$$\sum_{i=0}^{MH-2} \sum_{(u,v)\in p_{m,n}} h_{u,v}^{f(i),f(i+1)} \leq bw(u,v), \; \forall \left(p_{m,n}\right) \in \mathcal{P}_S \tag{29}$$

$$k_n^{f(i)} \geq c\left(f_{(i)}\right), \qquad \forall i \leq (MH-1) \tag{30}$$

$$\left(\sum_{n\in V_S} \sum_{i=0}^{MH-1} (a^{f(i)} * k_n^{f(i)} + b^{f(i)}\right) * x_n^{f(i)} + \sum_{i=0}^{MH-2} \sum_{(u,v)\in p_{m,n}} \frac{1}{h_{u,v}^{f(i),f(i+1)}}) \leq D_{Max}^{NSR} \tag{31}$$

The mathematical model of the network slicing problem is NP-Hard, non-convex, mixed-integer, nonlinear program (MINLP) as validated by Bliek1ú, Bonami & Lodi (Oct. 2014) because they have binary variables in the quadratic term. MP solvers such as IBM CPLEX and AIMMS have investigated methods to solve non-convex MINLP. This is further explored in our solution strategy in section 6.3.5.

The objective function (21) attempts to minimize the CPU and bandwidth resources that are allocated to the VNFs and to virtual links (the resources are normalized into resource unit). The first term denotes the CPU resource allocation to VNFs; each VNF f_i is allocated $k_n^{f(i)}$ CPU resource from the substrate node $n \in V_S$; it does not consider the parameters $a^{f(i)}$ in equation (14) and $b^{f(i)}$ in equation (15) because the two parameters will be leveraged in the delay constraints (31). The second term is the bandwidth resource allocation to virtual links connecting VNFs, each virtual link $(f_i, f_{i+1}))$ is allocated the bandwidth resource $h_{u,v}^{f(i),f(i+1)}$ on all the substrate links

$(u,v) \in E_S$ of the substrate path $p_{m,n} \in \mathcal{P}_S$ having two ends (m,n) where the VNF f_i and VNF $f_{(i+1)}$ are embedded to.

Constraint (22) guarantees that the allocated CPU resources for every VNF f_i is positive. Constraint (23) ensures that the allocated CPU resources respect the CPU capacity of the substrate node in which the VNFs are embedded. Constraints (24) guarantees that the allocated bandwidth resource for the virtual link $(f_i, f_{i+1)})$ must satisfy its demand. Constraint (25) ensures that the allocated bandwidth for virtual link $(f_i, f_{i+1)})$ will result in the satisfaction of the transmission threshold. Constraint (26) guarantees that VNF f_i is embedded on one substrate node only. Constraint (27) ensures that $x_n^{f(i)}$ is a binary variable. Constraint (28) is a conditional constraint expressing that $h_{m,n}^{f(i),f(i+1)}$ is positive if both binary variables $x_m^{f(i)}$ and $x_n^{f(i+1)}$have a value of 1. Constraint (29) guarantees that all the allocated bandwidth for the virtual link connecting VNF f_i and VNF f_{i+1} respects the substrate link's bandwidth capacity. Constraint (30) ensures that the allocated CPU resource for each VNF f_i must satisfy the CPU demand of the VNF. Constraint (31) consists of two terms, the first term is the total processing time of all VNFs in the NSR expressing as a function of $a^{f(i)}$ and $b^{f(i)}$ as calculated in equations (14-15), the second term is the total transmission delay on all virtual links as results of the bandwidth allocation on the substrate links; the total of processing delay and transmission delay of the NSR must satisfy the NSR total delay threshold.

6.3 Solution design

In this research work, an innovative solution design is proposed in order to get optimal results for the networks slicing problem, which consists of two parts. The first part explains the strategy to solve our proposal as a non-convex, NP-Hard, MINLP, and the second presents the detailed solution.

6.3.1 Solution strategy

The non-convex, NP-Hard, MINLP (Bliek1ú, Bonami & Lodi Oct. 2014) can be solved on a case-by-case basis, which shows that it is a steep learning curve to investigate and build the algorithms (Rardin 2017); an example of this approach is the comprehensive thesis about latency in Telecommunication by Hijazi (2010). IBM and AIMMS recommend automation approaches so they can be leveraged in different optimization problems, hence, to save cost and time.

6.3.2 Convex relaxation

One approach is called convex relaxation, the term that causes the MP non-convex, non-linear is linearized. To one such case, the term is a multiplication (M) of a binary variable and a continuous variable (Bliek1ú, Bonami & Lodi Oct. 2014), M is replaced with a new variable that has the same upper bound and lower bound as the continuous one.

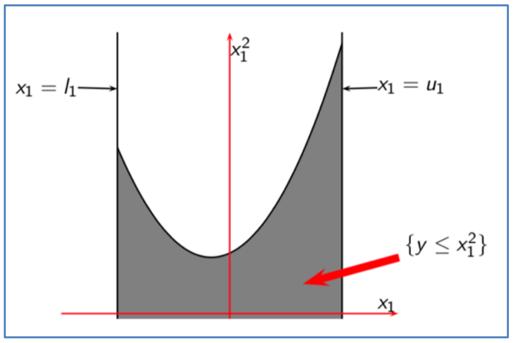


Figure 6.11: Non-convex constraint $y \leq x_1^2$

We will search for the non-convex, non-linear terms in the MP and linearize them. An example of the linearize approach is presented (Bliek1ú, Bonami & Lodi Oct. 2014), the non-convex constraint $y \leq x_1^2$ between $[l_1, u_1]$ is convex relaxed to a new constraint $y \leq y_{11} = (l_1 + u_1)x_1 - l_1 u_1$ as in Figure 6.12

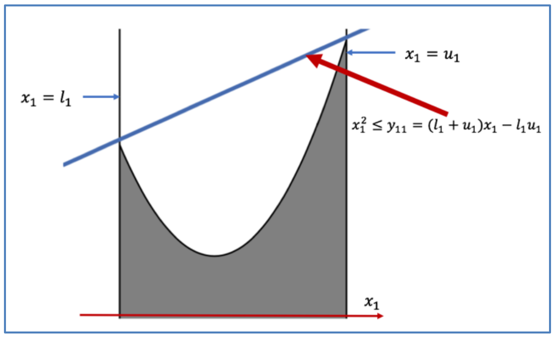


Figure 6.12: Convex relaxation a linear constraint

6.3.3 Decomposition

The model of network slicing is a structure model with two sub-processes: the VNF resource allocation is followed by the virtual link embedding. The reverse order is also correct if the virtual link embedding starts first, its results are applied to the VNF resource allocation process. But the two sub-processes could not occur independently. Therefore, it is suitable for the decomposition technique proposed by Lasdon (1970). The VNF allocation process and virtual link embedding process are considered sub-problems. A master problem (Lasdon 1970) (Palomar & Chiang 2006) is required to transfer data between them.

The IBM OPL flow control mechanism is designed to implement the decomposition in large programs. It allows to decompose a model into smaller, more manageable models and solve them with the control of the master model (Arkalgud & Rux 2018); therefore, it will generate an optimal solution of the original model. The flow control uses OPL script to define the master problem and sub-problems. The master problem transfers results between sub-problems using the OPL script "pre-processing" and "post-processing" functions. Our network slicing problem utilizes the IBM OPL flow control as the decomposition mechanism and is presented in Figure 6.13.

The algorithms VNF resource allocation optimization, and virtual link embedding are utilized to build the corresponding sub-models in OPL script. The master problem is the main script that connecting to both sub-models. The three algorithms are presented in following paragraphs. Using the flow control provided by IBM OPL, the main and sub-models will generate optimal solutions (Dantzig 1998), (Lasdon 1970). As our network slicing program has been convex-relaxed and linearized, the solutions are only sub-optimal.

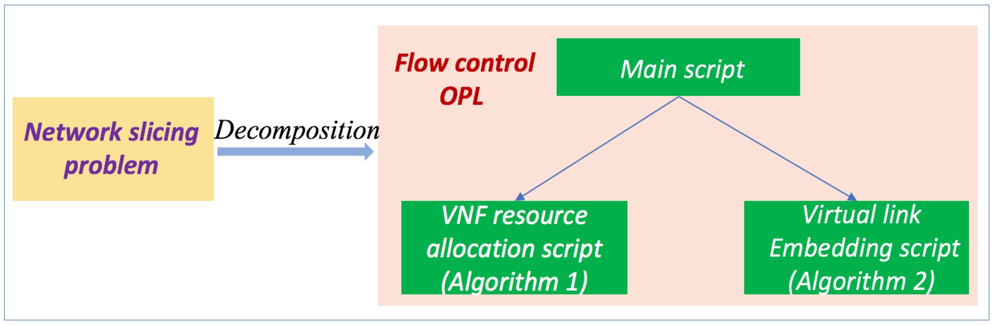

Flow control OPL
Main script
Network slicing problem
Decomposition
VNF resource allocation script (Algorithm 1)
Virtual link Embedding script (Algorithm 2)

Figure 6.13: The decomposition of the network slicing optimization problem

6.3.3.1 Strategy for resource optimization

The second part focuses on the strategy to achieve optimal resource allocation and latency threshold satisfaction. To minimize the allocated resource, we set the VNF CPU demand at the processing delay threshold using equations (13), (14) and (15), and the bandwidth demand at the transmission delay threshold using equation (16). The approach is feasible because the VNE embedding and virtual link embedding sub-problems have been linearized. This strategy might have advantage over FRAM (Alleg et al. 2017), which is a MIQCP, and hence it allocates resources fluctuating at or above the thresholds.

6.3.4 Linearized problem formulation

In the constraint (23), the product of a binary variable $x_n^{f(i)}$ and a continuous variable $k_n^{f(i)}$ causes the problem non-convex (Bliek, Bonami and Lodi 2014), these variables are replaced with a new variable $Z_n^{f(i)}$ (Rubin 2010) as follows.

$$Z_n^{f(i)} = x_n^{f(i)} * k_n^{f(i)} \tag{32}$$

To ensure the optimization problem is linear, additional constraints are added so $Z_n^{f(i)}$ will have the same upper bound U and lower bound L (Rubin 2010) as variable $k_n^{f(i)}$.

$$Z_n^{f(i)} \leq U * x_n^{f(i)} \tag{33}$$

$$Z_n^{f(i)} \geq L * x_n^{f(i)} \tag{34}$$

$$Z_n^{f(i)} \geq k_n^{f(i)} - U * \left(1 - x_n^{f(i)}\right) \tag{35}$$

$$Z_n^{f(i)} \leq k_n^{f(i)} - L * \left(1 - x_n^{f(i)}\right) \tag{36}$$

When $x_n^{f(i)} = 0$, it needs to prove that the product $Z_n^{f(i)} = x_n^{f(i)} * k_n^{f(i)}$ should be 0. The first two formulas (33), (34) give $0 \leq Z_n^{f(i)} \leq 0$, or $Z_n^{f(i)} = 0$; the second two formulas (35), (36) give $(k_n^{f(i)} - U) \leq Z_n^{f(i)} \leq (k_n^{f(i)} - L)$; $Z_n^{f(i)} = 0$ is the solution because U, L are bounds of $k_n^{f(i)}$.

When $x_n^{f(i)} = 1$, it needs to prove that $Z_n^{f(i)} = k_n^{f(i)}$. The first two formulas (33) and (34) give: $L \leq Z_n^{f(i)} \leq U$, which is satisfied if $Z_n^{f(i)} = k_n^{f(i)}$; the second two formulas (35) and (36) give $k_n^{f(i)} \leq Z_n^{f(i)} \leq k_n^{f(i)}$, which means $Z_n^{f(i)} = k_n^{f(i)}$.

In constraint (31), in the first term, the non-linear term is replaced with $Z_n^{f(i)}$ as in equation (32), the second term is replaced with the transmission threshold $\mathrm{D}_{\mathrm{tran}}^{\mathrm{f(i),f(i+1)}}$ in the constraint (25). Therefore, constraint (31) is consolidated to constraint (37) as follows:

$$(\sum_{n\in V_S}\sum_{i=0}^{MH-1}\left(a^{f(i)} * \boldsymbol{Z_n^{f(i)}} + b^{f(i)}\right) + \sum_{i=0}^{MH-2} D_{tran}^{f(i),f(i+1)}) \leq D_{Max}^{NSR} \tag{37}$$

6.3.5 Algorithms

The MPs *are decomposed* into the VNF mapping and virtual link mapping algorithms because each process is in their own domain so the integrity of the MP is ensured (Williams 2013). A master problem is created to control the two algorithms (Lasdon 1970). Each algorithm is presented in the next sub-sections.

6.3.5.1 The master algorithm

The master algorithm controls the sub-algorithms, it invokes the VNF embedding and transfers the results to the virtual link embedding before invoking it. The master algorithm is presented in Figure 6.14.

6.3.5.2 VNF embedding optimization algorithm

The VNF embedding optimization algorithm will optimize the CPU allocation for all VNFs and meet the processing constraint and other constraints. The objective function and constraints are extracted from section 4.3 and section 5.2, are presented as follows:

$$\min\left(\sum_{i=0}^{MH-1}\sum_{n\in V_S} Z_n^{f(i)}\right) \tag{38}$$

$$Z_n^{f(i)} = x_n^{f(i)} * k_n^{f(i)} \tag{39}$$

$$k_n^{f(i)} \geq 0, \forall i \leq (MH-1) \tag{40}$$

$$k_n^{f(i)} \ \leq cpu(n), \forall n \in V_S, \forall f(i) \in NSR \tag{41}$$

$$\sum_{n\in V_S} x_n^{f(i)} = 1\,, \qquad \forall i \leq (MH-1) \tag{42}$$

$$x_n^{f(i)} \in \{0,1\}, \qquad \forall i \leq (MH-1) \tag{43}$$

$$k_n^{f(i)} \geq cpu\left(f_{(i)}\right), \forall i \leq (MH-1) \tag{44}$$

$$\left(\sum_{n \in V_S} \sum_{i=0}^{MH-1} (a^{f(i)} * Z_n^{f(i)} + b^{f(i)}\right) \leq P_{Max}^{NSR} \quad (45)$$

$$Z_n^{f(i)} \leq U * x_n^{f(i)} \quad (46)$$

$$Z_n^{f(i)} \geq L * x_n^{f(i)} \quad (47)$$

$$Z_n^{f(i)} \geq k_n^{f(i)} - U * \left(1 - x_n^{f(i)}\right) \quad (48)$$

$$Z_n^{f(i)} \leq k_n^{f(i)} - L * \left(1 - x_n^{f(i)}\right) \quad (49)$$

The explanation of formulas (38-49) is alike to formulas (21-31). The objective function (38) minimizes the allocated CPU resource to VNFs, ensures processing constraint and other constraints. Constraint (39) is the newly added variable $Z_n^{f(i)}$ as in section 5.2. Constraint (40) guarantees the allocated CPU resource is positive. Constraint (41) ensures that the allocated CPU resource respects the CPU capacity of substrate node n. Constraint (42) guarantees that VNF f_i is embedded on one substrate node only. Constraint (43) ensures that $x_n^{f(i)}$ is a binary variable. Constraint (44) guarantees that the allocated CPU resource satisfies the CPU demand of f_i. Constraint (45), which is extracted from the delay constraint (31), ensures the allocated CPU resource to all VNFs satisfying the total processing delay threshold of the NSR. Constraints (46-49) are extracted from section 5.2 about the variable $Z_n^{f(i)}$.

If there are optimal results in the VNF embedding process, we will proceed to virtual link embedding. The flow control mechanism in OPL (Arkalgud & Rux 2018) is leveraged to transfer the VNF embedding results. Figure 6.15 presents the VNF embedding optimization process, which consists of the input parameters, the execution of the VNFmapping.mod file that consists of formulas (38-49) coded using OPL script syntax, and the output results.

Figure 6.14: the algorithm of the master model

Algorithm 1: The main script	
1	Function main
2	declare data files
3	declare structure mappedVL
4	declare set mappedVLlink of type mappedVL

```
5       generate the main_OPL_model
6       r1=invoke VNF_embedding;
7       if (r1 is solved)
8           mappedVLlink=r.VNF_results
9           r2=invoke virtual_link_embedding
10          if (r2 is solved)
11              exit with success status
12          Else
13              exit with fail status
14          end if
15      Else
16            exit with fail status
17      end if
18    End Function
```

Figure 6.15: the algorithm of the VNF embedding

Algorithm 2: VNF embedding algorithm

```
1     Input:
2         Infratructure_networks;
3        VNFs_demand[numOfVnfs];
4        VNF embedding model file .mod
5        VNF embedding data file
6     Output:
7        VNF Optimal embedding results
8     Function VNF mapping
9        OpenFile VNFmapping.mod file
10       create_vnfmapping.dat;
11       Execute VNFmapping.mod with vnfmapping.dat
12       If status <> 1    // status 1 is internal value of OPL showing there are
      optimal results
13            Exit with failure
14       Else
15           Save_Results(optimalVNFs[numOfVnfs]);
16           Exit
17       Endif
18    End function
```

```
Function create_vnfmapping.dat;
OpenFile vnfmapping.dat;
fwrite (Infrastructure_networks, vnfmapping.dat);
fwrite (VNFs_demand[numOfVnfs], vnfmapping.dat);
SaveFile
End function
Function Save_Results(optimalVNFs[numOfVnfs])
   For each optimalVNFs
       virtualLinkR = new VirtualLinkRequest;
       virtualLinkR.serverSource = server_id[vnf_source];
       virtualLinkR.serverDest=server_id[vnf_destination];
       virtualLinkRequests.add(virtualLinkR);
   End For
   Return virtualLinkRequests;
End function
```

6.3.5.3 Virtual links connecting VNFs Embedding optimization algorithm

The objective function and constraints of the virtual link embedding process are extracted from the original ones in section 4.3, and section 5.2, as are presented in Figure 6.16.

$$\min\left(\sum_{p_{m,n}}\sum_{i=0}^{MH-2}\sum_{(u,v)\in p_{m,n}} h_{u,v}^{f(i),f(i+1)}\right) \tag{50}$$

$$h_{u,v}^{f(i),f(i+1)} \geq bw_{u,v}^{f(i),f(i+1)}, \forall(u,v)\in E_S, \forall i \leq (MH-2) \tag{51}$$

$$\sum_{i=0}^{MH-2}\sum_{(u,v)\in p_{m,n}} h_{u,v}^{f(i),f(i+1)} \leq bw(u,v),\ \ \forall(p_{m,n})\in \mathcal{P}_S \tag{52}$$

$$h_{m,n}^{f(i),f(i+1)} > 0\ \ if\ \ \left(x_m^{f(i)} * x_n^{f(i+1)}\right) = 1, \forall i \leq (MH-2) \tag{53}$$

$$\sum_{i=0}^{MH-2} D_{tran}^{f(i),f(i+1)} \leq D_{tran}^{NSR} \tag{54}$$

The explanation of the formulas (50-54) is alike to formulas (21-31). The objective function (50) guarantees that the allocated bandwidth resource to all virtual links is minimized. Constraint (51) ensures that the allocated bandwidth fulfills the bandwidth demand of the virtual link connecting the VNFs f_i and f_{i+1}. Constraint (52) guarantees that the allocated bandwidth respects the bandwidth capacity of substrate links on the embedded substrate path. Constraint (53) ensures that the bandwidth is allocated only if the two substrate nodes are the embedded nodes of VNFs f_i and f_{i+1}. Constraint (54) is the transmission delay constraint that is related to the linearization of the NSR delay in section 5.2.

In the realization process, the 5G Core management system will employ the SDN controller to create a virtual LAN for the NSR with two purposes: (i) to route packets in the NSR, and (ii) to isolate the NSR from other NSRs on the same infrastructure. Algorithm 3 presents the virtual link embedding optimization algorithm with input parameters, optimal VNF embedding results, the execution of the VirtualLinkMapping.mod file that consists of formulas (50-54) coded using OPL script syntax, and output results.

Figure 6.16: the virtual link embedding algorithm

Algorithm 3: Virtual link embedding optimization algorithm

```
Input:
    Virtual link mapping demands
    VNF optimal mapping results
    Virtual link mapping model file .mod
    Virtual link mapping data file
Output:
    Optimal virtual link mapping results
Function virtual link mapping
    OpenFile virtuallinkmapping.mod
    virtualLinkRequests= LOAD_Results;
    create_VirtualLinkMapping.dat;
    Execute VirtualLinkMapping.mod with VirtualLinkMapping.dat
    If status <> 1   // status 1 is internal value of OPL showing there are
optimal results
        Exit with failure
    Else Get optimalVirtualLinkMapping[numOfLinks];
    End if
End function
```

```
Function create_ VirtualLinkMapping.dat;
  OpenFile VirtualLinkMapping.dat;
  Fwrite (Infrastructure_networks, virtualLinkMapping.dat);
  Fwrite (virtualLinkRequests, virtualLinkMapping.dat);
  Fwrite (virtualLinkDemands[numOfVirtualLinks],
virtualLinkMapping.dat)
  SaveFile
End function
```

6.3.5.4 Multiple NSR embedding optimization algorithms in a time window

When multiple NSRs are requested by different mobile services, they are processed in the order of arrival to the 5G Core system; each NSR is provisioned in two steps: VNFs embedding optimization, and virtual link embedding optimization. After the NSR is successfully provisioned, the residual substrate resources of nodes and links are calculated before processing the next NSR.

6.4 Performance evaluation

In this section, we use realistic scenarios to evaluate the performance of our algorithms and compare with the-state-of-the-arts. The performance parameters in the scenario (Malandrino et al. Dec. 2019) are extracted and used to design the NSR requests. The traffic models are based on the scenarios for the entertainment (EN) service for eMBB type and the intersection collision avoidance (ICA) service for uRLLC type, in which the max delay and CPU demands are extracted into the NSRs.

Table 6.2: Realistic scenarios (Malandrino et al. Dec. 2019)

VNF	**Rate $\lambda(s,v)$**	**Resource for 1 flow/ms $l(v)$**	**Service rate (CP and UP)**	**CPU demand (CP)**
Intersection collision avoidance (ICA)				
eNB	117.69	10^{-4}	152.997	15.3 / 10
EPC PGW	117.69	10^{-4}	152.997	15.3 / 10
EPC SGW	117.69	10^{-4}	152.997	15.3 / 10
EPC HSS	11.77	10^{-4}	15.301	1.5 / 10
EPC MME	11.77	10^{-3}	15.301	15.3 / 10

Car info management	117.69	10^{-3}	152.997	153**
Collision detector	117.69	10^{-3}	152.997	153**
Car manufacturer DB	117.69	10^{-4}	152.997	15.3 / 10
Alarm generator	11.77	10^{-4}	15.301	1.5 / 10
Entertainment (EN)				
eNB	179.82	10^{-4}	233.766	23.4 / 20
EPC PGW	179.82	10^{-4}	233.766	23.4 / 20
EPC SGW	179.82	10^{-4}	233.766	23.4 / 20
EPC HSS	17.98	10^{-4}	23.374	2.4 / 20
EPC MME	17.98	10^{-3}	23.374	23.4 / 20
Video origin server	17.9	10^{-3}	23.27	23.3 / 20
Video CDN	179.82	10^{-4}	233.766	23.4 / 20

** these are left out as they are image-processing functions

6.4.1 The evaluation setup

6.4.1.1 The infrastructure

The infrastructure is generated as a three-layer leaf-spine fabric in an edge cloud. The fabric has two spine switches (S1 and S2) in layer 1, leaf switches in layer 2, and servers in layer 3. We assume VNFs will be embedded onto the servers only. The infrastructure is designed with two sizes: SN1 and SN2 (SN1 is smaller) as depicted Figure 6.17. The SNs for eMBB service and uRLLC service are also presented. Figure 6.17 presents the fabric SNs.

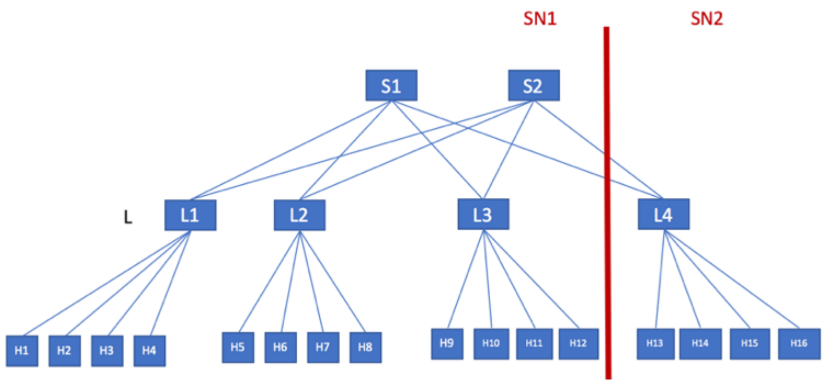


Figure 6.17: The leaf-spine fabric test beds

Table 6.3 presents the infrastructure with the number of servers and substrate links, the substrate resources of servers, routers and substrate links.

Table 6.3: The substrate fabric network

	CPU resources (GHz)	BW Resources (Gbps)	
SN1			S
Servers	10	0	12RT
Routers	0	0	5LK
Sub Links		20	36LK
SN2			
Servers	10		16SV
Routers	0	0	6RT
Sub Links		20	48LK

6.4.1.2 The network slice requests

NSR are designed based on three topologies of SC, LM and FM presented in Figure 6.8. Services are designed to have 6 VNFs. The three topologies for services that consist of 8 VNFs are depicted in Figure 6.18. The NSR demands are extracted from the realistic scenarios.

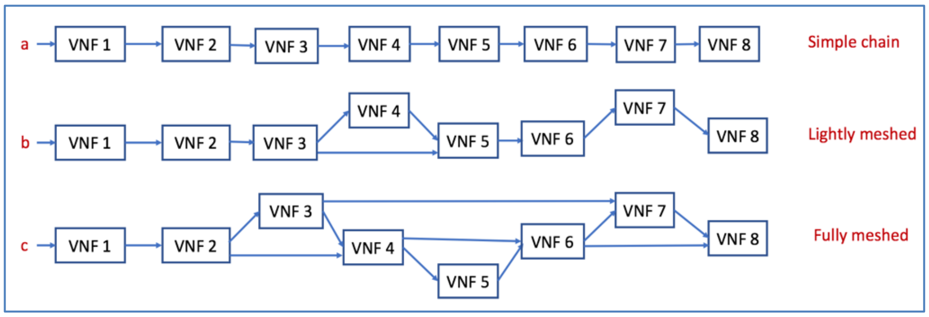


Figure 6.18: The NSR topologies with 8 VNFs

Table 6.4 presents the NSRs in each test scenario, the parameters are calculated from the realistic scenarios in Table 6.2; ICA services are of type uRLLC, EN services are of type eMBB. The parameters are inferred from Table 6.2 to be used in the synthetic scenarios as follows.

Table 6.4: NSR demand and thresholds

	Number of VNFs		CPU Demand (GHz)	BW Demand (Gbps)	NSR delay (ms)	Trans delay (ms)
Entertainment (EN) (eMBB)						
SC	6	8	1.1	1	200	10
LM	6	8	1.1	1	200	10
FM	6	8	1.1	1	200	10
ICA (uRLLC)						
SC	6	8	5.5-5.9	4 – 5	50	0.01
LM	6	8	5.5-5.9	4 – 5	50	0.01
FM	6	8	5.5-5.9	4 – 5	50	0.01

Table 6.5: Test scenario design

	Topology	VNF	Application type
Scenario 1	Simple chain	6	eMBB
Scenario 2	Simple chain	6	uRLLC
Scenario 3	Lightly meshed	6	eMBB
Scenario 4	Lightly meshed	6	uRLLC
Scenario 5	Fully meshed	6	eMBB
Scenario 6	Fully meshed	6	uRLLC
Scenario 7	Simple chain	8	eMBB
Scenario 8	Simple chain	8	uRLLC
Scenario 9	Lightly meshed	8	eMBB
Scenario 10	Lightly meshed	8	uRLLC
Scenario 11	Fully meshed	8	eMBB
Scenario 12	Fully meshed	8	uRLLC

6.4.1.3 The simulation environment

The simulation environment is the OPL IDE, in a MacBook Pro 2.2 GHz, Intel Core i7, 16G RAM. We design the model files (.mod) for the main script, VNF mapping and the virtual link mapping using their objective functions and constraints in subsections 6.3.5.2 and 6.3.5.3. The data files are created for each embedding process consisting of the substrate resources and NSR demands as described in Table 6.3 and Table 6.4. The OPL IDE is used to view the results of the script execution. Table 6.5 presents 12 scenarios that are designed for eMBB and uRLLC services in three topologies SC, SM and FM.

6.4.2 Evaluation results

The performance of our network slicing algorithms are evaluated against the State-Of-The-Arts with SRAM and FRAM (Alleg et al. 2017), and the confident placement MaxZ and CPU allocation algorithms (Agarwal et al. 2018) in resource allocation (CPU or bandwidth), latency satisfaction, execution time and the effect of arrival rates.

6.4.2.1 The embedding based on NSR topologies

The NSR SC is embedded as in Figure 6.19, VNFs are embedded on servers having the available resources meet the VNF demand, virtual links are embedded to the substrate links having the available resource meet the bandwidth demand.

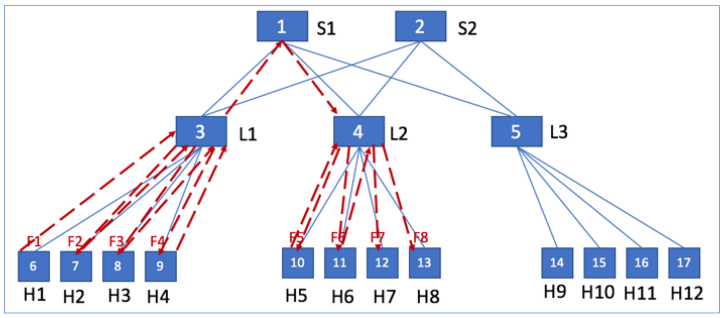


Figure 6.19: Results of the simple chain topology

The NSR LM topology is embedded as in Figure 6.20, the branches are mapped to the substrate links having the available bandwidth meet the bandwidth demand.

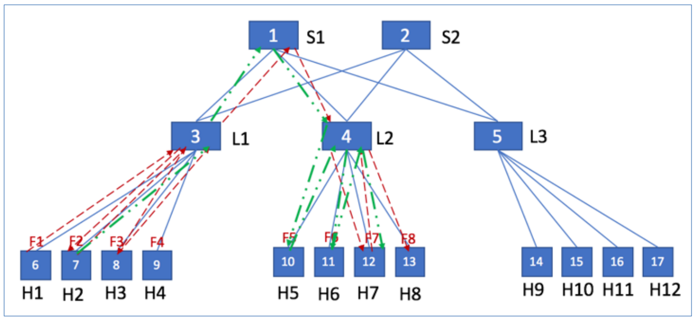


Figure 6.20: Results of the lightly meshed topology

6.4.2.2 Resource allocation by service types

The allocated CPU and bandwidth resources result in the total latency of each VNF, which are tested in related scenarios. The results of ICA test case (uRLLC) are recorded in Figure 6.21, the CPU resources for each VNF are in the range [5.5 – 5.9] GHz, the allocated bandwidth resources are in the range [4.0 – 4.6] Gbps; the VNF latencies are in the range of [6 – 7.5] ms; the total delay of the NSR is less than 55ms.

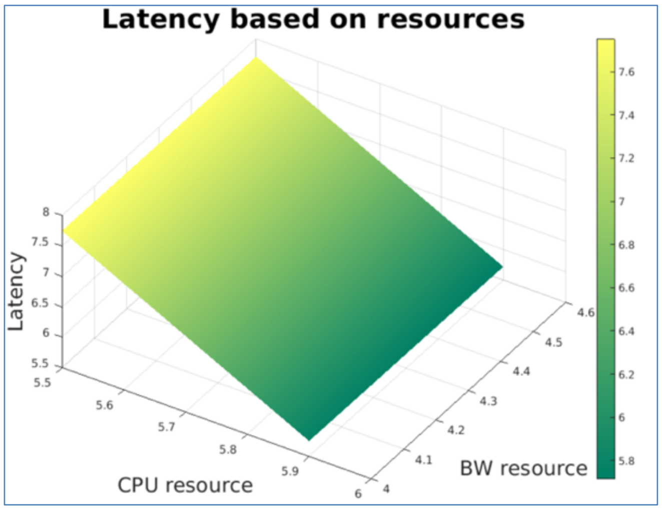


Figure 6.21: Latency of an IAC service (uRLLC)

The results of EN test cases (eMBB) are recorded in Figure 6.22; the CPU resources for each VNF are in the range [1.02 – 1.14] GHz, the allocated bandwidth resources are in the range [4.0 – 4.6]

Gbps; the VNF latencies are in the range of [30 – 32] ms; the total delay of the NSR is less than 185ms.

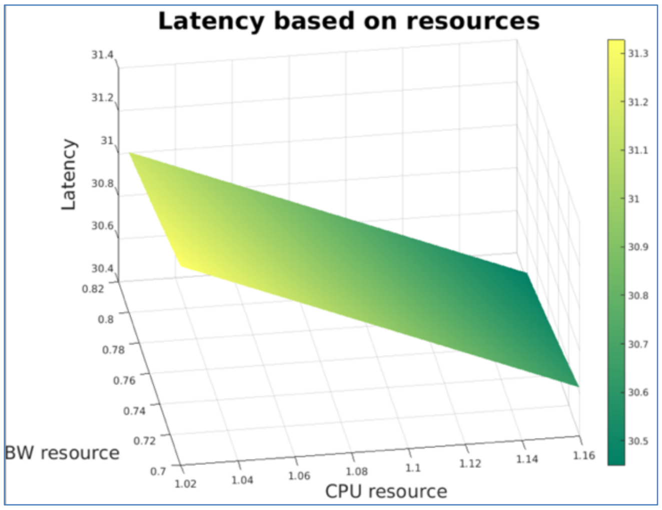


Figure 6.22: Latency of an EN service (eMBB)

6.4.2.3 Resource consumption

We compare the allocated resource with FRAM and SRAM of the State-Of-The-Art (Alleg et al. 2017), which are based on completely different 5G applications; FRAM is based on the linear proportion of the processing delay over allocated resources, it is modelled as a MIQCP, and it is solved using AIMMS 4.3; SRAM is the strict resource allocation proposal, it is modelled as a MIP and it allocates at the max demands.

In this experiment, the resource consumption is recorded as a percentile of the available resources. The eMBB NSR in SC topology is repeatedly tested into groups of 5, 10, 15 and 20 NSRs, the results are recorded and adjusted based on the number of VNFs in the NSR as our eMBB NSR has 6 VNFs while FRAM and SRAM have 4 VNFs. The test results, which are presented in Figure 6.23, showed that the our NSL resource allocation outperforms FRAM by 10% - 15% because the resources are allocated to the exact demand (our program has been linearised into a linear program), whilst FRAM is a MIQCP, and it allocated resource at or above the threshold; SRAM allocates resources at the upper bound demands. In the State-Of-The-Art results that are presented as Fig. 4

in their paper, FRAM outperformed SRAM by ~30%, so our test results outperformed SRAM by ~30%.

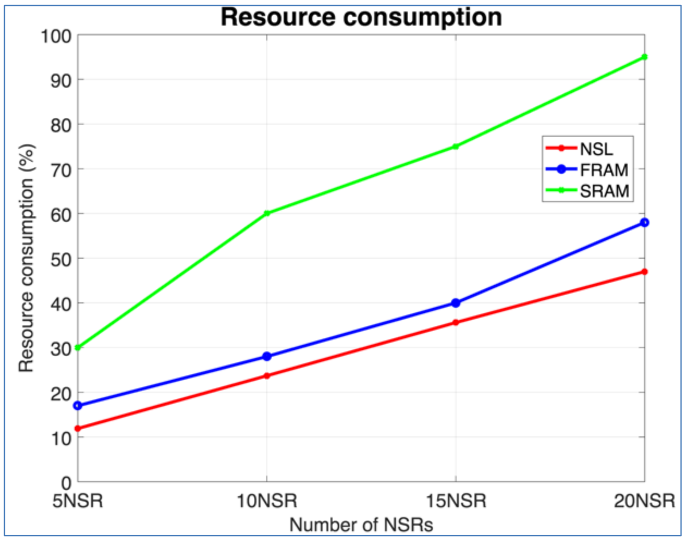


Figure 6.23: CPU resource consumption

6.4.2.4 Latency satisfaction

The latency satisfaction metric is evaluated based on the processing and transmission delays of the total delay of each VNF in the NSRs of ICA and EN applications. The results of VNFs in the EN application is presented in Figure 6.24. The results show that the total latency (max 32ms) is occupied by the processing latency (max 30ms).

The results of VNFs in the ICA application is presented in Figure 6.25. The results show that the total latency (max 7.5 ms) is occupied by the processing latency (max 7.3 ms).

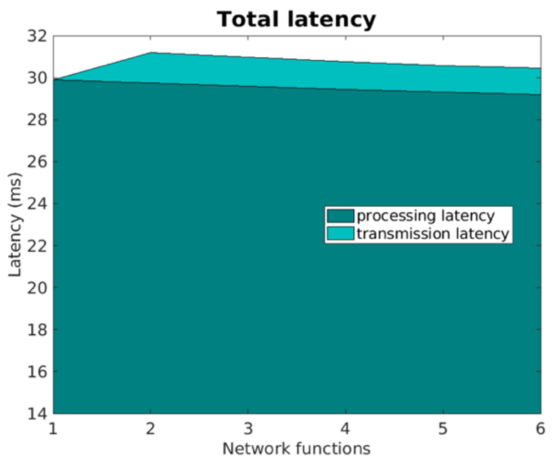


Figure 6.24: Latency of EN with processing and transmission

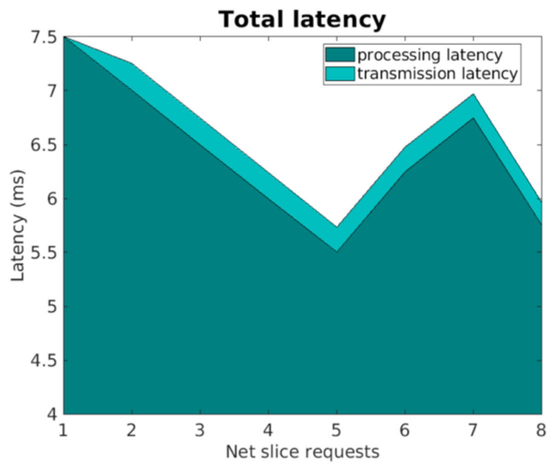


Figure 6.25: Latency of IAC with processing and transmission

6.4.2.5 Total latency by NSR topologies

In the EN application, the NSR with SC topology has the longest latency, the SM topology has the shorter latency as the branch comprises less VNFs, the FM topology may have the shortest latency as its branches are complex, so they comprise the least number of VNFs. The SC topology comprises all VNFs in the sequence, hence, its total latency is above 180ms. The LM topology comprises branches, hence, one or more VNFs are bypassed, its total latency is just below 160ms, the FM topologies comprises many different branches, the latency of the shortest branch is just 120ms, other branches latency is above 150ms. Figure 6.26 depicts the total NSR latencies based on topologies of SC, SM, FM of EN services.

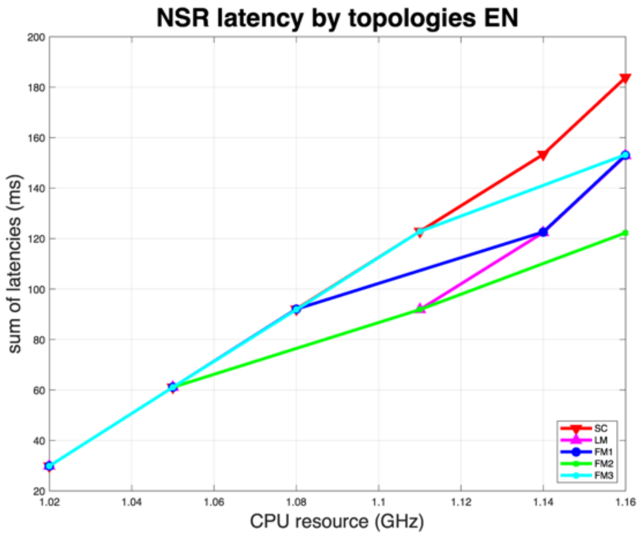


Figure 6.26: EN application, NSR latencies by topologies SC, LM, FM

In ICA application, the number of VNFs is more than the number in EN application, but its latency is much shorter, the SC and LM topologies are similar to the EN application, the FM topology comprises more branches than the EN application. The SC NSR latency is just under 55ms, the LM topology with a smaller number of VNFs compared to the SC has its latency at just above 45ms. The FM topology has different branches, each branch comprises different number of VNFs, their latencies are spread from under 35ms, to 40ms to under 45ms. Figure 6.27 depicts the total NSR latency based on topologies of SC, SM, FM of ICA services.

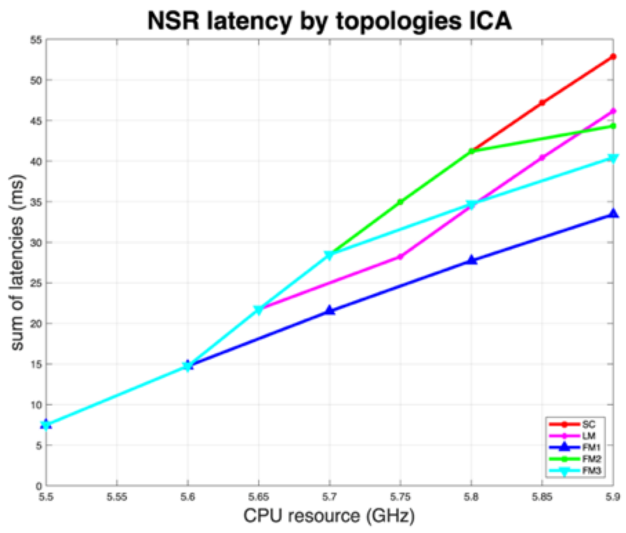


Figure 6.27: ICA application, NSR latencies by topologies SC, LM, FM

6.4.2.6 Execution time

The execution time of the network slicing algorithms is evaluated against FRAM and SRAM in the State-Of-The-Art (Alleg et al. 2017). The results are presented in Figure 6.28 that show NSL outperforms the FRAM and SRAM. The test results show that our execution time was only one tenth of FRAM, and one twentieth of SRAM. The main contributions are in the proposed solution strategy with linearization and decomposition of the MP into sub-processes that become smaller and simpler to process; and the leverage of flow control mechanism in IBM OPL script. In the State-Of-The-Art result, FRAM execution time outperforms SRAM 50% (Fig. 9), therefore, our result outperforms SRAM.

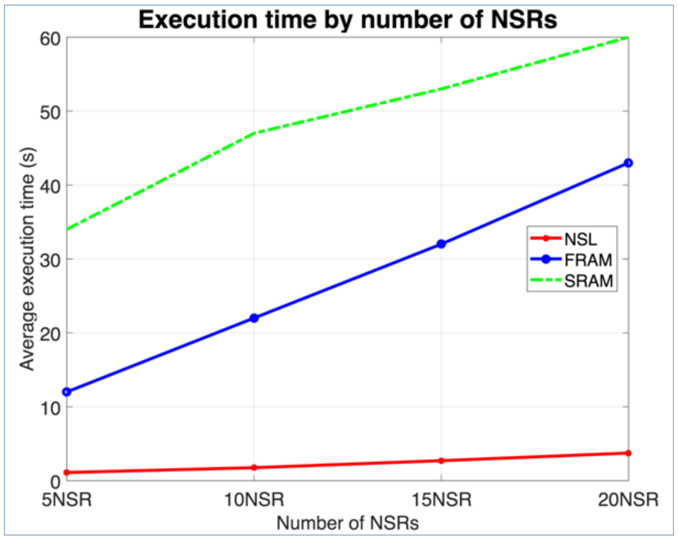


Figure 6.28: The execution time compared to FRAM, SRAM

6.4.2.7 Expected total delay by arrival rates

The expected total delay by arrival rates is calculated based on the expected time in the VNF as specified in formula (3). The higher the arrival rates, the longer the waiting time in the queue and the longer the total delay. The evaluation will be conducted with MaxZ and Optimum of the State-Of-The-Art (Agarwal et al. 2018); the expected delay is normalized against the delay for QoS. The results are presented in Figure 6.29. Our NSL (red line) based on formula (3) is different from MaxZ and Optimum (the blue and green line) as we based on a different queueing model $M_t/G/1$, while the State-Of-The-Art is based on M/M/1. The results in Figure 6.29 show that our model reflects the changes in the real-world in a simple and realistic way.

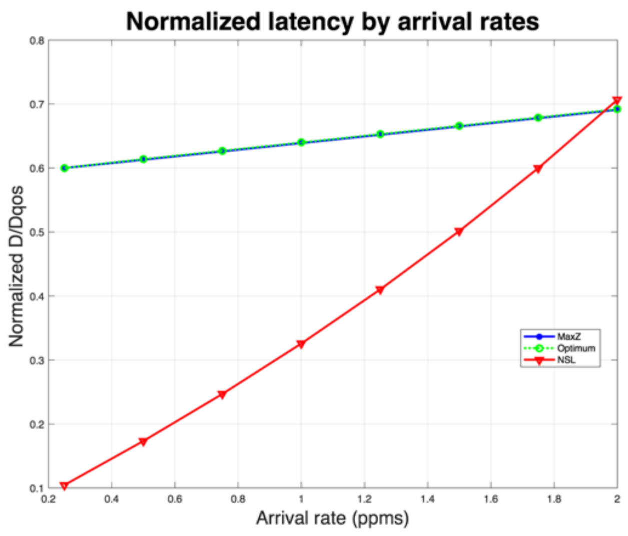


Figure 6.29: Normalized delay by arrival rates with the-state-of-the-arts (Agarwal et al. 2018)

6.5 Summary

In this chapter, the research objective is to investigate and to propose a novel mathematical model for the network slicing problem that focuses on multiple objectives of resource optimization and latency satisfaction of different 5G application types: eMBB, uRLLC and mMTC. The analysis model and resource model are built for the NS problem. The NS performance model is built using queueing theory, each VNF is modelled as an $M_t/G/1$ queue, in which the Markov-modulated Poisson process (MMPP) is applied to the VNF input queue. We assume there is a linear function between processing latency and CPU resource allocation (Alleg et al. 2017). The NS mathematical model is considered as a structure system; it is NP-Hard, non-linear, non-convex, and mixed integer, our approach is convex relaxation, linearization and decomposing the problem into the master model and two sub-models: VNF embedding algorithms as well as virtual link embedding, and solved using IBM OPL flow control and scripts. The evaluation results show that our NS solution outperforms the State-Of-The-Art in resource allocation, runtime, latency satisfaction and acceptance ratio.

CHAPTER 7 CONCLUSIONS AND FUTURE WORK

In this chapter, we conclude the thesis with the summary and contributions of all the research solutions to the existing body of knowledge. The novelties and significance of the thesis will be outlined. Lastly, the future works will be presented.

7.1 Summary and contributions of the thesis

The primary focus of the thesis is to investigate and to develop solutions for resource request provisioning, and resource optimization in network virtualization and 5G core network slicing applying SDN technology.

We identified the challenge in network resource request provisioning, which is the ad-hoc and self-explored way, does not add value for reuse, and leads to 'reinventing the wheel'. This thesis identifies the challenge in network virtualization that requires a new virtual network embedding mechanism that focuses concurrently on control congestion, saving cost, saving energy objectives in the production operation environments. The thesis also identifies the challenge in virtual link embedding that requires a new resource allocation mechanism that selects actively the physical link resources based on multiple objectives that the virtual links will be embedded on; and the challenge in network slicing that requires a new resource allocation optimization mechanism that satisfies the latency constraints of different 5G application types: eMBB, uRLLC, mMTC in 5G mobile system, and constraints related to NV and NFV.

Software-defined networking is the network paradigm that (i) separates the control plane from the data plane in network devices, (ii) centralizes control in the SDN controller, (iii) introduces programmability to networking, and (iv) open source the SBI and NBI interfaces to provide the opportunities for the public to contribute. Our research investigates and develops solutions for the identified challenges applying SDN technology.

The novelties of our research are summarized as follows.

The research introduces a new architecture for an extended SDN application that is based on MSA, MS design patterns of service registration, service discovery, service composition; three-tier architecture, and domain driven design. The architecture promotes modular designs and software reuse, allows new service creation and service composition, and allows developers to concentrate on development. CEVNE is a new VNE mathematical model that focuses concurrently on multi-objectives: cost saving, energy saving and congestion control for the production network operations. The CEVNE is modelled using the weighting method with positive weights and the constraint method with binding constraint to generate the pareto-optimal solutions for all objectives. SDN-based heuristic algorithm promotes a new design of the VNE virtual link embedding algorithm, it opens the new way to realize the virtual link embedding on the three-tier architecture for SDN applications that applies MSA and MS design patterns. A new mathematical model for the network slicing in 5G core networks that optimizes resource allocation and satisfies latency of different 5G application types: eMBB, uRLLC and mMTC. The new solution approaches to the network slicing program apply convex relaxation, linearization, and the decomposition techniques of the structure systems.

The contributions of the thesis are summarized as follows.

A novel architecture to realize the extended SDN applications on top of the SDN controllers:

It is based on Micro-service architecture (MSA). It is a three-tier architecture with the database tier, business tier and web tier. The database tier ensures that the extended application can have its own database. The business tier consists of micro-services (MS) applying MS design patterns. It should allow the creation of new services, the reuse of existing services, and the composition of a service based on new and existing services. It is based on the domain driven design (DDD) principle. The web tier ensures the application can be accessed via the browser. The architecture ensures SDN applications to be integrated into vertical applications, workflows or business processes based on the programmability of the SDN paradigm.

A novel mathematical model and its algorithms for the VNE that focuses on multiple objectives: saving cost, saving energy and avoiding congestion (CEVNE) in network virtualization in cloud data centres. The cost saving objective allocates minimal substrate resources to virtual networks; the energy saving minimizes the number of active nodes; the congestion avoidance minimizes the maximum link utilization. These objectives are combined into the CEVNE mathematical model using the weighting methods with positive weights and the constraint method with binding constraints. The cost saving and energy saving objectives are normalised as a ratio. The mathematical program (MP) is a mixed-integer linear program (MILP), which is relaxed into a linear program, and solved by GLPK. As solving VNE is NP-Hard, heuristic algorithms are designed for CEVNE with node embedding, link embedding and integration algorithms. CEVNE node embedding algorithm applies the rounding function of D-ViNE-LB to find the approximation of integer variables in the sub-optimal solutions.

A novel SDN-based heuristic algorithm for the CEVNE link embedding process is proposed on an SDN substrate network that focuses concurrently on three objectives: saving cost, saving energy and avoiding congestion. The heuristic solution is realized as a composite SDN application using the architecture for extended SDN application that integrates ONOS core services, SR application, the monitoring application and new MS that are designed for each objective. The application is based on the three-tier architecture, MSA, MS design patterns and DDD principles. The CEVNE LiM application invokes the path selection service to choose the substrate path that satisfies multiple objectives. The substrate network is an SDN leaf-spine fabric running SR on an SDN controller.

A novel solution approach is presented for the network slicing in 5G core networks as network slicing is a complicated concept in 5G. The analysis model is built to understand how network slicing operates. The resource model is used to allocate resources to VNFs and to virtual links connecting VNFs. The performance model based on queueing theory is used to predict network slicing end-to-end latency. The VNF is modelled as a M_1/G/1 queueing system. The Markov-modulated Poisson process (MMPP) pattern is applied to the VNF input queue, and the linear function of the VNF processing latency on allocated CPU resource is applied to control both resource and performance models so that it satisfies the latency threshold. The network slicing problem is modelled as a structure system with two objectives, resource allocation minimization and latency satisfaction of 5G applications in different types: eMBB, uRLLC and mMTC. The MP is non-convex, non-linear, mixed-integer and NP-Hard. The novelty of the solution also lies in the solution strategy, which focuses on convex relaxation using linearization, and decomposition techniques for structure systems. The MP is decomposed into a master model and two sub-models: VNF embedding and virtual link embedding, and they are solved using IBM OPL scripts and the flow control mechanism.

We prototyped the proposed architecture using the dynamic resource management (DRM) application on the ONOS controller using the GEANT network on Mininet. The prototype showed that the proposed architecture is appropriate, effective, realizable, and practical for emerging applications.

We evaluated CEVNE in both challenging, near-congestion scenarios and normal scenarios. The results show that in challenging scenarios, CEVNE has double to three times the number of successful cases than D-ViNE-LB. In normal scenarios, CEVNE improved runtime by 50%, energy consumption by 20%, its average active node stress is higher than D-ViNE-LB because of the congestion avoidance objective. The performance evaluation showed that CEVNE outperformed D-ViNE-LB in the three objectives that it focuses concurrently.

We evaluated SDN-based heuristic CEVNE LiM algorithm and realized the CEVNE LiM application. The performance of CEVNE LiM is compared with the k-shortest path algorithm and shows its superior performance in terms of the overall runtime, the average path length, the average node stress, the average link stress, and the overall energy consumption.

We evaluated our network slicing solutions and compared them with the State-Of-The-Art. The results show that our network slicing solution outperformed the State-Of-The-Art in resource allocation by 25% in efficiency, 100% acceptance ratio and 50% or more in runtime, and satisfied latencies of different 5G application types.

7.2 Future work

- Architecture of extended SDN applications

The architecture for extended SDN applications will be considered to apply to open-source SDN controllers such as OpenDayLight, Ryu. As open-source SDN controllers provide the NBI to build SDN applications, we will examine the controller core services, the controller platform, and the application of MS design patterns: service registration, service discovery and service composition in the controller.

The current proposed architecture of extended SDN applications needs to integrate with Artificial Intelligence (AI): Machine learning and Deep learning algorithms. These algorithms may not be based on MSA, so the integration may be via their interfaces.

P4 programmable and Stratum in SDN 2.0 are new trends that allow the integration of data plane program with Openflow using the Flow Objective in the NBI; currently it is integrated to ONOS controller using P4Runtime. The extended architecture allows the SDN applications and services to integrate with P4 and Stratum in the data plane via the core services and NBI.

The architecture can be used to build custom 5G core services and the vertical 5G applications. As 5G exploits the SDN data plane, the architecture for extended SDN application and services allow the building of 5G custom services using P4 and Stratum in the data plane together with other 5G core services.

- Congestion-aware, energy-aware virtual network embedding

The CEVNE will be tested in the production environment and implemented in a field trial. We also consider testing CEVNE in different fabric topologies for the clouds such as the clos topology, fat tree topology, DCell topology and the hierarchy of leaf-spine fabric.

CEVNE node mapping process is modelled using MP. Currently, the performance evaluation is tested using the Alevin framework. It needs to be tested in the production environment in cloud data centers to gain the real-time response. With AI and ML, the application of AI to optimization, and the application of AI into the structure of solvers, the modeling techniques and solution approaches in CEVNE can be applied to build a good ML model. The multi-objective VNE can be extended to apply to other combinations of multi-objectives VNE such as QoS, multi-path embedding, and security, to name a few.

- Realization of virtual link mapping in CEVNE

We will add different objectives such as security, reliability to the virtual link embedding algorithm, each objective has its own algorithms that will be implemented as MS. We will apply CEVNE LiM in a hierarchical leaf-spine fabric with many layers. We will try different formulation and algorithms for each objective: cost saving, energy saving and congestion avoidance.

The realization of CEVNE link mapping process is tested in the Mininet environment. It needs to be tested in the production environment in cloud data centers to gain real-time feedback, especially the monitoring component in the proposed architecture. The SDN-based heuristic approach can be extended to focus on different objectives for the virtual link mapping process such as QoS, with different monitoring applications.

- Latency-aware resource allocation optimization in network slicing in 5G core network

In this research, the parameters $a^{f(i)}$ and $b^{f(i)}$ are set at the same values for all VNFs; this is intuitive as VNFs are considered having similar upper bounds and lower bounds of the processing latencies. If a VNF requires different values of the two parameters $a^{f(i)}$ and $b^{f(i)}$ from other VNFs, constraints (31), (37) and (45) will provide each parameter of each VNF in the NSR.

The network slice proposal can be realized using the architecture solution in the 5G core system such as the Open Baton project (Fraunhofer 2019), or COMAC (Sunay 2019). Open Baton is an extensible and customizable NFV MANO-compliant framework (Fraunhofer 2019), which provides the module “network slice engine” to create network slices and to integrate with Open Stack. COMAC is an exemplar platform that delivers next-generation services over both mobile and broadband networks. COMAC will create diverse mobile network slices, combining access and core technologies to support use cases and services (Sunay 2019).

The structure system and the decomposition solution approach can be extended to network slicing for vertical 5G applications such as in the V2X application, fast train, factories 4.0. As the vertical applications are using 5G services so they will be utilizing network slicing in 5G. That area will show that our network slicing research is beneficial for future 5G vertical applications.

7.3 Conclusion remarks

We conclude that our aims “to investigate and to develop solutions for resource request provisioning, resource optimization in network virtualization and 5G core network slicing applying SDN technology” have been achieved. In our thesis, we address each research question carefully, we investigate and carry out the objectives and their tasks thoroughly in the researching for solutions. To summarize, we highlight the significance of our research as follows.

We have performed a comprehensive study of the resource optimization problem and investigate the architecture for extended SDN applications. The outcomes demonstrate the novelties, the efficiency, and the applicability of our solutions. They demonstrate the innovation and effectiveness of our solution approaches in solving optimization problems. The results of this thesis can readily be applied to 5G vertical applications where resource optimization and network routing problems exist naturally in multiple domains and require software-defined networking logically centralized control architecture for efficient and dynamic solutions.

BIBLIOGRAPHY

5GEx 2020, *5G Exchange*, H2020, viewed 24 April 2020, <https://cordis.europa.eu/project/id/671636>.

ABDULLAH, Z.N., AHMAD, I. & HUSSAIN, I. 2018, 'Segment Routing in Software Defined Networks: A Survey', *IEEE Communications Surveys & Tutorials.*, vol. 21, no. 1, p. 23.

ADDAD, R., BAGAA, M., TALEB, T., DUTRA, D. & FLINCK, H. 2019, 'Optimization model for Cross-Domain Network Slices in 5G Networks', *IEEE Transactions on Mobile Computing*.

AFOLABI, I., TALEB, T., SAMDANIS, K., KSENTINI, A. & FLINCK, H. 2018, 'Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions', *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, p. 25.

AGARWAL, S., MALANDRINO, F., CHIASSERINI, C. & DE, S. 2018, 'Joint VNF placement and CPU allocation in 5G', paper presented to the *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*.

AIMMS 2020, *Developers Home*, viewed 16 April 2020, <https://www.aimms.com/english/developers/>.

ALIZADEH, M., EDSALL, T., DHARMAPURIKAR, S., VAIDYANATHAN, R., CHU, K., FINGERHUT, A., LAM, V., MATUS, F., PAN, R., YADAV, N. & VARGHESE, G. 2014, 'CONGA: Distributed Congestion-Aware Load Balancing for Datacenters', *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, p. 12.

ALLEG, A., AHMED, T., MOSBAH, M., RIGGIO, R. & BOUTABA, R. 2017, 'Delay-aware VNF placement and chaining based on a flexible resource allocation approach', paper presented to the *2017 13th International Conference on Network and Service Management (CNSM)*.

AMALDI, E., CONIGLIO, S., KOSTER, A.M. & TIEVES, M. 2016, 'On the computational complexity of the virtual network embedding problem', *Electronic Notes in Discrete Mathematics*, vol. 52, p. 8.

ARKALGUD, N. & RUX, S. 2018, *OPL with Flow Control Examples*, IBM, <https://developer.ibm.com/docloud/blog/videos/opl-flow-control-examples/>.

BASS, L. 2013, *Software architecture in practice*, third edition edn, Addison-Wesley.

BAYS, L.R., GASPARY, L.P., AHMED, R. & BOUTABA, R. 2016, 'Virtual Network Embedding in Software-Defined Networks', paper presented to the *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*.

BECK, M.T., LINNHOFF-POPIEN, C., FISCHER, A., KOKOT, F. & DE MEER, H. 2014, 'A simulation framework for virtual network embedding algorithms', paper presented to the *2014 16th international telecommunications network strategy and planning symposium (Networks)*

BERNARDO, D. 2012, 'Network security mechanisms and implementations for the next generation reliable fast transfer protocol', UTS, UTS Sydney, Australia.

BLENK, A. & KELLERER, W. 2019, 'Towards Virtualization of Software-Defined Networks: A Journey in Three Acts', paper presented to the *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*.

BLIEK1Ú, C., BONAMI, P. & LODI, A. Oct. 2014, 'Solving Mixed-Integer quadratic programming problems with IBM-CPLEX: a progress report', paper presented to the *Proceedings of the twenty-sixth RAMP symposium*, Hosei University, Tokyo.

BOLCH, G., GREINER, S., DE MEER, H. & TRIVEDI, K.S. 2006, *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*, John Wiley & Sons.

BOSSHART, P., DALY, D., GIBB, G., IZZARD, M., MCKEOWN, N., REXFORD, J., SCHLESINGER, C., TALAYCO, D., VAHDAT, A., VARGHESE, G. & WALKER, D. 2014, 'P4: Programming protocol-independent packet processors', *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, p. 9.

Broadcom 2019, *Openflow Data Plane Abstraction: Abstract switch specification*, viewed 26 September 2017, <http://www.broadcom.com>.

CABALLERO, P., BANCHS, A., DE VECIANA, G., COSTA-PÉREZ, X. & AZCORRA, A. 2018, 'Network slicing for guaranteed rate services: Admission control and resource allocation games', *IEEE Transactions on Wireless Communications*, vol. 17, no. 10, p. 14.

CHANDRAMOULI, D., LIEBHART, R. & PIRSKANEN, J. 2019, *5G for the connected world*, Wiley.

CHAYAPATHI, R., HASSAN, S.F. & SHAH, P. 2016, *Network Functions Virtualization (NFV) with a Touch of SDN*, 1st edn, Addison-Wesley Professional.

CHENG, X., SU, S., ZHANG, Z., WANG, H., YANG, F., LUO, Y. & WANG, J. 2011, 'Virtual network embedding through topology-aware node ranking', *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, p. 10.

CHOWDHURY, M., RAHMAN, M.R. & BOUTABA, R. 2012, 'Vineyard: Virtual network embedding algorithms with coordinated node and link mapping', *IEEE/ACM Transactions on Networking (TON)*, vol. 20, no. 1, p. 14.

CHOWDHURY, S.R., AHMED, R., SHAHRIAR, N., KHAN, A., BOUTABA, R., MITRA, J. & LIU, L. 2017, 'Revine: Reallocation of virtual network embedding to eliminate substrate bottlenecks', paper presented to the *IEEE/IFIP Integrated Network Management Symposium (IM)*.

CLAISE, B., CLARKE, J. & LINDBLAD, J. 2019, *Network programmability with YANG : the structure of network automation with YANG, NETCONF, RESTCONF, and gNMI*, Addison-Wesley, Boston.

COELLO, C.A.C., LAMONT, G.B. & VAN VELDHUIZEN, D.A. 2007, *Evolutionary algorithms for solving multi-objective problems*, vol. 5, Springer, New York.

COHON, J.L. 1978, *MultiObjective Programming and planning*, vol. 140, Courier Corporation.

DAHIR, M., ALIZADEH, H. & GÖZÜPEK, D. 2019, 'Energy efficient virtual network embedding for federated software-defined networks', *International Journal of Communication Systems*, vol. 32, no. 6, p. 17.

DANTZIG, G.B. 1998, *Linear programming and extensions*, vol. 48, Princeton University Press.

DAS, S. 2015, *Segment routing architecture in ONOS SDN controller* ONOS Wiki, viewed 15/09 2017, <https://wiki.onosproject.org/display/ONOS/Software+Architecture>.

DAS, S. 2019, *Leaf-spine fabric configuration*, ONOS, <https://github.com/opennetworkinglab/routing>.

DAS, S. & PETERSON, L. 2019, *Trellis: CORD network infrastructure*, ONF, viewed July 5th 2017, <https://wiki.opencord.org/display/CORD/Trellis%3A+CORD+Network+Infrastructure>

DIETRICH, D., PAPAGIANNI, C., PAPADIMITRIOU, P. & BARAS, J.S. 2017, 'Network function placement on virtualized cellular cores', paper presented to the *IEEE COMSNETS 2017*, Bangalore, India.

DOHLER, M. 2019, *Introduction to 5G*, 5g-courses.com, King College, London.

DUAN, Q. & TOY, M. 2017, *Virtualized Software-defined networks and services*, Artech House.

ELIAS, J., MARTIGNON, F., PARIS, S. & WANG, J. 2014, 'Optimization models for congestion mitigation in virtual networks', paper presented to the *IEEE 22nd International Conference on Network Protocols*.

ELIAS, J., MARTIGNON, F., PARIS, S. & WANG, J. 2017, 'Efficient orchestration mechanisms for congestion mitigation in NFV: Models and algorithms', *IEEE Transactions on Services Computing*, vol. 10, no. 4, p. 13.

EPPSTEIN, D. 1998, 'Finding the k shortest paths', *SIAM Journal on computing*, vol. 28, no. 2, p. 22.

ERICKSON, D. 2013, 'The beacon openflow controller', *The second ACM SIGCOMM workshop on Hot topics in software defined networking*, ACM, p. 6.

FENDT, A., SCHMELZ, L.C., WAJDA, W., LOHMÜLLER, S. & BAUER, B. 2018, 'A Network Slice Resource Allocation Process in 5G Mobile Networks', paper presented to the *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*.

FILSFILS, C., NAINAR, N.K., PIGNATARO, C., CARDONA, J.C. & FRANCOIS, P. 2015, 'The segment routing architecture', paper presented to the *2015 IEEE Global Communications Conference (GLOBECOM)*.

FISCHER, A. 2016, 'An evaluation methodology for virtual network embedding', PhD Thesis thesis, University Passau, Germany.

FISCHER, A., BECK, M.T. & DE MEER, H. 2013, 'An approach to energy-efficient virtual network embeddings', *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, ed. IEEE, p. 6.

FISCHER, A., BOTERO, J.F., BECK, M.T., DE MEER, H. & HESSELBACH, X. 2013, 'Virtual Network Embedding: A Survey', *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, p. 19.

FISCHER, W. & MEIER-HELLSTERN, K. 1993, 'The Markov-modulated Poisson process (MMPP) cookbook', *Performance evaluation*, vol. 18, no. 2, p. 53.

FORTZ, B. & THORUP, M. 2000, 'Internet Traffic Engineering by optimizing OSPF weights', *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, IEEE, p. 10.

FOUKAS, X., PATOUNAS, G., ELMOKASHFI, A. & MARINA, M.K. 2017, 'Network slicing in 5G: Survey and challenges', *IEEE Communications Magazine*, vol. 55, no. 5, p. 7.

FOWLER, M. 2014, *Microservices*, 2019, <http://martinfowler.com/articles/microservices.html>.

Fraunhofer 2019, *Open Baton*, viewed 16 April 2020, <http://openbaton.github.io>.

GNS3 2020, *The software that empowers network professionals*, viewed 17 April 2020, <https://www.gns3.com>.

GORANSSON, P., BLACK, C. & CULVER, T. 2016, *Software-defined networks: A comprehensive approach*, Morgan Kaufmann.

GOUAREB, R., FRIDERIKOS, V. & AGHVAMI, A.H. 2018, 'Virtual network functions routing and placement for edge cloud latency minimization', *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, p. 12.

GUPTA, V., HARCHOL-BALTER, M., WOLF, A.S. & YECHIALI, U. June 2006, 'Fundamental characteristics of queues with fluctuating load', *ACM Sigmetrics Performance Evaluation Review*, vol. 34, no. 1, p. 13.

HAGHANI, M., BAKHSHI, B. & CAPONE, A. 2018, 'Multi-objective embedding of software-defined virtual networks', *Computer communications*, vol. 129, no. 2018, p. 11.

HANTOUTI, H., BENAMAR, N., TALEB, T. & LAGHRISSI, A. 2018, 'Traffic Steering for Service Function Chaining', *IEEE Communications Surveys & Tutorials.*

HARCHOL-BALTER, M. 2013, *Performance modeling and design of computer systems: queueing theory in action*, Cambridge University Press.

HARIGANESH, S. 2018, *Problems on Pollaczek-Khinchine formula*, viewed 16 April 2020, <https://www.youtube.com/watch?v=HkappM7G0kw>.

HE, K., ROZNER, E., FELTER, W., CARTER, J., AGARWAL, K. & AKELLA, A. 2015, 'Presto: Edge-based Load Balancing for Fast Datacenter Networks', *ACM Sigcomm Computer Communication Review*, vol. 45, no. 4, p. 14.

HE, M., ALBA, A.M., BASTA, A., BLENK, A. & KELLERER, W. 2019, 'Flexibility in softwarized networks: Classifications and research challenges', *IEEE Communications Surveys & Tutorials*.

HESSELBACH, X., AMAZONAS, J.R., VILLANUEVA, S. & BOTERO, J.F. 2016, 'Coordinated node and link mapping VNE using a new paths algebra strategy', *Journal of Network and Computer Applications*, vol. 69, p. 13.

HIJAZI, H. 2010, 'Mixed-integer nonlinear optimization approaches for network design in telecommunications', PhD thesis, Université d'Aix Marseille, France.

HIRA, M. & WOBKER, L. 2015, *Improving Network Monitoring and Management with Programmable Data Planes*, P4 Organization, viewed July 5th 2017, <http://p4.org/p4/inband-network-telemetry/>.

HOANG, D.B. 2015, 'Software Defined Networking – Shaping up for the next disruptive step?', *Australian Journal of Telecommunications and the Digital Economy*, vol. 3, no. 4, p. 15.

HOANG, D.B. & PHAM, M. 2015, 'On software-defined networking and the design of SDN controllers', paper presented to the *2015 6th International conference on the Network of the Future (NOF)* Montreal, Canada.

HOUIDI, I., LOUATI, W. & ZEGHLACHE, D. 2015, 'Exact multi-objective virtual network embedding in cloud environments', *The Computer Journal*, vol. 58, no. 3, p. 13.

ITU-R_M.2083 2015, *IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond 09/2015*, Technical Report, ITU-R M.2083-0 09/2015.

JARRAYA, Y., MADI, T. & DEBBABI, M. 2014, ' A survey and a layered taxonomy of software-defined networking', *IEEE communications surveys & tutorials*, vol. 16, no. 4, p. 26.

JENNINGS, B. & STADLER, R. 2014, 'Resource management in clouds: survey and research challenges', *Network System Management*, vol. 2015, no. 23, p. 54.

KAMMOUN, A., TABBANE, N., DIAZ, G., DANDOUSH, A. & ACHIR, N. May 2018, 'End-to-End Efficient Heuristic Algorithm for 5G Network Slicing', paper presented to the *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, Pedagogical University of Cracow, Poland.

KATTA, N., GHAG, A., HIRA, M., KESLASSY, I., BERGMAN, A., KIM, C. & REXFORD, J. 2017, 'Clove: Congestion-Aware Load Balancing at the Virtual Edge', paper presented to the *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, Seoul, South Korea.

KATTA, N., HIRA, M., GHAG, A., KIM, C., KESLASSY, I. & REXFORD, J. 2016, 'CLOVE: How I Learned to Stop Worrying About the Core and Love the Edge', paper presented to the *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*.

KATTA, N., HIRA, M., KIM, C., SIVARAMAN, R. & REXFORD, J. 2016, 'HULA: Scalable Load Balancing Using Programmable Data Planes', paper presented to the *Proceedings of the Symposium on SDN Research*, Sanat Clara, CA, USA.

KHAN, M.M.A., SHAHRIAR, N., AHMED, R. & BOUTABA, R. 2016, 'Multi-path link embedding for survivability in virtual networks', *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, p. 14.

KIM, C., BHIDE, P., DOE, E., HOLBROOK, H., GHANWANI, A., DALY, D., HIRA, M. & DAVIE, B. 2016, *Inband network telemetry (INT) specification*, P4 Organization, viewed July 5th 2017, <http://p4.org/wp-content/uploads/fixed/INT/INT-current-spec.pdf>.

KNIGHT, S., NGUYEN, H.X., FALKNER, N., BOWDEN, R. & ROUGHAN, M. 2011, 'The internet topology zoo', *EEE Journal on Selected Areas in Communications*, vol. 29, no. 9, p. 11.

KOCHER, P.S. 2018, *Microservices and containers*, Addison-Wesley.

KREUTZ, D., RAMOS, F.M., VERISSIMO, P., ROTHENBERG, C.E., AZODOLMOLKY, S. & UHLIG, S. 2015, 'Software-defined networking: A comprehensive survey', *Proceedings of the IEEE*, vol. 103, no. 1, p. 63.

LANTZ, B. 2014, *Mininet*, <https://github.com/mininet/mininet/wiki/Documentation>.

LASDON, L.S. 1970, *Optimization theory for large systems*, 1 edn, The Macmilan Company, New York.

LEBRUN, D., PREVIDI, S., FILSFILS, C. & BONAVENTURE, O. 2018, *Design and implementation of IPv6 Segment Routing*, <http://www.segment-routing.org>.

LECONTE, M., PASCHOS, G.S., MERTIKOPOULOS, P. & KOZAT, U.C. 2018, 'A resource allocation framework for network slicing', paper presented to the *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, Honolulu, USA.

LEE, M.C. & SHEU, J.P. 2016, 'An efficient routing algorithm based on segment routing in software-defined networking', *Computer Networks*, no. 103, p. 12.

LEWIS, J. 2012, *Micro services - Java the UNIX way*, <http://2012.33degree.org/pdf/JamesLewisMicroServices.pdf>.

LI, J., KOSHIBE, A. & PRETE, L. 2019, *ONOS monitoring applications*, ONOS, viewed 12 October 2019, <https://wiki.onosproject.org/display/ONOS/Monitoring>.

LI, X., LUNG, C. & MAJUMDAR, S. 2015, 'Energy Aware Green Spine Switch Management for Spine-Leaf Datacenter Networks', paper presented to the *2015 IEEE International Conference on Communications (ICC)*.

LI, X., LUNG, C. & MAJUMDAR, S. 2016, 'Green spine switch management for datacenter networks', *Jounal of Cloud Computing: Advances, Systems and Applications* vol. 5, no. 1, p. 9.

LI, Z., LU, Z., DENG, S. & GAO, X. 2018, 'A Self-Adaptive Virtual Network Embedding Algorithm Based on Software-Defined Networks', *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, p. 12.

MALANDRINO, F., CHIASSERINI, C.F., EINZIGER, G. & SCALOSUB, G. Dec. 2019, 'Reducing Service Deployment Cost Through VNF Sharing', *IEEE/ACM Transactions on Networking.*, vol. 27, no. 6, p. 14.

MAROTTA, A., ZOLA, E., D'ANDREAGIOVANNI, F. & KASSLER, A. 2017, 'A Fast Robust Optimization-based Heuristic for the Deployment of Green Virtual Network Functions', *Journal of Network and Computer Applications*, vol. 95, p. 12.

MARSCH, P., BULAKCI, O., QUESETH, O. & BOLDI, M. 2018, *5G system design: architectural and functional considerations and long term research*, 1st edn, Wiley Telecom.

MAYER, G. 2018, 'RESTful APIs for the 5G service based architecture', *Journal of ICT Standardization*, vol. 6, no. 1, p. 16.

MEDHI, D. 2017, *Network routing: algorithms, protocols, and architectures (Ed 2)*, 2nd edn, Morgan Kaufmann.

MIJUMBI, R., SERRAT, J., RUBIO-LOYOLA, J., BOUTEN, N., DE TURCK, F. & LATRE, S. 2014, 'Dynamic resource management in SDN-based virtualized networks', *10th international conference on network and service management (CNSM) and workshop*, p. 6.

MILLETT, S. 2015, *Patterns, principles, and practices of domain-driven design*, Wrox.

MIMIDIS-KENTIS, A., SOLER, J., VEITCH, P., BROADBENT, A., MOBILIO, M., RIGANELLI, O., ROSSEM, S., TAVERNIER, W. & SAYADI, B. 2019, 'The Next Generation Platform as A Service: Composition and Deployment of Platforms and Services', *Future Internet*, vol. 11, no. 5, p. 119.

MOORTHY, A. 2016, *Connecting the world: a look inside facebook's networking infrastructure*, 2018, <http://cs.unc.edu/xcms/wpfiles/50th-symp/Moorthy.pdf>.

NECOS 2019, *Novel Enablers for Cloud Slicing*, viewed 16 April 2020, <http://www.h2020-necos.eu/>.

NELSON, R. 2013, *Probability, stochastic processes, and queueing theory: the mathematics of computer performance modeling*, Springer Science & Business Media.

NFV-MAN 2014, *Network function virtualization: management and orchestration*, ETSI.

NG, T.M.J. & HOANG, D. 1987, 'Joint optimization of capacity and flow assignment in a packet-switched communications network', *IEEE Transactions on Communications*, vol. 35, no. 2, p. 8.

ODL 2019, *OpenDayLight Controller Use cases*, 2019, <https://www.opendaylight.org/use-cases/>.

OKI, E. 2012, *Linear programming and algorithms for communication networks: a practical guide to network design, control, and management*, CRC Press.

ONF 2014, *Openflow Switch specification 1.3.4*, Version 1.3.4, ONF.

ONF 2019a, *Advanced ONOS+P4 tutorial Building an SRv6-enabled fabric with P4 and ONOS*, ONF, viewed 15 June 2019.

ONF 2019b, *Atrium Project*, viewed 10 February 2019, <https://www.opennetworking.org/projects/atrium/>.

ONOS 2019a, *The Intent framework*, viewed 13 October 2017, <https://wiki.onosproject.org/display/ONOS/Intent+Framework>.

ONOS 2019b, *ONOS wiki*, ONF, viewed 6 Nov 2019, <https://wiki.onosproject.org/>.

ONOS 2019c, *Segment Routing*, ONOS, viewed 6 October 2019, <https://github.com/opennetworkinglab/onos/tree/master/apps/segmentrouting>.

openvswitch.org 2016, *Production Quality, Multilayer Open vVrtual Switch*, viewed Jan 30 2018, <http://openvswitch.org>.

OSGi_Alliance 2016, *OSGi Specification*, viewed 14 October 2019, <https://osgi.org/download/r6/osgi.core-6.0.0.pdf>.

ÖZBEK, B., AYDOĞMUŞ, Y., ULAŞ, A., GORKEMLI, B. & ULUSOY, K. 2016, 'Energy aware routing and traffic management for software defined networks', paper presented to the *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, Seoul, Korea.

PALOMAR, D.P. & CHIANG, M. 2006, 'A Tutorial on Decomposition Methods for Network Utility Maximization', *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, p. 13.

PASCHOS, G.S., ABDULLAH, M.A. & VASSILARAS, S. 2018, 'Network Slicing with Splittable Flows is Hard', paper presented to the *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*.

PAUL, S., JAIN, R., IYER, J. & ORAN, D. 2014, 'Mobie applications on global clouds using openflow and software-defined networking', *Network innovation through OpenFlow and SDN: principles and design*, CRC Press.

PEI, X., MARTINY, K., OBANA, K., GAMELAS, A. & LEE, D.K. 2017, *Network Functions Virtualisation (NFV)*, White Paper, vol. 2019, ETSI.

PENTTINEN, J.T. 2019, *5G Explained: Security and Deployment of Advanced Mobile Communications*, Wiley.

PEREZ, X. 2017, *5G network slicing for verticals*, viewed March 11 2019, <http://www.5gsummit.org/greece/slides/Session1_02_Xavier%20Costa%20Perez.pdf>.

PETERSON, L. 2019, *Open CORD wiki*, viewed 20 Apr 2017, <https://wiki.opencord.org>.

PHAM, M. & HOANG, D.B. 2016, 'SDN applications-The intent-based Northbound Interface realisation for extended applications', paper presented to the *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, Seoul, Korea.

PRADOS-GARZON, J., LAGHRISSI, A., BAGAA, M., TALEB, T. & LOPEZ-SOLER, J. 2019, 'A complete lte mathematical framework for the network slice planning of the epc', *IEEE Transactions on Mobile Computing*, vol. 19, no. 1, p. 14.

RARDIN, R. 2017, *Optimization in operations research*, vol. 166, Pearson Higher Education, Inc., Hoboken, NJ.

REVIRIEGO, P., SIVARAMAN, V., ZHAO, Z., MAESTRO, J., VISHWANATH, A., SÁNCHEZ-MACIAN, A. & RUSSELL, C. 2012, 'An Energy Consumption Model for Energy Efficient Ethernet Switches', paper presented to the *2012 International Conference on High Performance Computing & Simulation (HPCS)*.

RICHARDSON, C. 2018, *Microservices Patterns*, Manning Publications.

RICHARDSON, C. 2019, *Microservice Architecture Patterns*, 2019, <http://microservices.io/patterns/microservices.html>.

RODRIGUEZ, E., ALKMIM, G.P., DA FONSECA, N.L. & BATISTA, D.M. 2015, 'Energy-Aware Mapping and Live Migration of Virtual Networks', *IEEE Systems Journal*, vol. 11, no. 2, p. 12.

ROST, M. & SCHMID, S. 2018, 'Charting the complexity landscape of virtual network embeddings', paper presented to the *2018 IFIP Networking Conference (IFIP Networking) and Workshops*.

ROUT, S., PATRA, S.S. & SAHOO, B. 2018, 'Saving energy and improving performance in SDN using rate adaptation technique', *International Journal of Engineering & Technology*, vol. 7, no. 2.6, p. 6.

RUBIN, P. 2010, 'Binary variables and quadratic terms', <https://orinanobworld.blogspot.com/2010/10/binary-variables-and-quadratic-terms.html>.

SCHMITT, P., LANDAIS, B. & YONGYANG, F. 2018, *Control and User Plane Separation of EPC nodes (CUPS)*, 2019, <https://www.3gpp.org/cups>.

SCIANCALEPORE, V., SAMDANIS, K., COSTA-PEREZ, X., BEGA, D., GRAMAGLIA, M. & BANCHS, A. 2017, 'Mobile traffic forecasting for maximizing 5G network slicing resource utilization', paper presented to the *INFOCOM 2017-IEEE Conference on Computer Communications*.

SUNAY, O. 2017, *M-CORD platform and Roadmap*, Linux Foundation, viewed 27 Februaury 2018, <youtube M-CORD Platform and Roadmap >.

SUNAY, O. 2019, *COMAC: converged Multi Access and Core*, ONF, viewed 16 April 2020, <https://www.youtube.com/watch?v=qHRCpXgKg0w>.

TAHMASBI, M. 2017, *Lab 3 Monitoring and Debugging*, P4.org, viewed 18 Jan 2018, <https://www.youtube.com/watch?v=1lHxU1EpeF0&feature=youtu.be>.

TOOTOONCHIAN, A., GORBUNOV, S., GANJALI, Y., CASADO, M. & SHERWOOD, R. 2012, 'On Controller Performance in Software-Defined Networks', paper presented to the *2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services.*

TR_22.891 Sep 2016, *Feasibility Study on New Services and Markets Technology Enablers, Stage 1 (Rel. 14)*, Technical report, TR 22.891 Sep 2016 (Rel-14).

TR_28.801 Jan. 2018, *Study on management and orchestration of network slicing for next generation network (Rel-15)*.

TR-521 2016, *SDN Architecture v1.1*, Technical report, ONF TR-521.

TR-526 2016, *Applying SDN Architecture to 5G Slicing*, ONF.

TS_23.501 Jun. 2019, *System architecture for 5G systems; Stage 2 (Rel-16)*, Technical report.

TS_23.502 2019, *Procedures in 5G systems*, Technical Report, TS 23.502 V16.1.1 (2019-06).

TSAI, P., TSAI, C., HSU, C. & YANG, C. 2018, 'Network monitoring in software-defined networking: a review', *IEEE Systems*, vol. 12, no. 4, p. 12.

University_of_Adelaide 2012, *The internet topology zoo*, <http://www.topology-zoo.org>.

VAHDAT, A. 2015, *Pulling Back the curtain on Google's Network Infrastructure*, viewed Jan 30 2018, <http://research.googleblog.com/2015/08/pulling-back-curtain-on-googles-network.html>.

VAN ROSSEM, S., SAYADI, B., ROULLET, L., MIMIDIS, A., PAOLINO, M., VEITCH, P., BERDE, B., LABRADOR, I., RAMOS, A., TAVERNIER, W. & OLLORA, E. 2018, 'A vision for the next generation platform-as-a-service', paper presented to the *2018 IEEE 5G World Forum (5GWF)*.

VASSILARAS, S., GKATZIKIS, L., LIAKOPOULOS, N., STIAKOGIANNAKIS, I.N., QI, M., SHI, L., LIU, L., DEBBAH, M. & PASCHOS, G.S. 2017, 'The algorithmic aspects of network slicing', *IEEE Communications Magazine*, vol. 55, no. 8, p. 8.

WANG, Q., ALCARAZ-CALERO, J., RICART-SANCHEZ, R., WEISS, M.B., GAVRAS, A., NIKAEIN, N., VASILAKOS, X., GIACOMO, B., PIETRO, G., RODDY, M. & HEALY, M. June 2019, 'Enable advanced QoS-aware network slicing in 5G networks for slice-based media use cases', *IEEE Transactions on Broadcasting*, vol. 65, no. 2, p. 10.

WILLIAMS, H.P. 2013, *Model building in mathematical programming*, Fifth edn, John Wiley & Sons., England.

WOLFF, E. 2017, *Microservices Flexible software archiecture*, Peason Education.

Y.3011 Jan 2012, *Framework of network virtualization for future networks*, Technical report.

YE, Q., ZHUANG, W., LI, X. & RAO, J. 2018, 'End-to-End Delay Modeling for Embedded VNF Chains in 5G Core Networks', *IEEE Internet of Things Journal*, vol. 6, no. 1, p. 13.

YU, M., YI, Y., REXFORD, J. & CHIANG, M. 2008, 'Rethinking virtual network embedding: substrate support for path splitting and migration', *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, p. 13.

ZAHAVI, E., SHPINER, A., ROTTENSTREICH, O., KOLODNY, A. & KESLASSY, I. 2016, 'Links as a Service (LaaS): Guaranteed tenant isolation in the shared cloud', *Proceedings of the 2016 Symposium on Architectures for Networking and Communications Systems*, ACM, p. 12.

ZHANG, Z., SU, S., LIN, Y., CHENG, X., SHUANG, K. & XU, P. 2015, 'Adaptive multi-objective artificial immune system based virtual network embedding', *Journal of Network and Computer Applications*, vol. 53, p. 16.

ZHU, Y. & AMMAR, M.H. 2006, 'Algorithms for Assigning Substrate Network Resources to Virtual Network Components', *INFOCOM*, vol. 1200, no. 2006, p. 12.

ZINNER, T., GEISSLER, S., LANGE, S., GEBERT, S., SEUFERT, M. & TRAN-GIA, P. 2017, 'A discrete-time model for optimizing the processing time of virtualized network functions', *Computer Networks*, vol. 125, p. 11.