

**A Dissertation submitted in fulfilment of the
requirements for the degree of Doctor of
Philosophy**

Autonomic Management of Software Defined Networks

**DAIM can provide the environment for building autonomy in
distributed electronic environments - using OpenFlow networks
as the case study**

Ameen Reda Banjar

Spring 2016

University of Technology, Sydney,
Faculty of Engineering and Information Technology
Centre for Real Time Information Networks

Supervisor

Professor Robin Braun

Co-supervisor(s)

Dr. Bruce Moulton

Certificate of Original Authorship

I certify that the work in this dissertation has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the dissertation has been written by me. Any help that I have received in my research work and the preparation of the thesis it self has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the dissertation.

Signature of Candidate

_____ day of _____, 2016

Acknowledgments

I would like to express my sincere gratitude to my supervisor Professor Robin Braun, who guided me through this research, was a great source of encouragement, and always made me go that extra mile to resolve the various dilemmas that lead to this dissertation. I learned a lot from his mature view on topics from a high level and excellent computer and networking skills. I am gratefully thanking him for his guidance to me throughout the years. Also, his patience with me is always acknowledged and appreciated. I would also thank my Co-supervisor, Dr Bruce Moulton, for supporting me throughout this research work.

I am grateful to Dr Zenon Chaczko from the School of Computing and Communications for assistance in the publication of Springer book chapters, Chairing AP-CASE'14 in Bali and being a member of my CA panel. Also, for the very helpful technical chats and continuous support in the past years.

I am pleased to work in close collaboration with Pakawat Pupatwibul, my research collaborator and best friend, and I am gratefully thanking him for his invaluable contributions and innovative ideas towards this research project. I am so proud of what we have achieved together, thank you Pakawat.

During the period of this research, I also benefited greatly from interactions and discussions with my colleagues, namely Dr Abdallah AL Sabbagh and to whom I'm deeply indebted. A special mention also goes to Md Imam Hossain from the School of Electrical and Mechanical Engineering of ongoing solutions to the problems in C/C++ programming and extensive help in setting up the testbed used for this research. My big thanks also go to the anonymous reviewers for their work in improving the quality regarding my publications which have been published and being under review now.

The interactions, technical discussions and conversations with my friends and members of the Centre for Real-time Information Networks helped me during the period of this research, not only regarding resolving quick technical challenges but also with regards to "lightening up". I have enjoyed many useful and happy chats with them. I have been very fortunate to have them around during my Ph.D. research study.

I am honoured to have got government's scholarship from Saudi Arabia. Thank for their generosity, which has allowed me to be able to pursue the degree. Without this, I would not have been able to do it. I would also like to acknowledge the School of Computing and Communications at UTS for ongoing travel and infrastructure support.

Last, but by no means the least, I would like to thank my family for their endless support and all of the sacrifices that they have made on my behalf. Thank you to my mother, Jamilah Banjar, also to my daughter, Miar. You have provided me with plenty of love, patience, support, and encouragement over the years and for that I am incredibly grateful.

Some contents in chapter 1 and chapter 2 have been submitted to UTS subjects (32144: IT Research Preparation and 32931: Technology Research Methods). These subjects aim to guide research students to prepare literature reviews also experience in the design of research studies, in the analysis and interpretation of data, and in report presentation. As these contents have not been published; they can not be referenced but they can be searched by some software such as Turnitin.

Abstract

Next generation networks need to support a broad range of services and functionalities with capabilities such as autonomy, scalability, and adaptability for managing networks complexity. In present days, network infrastructures are becoming increasingly complex and challenging to administer due to scale and heterogeneous nature of the infrastructures. Furthermore, among various vendors, services, and platforms, managing networks require expert operators who have expertise in all different fields.

This research relied on distributed active information model (DAIM) to establish a foundation which will meet future network management requirements. The DAIM is an information model for network solutions which considers challenges of autonomic functionalities, where the network devices can make local and overall network decisions by collected information. The DAIM model can facilitate networks management by introducing autonomic behaviours. The autonomic behaviours for communication networks lead networks to be self-managed and emerge as promising solutions to manage networks complexity.

Autonomic networks management aims at reducing the workload on network operators from low-level tasks. Over the years, researchers have proposed a number of models for developing self-managed network solutions. One such example is the common information model (CIM), which is described as the managed environment that attempts to merge and extend the existing conventional management and also uses object-oriented constructs for overall network representation. However, the CIM has limitations coping in complex distributed electronic environments with multiple disciplines.

The goal of this research is defined as development of a network architecture or a solution based on the DAIM model, which is effectively distribute and automate network's functions to various network devices. The research first looks into the possibilities of local decision-making and programmability of network elements for distributed electronic environments with an intention to simplify network management by providing abstracted network infrastructures. After investigating and implementing different elements of the DAIM model in network forwarding devices by utilising virtual network switches, it discovers that a common high-level interface and framework for network devices are essential for the development of network solutions which will meet future network requirements.

The outcome of this research is the development of (DAIM OS) specification. The DAIM OS is a network forwarding device operating system which is compliant with

the DAIM model when it comes to network infrastructure management and provides a high-level abstracted application programming interface (DAIM OS API) for creating network service applications. Through the DAIM OS, network elements will be able to adapt to ever changing environments to meet the goals of service providers, vendors, and end users. Furthermore, the DAIM OS API aims to reduce complexity and time of network service applications development.

If the developed DAIM OS specification is implemented and if it functions as predicted in the design analyses; that will result in a significant milestone in the development of distributed network management.

This dissertation has an introduction in chapter 1 followed by five parts in order to draw a blueprint for information model as a distributed independent computing environment for autonomic network management. The five parts include lending weight to the proposition, gaining confidence in the proposition, drawing conclusions, supporting work and lastly is appendices.

The introduction in chapter 1 includes motivations for the research, main challenges of the research, overall objectives, and review of research contributions. After that, to lend weight to the proposition as the first part of the dissertation, there is chapter 2 which presents the background and literature review, and chapter 3 which has a theoretical foundation for the proposed model. The foundation consists of a generic architecture for complex network management and agents to aggregate distributed network information. Moreover, chapter 3 is probably more about a state of the art in software engineering than about real implementation to engineer autonomic network management.

The second part of the dissertation is to gain confidence in the proposition which includes attempting to implement the DAIM model in chapter 4 with some tests to report good performance regarding convergence and robustness for the service configuration process of network management. Also, the second part has a specification of true abstraction layers in chapter 5. The specification of true abstraction layers proposes a high-level abstraction for forwarding networking devices and provides an application program interface for network service applications developed by network operators and service providers. The implementation in chapter 4 is supported by the fourth part of the dissertation in chapter 10 which supports the theoretical foundation, designing, modelling, and developing the distributed active information model via simulation, emulation and real environments.

The third part of this dissertation provides the way to draw conclusions as shown in chapter 7 which has the overall research summary, validation of the propositions, contributions and discussion, limitations and finally recommendations for future works.

Finally are the appendices in Appendix A, Appendix B, Appendix C and Appendix D which provide a developing code of the core DAIM model and show different setting up for testbed environments.

My Related Publications

Most of the technical discussions, contributions, and theories in this dissertation are based on the following publications written by the author, and the other three are co-authored:

A. International Journal Publications:

- [J1] Pakawat Pupatwibul, Ameen Banjar, Abdallah AL Sabbagh, and Robin Braun. A comparative review: Accurate OpenFlow simulation tools for prototyping. *Journal of Networks*, 10(5):322–327, 2015.
- [J2] Pakawat Pupatwibul, Ameen Banjar, and Robin Braun. Performance evaluation of tcp/ip vs. OpenFlow in INET framework using OMNeT++, and implementation of intelligent computational model to provide autonomous behaviour. In *Osaka acbpps actis 2014*, number 2189-1028, pages 43–56, Osaka, Japan, 2014. The International Academic Forum (IAFOR), The Asian Conference on Technology, Information & Society 2014.
- [J3] BANJAR, A., PUPATWIBUL, P., BRAUN, R. 2014, DAIM: a Mechanism to Distribute Control Functions within OpenFlow Switches. *Journal of Networks*, North America, 9, jan. 2014. Available by: google scholar <jnw090119>. Date accessed: 08 Oct. 2014.
- [J4] Banjar, A., Pupatwibul, P., Sabbagh, A.A. & Braun, R. 2014, 'Using an ICN Approach to Support Multiple Controllers in OpenFlow', *International Journal of Electrical & Computer Sciences*, vol. 14, no. 2.

B. Chapter of Book Publications:

- [B1] Banjar, A., Pupatwibul, P., Sabbagh, A. & Braun, R. 2015, 'Comparison of TCP/IP routing versus OpenFlow table and implementation of intelligent computational model to provide autonomous behavior', In *Computational Intelligence and Efficiency in Engineering Systems*, pages 121–142. Springer, 2015.
- [B2] Pupatwibul, P., Banjar, A., Sabbagh, A. & Braun, R. 2014, 'An Intelligent Model for Distributed Systems in Next Generation Networks', in R. Klempous,

J. Nikodem, W. Jacak & Z. Chaczko (eds), Advanced Methods and Applications in Computational Intelligence, vol. 6, Springer International Publishing, pp. 315-34.

C. International Conference Publications:

- [C1]** Banjar, A.; Papatwibul, P.; Braun, R.; Moulton, B., "Analysing the performance of the OpenFlow standard for software-defined networking using the OM-NeT++ network simulator," Computer Aided System Engineering (APCASE), 2014 Asia-Pacific Conference on , vol., no., pp.31,37, 10-12 Feb. 2014
- [C2]** Papatwibul, P., Banjar, A., Al Sabbagh, A. & Braun, R. 2013, 'Developing an application based on OpenFlow to enhance mobile IP networks', Local Computer Networks Workshops (LCN Workshops), 2013 IEEE 38th Conference on, IEEE, pp. 936-40.
- [C3]** Al Sabbagh, A., Papatwibul, P., Banjar, A. & Braun, R. 2013, 'Optimization of the OpenFlow controller in wireless environments for enhancing mobility', Local Computer Networks Workshops (LCN Workshops), 2013 IEEE 38th Conference on, IEEE, pp. 930-5.
- [C4]** Papatwibul, P., Banjar, A. & Braun, R. 2013, 'Using DAIM as a reactive interpreter for OpenFlow networks to enable autonomic functionality', Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM, ACM, pp. 523-4.
- [C5]** Papatwibul, P., Sabbagh, A.A., Banjar, A. & Braun, R. 2012, 'Distributed Systems in Next Generation Networks', 1st Australian Conference on the Applications of Systems Engineering ACASE'12, p. 32.

List of Figures

1.1. Conventional Networks Vs. Programmable Networks	9
1.2. SDN architecture Vs. SDN with DAIM architecture	14
1.3. Overview of research structure	16
1.4. Dissertation plan	19
2.1. Centralised Management Paradigm	28
2.2. Distributed Management Paradigm	29
2.3. Autonomic Network Management	36
2.4. Increasing complexity to manage complicated system	38
2.5. Computer hardware architecture with GPU	39
2.6. Comparison between OpenGL and Vulkan with increasing complexity of computer application	40
2.7. Drawing Triangle by using OpenGL	43
2.8. Components of Smart Packets project, including ANEP Daemon and VM within each device	44
2.9. Software-Defined Networks (SDN) architecture	46
2.10. Routing Control Platform RCP	48
2.11. Distributed Routing Control Platform RCP	48
2.12. Ethane architecture decoupling control logic	50
2.13. OpenFlow architecture, theoretically similar to Ethane	52
2.14. OpenFlow packet processing flowchart	54
2.15. Actions associated with Flow entry of the flow table within switch . .	55
2.16. Flow table entry includes matching, status and action	55
2.17. Header fields in the flow entry for (wildcard and exact fields)	56
2.18. OpenFlow stack layers and TLS/SSL channel to controller	58
2.19. NOX structure and components	64
2.20. Overview of Beacon Architecture [59]	65
2.21. Project Floodlight Infographic [72]	66
2.22. Kandoo architecture	67
2.23. HyperFlow architecture	68
2.24. Onix architecture	69
3.1. Software-Defined Networking Architecture	75
3.2. OpenFlow networks structure	75
3.3. DAIM model structure	77

3.4. (A) Computer architecture, (B) SDN architecture, (C) DAIM architecture	81
3.5. Left side shows the northbound (NBI) and southbound interface (SBI) of SDN architecture compared to right side which presents DAIM module architecture with super southbound interface (SSBI)	82
3.6. DAIM module structure supported by agents	83
3.7. DAIM agents provider and components	84
3.8. Packet processing within OpenFlow switch integrated with DAIM model	86
3.9. DAIM agent owns flow entries in DAIM model	87
4.1. Integrating DAIM model to OpenFlow switch	92
4.2. Establish the connection to DAIM	93
4.3. DAIM exchanging information with OpenFlow switch	93
4.4. Simple setup of DAIM model with simple controllers and system requirement database	96
4.5. Hello message structure for local links discovery	99
4.6. Activity diagram of hello message (sending/receiving) sides	100
4.7. Links share message structure for linear topology	102
4.8. Links share message structure for tree topology	102
4.9. Links share message structure for ring topology	103
4.10. Links share message structure for mesh topology	104
4.11. Activity diagram of links share message (sending/receiving) sides	105
4.12. Activity diagram of switch alive message (sending/receiving) sides	106
4.13. Shortest path calculated by each DAIM and setting no flooding ports	108
4.14. Flow entries installation	108
4.15. Mininet emulator shows empty flow entries tables	109
4.16. DAIM shows ARP broadcast flood (Mininet emulator)	110
4.17. flow-mod packet for matching and packet comes to SW_1	112
4.18. flow-mod sent by the DAIM model (Wireshark captured)	113
4.19. Flow tables after the first ping (Mininet emulator)	114
4.20. Flowchart of packet-in to the DAIM model	115
4.21. Ping traffic of DAIM model vs Legacy switch in linear topology	118
4.22. Ping traffic of DAIM model vs Legacy switch in tree topology	120
4.23. Ping traffic of DAIM model vs OpenFlow-NOX in ring topology	121
4.24. Ping traffic of DAIM model vs OpenFlow-NOX in mesh topology	123
4.25. Average RTT of different DAIM model topologies	125
4.26. Latency of DAIM vs. Latency of NOX vs. Latency of POX	126
4.27. Latency responses of DAIM, NOX and POX	127
5.1. DAIM OS network architecture	136
5.2. Different components in the DAIM switch platform	139
5.3. Interfaces between different DAIM switch platform components	140

5.4.	The DAIM OS is kept compatible across different hardware and software configurations via different middleware	141
5.5.	DAIM OS modules interconnectivity	143
5.6.	Entries indicators in a switch port for a table configuration	144
5.7.	The DAIM OS means of information retrieval	145
5.8.	Link connecting two adjacent DAIM switch devices	157
5.9.	Packet flow in a DAIM switch	170
5.10.	Retaining of incoming packets by the DAIM OS	175
5.11.	DAIM cloud fetching DAIM switch id using DAIM OS cloud messages	184
5.12.	DAIM OS cloud message structure for data ordering	185
5.13.	DAIM OS packet forwarding pipeline	187
5.14.	Forwarding of matching packets by the DAIM OS	188
5.15.	Swapping internet access to hosts under two DAIM switches	189
10.1.	The DAIM model uses simulation, real network, and emulation based for implementation to draw the conclusion	240
10.2.	Traditional network topology in OMNeT++	241
10.3.	OpenFlow network topology in OMNeT++	241
10.4.	Data Transfer Rate DTR of traditional and OpenFlow networks . . .	242
10.5.	Round Trip Time RTT of traditional and OpenFlow networks	243
10.6.	OpenFlow network topology for Australian cities	245
10.7.	Performance of OpenFlow networks based on controller location . . .	246
10.8.	RTT for controller location near to (Sydney/Perth)	247
10.9.	Controller near Sydney (RTT of traditional vs. OpenFlow networks)	248
10.10.	Controller near Perth (RTT of traditional vs. OpenFlow networks)	248
10.11.	Connection Establishment and Termination of TCP	250
10.12.	Connection Establishment and Termination of OpenFlow Networks .	251
10.13.	Architecture of the DAIM model for Basic carrier functionality . . .	252
10.14.	Basic carrier topology testbed connected with one controller	252
10.15.	Latency of NOX vs. Latency of DAIM with NOX	253
10.16.	Latency of POX vs. Latency of DAIM with POX	254
10.17.	RTT of POX vs. RTT of DAIM with POX	255
10.18.	Average Response for Latency of NOX and DAIM with NOX/ POX and DAIM with POX	256
10.19.	Throughput of NOX vs. Throughput of DAIM with NOX	256
10.20.	Throughput of POX vs. Throughput of DAIM with POX	258
10.21.	Average Response for Throughput of POX, NOX and DAIM with NOX/ POX	258
10.22.	Semi-Autonomous functionality of DAIM model	259
10.23.	Semi-Autonomous DAIM model testbed	260
10.24.	Semi-Autonomous of DAIM model with packet size increasing	261
10.25.	RTT for NOX, POX and Semi-Autonomous DAIM	262
10.26.	RTT for NOX and Semi-Autonomous-of-DAIM with packet size increasing	263

10.27	Fully Autonomous functionality of DAIM model	264
10.28	Integrating the DAIM model to Raspberry Pi	264
10.29	Topology of DAIM model integrated to Raspberry Pi in Lab	265
10.30	RTT for DAIM vs. POX	266
A.1.	GitHub page for DAIM model	288
A.2.	Memory Block for storage module	315
C.1.	Mininet structure and components [77]	332
C.2.	GUI for NOX controller with log view and topology view	334
D.1.	Raspberry Pi hardware components of model (B)	339
D.2.	OpenFlow networks using Raspberry Pi integrated with DAIM model	340
D.3.	Verifying OpenFlow dissector in Wireshark	345
D.4.	Captured packets in Wireshark for OpenFlow	346

List of Tables

1.1. Decomposition of the knowledge	10
2.1. IBM autonomic computing Self-CHOP concepts [40]	33
2.2. Autonomic network management and traditional network management	37
2.3. Comparing between OpenGL and Vulkan [145, 127]	41
2.4. Controller to switch messages	57
2.5. Symmetric messages	59
2.6. Asynchronous messages	60
2.7. Comparison of network layers and functionality between TCP/IP, OpenFlow, and traditional telephone	61
2.8. Distributed controllers approaches	70
3.1. Comparison of DAIM Protocol and DAIM API	84
3.2. Comparison of OpenFlow and DAIM model processes	88
4.1. Discovery of Topology by DAIM model as steps	97
4.2. packet-in for ARP request in Wireshark	110
4.3. packet-in for ARP reply in Wireshark	111
4.4. packet-in for ICMP request in Wireshark	113
4.5. Testing TCP bandwidth between h1 and h2 of Linear topology	117
4.6. Testing UDP bandwidth between h1 and h2 of Linear topology	119
4.7. Testing TCP bandwidth between h1 and h2 of Tree topology	119
4.8. Testing UDP bandwidth between h1 and h2 of Tree topology	120
4.9. Testing TCP bandwidth between h1 and h2 of Ring topology	122
4.10. Testing UDP bandwidth between h1 and h2 of Ring topology	122
4.11. Testing TCP bandwidth between h1 and h2 of Mesh topology	124
4.12. Testing UDP bandwidth between h1 and h2 of Mesh topology	124
5.1. DAIM OS modules tasks	142
5.2. The DAIM switch state flags and their meaning of setting	149
5.3. The DAIM switch port flags and their meaning of setting	153
5.4. Entity state flags and their meaning of setting	154
5.5. Switch link state flags and their meaning of setting	158
5.6. Port state flags and their operating mode	163
5.7. Entry identification variables for configuration tables	176
5.8. DAIM tables configuration in DAIM switch 1 by DAIM application	189

5.9.	DAIM tables configuration in DAIM switch 2 by DAIM application .	190
6.1.	DAIM OS actions on DAIM information table fields	194
6.2.	DAIM OS actions on Switch table fields	195
6.3.	DAIM OS actions on Switch port table fields	196
6.4.	DAIM OS actions on Entity table fields	197
6.5.	DAIM OS actions on Entity port table fields	198
6.6.	DAIM OS actions on Entity ARP table fields	198
6.7.	DAIM OS actions on Link table fields	199
6.8.	DAIM OS actions on Packet forwarding table fields	200
6.9.	DAIM OS actions on Switch configuration table fields	201
6.10.	DAIM OS actions on Switch port configuration table fields	201
6.11.	DAIM OS actions on Entity configuration table fields	202
6.12.	DAIM OS actions on Link configuration table fields	203
6.13.	Flow rules by different entries in Packet forwarding table	204
10.1.	OMNeT++ channel types	244
10.2.	Distance between Australian cities	244
A.1.	Actions of public functions in the object list class	317
A.2.	API dependencies for DAIM modules	318
B.1.	Real network vs. network simulation [5]	326
D.1.	Open vSwitch configuration Summary	341

Nomenclature

AC	Autonomic Computing
ACS	Autonomic Computing System
AI	Artificial Intelligence
ANM	Autonomic Network Management
ANS	Autonomic Nervous System
API	Application Programming Interface
AS	Autonomous System
CIM	Common Information Model
CLI	Command Line Interface
CMIP	Common Management Information Protocol
CMIS	Common Management Information Service
CPU	Central Processing Unit
DAIM	Distributed Active Information Model
DAIM OS	DAIM Operating System
DMTF	Distributed Management Task Force
DTR	Data Transmission Rate
GPU	Graphics Processing Unit
HD	Hard Drive
iBGP	interior Border Gateway Protocol
ICN	Information-Centric Networking
IOS	International Organisation for Standardisation

IP	Internet Protocol
LLDP	Link Layer Discovery Protocol
MAC	Media Access Control
MIB	Management Information Base
MTU	Maximum Transmission Unit
NE	Network Elements
NETCONF	Network Configuration Protocol
NGN	Next Generation Network
NIB	Network Information Base
NM	Network Management
NMS	Network Management System
NOS	Network Operating System
NSOS	Network Switch Operating Systems
OS	Operating System
OSS	Operating Support System
P2P	Peer-to-Peer
POSIX	Portable Operating System Interface
POX	Python based controller
QoS	Quality of Service
RAM	Random-access memory
RCP	Routing Control Platform
SDN	Software-Defined Networking
SID	Shared Information and Data Model
SNMP	Simple Network Management Protocol
SRD	System Requirement Database

TCP	Transmission Control Protocol
TLS	Transport Layer Security
TMF	TeleManagement Forum
UML	Unified Modelling Language
XML	Extensible Markup Language
XML-RPC	XML-encode Remote Procedure Call

Contents

Acknowledgments	iii
Abstract	v
Nomenclature	xv
1. Introduction	1
1.1. Motivations	3
1.1.1. The Swinging Pendulum of Network Structure	4
1.1.2. Evolve Networks Regularly	5
1.1.3. Limitation of Current Networking	5
1.2. Autonomic Communication	7
1.2.1. Objective of Autonomic Network Management	7
1.2.2. Challenges of Fully Freedom Management	8
1.2.3. Autonomics Abstraction Layers	8
1.2.4. Possible Development Environments	9
1.2.5. Network management issues	9
1.2.6. Using OpenFlow networks as a case study	11
1.3. Research Objectives and Scope	11
1.3.1. Objective	12
1.3.2. Scope	12
1.4. Research Proposition	13
1.4.1. Core Proposition	13
1.4.2. Attendant Propositions	13
1.5. Overview of Methodological Framework	15
1.6. Statement of Contributions	15
1.7. Overview of Dissertation Structure	18
 I. Lending Weight to the Thesis	 21
2. Background and Literature Review	23
2.1. High level background	23
2.2. Network Management	24
2.2.1. Network management functions and protocols	25
2.2.2. Network Managing Paradigms	27

2.3.	Automatic and Autonomic Systems	31
2.4.	Autonomic Computing Concepts	32
2.5.	Introducing Autonomic Communication	32
2.5.1.	Lesson Learnt from Similar Fields	34
2.5.2.	Human Biological Mechanisms	34
2.5.3.	Possibility of Autonomic Network Management	35
2.6.	Managing Complicated System by Increasing Complexity	38
2.6.1.	Understanding Complex and Complicated system	39
2.6.2.	Possibility of Implementing Complexity to Complicated System	42
2.7.	Introducing a New Norm of Networking (SDN)	45
2.7.1.	Major contributions toward SDN paradigm	47
2.7.2.	Centralised and Distributed SDN Controllers	59
2.8.	Information Models for network management	69
2.8.1.	CIM from (DMTF)	71
2.8.2.	SID from (TMF)	71
2.8.3.	Issues of CIM and SID	71
2.9.	Summary	72
3.	Theory of The Proposed Model	73
3.1.	Restatement of the Thesis	74
3.1.1.	Uniqueness of DAIM model	76
3.1.2.	DAIM model Vs. current Information Models	77
3.1.3.	Objectives of the DAIM model	78
3.2.	DAIM model Architecture	79
3.2.1.	Describing the DAIM model	82
3.2.2.	Packet processing within DAIM model	85
3.2.3.	Risk Scenarios of DAIM model	87
II.	Gaining Confidence in the Thesis	89
4.	Attempting to Implement DAIM Model	91
4.1.	Introduction	91
4.2.	Setting up the Demonstration	91
4.2.1.	Integrating DAIM model to OpenFlow Switch	92
4.3.	Simple Reference Implementation:	94
4.3.1.	DAIM model components	94
4.3.2.	Discover Networks Topology	95
4.3.3.	Calculating the Shortest Path with DAIM model	106
4.3.4.	Flow Table in DAIM model with Example of Ping	107
4.3.5.	Verifying DAIM model Functionalities	114
4.3.6.	DAIM model Recovery	127
4.3.7.	Discrepancy of the DAIM model implementation	128

5. Specification of DAIM OS	131
5.1. Introduction	131
5.1.1. Scope	132
5.1.2. Terms and definitions	132
5.2. DAIM Operating System (OS) overview	135
5.2.1. DAIM operating system (DAIM OS)	137
5.2.2. Network flow management by DAIM OS	137
5.2.3. Tables in DAIM OS	137
5.3. DAIM switch overview	138
5.3.1. DAIM switch	138
5.3.2. DAIM application	138
5.3.3. DAIM switch platform	138
5.3.4. DAIM switch stack	138
5.4. DAIM OS model	141
5.5. DAIM OS table model	143
5.6. DAIM OS information retrieval model	144
5.7. DAIM OS tables description	146
5.7.1. Information tables	146
5.7.2. Network management tables	158
5.8. DAIM OS API description	166
5.8.1. Reading of DAIM tables data	167
5.8.2. Writing data to DAIM tables	169
5.8.3. Catching signals from the DAIM OS	176
5.9. DAIM Cloud	179
5.10. DAIM OS System API	180
5.11. DAIM OS cloud protocol	181
5.11.1. DAIM OS cloud message header	181
5.11.2. DAIM OS cloud messages	181
5.12. DAIM OS packet forwarding pipeline	187
5.13. DAIM Switch network design scenario	188
5.13.1. Scenario A: a simple enterprise network	188
5.14. Conclusion	190
 6. Extra Details for DAIM OS Specification	 191
6.1. DAIM OS API description	191
6.1.1. Reading of DAIM tables data	191
6.1.2. Writing data to DAIM tables	194
6.1.3. Catching signals from the DAIM OS	215
6.2. DAIM OS cloud protocol	218
6.2.1. DAIM OS cloud messages	218

III. Drawing Conclusions	221
7. Conclusion	223
7.1. Research Summary	223
7.2. Validation of Research Propositions	224
7.3. Research Limitation	229
8. Research Contribution and Discussion of Findings	231
9. Direction for Future Works and Recommendations	235
IV. Supporting Work	237
10.Supporting the Proposed Model Implementation	239
10.1. Exercise OpenFlow Networks in OMNeT++	239
10.1.1. Performance of OpenFlow Vs. Traditional Networks	239
10.1.2. Implement OpenFlow to Australian cities	243
10.2. Exercise the DAIM model via Mininet	249
10.2.1. Integrating DAIM model to OpenFlow Switch	249
10.3. Integrating DAIM model with Raspberry Pi	264
V. Appendices	285
A. DAIM model with a Simple Controller	287
A.1. DAIM model Header Files	287
A.1.1. Controller Header	287
A.1.2. ARP Header	290
A.1.3. Ethernet Header	291
A.1.4. Communication Header	291
A.1.5. OpenFlow Header	292
A.2. Code of DAIM model and Controller	294
A.2.1. Main Cpp	294
A.2.2. Controller Cpp	296
A.2.3. Communication Cpp	312
A.3. Storage module of the DAIM model	315
A.4. API dependencies for DAIM modules	317
A.5. Basic carrier for OF messages	319
B. OMNeT++ (Simulation Environment)	325
B.1. Installing OMNeT++ in Linux OS	325
B.2. Import INET framework to OMNeT++	327
B.3. Import OpenFlow in OMNeT++:	328
B.4. OMNeT++ components:	328

B.5. Establish / Terminate Connection	329
B.5.1. Traditional Networks	329
B.5.2. OpenFlow Networks	330
C. Mininet (Emulation Environment)	331
C.1. Building SDN lab environment	331
C.1.1. Installing SDN Controller (NOX)	331
C.1.2. Installing Mininet	333
C.2. Basic DAIM model with OpenFlow	336
C.2.1. Introducing controller benchmarker	337
D. Raspberry Pi (Real Environment)	339
D.1. Setting up OpenFlow on Lab	340
D.2. Configuration summary	340
D.2.1. Assign static IP for network interfaces	341
D.2.2. Setting up OpenFlow and DAIM	341
D.2.3. Setting OpenVswitch for Raspberry Pi	342
D.2.4. General configuration for Open vSwitch	343
D.2.5. Running DAIM model in Raspberry Pi	344
D.3. Installing Wireshark Dissector for OpenFlow	344

