

SYMPHONY - A CONTROLLER FOR HYBRID SOFTWARE DEFINED NETWORKS

A thesis submitted by **Vijaya Durga Chemalamarri**
in fulfilment of the requirements for the award of the degree
Master of Science in Computing Sciences (Research)

Faculty of Engineering and Information Technology
University of Technology Sydney

March 2016

© Copyright Vijaya Durga Chemalamarri March 2016.

All Rights Reserved

Certificate of Original Authorship

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

Production Note:

Signature removed prior to publication.

Acknowledgement

I would like to acknowledge and thank for all the encouragement, guidance and support provided by my supervisors Dr.Karla Felix Navarro and Dr.Priyadarshi Nanda.

I am grateful for having a family that installed in me the value of knowledge since early age. I am very grateful to have my sister Rohini as a constant source of motivation. I cannot insist enough on the support provided by my husband Deba and my in-laws.

I also thank all my friends for cheering and motivating me.

Abstract

Software Defined Networks (SDN) is currently an active area of research. As enterprises migrate to SDN, an inevitable network transitional state is a brownfield state, where both Software Defined and Legacy networks coexist. To achieve interoperability between legacy and Software Defined Networks and to leverage the existence of OpenFlow devices in the traditional network to improve existing network state, a Hybrid SDN controller is a desirable addition to any brownfield deployment of SDN. The thesis of this work aims to further the knowledge in the area of Hybrid Software Defined Networks by highlighting the requirements and challenges to be addressed while integrating legacy and Software Defined Networks. The requirements and challenges discussed in this thesis focus on path computation, packet forwarding, and centralised policy application. This is achieved by building a Hybrid SDN Controller. The main controller components - controller application that runs on POX, a route server to store legacy network information, a next-hop module, a path discovery module and a policy module. We also discuss three use cases of SYMPHONY hybrid SDN controller to demonstrate the controller's application and usefulness. We also perform few key tests to determine the efficiency of deploying a hybrid SDN controller in a network.

Publications Supporting this Thesis

The following is a list of publications resulting from this thesis.

Conferences

Fourth European Workshop on Software Defined Networks

1. Chemalamarri, V.D., Nanda, P. and Navarro, K.F., 2015, September. SYMPHONY—A Controller Architecture for Hybrid Software Defined Networks. In Software Defined Networks (EWSDN), 2015 Fourth European Workshop on (pp. 55-60). IEEE.

Contents

1	Introduction	12
1.1	KeyWords	12
1.2	Background	13
1.3	Aims and Objectives	14
1.4	Significance	14
1.5	Research Methodology	16
1.6	Key Contributions	17
1.7	Thesis Structure	17
1.8	Conclusions	18
2	Background and Related Work	19
2.1	Why Software Defined Networks?	19
2.2	SDN Architecture	20
2.2.1	Control plane	20
2.2.2	Data plane	21
2.2.3	OpenFlow	21
2.3	Hybrid Software Defined Networks	25
2.3.1	Existing Frameworks for deploying Hybrid Software Defined Networks	26
2.4	Conclusion	30
3	SYMPHONY—A Controller for Hybrid SDN	31
3.1	Need for Hybrid SDN Controller	31
3.2	Hybrid SDN Controller Design Challenges	32
3.3	HYBRID SDN CONTROLLER ARCHITECTURE	36
3.3.1	Packet-Forwarder	37
3.3.2	Legacy Routing Server	40
3.3.3	Path-finder	40

3.3.4	Next-hop	40
3.3.5	LLDP module for OF data plane discovery	41
3.3.6	Policy Transformation	41
3.3.7	Topology Discovery	42
3.4	Conclusion	48
4	Implementation and Use Cases	50
4.1	Use Case I: Establish communication	50
4.2	Use Case II: Centralised policy application	55
4.3	Use Case III: Using the TCP-ECN bit to divert traffic dynamically onto other links.	56
4.4	Conclusion	59
5	Testing and Results	60
5.1	Test Case1: Convergence over OF domain	60
5.1.1	Spanning Tree Protocol	60
5.2	Test Case2: Effect of presence of multiple egress nodes on routes learnt by LRS	62
5.3	Test Case3 :Applying Centralised Policy	64
5.4	Test Case 4: Traffic Engineering using Explicit Congestion Notification	67
5.5	Conclusion	67
6	Conclusions	68
6.1	Summary	68
6.2	Future Work	69
	Appendices	71
A	Chapter3	72
B	Chapter4	73
C	Chapter5	81
	Bibliography	86

List of Tables

2.1	flow table	22
2.2	Match Fields	22
2.3	Types of Actions	23
2.4	ofp_packet_in message structure	23
2.5	ofp_flow_mod message structure	24
2.6	ofp_packet_out message structure	24
2.7	ofp_port_status message structure	24
2.8	List of Events	25
3.1	Sample Policy.csv	42
4.1	OSPF Neighbour table of LRS	51
4.2	Policy Table	56
4.3	Dynamic policy inserted in policy table	58
5.1	STP Convergence time	61
5.2	OSPF Convergence time	61
5.3	Policy Table	65

List of Figures

1.1	Agile Model	16
2.1	SDN Architecture	20
2.2	Interaction between OpenFlow Control plane and Data plane	21
3.1	Hybrid SDN	33
3.2	Legacy control traffic	33
3.3	Selecting an egress node	34
3.4	Invalid routes	35
3.5	Application of inconsistent policy	35
3.6	SYMPHONY Controller	36
3.7	Switches_port dictionary	37
3.8	Multicast process to determine packet destination	38
3.9	Unicast process to determine destination	39
3.10	LRS connectivity with host	40
3.11	LRS connectivity with egress routers	40
3.12	Dijkstra's algorithm	41
3.13	Sample switches.txt	41
3.14	LLDP packet structure	42
3.15	OF node discovery process	43
3.16	Boundary nodes discovery process	44
3.17	Sample Edges.txt	44
3.18	Hosts discovery process	44
3.19	Sample Hosts.txt	45
3.20	Identification of nodes in Hybrid SDN	45
3.21	Emulated network topology	45
3.22	Switches.txt to store OpenFlow topology	45

3.23	Edges.txt to store boundary connections and nodes	46
3.24	FlowControl of SYMPHONY Controller for Hybrid SDN	48
4.1	OSPF Neighbour relation between LRS and edge routers	51
4.2	Communication between two hosts in a OF domain	52
4.3	Alternate paths incase of link failures	53
4.4	Choice of different edge routers for different remote destinations	54
4.5	Legacy topology	55
4.6	Connectivity setup for policy application	55
4.7	Policy application in action	56
4.8	Effect of ECN	58
4.9	ECN enabled packet as captured in wireshark	58
5.1	OSPF convergence Comparison graph	62
5.2	Hybrid topology with a single edge router	63
5.3	LRS routing table	63
5.4	Hybrid topology with multiple edge routers	64
5.5	Better routes learnt by LRS	64
5.6	Traffic blackhole in legacy networks incase of link failures	65
5.7	Alternate routes selected in SDN incase of link failures	65
5.8	Application of Policy-1	66
5.9	Application of policy-II	66
5.10	Response time	66
B.1	Flow table entries for flow between h3,h4 when all the links are operational	73
B.2	Flow table entries for flow between h3,h4 when link between OFSw3 and OFSw5 goes down	74
B.3	Flow table entries for flows between h1, h2 when all links are operational	75
B.4	Flow table entries for flows between h1 , h2 when the link between OFSw1, OFSw3 is down.	76
B.5	Flow table entries for flow between h3 ,h1	77
B.6	Flow table entries for flow between h3 , h2	78
B.7	Flow table entries for flows between h1,h2 with policy applied to divert the traffic via R3	79
B.8	Flow table entries for flows between h1,h2 with policy applied to divert the traffic via R3 and link R1, R3 is non operational	79

B.9	Flow table entries under normal conditions	80
B.10	Flow table entries when ECN is received by the h1 from h2	80
C.1	Topologies used for testcase 1. (A) 3 OF Switches,(B) 5 OF Switches,(C) 7 OF Switches, (D)14 OF Switches,(E) 21 OF Switches	82