Enhancing Network Scalability by Introducing Mechanisms, Architectures and Protocols

A thesis submitted for the degree of Doctor of Philosophy

Department of Electronic and Computer Engineering

College of Engineering, Design and Physical Sciences

Brunel University London

United Kingdom

By

Emad Alasadi

Supervised By:

Professor Hamed Al-Raweshidy

June 2017

Dedication

Dedicated to my dearest love, my wife (Yasameen),

my sweetheart, my son (Ali),

my little sweet daughter (Rose),

and the merciful heart my mother.

Abstract

In this thesis, three key issues that restrict networks from scaling up so as to be able to cope with the rapid increase in traffic are investigated and series of approaches are proposed and tested for overcoming them.

Firstly, scalability limitations owing to the use of a broadcast mechanism in one collision domain are discussed. To address this matter, servers under software-defined network architectures for eliminating discovery messages (SSED) are designed in this thesis and a backbone of floodless packets in an SDN LAN network is introduced. SSED has an innovative mechanism for defining the relationship between the servers and SDN architecture. Experimental results, after constructing and applying an authentic testbed, verify that SSED has the ability to improve upon the scalability of the traditional mechanism in terms of the number of switches and hosts. This is achieved by removing broadcast packets from the data and control planes as well as offering a better response time.

Secondly, the scalability restrictions from using routers and the default gateway mechanism are explained. In this thesis, multiple distributed subnets using SDN architecture and servers to eliminate router devices and the default gateway mechanism (MSSERD) are introduced, designed and implemented as the general backbone for scalable multiple LAN-based networks. MSSERD's proposed components handle address resolution protocol (ARP) discovery packets and general IP packets across different subnets. Moreover, a general view of the network is provided through a multi-subnets discovery protocol (MDP). A 23 computers testbed is built and the results verify that MSSERD scales up the number of subnets more than traditional approaches, enhances the efficiency significantly, especially with high load, improves performance 2.3 times over legacy mechanisms and substantially reduces complexity.

Finally, most of the available distributed-based architectures for different domains are reviewed and the aggregation discovery mechanism analysed to establish their impact on network scalability. Subsequently, a general distributed–centralised architecture with open-level control plane (OLC) architecture and a dynamic discovery hierarchical protocol (DHP) is introduced to provide better scalability in an SDN network. OLC can scale up the network with high performance even during high traffic.

Acknowledgements

First of all, I would like to express my appreciation and gratefulness to the Ministry of Higher Education-Iraq for funding my PhD study.

I would also like to offer my sincere gratitude to my supervisor Professor Hamed Al-Raweshidy for his guidance, advice and encouragement. This work would have not been possible without his help and support. He has helped me in many different ways and supported me through the regular discussions as well as providing the testbed items that I have used to verify the work developed in this thesis.

I am thankful to my second supervisor Professor Maozhen Li for his encouragement and support. My gratitude goes to my colleagues in the Wireless Networks and Communications Centre (WNCC), who have been supportive throughout this period.

I would like to acknowledge the great attitude of all the staff members that I have interacted with at Brunel University, especially Colin Millins, John Morse and Matthew Smith and the services team.

I would like to express gratitude and deep thanks to my wife for supporting me at all times. I am also indebted to my mother, brother and sister for their precious moral support and encouragement throughout the years of my study, whose love and inspiration travelled alongside me on my PhD journey.

I am very grateful to all my friends who have always given me their care and support. A special thanks to my best friend Hayder Abdullah whose support was precious.

Table of Content

Declaration

I certify that the effort in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree. I also certify that the work in this 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 and referenced.

Emad Alasadi

June 2017, London

List of Figures

List of Tables

List of Abbreviations

AMAC	Actual MAC
AMQP	Advanced Message Queuing Protocol
ARP	Address Resolution Protocol
AS	Autonomous System
BGP	Border Gateway Protocol
BPDU	Bridge Protocol Data Unit
Bpfes	Best path between the furthest edges of subnet
CAN	Campus Area Network
CAPEX	CAPital EXpenditures
DHCP	Dynamic Host Configuration Protocol
DHP	Dynamic Discovery Hierarchal Protocol
DoS	Denial of Service
Dst	Destination
D_t	Discovery time
Gbit/sec	Gigabit per second
GUI	Graphic User Interface
H_{cl}	Highest controller latency
HD	High Definition
H_{ll}	Highest link latency
HOL	Head-Of-Line
HPC	High Performance Competing
IANA	Internet Assigned Numbers Authority
IoT	Internet of Things
IP	Internet Protocol
IS-IS	Intermediate System to Intermediate System
L3_switch	Layer3 switch
LAN	Local Area Network
LAT	Last Activity Time
LLDP	Link Layer Discovery Protocol

L_p	Latency in each phase
MAC	Media Access Control
MAN	Metropolitan Area Network
MDP	Multi-subnets Discovery Protocol
MLD	Multicast Listener Discovery
M-LLDP	Messenger-Link Layer Discovery Protocol
MPLS	Multiple Protocols Labels Switching
MSSERD	Multiple distributed Subnets using SDN architecture and Servers to Eliminate Router Devices and the default gateway mechanism
MTO	Multi-to-One
MTU	Maximum Transmission Unit
NAT	Network address translation
N_{cp}	Number of packets in control plane
N_{dc}	Number of devices performing caching
N_{def}	Number of different Ethernet frames
N_{dp}	Number of packets in data plane
NFV	Network Functions Virtualisation
N_l	Number of data links
N_{LR}	Number of Links from routers
N_{Lsw}	Number of links from switches
N_m	Number of messages
N_{ol}	Number of out links
N_{op}	Number of phases
NOS	Network Operating System
N_{pDp1}	Number of packets of the discovery process for one phase
N_{pDpF}	Number of packets of the full discovery process
N_r	Number of routers
N_s	Number of switches
N_{sb}	Number of switches in the best path
N_{su}	Number of subnets
N_{sw}	Number of switches

NTT	Nippon Telegraph and Telephone
OLC	Open-Levels Control plane architecture
OPEX	OPerating EXpenses
OSPF	Open Shortest Path First
OVS	Open Virtual Switch
PMAC	Pseudo MAC
QoS	Quality of Service
RIP	Routing Information Protocol
RPS	Requests Per Second
RSTP	Rapid Spanning Tree Protocol
SDN	Software Defined Networking
Src	Source
SSED	Servers under Software-Defined network architectures to Eliminate Discovery messages
STP	Spanning Tree Protocol
SVR	SDN Virtual Router
T	Time
TCP	Transmission Control Protocol
TLV	Type Length Value
TTL	Time To Live
VLAN	Virtual LAN
VM	Virtual Machines
WAN	Wide Area Network

Chapter 1

Introduction

1.1 Introduction

The Internet has changed the world, regarding how we lead our daily lives. In recent years, network load has increased rapidly [1][2], because different network technologies and internet applications have appeared, such as the Internet of things (IoT), Internet applications including social media (e.g. Facebook, Instagram, etc.), VoIP applications (e.g. Viber, What's up, etc.) and high definition (HD) video on demand applications (e.g. YouTube, Dailymotion, etc.). These technologies and applications are used for communication among people/things for different purposes, including education (e.g. distance learning), business (e.g. video conferencing), to keep in touch with family friends as well as for entertainment and smart environment purposes. Moreover, it seems that this trend is very unlikely to stop as we are becoming increasingly surrounding by smart devices [3]–[5].

As a consequence, the load on networks will continue to increase in terms of the number of devices added and also regarding the amount of user data and control messages generated, the massive amount of huge control traffic is putting more and more pressure on network busses. The extensive traffic on the network, which it is not designed for, leads to congestion that, in turn, slows down the network [6], possibly even shutting down the connection with customers (i.e. enterprises/individuals) [7] or causing other reliability issues [8]. Accordingly, the spotlight is put on the scalability of the network as this remains an unsolved issue [3][9].

The legacy network mechanisms, devices and architectures have various limitations, which prevent networks from being scale up to cope with demand requirements. Firstly, using traditional mechanisms, such as a broadcast discovery, limits the number of hosts in one collision domain [10]. Secondly, using legacy middlebox devices (e.g. routers) leads to degrading of the number of requests that the network can deal with per second, because of congestion on ports [11], whilst using the default gateway mechanism leads to an increase in the number of control packets in network. Thirdly, using conventional fully distributed architecture and an aggregation discovery mechanism results in an extensive amount of control signals in data plane that have a negative effect on discovery and rediscovery times in the intra and inter domains, which in turn prevents scaling up of the network.

There are different definitions for scalability, in general [12] and in networks, in particular. Scalability of a network is *the ability to increase the number of serviced customers without degrading the performance of network*. some researchers [13]–[15] attribute it to the performance of the controller, while others associate it with the number of requests that can be handled per second [12]. This raises the following two questions:

- *How can the scalability in a network be defined?*
- *What metrics should be used for measuring it?*

To overcome the scalability challenges that occur owing to legacy network elements (i.e. devices, architectures and mechanisms) a number of different studies aimed at enhancing network scalability have been carried out. These include eliminating broadcast messages [16][17] inside the subnets, using virtual middlebox devices (e.g. virtual devices) instead of hardware ones [18][19] in order to enhance the router abilities and introducing different architectures [20][21].

However, most of the proposals have not solved the scalability issue completely, whilst at the same time taking into account other network factors, such as security [22], complexity [23] and reliability [24]. In addition, none have appeared as standard or officially used, as yet. Furthermore, most of them not involve enhancing the scalability with peak or overloaded traffic, which is the main cause of the scalability issue.

In Section 1.2, the thesis's motivation is explained, while the aim and objectives are provided in Section 1.3. The challenges are explained in Section 1.4 and Section 1.5

presents the contributions of this thesis. Finally, the thesis scope and content, with some scalability percentages of achievements, are covered in Sections 1.6 and 1.7, respectively.

1.2 Motivations

The motivations for carrying out this study are as follows.

- The scalability of networks remains a salient issue that needs to be considered by the research community [3][25].
- The elements of the traditional network have various limitations that hinder scaling up. Firstly, in terms of the traditional mechanism element, a collision domain (e.g. a subnet) cannot scale up practically to more than 500 hosts, as recommended by Cisco [10], because of the broadcast discovery mechanism, which is an insufficient number for meeting IoT requirements. Secondly, in terms of the algorithm element, the back-off algorithm used in collision networks to solve collisions, theoretically, leads to the number of hosts being limited to 1,024 [26], which also raises the scalability problem in these networks. Thirdly, regarding the protocol element, the Spanning Tree Protocol (STP), which is one of the important ones for preventing loop storm and hence, is essential for any network, supports only seven hops as a maximum bridged LAN diameter [27], thus restricting network growth. Fourthly, in terms of legacy devices, A growing amount of traffic puts more pressure on the middlebox devices (e.g. routers), which increases the likelihood of congestion [11]. Moreover, when routing packets, routers modify the layer two header in the Ethernet frame at each hop [28], which increases latency and in turn, affects network performance and scalability. In addition, using router devices leads to the compulsory default gateway mechanism, which has broadcast mechanism limitations, hence inhibiting scalability [10]. Finally, in relation to traditional architectures, the use of a distributed architecture with aggregation discovery mechanism in different domains leads to the data plane being used to transfer the discovery packets through the network, which results in more load and the consumption of the resources of that plane. This, consequently, has a negative effect on the discovery and convergence time, which, in turn, impacts on the number of subnets that can

be discovered in a network, because the discovery time for any new event in the fully distributed architecture has a direct relationship with network size [29].

- In a traditional network, different protocols are used to perform the discovery process, where for example the Link Layer Discovery Protocol (LLDP) [30] is used inside a subnet to discover switches, while for intra domains among routers the Open Shortest Path First (OSPF)[31] is commonly used to discover the network. In addition, the Border Gateway Protocol (BGP) [32] is used to discover the inter-domains. As consequence, there is a need to decrease the number of protocols to reduce the complexity and the effort and configuration times.
- The distributed ARP and DHCP servers are neglected by most of the proposed architectures despite having an important role to play in real networks [33] and they can be used to enhance scalability.

1.3 Aim and Objectives

The overall aim for this thesis is to introduce mechanisms, architectures and protocols that enhance the scalability of network, whilst at the same time taking into account improving other network factors (e.g. reliability, security, efficiency, performance and complexity) in intra and inter domains. This aim is addressed through the pursuit of the following objectives:

1. Investigating in one collision domain (i.e. subnet) how to reduce the response time even with overloaded customer traffic, reducing overhead on control and data planes during operation time and bootstrap time as well as reducing CPU usage of network components in order to enhance subnet scalability;
2. Discovering in one network (i.e. containing multi-subnets) how to reduce the response time during load, reduce overhead on control and data planes, reduce the percentage of packet loss during heavy load and reduce discovery and rediscovery time for any new network's events;
3. Investigating intra and inter domains (e.g. a campus network and a core network) regarding how to reduce discovery and rediscovery time for any event in these domains even with very high load (millions of items of traffic) as well as reducing overhead resulting from discovery messages;

1.4 Challenges

The following challenges had to be met during the implementation of the proposed models:

- Building a testbed containing SDN technology from scratch is a real challenge requiring much effort to develop SDN components that work with standard protocols as well as solving many technical issues. In addition, regarding the building of a real test bed, about seven days is needed to setup the environment for each experiment. Each of 23 computers needs to be setup, configured, cable connections require changing and then the experiments run. Errors need to be checked for and each result repeated at least five times to be sure it is a correct result, all of which requires much effort by the researcher.
- High and even overloaded traffic must be implemented in a testbed so as to mimic a real future network environment in order to verify the proposed models in this thesis and to test the ability to enhance the scalability of these models. The limited number of computers available in the testbed (i.e. 23 PCs) could be problematic in relation to this objective, because a real network will in all likelihood have more than that number. To address this, the virtualisation principle using virtual machines needs to be used on the host side to reach more than 500 hosts, which is the maximum number in one collision domain. In addition, special programs for generating high rates of packets need to be designed.

1.5 Contributions

There are six main contributions of this thesis, which are as follows:

1. It introduces a Multi-to-One (MTO) mechanism to define the relation between the servers and the SDN architecture proactively and reactively so as to reduce the overhead in the control and data planes as well as reducing CPU usage of network components;
2. It proposes Servers under Software-Defined Network Architectures to Eliminate Discovery Messages in a subnet (SSED) architecture and mechanism for discovery in a subnet and forwards packets between sources and destinations so

as to enhance response time. As a consequence of contributions 4 and 5, the following results in terms of scalability have been achieved:

- SSED with MTO scales the network approximately up to 161 switches when compared to the broadcast mechanism;
- In terms of the number of hosts, SSED with MTO scales up the host number to 2500 hosts when compared to the traditional mechanism in one collision domain;
- In terms of CPU usage, SSED scales up the number of hosts that can be handled to 3,076 with same CPU usage as the legacy mechanism needs to deal with 500 hosts.

In addition, SSED, in order to find balance among the network parameters, (scalability, security, reliability, etc.) offers a unique design for solving different network issues, including IP confliction, security and Head-Of-Line blocking (HOL);

3. It provides an innovative multi-subnets dynamic discovery method by introducing the Multi-subnets Discovery Protocol (MDP), which delivers dynamic fast discovery time in distributed architectures;
4. It introduces Multiple Distributed Subnets using SDN Architecture and Servers to eliminate router devices and the Default Gateway Mechanism (MSSERD) to discover multiple distributed subnets. It also forwards packets between sources and destinations in a way that enhances response time efficiency in Ethernet networks, even with high traffic loads. As a consequence of contributions 6 and 7, the scalability of network in terms of number of subnets, scale up to 52 subnets comparing to traditional mechanisms and architectures when dealing with the same response time;
5. It introduces a multi-subnets/networks dynamic discovery method by developing the Dynamic Discovery Hierarchal Protocol (DHP) that supplies dynamic fast discovery time in distributed-centralised architectures;
6. An Open-Levels Control plane architecture (OLC) is proposed, which is an innovative architecture and mechanism for working in intra and inter domains so as to, thereby providing better scalability. As a consequence of contributions 8 and 9 the following result in term of scalability have been achieved:

- In terms of the number of subnets regarding discovery time the OLC can scale their number by up to 3.2 times when compared to the fully distributed architectures;
- In terms of the number of subnets in relation to rediscovery time, the OLC can scale up the number of subnets to 31 subnets when compared to the fully distributed architectures;
- In terms of the number of subnets regarding the number of packets in the data plane, OLC can scale up the number of subnets to 82 subnets when compared to the fully distributed architectures.

1.6 Thesis Scope and Limitations

This thesis involves drawing on aspects of different paradigms that are at the forefront of contemporary research, namely, SDN, virtualization, wired communication and network architecture, which are wide ranging. Consequently, it was essential to clarify the scope of this thesis in order for the objectives and main goal to be met within the time line.

Firstly, in the SDN area, this study involves using the main principles of its technology, which concern decoupling the control plane from the data plane and proactive behaviour in order to scale up the network. Secondly, in term of virtualisation [34], this principle is deployed on the host side with the aim of growing the number of hosts using a limited number of PCs to test and verify the ability of proposed systems with very high network traffic. Thirdly, regarding the network architecture field, this study investigates the legacy architectures, concepts, mechanisms and protocols in order to discover the main reasons inhibit scaling up of a network. Then, systems (i.e. mechanisms, architectures and protocols) for overcoming these are proposed. Finally, regarding the wired communication field, its powerful features, especially Ethernet wired technology cannot be neglected and hence, in this study is used as the backbone of deployed system.

Regarding the limitations of the proposed models, these three-fold, as explained below.

- The proposed models are pure SDN-based models, so they need to be developed to support hybrid network devices (i.e. legacy and SDN devices);

- All the SDN controllers and switches must use LLDP-based protocols to perform the discovery processes;
- IPv6 is the newer version of address system in computer network, which offers a much larger pool of addresses. However, the proposed models have been experimentally tested using IPv4 rather than IPv6, because most users nowadays use the former, with just 20% of them using IPv6 [35]. Nevertheless, theoretically there is no conflict in deploying IPv6 with the proposed models.

1.7 The Thesis Structure

This thesis is organized into seven chapters, each starting with a brief introduction providing an overview of the main contributions in that chapter. At the end of each chapter a brief summary is given.

Chapter 2: This chapter presents the background of the technologies that could be used to provide a scalable network environment. In addition, it defines the scalability term and explains how this can be measured.

Chapter 3: This chapter contains discussion on the three main causes of restrictions to network up scaling. Subsequently, previous works that have set out to improve network scalability and overcome legacy network system limitations are reviewed. Finally, the relevant gaps these other studies' have failed to address are presented in the summary.

Chapter 4: This chapter starts with analysis of the current network discovery mechanism inside a subnet, with mathematical formulae being generated for that mechanism. Then, the MTO mechanism is introduced to define the relation between the servers and the SDN architecture proactively and reactively so as to reduce overhead in the control and data planes as well as reducing CPU usage of network components. Subsequently, SSED architecture and mechanism are proposed for discovering subnet and forwarding packets between sources and destinations in a manner that enhances response times. Next, seven scenarios involving extensive experiments with a testbed of 23 PCs are built to verify the effectiveness of the proposed model.

Chapter 5: This chapter reviews the current router and default gateway mechanisms, also identifying the limitations of both. Next, mathematical formulae are developed to evaluate, theoretically, some legacy statistics relating to the use of middlebox devices and the default gateway mechanism. Subsequently, the MDP protocol, which provides dynamic fast discovery/rediscovery time in a fully distributed architecture is introduced. Then, the MSSERD architecture and mechanism is proposed for discovering multiple distributed subnets and forwarding packets between sources and destinations in a way that enhances scalability. The testbed is built at the end of this chapter, with five scenarios using 23 PCs and extensive experiments.

Chapter 6: This chapter discusses the traditional distributed control plane architectures with their aggregation mechanism and the limitations regarding this. Several equations are evaluated to for theoretically analyzing this architecture and mechanism. Then, the Dynamic Discovery Hierarchal Protocol (DHP), which provides dynamic fast discovery/rediscovery time in a distributed-centralized architecture is introduced. Subsequently, the OLC model is proposed as an innovative open-level control plane architecture and mechanism for working in intra and inter domains for better scalability. Finally, a large testbed is built using up-to-date hardware and software components to verify the practical efficiency of OLC.

Chapter 7: The study findings and conclusions are summarised in this chapter as well as the impacts of study outcomes being briefly explained. Finally, the recent developments are discussed and suggestions for short and long-term future work are put forward.

Chapter 2

Scalability Fundamentals

2.1 Introduction

This chapter begins by answering the two questions that were advanced in chapter 1 regarding how scalability can be defined and how it can be measured. Then, it provides background knowledge about the network technologies that will be used in order to implement the proposed models in this study so as to scale up the network. These technologies have been chosen for their features and advantages as well as taking into account up-to-date ones.

2.2 Scalability

Generally, Scalability is a claimed feature for systems (i.e. a sought after attribute that is beneficial to system behaviour; it does not refer to the same concept for everyone so there is no consensus regarding its definition [12]. That is, inspired by the term *Isoeffeciency* [36], scalability of a network is *the ability to increase the number of serviced customers without degrading the performance of network*. This means the scalability is not an independent issue and designers need to take care of other network parameters, such as overhead, efficiency, reliability, complexity [36].

When increasing the number of hosts and amount of traffic, this requires optimal use of resources in order to provide an environment that passes traffic fast and accurately. For example, some researchers argue that SDN networks lack scalability owing to increased load that leads to more pressure on the controller, whereby it has to process higher rates of flow per second [37]. However, they overlook the optimal of use of the network resources that could be achieved using SDN technology owing to the powerful

proactive feature that it has, which could significantly enhance scalability without extra overhead in the control plane.

2.2.1 Scalability metrics and measurement

Some researchers consider the control plane in an SDN network as a platform for evaluating scalability, whereby they focus on the power of processing of the controller, the number of requests can it handle per second and flow setup latency as metrics for measuring it. Similarly, some other studies [13]–[15] use the performance of the controllers in relation to load in an SDN network as a tool (i.e. metric) for evaluating scalability. While other researchers consider the data plane as a medium for calculating scalability by evaluating the ratio of flows entering the network through the data plane [38][39]. It is contended here that scalability should be measured according to the performance of the whole network (i.e. not focusing on a single part such as the SDN controller and ignoring other parts) and it should involve taking into account both the control plane and the data plane. The contention is that two metrics, namely, the response time and availability of network resources, e.g. bandwidth, are the most important. Accordingly, in this study, the focus is on enhancing the response time and the availability of links' bandwidth and use these as metrics to verify the enhancements in network scalability. The measurement for these metrics must be performed whilst increasing in the amount of traffic (i.e. load) and the number of end devices. In addition, other parameters of the network, such as complexity, reliability, flexibility and so on, need to be measured concurrently, in order to get a balance between solving the scalability and other network issues.

2.3 SDN technology

It is a promising technology that has not yet reached a standardised form and hence, needs more study. It has been created for several reasons:

- Meeting the increase in the demand for traffic owing to the appearance of different new technologies and Internet applications, such as IoT [1][40];
- Because of the limitations of current network architectures [12];

- Introducing standardised hardware boxes that are manipulated by separated software in order to provide more interoperability and flexibility as well as minimising the cost of hardware [1].

This technology has several advantages, such as:

- It can be used to ease modifying all network features programmatically [12], which gives more flexibility, less complexity in configuring and managing the network and less effort from the administrators;
- It provides dynamic behaviour by supporting proactive and reactive modes that are available through the general view of the whole network, which is one of the powerful features of SDN technology. This behaviour provides enhanced use of resources, management flows among data plane devices [41][12], migrating Virtual Machines (VM), proactive updating of the routing tables by detecting network event changes [41] and recalculated link cost (i.e. weight);
- It allows for centralise system features, such as the administrator's centralised monitoring;
- It simplifies network design [1], which lead to less effort being required by the designers;
- It simplifies troubleshooting, which leads to reduce operational costs [1].

SDN technology works by disaggregating the control plane from data plane, which results in network resources being used in a more effective way. It is not like traditional network architecture, where switches and routers forward incoming packets depending on one or a few parameters in the packet headers (e.g. IP address, MAC address, etc.), for with SDN more attributes can be compared in order to make matching rules prior to forward packets [12]. It uses SDN protocols, such as the OpenFlow protocol, for this purpose. Figure 2.1 shows the differences between the legacy and SDN architectures.

2.3.1 OpenFlow protocol

It was proposed by Stanford University [42] as the first and only SDN standard protocol [1] for linking the control plane with the data plane in SDN networks. Many service providers, such as AT&T and Google as well as many networking venders [1],

use it. It has the advantages of ease of use, many parameters for deployment in matching rules and it has many flexibility management features. The OpenFlow protocol has multiple versions (1.0, 1.1, 1.2, 1.3, 1.4 and 1.5).

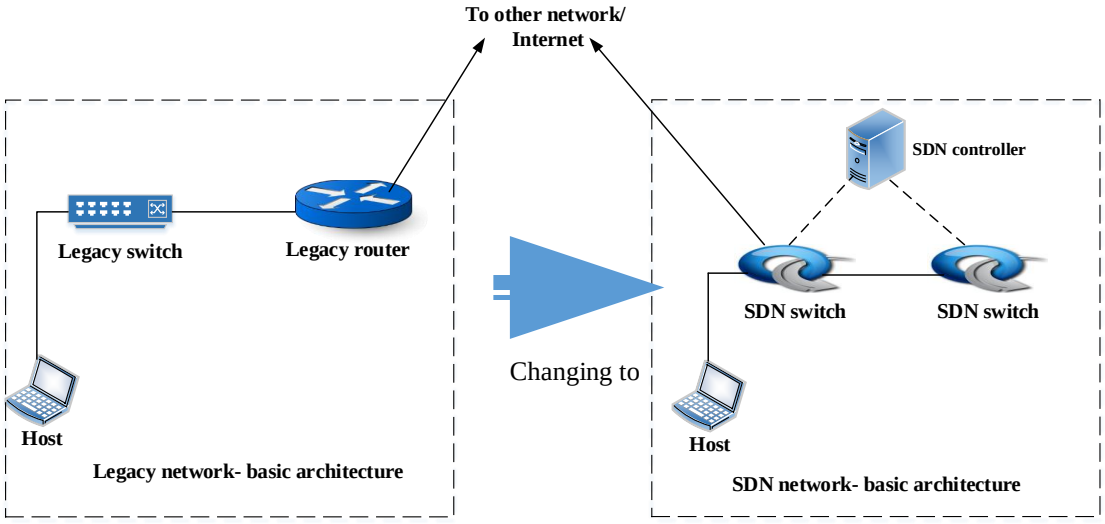


Figure 2.1: Legacy and SDN basic architectures

2.4 Ethernet

Ethernet is a widespread technology used in local and metropolitan zones, where Local Area Networks (LANs) and Metropolitan Area Networks (MANs) are deployed, because of its low-price, high speed, centralised administration and sharing of peripheral device features. In more detail, the Ethernet protocol is a link layer protocol used as a backbone for different types of network, such as home, campus and enterprise networks. Its popularity came from providing services for its own layer and all the upper layers' protocols as well as its simplicity [43]. Ethernet devices have one or more Ethernet ports, which have been pre-set by the manufacturer to have unique MAC addresses that each have 48 bit pairs (e.g. 00:11:23:45:5A:22) containing numbers and letters and it can be scaled up well to cover a new product. In addition, Ethernet is used perform the main functions of a network automatically [43], such as the plug and play function. Moreover, it has a high bit rate of 100 Gigabyte per second (Gbit/sec) and the anticipated rate in the future is 400 Gbit/sec [44]. Finally, Ethernet switches are fast network switches with low cost [45][46].

As a consequence of the Ethernet features, it has been deployed widely, being used in many architectures, including wired data centre networks and High Performance Competing (HPC). More than 40% of super computers apply it [47] and in space communication it is deployed to produce images and information because of its reliability [46].

2.5 Network address translation (NAT)

The increase in end system devices led to an increase in demand for IPv4 addresses, which have been playing an important role in the Internet and network backbone even after the appearance of IPv6 and they can be classified into two types. Firstly, there are public IP addresses, which are used at the edge of networks, such as an enterprise network, where each network can have one or several public IP addresses, depending on the number of hosts that need to connect to the Internet. This type of IP is assigned by the Internet Assigned Numbers Authority (IANA) [48]. Secondly, there are private IP addresses, which are used for Internal purposes, such as inside one company. Packets with private IP addresses are not suitable for forwarding directly to the Internet. In order to send a packet outside the network, mapping from the private to the public address has to be perform and inserted into the packet by the edge device (e.g. router) by using the NAT map mechanism. NAT maps multiple private IP addresses to a single public IP address and it can be used to map private to private or public to public addresses [48]–[50].

Using the above technologies the testbed is built using 23 PCs, as can be seen in Figure 2.2, twenty of which have the specifications of core 2 Quad, 2.66 GHz, 2.9 GiB memory and an Ubuntu 14.04 operation system. Of the remaining three PCs, one is with the specifications of core i7, 3.40 GHz, 3.8 GiB memory and an Ubuntu 14.04 operation system. The final two computers are Samsung laptops, both having the specifications of core i7, 2.20 GHz, 7.8 GiB memory and an Ubuntu 14.04 operating system. A Ryu SDN controller is used as the network operating system, that provides tools and libraries for design SDN components, which was written using the Python language for fast, easy and community supportive development.

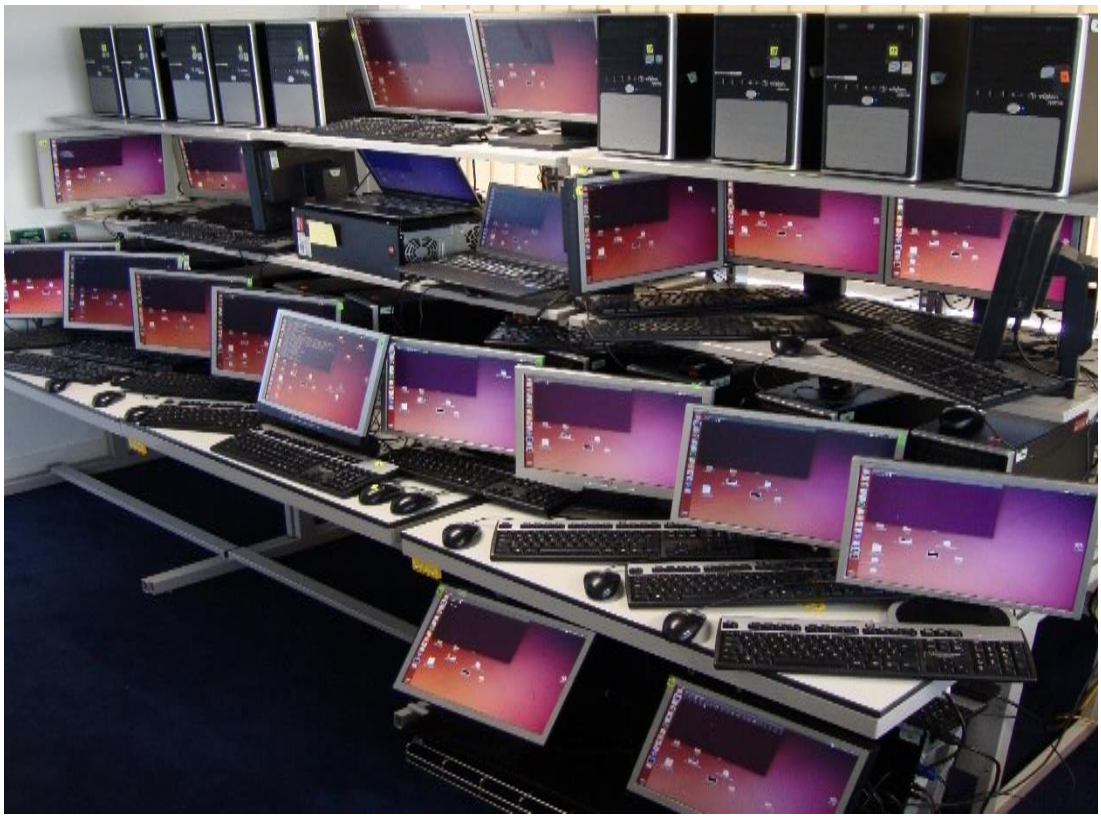


Figure 2.2: SDN Testbed environment with 23 computers

2.6 Summary

The scalability term has been defined in this chapter as *stable system performance during load (e.g. traffic/end-devices)*, with the response time and ratio of packets in the control and data planes being proposed as metrics for measuring it. In addition, SDN, Ethernet and NAT technology concepts, which represent the backbone for the models in this study, have been described and their advantages have been stated.

Chapter 3

Literature Review

3.1 Introduction

Having defined and explained the importance of scalability for networks and as well as considering how it could be measured, this chapter will discuss the three main reasons why legacy architecture hinders scaling up. Subsequently, the limitations arising from each feature are discussed in detail with practical evidence. The recent salient studies that have been undertaken to improve network scalability are reviewed, listing their advantage(s) and disadvantage(s), Finally, the gaps pertaining to all of these studies are summed up. The extant literature that has attempted to improve scalability can be classified into three types according to the different aspects focused upon.

3.2 Broadcast discovery mechanism in one domain (e.g. subnet)

This is the first reason that prevents a one collision domain from scaling up. Despite broadcast and multicast protocols having the advantage of providing different services, such as getting destination MAC addresses, obtaining new IPs, loop free networking and discovering neighbouring nodes, the compulsory broadcast mechanism has resulted in multiple negative consequences.

3.2.1 The weaknesses owing to the broadcast discovery mechanism

The negative impact from broadcasting is five-fold, as explained below.

- As a consequence of broadcast packets, broadcast storms can happen in network topologies with multi-levels of connection, such as tree topology, causing further

problems, such as congesting links, overloading the controller's/switch's CPU (it was observed that experimentally the CPUs overloaded 100 % during approximately 0.5 sec of a storm) and generating MAC address flaps. The STP protocol in legacy networks is used to overcome the loop storm, however, it has the limitation of generating multicast traffic, which consumes bandwidth and supports only seven hops as a maximum bridged LAN diameter [27], thus restricting the network to scale. Practically, when increasing this to more than seven, the bridging loop problem takes down the whole the network as happened in [51], one of the worst IT crises in history.

- The broadcast mechanism leads to leaks in security, such as when the Address Resolution Protocol (ARP) is used for different types of attacks, such as broadcast attacks, poisoning, spoofing and flooding, which result in the network being completely stopped or make resources unavailable, such as through a Denial of Service (DoS) flooding attack. In addition, sniffing can occur, whereby broadcast packets reach all of the hosts, even if they did not make a request, which can lead to data being intercepted by unauthorised hosts.
- Broadcasting leads to increased network traffic resulting in collision and competition at the same link, which leads to loss of packets, congestion and negative impact on response time. Hence, Cisco recommends in [10], which is a practical study, using no more than 500 devices in one collision domain. However, this limit to the Ethernet network means it cannot meet the needs of recent technology, such as the Internet of Things (IoT). In addition, the back-off algorithm used in collision networks to solve collisions, theoretically leads to the number of hosts being limited to 1,024 [26], which raises the scalability problem in these networks.
- The broadcast mechanism leads to increased CPU usage by the hosts, where practically hosts capture many broadcast packets which are irrelevant to them [22].
- All of the above limitations become worse, if the number of requests per second per workstation increases.

In sum, the above issues have resulted in Ethernet-based network being subject to lack of scalability [9], security leaks and limited reliability.

3.2.2 Related work aimed at handling broadcast limitations

To solve the challenges that are related to the broadcast discovery mechanism, different architectures and techniques have been proposed for scaling Ethernet networks by other researchers that can be classified into two groups, according to the architecture of the different types of switches that they use.

3.2.2.1 Related work with legacy switches

With this group, the proposed architectures use legacy switch architecture in their design to solve broadcast issues. In SEATTLE [52], the scalability is enhanced by using short path routing and a hash table, whilst no configuration (plug and play) is required, because it uses flat addressing. However, it adds another software program to the switch to perform some functions that are not supported by all types of switches, which leads to a lack of backward compatibility. Moreover, this switch requires an increase in the cache size when the number of hosts increases, for it has the responsibility of storing their additional information and also, it costs more than a traditional switch. In EtherProxy [53], a new device is introduced and inserted into the network, which partially stops broadcasting, but still causes a delay in response times owing to sniffing and analysis of each packet so as get its information. In addition, it uses distributed multiple EtherProxy devices that can find it difficult to synchronize the resolution tables among them. Moreover, a failing EtherProxy device can lead to isolation of all the switches and hosts behind it from the operation. In contrast, SAL [54] uses a distributed database in which the edge bridges store the host information. It does not completely eliminate the broadcast, because it still uses it among these edge bridges to retrieve the destination information and if a failure occurs in them all the switches and hosts will become isolated.

3.2.2.2 Related work with the SDN switches concept

These proposals use SDN concept architecture in their design to eliminate or reduce the effect of broadcasted protocols. In Portland [55], a dedicated centralised fabric manager device is introduced that contains information about the network. It adds another MAC format that is Pseudo MAC (PMAC) to encode the hosts' position in the topology. The

PMAC packet is forwarded to the Portland switch that contains special software for converting it back to Actual MAC (AMAC), which increases the complexity of the network and the processing time in the switch. The other disadvantage of this method is that it cannot work with other switches, because the Portland switch has specific features, such as announcements to itself periodically. In Po-Wen, et al.'s framework [22] and the CPA framework [16] an ARP proxy in the Ethernet network is proposed as a module inside the SDN controller for handling ARP packets. It forwards every broadcast ARP request to the controller plane and generates an ARP reply message, which it sends back to the requested host. In addition, the former framework builds the Dynamic Host Configuration Protocol (DHCP) function inside the SDN controller to deal with the DHCP broadcast packets. In FSDM [17], an ARP proxy and DHCP relay is introduced inside the SDN controller to handle ARP and DHCP broadcasted packets. The DHCP relay logically links the hosts and the DHCP server, which leads to an increase in the number of packets the controller deals with, thus increasing the overhead on it. In SDARP[56], a Software Defined Address Resolution Proxy is proposed as a centralised ARP application inside the controller to suppress broadcast messages by centrally answering ARP messages. It differs than other proxy models in using only the ARP messages to build the ARP table inside the controller. However, it still broadcasts ARP messages, if the ARP table does not have the information regarding the destination host.

For all the last three proposals (FSDM, CPA , the Po-Wen framework and SDARP), proxy techniques are used inside the SDN controller, which has several disadvantages, such as lack of scalability in large networks at peak load due to increased request rates, resolution updates and mobility. This results in greater latency and response times, which get increasingly worse over time. In addition, there are controller overhead issues, fault-tolerance issues and single point of failure problems. Moreover, this increases the probability of attacks and as a consequence raises security issues, which is considered the most important problem nowadays, as reported by the SDN community [56].

3.2.2.3 Virtual LAN (VLAN)

VLAN is a broadcast domain, which contains a group of end-devices that logically have the same requirements regardless of their location in the network. As a consequence, in order to communicate VLANs layer3 devices must be used [57]. VLAN has the benefit

of minimising the size of each domain by increasing the number of domains (i.e. same as the router function). However, the number of end devices still cannot exceed more than 500 hosts in a one collision domain.

3.3 Middlebox devices between subnets

In order to build a large network, such as a Campus Area Network (CAN) or a Metropolitan Area Network (MAN) a LAN is interconnected with other LANS using middlebox devices, e.g. (router/default gateway). A router is a complex system [58] which can be a software or hardware device. It forwards packets at least between two subnets using Internet Protocol (IP), as directed by the routing table and the IP packet header control information . It works on behalf of IP-based hosts in its subnet, when they need to connect outside their subnets/networks, which it does either by a static default gateway or proxy ARP [57] settings. As a consequence, using a router as a middlebox device results in several limitations.

3.3.1 Different limitations owing to using a router device

The router and the default gateway mechanism negative consequences are as follows.

- Configuration limitation on the host side. A network with a router currently needs to use a default gateway setting, which means that every host in the network must add this setting to its routing table before it is allowed to send packets outside its network, otherwise a “Network is Unreachable” message will appear when using the ping command. Default gateway configuration can be performed on the host side either automatically, if it has dynamic IP configuration that lets the host use the information (i.e. gateway IP address) obtained from a Dynamic Host Configuration Protocol (DHCP) server to connect outside its subnet, or it can be set manually by the user. Regarding automatic configuration using DHCP the drawback is that if just a single DHCP server is used and it fails, the new hosts cannot get the IP address of the default gateway, which leads to them being isolated from communicating with other networks. Moreover, if the router that is configured as the default gateway stalls, there is no dynamic way of identifying the new router device and its default gateway IP address [59][60] and hence, all hosts cannot connect to outside the network during that period.

While regarding the manual configuration by the user (the latter method), human error could lead to misconfiguration the default gateway IP address, especially if multiple routers use in the same subnet, and also manual configuration takes up user time. As consequence of this limitation, the network is complex as well as lacking flexibility and reliability.

- Regarding the inner edge of the router side, each router, when connected to the network for the first time, needs to be manually configured by the administrator in terms of different settings, such as internal and external IPs, buffer sizes and Quality of Service (QoS) that are a time cost. Moreover, this configuration is prone to mistakes owing to human error. For example, different venders have different default settings for the router's internal IP, such as Apple routers having 10.0.1.1, whilst Netgears' routers have 192.168.0.1 or 192.168.0.227 [61]. This can make it confusing for the administrator, if the decision is to use the default setting, as there needs to be consistency with DHCP pool IPs. In addition, adding additional services, such as IP telephony or a video service (e.g. IPTV), leads to more complicated configuration, which also makes the network more complex.
- If the router is set up as a proxy ARP, this means it offers its MAC address when receiving ARP requests from clients that are making it work on behalf of its subnet, which leads to every device in the subnet having to make ARP resolution to the router before connecting to outside its subnet. This, in turn, leads to scalability limitation especially in relation to broadcasting issues. In addition, using a router as an ARP proxy can reduce reliability, because it does not have a fall back mechanism and masquerading in some situations could be confusing [62]. Moreover, if multiple routers are deployed in the same subnet, problems such as MAC flapping can occur [60].
- Another issue related to configuring a router as a proxy ARP is the security issue, whereby misconfiguration leads to a DoS attack. In such cases, black holes might happen as a consequence of the router not having the ability to correctly forward received packets to their destination hosts and hence, dropping them [63].
- Regarding outside of the router, if we need to make the network bigger by adding another subnet, then the outer IPs for both routers must be in the same subnet (i.e same Ethernet segment) in order to reach each other using an Ethernet connection. For example, a router with outer IP 172.168.0.100 cannot connect to another

router with outer IP 172.166.0.99 as they are in a different subnet. In addition, if the router is connected to multiple routers using multiple ports then the administrator needs to be careful to perform the configuration correctly otherwise the packets will be dropped at the router. This restricts the administrator, because if any change is required to the outer IP of one of the routers this leads to a sequence of modification for all affected routers in the same network. This leads to effort and time costs for the administrator and greater complexity especially when configuring a large network [64].

- If a subnet has one router and a fault happens to it, then there will be no connection to another subnet until it is changed for another, which raises a reliability issue.
- If a network contains Virtual LANs (VLANs) or if the subnets in the same network connect in a topology that, as a result, could take the form of a loop topology, such as mesh topology, then the Spanning Tree Protocol (STP) [27] should be activated in the routers to make a loop free network, such as in Cisco routers [65], especially if static routing is used. In turn, using STP leads to restricting the network to seven hops as the maximum diameter, thereby limiting scalability [27].
- Routers when routing packets, modify the layer two header in the Ethernet frame at each hop, which increases latency.
- Increasing features and the ability of the router lead to it being more expensive [66].
- A growing amount of traffic puts more pressure on a router, which increases the likelihood of a single point of failure owing to congestion [11], which will affect the network performance.

The above limitations result in complexity, along with lack of reliability, flexibility, performance and scalability, that restrict the development of Ethernet-based network, especially when building large networks.

3.3.2 Related work handling the scalability issue by focusing on the router device concept

To solve the above issues that appear with multiple subnets different mechanisms and architecture have been introduced. The authors in [18] have proposed a model to enhance the network's flexibility, reliability and scalability by connecting different networks using

a virtual router under SDN architecture by combining the network functions virtualisation (NFV) concept with Multiple Protocols Labels Switching (MPLS). Similarly, in [19], it is claimed that a virtual router in SDN will allow for automatic topology discovery for transport networks for better performance. This is achieved through router virtualisation in the core network, while the edge network still uses the router device. Both these studies merely introduced the concepts that would involve changing the hardware router with software router one, but the same limitations cited above would be inherited in their models. In [67] a static virtual router is applied to an SDN controller in order to enhance the round trip time, but this has a scalability issue as it can only connect a limited number of subnets. In [24], different types of routers (i.e. MPLS, hybrid, open flow and general routers) used in the same network with a single SDN controller are proposed in order to perform tunnels splicing system, which improves network scalability. However, in this model, using one controller to control all those routers in same network, leads to single point of failure issues and deploying different types of routers in same network results in the limitations mentioned above.

In a different context, in [68] the delay issue which affects network performance owing to congestion in the edge links in campus networks during load time, is solved by using routers with caching ability in an SDN network. However, this solution increases the router costs. Moreover, all the above models must use the default gateway settings on the hosts side to connect to their network, which leads to several restrictions owing to default gateway limitations.

In CPA [16], the authors claim to minimize ARP broadcast in a data center network by using an ARP proxy as a module inside the SDN controller. It forwards every broadcast ARP request to the controller plane and generates an ARP reply message, which it sends back to the requested host. However, this work is different to our proposed model as it is limited to being applied inside a single subnet, which means CPA still needs to use L3_devices (e.g. routers) to connect multiple subnets. In addition, proxy techniques are used inside the SDN controller, which has several disadvantages, such as lack of scalability in large networks at peak load due to increased request rates, resolution updates and mobility. Moreover, there are controller overhead issues, fault-tolerance issues and single point of failure problems. Furthermore, it increases the probability of attacks and as a consequence, raises security issues.

The model in [69] is identical to the three tier Cisco architecture and is claimed to enhance network scalability, however, it has several drawbacks. Firstly, it uses two

controllers in the whole network to perform different jobs, which raise consistency issues between them. In addition, the overhead on the controllers will increase when scaling up the network. Secondly, using in-band control in all SDN switches leads to different security and reliability issues, which will increase with load on the network owing to control packets using the same plane as the data packets. Thirdly, one of the controllers in the core network works an ARP proxy server that replies to all ARP requests, which leads to a high response time. Fourthly, using a centralized database leads to a possible single point of failure problem. Finally, the model modifies MAC and IP addresses in the packet frame in each layer of switches, which is similar to the router function that leads to all the router limitations that mentioned above.

3.4 Distributed architectures and their mechanisms

It is hard to extend the traditional fully distributed architecture and distributed aggregation mechanism to a large scale, because they suffer several drawbacks by using the data plane as a bus to transfer the control discovery messages, which increases the traffic on that plane. The SDN appears to overcome the traditional architecture issues by decoupling the control plane from the data plane to give more flexibility [70]. However, the standard SDN paradigm contains one controller in each network [71], which raises other issues, such as difficulties in the scalability of large networks and potential single point of failure [72]. Consequently, using multiple controllers and distributing them properly at locations in SDN architecture is an essential parameter for scaling the network [73]. Various solutions have been proposed to overcome the scalability issue since the arrival of SDN, which can be categorised according to the control plane architecture, as follows.

3.4.1 Fully distributed control plane architecture

When designing a network that covers distributed areas, it has to be divided into multi-subnets/networks, with each having its own SDN controller. In addition, to scale a network with high performance, the proactive behaviour that is a powerful feature of SDN should be used, i.e. to install rules proactively along paths between sources and destinations, regardless of whether they are in the same subnet/network or belong to different ones. The proactive behaviour of SDN relies on providing a general view of the

network to each subnet/network in order to find and install routes in the routing tables between the edge devices (e.g. routers). This general view in the distributed architecture can be obtained by using a well-known discovery mechanism, i.e. a distributed discovery aggregation mechanism [20]. The Open Shortest Path First (OSPF) [74] traditional protocol is the most commonly used for this purpose for fully distributed architecture in intra-domain among subnets/networks within the same Autonomous System (AS), such as in [20].

However, there are several limitations as a consequence of using the aggregation discovery mechanism in distributed architecture, which are as follows.

- The aggregation discovery mechanism by distributing the discovery information to all subnets leads to the use one or multiple protocols to implement this mechanism, such as in the Disco model [23], where Messenger-Link Layer Discovery Protocol (M-LLDP) and Advanced Message Queuing Protocol (AMQP) are deployed to discover the network. As a consequence, this leads to an increase in the complexity of the controllers and more latency when performing the discovery. In addition, as these protocols must work synchronously and they need manual configuration, this increases the probability that the whole system will fail due to human error.
- With such an aggregation discovery mechanism, the data plane is used to transfer the discovery packets through the network, which results in more load and the consumption of the resources of that plane, which consequently has an effect on the discovery and convergence time.
- During peak load, the probability of failing in the discovery process for a new event (e.g. add/delete subnets) increases, because both customer data and the control discovery signal use the same plane (i.e. data plane), which can lead to congestion in the network. Consequently, the fully distributed discovery mechanism could lead to reliability issues [75], so the best discovery time with the optimum discovery path should have little or even no congestion [76].
- Aggregation of the distributed mechanism results in complexity [76], which in turn increases the latency of the discovery process, given that a number of phases (i.e. rounds) are needed to complete the whole discovery process.
- The size of discovery packets has a direct relationship with the number of subnets/networks [29], whereby the former increase when the best path becomes

longer between the furthest edges (i.e. subnets) of the networks. This will be conflict with the size of the Maximum Transmission Unit (MTU) of a link that passes the discovery messages. This, in turn, leads to performing message fragmentation that is used in cases when the MTU size in less than the protocol data unit [77]. Dividing the discovery message into pieces and send them individually on the data plane leads to an increase in the probability collisions and competition, which in turn lengthens the discovery time. In addition, the limited number of available fragmentations [e.g. Intermediate System to Intermediate System (IS-IS) routing protocol which is limited to 256 fragments] leads to the inability of up scaling for large networks [77].

Relying on fully distributed control plane architecture, the Onix model [20] is proposed for enhancing the scalability of a network. However, it is not efficient for one with rapid changes in its conditions and states, whilst it also has the above limitations. Moreover, it uses the OSPF protocol to make the discoveries in intra-domains with no information about how this could support proactive SDN behaviour. In addition, it is not sufficient for discovering inter-domains and hence, it has to rely on other models [20].

3.4.2 Distributed architecture with a logically centralized control plane

With this architecture, the controllers are each allocated to a single subnet as with the fully distributed architecture; however, a new top layer is defined. Firstly, this layer in some proposals, such as in [21] and [78], is used as a data store in order to be the link among the subnets' controllers and in [21] each controller can be used to control all of the network. Nevertheless, this architecture also has drawbacks, as each controller in each specified time will retrieve the full data from the data store, which will result in each having an increased cache size. In addition, each controller will increase its CPU usage and power consumption owing to it having to perform the best path calculation for the whole network. Secondly, in [21] the top layer is used also as a control channel to make connection to transfer commands between the controllers that return the architecture to the single point of failure, increase the complexity of the system and increase the response time. Thirdly, other studies, such as in [79] use the top layer as a root controller that connects directly to the local controllers, which are used as switches proxies for it. In this

architecture, a specific protocol needs to be designed to connect the local controllers to the root controller, which increase number of protocols that are used for discovery and hence, the synchronization among these protocols could affect the general view consistency. In addition, there is complexity in the root controller as its role is not just the discovery process, for it also has to answer the outgoing requests from that subnet/network. In more detail, the outgoing requests from the subnet pass from the local switch to the root controller, which installs rules in all local switches along the path to the edge device. This leads to increased response time as well as overhead for the root controller. As with [79], [71] uses a coordinate controller in the top layer, with one controller for each domain, thereby limiting the scalability. In addition, it uses unified restful API between the local controllers and coordinating controller, which leads to a backwards compatibility problem as well as increased network complexity. However, there is no mechanism regarding how to discover domains and how the local controllers gather the information. Moreover, the calculation for the global path occurs in the top controller after it receives a request (i.e. not in proactive manner), which means that it neglects the most powerful feature of an SDN. Furthermore, [71] results in an increase in the size of the cache in each local controller, because all the input and output ports of each domain have to be stored in them.

3.5 Summary

This chapter has presented the main causes of restrictions to scalability, with the issue being dealt with starting from one subnet to multi-subnets and then from intra domain to inter domain perspectives. Different studies have been analysed and the gaps in them are highlighted as follows:

- Generally, there is no consideration of the peak load in these studies in terms of its effect on scalability;
- Generally, there is no complete architecture combined with a complete mechanism to work for all sizes of networks (i.e. inside the subnet, in multi-subnet and in intra and inter domain;
- There is significant overhead on the control and data planes;
- Using different discovery protocols to discovery intra-domain (one subnet and multi-subnets) and inter-domain has led to several drawbacks;

- Generally, the powerful proactive feature of SDN has been neglected.
- Generally, using servers in their architectures has been overlooked.
- None of the previous models have been accepted officially or as standard.
- The negative consequences of the default gateway mechanism regarding response time and number of control packets that are generated as consequence of using it, have been completely neglected;
- Most of these studies have not applied the fundamental requirements, which will be covered in chapters 4, 5 and 6.
- Most previous studies have treated the scalability issue separately from other network problems that compromise network efficiency, such as the backward compatibility problem, complexity, security, reliability, flexibility among others, all of which should be taken into account when designing a network [80].

Chapter 4

ARP and DHCP Servers under Software-Defined Network Architectures

4.1 Introduction

The Ethernet is the most popular technology in local area networks that can be found in small geographic zones, such as in the home, on campuses and in enterprise network [81]. It allows for the sharing of resources with high performance, which supports virtualization principles and the client-server scheme in relation to the distribution of load among the servers as well as assisting in administration. The Ethernet protocol resides in the data link layer in the Internet protocol suite, providing services for its own layer and up layer protocols, such as broadcast ones like the Address Resolution Protocol (ARP) [82] in the data link layer and the Dynamic Host Configuration Protocol (DHCP) [83] in the application layer. It also services multicast protocols, such as the Bridge Protocol Data Units (BPDUs) [84], which is a multicast packet used by the Spanning Tree Protocol (STP) [84] in the link layer and the Multicast Listener Discovery (MLD) protocol [85] in the internet layer.

There are effective features of the Ethernet protocol, such as self-configuration for serving the plug and play feature, centralised administration and the use of distributed servers, all of which should be retained when designing SDN-LAN architecture. However, when designing an SDN-LAN network the aim should be to eliminate unwanted features potentially inherited from legacy networks, such as the broadcast and multicast features that lead to increases in the number of hosts in one collision domain,

minimise the number of protocols that are used in one domain over the Ethernet protocol, such as STP and increase the security as well as privacy for users. In addition, the SDN-LAN architecture design should possess some mandatory features so as to retain compatibility with legacy networks' hardware and software, such as keeping the stander protocols code without touch, thereby allowing for a gradual transfer from these networks to SDN. Moreover, no new architecture hardware should be added to the network as this would make it extremely difficult to change from the legacy architecture to the new SDN one. Finally, so as to allow for the continuing use of legacy switches no new software should be added to the host and switch side, otherwise there could be a backward compatibility problem among the hosts.

Hence, so as to satisfy all the aforementioned mandatory features, whilst solving the Ethernet broadcast mechanism challenges mentioned in chapter 3, the Servers under Software-defined network architectures to Eliminate Discovery messages (SSED) is proposed. SSED, firstly, introduces new mechanism for defining the relation between servers and the control plane in SDN architectures. Secondly, it handles broadcast and multicast messages that are generated by the most important type of broadcast protocols in the current Ethernet network paradigm. It eliminates all types of broadcast and multicast messages that could be generate by the broadcast or multicast protocols, such as ARP, DHCP, Network Time Protocol NTP [86], multicast STP BPDUs and multicast MLD protocol as well as any other future broadcast and multicast protocol, through the same SSED concept. In addition, it takes into account the peak load traffic and overhead issues.

4.2 Current system description

In this section, we describe the current broadcast Ethernet system by mathematically analyzing the broadcast phase using the learning switch mechanism and then explain the supported protocol.

4.2.1 Learning switch mechanism

Whilst the broadcast protocols are used for different services, such as obtaining destination MAC addresses or assigning new IP addresses for hosts, it uses the same broadcast mechanism. Generally, there are two different distribution phases to connect

between source and destination hosts or servers. First, there is the broadcast phase from the source to request destination information or a service, and second, there is unicast, which is from the destination to the source to reply with destination information or in response to a requested service. There are some services, such as the DHCP containing more than two phases, for which every packet from source to destination before using the offering IP address is dealt with in a broadcast manner. The learning switch in SDN architecture it has same principles as a legacy switch. For the latter, see Figure 4.1(a), when the switch receives the broadcast packet for different types of broadcasted protocols, which is usually with the destination MAC address 'ff:ff:ff:ff:ff:ff ', it saves the source MAC address in MAC-to-port table so as to prevent broadcasting subsequently, if the switch deals with same address again. Then it forwards the packet to all the other ports in that switch except that which inputted the packet. Then, the next switches use the same mechanism until the requested packet reaches the destination host or server. In case of the ARP service, the host that matches its IP address with the IP address field in the broadcast packet will respond with a unicast message that contains the IP and MAC addresses for the source and destination host. This unicast packet is, firstly, forwarded back to the switch that is connected to the destination host. After that, the switch saves the port and MAC addresses for the source and forward the packet back in a unicast manner to the next switch that is already known, because it has already dealt with it during the first phase and so on. The learning switch in the SDN architecture can be seen in Figure 4.1 (b) and the difference is that the MAC-to-port table is stored in the SDN controller.

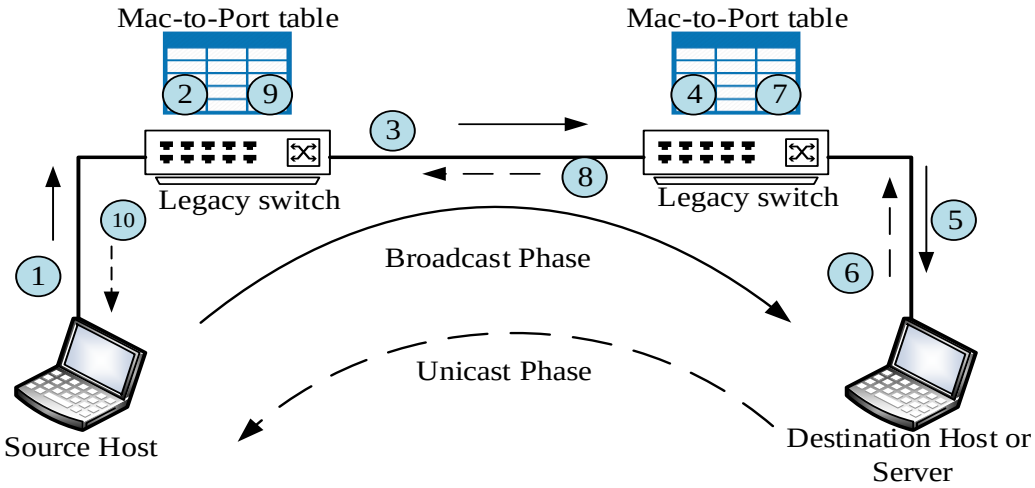
The number of packets in a learning switch network for a broadcast phase in the control plane and data plane in the SDN architecture depends on the number of switches and the number of data links connect to each switch. That is, to reach the destination host or server in broadcast mechanism the number of packets that are generated in control plane during the broadcast phase is a function of the number of switches $F(N_s)$. The number of messages between each switch and the controller is equal to two: one Packet_in message to the controller from the switch and one Packet_out from the controller to the switch to flood the packet to all the output ports, as in (4.1).

$$N_{cp} = N_m * N_s \qquad (4.1)$$

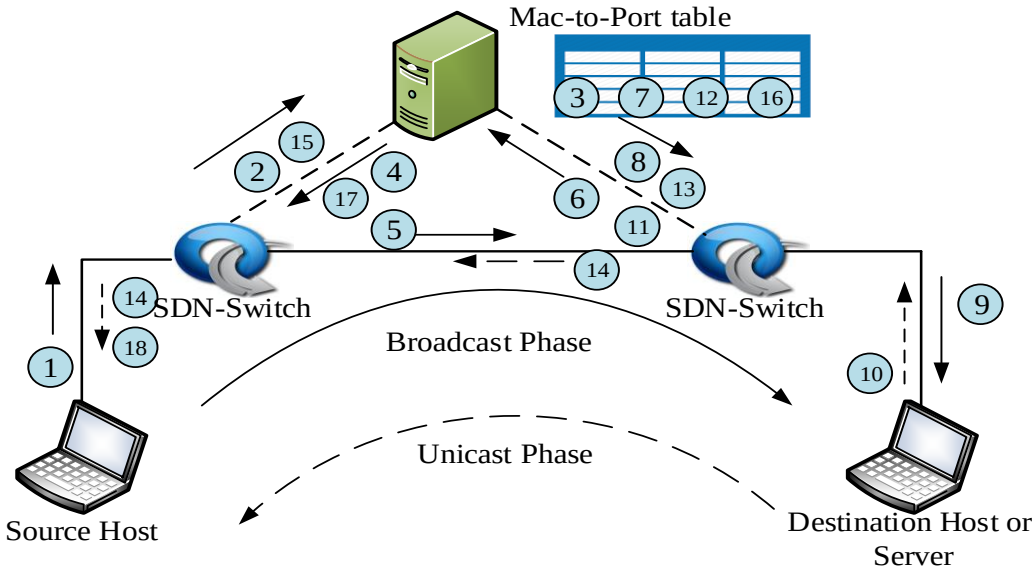
Where, N_{cp} denotes the number of packets that are generated in control plane during the

broadcast phase, N_m is the number of messages between each switch and the controller, whilst N_s is the number of switches.

The number of packets that are generated in the data plane during the broadcast phase is a function of the number of switches and number of links $F(N_s, N_l)$, where one packet is excluded from each calculation, because it represents the input port and hence, is exempted from the flooding. Equation (4.2) represents approximately the number of packets generated in the data plane as a result of one requested broadcast packet from the host.



(a) Legacy switch



(b) Legacy learning SDN switch

Figure 4.1: Steps for filling the MAC-to-Port table and broadcast mechanism using legacy and SDN switches

$$N_{dp} = \sum_{n=1}^{Ns} (N_{I\,n-1}) \quad (4.2)$$

Where, N_{dp} is the number of packets that are generated in the data plane during the broadcast phase, N_s is the number of switches and N_I is the number of data links connected to each switch.

4.2.2 STP and the learning switch mechanism

Practically, the learning switch forward mechanism needs support protocols that let it complete its work without a loop-network issue. Spanning-tree protocols such as STP, Rapid Spanning Tree Protocol (RSTP) [84] are used in spanning loop topologies to prevent a broadcast loop (loop-storm). A loop storm can happen between two switches that have multiple possible paths connecting them. In this case, the STP manages the network logically by ensuring the availability of just one possible path between two switches, which thus prevents a loop storm. The main disadvantage of STP is that it is a multicast protocol and to do its job it has to multicast BPDU packets among switches every 2 seconds [84], which puts more traffic in network and hence, causes delays in response time. In addition, in some cases the whole network can break down if it exceeds seven hops as the maximum bridged LAN diameter, which thus has to be taken into account when designing the network [51].

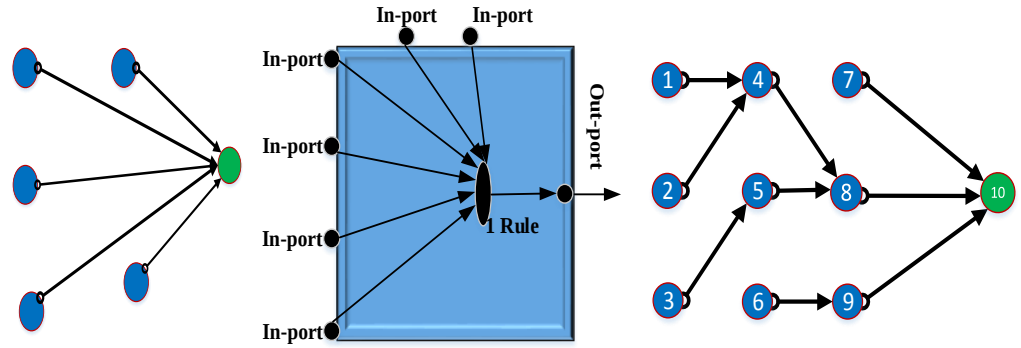
4.3 SSED design

To deal with all the factors discussed in section 4.1 and to overcome the weaknesses in previous architectures, which have explained in chapter 3, the SSED flexible framework has been designed by the introduction of Multi-To-One (MTO) collective service method and this is introduced first. SSED is used to define the relation between the SDN architecture and the servers proactively and reactively as well as eliminating different types of broadcast packet. The flexibility of SSED stems from its ability to forward packets to destinations chosen by the controller (not by the host), which depends on SDN controller management algorithms. This feature gives the SDN architecture flexible behaviours so as to be able to perform different functions, such as load balance, management and packet forwarding. Whilst the focus is on the ARP and DHCP broadcast messages, the same concept will work for all other broadcast messages.

4.3.1 Multi-To-One collective method

A new flexible collective service method that can be deployed in an SDN network is proposed, based on the ability to the controller to have a general view of this network. Its name comes from its job, which is defined as:

Directing multiple nodes that request the same service to a final node that offers the requested service using the unicast concept, in place of the usual broadcast request concept, with just one installed forwarding rule for each service.



(a) MTO general concept

(b) MTO rule connects infinite number of In-port to one Out-port in SDN switch

(c) MTO logically connects multiple source nodes to one destination node using unicast concept

Figure 4.2: MTO concept with different applications in the SDN architecture

Hence, it is called Multi point-To-One point or simply Multi-to-One (MTO). As shown in Figure 4.2 (a), MTO is a group communication and routing methodology, for which a set of nodes (or points) that needs a specific service is routed logically to a single node (or point) independently and in a unicast manner. By so doing, a message can be transmitted from any member in that set to the final single node, independently. The main features of the method are that the source nodes may not be related to each other, there is no limitation for number of nodes between the source nodes and the destination node and source node request service in a unicast manner, rather than through broadcast. MTO can be applied in different applications, for example, in a single SDN switch MTO connects logically multiple input ports from different sources to one output port using a single installed rule in the switch forwarding table, as shown in Figure 4.2 (b). The main advantage is that the switch can deal with N number of hosts without affecting the size of the forwarding table inside the switch, which leads to a decrease in its memory size and

hence, lower cost.

Figure 4.2 (c) shows another example for implementing MTO in multiple switch SDN architecture. Nodes (switches) 1,2,3,6 and 7 can connect logically to node (switch) 10 using MTO collective method with a unicast manner. The main advantage that is added when using the SSED architecture is that the controller can have a view of all the switches in its data plane by using the discovery mechanisms. So, it can forward the packet anywhere in its network, even if the message is not addressed to that destination and does this without modifying the packet fields and as a consequence, this supports the client server concept.

4.3.2 SSED mechanisms

In the current Ethernet network, the broadcast mechanism plays an important role in performing different function, such as looking up MAC addresses associated with destination IP addresses by using the ARP protocol, assigning automatic IP addresses for hosts by using the DHCP protocol, solving duplicate IPs by using the ARP protocol and keeping time synchronous among hosts with NTP broadcast protocol. In this chapter, the broadcast mechanism is replaced by the SSED mechanism, which makes the SDN controller and server share the responsibly of responding to reply messages depending on the type of service that is requested by the host. The server's role is to provide the service and the SDN controller's role is to ensure that it provides the best path to the source host and the resultant reduced management leads to lower overhead in its control plane.

By applying the MTO concept *inside* an SDN switch, SSED forwards packets that need the same service and come from different input ports to one output port. As a consequence, the number of rules that are installed in each switch to reach a specific server equates to one, regardless of the number of source hosts connected to that switch. In addition, the forwarding decision inside the SDN switch will depend on the layer 2 (packets with destination MAC equal to 'ff:ff:ff:ff:ff:ff') and layer 3 protocols that are supported by the switch to distinguish the type of service that is asked for by the host. For instance, the *arp_type* field needs to be able to distinguish an ARP packet and that the service that is needed is hence an ARP one. Moreover, if ports 67 and 68 are inside a UDP packet this can be used to distinguish it from a DHCP service and port 123 in such a packet can distinguish it from an NTP protocol [87]. By applying the MTO concept

outside the SDN switch, the controller in SSED can manage flexibly the requested service from a host to any server that offers it with just one rule in each switch.

The SSED mechanism distributes packets according to packet type directly to a specific server using proactive installed rules, so that the number in data plane in the *Request* phase depends on the number of switches in the best path between the host and the server N_{sb}, while no packets enter into the control plane during this phase. During the *Reply* phase, the data plane also will depend on the N_{sb}, while in the control plane there is one Packet_in message from the server plus a number of Packet_out messages equal to the N_{sb} between the server and the requesting host. Equations (4.3) and (4.4) represent the number of packets generated in the data plane and control plane, respectively, as a result of one requested packet from the host.

$$N_{dp} = 2 * N_{sb} \qquad (4.3)$$

$$N_{cp} = 1 + N_{sb} \qquad (4.4)$$

Where, $N_{sb,}$ N_{dp} and N_{cp} denote the number of switches on the best path between the host and the server, the number of packets generated in the data plane, and the number of packets generated in the control plane, respectively. SSED contains some other mechanisms to perform its role, as follows.

4.3.2.1 Relationship between SDN controller and server's location

With SSED, the server's location is flexible in that it can be in the nearest switch to the controller as a data centre or can connect to any switch distributed in the same subnet. The network administrator can benefit from this flexibility in solving network issues, such as putting the server near to the switch that has heavy active users so as to reduce service access time [88]. In addition, the administrator can spread out the servers in the network to share the load among those offering the same service. This behaviour not only supports servers with broadcast protocols as it can also be used for any type of server in the same subnet.

4.3.2.2 SSED operation modes

In SSED, there are two types of behaviour that the controller uses to manage the connection between the servers and the hosts. Firstly, in bootstrap time, it uses the proactive mode to install rules in every switch so they can reach the Arp and DHCP servers (DHCP and ARP just as example; not limited to these) using the best paths available in that particular moment. If the network contains more than one server offering the same service, the controller can use distributed load algorithms, such as the Round Robin algorithm, to distribute the load amongst them. Secondly, if the network conditions change, such as a new server is added or removed, a switch is added or removed. Moreover, when there is congestion at the links then the reactive mode can be used to redistribute the load among servers or to change paths so as to be the best for reaching the servers.

4.3.2.3 SSED failure handling mode

Since there are different types of failure can be happened in an SDN network, SSED covers most of the important ones and gives solutions.

4.3.2.3.1 Handling switch failure

The route between sources and destinations can be disrupted owing to failure in the routed switch or in its links. The controller in SSED uses a priority feature provided by OpenFlow protocol [89] to install two different rules in the same switch: a high priority rule for the normal route and a low one for failure mode, which can be used if a failure happens in neighbouring switches. In more detail, if a switch fails or the link to one goes down, then all the routes going through that switch will be disrupted. However, SSED deploys a new failure mode mechanism, whereby the controller identifies where the failure has happened by detecting the deleted port from a neighbouring switch and reporting a change in the port status. Then the controller reactively installs a new rule in the neighbouring switch or this switch will employ the second priority rule, proactively installed by the controller, in its forwarding table in order to keep the service offered by the servers working.

4.3.2.3.2 Handling server failure

The controller, to do its management job properly, needs to create tables to use so it can track changes in the network. SSED creates a server-switch table in the controller containing the information about which switches are routed to which servers. For example, switch1 is set to reach server X that offers service Y. So, if failure happens to server X, it is easy to know the switches that direct packets to it and hence, install a new rule in switch1 to redirect packets to another server that also offers service Y.

There is the possibility of a server failing performing any service in any network, not just an SDN, SSED uses an echo message that it sends periodically to each server in the server-information table that is stored in the controller to check if it still alive or not and accordingly, the controller updates the *status* field in this table as required. SSED, after it detects the failed server, will install new rules in the affected switches to route to another server that offers same service, if there is one. Otherwise, the controller forwards it to the default gateway so as to obtain the same service from servers in another network.

4.3.2.3.3 Handling controller failure

There are several previously devised mechanisms that SSED can use to overcome this issue, such as back up with a different SDN controller [90] and using or changing the switches to standalone mode so as to take the responsibility for dealing with packets without an external controller [91].

4.3.2.4 Handling discovery (join, leave and mobility)

One of the most powerful aspects of SDN is its ability to discover its controlled network components in relation to cases of join, leave (normally or in a failure case) or mobility. SSED performs discovery with high accuracy and speed as well as dynamically detecting changes. There are two types of discovery in SSED a switch discovery mechanism and a host (could be a server too) discovery mechanism.

4.3.2.4.1 Switch discovery

SSED generates a switch-information table in the initial setup of the network and puts it in the SDN controller's memory. When a new SDN switch connects with the controller, there is an exchange of negotiate messages between the two, the information from which being used by the controller to register the switch in the switch-information table. Then, the controller starts sending discovery packets using the Link Layer Discovery Protocol (LLDP) [30] to discover the topology (how these switches are connected together) and registers that relation in the same switch-information table as pairs in the hash table. The important difference with SSED from other platforms is that it stops sending LLDP, because of the fact that any port or switch added to the SDN network must be reported to the controller [89]. This use of LLDP leads to a decrease in the excessive number of multicast packets that are usually generated [92]. In addition, if a switch leaves, then SSED performs the same procedure as for failure mode, as discussed above and then, removes the switch from the switch-information table.

4.3.2.4.2 Host discovery

SSED creates host-passport table when first establishing the network and puts it in the SDN controller's memory. If a new host joins the network, then there are two standard possible ways for setting up its IP, manually or dynamically, using the DHCP service. In both cases, static IP or dynamic IP, the host firstly must send an *ARP probe* packet, which is an ARP request packet with the sender IP address equal to all zeroes and the destination IP address equal to checked IP [93]. ARP probes are sent by the host in order to detect if there is a conflict IP with other hosts before commencing to use that IP. Then, the switch forwards that request packet to the SSED controller, which will register in the host-passport table all the requesting host's information, including the IP and MAC addresses provided by the host as well as the port number and switch ID resting with the switch. After this, the controller will forward the host's IP and MAC information to all ARP servers in the local network. This forwarding to all servers is an important step to load balancing among ARP servers that SSED is able to deploy. It is important to note that the host sends another type of packet, an 'ARP Announcement', which is an ARP request packet with the sender's IP address equal to the destination one, if no ARP conflicting

reply message has been received as a result of the generated ARP probe packet [93]. An ARP Announcement is sent by the host to tell the other hosts that it is commencing to use the announced IP and the controller uses that packet to update the host-passport table with the valid IP address.

The host sends an ARP probe and announcement packet each time its IP changes. If the host leaves the network for any reason, such as failure or normal leaving and its departure is of no consequence, then no action is taken. However, SSED will wait for a set time of no activity for that host and will then delete it from the host-passport table, sending update messages to all the ARP servers instructing them to delete it too. The host-passport table contains a field with the name Last Activity Time (LAT) to record the time of the last activity by the host. Otherwise, if the host moves from one switch to another, it sends an announcement message to the controller that leads to the updating of the field *switch ID* just in host-passport table and the host can still use the same IP.

4.3.2.4.3 Server discovery

The controller, when the network is first established, creates a server-information table and any server joining the network sends an announcement message using the UDP protocol to define itself to the controller, which then inserts the server information in the server-information table. The server-information table is a hash table containing the MAC address, IP address (usually static IP), join date, leave date, status, pool range (used for DHCP), type of service, port number and switch ID fields. If a server leaves by planned leave, as stated in the *leave date* field in the server-information table, then prior to this at a set time, the controller will withdraw responsibility from that server and give it to another offering same service, such as ARP, DHCP, and NTP among others. For the controller to do this, the same procedure in proactive mode that we discussed above to install rules in selected switches and forwarding packets to the new server is used. However, if any server leaves unplanned, such as in a failure situation, the controller, firstly, will remove it from the server-information table and will make another server available to provide its service (see subsection 4.3.2.3.2). Finally, in the case of a move of a server from one switch to another, such as from one VM to another, the server will send an announcement message using the UDP protocol to define its new position and which switch it connects with. Then, the controller updates the *switch ID* field (which

records the switch ID that is connected to that server) in the server-information table and server-switch table. The controller then updates the affected switches and instructs them to forward packets to the server's new location.

4.3.2.5 SSED without STP

As SSED stops all broadcast and multicast packets as well as taking responsibility for managing the forwarding of packets from sources to destinations, there is no possibility of a broadcast or multicast storm occurring in a loop network topology. Experimentally we stop STP from working in the SSED architecture in a loop network topology and we check the network to see that it is free from any storm. In contrast, stopping STP is impossible with legacy switches in a loop network topology.

4.3.2.6 Handling other issues related to broadcast

Broadcast packets can be used for different purposes even though they have not been actually designed for them, such as an attack on other components of network (controller, servers, hosts etc.) that lead to several security issues. In addition, broadcast packets can be used to solve duplicate IPs among hosts in same network, but these lead to high consumption of bandwidth and congestion at the links.

4.3.2.6.1 Handling security issues

Security matters have yet to be completely solved in legacy networks, as well as in SDN ones, as has been widely reported [56]. SSED handles security issues by dealing with each leak individually. Firstly, it does not allow any broadcast packet to reach the controller, thereby stopping any flood of ARP *request* or *reply* that is often used by hackers to attack it, stop its work or spoil its tables. In addition, SSED stops any broadcast packets among hosts, because all the packets it forwards are to a specific server depending on the service requested. This avoids ARP cache poisoning inside a host as can happen during a man-in-middle attack, one of the most common forms. Moreover, it results in the avoidance an Arp flood attack to a victim host that could stop it functioning or consume its resources. Furthermore, there is the absence of flooding packets that consume

shared resources in network. SSED also does not allow any Arp reply packet to transfer through the network components except that containing in its IP field the IP for the ARP server as the source server. The IP server is completely transparent to users, which leads to increased security. Taking all these factors into account, these lead to reduced security overhead (high overhead generated as a result of checking each packet) on the controller, because the servers work as filters to it and just pass tested active packets. That is, the server will check the validation of a request then reply, whilst the controller simply undertakes management of it. For example, if host1 is an attacker and sends one or multiple ARP requests to attack the SDN controller, the ARP server will detect it is an invalid request by using a specific algorithm, and the server will not forward this request to the controller. As a consequence, there is reduced probability of an attack on the controller and reduced overhead in the control plane.

4.3.2.6.2 Handling duplicate IPs

Duplicate IPs occur when two clients on the same link use the same IP address concurrently, as a result, problems happen for one or both clients [93] such as over consumption of resources, flood storms and security issues. To solve this issue, usually in legacy networks the host, before using the IP address assigned to it by the DHCP server, broadcasts ARP Probe packets which is an ARP request with the 'sender IP address' field set to all zeroes and the 'destination IP address ‘set to the IP address being probed [93]. The purpose of this is to check whether there is the same IP address available in the network so as to avoid duplicate IPs. To this end, SSED stops any such broadcast by immediately dropping the packet and the duplication problem is solved by a specific code in the controller. This is a major feature of the controller’s management, especially when the network becomes big with many connected hosts. The controller checks the IP field in all the host-passport table rows in case of a new host joining the network or changing its IP address as a result of it awakening from sleep mode, changing the network interface from inactive to active mode and for other cases of change in connectivity [93]. Then, if there is duplication of IPs between at least two hosts, the controller sends an ARP conflicting reply message to at least one of the duplicated hosts, the choice depending on the registered times for that IP and the newer registered hosts will be chosen to change IP(s). Specifically, the conflict message notifies the user through a screen message to

change the IP manually or to activate the sending of a DHCP discovery message to the DHCP server so as to obtain a new IP.

4.3.2.6.3 Handling Head-Of-Line blocking (HOL) phenomena

SSED deals with the head-of-line blocking (HOL) phenomena, which occurs owing to other packets are blocked when the packet at the front of the FIFO queue cannot move. The probability of its occurrence increases with the broadcast mechanism as more packets are generated on the output ports and so more compete to use them. SSED solves this, whereby the controller in SSED has a whole view on its network and installs different rules/paths to the ARP server in each switch with different priority levels. Hence, the first packet in FIFO is no longer waiting if the output port busy as the packet can go through different ports to reach its destination. In addition, SSED by eliminating the broadcast mechanism reduces the probability of HOL happening.

4.4 SSED implementation

We show in detail how SSED implementation handles ARP and DHCP broadcast messages by using a server-based concept with that of MTO. We extend the Ryu controller by adding the three SSED components as follows.

4.4.1 SSED bootstrap, proactive and reactive components

SSED combines proactive and reactive mechanisms, as can be seen in Figure 4.3. Firstly, SSED-bootstrap starts with establishing the network by joining the SSED SDN controller, which directly starts to discover the network under its control. SSED establishes a switch-information table that contains a hash table (e.g. Source switch: [Destination switch, Source port, Destination port and Cost]), which will start being filled when a new switch joins the network. The SSED uses LLDP to discover the network and continues filling the switch-information table. The discovery of the switches process will be continued until a specific time as configured by the administrator so as to let all the available switches join the network. In addition, SSED creates a host-passport table that

contains host discovery information and a server-information table that contains information about each server joining the network.

After finishing the bootstrap time, both the switch-information table and the server-

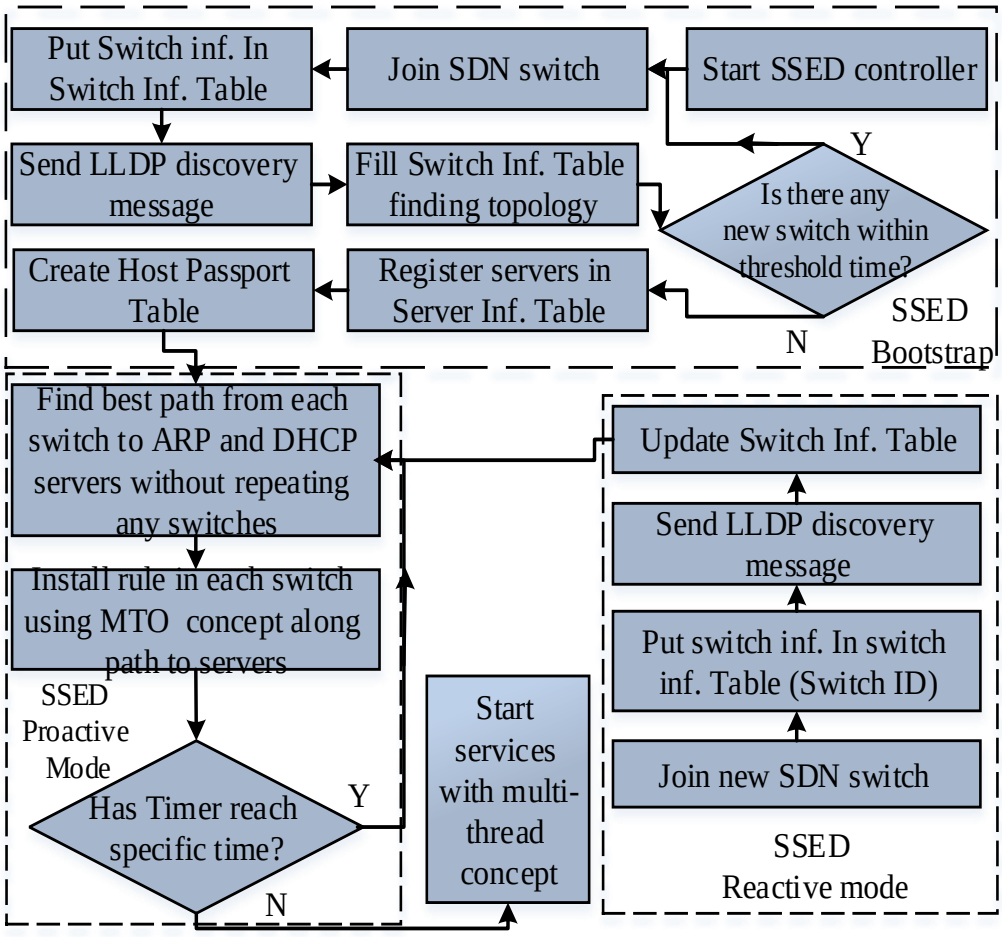


Figure 4.3: A flowchart of the SSED bootstrap, proactive and reactive components

information table will be used in the SSED proactive-mode to find the best path between each SDN switch and the ARP and DHCP servers by using Dijkstra's algorithm, after some development. SSED proactively installs one forwarding rule for each server in each switch so as to let ARP and DHCP broadcast messages be forwarded directly to their respective servers without going through the controller. This mode uses the MTO concept as SSED completely ignores where the broadcast packets are coming from (i.e. from which sources) and just focuses on the output port (the rule is: *do not care about input ports and go to specific output port*). MTO will make one forwarding broadcast rule work with an infinite number of hosts to reach the ARP and DHCP servers that has no effect on switch memory. Proactive mode is repeated every threshold time to deal with any

changes in the switches topology, whilst concurrently the ARP and DHCP components start working in the multi-thread concept. If a new switch joins the network, the SSED reactive mode adds it to the switch-information table and starts to discover how it connects to other switches.

4.4.2 DHCP component

There are two ways to assign an IP address to a host, statically by using manual configuration or dynamically by using the DHCP protocol. With a dynamic IP address, the host sends broadcasted DHCPDISCOVER message to request an address from the DHCP server, which has pool of them to offer. The message will be entered into the nearest SDN switch, which connects to that host and the switch uses the MTO rule, which has already been installed in proactive mode in the bootstrap time, in all the switches along path to the DHCP server so as to forward that message. The DHCP server answers with a DHCPOFFER message, which is a unicast one that contains the host's MAC address in the target MAC address field in an Ethernet packet and this server's MAC address in the source MAC address field. This DHCPOFFER contains an offer of an IP from an IP pool in the DHCP server. The message will go back to the nearest connected SDN switch, which does not have a rule for forwarding and so it sends the message using the OpenFlow protocol as Packet-in to the SSED controller. SSED uses just one packet-in message to complete all stages of the requested service in order to eliminate overhead on the controller, especially during peak load. The controller catches the packet-in message and decapsulates it to get the DHCP information, subsequently checking the type of DHCP packet. Then, the controller checks whether it is a DHCPOFFER packet and the source MAC address to see that it belongs to one of the DHCP servers in the server-information table, for security reasons.

If *NO*, the controller will drop the packet, because it has come from an unauthorised source, if *YES*, the controller will look inside the host-passport table to update the existent host record and reset the expire-host-timer field that is used to check whether the host is still alive. SSED uses the hash function to perform lookup in the host-passport table and considers the IP/MAC field in received packets as the key to finding the host's record. Following this, SSED controller finds the best path back to the host that has made the DHCPDISCOVER, installs rules to prevent next time Packet-in to the controller and then,

forwards DHCPOFFER to the host. The host will generate a DHCPREQUEST message, which is also a broadcast message and sends it to the nearest switch that then forwards it to the server without notifying the controller with a packet-in, as the MTO rule already exists in the switch. The DHCP server will receive a DHCPREQUEST and then sends

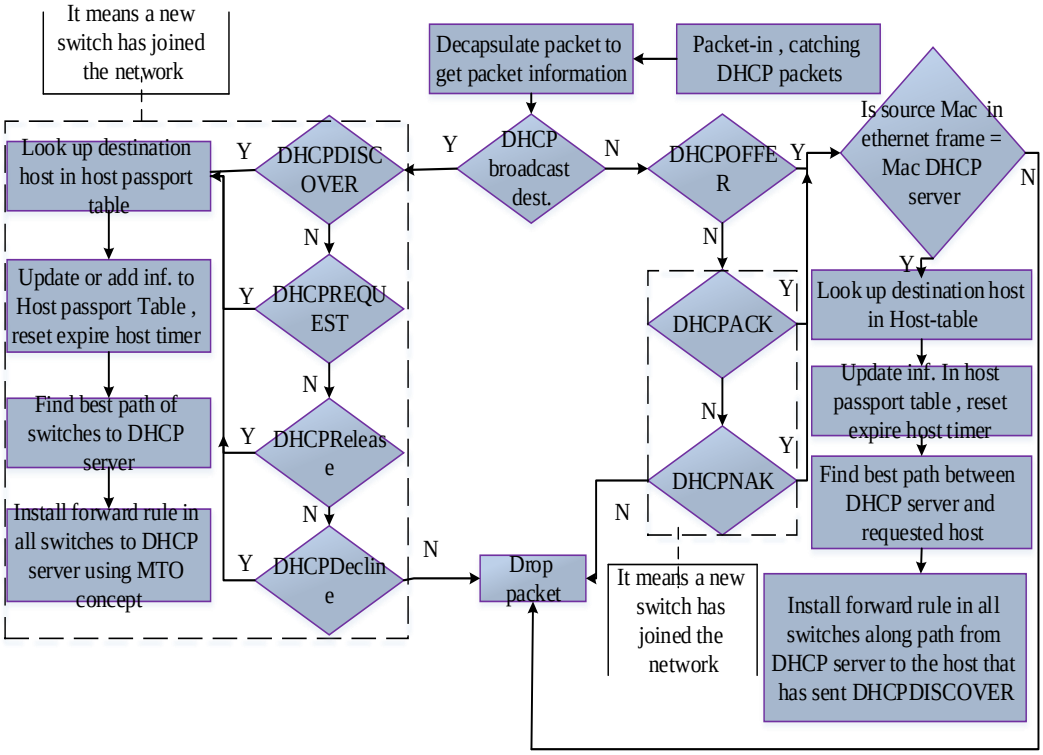


Figure 4.4: Flowchart of the SSED DHCP component for handling DHCP messages architecture

back a DHCPACK as a final agreement that allows the host to use the requested IP address. The DHCPACK will be forwarded directly to the host that has made the request without notifying the controller as there is a rule already installed in that switch from the previous DHCPOFFER phase.

There are other types of DHCP messages that transfer between the DHCP server and host without send notification to controller in the SSED architecture. A DHCPNACK message is a unicast from the server to the host, letting it know that the requested IP address is not allowed owing to an error, such as it now being used by another host or it is no longer valid. In addition, there is DHCPRELEASE, which is broadcast message sent by the host to the DHCP server to let the server know it will log out from network. Moreover, DHCPDECLINE is a broadcast message from host to server to notify it there is an error in the configuration parameters. In exceptional cases, if DHCPDISCOVER, DHCPNACK, DHCPACK, DHCPREQUEST, DHCPDECLINE and DHCPRELEASE

are sent as Packet-in to the SSED controller, this means that the switch sending the messages to the controller has just joined the network and so, the MTO rule has not yet been installed. Only a DHCPOFFER message should be sent in the Packet-in to the controller and just for the first time, because after that the server knows the route to the requesting host, unless there has been a change in the network topology. A detailed flowchart of the DHCP component in the proposed model is shown in Figure 4.4.

4.4.3 ARP component

The implemented Arp component contains two parts. Firstly, there is the ARP host discovery, which is described in detail in section 4.3 (host discovery mechanism). Secondly, the ARP service part refers to when a host needs to connect to other host it first having to get its MAC address so as to be able to send messages over the Ethernet. It sends an ARP broadcast REQUEST message to the nearest connected switch, which

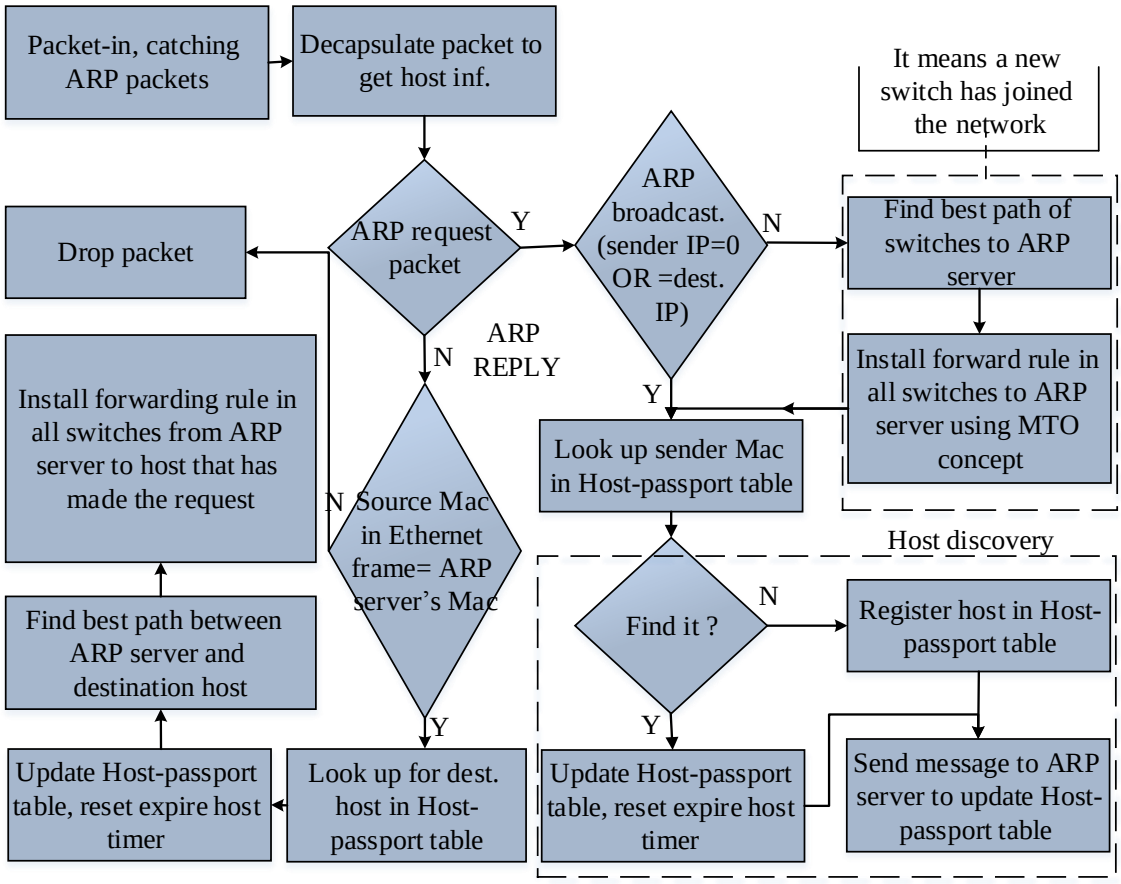


Figure 4.5: Flowchart for handling ARP messages

already proactively has had the MTO rule installed in it to forward any broadcast ARP REQUEST to the ARP server. Consequently, there is just one rule to forward messages from infinite hosts to the ARP server following a unicast concept. The request is forwarded along the switches until it reaches the ARP server. The server then decapsulates the message and sends an ARP REPLY message with the MAC address for the destination host in the source MAC field, if it finds it in the host passport table, if does not then the server drops the packet.

It is very important to let the server work as a filter just for valid requests so as to minimise the overhead on the controller. The switch that is connected to the ARP server will encapsulate ARP REPLY in a Packet-in message and sends it to the controller just the first time. The ARP component in SSED will be triggered by the ARP packet and decapsulates it to find the type of ARP message and the destination host for it. If the message is a broadcast request message with a zero in the IP source field or the IP source is equal to the IP destination, then SSED deals with the packet as an advertisement message. If not, this means there is new switch joining the network and hence, the MTO rule has not yet been installed in it. So, SSED finds the best path between that switch and the ARP server and installs the MTO rule. However, if the message is ARP REPLY, then SSED checks the source field for that message and if it is not from the ARP server it then drops the message for security reasons. Otherwise, it looks up the host-passport table to update the host information and resets the expire timer, whilst also calculating the best path back from the ARP server to the host that has made the request. A detailed flowchart of the ARP component in the proposed algorithm is shown in Figure 4.5.

4.5 Testbed results

In this section, comprehensive testbed results are provided to demonstrate the performance of the proposed SSED model. The testbed was built using 23 PCs, as can be seen in Figure 4.6, twenty of which have the specifications of core 2 Quad, 2.66 GHz, 2.9 GiB memory and an Ubuntu 14.04 operation system. These can be used either as SDN switches by activating an open virtual switch (OVS) or as a host performing role of a single host or multiple-virtual hosts, depending on the experimental scenario. Of the remaining three PCs, one works as a SDN controller, with the specifications of core i7, 3.40 GHz, 3.8 GiB memory and an Ubuntu 14.04 operation system. SSED uses a Ryu SDN controller as the network operating system (NOS), which was developed by Nippon

Telegraph and Telephone (NTT) as an open source operating system [94] that provides tools and libraries for design SDN components, which was written using the Python language for fast, easy and community supportive development.

The SSED components are implemented under Ryu using its expansive library. The final two computers are Samsung laptops, one of which works as the ARP server and the other as the DHCP server, both having the specifications of core i7, 2.20 GHz, 7.8 GiB memory and an Ubuntu 14.04 operating system. The SDN controller and OVS switches use the OpenFlow protocol [89]. The experiments use one of two types of topology, linear

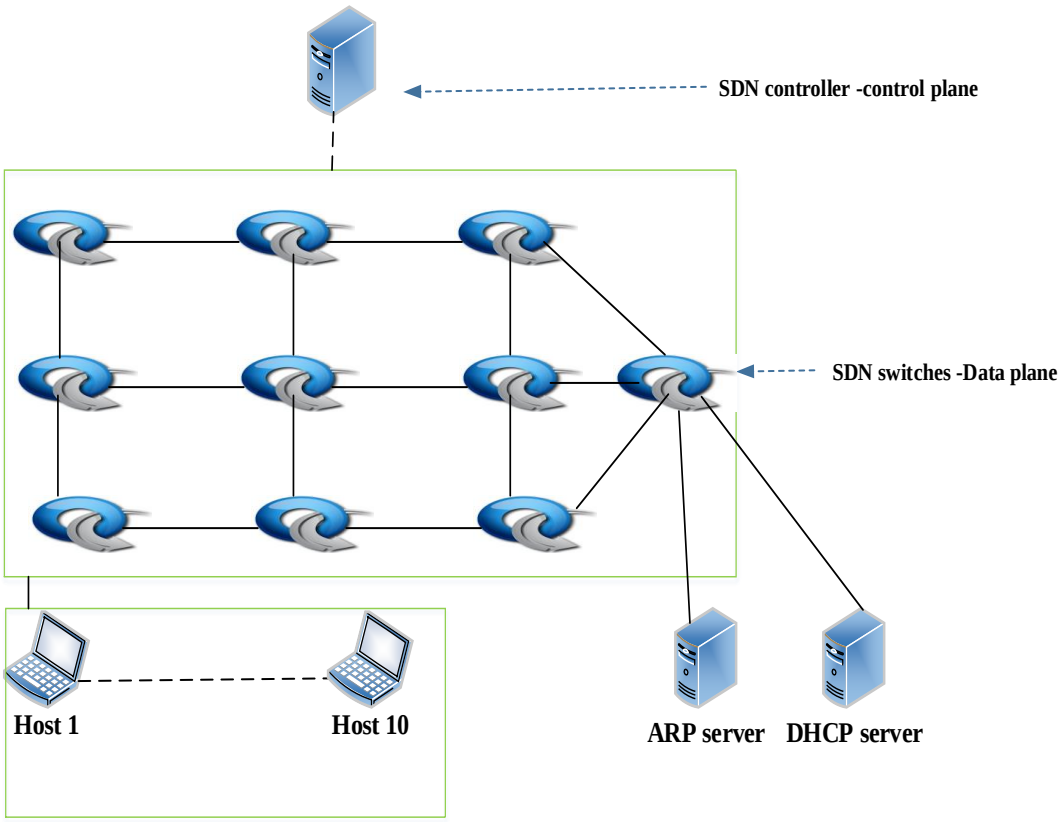


Figure 4.6: Logic diagram of SDN Testbed environment with 23 computers

or hybrid, depending on the purpose that they are designed to perform. Specifically, the linear topology is used in experiments that check the response time, because the effect element in this is the distance between the source and the destination, which is defined as the number of hops between them.

The hybrid topology (it is used in real networks) is used in the experiments that are designed to evaluate the network traffic ratio (see Figure 4.7), because it is affected by

the path that is chosen to reach the destination, which definitely is impacted upon by the mechanism that is used to forward packets.

To prove the efficient performance of SSED, several experimental scenarios were designed as follows. Note that the comparison involves the legacy switch scheme and the proposed scheme. There are two parts in this section, with the first dealing with traffic, whilst the second pertains response time.

The first part contains four experiments and concerns traffic in both the control and data planes. Ten PCs are used as OVS switches with a hybrid topology, as shown in Figure 4.7.

- The first experiment is performed to measure packet traffic in the control plane during 120-seconds bootstrap time and idle network behaviour using the SSED model and the legacy learning switch model.
- The second experiment is designed to measure the ratio of the network traffic to generation traffic, in both the control and data planes when generating 1 ARP of requested traffic under a traditional flooding scheme and the proposed scheme.
- The third experiment considers resource consumption for uncompleted requests, by calculating the ratio of network traffic to the generated failed requests in the control and data planes for both the SSED proposed scheme and the legacy scheme.

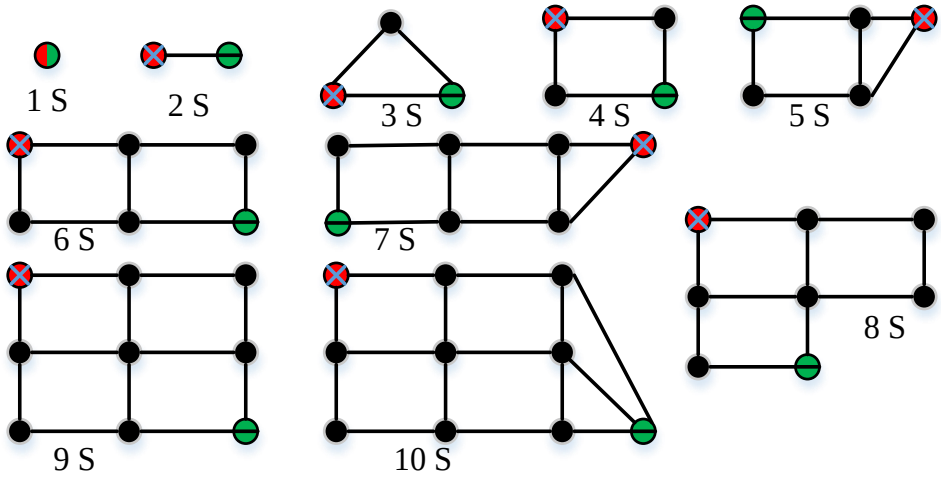


Figure 4.7: 1-10 switches (S) in hybrid topology.

The red circles with an (X) connect to the host and the green circles with a (-) connect to the server

- The fourth experiment is performed to measure and compare the controller's CPU usage under the legacy broadcast scheme and SSED.

In second part, three experiments are performed to measure the response, latency and discovery time with 10 PCs being used as OVS switches in a linear topology.

- The fifth experiment is performed to evaluate the time for discovering host information on the server side and the latency in the controller when dealing with discovery packets. This is achieved by generating an ARP discovery packet from the host side and recording the receiving time for that packet on the server and controller side.
- The sixth experiment is run to measure the response time for receiving a service that is requested by a host.
- The seventh experiment is performed by increasing the number of hosts' requests per second on the SDN network. The aim is to evaluate the performance of the SSED during generated light, medium, heavy load from users working concurrently (how does increasing the number of requests affect the response time) on this network. It differs than previous experiment, in that it involves measuring how the response time is affected by sharing the network with multiple users. Then, the performance of SSED is compared with that for legacy schemes.

4.5.1 Bootstrap traffic: SSED and legacy scheme comparison

In this experiment, we first fix number of switches in testbed to 10 and use the hybrid topology in Figure 4.7. In addition, no hosts are connected to the network so as to avoid traffic from them, with the focus thus being on traffic that is generated to establish the main parts of network, including the SDN switches and controllers. There are some processes start automatically during the bootstrap process without any external input (e.g. hosts), for example, the SSED switch discovery process and the legacy switch learning process. Thus, during the bootstrap time we can calculate how many packets are *in* and *out* from the control plane to establish the network before any host is connected to it. The experiment is run for 120 seconds, which is approximately enough bootstrap time for the discovery of 10 switches and the Wireshark tool is used to measure network traffic in the control plane. Gradually, over time, as can be seen in Figure 4.8, the number of Packet_in and Packet_out in the control plane by using SSED is increased to reach 8,022 packets.

That number of packets is being generated because SSED during the bootstrap mode generates LLDP packets for management and switch discovery purposes. In addition, it

drops all legacy management packets, such as a multicast listener report message in the MLD protocol, which is multicast by an IP node to report their interface status to their neighbours [85] and a multicast STP, which is used to build loop-free topology in a legacy network [84].

Regarding the traffic from using the legacy learning switch, this is significantly greater, rising to 54,653 packets, because it relies on flooding for discovery and on management services using MLD and STP. That is, when the time reaches 120 seconds, SSED only makes 14.67 % of the overhead (number of packets) on the control plane that the legacy learning switch deploys for discovery and management services in the bootstrap period and when the network is idle. The idle network statistic is beneficial for

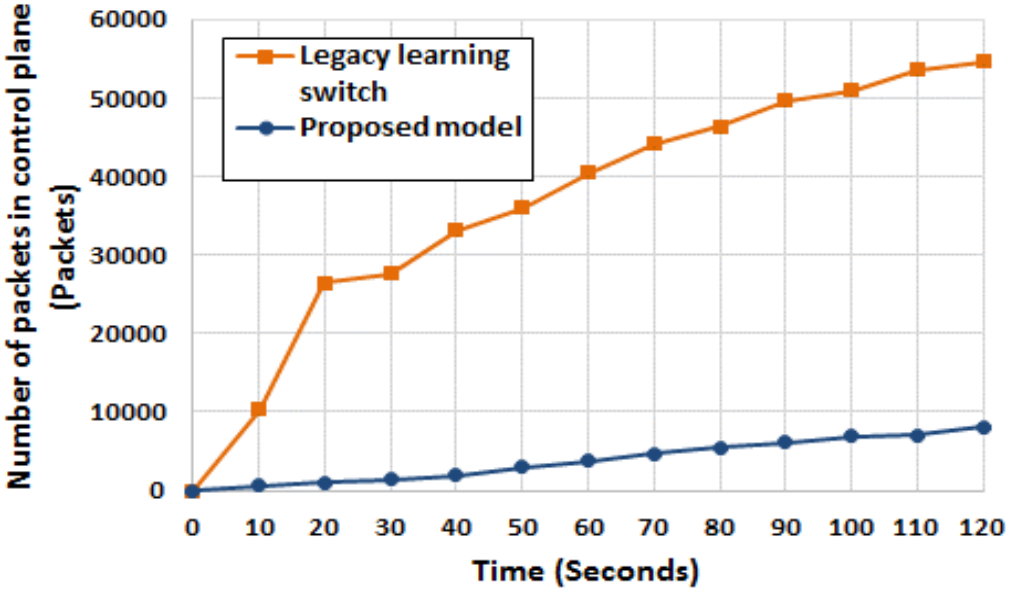


Figure 4.8: Network traffic with SSED compared to the legacy learning switch mechanism during the bootstrap and idle network condition when using hybrid topology.

evaluating the standard calculation that can help administrators find the threshold overhead on the control plane, thus potentially allowing for the determination of the hardware and software specifications needed for that plane.

4.5.2 Ratio of network traffic to generated traffic: SSED and legacy scheme comparison

In this experiment, we use hybrid topology and increase the number of switches from 1 to 10 switches. Only one ARP request message is generated from an edge host to the

edge ARP server, as can be seen in Figure 4.7, where the red circle with an (X) is the server and the green one with a (-) is the host. Then, by using the Wireshark tool the ratio of network traffic to generation traffic is calculated in both the control and data planes under the legacy flooding scheme and the proposed scheme. The main idea is send one ARP request packet and to measure how many packets will be generated in these two planes to get an ARP reply to the requesting host.

Regarding the control plane statistics, with an increase in the number of switches in the network the number of control message in control plane is approximately still the

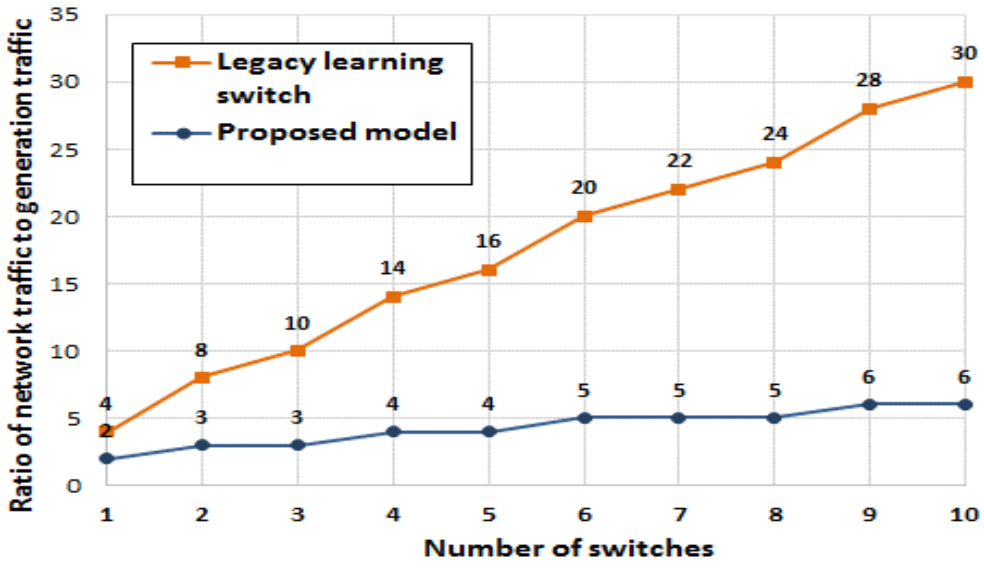


Figure 4.9: Reduce network traffic in the control plane: comparison of SSED with and legacy learning switch mechanism

same or just slightly increases when using the SSED proposed model, because the ARP reply process needs just one Packet_in as the control message from the server to the controller plus a number of Packet_out messages equal to the number of hops for the best path between the server and the requesting host. For example, in Figure 4.9 it can be seen that no matter whether the network has 6, 7 and 8 switches, the number of control messages remains the same, i.e. five control messages, because there is one Packet_in to the controller from the server and four Packet_out to install the forwarding rule in the switches along the best path from the server to the host. With hybrid topology, sometimes the distance stays unchanged between the source (host) and the destination (server) when increasing the number of switches in the network, because there are a number of possible paths to connect these two entities, which is different to linear topology with only one

possible path. On the other hand, for the legacy learning switch the network traffic increases significantly when the number of switches in the network becomes greater, because of the flooding of ARP broadcast packets to reach every switches in the network even though some are not on the path for reaching the destination.

The data plane statistics in Figure 4.10, show that by generating one ARP request from an edge host with hybrid topology SSED keeps or uniformly increases (+4 packets per new switch) the ratio of packets to handle sending ARP reply packets by the ARP server to the host that has made the request (keeping or increasing the ratio depends on number of links that is needed to connect the host with the server). In contrast, the legacy learning switch increases the ratio practically linearly, because it floods the packet to every node in the network. SSED needs just 44.44 % of the number of packets needed by the legacy learning switch to deal with 10 switches in order to send back an ARP reply to

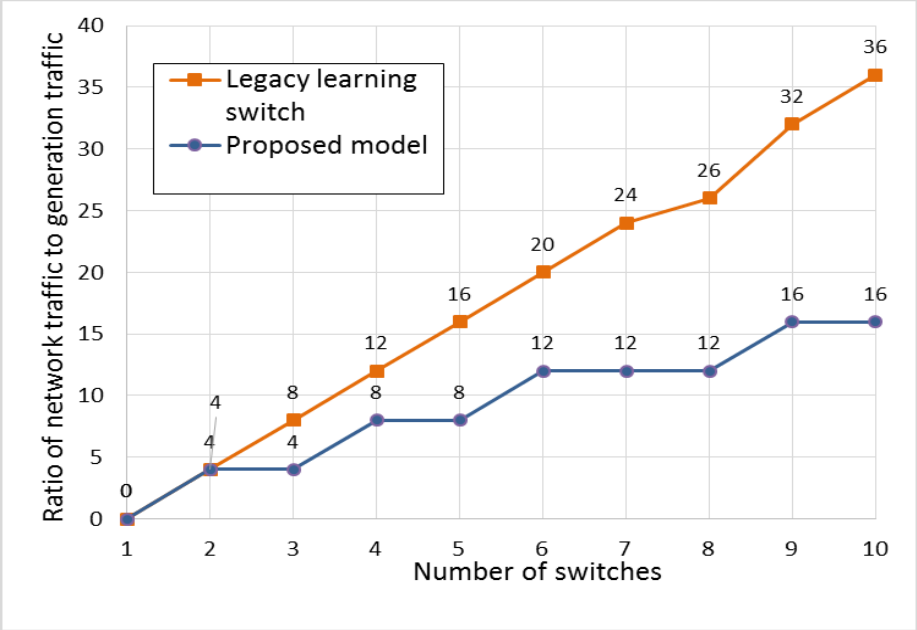


Figure 4.10: Network traffic in the data plane: comparison between SSED and the legacy learning switch mechanism

the sender because it benefits from the proactive installed rule in the switches using the MTO method to reach quickly the ARP server. In addition, it uses the Dijkstra algorithm to find the best path between the server and the host who has made the request.

4.5.3 Effect of retransmission traffic (resources consumption) on the control and data planes: comparison between SSED and legacy switches

To evaluate the effect of retransmitting traffic (that normally occurs in a daily network) with SSED, experiments for 10 hosts connected to an edge switch with 10 fixed SDN switches in a hybrid topology network is utilised. The hosts generate ARP faulty requests in order to obtain the goal of evaluating the effect of retransmitting traffic on resources consumption. A failed ARP request refers to not being able to find a requested destination's MAC address in the ARP hash table in the ARP server, which can be performed by sending requests to an unreached random destination. As a consequence of the failed request, the source starts retransmission of the ARP request multiple times as this is the normal behaviour of the Transmission Control Protocol (TCP) [95].

The main reason for retransmitting is that the source that made the request has not received a reply within a specific time period, which could be due to several causes, such as network congestion, interface error or buffer overflow. Other reasons for this are that the ARP server has not yet registered the destination host in its ARP hash table or the server has blocked its address owing to a security issue. Consequently, any ARP request asking for that a host's MAC address will not get a reply and this results in the retransmission of the same request from the source. To make the experiment replicate daily used network traffic as closely as possible, different ARP request rates are generated

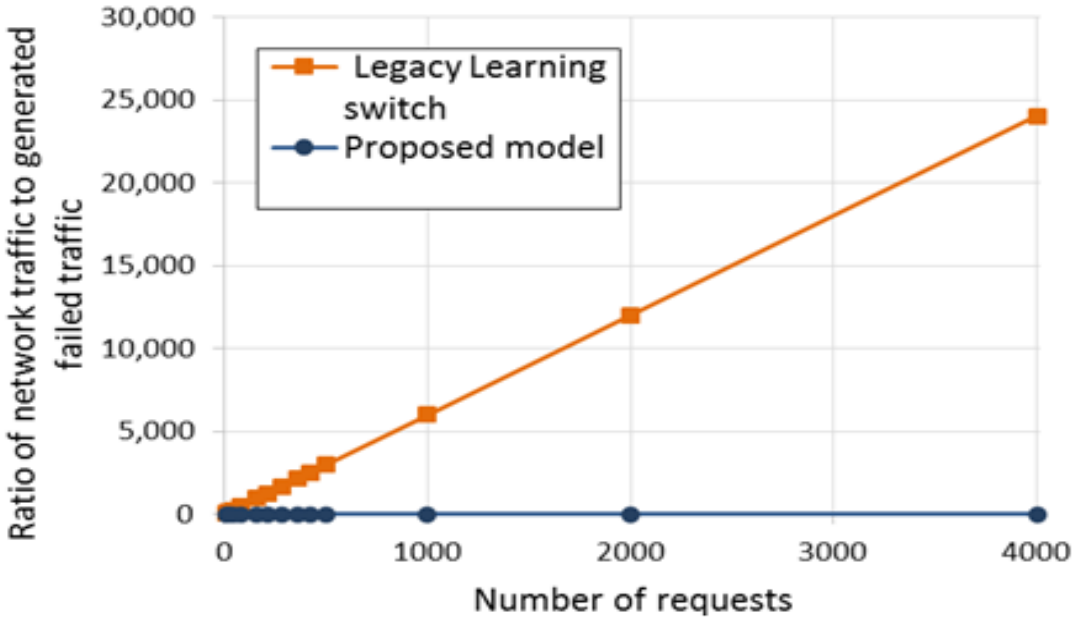


Figure 4.11: Effect of the retransmission of traffic in the control plane: comparison between SSED and the legacy learning switch mechanism

from the 10 hosts at the same time. Regarding control plane evaluation, Figure 4.11 shows that with an increasing number of requests there is absolutely no effect on it when using SSED, because it proactively gives the responsibility to answer requests to the server that uses data plane for the purpose. The server works as a filter for reducing the number of requested packets and just passes the valid ones to the controller. This results in reduced overhead on the controller and eliminates the possibility of an attack on the controller. In contrast, the ratio of traffic using the legacy learning switch increases linearly to reach 24,000 packets of a Packet_in and Packet_out form for 4,000 requests in the control plane, because it floods every retransmitted request to everywhere in the network.

In relation to the data plane evaluation (see Figure 4.12), SSED generates 20% of the legacy learning switch traffic. That is, it provides a 80% reduction in the consumption of network resources in the data plane when the number of failed requests reaches 4,000 with all hosts working concurrently. In addition, compared with Cisco's recommendation [10], which is that there should be no more than 500 devices in one collision domain, if

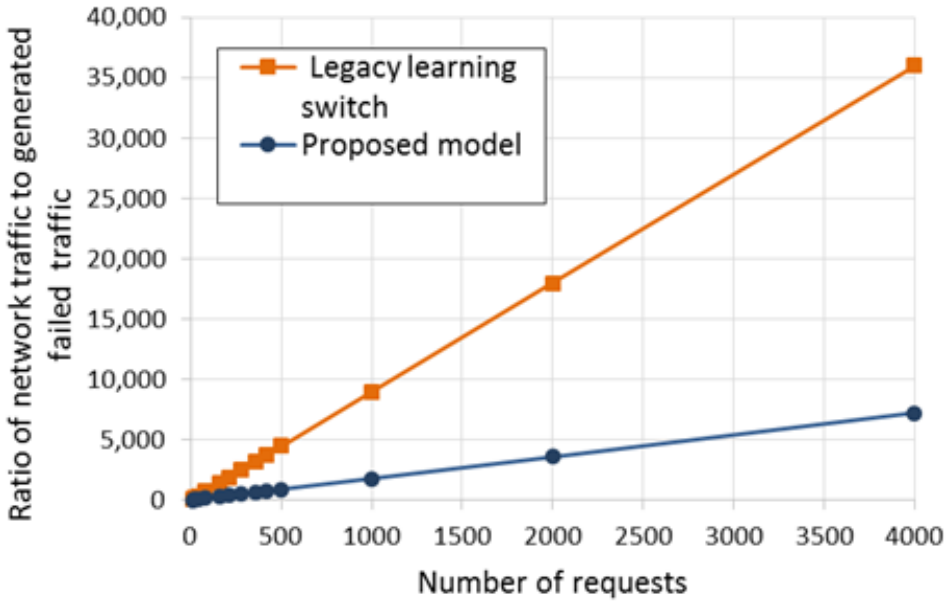


Figure 4.12: Resource consumption during concurrently failed requests in the data plane: comparison between SSED and the legacy learning switch mechanism

each host is assumed to generate 8 RPS at peak load [53], then from Figure 4.12 the legacy architecture with 500 (i.e. 4000/8) hosts generates 36,000 packets in the data plane, whilst SSED generates 7,200 . As a consequence, SSED, with same number of packets(i.e 36000) in the data plane of the legacy architecture, can increase the number of hosts to 2500, thereby scaling up the number of hosts by 500% when compared to the legacy mechanism.

4.5.4 CPU usage in the controller (SSED scalability)

In this experiment, a hybrid topology with 10 switches is connected and N number of virtual hosts are created and connected to the network. A fixed request number of a heavy user generation rate of 8 ARP requests per second is generated per virtual host with random IP addresses for sources and destinations. For this, the system monitor CPU tool in Linux is deployed to monitor the CPU usage, the measurement of which before any model being applied is 2.9% of core i7 CPU with 3.40 GHz, while it is 3.36% for SSED and 4.96 % for the legacy learning switch for the bootstrap communication management network.

As can be seen from Figure 4.13, with a growth in the number of virtual hosts and a fixed rate of eight requests per second per host, the average percentage of CPU usage under SSED increases slightly from 6.31% to 12.5% for 1 to 500 virtual hosts (at peak load), respectively. It can clearly be seen that it reaches approximately a stable state after the connection of 50 virtual hosts concurrently owing to SSED's balanced multithread algorithm. This percentage of CPU usage is for handling ARP replies by finding the best path and installing rules in switches from the server to the host. However, when using the legacy learning switch the increase in (N) leads to an increase CPU usage from 6.42 to 43.12 for 1 to 500 (peak load host number [10]) virtual hosts, respectively. This is because

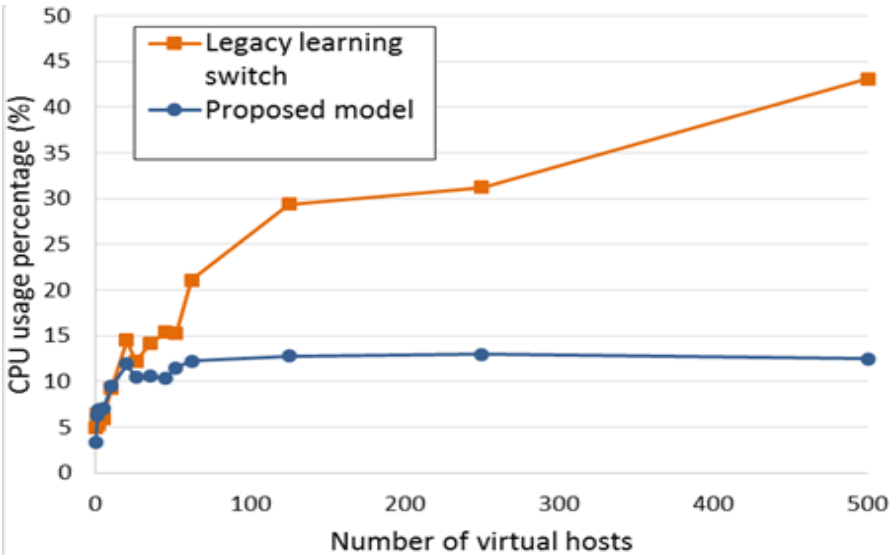


Figure 4.13: Average CPU usage in the controller for SSED and the legacy learning switch mechanism

it has to handle many Packets-in and Packets-out per each ARP request owing to the flooding scheme as well as the complexity of the algorithm for finding the shortest path

to the source. Next, the results of the second part of the experiments using the same constructed testbed, but with linear topology are reported regarding the three experiments relating time.

4.5.5 SSED host discovery time and controller latency

In this experiment, with the SSED model, linear topology is used with an increased number of switches from 1 to 10 and one host connects to one edge switch, while the ARP server connects to another. The host generates an ARP discovery packet, as explained in section 4.3, which is entered as Packet_in to the controller, which then forwards this to the server. The latency time for the controller to complete the discovery packet forwarding process is evaluated and the results can be seen in Figure 4.14. After that, the discovery packet will be received by the ARP server, which adds or updates the record in host passport table. The time that the discovery packet needs to reach the server from the requesting host is evaluated.

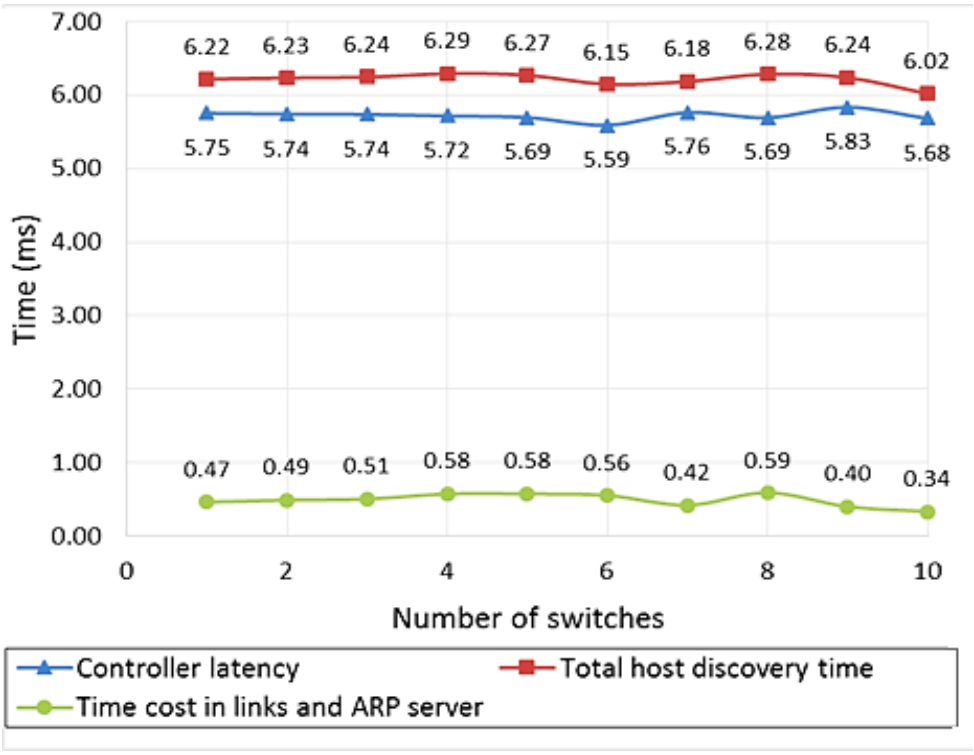


Figure 4.14: Time spent on the host discovery process using SSED

It can be seen from Figure 4.14 that the latency value and discovery host time have negligible impact on scalability, i.e. when the number of switches is increased, with the average values for these being 5.71 ms and 6.21 ms, respectively.

This result is because SSED uses the controller as a link between the host and the server for the host discovery service, so the number of hops between the host and the sever is immaterial. In addition, the time that the host discovery packet spends at links and the ARP server is calculated by subtracting the total discovery time from the controller latency time, the average being 0.49 ms. This means that the discovery packet needs just 8.58 % of the time spent in the controller to pass through the links and server, which proves that the SDN controller owing to its complex nature (performing multiple jobs at the same time) spends more time compared with the distributed services that SSED uses to deal with broadcasted packets.

There are some slight fluctuations in the results, which is firstly because of collisions and competition between packets (i.e. OpenFlow negotiation packets and ARP discovery packets) using the same links (i.e. links between the controller and source/destination switch) at the exact same time. Secondly, the process time inside the controller fluctuates depending on the load.

4.5.6 Response time

In this testbed experiment, a linear topology with an increase in the number of switches from 1 to 10 and generating one fixed ARP request using the ARPing tool from the host connected to an edge switch requests the MAC address for a destination host that connects to the network randomly. From Figure 4.15, it is clear that with an increase in the number of switches, the ARP response time using SSED increases gradually at average of 0.19 ms for each added switch to the network.

This rate is as a consequence of sending a Packet_out message from the controller to the added switch to install the matching rule in order to match and forward the ARP reply from the ARP server to the host. By contrast, when using the legacy learning switch the growth rate is 3.29 ms on average, when adding a new switch to the path between source and destination hosts. This is because each added switch deals in the broadcast phase with one Packet_in by sending it to the controller as an ARP request and one Packet_out is sent by the controller to instruct the switch to flood that packet to all neighbouring nodes. After that, to handle the ARP reply on the way back from the destination host, the added switch sends one Packet_in to the controller to look up in the MAC-to-port table the source node that has made the request, which in turn, sends one Packet_out in the form

of an ARP reply to the requesting host. As a consequence, as can be seen in Figure 4.15, the response time practically linearly increases in proportion to the network switch scalability.

Regarding the scalability in relation to this experiment, by using the broadcast mechanism the ARP packet during the *Request* and *Reply* phases passing 10 switches requires 35.57 ms. Whilst SSED with that response time (i.e. 35.57 ms) can pass approximately 161 switches (whereas, as can be seen in Figure 4.15, the response time for one switch is 4.95 ms and each added switch needs 0.19 ms). As a consequence, SSED scales the network approximately 1510% more than the broadcast mechanism.

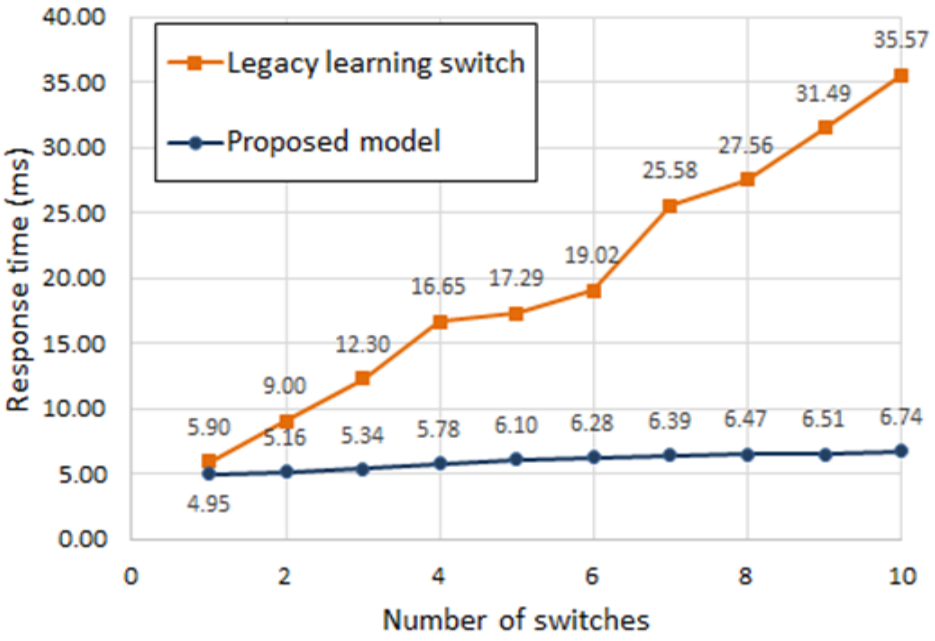


Figure 4.15: ARP response time, comparing the proposed SSED model with the legacy learning switch mechanism

4.5.7 SSED Performance during different load

The performance and its stability for the proposed model is evaluated by generating light, medium, heavy and overloaded traffic from 10 concurrently working hosts. For this experiment, 10 fixed switches are connected with a linear topology. There are 10 hosts, each being connected to one SDN switch and the ARP server is connected to the fifth switch. The different rates of traffic sent concurrently from each host, are 1-4 requests per second (RPS) as light traffic users, 5-6 RPS as medium traffic users, 7-8 RPS heavy traffic users (at peak load [53]) and 10-12 RPS as overloaded traffic users. Each source

host generates an ARP request for a MAC address for random destinations, each being designed to be unique in relation to all other requests from the same host so as to guarantee that they are not affected in any way by others' requests.

As can be seen Figure 4.16, with a light load traffic of 40 RPS as the total number of requests from 10 hosts working at the same time, the proposed model offers a better average response time than the legacy learning switch with the values being 16.34 ms and 23.86 ms, respectively. For a medium load with 6 RPS from each host, SSED also offers a better response time, the figures this time being 20.806 ms and 24.481 ms, respectively.

The same trend occurs for the heavy and overloaded scenarios, lead to the conclusion that SSED is efficient in terms of its performance as it well dealing effectively with increasing traffic rates. This is mainly because it handles ARP requests with fewer Packet_in and Packet_out than the legacy learning switch, which means less traffic is transmitted across the network and as a consequence, there is less competition as well as congestion at links, which in turn leads to lower response times. However, the average response time using the legacy learning switch increases with an increasing number requests, whereby each *request* entered to the switch will generate 1 Packet_in and 1 Packet_out until reaching the destination through all switches using the broadcast mechanism. Subsequently, each *reply* generates another 1 Packet_in and 1 Packet_out, if the switch was chosen as a hop within the shortest return path, otherwise (i.e if the switch is not chosen within the return path) the switch deals with just the 1 Packet_in and 1

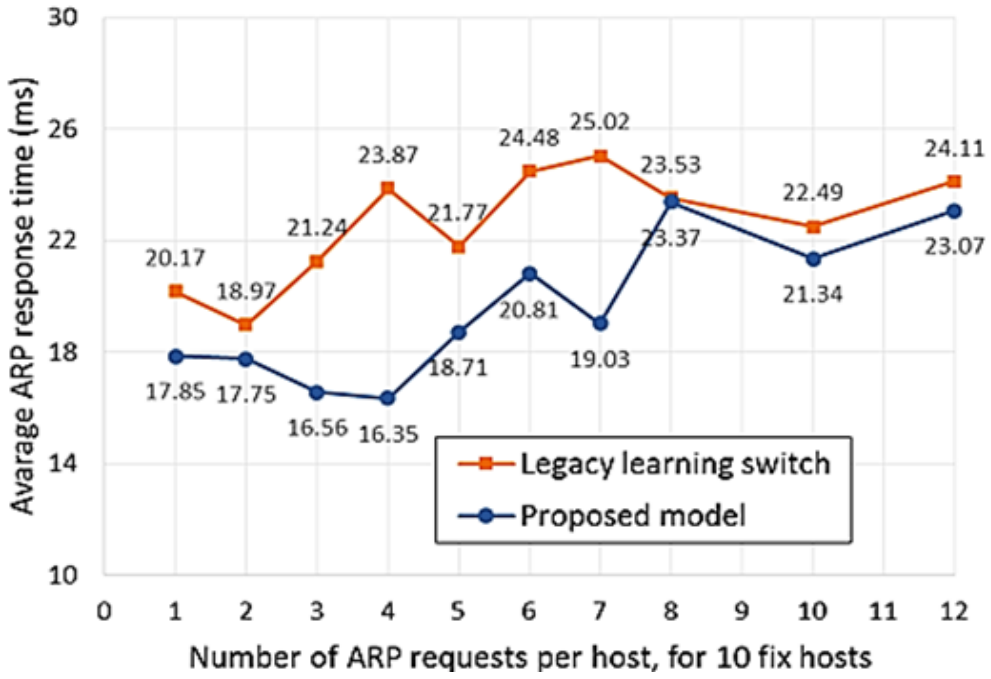


Figure 4.16: The performance measures according to average ARP response time with different request rates from 10 fixed hosts connected to 10 switches

Packet_out that were generated during the broadcast phase.

As can be seen from Figure 4.16, both approaches approximately meet at 8 RPS and there is 1 ms difference between them in 10-12 RPS, for two reasons. Firstly, there is the use of linear topology, which reduces the detrimental effect of the broadcast mechanism and hence, diminishes the response time when using that mechanism. Secondly, SSED, by using the ARP server at the middle switch, leads to more competition on that switch when increasing the requests and hence, increases the response time

4.6 Summary

In this chapter, firstly, the Ethernet network with its current switch features and the requirements for designing scalable networks have been discussed. Subsequently, there were analyses of the broadcast mechanism, where it was explained that most of Ethernet's drawbacks occur owing to the nature of the usage of broadcast packets. To address these, the SSED architecture, design and implementation using several constructed testbed experiments with 23 computers to handle broadcasting packets was introduced, in particular, with the purpose of overcoming the side effects of broadcasting. The results have shown that the proposed model can eliminate broadcast packets from the network, providing: a reduction in the consumption of network resources in the data plane by 27,800 (35,000 - 7,200) packets, a reduction of control packets in the control plane to 8,022 packets during bootstrap time; less peak overhead on the controller, which prevents it experiencing failed requests; a better response time; and more efficient performance. In sum, SSED provides better scalability, increasing the capacity of one domain to deal with 2,500 - 3,076 hosts and 161 switches, when compared with broadcast-based architectures.

Chapter 5

Multiple Distributed Subnets using SDN Architecture and Servers

5.1 Introduction

Bargain-basement, high speed and plug and play are important features that make Ethernet widely utilised in local area networks (LANs), which are the backbone for other networks. However, interconnected LANs exhibit several limitations, restricting scalability and efficient performance, including the use of router devices between LANs and the default gateway mechanism. It would appear that a Software Defined Networking (SDN) can help to enhance an Ethernet-based network, in terms of better efficiency and scalability. However, it is important not to inherit inferior features from legacy networks and instead, develop new architecture, especially when building a large network. That new/developed architecture has several requirements to ensure it is compatible (e.g. network protocols) with legacy network architecture that will help in the future migration from legacy to the SDN network. Firstly, the host side should not be modified by a third party program between the host and network protocols so as to prevent compatibility issues. Secondly, network transparency so that hosts do not know how the packets will be routed to their destinations, thereby delivering better security. Thirdly, there should be no new hardware added to the network as this could lead to backward compatibility issues. Finally, minimising the number of protocols that use in network will help to reduce the complexity.

Hence, to satisfy all the requirements above and to solve all the listed limitations in chapter 3, regarding the use of middlebox devices and the default gateway mechanism, as well as to improve network efficiency and scalability using Ethernet technology,

Multiple distributed subnets using SDN architecture and Servers to Eliminate Router devices and the Default gateway mechanism (MSSERD) is proposed in this chapter. MSSERD, firstly, analyses the characteristics of the most important connection packets in Ethernet environment, namely, the ARP packets that provide MAC addresses, which are essential for Ethernet devices to communicate. Then, a method to connect multiple subnets is proposed, which eliminates router devices (either as hardware or software) and the default gateway settings completely from the network. In addition, an innovative dynamic multiple subnets discovery approach is introduced, which can discover multiple distributed SDN-subnets.

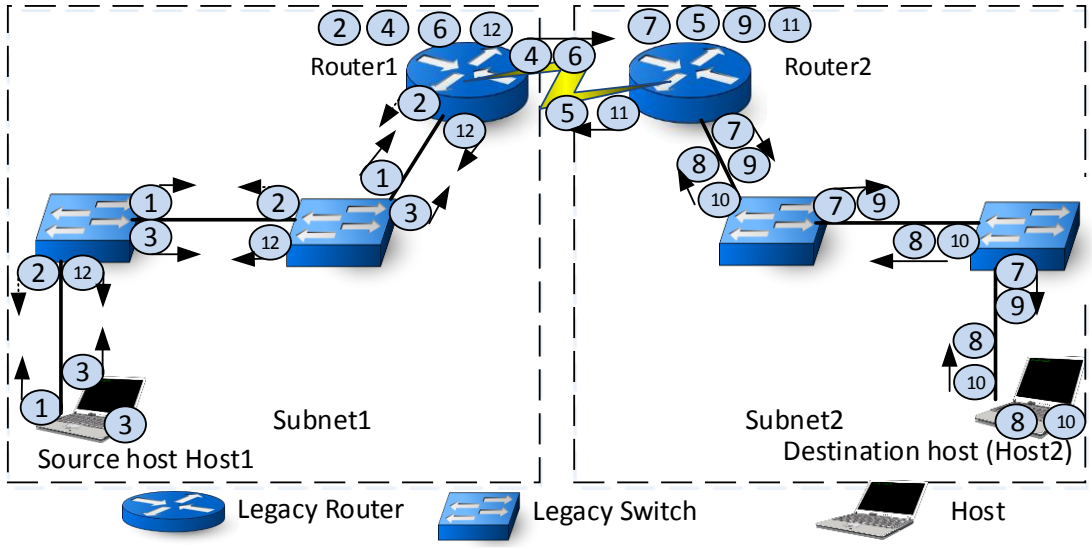
5.2 Legacy system description and analytical model formulation

In this section, the current mechanisms for connecting multiple subnets by ARP packets using router devices and default gateway settings under legacy and SDN architectures are analyzed. In addition, some concepts related to that mechanism are covered.

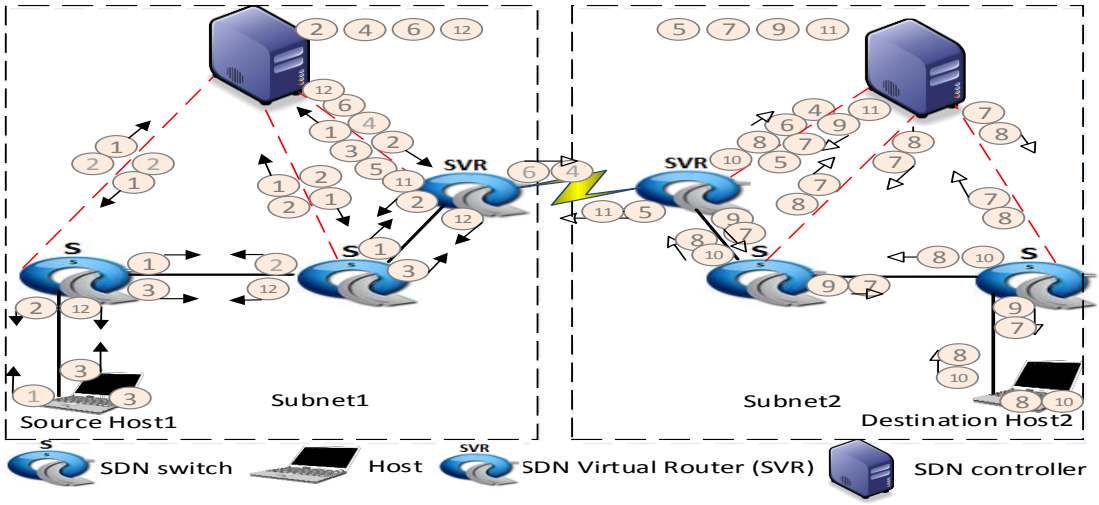
5.2.1 ARP under router based architectures

The ARP protocol is one of the most important protocols in IP-based Ethernet networks, because it performs resolution from IP to MAC addresses in order to connect different devices. It can be described as like making a call to setup a channel line between two phones, where the ARP component initiates the connection between the source and the destination, which is equivalent to the ringing of the destination phone. After the metaphorical ring at the destination phone, it answers, which in this case is equivalent to an ARP reply generated by the destination and then, communication can occur. The ARP having done its job then passes the connection responsibility to the Ethernet protocol to continue connecting and transferring different types of packets between the source and destination hosts by using layer 2 rules, if they are in the same subnet. However, if the two hosts are in different subnets the mechanism is different, because router devices need to be used between those subnets to let those hosts be connected and the router continues to use layer 3 rules for all future packets between those two hosts. To clarify this idea, how the ping process, which contains ARP and ICMP, is performed between two hosts across two different subnets is explained.

In current network architecture, when the source host (Host1) wants to ping the destination host (Host2) in another subnet, there are several packets generated in a legacy network, as shown in Figure 5.1 (a). The source host determines whether the target host's



(a) Routers under legacy architecture



(b) Routers under SDN architecture

Figure 5.1: Shows generation of different packets to discover the destination MAC address between different subnets in legacy and SDN network architectures

IP address in the ARP/Ping command belongs to its same subnet or if it belongs to one outside. Host1 does this by finding the network addresses (network ID) for its local IP and the destination IP by performing an AND operation for both IPs with the source

subnet *Mask* address [96]. If both network IDs are the same, this means both IPs are in the same subnet, otherwise they are in different ones. If the source host discovers that the target IP address is in the same subnet, it will just broadcast an *ARP request* to find the local destination MAC address and wait for an ARP reply sent by the destination host to provide its MAC address.

If the target IP address is in a different subnet to the source host, Host1 will, firstly, look in its routing table to find the default gateway IP address that has been provided to it by the DHCP server or assigned to it through manual setting and if it does not find it, the source host will drop the packet. If it does find the gateway, which represents the IP address of the internal interface of the router, it will broadcast an *ARP request* packet to discover the default gateway MAC address in its subnet, which appears as the first packet ① in Figure 5.1 (a). This packet has an IP destination equal to the gateway IP address (internal interface address of the router) and destination MAC equal to '00:00:00:00:00:00'. The switch that connected with Host1 receives this packet. Then, in turn, each switch in subnet1 continues to broadcast the *ARP packet* ① until it reaches to the router (Router1) that works as default gateway for Subnet1. After that, the Router will send back *an ARP reply*, i.e. packet ②. This pertains to the MAC address, which is sent along the shortest path in a unicast concept back to the Host1, which made the request. In this way, the router is telling Host1 that "*Any types of packets you (Host1) want to send it outside our subnet, just send them to me (Router) by using layer2 and I will send it to its destination on behalf of you by using layer 3, after which I will send back the reply to you when I (Router) get it*". As a consequence, the legacy system has two rules:

- Legacy Rule1: Host1 will never know the MAC address of the destination host (Host2) if they are in different subnets.
- Legacy Rule2: No ARP packet containing Host1's MAC address will go outside its subnets asking for the MAC addresses of other hosts in other subnets. That is, the router in the legacy network will drop any ARP request with an IP destination not equal to its IP.

In order to understand how the full ping connection procedure between Host1 and Host2 works, the procedure for sending an ICMP packet is described. After Host1 has received the MAC address of Router1, it will send an ICMP packet by using the Internet

control message protocol, which contains the IP source field equal to Host1's IP address, the IP destination field equal to Host2's IP address, the MAC source field in the Ethernet frame equal to Host1's MAC address and the destination MAC field in the Ethernet frame equal to Router1's MAC address. The packet is represented in Figure 5.1 by packet ③, which goes through the switches by using layer2 rules in the unicast concept to reach Router1 in Subnet1.

Router1 then checks the destination IP in the ICMP packet and looks in its routing table to find the matching rule. If it does not find any rule (no matter whether it is set up by the static or dynamic concept), it will send the ICMP packet with "unreachable destination IP" to host1 [28] in order to prevent it continuing to send another ICMP request. However, if the router finds a rule for reaching the next IP hop, it will, firstly, check its local ARP cache table and if there is no MAC address for the next hop, then it will send an ARP request packet to ask for the next hop's MAC address, which is packet ④. After that, a router in subnet2 (Router2) gets an ARP request and it will generate an ARP reply with its MAC address and send it back to subnet1, which is packet ⑤. Router1 in subnet1 will update its ARP table by adding Router2's MAC address. It then encapsulates the ICMP request packet that it received from Host1 in the modified Ethernet frame, where the source MAC field equals Rouer1's external interface MAC address, whilst the destination MAC field equals Router2's external interface MAC address. Then it sends the ICMP with the modified Ethernet frame to Router2, which is packet ⑥. Router 2 will look in its ARP cache table and if does not find Host2's MAC address, then it broadcasts an ARP request to its local subnet, namely, packet ⑦. The destination host receives the ARP request and generates an ARP reply, packet⑧ and sends it back with Host2's IP and MAC addresses in the source fields in the ARP frame, with Router2's internal interface IP and MAC addresses in the destination fields. Packet ⑧ goes through the switches using the shortest path to reach Router2.

Router2 will use Host2's MAC address to encapsulate the ICMP request packet, i.e. packet ⑥, that already has been received from Subnet1 in the Ethernet frame with a source MAC equal to Router2's MAC address and a destination MAC equal to Host2's MAC address, namely, packet ⑨, Then, it will be sent to the Host2, which generates an ICMP reply packet, packet ⑩, with an IP source field equal to Host2's own IP address and IP destination field equal to Host1's IP address. Host2 then encapsulates it with the Ethernet frame with a source MAC field equal to Host2's MAC address and a destination

MAC field equal to Router2's internal interface MAC address. Packet⑩ goes through the switches in Subnet2 in a unicast concept until it reaches Router2, which then makes packet⑪ by encapsulating the ICMP reply packet with a modified Ethernet frame that contains Router2's MAC address in the source MAC field and Router1's MAC address in the destination MAC field and then sends the packet to the Router1. As a consequence of this, the legacy system has a third rule:

- Legacy Rule3: Each router must change source MAC address and destination MAC address in the Ethernet frame each time it passes packets to another subnet or to its own subnet.

Router1, after that, decapsulates the Ethernet frame to check the destination IP in the ICMP reply packet. When it realises the destination IP belongs to Host1, it will encapsulate the ICMP reply packet in a modified Ethernet frame that contains Host1's MAC address as the destination MAC address and will send packet ⑫ along the shortest path in the unicast concept, until it reaches Host1.

It can be seen that regardless the number of switches in the legacy network, the number of different packets that are generated to get the ICMP reply is 12 (6 ARPs and 6 ICMPs) and the original ICMP packet changes its Ethernet frame 4 times which leads to the following equation being derived:

$$\mathbf{N}_{def} = N_r * 2 \qquad (5.1)$$

Where, $\mathbf{N}_{def}$ denotes the number of different Ethernet frames (i.e. the number of modifications to the Ethernet frame) for the completed process (1 Request and 1 Reply) and N_r is the number of routers along the path.

It should be noted that the layer2 path between the source and destination needs to refresh after a specific time depending on the vender of devices that are performing caching. For example, Cisco routers take 4 hours and they do not use entries into the ARP table then they will make an ARP requests for the IPs in that entries, whereas for devices using Linux and Windows the ARP timeout is 5-20 minutes [97]. Accordingly, the equation for driving the number of devices performing caching in legacy architecture, if it is assumed that there are just two hosts in different subnets, is equal to:

$$N_{dc} = 2 + N_r \qquad (5.2)$$

Where, N_{dc} denotes the number of devices performing caching and Nr represents the number of routers on the path between the source and destination hosts.
In the example, $N_{dc}=2+2=4$ devices and they need to refresh their ARP caching tables after each a specific time.

In SDN architecture, as can be seen from Figure 5.1(b), there is the same procedure for legacy architecture and the same principle, whereby it contains SDN controller and SDN switches that can be work as normal switches or as an SDN Virtual Router (SVR). SDN implements the virtual router inside the controller and chooses one/or more of the switches to be routers with virtual IPs for the *in* and *out* interfaces. For any packet coming to the SVR that it does not have a rule for routing to its destination, then the packet will be encapsulated with an OpenFlow protocol frame and sent to the controller to get packet information. In addition, any ARP request/reply coming through SVR to the SDN controller must go directly to the virtual router module inside in order to update the virtual router ARP table. For example, when the SVR in Subnet1 receives the broadcast message from Host1 that asks for the default gateway's MAC address, it forwards that packet to the controller in order to cache Host1's MAC address and generates an ARP Reply message with the gateway's MAC address which is sent to the SVR that directs it back to the requesting host (Host1).

5.2.2 Number of packets in router-based SDN architecture

In multiple subnets SDN architecture the parameters that are used to evaluate the number of packets in the control plane are number of packets that are exchanged between the switches and the controllers (in just the source and destination subnets) plus the number of packets exchanged between the routers and controllers (along the subnets' path from the source to destination subnet). Regarding the number of packets between the switches and the controllers, if a switch is in the chosen path between the source/destination host and the router, it deals with the controller using 2 packets (1 Packet_in and 1 Packet_out) in response to an ARP request from the source. In relation to the number of packets from the routers to the controllers, the number of subnets (Nsu) between the source and the destination affects the calculation, because each router in the middle subnets, if it is in the chosen path, broadcasts an ARP discovery request to all connected subnets and receives 1 ARP Reply in order to add/modify the entry in its ARP

table that lets it uses Ethernet links. Moreover, if the router is in the source/destination subnets that enables it to deal with two ARP requests and two ARP replies for one ARP request from the source host, as can be seen in Figure 5.1 (b). As a consequence, the number of packets in the control plane is a function of the number of switches (N_{sw}) and N_{su}, i.e. $F(N_{sw},N_{su})$, if it is assumed that each middle subnet has one effective router.
On the other hand, to evaluate the number of packets in the data plane, the broadcast discovery mechanism in a legacy network leads to the parameters that affect multiple subnets being distinguished, which is the number of switches in the source and destination networks (N_{sw}), the number of links (N_{Lsw}) from those switches (except for 1, the incoming port in the broadcast process), the number of subnets(assuming 1 router for each) along the path from the source to destination (N_{su}) and the number of links (N_{LR}) from those routers (with no exception). As a consequence, the number of packets in the data plane in multiple subnets is a function of those parameters, i.e. $F(N_{sw}, N_{Lsw}, N_{su}, N_{LR})$.

5.2.3 Switches Vs router concepts

In legacy networks architecture, the connection between the subnets can be built by using another middlebox device, which is called a layer3 switch (L3_switch) in addition to the router. *L3_switch* is an intelligent switch with the functions of traditional router: it must be assigned an IP address for each interface [98], including the internal default gateway address; it can use routing protocol; it has the ability to record routings between subnets; a graphic user interface (GUI) is used to configure it; it modifies MAC addresses of the Ethernet frame in each hop; and decrements the Time To Live (TTL) in each routing packet [28]. As with a router, an L3_switch can be difficult to setup and manage in a large distributed network [99] as well as it having the same router limitations. In addition, the *multi-layer switch*, which is another type of middlebox device, is used to connect subnets providing low latency and high speed. It uses layers 2 - 4 to make forward decisions depending on the MAC address, IP, type of protocol and port number. On other hand, the *L2_switch* is used just inside collision domains (i.e. it cannot be used to connect subnets) to forward frames using the destination MAC address. In subsection (5.3.2), the L3_switch will be compared with the standard SDN switch that is used in the proposed model.

5.2.4 Multiple subnets legacy discovery protocol

Multiple subnets connect with each other to build a network, such as corporate networks, which can be centralised or distributed networks, with the latter being more scalable. In order to achieve a global view of the network, each router in it should have information regarding all the other routers in the network. This global view can be achieved either by intervention of the administrators through their knowledge about their network design or dynamically by using routing protocols. The global view is needed by the administrators to install static routes and by the routers to install dynamic routes inside their routing tables. The most well-known routing protocol is Open Shortest Path First (OSPF), which is better than static routing or even the Routing Information Protocol (RIP) [100]. OSPF has the advantages of loop-free topology and fast convergence time in an IP-based network. However, it is a complicated protocol to manage and configure as well as being difficult to troubleshoot. In addition, it needs higher processing and storage requirements in each router as well as time to configure *areas*, which is one of its main fundamental principles [101].

5.3 MSSERD DESIGN

In order to overcome the limitations and issues relating to legacy multi-subnets architecture, others' proposed mechanisms and architectures in distributed networks as well as meeting the requirements stipulated in section 5.1, MSSERD is deigned in this section focusing on connecting different subnets to build a large network depending on the principles of Ethernet technology without using middlebox devices. MSSERD's framework design involves removing the routers and default gateway settings completely from the SDN architecture and using proactive along with reactive behaviours in the presence of servers in the network for connecting between sources and destinations in different distributed subnets. In addition, MDP protocol is introduced, which is developed from LLDP to provide a dynamic flexible mechanism that gives a general view of the network. As ARP is the most important protocol in Ethernet network, the focus will be on its messages.

5.3.1 MDP design

One of the reason for the complexity of the network is related to the protocols in terms of their mechanisms and the number working concurrently, which makes them difficult to configure, manage and troubleshot [101]. Owing to LLDP being a well-known and highly efficiency protocol inside the SDN subnet for discovering its local switch, MDP is proposed for the MSSERD architecture, which is based on LLDP, for discovering both inside and outside the subnet in order to decrease the number of protocols used concurrently in the same network. Regarding MDP's design, ease of use and dynamic configuration have been taken into account, thus making it a standard discovery protocol for an SDN network. MDP is used in distributed SDN subnets to discover the whole network's IPs through an aggregation mechanism, which lets each SDN controller in each subnet to have the same general view over the network. This, in turn, gives MSSERD the ability to install rules proactively on SDN switches and also reactively to changes in subnet circumstances. MDP when compared with legacy discovery protocols overcomes their disadvantages and also has its own beneficial features, including not needing consumption time from the administrator to undertake management and configuration. In addition, it is applied in the SDN controller, so there need be no concern about the CPU and memory usage as this controller should be powerful computer. Furthermore, it can be scheduled to give the network's status and potential behaviour reports to the administrators at specific times, which makes troubleshooting the network easier. It has a significant Dynamic_reaction feature, which allows it to dynamically adjust/tune the sending of its multicast/unicast messages in order to minimise bandwidth consumption, where MDP defines the discovery_time parameter that is adjusted in reaction to the network size and behaviour. For example, the discovery_time is affected by network size in that it should be increased when the network size increases in order to give sufficient time for far subnets to send their information to all other subnets. Specifically, MDP sets the time for discovery_time parameter in each subnet depending on its location in the network. Moreover, if the network topology reaches a steady state, MDP increases the parameter value so as to reduce number of discovery packets and also the particular subnets change from exhibiting MDP multicast to MDP unicast behaviour. Another benefit is that, if any type of packet comes from neighbouring subnets, then the discovery_time parameter is reset because it gives a sign

that the next subnet is still alive. The administrator could also disable the dynamic behaviour and control the network manually.

5.3.2 MSSERD's SDN switches vs legacy Routers/L3_switches

MSSERD architecture has SDN switches, which can use layer 3 and layer 2 in its rules [89]. It is important to distinguish a Layer3_switch/router from an SDN switch, because both use layer 3, but have different mechanisms and this could be confusing.
Firstly, the SDN switch in MSSERD does not have IPs for its data plane interfaces nor does not need any configuration. In addition, it does not modify MAC addresses of the Ethernet frame for passing packets when it is used as juncture to connect its subnet to other subnets and simply passes packets the same as inside the subnet. The SDN switch that connects its subnet to other subnets is called an *Exit_switch* and the other SDN switches, which are just connected locally inside the subnet are termed *Internal_switches*, so as to distinguish them from each other. It should be noted that each type can change into another type at any time by changing the link-connected type (from locally to remotely and vice versa) for scalability or recovery purposes, amongst other reasons.

5.3.3 MSSERD general framework advantages/mechanisms

As a router is not used in either hardware or software in MSSERD's Ethernet architecture, this leads to the removal of the three limitations rules defined in subsection (5.2.1). In addition, with same network design, the proposed model has the advantage over legacy models in that it does not need to define *subnets management*. That is, regarding the routers in legacy architecture, in order to connect to another router in another subnet, both of them need to have external IPs which must belong to the same subnet that will be defined by using both outer edges IPs of the routers [100]. This subnet is different to both subnets' internal IP addresses and as consequence, when connecting two subnets three different subnets are needed in the legacy architecture. However, in MSSERD there are no IP interfaces in Exit_switches so it connects to other Exit_switches no matter whether they are in the same or different subnets. For example, to connect two subnets just two different subnets are needed, thus providing more scalability, flexibility and less complexity than with legacy architecture.

As explained earlier, ARP is important inside a subnet for translating IP addresses into MAC ones. Regarding MSSERD in this context, having eliminated the legacy broadcast mechanism, which is compulsory in legacy network's design, a mechanism for sending ARP packets also outside SDN subnets is introduced in order to get the destination MAC address in another subnet in the same network. There should be no concern about number of IPs used in the proposed model because, for example, there are 65,534 different IPs when use class B in the same network. In addition, IP version 6 can be utilised, which has plenty of IPs. As consequence, this is an improvement on legacy architecture in that it eliminates the need to modify the Ethernet frame at each hop in the network, which is time consuming. MSSERD uses both the SDN controllers and servers principle to answer ARP requests among different subnets, whereby the controller provides the general view of the whole network and installs switching rules proactively and reactively between sources/destinations and the servers, whilst the servers provide the requesting service, i.e. ARP service [102]. This mechanism of using both servers and controllers delivers a loop-free network and removes broadcast concerns outside the subnets. This in turn, leads to better performance and less complexity. MSSERD contains other mechanisms to perform its role, as follows.

5.3.3.1 ARP servers' location

MSSERD is proposed as having at least one ARP server in each subnet for a distributed network and the servers reply to any ARP requests, whether from their local subnet, as in [103], or other subnets in the same network. Server location can be anywhere in the subnet depending on the load, for example, connects to a switch that has high demanding traffic users to decrease access time or could be put in the network servers' pool, such as in a data centre that supports Network Functions Virtualization (NFV) architecture. In addition, by using more than one ARP server in a subnet, this allows for load balance and hence, better performance.

5.3.3.2 MSSERD operations mode

The first packet when entering the SDN switch, faces a delay because the switch sends it to the controller [103] for the MAC learning process. However, MSSERD uses

proactive behaviour to prevent this delay, thereby providing a better response time and giving the SDN controller the ability to respond proactively to any change in network conditions. MSSERD starts working in bootstrap mode to discover its local and abroad subnets, followed by two main different modes, which work concurrently for better efficiency. First, there is a proactive mode, which starts working after bootstrap time in order to install MSSERD's rules in the SDN switches (*Internal_switches* and *Exit_switches*) so as to be ready to forward packets between sources and destinations/ARP servers that are located in different/ the same subnets. Second, there is the reactive mode, which is triggered when there is a change in network conditions or after a specified time. It sends MDP packets to detect any changes and subsequently, triggers a proactive mode to install the required rules where necessary.

5.3.3.3 Handling failure

In any network, there are different types of failures that happen as a consequence of hardware/software malfunctioning. MSSERD, by focusing on connecting among subnets, means that Exit_switch failure is the greatest concern. MSSERD uses the same SDN switch element for all switches inside the subnet and so there is no difference between the Internal_switches and Exit_switches. In addition, it uses a redundancy link from different SDN switches to connect to the same next neighbour subnet. Those two parameters give dynamic failure handling, whereby if any Exit_switch has failed then the SDN controller will detect it by using the report status port feature from the next connected switch, which are available from the Openflow protocol [89]. Then, the SDN controller directly changes the working affected switches to use next priority rules, which will utilise the other redundant links to connect to the neighbour subnet. On other hand, if there is just one link with the next subnet and its Exit_switch has failed, the administrator, with the plug and play feature of Ethernet, can easily just plug its link into any other Internal_switches, which will be directly detected by the SDN controller as it has become an Exit_switch. Consequently, it can be seen this mechanism is completely different to the legacy architecture, where it is necessary to change one router with another and this consumes time configuring the new router, Figure 5.2 contains examples of possible connections to recover failures.

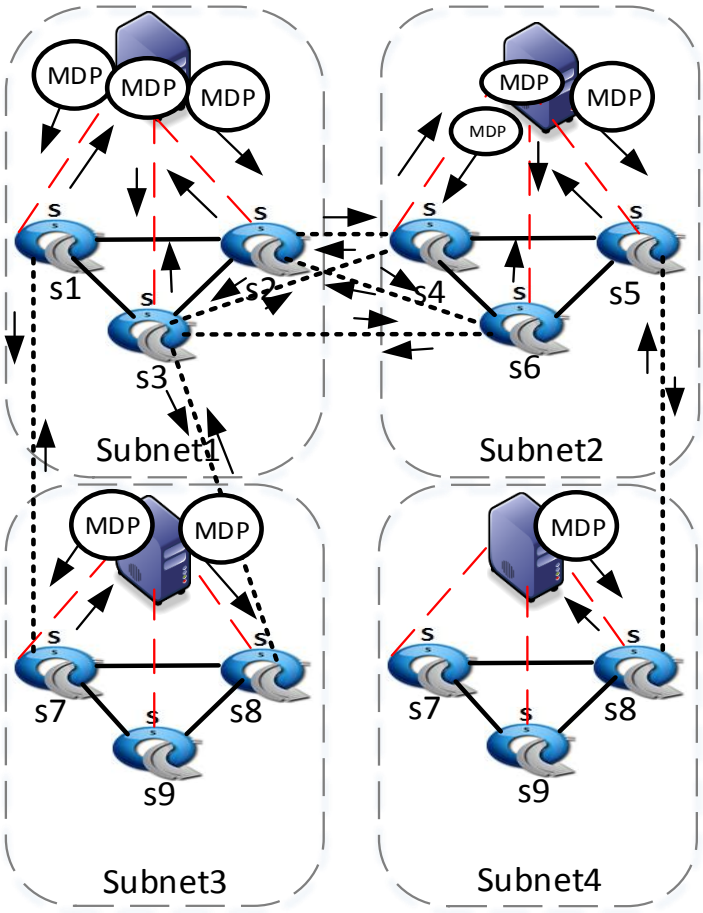


Figure 5.2: Examples of possible failure recovery connections among subnets and multicast/unicast MDP packets mechanism under MSSERD

5.3.3.4 Handling discovery

There are different elements to discovery in an SDN network that give it the ability to have a general view of its subnet and other subnets, which are described as follows.

5.3.3.4.1 Local hosts/servers discovery

Each host in each subnet when it joins is assigned an IP dynamically by using DHCP or manually. Subsequently, the host generates an ARP probe packets [93], which it sends to the SDN controller to register host information, including IP, MAC, Port and switch numbers. The controller then sends the same information to all the ARP servers in its subnet. If there is no activity from the host for a specified time, the controller will delete the host's entry from its discovery table and then update all the ARP servers to delete it too. In each subnet there is at least one ARP server; when one joins the subnet, it connects to the SDN controller that leads to the controller registering this server and starting to

send requests to it after installing the proactive rules. The SDN controller will check using an *echo* message whether the server is still available after each a specified time and if it is not, the controller will remove it and update all the affected switches with the new ARP server directions. It should be noted that this discovery mechanism for hosts and servers is proposed in [102].

5.3.3.4.2 Subnet/Switch discovery (join, leave)

First of all, as can be seen in Figure 5.2, each MSSERD controller in each subnet joining the network starts to multicast MDP packets until network steady state is reached, which depends on how many devices (switches in the local subnet and the number of subnets) in the network it needs to discover. In addition, an MDP multicast occurs, if there is any change in network conditions, such as adding a port for a switch. Then, the MSSERD unicasts MDP packets at specified time intervals, according to the Dynamic_reaction feature. These MDP packets are used to discover the local switches (as same as LLDP), next neighbour subnets and far subnets in same network.

If a new switch **joins** the local subnet, the controller multicasts MDP packets from all the output ports in that switch (e.g. S1) involved in its subnet. Then, it receives the same MDP packets sent back to it from other switches, which leads to the controller being aware that S1 has a direct connection with them. In addition, the MSSERD controller can calculate the link-delay between each pair of the direct connection switches to help it to find the best path between the sources and destinations among multiple switches in same subnet by using the Dijkstra algorithm. After this, MSSERD stops local multicast MDP packets and relies on the report status port mechanism to trigger the reactive mode. Moreover, if a switch **leaves,** the controller directly detects this through the OpenFlow protocol's status report.

If a new neighbour subnet **joins** the network, the MSSERD controller in it carries its own information and sends multicast MDP packets. These packets will be sent by Exit-switches to the next neighbour subnets that will register that information as next neighbour information in the Exit_switches_discovery table . In addition, it will timestamp received MDP packets to use for calculating the Link_delay with neighbours. The Exit_switches_discovery table is updated after a specified time, depending on the

number of next neighbour subnets and the Dynamic_reaction feature /administrator's decision.

If a new far subnet **joins** the network, the MSSERD controller in that subnet will send its information to the next subnet, which while filling the Exit_switches_discovery table starts to fill a Subnets_discovery table (IPs of the source and destination subnets, subnet Mask source and destination, and link_delay). Then, it exchanges this table information with its next neighbours in dictionary format, for example subnet1 :{subnet 2:delay_X, subnet 3:delay_Y}, which means subnet1 connects separately and directly with subnet2 and subnet3 by a link delay equal to delay_X and delay_Y, respectively. When the next subnet receive the MDP packet it will update its Subnets_discovery table, if there are any changes and add its own connections with other subnets as well as sending the new MDP packet to next neighbours and so on, in an aggregated way. For example, in Figure 5.2, when joining subnet4, after period of time when the network reaches steady state, all the subnets' controllers will send the same MDP packets, which have all the subnets' information i.e. sub4_inf + sub2_inf + sub1_inf+ sub3_inf and has the same Subnets_discovery table. If there is no change in the received MDP and in the local information, then MSSERD will not multicast MDP to save bandwidth and prevent congestion on network and just send unicast messages to neighbouring subnets at the specified time to say *I'M Alive without change*, which uses just the subnet ID.

On the other hand, if the next neighbour subnets leave the network, the MSSERD controller in the neighbouring subnet will detect this by checking the Expire_time field in the Exit-switches table, which records the last time it received an MDP multicast/unicast packet from that subnet. After Expire_time reaches zero the neighbouring subnet controller will send a ping to the controller in that subnet three times and await a reply. If no reply arrives, it will delete all of its information from the Subnets_discovery table, the Exit_switches_discovery table and all the rules that were installed in the Exit_switches to direct packets to it. While if a far subnet leaves, the controllers in other subnets will detect this by monitoring the Subnets_discovery table and if there is an entry that has subnet information that is not updated after a specified time, then that entry will be deleted from the table in every controller and so too, every rule which relate to that entry in the switches.

5.4 IMPLEMENTATION FOR MSSERD

In this section, MSSERD implementation for SDN networks is covered in detail in relation to dynamic discovery and handling ARP packets both within and out of subnets, as an extension to Ryu's OpenFlow controller and by using an Open Virtual Switch (OVS). All the requirements set out in section 5.1 are met by MSSERD, and in what follows its components are described.

5.4.1 MSSERD's rules implementation

In order to manage the connection between sources and the destinations locally or remotely, two types of switches introduced above, namely, Exit_switches and Internal_switches, have two forms of MSSERD rules. These are, proactive rules, called Switch_proactive rules that are installed in switches in the proactive mode and reactive rules, termed X_packet_L2 rules, which are installed in switches, if they are chosen as belonging to the best path during the processing of the ARP component, as described later in subsection (5.4.4).

5.4.1.1 Switch_proactive rules

These are used to deal with ARP packets in order to get the destination's MAC address. They are designed to transfer ARP requests/replies inside the same subnet (i.e. locally) and among multiple subnets (remotely). There are request rules and reply rules according to the different ARP packet types. After bootstrap time, the MSSERD controller installs these rules in the Internal_switches and Exit_switches in proactive mode. The ARP request rules have the highest priority in the Internal_switches, while ARP reply rules do so in Exit_switches. It should be noted that if the subnet has one Exit_switch, the Internal_switches need just one ARP request and reply rule per switch to connect to the other subnets, because the rules will depend on packet type, i.e. ARP type, no matter what the subnets' IP. To provide a better understanding in relation to how these rules are working inside OVS switches, Table 5.1 has been created, which describes this in detail and the flowchart of manual packet tracing, as shown in Figure 5.3, traces the packets from the OVS point of view. This type of rules appears in Figure 5.4 as $\longrightarrow$ for clear tracing and hence, better understanding.

Table 5.1: MSSERD's Switch_proactive rules strategies in Exit-switches and Internal_switches to forward packets to their destinations locally and remotely

Rules in Internal_switches					
ARP request received packet			ARP reply received packet		
Source IP in received packet	Destination IP in received packet	Rule direction	Source IP in received packet	Destination IP in received packet	Rule direction
local	local	Go to the ARP server	local	local	If the switch connects to an ARP server→ Go to the controller, else use x_packet_L2 rules
local	remote	Go to a specific Exit_switch	local	remote	Go to a specific Exit_switch
remote	local	Go to the ARP server	remote	local	Go to the controller
remote	remote [i.e. middle subnet between source (Src.) and destination(Dst.) subnets]	Go to a specific Exit_switch	remote	remote	Go to a specific Exit_switch
Rules in Exit_switches					
ARP request received packet			ARP reply received packet		
Source IP in	Destination IP in received packet	Rule direction	Source IP in	Destination IP in	Rule direction

received packet			received packet	received packet	
local	local	Go to the ARP server	local	local	Go to the controller
local	remote	Go to a hop in next subnet	local	remote	Go to the controller
remote	local	Go to the ARP server	remote	local	Go to the controller
remote	remote (i.e. middle subnet between Src. and Dst. subnets)	Go to a hop in next subnet	remote	remote	Go to a hop in next subnet/ Go to controller

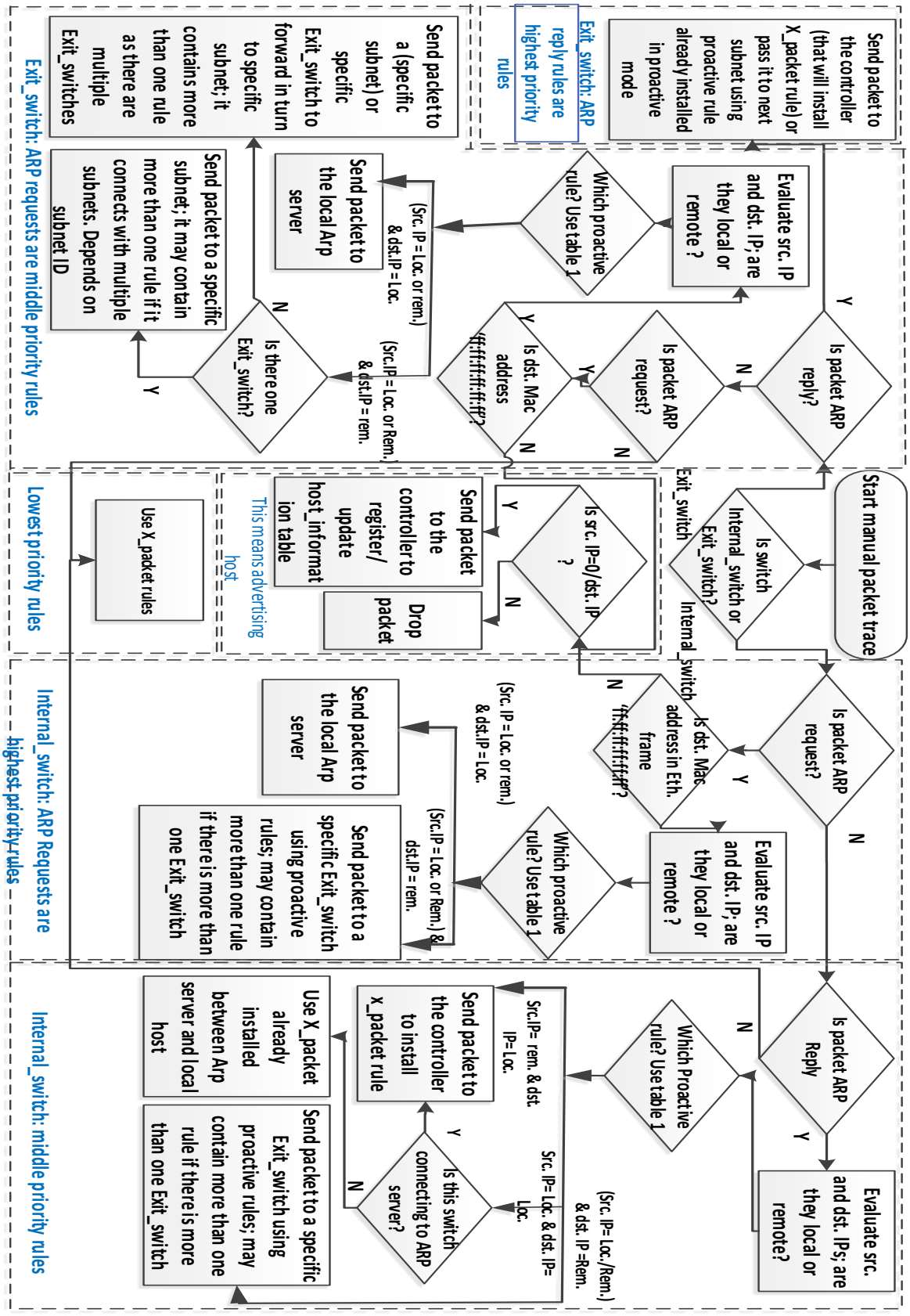


Figure 5.3: A flowchart showing the manual tracing of packets from an SDN switch view point; It contains steps how switches deal with incoming packets under MSSERD, and the priority of MSSERD's rules

5.4.1.2 X_packet_L2 rules

These are rules installed in switches (no matter whether they are Internal_switches or Exit_switches) during the ARP reply processing time using an ARP component. This is called an X_packet_L2 rule, because it forwards any type of packet (X) between sources and destinations located in the same or different subnets by using just layer2 MAC rules. It should be noted that the controller will install X_packet_L2 rules after forwarding the ARP reply packet to its destination (the host that makes the request), i.e. installed X_packet_L2 rules do not affect the ARP response time. For a better understanding of the X_packet_L2 rules, it is shown in Figure 5.4 with an arrow $\rightarrow$.

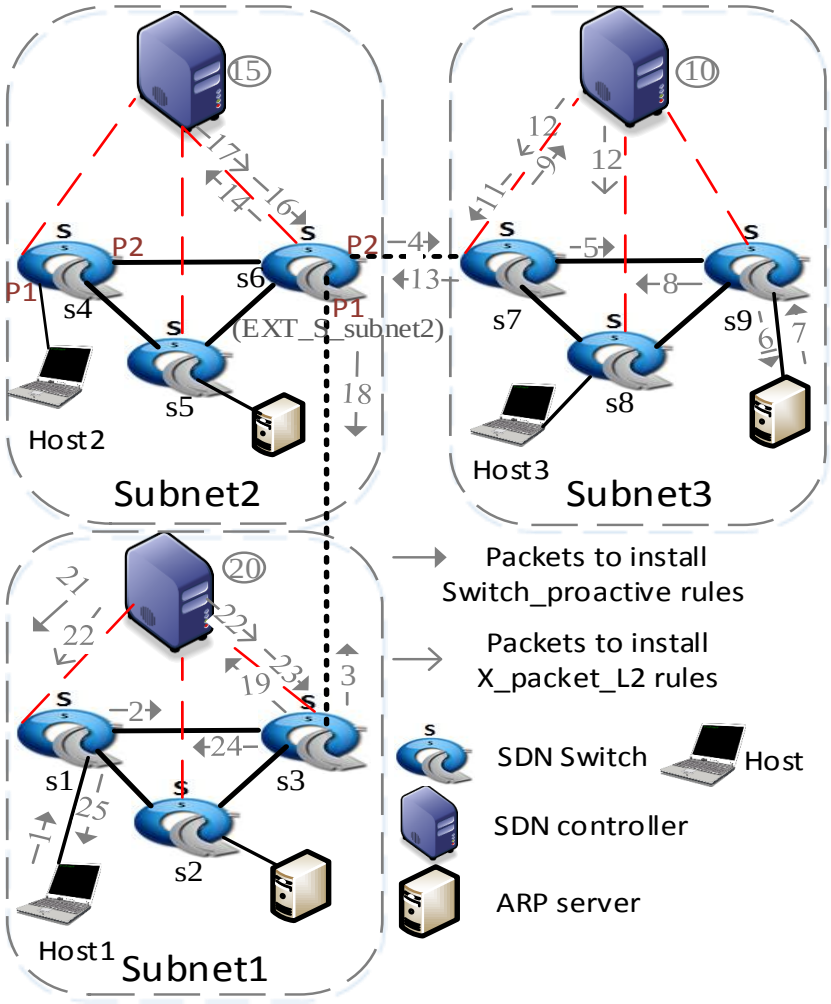


Figure 5.4: MSSERD's rule types and steps tracing the ARP process between subnet1 and a far subnet (subnet3)

5.4.2 MSSERD's bootstrap, proactive and reactive components

MSSERD contains three different operation modes that are bases on each other, as can be seen full detail in the flowchart in Figure 5.5. Firstly, the MSSERD controller starts bootstrap mode by creating tables to manage its subnet and discover abroad subnets (next neighbouring and far subnets). That is, it creates a set of local topology tables, including a *Host_information* table, *Server_information* table and a *Switch_ discovery* table, which are dictionary based tables that store all the local subnet information. In addition, it establishes an *Exit_switches discovery* table, which discovers local switches that connect the local subnet to its next neighbour subnets containing: Exit_switch_ID, Exit_port_number, Next_neighbour subnets_IP/ID, Link_delay, local timestamp and expiry time. Moreover, MSSERD creates a *Subnets_ discovery* table, which stores far subnets information, including: Source_subnet_IP, Source_subnet_Mask, Destination_subnet_IP, Destination_ subnet_Mask and Link_ delay. This table is used by the MSSERD controller to draw the General_view_topology, which has all the subnets around it, whether next or far, which helps the controller to decide which Exit_switch it should forward the packet to and then it installs rules in the switches accordingly.

After creating all these tables, MSSERD starts the process of filling them up. it registers any joining switch, which exchanges with the controller the Openflow negotiation messages. Then, the controller multicasts MDP packets to all connected switches, such that they can reach both local switches and those switches in the next neighbour subnets. After that, the controller keeps listening for any MDP returned packets and uses them to fill the discovery tables, as is explained in relation to MDP implementation in subsection (5.4.3). The servers will then be registered in the *Server_ information* table, so that this information can be used during the next steps.

The proactive mode is established next which pertains to analysing the filled tables and installing rules in SDN switches. The controller starts this mode by examining the Exit _switch table to make a dictionary (Exit_switch ID): {next neighbour subnet ID: link_delay}. In addition, it analyses the Subnets_discovery table to make a dictionary of: "Which subnet connects to which subnet with what delay" (i.e. source subnet ID: {destination subnet ID: link_delay}). After this, the controller starts to choose switches one by one or in parallel by using a multithread concept to find the best path from each switch to the ARP server and then installs rules to serve local requests. At the same time, it finds the best path from each switch to the Exit_switches to serve abroad requests, depending on the dictionaries compiled in the previous steps. The number of rules

installed in each switch depends on the number of subnets, however, if there is just one Exit_switch in a subnet, then one rule is enough regardless the number of subnets.

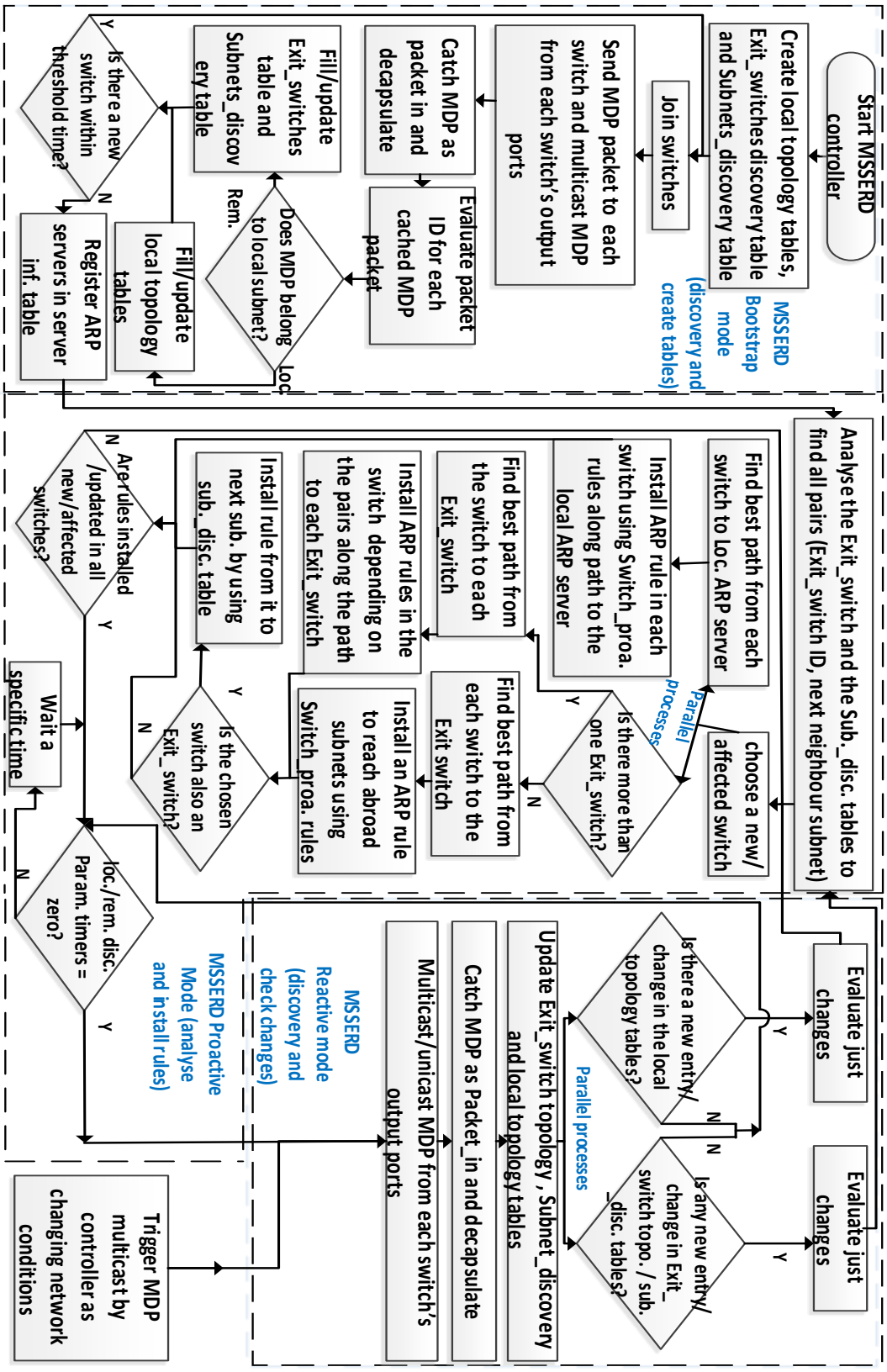


Figure 5.5: A flowchart of MSSERD's operation mode components

In is way, there is a rule installed proactively in each switch for each external subnet, no matter if it is a far or the next subnet the aim is to make one rule in OVS to connect to each set of subnets. It should be noted that if the switch has dual operations, i.e. works as Internal _switch whilst at the same time working as Exit_switch, first, the controller installs in it the rules for abroad subnets and then, those appropriate for the local subnet. After this, the subnets are now ready to answer requests. The MSSERD controller repeats multicast MDP, if there is a change in the network conditions or performs unicasts at specified times in reactive mode so as to ensure that it stays up-to-date with any changes in its subnet status and those of abroad subnets, such as subnet leaves/joins and switch leaves/joins.

5.4.3 MDP component implementation

In the MSSERD controller, MDP plays the salient role of allowing it to discover other subnets in its network. It starts working during bootstrap time and reactive time in the proposed model. It has LLDP abilities plus several others that are not available in that protocol. It discovers the next neighbor subnets, far subnets and it sends crucial information that help controllers choose the best path, because it can be used to calculate cost, including link delay and link capacity. The implementation of MDP contains two parts. The first contains a new type-length-value (TLV) class (MSSERD class with TLV number 125) and then, two subclasses are added (Local_Subnet_ID and All_Pairs_subnets. The Local_Subnet_ID represents the subnet ID, whilst the All_Pairs_subnets pertains to a hash table of pairs of connected subnets in the same network as well as the link cost. For the second part, an algorithm to discover the local subnet, next neighbour subnets and far subnets as can be seen in algorithm 1 is implemented.

To discover the local and next neighbor subnets, the MSSERD controller generates MDP packets with just first the subtype (Local_Subnet_ID). Then, it multicasts those packets to its local subnet and to the next neighbour subnets. After that, the MSSERD controller listens to catch MDP packets as Packets_in. The controller will check whether the packet is sent by itself and then it will use it to update the local switch_discovery table. If the packet does not belong to itself, it will be used to add /update the subnet_discovery table and the controller will identify the switch that entered that packet

as an Exit_switch and put its information in the Exit_switches_discovery table. The controller will repeat this procedure in a concatenating with a reactive operation mode, if there is a change in its subnet conditions.

On the other hand, to discover far subnets, the MSSERD controller waits for the specified time, which must be sufficient for each controller to discover its local subnet's switches and next neighbour subnets. Subsequently, the controller uses the Subnet_discovery table to create a dictionary, which contains just its subnet pairs and link cost with its next neighbour subnet (i.e my_subnet_id:{next_subnet_id:cost}). The controller then creates MDP packets, especially to help discover far subnets and uses the dictionary as the parameter for All_Pairs_subnets, whilst it uses local controller IP as a parameter for Local_Subnet_ID. The MDP packets will be unicast over all Exit_switches. Each MSSERD controller keeps listening to all MDP packets, if one is from another subnet, then the received dictionary is used to update its subnet_discovery table. Then, the updated Subnet_discovery table will be used to make a new dictionary and the MSSERD controller repeats the procedure to send the new dictionary for all next neighbour subnets, as can be seen in algorithm 2. After a period of time all the controllers in all the subnets will have the same dictionary and same Subnet_discovery_table.

Algorithm **1 Discovering local and next neighbour subnets**
Input: Local_Subnet_ID

Output: MDP packets with Local_Subnet_ID,
Update Switch_discovery table,
Update Subnet_discovery table,
Update Exit_switch_discovery table.
START
declare Local_Subnet_ID
SET Local_Subnet_ID to specific IP, got automatically from the system
REPEAT
Generate MDP packets with just Local_subnet_ID

Multicast/unicast the packets from each output port in each switch.
Listening to Packet_in to catch MDP packets.

Decapsulating each MDP packet

Read IP address from each decapsulated packet

IF Local_Subnet_ID = (packet_ip AND Mask) then

Update Switch_discovery table.

ELSE

Update Subnet_discovery table

Update Exit_switch_discovery table.

END IF

Wait the specified time or wait for any change in subnet conditions

UNTIL terminated by the administrator

Algorithm **2 Discovering far subnets**

Input: Local_Subnet_ID

Outputs from the first iteration of algorithm 1

Discovery time parameter

Output: MDP packets with Local_dict. and Local_Subnet_ID ,

Update Switch_discovery table,

Update Subnet_discovery table,

Update Exit_switch table.

START after first iteration of algorithm 1

DECLARE global Local_dict

Local_dict ← **CALCULATE** pairs of (my_subnet: next_subnet, cost)

REPEAT

Generate MDP packets with Local_subnet_ID and All_pairs_subnets(Local_dict.).

SEND unicast MDP packets over all Exit_switches

LISTEN to Packet_in to catch MDP packets

DECAPSULAT each MDP packet

READ IP address from each decapsulated packet

IF Local_Subnet_ID ≠ (packet_ip AND Mask), then,

Update Exit_switch table

Update Subnet_discovery table

Local_dict ← **CALCULATE** pairs of next neighbour subnets

Merge dictionary from decapsulated packets with local dictionary

Local_dict ←local_dict + Decapsulated_dict

DELETE any repeated pairs

Else

Update Switch_discovery table

END IF

WAIT the specified time, as defined by the Discovery_time parameter

UNTIL terminated by the administrator

5.4.4 ARP component implementation

In the proposed model, to complete an ARP process, Switch_proactive rules are installed in the proactive mode that is supported by the reactive actions of using an ARP component. The MSSERD controller contains an ARP component, which deals with ARP request and reply messages from both the local and abroad subnets to acquire the destination's MAC address for the requesting host and also to install the Ethernet-channel/mix channel (it is the path that uses just layer 2/a mix of layer2 and layer3) between sources and destinations. The ARP component starts listening to incoming packets after bootstrap mode and after the first set of iterations in proactive mode in order to detect whether a new Internal_switch or Exit_switch is joining (i.e. by catching packets that are not caught by the discovery reactive mode). It then updates the Host_information table and installs X-packet_L2 rules. The ARP component can deal with two types of ARP request/reply depending on its destination. This could be a Local_ARP request/reply, with the destination host in the same subnet of source host or an Abroad_ARP request/reply. The latter is defined as an ARP request /reply sent to the next neighbor subnet or to a far subnet so as to obtain the destinations' MAC addresses. To clarify the functioning of this component, the example of a host needing to connect to another host at a different location is taken (locally, next subnet and far subnet). The full details of the connection steps are available in Figure 5.4 and the ARP component architecture is provided in the flowchart in Figure 5.6.

1) Firstly, if the source and destination are in same subnet, then the source host (Host1) sends an ARP request message with its source IP and MAC addresses in the packet's source IP and MAC fields, respectively. Whilst the destination IP field is equal to the destination host's IP address and the destination MAC field equals the broadcast address '00:00:00:00:00:00'. The switch connected to Host1, (S1), receives this packet and uses a Switch_proactive rule that is already installed in proactive mode. The switch decapsulates and compares the destination IP in the received packet with network ID field in the installed rule. As the IP of the destination host is located within the same subnet ID of the local subnet, then the nearest connected switch (S1) will forward this packet to the local ARP server using proactive rules. The ARP server then answers with an ARP Reply. This packet goes to the MSSERD controller that triggers the ARP component in order to find and install the best path among to switches between the ARP server and Host1, which subsequently forwards the ARP Reply packet to it using the X_packet_L2 rules.

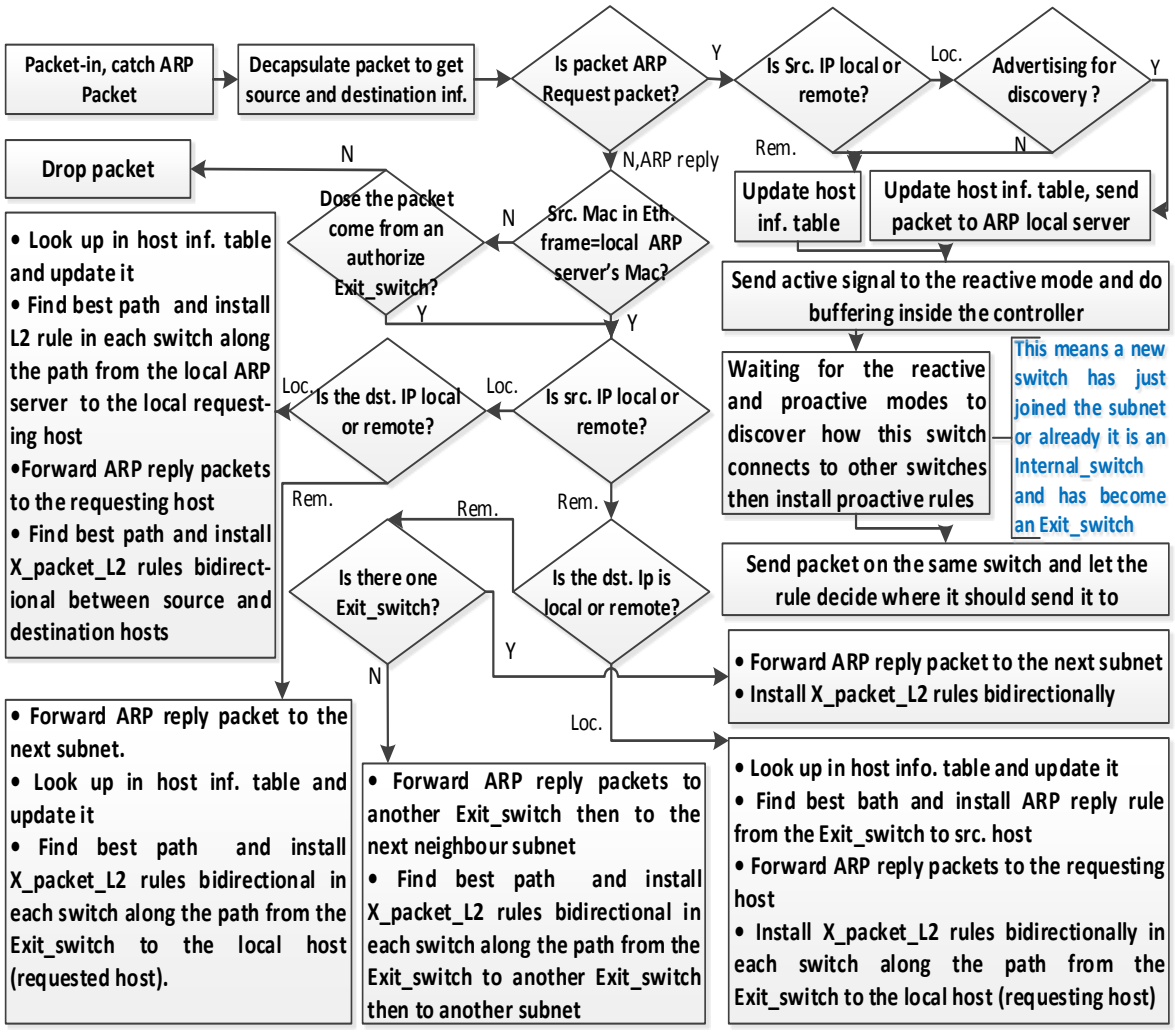


Figure 5.6: A flowchart of MSSERD's ARP component

2) Secondly, for the destination host (Host2) located in the next neighbour subnet, Host1 generates an ARP request with the same fields that are generated locally. It should be noted that this is completely different to what the host does in a legacy network, for as explained before in section 5.2, when a host needs to connect to a destination outside its subnet it has to first know the default gateway's MAC address.

Hence, the host in legacy architecture initially sends an ARP request with the gateway IP and not the host destination's IP. In the proposed model, if the IP of the destination host in the ARP request packet is located outside the subnet (subnet1) (e.g. the destination is in the next subnet) and if there is just one EXIT-switch, switch1 (S1) will use the Switch_proactive rule to forward this packet to that EXIT-switch (S3). However, if there is more than one EXIT_switch, switch1 will look up in its forwarding rules which rule matches the destination IP of the packet and then, will forward the packet to the specific Exit_switch that leads to subnet2. All Internal_switches between S1 and S3 follow the same procedure. MSSERD has already installed in each Exit_switch (if there is more than one) the rules by depending on the general view of all the network topology in order to forward the ARP request packet to the next hop in the next subnet (subnet2).

When the ARP request packet reaches Exit_switch (EXT_S_subnet2) in the next neighbour subnet (subnet2), the EXT_S_subnet2 forwards the packet to the local ARP server using the proactive installed rules along all path from the EXT_S_subnet2 to the local ARP server in subnet2. Then, the ARP server looks in its ARP_serving table and drops packet if there is no record of such a destination host (Host2) or generates an ARP reply packet with an Ethernet frame containing the MAC source field equal to the ARP server's MAC address and the destination MAC field equal to the requesting host MAC address (Host1) that made the request. This is different to the legacy architecture mechanism, where the destination MAC field equals the default gateway's MAC address.

In addition, in the proposed model the source IP and MAC fields in the ARP frame are equal to the IP and MAC addresses for Host2 and the destination IP and MAC fields are equal to the IP and MAC addresses for Host1. As a consequence, It can be seen from the reply packet frames that in the proposed model there is no difference between the connection to outside the subnet or inside the same subnet, which should minimise complexity on the host side. Subsequently, the ARP server in subnet2 forwards the reply packet to the nearest connected switch (S5). This Internal_switch will use the proactive rule to forward the reply packet to a specific Exit_switch (i.e. EXT_S_subnet2), depending on the destination IP (Host1). It should be noted that it is not obligatory for

subnet2 to use the same Exit_switch (which entered the ARP request) to send back the ARP reply, if it has more than one connection with subnet1 using different Exit_switches, which leads to more flexibility regarding the recovery and load balance processes.

The EXT_S_subnet2 will send any ARP reply packet coming to it to the MSSERD controller by using middle priority rules of the ARP reply in Exit_switch, see Figure 5.3 showing the manually traced packet. The controller decapsulates the ARP reply and recognises that the destination IP is outside its subnet. Then, it gets the source and destination MAC addresses from the ARP reply packet and looks in the host-information table to get the Internal_switch and its port that connect to the local requested host (Host2). The controller then use this information to install bidirectional layer 2 rules (X_packet_L2) in the switches along the path between Host2 and a specific Exit_switch (i.e EXT_S_subnet2). For example, the rule in switch (S4) in subnet2 will be "*Any packet (X) that has a destination MAC address equal to Host2's MAC address goes to the port (P1)*". In addition, it installs X_packet_L2 rules (in the same way as above) in EXT_S_subnet2 to exchange any type of future X packets to Host1 and Host2. Then, the controller will forward the packet to the subnet1using the Packet_out message of Openflow .

The Exit_switch in subnet1, after receiving the ARP reply will forward the packet to the MSSERD controller in subnet1, which leads to triggering of the ARP component. The MSSERD controller will receive the packet as a Packet_in and decapsulate it determine whether it is an ARP reply, whether or not it is from a local ARP server, and whether it is coming from an authorised Exit_switch (meaning there must be a match between the pairs next neighbour subnet and the Exit_switch), for security purposes. The controller then checks if the destination IP/ MAC in the ARP packet is going to one of hosts under its service, subsequently ascertaining whether the source IP came from another subnet. Then, MSSERD will look up the host information table, update it, find the best path and install the ARP reply rule as well as the X_packet_L2 rule in each switch along the path from the Exit_switch to Host1. Next, the controller will forward the packet to the next switch using a Packet_out message. Then, the ARP reply packet goes through all the switches using the stored rules until it reaches Host1. As a consequence, the subsequent packets between Host1 and Host2 will use layer2.

3) Thirdly, the destination host (Host3) is located in a far subnet (subnet3), with Figure 5.4 describing in detail the steps to connect between Host1 and Host3. In addition, for the

trace the manual trace figure, Figure 5.3, can be used to understand how each switch deals with incoming packets and how to forward that packet as well as being utilised to understand the sequence of priority of the rules inside each Internal_switch or Exit_switch. In this case, the same procedure will be followed as in 2 (i.e. previous point) for each source subnet (subnet1) and destination subnet (subnet3). Regarding the middle subnet (subnet2), when the Exit_switch receives an ARP request, it uses the proactive rule to determine the source and destination IPs, to forward the packet to the next neighbour subnet (subnet3).

This procedure will be repeated in each middle subnet until the ARP request packet reaches the destination subnet (subnet3). Then, subnet3 will answer with an ARP Reply message from its ARP server to a specific Exit_switch, as describe in 2 and subsequently, sends it to the controller, which then forward it to the nearest middle subnet (subnet2). After this, the Exit_switch in the middle subnet receives the ARP reply packet, having two different mechanisms depending on the requirements, available hardware features and administrator decision, i.e. if he/she needs subsequent X packets to forward using layer3 or layer2 in his/her subnet if it works as middle subnet.

- Layer3 usage: the Exit_switch already uses Switch_proactive rules to forward ARP reply packets to the next subnet proactively, which prevents such packets being sent as Packet_in to the controller in subnet2 and in turn, all subsequent X_packets will use the same mechanism. This mechanism has the advantage of no overhead in the control plane for all middle subnets between the source and destination subnets as well as no rule for each X_packet. This is supported by switches' features currently, because they are processing layer3 with the same latency as layer2 [104].
- Layer2 usage: the Exit_switch sends an ARP reply packet as a Packet_in to the controller. The MSSERD controller decapsulates it and uses its information to install the X_packet_L2 rules in all affected Internal/Exit switches (i.e. EXT_S_subnet2) that will forward the packets to the source subnet (subnet1). Moreover, so as to be able to forward any type of packet to Host1 and Host3 in the future by using layer2 rules. The controller then passes the packet to the EXT_S_subnet2 that will forward it to the next hop in the next neighbour subnet (subnet1).

If there are multiple middle subnets between the source and destination subnets, all the middle subnets follow the previous procedure until the reply reaches the source subnet. The Exit_switch in subnet1 is not cognisant of whether the reply came from the next neighbour subnet or from a far one. It directly forwards the packet after encapsulating it through the Openflow protocol to the controller, which will install the rules along path to the Host1 and forward the packet to its destination.

There are some other decisions that are taken by the MSSERD controller, as can be seen in the ARP component flow chart. For example, if the packet coming to the controller as Packet_in is an ARP request, the controller by checking the IP field can recognise whether a new switch has joined its subnet or an Internal_switch has become an Exit_switch. As consequence, this leads to the triggering of the Fast_active signal, which is used to interrupt the waiting timer for reactive mode, so as to get it to start sending MDP packets to discover what new events have happened in the subnet. In addition, the controller uses the ARP request coming with the IP source address equal to the IP destination address or IP source equal to zero when discovering the hosts in its subnet.

5.5 Experimental results

In this section, different types of experiments on authentically built testbed have been designed and implemented, with the results demonstrating improved performance and efficiency by using the proposed scheme when compared to legacy architecture. Firstly, the testbed is built from scratch and it consists of hardware and software parts. The former consists of twenty three computers, Ethernet cables with different lengths (1, 2 and 3 metres), and Ethernet LAN cards with a speed of 1000 Megabits per second. In more detail, twenty of the computers have specifications of Core 2 Quad, 2.66 GHz and 2.0 GiB memory and can be used as SDN controllers, OVS switches, legacy routers or hosts. In addition, there are two computers with the specifications of core i7, 3.4 GHz and 3.8 GiB memory, which are used as powerful SDN controllers when performing load experiments. Finally, a Samsung laptop with specifications of core i7, 2.20 GHz and 7.8 GiB memory is used as a host or ARP server depending on the particular experiment. The software part comprises Ubuntu version 14.04, which is installed on all computers and updated with all required libraries, OVS and Ryu's SDN controller, as the network

operating system (NOS). In addition, all the SDN switches and SDN controllers use the Openflow protocol, which is a well documented protocol and open source platform. The logic diagram for 23 PCs connected together can be seen in Figure 5.7

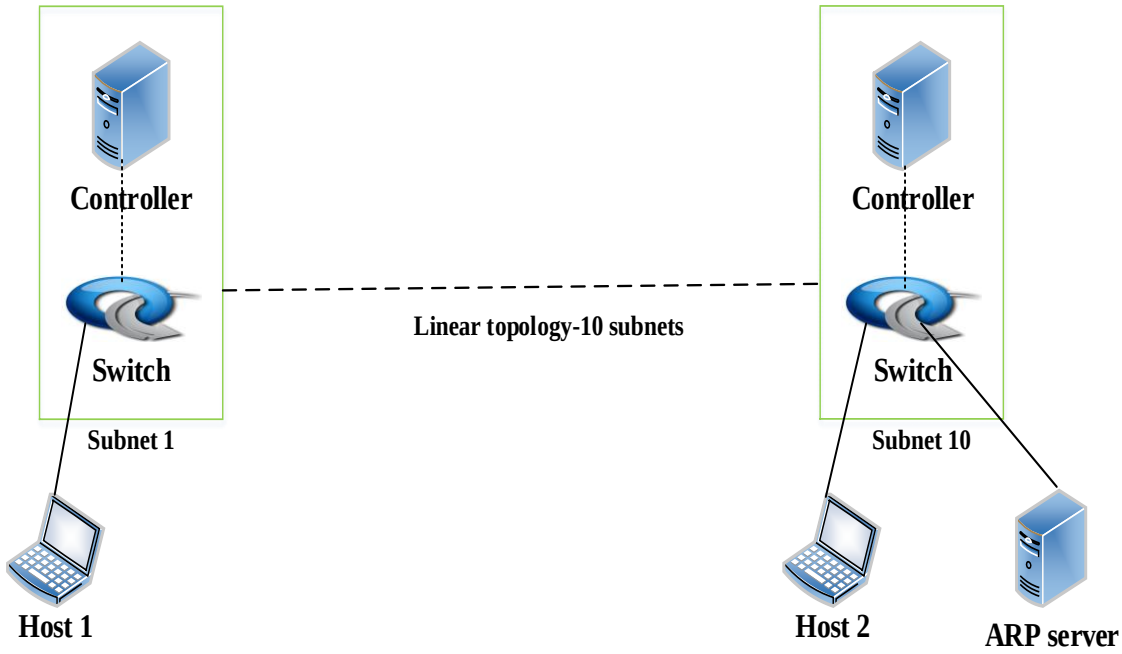


Figure 5.7: Logic diagram for built testbed that has 23 computers

To verify the efficient performance of the proposed model with scalability, different experimental scenarios were designed and implemented, with comparisons being made between the legacy router-based and MSSERD schemes. There are two parts in this section, with the first dealing with two subnets, whilst the second deals with multiple subnets (i.e. 2-10 subnets).

The first part has two experiments concerning the response time and reliability during loading. A linear topology is utilised with four PCs used as switches, two PCs as legacy routers, two hosts, two controllers and one ARP server are used for these particular experiments.

- The first experiment is designed to measure the response time when increasing the number of requests per second, for both MSSERD and legacy router-based architecture.
- The second experiment considers the reliability of both architectures. This is performed by evaluating the percentage of packet loss when increasing the number of requesting load packets.

In the second part, three experiments are performed to measure the number of packets in the control and data planes, number of Ethernet frame changes as well as the response time regarding bootstrap discovery time and rediscovery time in relation to network scalability. A linear topology is used comprising ten PCs working as legacy routers or as SDN switches, ten PCs used as SDN controllers, two hosts and one ARP server.

- The third experiment pertains to measuring the number of packets in the control and data planes, in general and in the legacy routers and Exit_switches, specifically, when generating one ping, whilst increasing the number of subnets under the traditional router-based scheme and the proposed model scheme.
- The fourth experiment is undertaken to measure performances in relation to network scalability, by increasing the number of subnets and evaluating the response time between the source and destination subnets in both legacy and the proposed schemes.
- The fifth experiment is run to measure the subnets' discovery and re-discovery times with the proposed model using the MDP protocol.

5.5.1 Performance with load: comparison between MSSERD and legacy router-based architecture

A linear topology with fix two subnets, each having two switches and one legacy router and the destination subnet having two hosts (one of them used as ARP server in the proposed model), is deployed to evaluate how the response time is affected by load on routers in a legacy network and by Exit_switch in the proposed model, after removing the routers and default gateway settings. A different load rate, called Load-requests, is generated from 10 virtual hosts on the router and Exit switch in the source subnet in the legacy and the proposed models, respectively. Then one ping is sent during the load and the response time in both architectures is evaluated. The legacy model response time statistics, in Figure 5.8, show that when increasing the load on the router in the source subnet from 17 Request per Second (RPS) to 100 RPS, the ping response time grows slightly from 27.6 ms to 34.6 ms. However, when the load is increasing to 1000, 2000 and 3333 RPS the response time is significantly increased, reaching 94.2 ms, 125 ms and 1054 ms, respectively. The reason for this trend is that the router in the destination subnet,

when it receives Load-Requests from source subnet, will broadcast an ARP request per each one coming from the source subnet in order to discover the destination's MAC address so as to be able to pass a Load_Request packet to it. That broadcast mechanism puts load on all the links in the destination subnet, which leads to congestion, especially if the destination host does not respond. This leads to the router retransmitting the request three times to discover the destination's MAC address, which in turn, affects the Reply packet of the ping request in terms of the response time and this will increase with increasing load.

However, in the proposed model, load has a negligible impact on response time when scaling the network, with an average response time of 17.38 ms. That is, there is increased scalability inside each subnet such that the number of hosts within can be increased. This happens in the proposed system, because the SDN Exit_switch in the destination subnet already has an installed proactive rule to switching any Load_Request to the ARP server, which in turn either replies or drops the packet, if there is no matching entry.

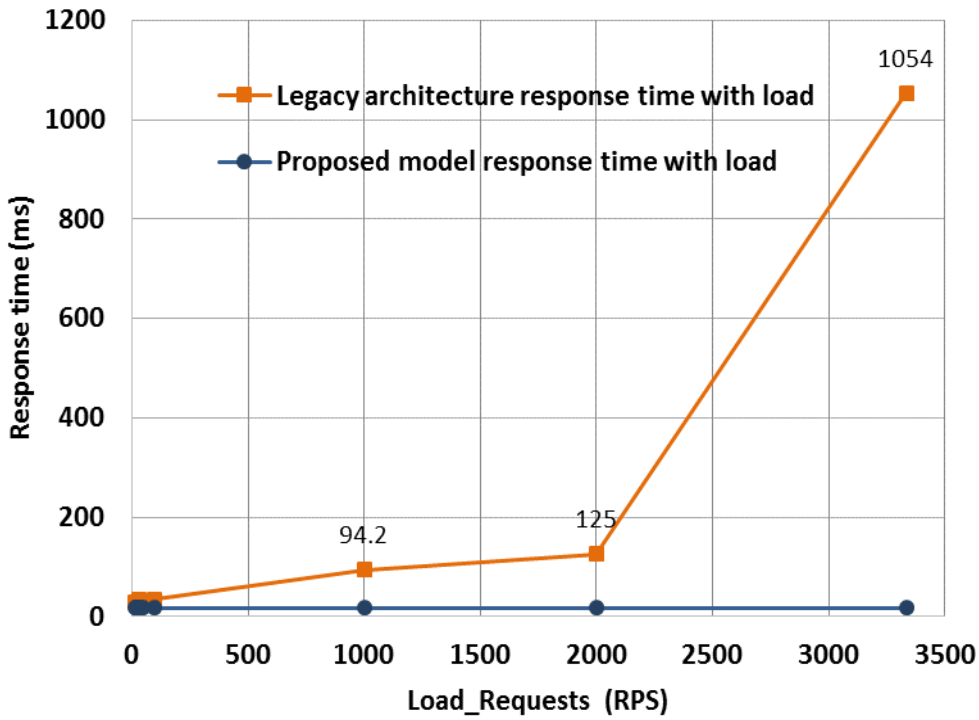


Figure 5.8: Response time during traffic loading: comparison between MSSERD and legacy router-based architecture

5.5.2 Reliability during loading: comparison between MSSERD and legacy router-based architectures

In this experiment, the reliability has been evaluated using packet loss percentage. The topology that is used in this experiment is the same as in previous one and 10 ping requests are sent from the source to the destination subnet whilst loading the network from 10 virtual hosts with a fix rate 3333 RPS during a specified time to generate different numbers of packets. Regarding legacy architecture evaluation, Figure 5.9 shows that with a load less than 8000 packets there is no packet loss, while it has 30 % packet loss when it is between 8000-11000 and this increases to 100% when the load reaches 15,000 packets. This occurs because increasing the number of packets with a rate of 3333 RPS will make the destination network too busy to answer each request and load occurs on both source and destination routers. Moreover, the worst case scenario is when the destination router does not reply to the source router, which leads to the latter resending the same requests again, because it works on behalf of the source host.

This results in congestion on the links between both routers, which leads to ICMP reply being impeded and hence, the source router sends an unreached destination message to the source host [28]. In addition, the destination router before sending the ICMP request must send an ARP discovery request to each requested host, which costs approximately 10.5 ms in this experiment topology without load, which increases with load growth as well as increasing number of hops between the router and the destination host. This, in turn, puts the router into waiting status until its get the reply from the destination host.

In contrast, with the proposed model, there is no packet loss with increasing load, where one of up to 60,000 packets at a rate of 3,333 RPS was tested, but there is an increase in response time, reaching 1,008 ms at that load. MSSERD uses the data plane more than the control plane for expected types of packet such as an ARP request, while only one ARP_Reply enters the control plane in the destination subnet, for just approximately 3.56 ms (without the effect of the number of hops between Exit_switch and ARP server/destination host), to get host destination information to install the X_packet_L2 rules between the Exit_switch and the destination host, In addition, another ARP_Reply enters the control plane in the source subnet to let the controller get the source host information to install X_packet _L2 rules and ARP reply rules in all the switches to the source host. Another reason for such a trend in MSSERD is that the load is divided between the ARP server, which has the responsibility to answer just ARP requests and the hosts which have the responsibility to answer ICMP requests. As a

consequence, the packets switching in the network is faster than with legacy architecture and has greater reliability in terms of driving packets to their destinations.

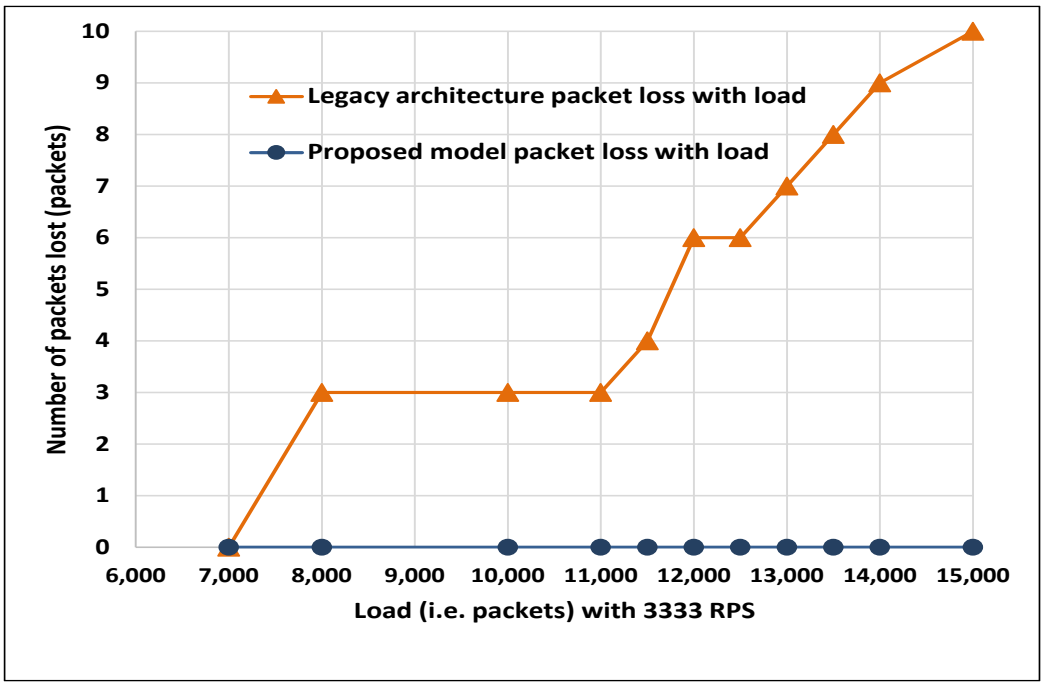


Figure 5.9: Number of packets lost during traffic load: comparison between MSSERD and legacy router-based architecture

5.5.3 Number of packets in the data and control planes: comparison between MSSERD and legacy model

In this experiment, linear topology is used when increasing the number of subnets from 2 to 10. The source and destination subnets each have the topology (host-switch-switch-router) in the legacy architecture, but the routers are removed in the proposed architecture and an ARP server is used. The middle subnets contain just one router/Exit_switch and a controller per subnet. Only one ping request is sent each time from the source to the destination subnets and by using the Wireshark tool all packet trajectories in both the control and data planes are recorded across all elements in both architectures.

Regarding the control plane statistics, as can be seen in Figure 5.10, in the legacy router-based scheme when increasing the number of subnets from 2 to10, the number of control packets is significantly increased from 32 to 96 for several reasons. Firstly, any host needing to send a message outside its subnet, must send an ARP request to its default gateway (router) that creates 1 Packet_in and 1 Packet_out in the control plane in the router section. Secondly, because of the broadcast discovery mechanism in the legacy

architecture, each switch in the source and destination subnets, when it first deals with the ARP request from a host, makes 4 packets in the controller plane, including 2 for the ARP request and 2 for the ARP Reply. Thirdly, the router in each subnet, after discovering the IPs for its neighbours' router, broadcasts ARP requests to all routers asking for their MAC addresses that will be needed for modifying the Ethernet frame in each packet going out from its subnet. Moreover, routers have to continue broadcasting at specified times to keep their MAC tables up-to-date.

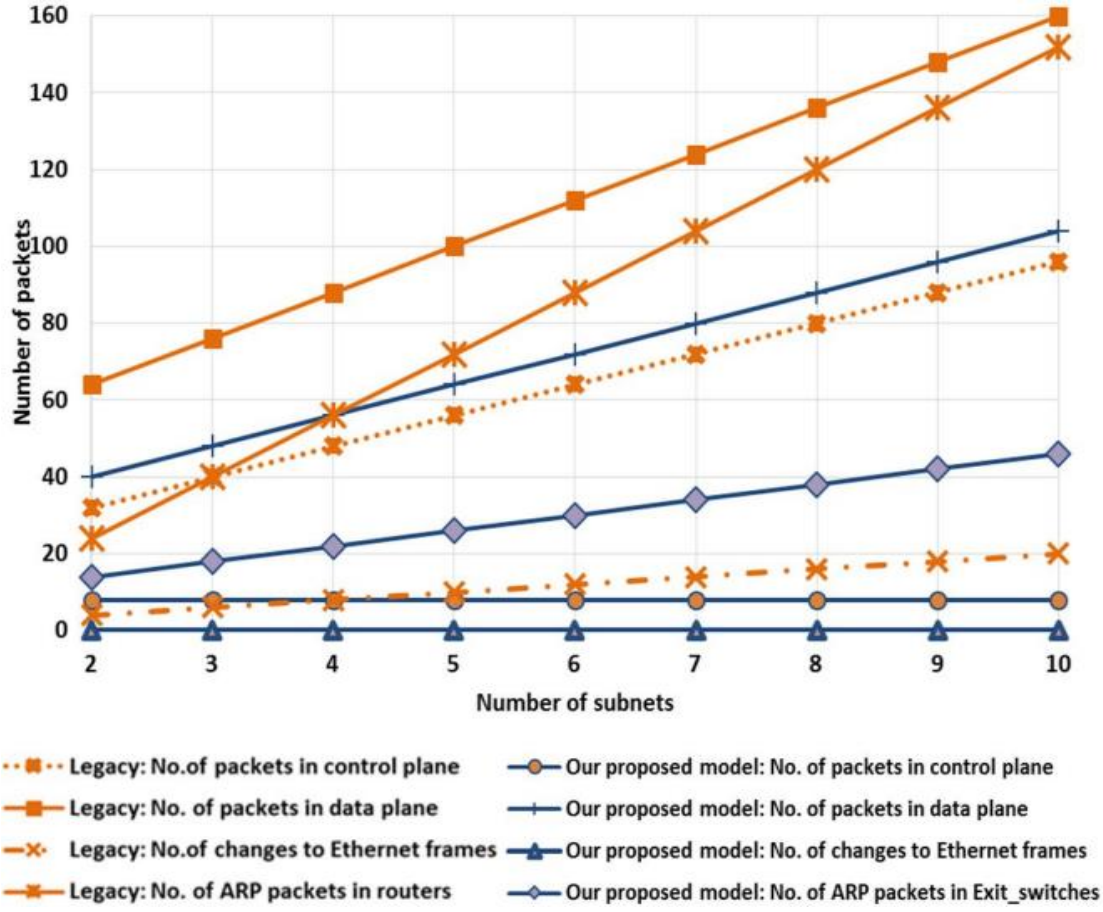


Figure 5.10: Number of packets in the control and data planes, number of changing Ethernet frames as well as the number of ARP packets when scaling network: comparison between MSSERD and legacy router-based architecture

As a consequence, each router in this experiment's topology deals with 1 ARP request and 1 ARP reply. Fourth, each router, after answering ARP requests from other routers, starts to generate its ARP requests to those asked to confirm its validation. This leads to at least 2 ARP packets being generated in each router and this number increases with increasing the number of routers, i.e. in mesh topology the number of broadcasts

increases when the number of output ports from router increases. Finally, the router in the destination subnet deals with an extra 2 ARP packets and 2 ICMP packets in the control plane, whereby after receiving ICMP from the source subnet, the router broadcasts an ARP request to discover the destination host's MAC address and on receiving the ARP reply, has to send the ICMP to that destination host.

On other hand, using the proposed model the packets in the control plane is fixed at 8 packets, regardless the number of subnets between the source and destination subnets, because it has a general subnets view feature and can consequently, install proactive rules between the source and destination subnets. Those 8 packets are generated only in the source and the destination subnets, i.e. there are no controller packets in middle subnets. In more detail, the proposed model involves just sending 1 ARP reply for the Exit_switch in the destination subnet and 1 ARP reply in the source subnet to release the hosts information in order to install ARP and the X_packet_L2 rules, which totals 6 packets.

Regarding the data plane statistics, the numbers of ARP and ICMP packets that are generated in the legacy architecture increases from 64 to 160 packets, when dealing with 2 and 10 subnets, respectively, because of the mechanism for discovering the default gateways and destination hosts. In contrast, for MSSERD, this figure rise from 40 to 104 packets for 2 and 10 subnets, respectively owing to the removal of the routers and the use of proactive rules. As a consequence, the proposed model consumes 35% less bandwidth than the legacy model in the data plane when dealing with 10 subnets. In addition, it can be seen from Figure 5.10, that the router generates approximately 3 times as many ARP packets than MSSERD when the number of subnet is 10, which, in turn impacts on the data plane in the legacy model.

Finally in this experiment, how many times both architectures are changing the Ethernet frame is calculated and it is clear that the legacy model with router devices changes from 4 to 20 times to deal with 2 to 10 subnets (i.e. increasing by a rate of 2 for each new subnet), because each router must modify the Ethernet frame for each ICMP packet by removing the source MAC address and put in its MAC address instead. As MSSERD does not modify the Ethernet frame of the ICMP packets that pass from one subnet to other, the number of times changing the Ethernet frame is equal to zero.

It should be noted that regarding all the statistics in this experiment, the worst scenarios, especially for legacy architecture when evaluating number of packets are, firstly, when a hybrid topology among routers (not linear) is used, as this increases the number of output ports from each router, which increases number of packets owing to the

broadcast mechanism. Secondly, when increasing the number of hosts in each subnet, for each specified time each host sends an ARP request asking for the router's MAC address and if the number of hosts increases then this also increases bandwidth consumption.

5.5.4 Performance with scalability: comparison between the proposed and legacy router-based architectures

In this testbed experiment, there is a linear topology with an increase in the number of subnets from 2 to 10 and generating one ARP request using the ARPing tool [105] and then generating one ping request using the ping command from the host, which is connected to an edge subnet requesting the MAC address for a destination host that connects to the subnet on other edge of the network. The Wireshark tool is used to trace the ARP and ICMP packets. Firstly, the legacy network is implemented by using the topology (host-switch-router) in the source and destination subnets as well as the routers in the middle subnets. Then, all the routers that work as default gateways are removed and SDN switches as switching devices are used to implement the proposed model.

From Figure 5.11, it can be seen that the legacy architecture failed to send an ARP request with broadcast value 'ff:ff:ff:ff:ff:ff' in the Ethernet's destination MAC address field outside the source subnet, which is why there is no line for this in the figure below. The reason for this is the legacy network architecture completely depends on the broadcast mechanism to discover the local components in each subnet. So, each subnet generates a huge amount of broadcast packets that will not be allowed to pass to the next subnets and they are dropped by the routers, because passing them will definitely have a bad effect on the other subnets, which is compatible with theoretical finding from Legacy Rule1 in subsection (5.2.1).

While in the proposed model the destination MAC's address is obtained with an average ARP response time = 8.71 ms. In addition, the trend with an increase in the number of subnets from 2 to 10, is that the ARP response time only slightly increases from 8.11 ms to 8.91 ms. The reason for this is that MSSERD has the proactive mode that installs general ARP rules in bootstrap time, which can be used by any ARP request to get the destination's MAC address from a different subnet.

On the other hand, by generating one ping request in legacy network and scaling the network, the general trend of the ping response time is to increase from 22.9 ms to 25.5 ms for 2 and 10 subnets, respectively, with an average of 23.65 ms. In more detail, with

legacy architecture, if optimum calculation is assumed, a source host when trying to connect to a host in another subnet, firstly, generates ARP request to discover the default gateway's MAC address (internal router's MAC address), which approximately practically costs 8 ms (for topology router-switch-host) and about 6.5 ms will be spent when the router in the destination subnet discovers the MAC address for destination host using the same topology. So, regardless of the number of subnets in the middle between the source and destination subnets and regardless of the time cost in/between the routers, at least one ping requires 14.5 ms with the legacy router architecture. The proposed model is 39.45% (i.e. (8.78-14.5)/14.5*100%) and 22.06% faster than the legacy model when dealing with 2 and 10 subnets (all the experimental subnets), respectively. In addition, according to the experimental calculation in Figure 5.11, the proposed model, on average, is 2.36 times faster than the legacy one. This is because MSSERD avoids using the broadcast legacy mechanism (used by legacy architecture to get next routers' MAC addresses) between routers, which costs in legacy scheme 2.4 ms as well as removing the router devices will eliminate the time taken to pass packets and multiple changing of the Ethernet frames (which costs in legacy scheme 0.08 ms for each incoming/outcoming packet per router).

In contrast, MSSERD installs proactive rules that send ARP packets with a quick response time. In addition, the controllers in the proposed framework are designed in a way as to experience little latency to response to an ARP reply packet. Moreover, the SDN Exit_switches in all the middle subnets use proactively installed IP_to_physical port rules to switch packets to their destination. Furthermore, one of the reason for the proposed system being faster than the legacy one is because latency in the latter is greater than the SDN switches in the proposed model architecture. Regarding which, when both switches deal with an ARP request the latency will be 0.1 ms in the proposed switches as compared with 2 ms in the legacy switches because the latter send ARP request packets to the controller to register the source MAC's address, which costs time.

There are some slight fluctuations in the legacy results, which is because of collisions and competition to use the same link exactly at the same time. One reason for these occurring is owing to each router in each subnet, during the bootstrap and after each specified period of time, generating broadcast ARP packets to all its neighbouring routers in order to keep its ARP table up-to-date with their MAC addresses. Regarding the scalability evaluations from this experiment, by using the Excel trend tool, it emerges that MSSERD with 25.5 ms can pass about 52 subnets, whilst for legacy this figure is 10,

which means MSSERD scales up the network by 524% in terms of subnets, when compared to the legacy router mechanism.

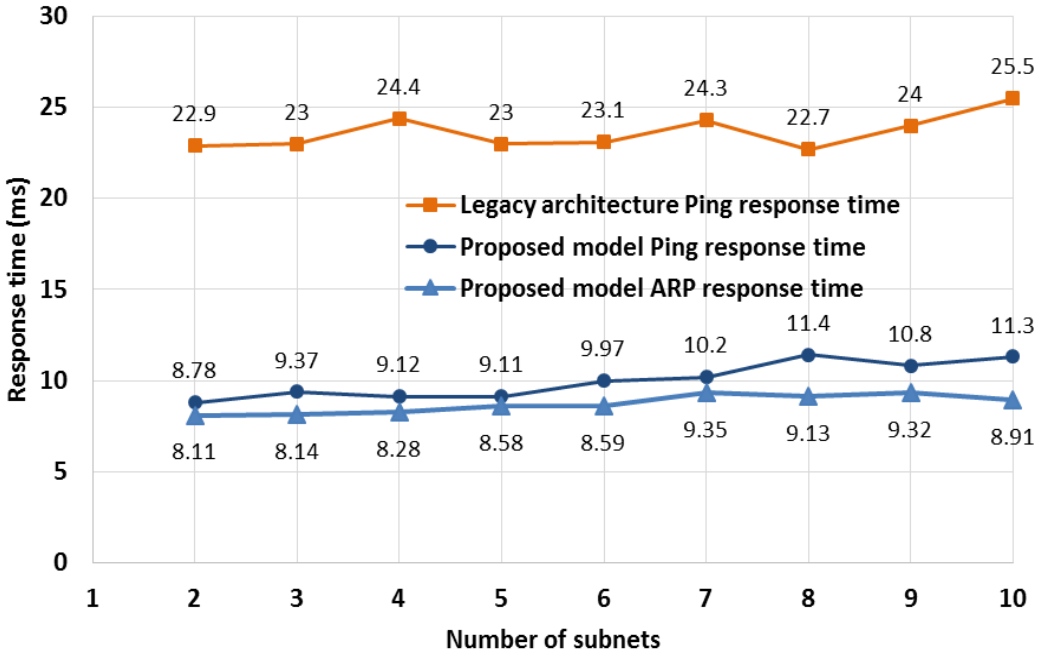


Figure 5.11: Ping and ARP response times whilst scaling the network: comparison between MSSERD and legacy router-based architecture

5.5.5 Bootstrap discovery and rediscovery time

In this experiment, the bootstrap discovery time and rediscovery time in the SDN multi-subnets architecture is evaluated using the proposed model. The linear topology is used with an increasing number of subnets from 2 to 10 and the discovery time interval for MDP is set at 2 secs, with all the subnets working concurrently. The discovery time in this experiment can be defined as the time spent by all the subnets to discover all other subnets' information that are in same network, fill the subnet_discovery table for all subnets and to install proactive rules in the SDN switches. The bootstrap time is the time taken by any system from the moment of starting to work until it reaches steady state. In this experiment, this is arrived at when all the subnets have a general view of all the other subnets i.e all have the same subnet_discovery table.

As can be seen in Figure 5.12, there are three bootstrap discovery time zones depending on their values, whereby zone1 contains (2,3,4) subnets with an average discovery time of 7.92 ms, zone2 contains (5,6,7,8) subnets with an average of 10 ms and zone3 has (9,10) subnets, with an average of 14.32 ms. Accordingly, when the number of subnets is increasing the bootstrap discovery time is also increasing, because every subnet

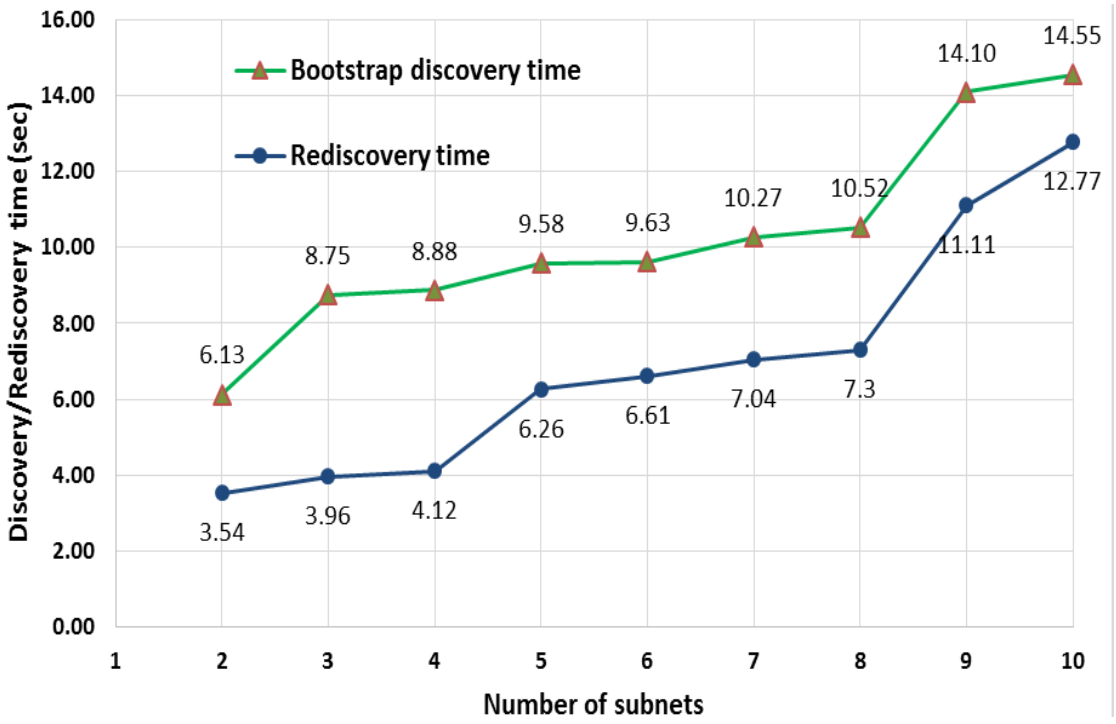


Figure 5.12: Bootstrap discovery and rediscovery times whilst scaling the network

spends time on the boot process until they start sending or receiving MDP discovery packets. As a consequence, when the number of subnets increases the total boot time spent by all also definitely increases.

In addition, the rediscovery time is evaluated using the proposed model, which can be defined as the time need by all subnets to detect adding/deleting one or more subnets to/from the network and this is evaluated after the finish of bootstrap time. In the rediscovery experiment, a new subnet is added or an existing one is deleted from the network and the time taken to detect those events is evaluated for all subnets. A subnet is inserted/deleted from one edge of the linear topology and record the rediscovery time to update the subnet discovery table on the other edge, as this is the worst time. As can be seen from Figure 5.12, the general rediscovery time values are less than the bootstrap discovery time, because all subnets during the rediscovery process are already running and each subnet already has reached its steady state.

There are three main zones in the rediscovery process that match the zones from the bootstrap discovery process, such that zone1 has (2,3,4) subnets with an average of 3.87

ms, zone2 contains (5,6,7,8) subnets with an average of 6.8 ms and zone3 has (9,10) subnets, with an average of 11.94 ms. It should be noted that the results in the same zone regarding the two aspects of time, shown in Figure 5.12, change slightly because all the subnets work concurrently. The discovery and rediscovery times in zone3 are different by approximately 4 secs than those in zone2, because when there is an increase in the number of subnets in linear topology the discovery packets generated by last subnet need to make all the subnets to reach the first subnet. This leads to an increase in discovery/rediscovery times with scaling of subnets. In addition, the size of the MDP packet increases only during the multicast phase when there is an increase in the number of subnets, which leads to an increase of latency time for transferring/dealing with it In this experiment the MDP is configured to continue multicast after bootstrap so as to get the worst case scenario owing to collisions.

5.6 Summary

In this chapter, legacy router-based network architecture and default gateway mechanism used to connect LAN-based networks have been analysed. Subsequently, meeting all the requirements and overcoming the limitations of the legacy router-based mechanism, the MSSERD design and implementation were introduced for handling ARP and general IP packets, supported by dynamic discovery of the whole network through the proposed MDP protocol. Through building an actual testbed and implementing several experiments, comprehensive results have shown that MSSERD enhances scalability in terms of the number of subnets up to 52 when compared to the legacy architecture. In addition, it improves the efficiency significantly, especially with high load, by reducing the overhead in the control and data planes, improving performance 2.3 times over legacy mechanisms, enhancing reliability without packet loss and by reducing complexity.

Chapter 6

Open-Level Control plane architecture

6.1 Introduction

Scalability of networks is a real issue in current network architecture [71] owing to the rapid increase in the traffic of hosts, for such as video on demand as well as the growing number of end devices, in particular, in relation to development of the Internet of Things (IoT) technology [106]. In order to design an architecture/mechanism that can scale the network into a large one, whilst concurrently enhancing network performance, the following requirements should be taken into account:

- The new architecture/mechanism needs to support SDN's powerful feature, i.e. proactivity, which leads to enhancement of the response time and load balance among the network resources. That is, the general view of network is the fundamental requirement to apply proactive behaviour in an SDN for traffic manipulation [107]. As the network general view relies on the discovery process, this leads to consideration of the process as an essential one that is sensitive to the time factor. Accordingly, the discovery packets should avoid the congestion plane (i.e. the data plane) as much as possible;
- No new hardware (e.g. middleboxes) should be added to the network and no new software should be added to the host or switch sides as this could lead to downward compatibility problems;
- Standard protocols should be used to support interoperability and openness [29], regarding which, [20] fails to support this point;

- The number of protocols used for the discovery process should be as few as possible so as to avoid inconsistency, complexity and latency as a consequence of their concurrent operation;
- There needs to be support for transparency, which means users can see the system as a single one [29];
- The complexity between the intra and inter-domains should be decreased as much as possible by using the same/consistent discovery protocols. Some other designs fail to apply this, such as in [20].

No previous study (see chapter 3 regarding scalability limitation owing to the use of the aggregation distributed mechanism in distributed architectures) has efficiently solved the scalability issue nor has completely taken into account the fundamental requirements set out above, which is the motivation behind the presenting of this chapter. We propose an *Open-Levels Control plane architecture (OLC)* to provide better scalability in an SDN network. OLC, firstly, analyses a well–known distributed mechanism, namely, the distributed aggregation mechanism, which is essential for performing the discovery process in traditional and SDN architectures. Then, novel architecture for the control plane is put forward, which defines open levels (i.e. multi-levels) of this plane with a distributed-centralized concept as well as defining the SDN switches between the control levels. In addition, an innovative dynamic discovery mechanism is introduced, which can discover multiple subnets and networks. In sum, OLC introduces full architecture and mechanisms for discovering intra and inter-domains.

6.2 Description of the distributed control plane architecture and analytical model formulation

In this section, we describe the distributed aggregation mechanism by analysing discovery packets in fully distributed architecture under both the current and SDN architectures. In addition, the mathematical formulation for this mechanism is calculated at the end of this section.

6.2.1 Distributed control plane architecture

In an SDN network that covers a large area, distributed subnets interconnect each other, with each subnet having its own switches and controller. The internal switch forwards packets within the same subnet, while the edge switches work as middlebox devices (e.g. routers) to forward packets outside/inside their subnets. The controller controls every packet in its subnet depending on its policy as well as exchanging its subnet's information with other subnets in the same distributed-based network, which is why this is called a distributed control plane. This type of network normally uses the data plane bus to transfer discovery packets through the edge devices, such as in [20].

6.2.2 Connectivity of distributed control plane

In order to make connections among subnets in same SDN network, the edge devices, such as routers/Exit_switches must exchange their information with their neighbors. In this architecture, the controller has the main script and the routing table (in the case of using a router as an edge device, then it is called virtual router [18]). The rules inside each edge device can be installed in two ways. Firstly, manually by the administrator, where he/she has to know each neighbor's information (IP address and subnet mask) in order to install a static route to it. Secondly, this can be done dynamically by using a routing protocol (e.g. OSPF, RIP or MDP), where each controller has the routing protocol's script and exchanges advertisement packets at specified times with its neighbors.

After the specified discovery time each controller has an understanding of the whole network topology and installs rules in the edge devices (e.g. virtual routers) to pass packets to outside the subnet. If the virtual router is used as an edge device then it needs to refresh its connection with its neighbors by exchanging ARP packets after the specified time in order to keep the ARP table in each router updated [97], because it depends on the default gateway mechanism. Whilst if the Exit_switch is used as an edge device then this omits the use of ARP packets and there is no need to for refreshment as the Exit_switch mechanism relies on the proactive behaviour of the SDN controller through the availability of the general view of the network.

6.2.3 Aggregation discovery mechanism to exchange network discovery information

The aggregation mechanism is used in fully distributed subnets to discover the whole network's IPs and to gather statistics [20] in order for each subnet to have a consistent general network view. As a consequence, this gives SDN the powerful ability to install

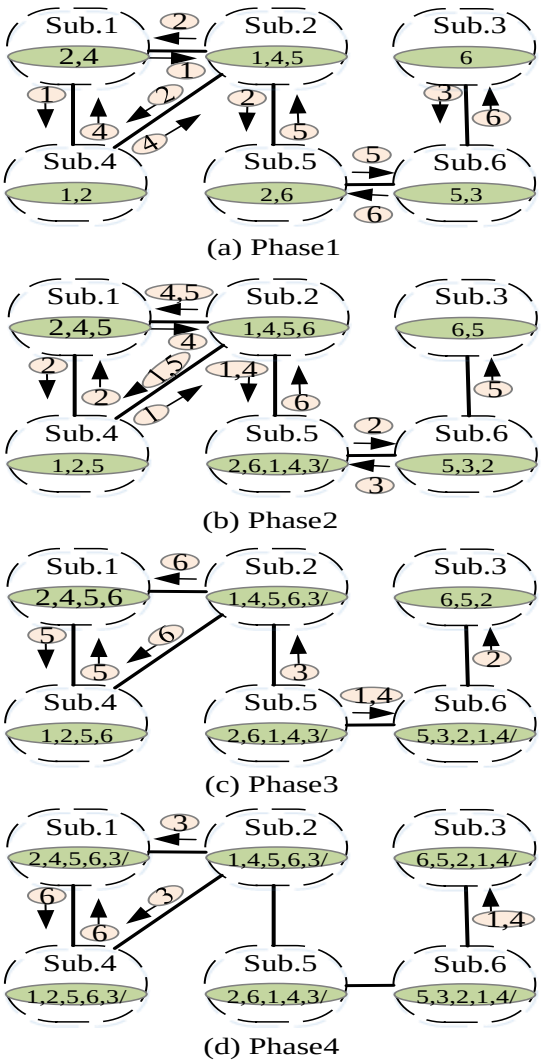


Figure 6.1: Shows the discovery phases when applying the aggregation mechanism in fully distributed architecture

(Note 1: in each phase the process on link happens before the result inside the subnets)
(Note 2: for the figure to be not fully packed we eliminate repeated discovery messages, however in practice there is a message on each port from each subnet in each phase)

rules proactively and reactively in SDN switches for better performance. Regarding the aggregation mechanism, I theoretically evaluate the number of phases that are needed by the network in order to let each controller in each subnet have the general view of its entirety. As can be seen in the example topology in Figure 6.1 (a), each controller in each subnet starts the first round by sending multicast discovery protocol's packets to all its neighbours, which is called phase1 of the discovery process. It should be noted the number in the pink shape reflects the subnet number with all its discovery information involving its routing and topology gathered from its neighbor subnets [31].

As consequence of the results from phase 1, each controller just has knowledge about its next neighbors and puts this information in the neighbors' tables as well as putting the network topology in the topology table. As a result of phase 1, Subnet1 just has information about subnets 2 and 4, while Subnet2 just has information about subnets 1, 4 and 5. In phase 2, the subnets will start the second round of multicasting, as can be seen in Figure 6.1 (b). In this round, each controller will be used as a bridge to exchange the information among its undirected connected neighbors. In this case, the information will go one subnet further than in phase 1. As a result of phase 2, the topology and neighbors' tables will be updated, for example Subnet1 has new information regarding subnet5 while Subnet5 has new information regarding subnets 1, 3 and 4 (satisfied). Continuing to phase 3, the tables will be updated and the discovery packets will continue multicasting to next neighbours, which leads to the discovery information going two subnets further than in phase 1 as can be seen in Figure 6.1(c). Finally, after finishing the fourth phase, all the controllers will have the appropriate information regarding all the subnets' topology tables, as can be seen in Figure 6.1(d).

As a result of using the distributed aggregation mechanism in traditional/SDN architectures the number of phases is equal to the best path between the furthest edges of network (i.e. furthest subnets), as in Equation 6.1.

$$N_{op} = B_{pfes} \qquad (6.1)$$

Where, N_{op} denotes the number of phases and B_{pfes} is the best path between the furthest subnets.

Regarding the discovery process latency, the highest controller latency refers to the time needed by the controller to multicast discovery packets, receive discovery packets and to store/retrieve information to/from the discovery tables. As the subnets work concurrently, highest controller latency is equal approximately to the latency of the

slowest controller. Whilst the latency in each phase is equal to the highest controller latency plus the highest link latency , as in Equation 6.2.

$$L_p = H_{cl} + H_{ll} \qquad (6.2)$$

Where, Lp denotes the latency in each phase, H_{cl} is the latency of the slowest controller and H_{ll} is highest link latency, which represents the slowest link in the network between subnets.

Accordingly, the discovery time needed each specified time (T) is approximately equal to the number of phases multiplied by the latency of each phase, as in Equation 6.3.

$$D_t = N_{op} * L_p \qquad (6.3)$$ Where Dt = discovery time

The number of packets generated in the network to complete the discovery process for one phase is equal to the summation of the number of out links from each subnet, as in Equation 6.4.

$$N_{pDp1} = \sum_{n=1}^{Ns} (\ N_{ol}\ )_n \quad (6.4)$$

Where, N_{pDp1} is the number of packets generated in the network to complete the discovery process for one phase, N_{ol} is the number of links from each subnet and N_s represents the number of subnets.

While the number of packets to complete the full discovery process is equal to the number of phases multiplied by the number of packets required to complete one phase, as in Equation 6.5.

$$N_{pDpF} = N_{op} * N_{pDp1} \qquad (6.5)$$

Where N_{pDpF} represents the number of packets to complete the full discovery process.

As can be seen from the equations, for a large network this requires many phases in relation to B_{pfes} (Equation 6.1) and also an extensive number of packets in each phase, which leads to consumption of data plane bandwidth, an increase in the requirements of the control plane [20] and longer discovery/rediscovery time.

6.3 OLC design

The OLC model is designed in this section, where a general architecture in order to enhance the discovery subnets/networks mechanisms in large Ethernet SDN networks is proposed. In addition, the dynamic discovery hierarchal protocol (DHP) for a multi-layer control plane is proposed to provide a general view of whole network, which supports SDN performing proactive behaviour. We focus on the control plane in intra and inter domains, where the data planes are already connected.

6.3.1 OLC units

OLC model contains several units for completing the purposes that it has been designed for, as can be seen in Figure 6.2. These units work with a multithread concept aimed at fast response and distributed loads on the cores of the CPUs. The *Received Unit* receives discovery packets from the same level, level minus 1 (level-1) and level plus 1 (level+1) controllers, subsequently sending the messages to the *Analysis and Calculation Unit* that has connections with all the discovery tables. This unit will obtain, analyze and perform calculations on the received information to fill the discovery tables, including the Neighbors_topology and All_topology. Then, it sends the information to the *Send Unit*, which has two subunits, DHP1 and DHP2, which were assigned their names from the dynamic discovery hierarchal protocol (DHP) proposed in this chapter. This unit takes its information from the discovery tables and sends discovery messages into the same/different level controllers, as is explained later in this section.

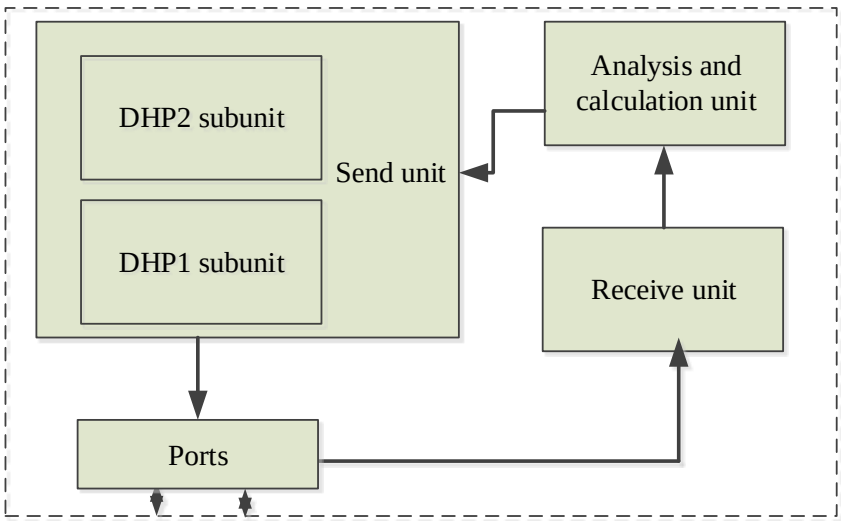


Figure 6.2: OLC units in a single controller

6.3.2 General network architecture under OLC

When a network is scaled up, important requirements are a fast discovery time during bootstrap time and a fast rediscovery time for any change in the network states, such as add/remove the link between two subnets/networks. That is, the latency of the discovery time is an important factor when scaling the network, whereby if this time is low, new subnets/networks can be added and hence, the network scaled up. In order to achieve the best performance with fast discovery/rediscovery times, we believe that the centralized architecture should be combined with the distributed one.

The proposed model involves dividing the scale concept for an SDN network into

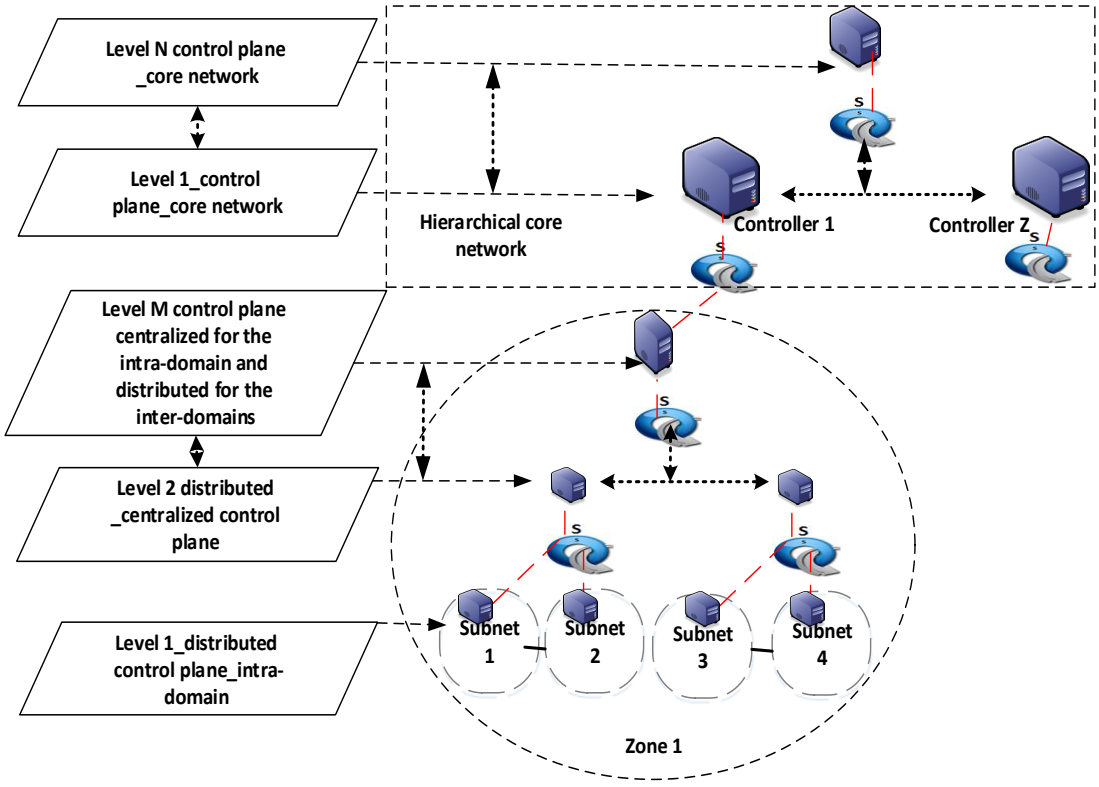


Figure 6.3: Overall OLC architecture

vertical and horizontal scales, where the former represents the scale of the control plane, whilst the latter pertains to that of the data plane. The ability to scale the control plane leads to scaling of the data plane, because it enhances the discovery time. As a consequence, we believe we have developed the best discovery architecture, for it combines both distributed and centralized architectures, which introduces an open-level distributed-centralized control plane architecture in an SDN network, as can be seen in Figure 6.3.

The vertical process in the figure pertains to the scaling up of the control plane. Regarding which, level 1 is the first level in the chain and contains distributed SDN controllers, each responsible for at least one subnet, whilst the second level contains the centralized SDN controllers. The proposed model uses SDN switches between each vertical neighbour level to give more flexibility and for recovery purposes. In order to scale the network up to a large one, such as a Metropolitan Area Network (MAN)/ Wide Area Network (WAN), the controllers in the last level of each network (i.e. level M) will represent distributed controllers for the level 1 controllers of the core network. The controllers keep connecting in a hierarchical way until those in level n are reached, which represent the top of the pyramid for all zones. The core of a network's control plane could start from level 3 or above depending on the size of network and the decision of the administrator.

On other hand, horizontally, each zone could represent a campus/enterprise/small city that connects to its neighboring zones using the data plane. By using this architecture, we can continue to link zones until cover a very large area, such as a country/group of cities. From the global perspective, we can imagine dividing the world into areas, with each containing one/more zones have one/more head controller(s) at the edge that can be connected in a distributed manner to exchange information. In addition, we could build data plane's core in the same way as that of the control plane, whereby there are SDN switches connected each other to exchange data among different networks.

6.3.3 OLC Discovery Mechanism

As the OLC model can be scaled up to support a very large area, such as a country or even the world, there are two discovery views, with the first being with regards to the same network (intra-domains), while the other relates to a large network (inter- domains).

6.3.3.1 Within the same network (intra-domain)

The type of discovery we propose in this chapter involves a hierarchal mechanism with M open level controllers in the intra-domains (Figure 6.4 shows two levels of controllers as an example). In order to perform it, the OLC model involves deploying a dynamic discovery hierarchical protocol (DHP), which is developed from the LLDP

protocol. As aforementioned, this contains two elements, specifically, a distributed one (DHP1) in the controller's DHP1 subunit and a centralized one (DHP2) in its DHP2 subunit. The hierarchical discovery mechanism starts from the controllers in the subnets. Firstly, each controller in each subnet in bootstrap time will create a Neighbors_topology table. which has the fields: Neighbors_ID, Timestamps_of_packets, which are use to calculate links' latencies with neighbors and hence, identify the best paths, Edge_switch_ID, which is used to identify a subnet's edge switch and the Edge_switch_port, identifying which port is going to which subnet.

As can be seen in Figure 6.4 (a), in each subnet the controller in level 1 during phase 1 multicasts its ID and timestamp of packet to the neighbors using one DHP1 message, while there are no messages being sent to the level 2 controller. Each controller will receive DHP1 messages from its neighbors, which it adds to the Neighbors_topology table. The controller will perform multicasting after a specified time or if there is a change

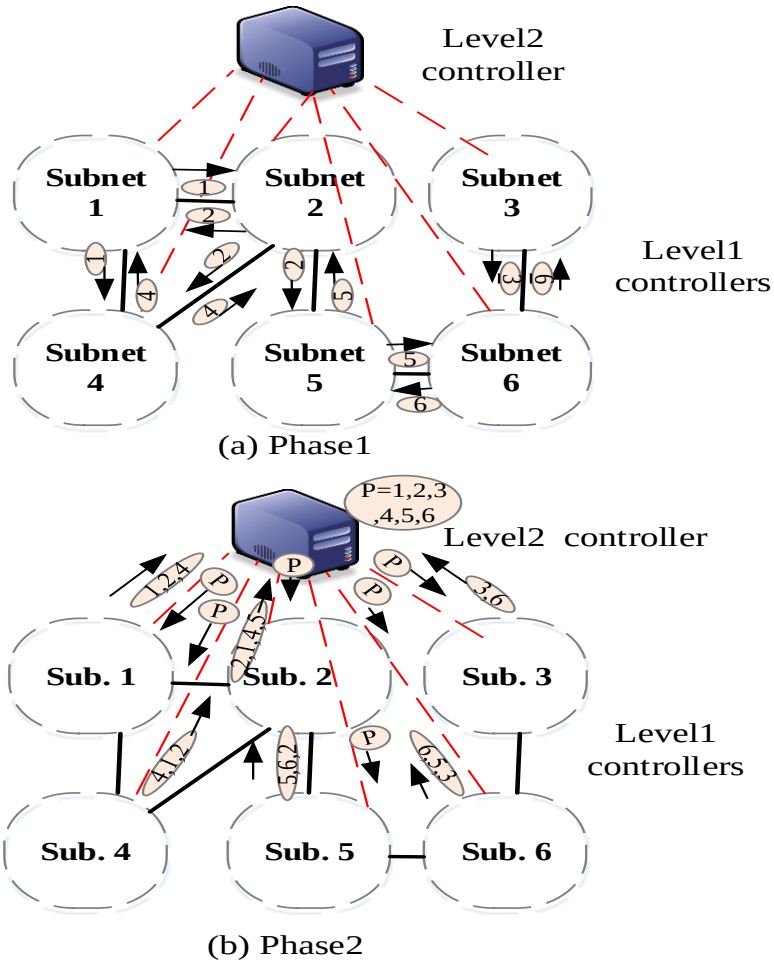


Figure 6.4: Example of the OLC discovery mechanism inside one network (i.e. intra-domain) containing six subnets with two levels of controllers

(P = discovery message containing route information calculated by a centralized controller for all subnets in dictionary format, i.e. Sub.x can go to Sub.y through Sub.z).

in network conditions. Secondly, the level 1 controllers will send DHP2 messages from the DHP2 subunit, which has some of the information that is in the Neighbors_topology table (i.e. Neighbors ID, link latency and Internal_subnet latency) in dictionary style, to the centralized controller in level 2, as seen in Figure 6.4 (b).

The centralized controller in bootstrap time creates an All_topology table, which has following fields: Source_ID, Destination_ID, Link_latency and Internal_latency. The centralized controller will combine all received DHP2 messages and using the Dijkstra algorithm will find the best paths between each pair of subnets and then, will fill the All_topology table. This controller will send back DHP2 messages which contain just the crucial information to each related subnet required to install rules for reaching the destination subnets.

It should be noted the DHP2's messages are different from each other, i.e. each is unique for each subnet in order to avoid sending information to unrelated one. The messages will be in dictionary format, i.e. net_X {net_Y: net_Z}, which means if subnet/network X wants to connect to subnet/network Y, it should go through subnet/network Z. The controller in level 1 will save this information in the All_view_discovery table that has been created in all controllers at all levels in bootstrap time. As a consequence, the controllers in level 1 will have a general view of the whole network. Then, the level 1 controller installs rules proactively in its switches to each destination subnet in its network relying on Edge_switch_ID and Edge_switch_port fields in the Neighbors_topology table.

Regarding the number of phases (Nop) in the OLC discovery mechanism, if we assume there are two levels of controllers in the intra-domain architecture, in order to compare our architecture with the aggregation distributed mechanism in section 6.2, each controller deals with one phase in the data plane and one in the control plane. Accordingly, there are two phases no matter how many subnets are in the network, as in Equation 6.6.

$$N_{op} = 2 \quad (6.6)$$ where N_{op}= number of phases

Regarding the discovery time, this is needed after each specified time (T) and approximately equals the latency of the one phase from Equation 6.2, plus the Highest level 1 controller latency when sending/receiving DHP2 messages ~~(multiplied by 2)~~, plus the maximum latency of the centralized Links, which connect level 1 to level 2 controllers (there and back), plus the latency of the centralize controller (L_{cc}), as in Equation 6.7.

$$D_t = L_p + 2\ L_{cL} + L_{cc} + H_{cl} \qquad (6.7)$$

Where, D_t denotes the discovery time, L_p is the latency of one phase, L_{cL} is the latency of the centralised links, L_{cc} is the latency of the centralised controller and H_{cl} is the highest level 1 controller latency.

While the number of packets generated in the network to complete the full discovery process is equal to the sum of the number of links from each subnet and the number of links from level 1 to level 2 (i.e. number of subnets, if each subnet connects with one controller in level 2), as in Equation 6.8.

$$N_{pDpF} = \sum_{n=1}^{Ns} (N_{ol})_n + N_s \quad (6.8)$$

Where, N_{pDpF} represents the number of packets generated in the network to complete the full discovery process, N_{ol} is the number of links from each subnet and N_s is Number of subnets

6.3.3.2 In the multiple networks (inter-domain)

The OLC model provides the same mechanism as inside the network (i.e. intra-domain) to connect multiple networks in order to cover a large area, where each controller in the last level of each network will represent its network by using Network Address Translation (NAT) [108]. In addition, it will be seen in a distributed manner in relation to other controllers in the last level from other networks, as can be seen in Figure 6.5.

Each intra-domain network will have an SDN-switch(es), which connect(s) directly to the controller in the last level of that network. That switch belongs to the data plane and is used to send information using DHP1 messages to the neighbor networks that are in different domains after applying the NAT mechanism. Whereas the DHP1 discovery messages will contain the Public_network_ID field, which represent the public IPs for that domain and Timestamp field to evaluate the link latency between two neighbor inter-domains. After receiving DHP1 messages, the relevant controller will send DHP2 messages to a one level up controller (e.g. level 3) that will perform path calculation among the inter-domain networks and send back this information to the related network in a dictionary style. For example, Network_X{Network_Y: Network_Z}, which means that if network X wants to connect to network Y, it should connect first to Network Z.

That information will be saved in the All_view_discovery table. The same OLC mechanism is applied when there are (n) levels of controllers covering a very large area.

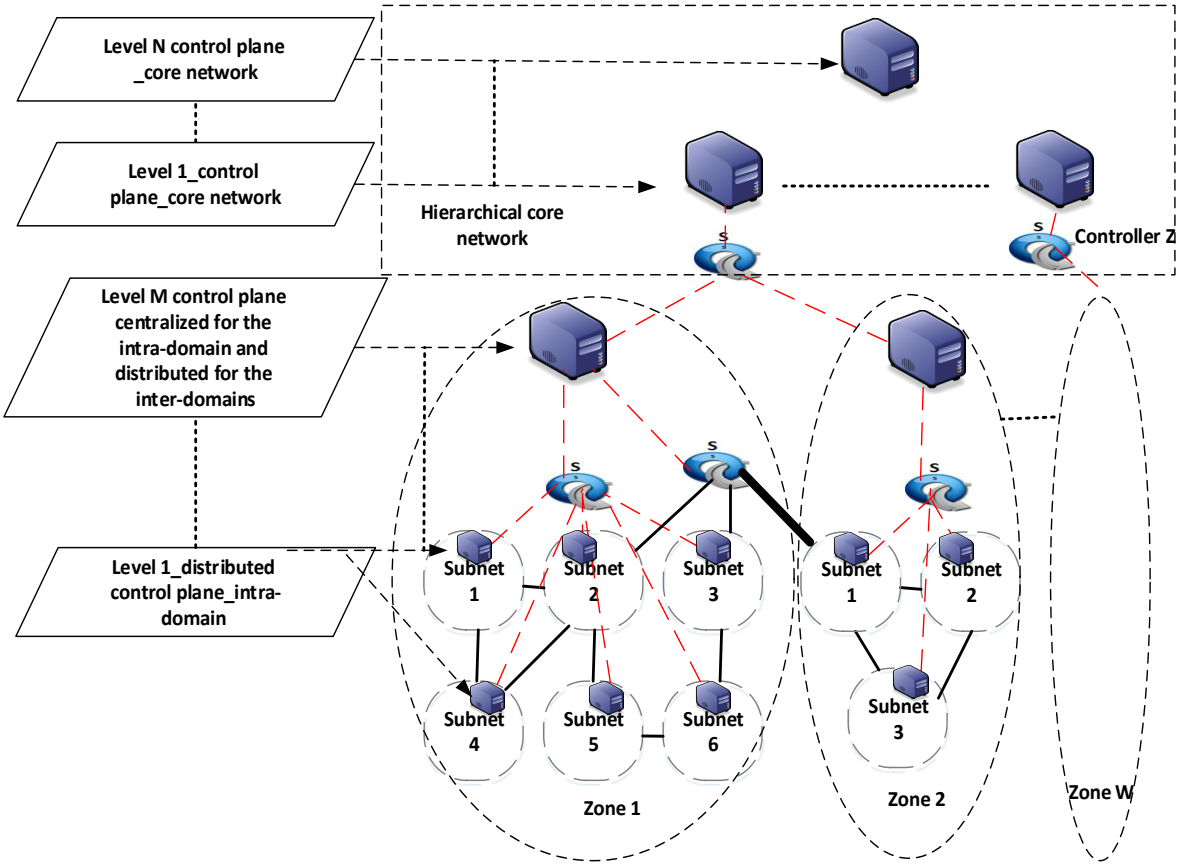


Figure 6.5: Open-levels OLC intra and inter domains

6.3.4 Location of the controllers

The OLC offers a flexible architecture for fulfilling different purposes. For example, if it is used on a campus/in an enterprise with long distances between departments then the level 1 controllers will be located near to the subnets, especially if there are many users, in order to reduce discovery/rediscovery time. While if it is used in a data center network, the level 1 controller can be located in the controllers' pool near to the last level of controllers in that network (e.g. level 2's controllers). In addition, the other controllers that represent the core network could belong to the same or different providers and could be located near to its serving zones.

6.3.5 Reacting when the network fails

There are different types of failure can happen in any network that could lead to the whole network grinding to a halt. OLC take different actions to overcome these failures and their consequences as follows.

6.3.5.1 Handling level 1 controller failure

OLC uses the standard master-slave mechanism offered by the Openflow protocol [89]. With this mechanism, the level x+1 controller works as a slave controller for the level x controllers (i.e. masters), where if any master controller related to a subnet fails, then the slave controller will take the responsibility of controlling that subnet.

6.3.5.2 Handling levels 2 to n controller failure

If a centralized controller in levels 2 to n fails, OLC provides a recovery feature by using the SDN switches in the control plane such that two or more controllers in the same level are connected to the same switch, so if the master fails the slave can serve the network. In addition, by using the same mechanism the load balance can be achieved among different controllers in same level, if they are serving the same zone/area.

6.3.5.3 Handling failed links

In the OLC architecture, more than one SDN switch could be used in the same level of the control plane to provide dependent links for recovery purposes. In addition, these links can also be used for load balancing purposes during peak control signals load.

6.3.6 Handling subnet/network discovery (join, leave)

Since SDN has to complete its function as a proactive installer of rules in devices along the path between the sources and destinations, it needs a dynamic fast subnet discovery mechanism to give it a general view of all subnet information. In addition, it

should have a fast rediscovery time for covering any changes in the network, such as a new subnet joining or one leaving.

If a new subnet joins the network, the level 1 controllers in that subnet will start multicasting to all linked neighbor subnets, whilst simultaneously receiving DHP1s from them and then sending a DHP2 to the centralized controller in level+1 in order to get back the related general view. If a subnet leaves the network, the centralized controller will detect this through periodically monitoring the Still_alive field in the neighbors_discovery table. As a consequence of no activity from a subnet being for a specified time, a 0 will put in the Still_alive field. The centralized controller will check that field before send back the DHP2 to the relevant level 1's controller. If the Boolean value in that field is equal to 1, the DHP2 will contain the related subnet information, whilst if it is 0 the centralized controller will delete that subnet from the evaluation. Through the same mechanism, the controllers in the core network can detect the join and leave network in inter-domain networks by monitoring the activity of the edge controllers in them.

6.4 Implementation of OLC

In this section, we explain our OLC implementation for an open-level control plane in SDN networks in detail in relation to dynamic discovery in order to provide general view for single and multiple SDN networks covering a large area, as an addition to Ryu's [94] controller using an OpenVswitch (OVS) [91]. All the requirements set out in section 6.1 are met by OLC, which implements the DHP, thereby providing the controllers with a general view of all destination subnets/networks. It has been developed from the LLDP protocol, with DHP being the new feature. While the LLDP standard protocol just discovers the SDN switches inside one subnet, our proposal has the ability to discover all the subnets in the same network (i.e. intra-domain network), whilst also discovering other networks in different areas (i.e. inter-domain networks). To do so, the DHP has two parts as follows.

6.4.1 Implementation of the DHP distributed part in the DHP1 subunit

This part of the protocol is located in any level of controllers in the OLC architecture that are connected to their same level neighbor controllers using SDN switches, being

called the distributed part of the DHP protocol (DHP1). For example, during the bootstrap time the level 1 controllers will use this part of the DHP protocol in order to carry its own information to all neighbor controllers in different subnets in a distributed manner. Concurrently, so as to know which SDN switch connects the subnet to the other subnets, the controller monitors all the local switches using Packet_in messages.

If the Packet_in message is a DHP1 message and has an ID different to the local subnet's controller ID, then OLC will register the SDN switch which enters that DHP1 as an edge switch and put Switch_ID, Switch_port, Source_subnet_ID and the Timestamp of the message in the Neighbors_Discovery table. In order to implement the DHP1 piece, we define in the DHP protocol a new type-length-value (TLV) with number 124 class and two subclasses named Type_of_DHP and Subnet_ID. Whilst The Type_of_DHP is equal to 0 for discovery messages sent at the same level, the value of the Subnet_ID subclass can be calculated by performing an AND operation between the subnet IP and subnet Mask. Subsequently, the OLC model will create DHP1 messages and multicast them to all neighbors in unicast/multicast manner. As a consequence, the DHP helps the destination controller to know to which source controller it is connected with. Each controller that connects to its same level neighbor controller using an SDN switch has to use the DHP1 piece (e.g. level 1 controller). It will repeat this listening and sending after the specified time or reverts to reactive mode when there are changes in subnet states, such as adding a new edge switch. We implement algorithm 1 to show how these steps are carried out in the OLC model.

Algorithm 1. Distributed discovery for neighbors' subnets/networks in the same level (e.g. level 1)

Input: Local_IP , Local_Mask, level_of_contoller, Network_public_IP, Public Mask
Output: DHP1 packets with Local_Subnet_ID/Public_network_ID, Timestamp, Updating Neighbors_Discovery table

1: **START**
2: **Declare** General_ID Timestamp, Type_of_DHP
3: **Get** controller IP , Mask and Level_of_contoller from the system configuration
4: **IF** controller IP = Network_public_IP then # use NAT
5: General_ID ← **Calculate (**Network_public_IPs AND Public Mask)
6: **ELSE**

```
7:      General_ID ← Calculate (Subnet_IP AND subnet_Mask)
8: END IF
9: Type_of_DHP ←0       #To send DHP1s to the same level controllers
10: REPEAT
11:     Generate DHP1 packets with General_ID and Type_of_DHP
12:     Multicast/unicast the packets from each output port in each switch
13:     Listening to Packet_in to catch DHP1 packets
14:     Decapsulating each DHP1 packet
15:     Read Message_General_ID, Type_of_DHP from each decapsulated packet
16:     IF Message_General_ID ≠ General_ID AND Type_of_DHP = 0 then # Means
                                    the message came from a neighbor in the same
                                    level
17:             Update Neighbors_Discovery table
18:     END IF
19:     Wait the specified time or wait for any change in subnet conditions in reactive
        manner
20: UNTIL terminated by the administrator
```

6.4.2 Implementation of the DHP centralized part in the DHP2 subunit

The centralized part (DHP2) located in the controllers in level 2 and core controllers (levels 3 to n). There are two roles that can be performed using this piece of the protocol depending on the sent packet direction. The DHP2 subunit sends discovery packets to the controller one level up in order to calculate the best routes and then sends back the calculated information to the level -1 controllers so as to get a full view of other subnets/networks for proactive SDN behavior. The Type_of_DHP in DHP2 packets has the value 1, if the packet is sent up to level+1 and -1, if sent down to level-1.

A new subclass (net_inf.) adds to the DHP protocol in part 2. which has subnet/network information in a dictionary style (net_X:{net_Y:delay_Z}). Another subclass (Internal_latency) is added, which stores the internal latency for the subnets/networks that help in evaluating the best paths. Then, the controller in level+1 will collect all DHP2 packets from all level-1 controllers and waits for a specified time

to let all join and send their information to it. After this, the controller in level+1 uses the Dijkstra algorithm to evaluate the best paths from each subnet/network to others depending on the Internal_latency field and delay between the subnets/networks. Then, it will save this information in dictionary format and send it with two types of DHP2 packets, the first of which having Type_of_DHP equal +1 for a one level up controller in order to inform the core network about the network information. While the second type, with value -1, are sent to all controllers in level-1, which will save them in the All_view_discovery table.

The centralized controllers will repeat this procedure individually according to changes in network/subnet states. The full details for the centralized part can be seen in algorithm 2.

Algorithm 2. Centralized discovery for the whole subnets/networks (level-1 and level+1)

Input: Local_IP, Local Mask, Level_of_contoller, Neighbors_Discovery table, DHP2 messages from level-1, Network_public_IPs, Public Mask

Output: DHP2 packets to level+1 and level-1
Update All_topology and All_view_*discovery table*s

1: **START**
2: **Declare** General_ID, Type_of_DHP
3: **Get** Controller IP, Mask and Level_of_contoller from the system configuration
4: **IF** Controller IP = Network_public_IP then # use NAT
5: General_ID ← **Calculate (**Network_public_IPs AND Public Mask**)**
6: **ELSE**
7: General_ID ← **Calculate (**Subnet_IP AND subnet_Mask**)**
8: **END IF**
9: Type_of_DHP ← +1 or -1
10: **REPEAT**
//////////////////////////////////start send DHP2 one level up
11: **IF** Level_of_controller = 1 then
12: **Generate** DHP2 packet with (General_ID, Type_of_DHP, Inf. from Neighbors_discovery table)

13: **ELSE** #means all other levels

14: **Generate** DHP2 packet with (General_ID, Type_of_DHP, Inf. from All_topology table)

15: **END IF**

16: **Unicast** the packet to the level+1 centralized controller.

//End of send one level up

//////////////////////////////Start listening to catch DHP2 from up and down level

17: **Listening** to a specified port to catch DHP2 packet from level+1(back), -1

18: **Decapsulating** each DHP2 packet

19: **Read** Message_General_ID, Type_of_DHP from each decapsulated packet

20: **IF** Message_General_ID ≠ General_ID AND Type_of_DHP = +1 then # means the message came from a level+1 controller

21: **Update** All_view_discovery table

22: **IF** Level_of_controller ≠ (1 or 2), then #2 because we use NAT, 1 because there is no level 0 ///this is specific to the core network

23: **Generate** DHP2 packet with (Local_Subnet_ID, Type_of_DHP, Inf. from All_view_discovery)

24: **Unicast** generated packet to the level (-1) controllers.

25**:** **END IF**

26: **ELSE IF** Message_General_ID ≠ General_ID AND Type_of_DHP = -1 then # Means the message came from level-1 controller

27: **Wait** specified time # to collect all tables from level-1 controllers

28: **Analysis** tables

29: **Apply** Dijkstra algorithm to find best paths

30: **Generate** DHP2 packet with (General_ID, Type_of_DHP, Inf. from All_topology table to level+1 and All_view_discovery to level-1)

31: **Unicast** generated packet to the level (+1 and -1) controllers

32: **END IF**

33: **Wait** the specified time or wait for any change in any subnet/network's status in a reactive manner

34: **UNTIL** terminated by the administrator

6.5 Experimental results

In this section, an extensive number of testbed experiments are performed in four scenarios, with the results being presented to show the effectiveness of our proposed model. In the experiments, open source SDN Ryu is used as an OpenFlow controller and OVS as an OpenFlow switch. Both software are installed under Ubuntu 14.04 on 22 computers, with two of these PCs having the specifications of core i7, 3.4 GHz and 3.8 GiB memory, whilst the other 20 have specifications of Core 2 Quad, 2.66 GHz and 2.0

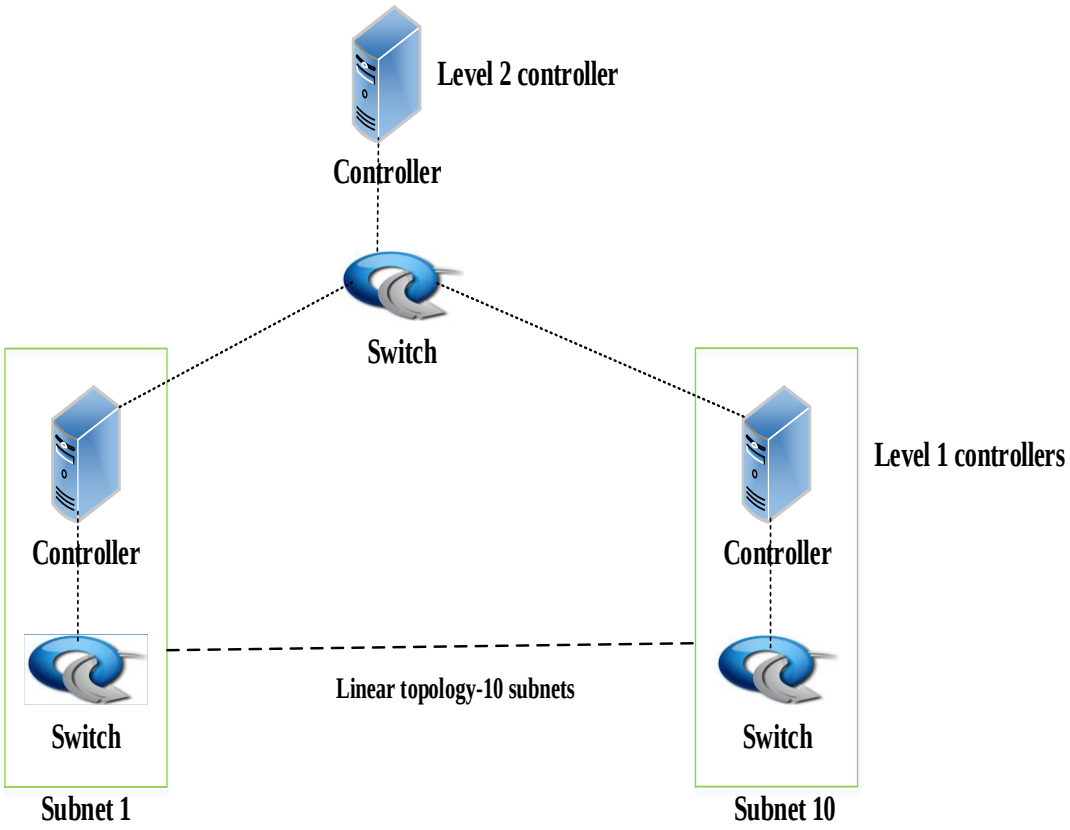


Figure 6.6: Logical diagram of built testbed with 22 computers

GiB memory. The connections between computers are made using Ethernet cables of different lengths of 1, 2 and 3 meters, with LAN cards of 1,000 Megabits per second being used in each computer; the logic diagram of the testbed environment can be seen in Figure 6.6.

The OLC components are implemented on Ryu. In order to approve the proposed model's performance and its suitability for large Ethernet networks four different scenarios are performed, all of them using linear topology because they rely on the number of hops. In the OLC experiments, two levels of controllers are used while in the fully distributed architectures one level is deployed. Each sub-experiment (i.e. result) is repeated five times and the average is taken, with the total number of runs of the testbed being 200 [i.e. 4 experiments*10 sub-experiments (i.e. results)*5 times. The four experimental scenarios are designed as follows:

- The first scenario is performed to measure the initial system discovery time for verifying the scalability and performance of OLC compared with fully distributed control plane architectures;

- The second scenario is run to measure the rediscovery time during no load on the network under both OLC and fully distributed control plane architectures;

- The third scenario is designed to evaluate the rediscovery time under load for both OLC and fully distributed control plane architectures;

- The fourth scenario is performed by increasing the number of subnets with the aim of evaluating the number of packets that are generated as a consequence in the data plane.

6.5.1 Initial system discovery time: comparison between OLC and fully distributed control plane architectures

Each system, when run from the shutdown state, takes time to reach steady state, called the boot time or bootstrap time. When connecting multiple subnets/networks it is important to measure this time for configuring the devices of the network and knowing when a steady state has been reached, such that services can be offered to the customers. A linear topology is used with an increasing number of subnets from 2 to 10, each having one SDN controller and one SDN switch. This experiment is deployed to evaluate the bootstrap discovery time under fully distributed control plane SDN architecture and the proposed open-level control plane SDN architecture. All the controllers and the switches are timed to work concurrently and we record the discovery time needed by each subnet to have a general view of the whole network. The worst (i.e. highest) time taken is usually by the edge subnet in the linear topology.

The fully distributed model discovery time statistics provided in Figure 6.7, show that when increasing the number of subnets from 2 to 10, the discovery time increases from 6.13 secs to 14.55 secs, which means it increases by 8.42 secs and the average is 10.27 secs. These experimental results are almost identical to the theoretical findings in Equation 6.3 (e.g. when the traditional architecture deals with 8 subnets, the bootstrap

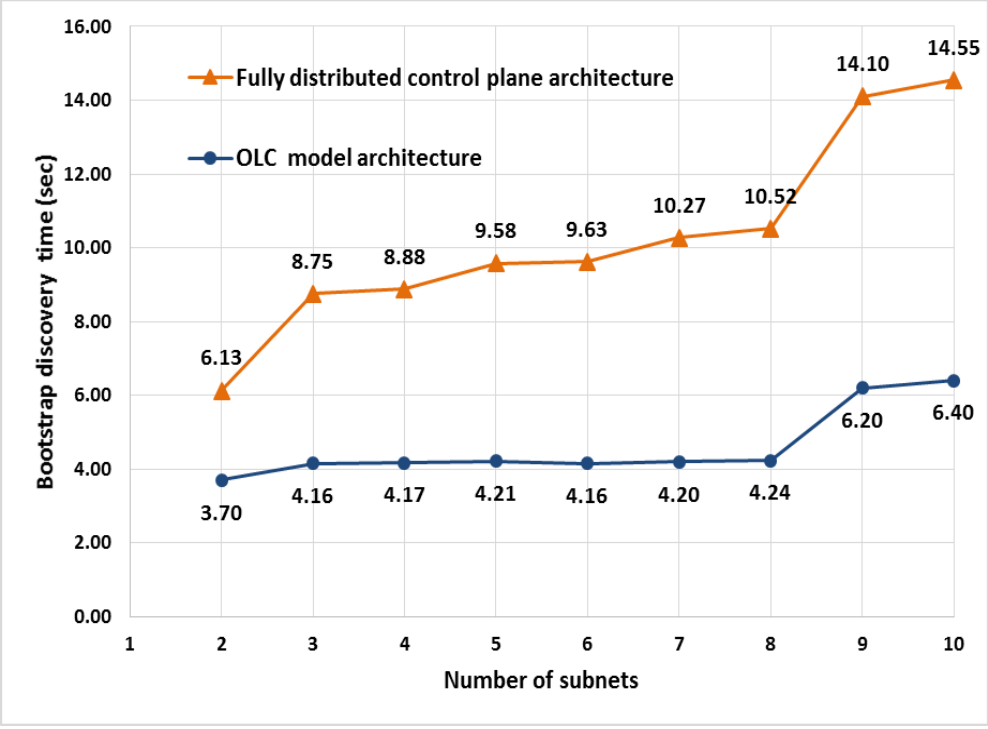


Figure 6.7: Bootstrap discovery time under the OLC and fully distributed aggregation mechanisms

discovery time is 10.01 secs theoretically and it is 10.52 secs experimentally). This trend of results occurs because each controller in each subnet needs to send multiple discovery packets during multiple phases through the network data plane which consumes time, as explained in detail in section 6.2. In addition, the phases are also overlapping with each other, which leads to the generation of more packets in the same link from both sides. As a result, there is congestion and an accompanying increase the time that needed to pass the information of each subnet to the neighbor subnets.

However, under the OLC model the discovery time only increases from 3.70 secs to 6.40 secs when the number of subnets is increase from 2 to 10 subnets. Moreover, the average bootstrap discovery time using OLC is 4.6 secs, which is nearly the same as the theoretical findings in Equation 6.7 (i.e. 4.8 secs). This, in turn, means the OLC discovery mechanism can discover a network that contains 2 to 10 subnets approximately 55.2% faster than the fully distributed aggregation mechanism.

The OLC model has the ability to discover at this speed, because it has multi-level control plane architecture, which leads to the allocation of a different control plane for different discovery phases. Specifically, the next neighbors are discovered within the first phase, which in this experiment have been allocated to the level1 controllers, while the other phase is allocated to the level2 controllers, which have centralized architecture. As a consequence, the network under the OLC model can scale to 32 subnets within the same discovery time (14.55 secs) that is needed by fully distributed discovery architecture for discovering 10 subnets, the equation for this calculation is the number of subnets*average change in discovery time (which pertains to how much time is needed to add a new subnet) + first state time = 14.55 secs → number of subnets*0.337+ 3.70 = 14.55 secs → 32 subnets. This means that OLC scales the network 3.2 times more than the fully distributed discovery architecture.

6.5.2 Rediscovery time without load: comparison between OLC and fully distributed control plane architectures

In this experiment, linear topology is used and the time needed for rediscovery is calculated, firstly, to detect a new event (e.g. add a new subnet to the edge of the network) and secondly, to distribute that new information to the whole network's subnets in order to update their switching tables. This experiment is deployed under fully distributed control plane SDN architecture and OLC architecture, as can be seen in Figure 6.8.

Regarding the statistics of the fully distributed architecture, it can be seen that when increasing the number of subnets from 2 to 10, the rediscovery time will increase by 9.23 secs with two different leaps (leap to 6.26 secs from 4.12 secs and to 11.11 secs from 7.3 secs). The rediscovery time in the fully distributed architecture has this trend because when adding a new subnet to the edge of the linear topology, the rediscovery time that is needed by furthest subnets will be impacted by the number of phases (Nop) to get the new added subnet's information multiple by the Latency of each phase (LP), as described in Equation 6.3 in section 6.2. As a consequence, we can expect more delay when we scale the network. The trend of leaps because the discovery protocol between five and eight subnets will increase the size of discovery packets and repeat this every four added subnets.

Regarding the statistics of the network under OLC, the rediscovery time is slightly increased from 3.7 secs to 5.34 secs, i.e. by 1.64 secs and with one small leap from 2.1 secs to 3.7 secs. This is because the level1 controllers just perform one distributed phase with their neighbor subnets, which has the most impact on the rediscovery time, then the remaining time is consumed by level2's controllers to multicast/unicast the switching tables to all the subnets. In addition, because OLC uses separated open-level control planes there will be no congestion on data plane links, which enhances the rediscovery

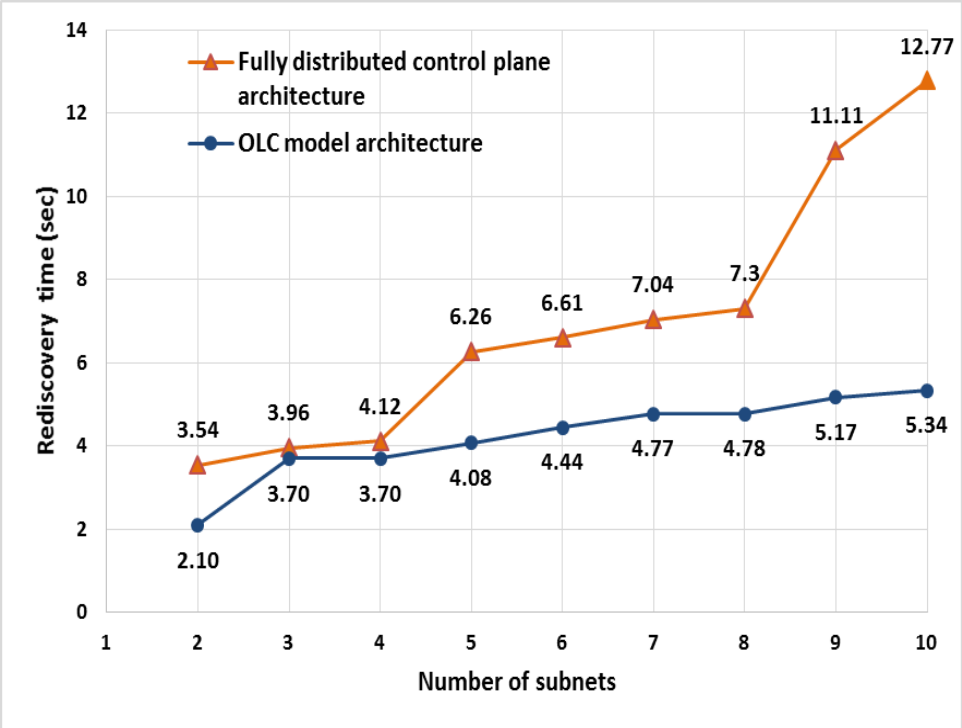


Figure 6.8: Rediscovery time for network events during no load on the network under the OLC and fully distributed aggregation mechanisms

time and this is opposite to the behavior of the fully distributed architecture. In addition, in terms of scaling up the number of subnets in relation to rediscovery time, the OLC can scale up the number of subnets by 313% when compared to fully distributed architecture.

6.5.3 The efficiency during load: comparison between OLC and fully distributed control plane architectures

The efficiency and effectiveness of the OLC model during different load rates are presented in this experiment, where the rediscovery time has been evaluated during a range of (200-33,333,333) requests per seconds (RPS). Three fix subnets are connected together using a linear topology and then we generate loads from 20 virtual hosts on the link between subnets 2 and 3 in order to make congestion on that link. Subsequently, a new subnet is added to the third subnet and the time needed by all the other subnets to discover that event is recorded.

Regarding the fully distributed architecture evaluation, Figure 6.9 shows that with an increase in the load from 200 to 222,222 RPS the rediscovery time is increased significantly from 20.78 secs to 80.07 secs. After that, it fluctuates with an average of 70.5 secs for loads between 333,333-13,333,333 RPS and then rises notably to reach 101 secs for 33,333,333 RPS. This trend occurs because the increase in the number of requests per second on the link between the second and third switch leads to more collisions and competition to use that link, which results in more congestion that impedes the discovery packets passing link and hence, causes a delay in rediscovery time.

In contrast, under the same circumstances, the OLC provides efficiency during different load rates, offering approximately a steady rediscovery time with an average of 4.34 secs. The reason behind this is that just one phase needs to be performed on the congested link in the data plane, which means that only one packet from each connected subnet passing that link is enough to let all the other subnets know about any new events. In addition, the centralized controller plays a big role in terms of multicasting/unicasting any changes so as to update the whole network. As a consequence, when comparing the averages of both models the proposed model has 93.5 % greater efficiency enhancement than the full distributed architecture by reducing the response time.

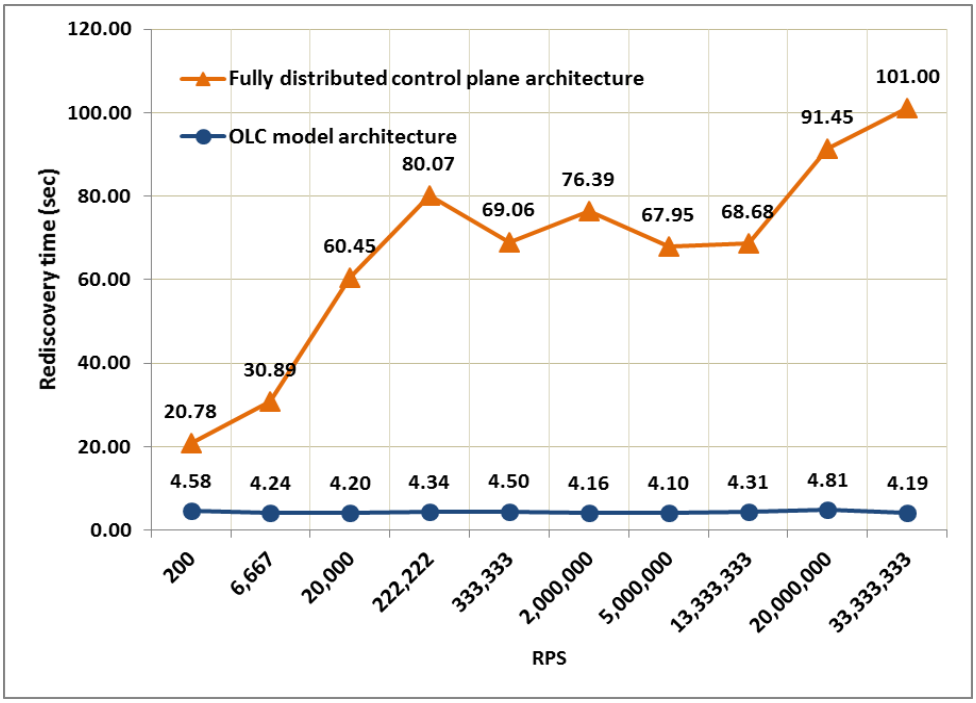


Figure 6.9: Models' efficiency during load

6.5.4 Data plane bandwidth consumption: comparison between the OLC and fully distributed control plane architectures

The bandwidth consumption from the discovery process is evaluated using the number of discovery phases and number of discovery packets, which are generated on the data plane to provide all the subnets a general view of the whole network. A linear topology is used whilst increasing the number of subnets from 2 to 10 and the Wireshark tool is used to evaluate the number of discovery packets and how many phases are needed to complete one discovery process.

Regarding the discovery process statistics under distributed architecture, as can be seen in Figure 6.10, when increasing in the number of subnets from two to 10 the number of phases is increased linearly from one to nine phases, which is identical with the theoretical finding in section 6.2, where this is equal to the best path between the furthest apart subnets.

As a consequence of using the linear topology, the number of phases is equal to the number of subnets minus 1. In addition, the number of discovery packets in the distributed architecture generated in the data plane increases exponentially from two to 162 packets. This is because approximately the same number of packets are generated per each phase

in the network in order to complete the whole discovery process. This, in turn, leads to more congestion on the links, which increases when the network is scaled up.

On other hand, the OLC model generates just one fix phase in the data plane without any effect due an increase in the number of subnets. This is because it relies on a separated open-level control plane architecture, where the centralized controller performs the second phase discovery of the network. Regarding the number of discovery packets in the data plane, this increases by a rate of two for every new subnet added to the network. This is because every new subnet sends and receives one discovery packet with its neighbour in linear topology, if it is at the edge of the network. As a consequence of all

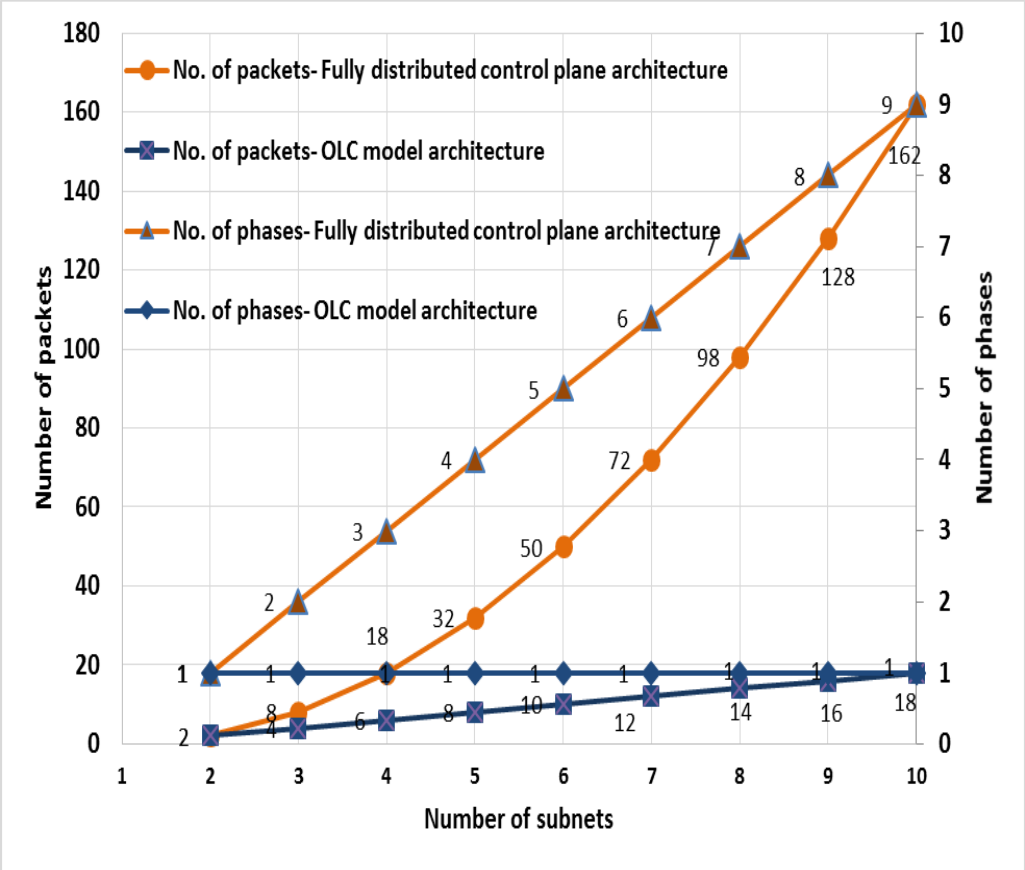


Figure 6.10: Number of discovery packets and phases generated under the OLC and fully distributed aggregation mechanisms

of the above, in this experiment, the OLC model, on average, reduces bandwidth consumption by about 84% more than the fully distributed discovery architecture. As the data plane is an important part of the network for transferring these data among the subnets, it is essential to decrease the load on that plane [76], which can be achieved by using the proposed model. In addition, in terms of increasing the number of subnets in

relation to the number of packets in data plane, OLC can scale it by 820% when compared to the fully distributed architecture.

6.6 Summary

In this chapter, the limitations of current distributed architectures and aggregation discovery mechanisms which are used to provide a network general view have been analysed. Subsequently, SDN based OLC architecture has been introduced and implemented to provide a general view discovery process taking into account all the fundamental requirements. By implementing an actual testbed and after an extensive set of experiments, the results have demonstrated that OLC offers a reduction in the number of discovery packets in the data plane to the average of 10 packets, 5.67 (10.27-4.6) secs faster discovery time and scaling up the number of subnets in an SDN network 3.2 times more than with the traditional distributed architecture and mechanism. Moreover, it provides an approximately steady rediscovery time of 4.34 secs even with a very high load.

Chapter 7

Conclusions, Impact and Future work

This chapter contains three parts, firstly, the main findings are presented and the conclusions to the whole thesis are provided. The second explains the potential impact of the study outcomes, while the last part considers other developments as well as containing suggestions in relation to future work.

7.1 Conclusions

In this thesis, scalability has been defined and different metrics for measuring it have been reviewed. In addition, a series of proposals have been made throughout the contributions chapters that have covered the enhancing of network scalability, starting with the smallest network unit, i.e. a LAN, and then regarding the biggest, i.e. an inter-domain network. This involved analyzing the legacy network designs, architectures, mechanisms and protocols to find their limitations that hinder scaling up of networks. Then, others' work has been studied to identify the gaps that have prevented them from solve the scalability issue efficiently and hence, why their models have not been officially used.

When delivering the contributions of this study, a balance between scalability and other important network factors, including reliability, security, complexity, performance and efficiency has been central to the process. The findings from this thesis lead to the conclusion these contributions are very promising. The salient outcomes in terms percentage improvements delivered by the proposed procedures over legacy architectures and mechanisms are summarized in the following.

- Enhancements provided in chapter 4: After thorough analysis of the broadcast mechanism and its limitations regarding network scalability, the SSED model was

proposed, thereby providing several contributions to the field. The main conclusions after applying this model to an actual testbed are that it can: eliminate broadcast packets from the network completely; provide a reduction in the consumption of network resources in the data plane by 27800 packets; reduce control packets in the control plane to 8,022 during bootstrap time; reduce peak overhead on the controller, thus preventing failed requests; enhance controller security; offer better response time; and provide more efficient performance. Moreover, it can provide better scalability such that it scales up the number of hosts to 2500-3076 hosts and the number of switches to 161 switches when compared with the legacy broadcast-based architectures.

- Enhancements delivered in chapter 5: As a consequence of highlighting how the middlebox devices (e.g. routers) are one of the causes of restricting networks (which contains multi-subnets) from scaling up, the router device and its default gateway mechanism were analyzed in depth. MSSERD has been proposed in multi-subnet distributed architectures to overcome router and default gateway mechanism limitations and enhance network scalability. Through building an actual testbed and implementing several experiments, comprehensive results have shown that MSSERD enhances scalability in terms of the number of subnets up to 52 subnets when compared to the legacy architecture. In addition, it improves the efficiency significantly, especially with high load, by reducing the overhead in the control and data planes through reducing bandwidth consumption over legacy mechanisms. In addition, it improves performance by 2.3 times, enhances reliability by providing 0% packet loss when dealing with 60,000 packets with rate 3333 RPS and reduces complexity when compared to legacy mechanisms by removing several of the network's manual configurations.

- Enhancements delivered from chapter 6: The third reason for restricted network scalability, i.e. the discovery aggregation mechanism, has been investigated in detail. The OLC model has been introduced, which contains open-level control plane architecture to use in intra and inter domains as well as the DHP discovery protocol which has the fastest discovery mechanism to the best of this researcher's knowledge. By implementing an actual testbed and after an extensive set of

experiments, the results have demonstrated that OLC offers a reduction in the number of discovery packets in the data plane to the average of 10 packets, 5.67 secs faster discovery time and scaling up of the number of subnets in an SDN network to 31 - 82 subnets over the traditional distributed architecture and mechanism. Moreover, it provides an approximately steady rediscovery time of 4.34 secs even with a very high load.

7.2 Future work

The results after applying a series of proposals in this thesis are encouraging and the models are promising. However, these accomplishments should be developed further to meet future changes and requirements; hopefully the models in this study will be officially used. This thesis opens the road to several other developments and here, some suggestions for future work are put forward.

7.2.1 Short term future work

The following developments, enhancements and suggestions need to be considered in the near future:

- Testing SSED for all well-known broadcast protocols and services;
- Applying SSED to load balancing among more than one server in a data centre network/campus network in order to increase scalability in terms of the number of end users per domain;
- Connecting MSSERD-based subnets to the Internet to assess the system with daily traffic;
- Developing MSSERD to support load balancing and multi-path transmissions in a LAN-based network, which will lead to improved network scalability in terms of response time;
- Connecting OLC to the Internet to check its validity for dealing with daily traffic load;
- Investigating how to enhance the MDP and DHP protocols to carry different control messages;

- Implementing a core network prototype using the OLC architecture and testing it by applying it across several virtual campus networks.

7.2.2 Long term future work

Suggested long term future work is briefly outlined as follows:

- The proposed models are pure SDN-based models so it would be beneficial if they were developed to support hybrid network devices (i.e. legacy and SDN devices);
- NFV technology could be deployed more than it was for this thesis by concatenation with the proposed models for enhancing network scalability further. For example, applying distributed NFV technology inside one large network could lead to a reduction in response time for several discovery-based protocols and in turn, enhance scalability.

References

[1] SDNCentral LLC, "Special Report : OpenFlow and SDN – State of the Union," 2016.

[2] D. Cotroneo, R. Natella, and S. Rosiello, "NFV-Throttle: An Overload Control Framework for Network Function Virtualization," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, no. 4, pp. 949–963, Dec. 2017.

[3] X. Meng, V. Pappas, and L. Zhang, "Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement.pdf," *2010 Proc. IEEE INFOCOM*, pp. 1–9, 2010.

[4] GSMA, "Data demand explained," 2015.

[5] Y. Hu and T. Li, "Towards efficient server architecture for virtualized network function deployment: Implications and implementations," *Proc. Annu. Int. Symp. Microarchitecture, MICRO*, vol. 2016–Decem, 2016.

[6] N. J. Mohamed, S. Sahib, N. Suryana, and B. Hussin, "Understanding network congestion effects on performance - Articles review," *J. Theor. Appl. Inf. Technol.*, vol. 92, no. 2, pp. 311–321, 2016.

[7] J. A. Bivens, B. K. Szymanski, and M. J. Embrechts, "Network Congestion Arbitration and Source Problem Prediction Using Neural Networks," *Int. J. Smart Eng. Syst. Des.*, vol. 4, no. January 2015, pp. 243–252, 2002.

[8] M. Saeed Ansari and A. Mahani, "Reliability evaluation methods for resilient wireless sensor networks," in *Mobile Computing and Wireless Networks: Concepts, Methodologies, Tools, and Applications*, vol. 3–4, 2015, p. 164.

[9] T. Mizrahi, E. Saat, and Y. Moses, "Timed Consistent Network Updates in Software-Defined Networks," *IEEE/ACM Trans. Netw.*, vol. 24, no. 6, pp. 3412–3425, 2016.

[10] J. Menga, "VLAN Operations," in *CCNP practical studies: Switching*, 1st ed., Cisco Press, 2003, p. 150.

[11] S. Krakowiak, S. K. Shrivastava, and Eds., "Advances in distributed systems," in *Advanced distributed computing - from Algorithms to systems*, Berlin,Germany: Springer-Verlag, 2000, p. 281.

[12] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in Software-Defined Networking (SDN)," *Comput. Networks*, vol.

112, pp. 279–293, 2017.

[13] F. Benamrane, M. Ben Mamoun, and R. Benaini, “Performances of OpenFlow-based software-defined networks: an overview,” *J. Networks*, vol. 10, no. 6, pp. 329–338, 2015.

[14] K. He *et al.*, “Latency in software defined networks: Measurements and mitigation techniques,” in *ACM SIGMETRICS Performance Evaluation Review*, 2015, vol. 43, no. 1, pp. 435–436.

[15] P. Isaia and L. Guan, “Performance benchmarking of SDN experimental platforms,” in *NetSoft Conference and Workshops (NetSoft), 2016 IEEE*, 2016, pp. 116–120.

[16] H. Cho, S. Kang, and Y. Lee, “Centralized ARP proxy server over SDN controller to cut down ARP broadcast in large-scale data center networks,” in *Information Networking (ICOIN), 2015 International Conference on*, 2015, pp. 301–306.

[17] W. Jian, H. Tao, L. Jiang, and L. Yunjie, “A novel floodless service discovery mechanism designed for Software-Defined Networking,” *China Commun.*, vol. 11, no. 2, pp. 12–25, 2014.

[18] M. A. Harrabi, M. Jeridi, N. Amri, M. R. Jerbi, A. Jhine, and H. Khamassi, “Implementing NFV routers and SDN controllers in MPLS architecture,” in *Information Technology and Computer Applications Congress (WCITCA), 2015 World Congress on*, 2015, pp. 1–6.

[19] M. Hayashitani, Y. Hasegawa, K. Suzuki, and Y. Mizukoshi, “Flexible and automated operational control in SDN transport-base virtual router,” *Conf. Opt. Fiber Commun. Tech. Dig. Ser.*, pp. 5–7, 2014.

[20] T. Koponen *et al.*, “Onix: A distributed control platform for large-scale production networks.,” in *OSDI*, 2010, vol. 10, pp. 1–6.

[21] A. Tootoonchian and Y. Ganjali, “Hyperflow: A distributed control plane for openflow,” in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, 2010, p. 3.

[22] P. W. Chi, Y. C. Huang, J. W. Guo, and C. L. Lei, “Give me a broadcast-free network,” in *2014 IEEE Global Communications Conference*, 2014, pp. 1968–1973.

[23] K. Phemius, M. Bouet, and J. Leguay, “DISCO: Distributed multi-domain SDN controllers,” in *2014 IEEE Network Operations and Management Symposium (NOMS)*, 2014, pp. 1–4.

[24] X. Tu, X. Li, J. Zhou, and S. Chen, “Splicing MPLS and OpenFlow tunnels based on SDN paradigm,” *Proc. - 2014 IEEE Int. Conf. Cloud Eng. IC2E 2014*, pp. 489–493, 2014.

[25] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, “On scalability of software-defined networking,” *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 136–141, Feb. 2013.

[26] P. Dordal, “Ethernet,” in *An Introduction to Computer Networks*, 1.8.22 ed., 2016, p. 48.

[27] The Institute of Electrical and Electronics Engineers, “IEEE Standards for Local and Metropolitan Area Networks:,” 1990.

[28] “RFC1812 :Requirements for IP Version 4 Routers,” 1995.

[29] K. S. Mishra and A. K. Tripathi, “Some Issues, Challenges and Problems of Distributed Software System,” *Int. J. Comput. Sci. Inf. Technol. Varanasi, India*, vol. 7, p. 3, 2014.

[30] M. A. N. S. Committee and I. C. Society, *Draft Standard for Local and Metropolitan Networks: Station and Media Access Control Connectivity Discovery*. 2016.

[31] J. Moy, “RFC 2328: OSPF Version 2,” 1998.

[32] Y. Rekhter, T. Li, and S. Hares, “RFC 4271: A Border Gateway Protocol 4 (BGP-4) Status,” 2006.

[33] Y. K. Singh, Upadhya, and T.K.S.A.D.B., “Internet technology,” in *Education Technology:teaching Learning*, APH Publishing Corporation, 2008, p. 111.

[34] A. Murphy, “Virtualization defined - eight different ways,” *F5 Networks, Inc.*, pp. 1–3, 2015.

[35] Google Inc., “Google IPv6 Adoption,” *Google*, 2017. [Online]. Available: https://www.google.com/intl/en/ipv6/statistics.html. [Accessed: 27-Dec-2017].

[36] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to parallel computing: design and analysis of algorithms*, vol. 400. Benjamin/Cummings Redwood City, CA, 1994.

[37] S. Sezer *et al.*, “Are we ready for SDN? Implementation challenges for software-defined networks,” *IEEE Communications Magazine*, vol. 51, no. 7, IEEE, pp. 36–43, 2013.

[38] M. Karakus and A. Durresi, “A Scalability Metric for Control Planes in Software Defined Networks (SDNs),” in *Advanced Information Networking and Applications (AINA), 2016 IEEE 30th International Conference on*, 2016, pp. 282–

289.

[39] P. Jogalekar and M. Woodside, “Evaluating the scalability of distributed systems,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 6, pp. 589–603, Jun. 2000.

[40] Open Networking Foundation, “OpenFlow,” 2017. [Online]. Available: https://www.opennetworking.org/sdn-resources/openflow. [Accessed: 31-May-2017].

[41] L. I. Dan, W. Songtao, Z. H. U. Konglin, and X. I. A. Shutao, “A survey of network update in SDN,” *Front. Comput. Sci.*, vol. 11, no. 1, pp. 4–12, 2017.

[42] N. McKeown *et al.*, “OpenFlow: enabling innovation in campus networks,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[43] N. Varis and J. Manner, “Evaluation of link layer mobility in Ethernet networks,” *Comput. Commun.*, vol. 103, pp. 193–209, 2017.

[44] Y. Trivedi, “Ethernet, the networking standard: more mature, more powerful where the whole world is going with ethernet,” *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 5–11, 2016.

[45] A. S. M. Asadujjaman, S. S. Moni, and M. S. Alam, “FCSEA: A Floodless Carrier-grade Scalable Ethernet Architecture,” in *Electrical and Computer Engineering (ICECE), 2016 9th International Conference on*, 2016, pp. 427–430.

[46] P. Reviriego, J. Lopez, M. Sanchez-Renedo, V. Petrovic, J.-F. Dufour, and J.-S. Weil, “The space ethernet physical layer transceiver (sephy) project: a step towards reliable ethernet in space,” *IEEE Aerosp. Electron. Syst. Mag.*, vol. 32, no. 1, pp. 24–28, 2017.

[47] M. Benito, E. Vallejo, R. Beivide, and C. Izu, “Extending commodity OpenFlow switches for large-scale HPC deployments,” in *High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB), 2017 IEEE 3rd International Workshop on*, 2017, pp. 41–48.

[48] A. Balchunas, “Network Address Translation 1.22,” *Topology*, pp. 1–5, 2013.

[49] L. Colitti, V. Cerf, Google, S. Cheshire, D. Schinazi, and A. Inc., “RFC 7934: Host Address Availability Recommendations,” 2016.

[50] Cisco Systems Inc., “Configuring Network Address Translation : Getting Started,” in *Security Guide, Cisco ACE Application Control Engine*, 2006, pp. 1–12.

[51] S. Berinato, “All Systems Down,” *CIO Magazine*, vol. 10, no. 54, 2003.

[52] C. Kim, M. Caesar, and J. Rexford, “Floodless in seattle: a scalable ethernet architecture for large enterprises,” *ACM SIGCOMM Comput. Commun. Rev.*, vol.

38, no. 4, pp. 3–14, 2008.

[53] K. Elmeleegy and A. L. Cox, “EtherProxy: Scaling Ethernet By Suppressing Broadcast Traffic,” in *IEEE INFOCOM 2009*, 2009, pp. 1584–1592.

[54] A. Shpiner, I. Keslassy, C. Arad, T. Mizrahi, and Y. Revah, “SAL: Scaling data centers using Smart Address Learning,” in *10th International Conference on Network and Service Management (CNSM) and Workshop*, 2014, pp. 248–253.

[55] R. Niranjan Mysore *et al.*, “Portland: a scalable fault-tolerant layer 2 data center network fabric,” in *ACM SIGCOMM Computer Communication Review*, 2009, vol. 39, no. 4, pp. 39–50.

[56] D. Jorm, “SDN and Security,” *ONOS Project*, 2015. [Online]. Available: http://onosproject.org/2015/04/03/sdn-and-security-david-jorm/.

[57] R. Froom, B. Sivasubramanian, and E. Frahim, “Implementing Cisco IP Switched Networks (SWITCH) Foundation Learning Guide: Foundation learning for switch 642-813,” Cisco Press: Cisco Press, 2010, pp. 289–291.

[58] B. Zhang, G. Wang, A. Y. Zhu, and T. S. E. E. Ng, “Router group monitoring: making traffic trajectory error detection more efficient,” *IEEE Trans. Netw. Serv. Manag.*, vol. 7, no. 3, pp. 158–171, Sep. 2010.

[59] C. N. Academy, “Scaling networks: Companion guide,” Cisco Press, 2009.

[60] R. Froom and E. Frahim, “Implementing Cisco IP Switched Networks (SWITCH) Foundation Learning Guide: CCNP SWITCH 300-115,” United States: Cisco Press, 2015.

[61] M. DeCarlo, “A list of common default Router IP addresses,” *TechSpot*, 2015. [Online]. Available: http://www.techspot.com/guides/287-default-router-ip-addresses/. [Accessed: 27-May-2017].

[62] Protogenius, “Address Resolution Protocol,” San Jose, 2006.

[63] Y. Zhang, Z. M. Mao, and J. Wang, “A firewall for routers: Protecting against routing misbehavior,” *Proc. Int. Conf. Dependable Syst. Networks*, pp. 20–29, 2007.

[64] K. Zheng, L. Wang, B. Yang, Y. Sun, and S. Uhlig, “LazyCtrl: A Scalable Hybrid Network Control Plane Design for Cloud Data Centers,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 1, pp. 115–127, Jan. 2017.

[65] Cisco Systems Inc., “Configuring STP and MST,” in *Cisco 7600 Series Router Cisco IOS Software Configuration Guide*, 12.1E., United States: Corporate Headquarters, 2003, pp. 1–2.

[66] P. Molinero-Fernández, “Circuit Switching in the Internet,” Stanford Univ., 2003.

[67] S. Kaur, K. Kaur, and V. Gupta, “Implementing Static Router based on Software Defined Networking,” *2016 Int. Conf. Comput. Tech. Inf. Commun. Technol. ICCTICT 2016 - Proc.*, pp. 358–360, 2016.

[68] M. Shen, B. Chen, X. Zhu, and Y. Zhao, “Towards optimal cache decision for campus networks with content-centric network routers,” *Proc. - IEEE Symp. Comput. Commun.*, vol. 2016–Augus, pp. 810–815, 2016.

[69] S. S. W. Lee *et al.*, “Design of SDN based large multi-tenant data center networks,” *2015 IEEE 4th Int. Conf. Cloud Networking, CloudNet 2015*, pp. 44–50, 2015.

[70] M. Corici, B. Reichel, B. Bochow, and T. Magedanz, “An SDN-based solution for increasing flexibility and reliability of dedicated network environments,” in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2016, pp. 1–6.

[71] J. Wang, G. Shou, Y. Hu, and Z. Guo, “A Multi-Domain SDN Scalability Architecture Implementation Based on the Coordinate Controller,” in *2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 2016, pp. 494–499.

[72] M. Azab and J. A. B. Fortes, “Towards proactive SDN-controller attack and failure resilience,” in *2017 International Conference on Computing, Networking and Communications (ICNC)*, 2017, pp. 442–448.

[73] M. T. I. ul Huque, W. Si, G. Jourjon, and V. Gramoli, “Large-Scale Dynamic Controller Placement,” *IEEE Trans. Netw. Serv. Manag.*, vol. 14, no. 1, pp. 63–76, Mar. 2017.

[74] F. J. Rodríguez, S. Fernandez, I. Sanz, M. Moranchel, and E. J. Bueno, “Distributed Approach for SmartGrids Reconfiguration Based on the OSPF Routing Protocol,” *IEEE Trans. Ind. Informatics*, vol. 12, no. 2, pp. 864–871, 2016.

[75] C. Cachin, R. Guerraoui, and L. Rodrigues, *Introduction to reliable and secure distributed programming*, 2nd ed. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2011.

[76] J. Aspnes *et al.*, “Eight Open Problems in Distributed Computing.,” *Bull. EATCS*, vol. 90, pp. 109–126, 2006.

[77] J. Doyle, “scaling,” in *OSPF and IS-IS: Choosing an IGP for Large-Scale Networks: Choosing an IGP for Large-Scale Networks*, Addison-Wesley

Professional, 2005.

[78] F. Botelho, T. A. Ribeiro, P. Ferreira, F. M. V Ramos, and A. Bessani, “Design and Implementation of a Consistent Data Store for a Distributed SDN Control Plane,” in *Dependable Computing Conference (EDCC), 2016 12th European*, 2016, pp. 169–180.

[79] S. Hassas Yeganeh and Y. Ganjali, “Kandoo: a framework for efficient and scalable offloading of control applications,” in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 19–24.

[80] B. J. van Asten, N. L. M. van Adrichem, and F. A. Kuipers, “Scalability and Resilience of Software-Defined Networking: An Overview,” *CoRR*, vol. abs/1408.6, 2014.

[81] P. T. Congdon, P. Mohapatra, M. Farrens, and V. Akella, “Simultaneously reducing latency and power consumption in openflow switches,” *IEEE/ACM Trans. Netw.*, vol. 22, no. 3, pp. 1007–1020, 2014.

[82] D. C. Plummer, “An Ethernet Address Resolution Protocol -- or -- Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware,” 1982.

[83] R. Droms, “Dynamic Host Configuration Protocol,” 1997.

[84] L. M. S. Committee, “IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges,” 2004.

[85] LIP6, R. Vida, and L. Costa, “Multicast listener discovery version 2 (MLDv2) for IPv6,” 2004.

[86] D. L. Mills, “RFC1305: Network Time Protocol (Version 3) specification, implementation and analysis,” 1992.

[87] M. S. and L. E. Joe Touch and M. K. E. L. A. M. Kumiko Ono, “Service Name and Transport Protocol Port Number Registry,” *Internet Assigned Numbers Authority*, 2012. [Online]. Available: www.iana.org.

[88] M. Bienkowski, A. Feldmann, J. Grassler, G. Schaffrath, and S. Schmid, “The Wide-Area Virtual Service Migration Problem: A Competitive Analysis Approach,” *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 165–178, Feb. 2014.

[89] Open network foundation, “Openflow switch specification, version 1.5.0,” 2014.

[90] G. Yao, J. Bi, and L. Guo, “On the cascading failures of multi-controllers in Software Defined Networks,” in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, 2013, pp. 1–2.

[91] O. V. Switch, “Open vSwitch,” 2014.

[92] U. C. Kozat, G. Liang, K. Kokten, and J. Tapolcai, “On Optimal Topology Verification and Failure Localization for Software Defined Networks,” *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2899–2912, Oct. 2016.

[93] S. Cheshire, “IPv4 Address Conflict Detection,” 2008.

[94] R. Kubo, T. Fujita, Y. Agawa, and H. Suzuki, “Ryu SDN framework-open-source SDN platform software,” *NTT Tech. Rev.*, vol. 12, no. 8, 2014.

[95] D. A. R. P. O. Agency, “Transmission Control Protocol DARPA Internet Program Protocol Specification,” 1981.

[96] B. Edgeworth, A. Foss, and R. G. Rios, “IP Routing on Cisco IOS, IOS XE, and IOS XR,” in *An essential guide to understanding and implementing IP routing protocols*, United State: Cisco Press, 2015, pp. 30–36.

[97] M. Arregoces and M. Portolani, “Data center fundamentals,” in *understand data center network design and infrastructure architecture, including load balancing, SSL, and security*, Indianapolis: Cisco Press, 2003, p. 526.

[98] Cisco Systems Inc., “Configuring Layer 3 Interfaces,” in *Catalyst 4500 Series Switch Cisco IOS Software Configuration Guide*, 12.2(37)SG., United States: Americas Headquarters, 2007, p. 2.

[99] A. Stevens and I. Pro, “Layer 2 and Layer 3 switches,” *IT PRO*, 2006. [Online]. Available: http://www.itpro.co.uk/88699/layer-2-and-layer-3-switches. [Accessed: 27-May-2017].

[100] Cisco Systems Inc., “High Availability Campus Network Design — Routed Access Layer using EIGRP or OSPF,” in *High Availability Campus Network Design — Routed Access Layer using EIGRP or OSPF*, no. 6387, United States: Americas Headquarters, 2007, pp. 2, 41.

[101] R. Deal, *CCNA Cisco Certified Network Associate Study Guide (Exam 640-802)*, 3rd ed. New York: McGraw-Hill Education, 2008.

[102] E. Alasadi and H. S. Al-Raweshidy, “SSED: Servers Under Software-Defined Network Architectures to Eliminate Discovery Messages,” *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 104–117, Feb. 2018.

[103] H. Huang, S. Guo, P. Li, W. Liang, and A. Y. Zomaya, “Cost Minimization for Rule Caching in Software Defined Networking,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 1007–1016, Apr. 2016.

[104] J. Gillette, R. Kovac, and I. Pelipenko, “Sizing Up the Gigabit Ethernet Switch

Players," *Network World*, p. 54, 2000.

[105] D. K. Bhattacharyya and J. K. Kalita, "Network anomaly detection," in *A machine learning perspective*, CRC Press, 2014, p. 277.

[106] S. Ray, Y. Jin, and A. Raychowdhury, "The Changing Computing Paradigm With Internet of Things: A Tutorial Introduction," *IEEE Des. Test*, vol. 33, no. 2, pp. 76–96, Apr. 2016.

[107] B. Kar, E. H. K. Wu, and Y. D. Lin, "The Budgeted Maximum Coverage Problem in Partially Deployed Software Defined Networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 13, no. 3, pp. 394–406, Sep. 2016.

[108] P. S. (Lucent Technologies) and M. H. (Lucent Technologies), "RFC2663:IP Network Address Translator (NAT) Terminology and Considerations," 1999.

Publications

Most of the work in this study has been submitted for publication and is currently under review, with details of a published one being provided below.

1. E. Alasadi and H. S. Al-Raweshidy, "SSED: Servers Under Software-Defined Network Architectures to Eliminate Discovery Messages," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 104–117, Feb. 2018.