Route aggregation in Software-defined Networks

SYED AMIR SHAHZAD

KTH Information and Communication Technology

Degree project in
Communication Systems
Second level, 30.0 HEC
Stockholm, Sweden

Route aggregation in Software-defined Networks

Syed Amir Shahzad

Master of Science Thesis

Communication Systems
School of Information and Communication Technology
KTH Royal Institute of Technology
Stockholm, Sweden

23rd June 2013

Examiner: Professor Gerald Q. "Chip" Maguire Jr.

Abstract

Software-defined Networking (SDN) is an emerging trend in communication networks that facilitates decoupling the control and data plane of multilayer switches. A logically centralized controller hosted on a server configures the forwarding tables (flow tables) of switches in order to route the various data flows. To implement SDN, OpenFlow technology has been adopted by packet switching vendors as it provides increased flexibility for the control and management of a packet switched domain. OpenFlow technology provides flow based switching that is controlled by a network management control application running in an OpenFlow controller. In this thesis work we investigate how an OpenFlow Controller communicates with a legacy network via the OSPF routing protocol, how the size of the OpenFlow network effects the resources (memory and CPU) of a legacy router to whom the controller communicates. Also we examine bandwidth utilization of the link (between the OpenFlow network and legacy router). The main goal of this thesis is to find methods to reduce the consumption of resources of a legacy router. This study shows that the size of OpenFlow network directly affects the usage of the link's bandwidth, and the memory and CPU usage of a legacy router. Aggregated information from the OpenFlow controller which is sent towards the legacy router can reduce the utilization of these resources. Finally we proposed several algorithms and design models that can be implemented for route aggregation in Software-defined Networks. Implementation of the solutions suggested in this thesis will allow automatic route aggregation in SDN. ISPs deploying SDN architecture could benefit from the proposed design models and route aggregation solution.

Sammanfattning

Software-definierade nätverk (SDN) är en framväxande trend i kommunikationsnät som underlättar frikoppling kontroll och uppgifter plan flerskiktade switchar. Ett logiskt centraliserad styrenhet på en server konfigurerar vidarebefordran tabeller (flödestabeller) av växlar för att dirigera de olika dataflöden. För att genomföra SDN har OpenFlow teknik har antagits av paketförmedlande leverantörer eftersom det ger ökad flexibilitet för kontroll och förvaltning av en påslagen paket domän. OpenFlow teknik ger flöde baserad omkoppling som styrs av ett nätverk ledningens kontroll som körs i en OpenFlow controller. I detta examensarbete undersöker vi hur en OpenFlow Controller kommunicerar med ett äldre nätverk via OSPF routing protokoll, hur storleken på OpenFlow nätverkseffekter de resurser (minne och CPU) av en äldre router till vilken styrenheten kommunicerar. Också vi undersöker bandbreddsutnyttjandet av sambandet (mellan OpenFlow nätverket och äldre router). Det huvudsakliga målet med detta examensarbete är att hitta metoder för att minska konsumtionen av resurser från en äldre router. Denna studie visar att storleken på OpenFlow nätverk direkt påverkar användningen av länkens bandbredd och minne och CPU-användning av en äldre router. Samlad information från OpenFlow styrenhet som sändes mot äldre router kan minska utnyttjandet av dessa resurser. Slutligen föreslog vi flera algoritmer och modeller konstruktion som kan genomföras för route aggregation i Software Defined-nätverk. Genomförandet av de lösningar som föreslås i denna avhandling kommer att möjliggöra automatisk route aggregation i SDN. Internetleverantörer distribuerar SDN arkitektur kunde dra nytta av den föreslagna utformningen modeller och route aggregation lösning.

Acknowledgements

I would like to acknowldge my adviser's help and guidlines at every step in completing my thesis work. He is really a very co-operative and kind person.

Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

ABR	Area Border Router
CPU	Central Processing Unit
CAPEX	Capital Expenditure
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISP	Internet Service Provider
LSA	Link State Advertisement
MAC	Medium Access Control
NOS	Network Operating System
OFS	OpenFlow Switch
OPEX	Operational Expenditure
ONF	Open Networking Foundation
OFLOPS	Open framework for OpenFlow switch evaluation
OSPF	Open Shortest Path First
QoS	Quality of Service
SPARC	Split architecture for carrier grade networks
SSL	Socket Secure Layer
TE	Traffic Engineering
TCAM	Ternary Content Addressable Memory

TCP	Transmission Control Protocol
VLAN	Virtual Local Area Network

Chapter 1

Introduction

The communication networks industry has adopted new designs for network architectures in order to fulfill the requirements of their users. However, these network architectures have become very complex and inflexible due to the increasing number of protocols, making it impossible for network operators, vendors, and researchers to innovate the communication networks to meet customer's requirements [1]. Modern mobile devices, server virtualization, and cloud services are driving the networking industry to rethink and redesign the existing networking architectures. Moreover, data flow patterns are changing and users with different types of devices and different applications are accessing various databases and servers. The traffic volume pressure on access networks is increasing rapidly due to the rapid grow in use of mobile devices, such as smart phones, notebooks, and tablets. The huge growth in the amount data traffic that is passing over the network requires parallel processing in order to satisfy the various customers. Some of the limitations of the existing networking architecture are high network complexity, inconsistent network policies, inability to easily scale the network, and vendor dependence. These limitations have lead the networking industry to the software-defined networking (SDN) architecture [2].

1.1 Software-defined Networking and Legacy Network Architectures

SDN is a modern trend in communication networks that facilitates decoupling the control and data plane of multilayer switches. A centralized controller hosted on a server, configures the forwarding tables (flow tables) of switches in order to route the various data flows. These switches realize the routes calculated by the controller by forwarding the packets according to the forwarding table entries that have been instantiated. The separation of management and the forwarding function has several advantages, specifically: flexibility, high efficiency, cost reduction, and ease of control. The OpenFlow protocol is widely supported by the communication networks industry. OpenFlow can be used to implement SDN. SDN is expected to have a strong influence on the future of the communication network industry. Figure 1.1 shows the difference between the existing legacy network architecture and a SDN architecture. In the legacy network architecture the control plane and the data plane are implemented in the same box. While in the SDN architecture the network control plane is decoupled from forwarding (i.e., data plane). The network control plane is programmable in the SDN architecture [2]. The protocol that couples the control and data plane is called the OpenFlow protocol. The control plane consist of a network OS (NOS) that realizes the logical view of the entire network and controls applications, written by programmers that manipulate the logical map of the network [2].

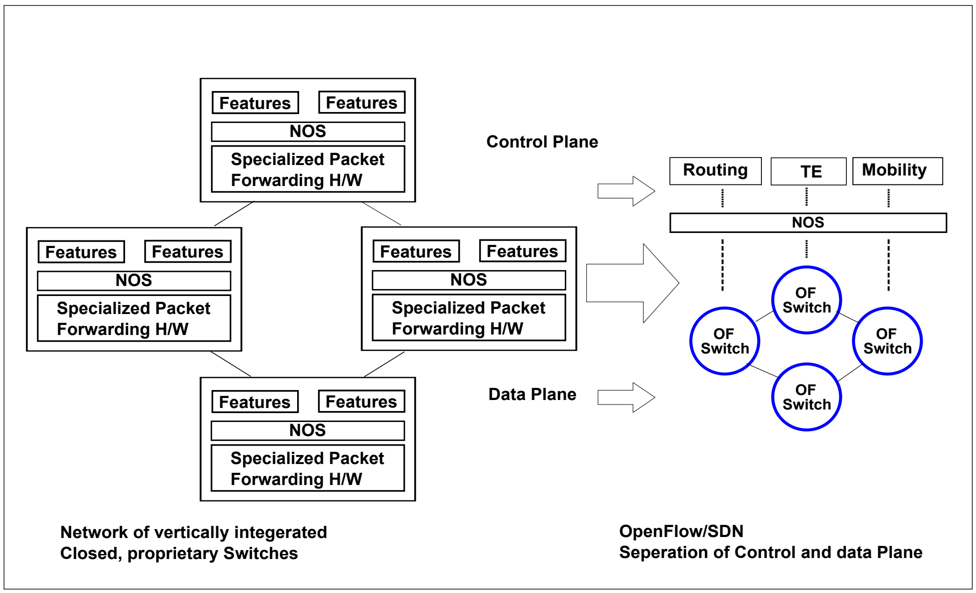
Features
Features
NOS
Specialized Packet Forwarding H/W
Control Plane
Routing
TE
Mobility
NOS
OF Switch
Data Plane
Network of vertically integrated Closed, proprietary Switches
OpenFlow/SDN Seperation of Control and data Plane

Figure 1.1: Traditional switches verses OpenFlow/SDN switches

SDN enables innovation by realizing a network operating system and allows network virtualization. SDN provides flexibility to network application developers who can now manipulate the actual network graph, without to worry about the complexity of the actual network topology. As a result a developer can manipulate the network graph by implementing different piece of codes, thus making it easier to perform experiments. The separation of data and control planes by an agnostic interface enables network operators to become independent of specific vendors of devices, hence they have a choice and can even select different control and data plane vendors. The resulting SDN is more flexible than today's network architecture which is complex and mostly dependent upon specific vendors. The SDN control plane is programmable, hence a researcher can experiment with his or her own ideas rather than being restricted to what a vendor has implemented in their router. A network can be logically divided into a research network and a production network, thus a researcher can play with the research network *without* effecting the production network[2].

Some of the world's largest network providers, including Deutsche Telekom, Facebook, Google, Microsoft, Verizon, and Yahoo!, have created the Open Networking Foundation (ONF), to standardize and promote the SDN/OpenFlow architecture. Over 40 companies and 15 vendors are now members of the Open Networking Foundation [2].

OpenFlow networks consist of two main elements: OpenFlow switches and one or more controller as shown in Figure 1.2. There are two main functions involved in routing: fast packet forwarding (along the data path) and high level routing decisions (which utilize the control path). In a traditional router these two function reside in the same device, but in the SDN architecture these functions can be located in different locations. The fast packet forwarding still resides in the network node, but the control functions reside in a separate controller. This controller is generally running in a networked attached server. As stated earlier the protocol connecting the control and data planes that is the OpenFlow protocol. OpenFlow technology will be discussed in more details in Chapter 2.

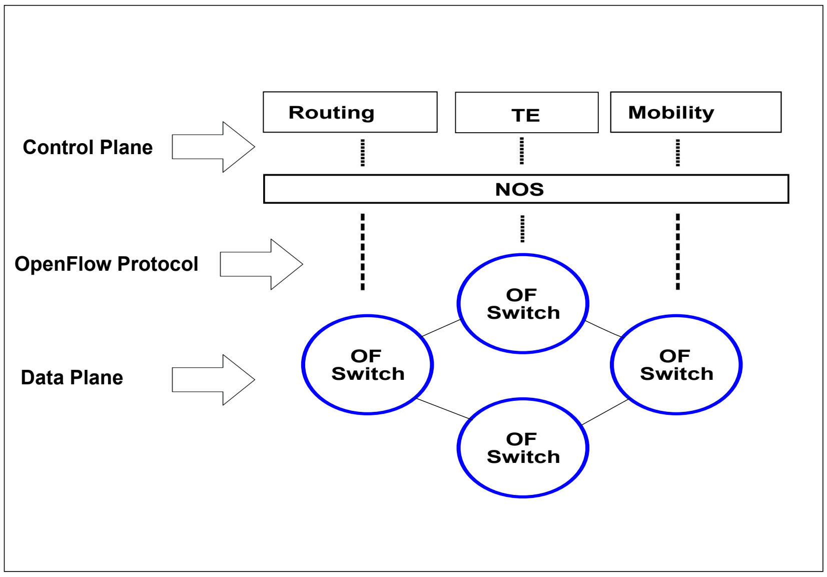

Figure 1.2: OpenFlow/SDN

1.2 This Thesis Project

The main object of this thesis project is to analyze the presentation of an OpenFlow network by an OpenFlow controller to a legacy router (in this thesis this is a Juniper M7i router). We have set of OpenFlow switches that are connected and we want to tell an existing legacy router about them, we do this by having a network application (speaking OSPF) running on an OpenFlow Controller which tells the router that the collection of OpenFlow switches is actually a single router, with many (up to say 10K) external interfaces. What are the limits? How many external interfaces are allowed in theory by the OSPF protocol? What are the limits in practice, how does a Juniper router react when you try telling it that an other router has 10000 interfaces? After finding the practical and theoretical limits we proposed solutions, in which the controller sends summarized information to the legacy router. The implementation of these proposed solutions will reduce the usage of link bandwidth and also memory and CPU usage on the legacy router. The OpenFlow controller is connected to a large number of OpenFlow Switches as shown in Figure 1.3. The OpenFlow switches are connected to customer networks. As noted earlier the OpenFlow controller communicates with the legacy network via OSPF.

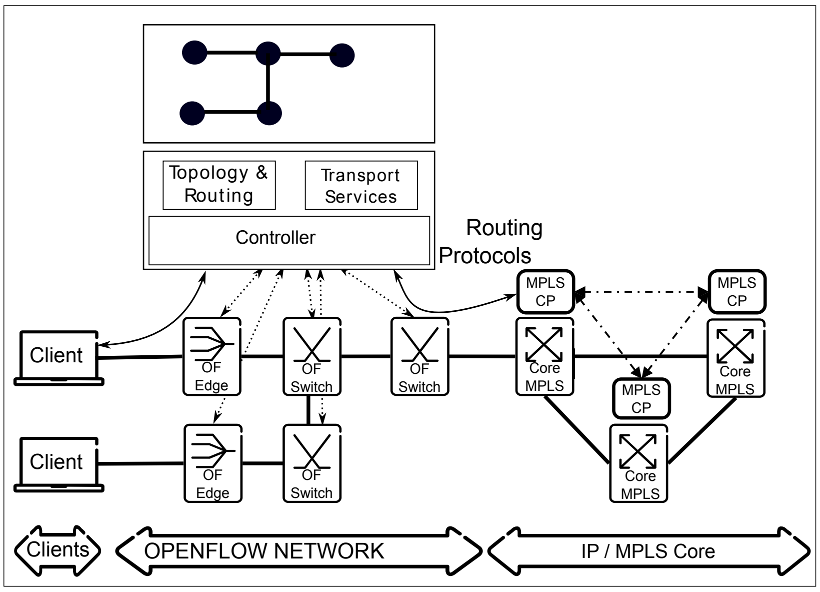

Figure 1.3: OpenFlow Network

1.3 Related Work

Manuel Palacin Mateo wrote a related master thesis entitled "OpenFlow Switching Performance"[3]. He analyzed the OpenFlow switching technology and deployed a test-bed to compare OpenFlow performance with layer-2 switch technology and layer-3 IP routing technology. In his conclusion he states: "OpenFlow switching technology is a serious alternative to software Ethernet Switching or IP Routing because it does the same layer-2 and layer-3 functions with a high performance and scalability. OpenFlow does not just do layer-2 and layer-3 forwarding, but also can do port forwarding and layer-4 forwarding, so we can consider it more flexible and configurable. This added with VLAN capabilities do it a highly recommended switching technology for isolate flows and network communications"[3].

Acreo AB is working on Software-defined Networking (SDN), primarily within the framework of the EU project SPARC (Split architecture for carrier-grade networks). In the SPARC project the Acreo AB has focused on different areas, namely ISP access and aggregation networks [4]. This thesis project is also a small part of this larger project.

Xin, Liu, Yaoqing, Wang and Zhang in a conference paper entitled "On the aggregatability of router forwarding tables" presented a deep analysis of FIB (Forwarding Information Base) aggregation and they have suggested an algorithm that can reduce of FIB by up to 70 percent. No software and hardware changes are required in the existing ISP setup. They state that their solution is a short term solution and they are expecting long-term solution by the research community. Moreover, their FIB aggregation can co-exist with a long-term solution to reduce the ISP's operational cost[5]. This paper gave us an idea how to store customers networks also it helps us in designing algorithm for automatic aggregation.

In the article entitled "Scalability Aspects of Centralized Control of MPLS Access-Aggregation Network" Jocha, Kern, and Yedavalli looked for possible scalability limitations when applying an OpenFlow-based centralized control solution to the access-aggregation segments of service provider networks. They proposed a numerical model to understand and discover possible scalability constraints. With this model they estimated scale and performance numbers, which need to be matched by a centralized controller, considering various access-aggregation network sizes [6]. From this article we learn how the size of aggregation network effects the performance of controller in OpenFlow networks environment.

1.4 Motivation

OpenFlow is an open standard that enables researchers to run experimental protocols in the campus and other networks. Support for OpenFlow has been added to many commercial Ethernet switches, wireless access points, and routers. As a result a researcher can perform experiments without knowledge of these network devices. OpenFlow enabled switches are available in the market from different vendors[7], for example HP and IBM. In our studies we are concerned with how the OpenFlow technology meets the requirements of households and business cutomers in an operator's network. Operators connect their customers via the operator's access network to the operator's legacy core network. We will be specifically concerned with how the OpenFlow controller represents the customer's networks to the legacy network, what are the effects of the OpenFlow network's size on the legacy network resources (specifically the router(s)). We expect that the large numbers of routing entries that a large OpenFlow network could potentially generate will add significant burdens to the legacy router(s). As a result of this study we will propose some solutions that can be implemented in an OpenFlow network in order to aggregate the customers' networks in order to reduce the amount of routing table entries that the legacy router(s) must deal with.

1.5 Methodology

This master's thesis consist of three logical parts. The first part describes SDN, OpenFlow technology, and the routing protocol that has been used. The second part concerns the design one or more potential solutions and describes an experimental setup and the experiments that will be used to evaluate this proposed solution. The third part presents the results of these experiments and discuss which solutions are most appropriate to efficiently represent the customers' networks in the legacy router(s).

This thesis consists of five chapters: the first chapter gives introduction that briefly describes the SDN/OpenFlow architecture. The second chapter provides further background about OpenFlow technology and the routing protocol that is used in our study. The third chapter describes the experimental setup, experiments, and descriptions of our findings. The fourth chapter proposed a solution, including several models and algorithms. The final chapter outlines our conclusions, suggests future work, and gives some required reflections.

Chapter 2

Technology

This chapter describes the technology that we used in our study. Section 2.1 briefly explains OpenFlow technology and its networks elements. Section 2.2 describes the routing protocol used between the OpenFlow Controller and legacy router, this section describes the OSPF protocol and presents some OSPF network design issues.

2.1 OpenFlow Technology

In today's communication networks, OpenFlow technology is being used as a control framework to enable a programmer to explore new networking protocols that could be used to better satisfy a user's requirements. This technology decouples the data and control plane, which are coupled in many legacy networking devices (e.g., switches, access points, and routers). To implement SDN, OpenFlow technology has been widely adopted by packet switching vendors as it provides increased flexibility for the control and management of a packet switched domain. OpenFlow technology provides flow based switching that is controlled by a network management application running in an OpenFlow controller[8].

OpenFlow technology is realized in a set of network elements (consisting of both hardware and software). These elements will be discussed in detail in the following sections. This discussion is based on an OpenFlow white paper[7] and the OpenFlow specification[7].

2.1.1 OpenFlow Switch

There are two types of OpenFlow switches in the market. The first type is a hardware based commercial switches whose flow table is constructed using a ternary content addressable memory (TCAM). Such a switch can forward packets at line rate to implement switching, routing, QoS, and other functions. The second type of OpenFlow switch is software based. Such a switch typically uses the UNIX/Linux system to implement the OpenFlow switching functions. In an OpenFlow switch the control of the OpenFlow table is done by the OpenFlow controller. The controller is programmable and hence the solution is quite flexible. Flows are defined broadly in OpenFlow switching, for example a flow can be a specific TCP connection, all packets from a particular MAC address, or all packets destined to a particular IP address, switch port, and having same VLAN tag. The main building blocks of an OpenFlow switch are a Flow Table, a secure channel to the controller, and the OpenFlow Protocol. Each of these will be described in further detail below.

2.1.1.1 Flow Table

Each flow table consists of flow entries and actions associated with each flow. The basic actions related to flows are:

- If the packet matches the flow entry then it is forwarded to a specific port associated with this specific flow.
- A packet can be encapsulated and forwarded towards the OpenFlow controller.
- The switch can drop the packet.

An example of an entry in a flow table is shown in Table 2.1. In this tables the field "Instructions" represent the action or actions that should be taken if the match fields match.

Table 2.1: Fields

Match Fields	Counters	Instructions

2.1.1.2 OpenFlow Secure Channel

An OpenFlow Secure Channel realizes an interface between the OpenFlow switch and the controller. Through this secure channel the controller configures and manages the switch, receives and sends packets, and sends and receive events to and from the switch. The secure channel messages utilize the OpenFlow Protocol format and are encrypted using Secure Sockets Layer (SSL).

2.1.1.3 OpenFlow Protocol

The OpenFlow protocol is a communication protocol between OpenFlow devices. It supports three important messages types: **(1)Controller-to-Switch** messages are initiated by the controller and used to directly manage or inspect the state of the switch; **(2)Asynchronous** messages are initiated by the switch and used to update the controller of network events and changes to the switch state; and **(3)Symmetric** messages are initiated by either the switch or the controller and sent without solicitation.

2.1.2 OpenFlow Network Controller

An OpenFlow network controller is responsible for adding and removing flow entries from the OpenFlow flow table in the appropriate OpenFlow devices. There are two types of controllers:

Static A static controller can be a device that can adds and removes flows statically.

Dynamic A dynamic controller dynamically manipulates the flow entries according to some configuration[7].

There are different types of controllers available in the market, for example: NOX and ONIX. In our studies we are using a NOX controller, as it can operate in both proactive and reactive modes. In proactive mode the controller first computes all the forwarding data, then forward this configuration to the switches; while in proactive mode, the first packet of each flow is sent to the controller which computes a new set of flow table entries, updates the relevant flow tables, and forwards the packet back to the switch that set it (for processing via the new flow table entry). The NOX controller provides a complete view of the network topology, hence it knows the location of all of the hosts. In larger OpenFlow networks more than one NOX controller can be used and they can work parallel. Whenever a new flow comes to the network, a notification is sent to one of the NOX controllers. According to Tavakoli, et al. currently deployed NOX controllers can handle at least 30K new flow installs per second while maintaining a sub-10ms flow install time[9].

2.2 Routing protocol

The OpenFlow controller can communicate with legacy routers by using different routing protocol. In our study we will use the OSPF protocols. The OpenFlow controller is responsible for informing the routers that it is link to about the networks behind it, just as if this controller was a traditional OSPF router. Note all information regarding OSPF are refrenced from the article entitled "Internetwork Design Guide – Designing Large-Scale IP Internetworks" [10].

2.2.1 OSPF

OSPF is a wel-known link state protocol. It was developed by the OSPF working group of the Internet Engineering Task Force. OSPF enable routers to advertise and learn routes from neighboring routers. The OSPF protocol sends link state advertisements(LSAs) that contains information about links. As OSPF is a link state protocol each OSPF router keeps track of links and uses these to compute route. It should be noted that OSPF and supports classless IP addresses. From the perspective of OSPF a network is divided into areas, so that OSPF can manage the network in a hierarchical structure. Route information is summarized based upon OSPF areas. A designated router and backup designated routers are elected through a OSPF process to reduce the frequency of LSAs.

In OSPF the best path is that path whose cost to the destination is lowest according to a selected metric. Every interfaces(with each interface connected to a link) of the router is assigned a cost. The total cost of a route is the sum of costs of all the links between the source and destination. For a successful OSPF implementation we should do the following activities carefully: define areas and make address assignments. If OSPF areas and address assignment are planned carefully and implemented properly, then the performance of OSPF in increased. This can make a large difference in the performance for a large OSPF domain. Next we will discuss OSPF network design, OSPF addressing and route summarization, OSPF convergence, and OSPF network scalability.

2.2.2 OSPF network design

The most important activity in designing an OSPF network is to decide which routers are included in backbone area (area 0) and which are to be included in different OSPF areas. When designing an OSPF topology we should keep in mind the following guidelines described in the following paragraphs.

2.2.2.1 Number of routers in an area

OSPF uses the Dijkstra algorithm for calculating the shortest path. This algorithm is CPU intensive and its calculation complexity depends on the number on nodes. For n number of nodes the computation complexity is of the order O(n log n). As the number nodes increase the algorithm is becomes more CPU hungry which can create performance problems. So it is recommended that an OSPF area should not have more than 50 routers. By limiting number of routers per area the computation time is more stable than where an area can have a large number of routers.

2.2.2.2 Number of neighbors for any one router

As OSPF floods all the link states changes to all the routers in an area, when there is any change in the link-state a router with a large number of neighbors has to do a lot. So it is better that a router should have maximum 60 neighbors.

2.2.2.3 Number of areas supported by any one router

A router runs Dijkstra algorithm for each area in which it resides, as this algorithm is CPU intensive, then to maximize stability a router should not be in more than three areas. Every area border router (ABR) is in at least two areas, area 0 and the area for with it is ABR.

2.2.2.4 Backbone area

The backbone area (area 0) is very important in an OSPF network domain. Keeping the size of backbone area small increases the stability and redundancy of the network topology. In the backbone area, every router is directly connected to the other backbone routers, so no single link failure can affect the network topology. OSPF includes the concept of virtual links. A virtual link creates a path between two ABRs that are not directly connected. A virtual link can be used to heal a partitioned backbone. However, it is a bad idea to design an OSPF network to require the use of virtual links. The stability of a virtual link is determined by the stability of the underlying areas.

2.2.2.5 Areas Consideration

While designing areas, we should keep in mind that individual areas are contiguous. Areas should contain a set of networks (and their corresponding subnet addresses) that can be easily summarized. To minimize the chance of a disconnection from the backbone area, an area can have more than one ABRs. While creating a large scale OSPF network the definition of areas and the assignment of resources to areas should be done very carefully, it ensures the flexibility, reliability and performance of the network. Wisely designed and small areas will reduce the effect of route flapping caused by unstable links.

2.2.2.6 OSPF addressing and Route summarization

To make an OSPF domain scalable and stable the address assignment to the OSPF areas should be done carefully, as a carefully designed addressing scheme facilitates route summarization. We should keep this addressing scheme as simple as possible. The simplicity in addressing saves time when operating and troubleshooting the network, it also simplifies the route summarization that must be performed by the ABRs. In the OSPF domain route summarization is done between each area and the backbone area. Summarization is configured manually in OSPF. When planning your OSPF internetwork, you must consider the following issues:(1) be careful that your network addressing scheme is configured so that the range of subnets assigned within an area is contiguous and create an address space that will permit you to split areas easily as your network grows; and (2) you should plan for future growth in number of routers in your OSPF environment. These new routers should be inserted appropriately as area, backbone, or border routers. However the addition will modify the network's topology, that effects the performance of OSPF route computation. It is recommended that each area should have a separate address structure. This approach offers the following benefits: configuration of routers is easy, address assignment is easy to remember, and it is simple to do route summarization.

2.2.3 OSPF convergence

Fast convergence is an attractive features of OSPF which enables it to quickly adapt to topology changes. There are two components of OSPF convergence: (1) detection of topology changes and (2) the computation of new routes. OSPF uses two mechanisms to detect topology changes, interface status changes (link failure), and when OSPF fails to receive hello packet from its neighboring router within a timing window called a dead timer. When the timer is expired, the router assumes that this neighbor is down. The dead timer is configured manually by using a user interface configuration command. The default value of the dead timer is four times the value of the Hello interval, this results in a dead timer default of 40 seconds for broadcast networks and 2 minutes for non broadcast networks. Once a failure has been detected, the router that detected the failure sends a link-state packet with the changed information to all routers in the area. All the routers run the Dijkstra algorithm to compute new routes. The time required to run the algorithm depends on a combination of the size of the OSPF database and number of routers in the area.

2.2.4 OSPF Network Scalability

As we have outlined in the preceding discussions concerning network topology and route summarization, adopting a hierarchical addressing environment and a structured address assignment will be the most important factors in determining the scalability of our OpenFlow Network in the OSPF domain. Network scalability is affected by two main considerations: **Operationally:** OSPF areas should be designed so that they can accommodate the network growth, hence address space should be reserved for new areas, and **Technically:** Scaling is determined by the utilization of three resources: memory, CPU, and link bandwidth. As each OSPF router in a given area has the same state information (since the LSAs were flooded), each of them performs the same computation, hence the memory and CPU resources are the same in each router in this OSPF area.

2.2.4.1 Memory

As the number of routers increases within the area, an OSPF router requires more memory to store all of the link states. Additionally, it must do so for all of the areas that it is in. The router can also store summaries and information about external networks. The preceding guidelines about network design and route summarization can useful for substantially reducing the amount of memory used.

2.2.4.2 CPU

The Dijkstra algorithm is CPU hungry computation with a computation complexity of $O(|E|+|V|\log|V|)$ (where $|V|$ is the number of node (vertices) and $|E|$ is the number of edges). Whenever a link-state change occurs every router in the area computes its routes using the Dijkstra algorithm. Keeping areas small and using summarization dramatically reduces CPU use and creates a more stable OSPF domain.

2.2.4.3 Bandwidth

OSPF sends partial updates when a link-state change occurs. These updates are flooded to all routers in the area. In a quiet network (i.e. one with few changes), OSPF is a quiet protocol. By following the guidelines mentioned in above discussion, we can design a reliable OSPF network with substantial topology changes, minimizes the amount of bandwidth used.

Chapter 3

Experimental Setup, Experiments, and Results

This chapter describes the resources used, experimental setup, experiments, and findings. Section 3.1 briefly describes the equipments used in this project. Section 3.2 describes the scenarios that were the basis for the experiments. Section 3.3 describes the experiments and our findings.

3.1 What Equipment was used

In our experiments we used a PC, Agilent Tester, and Juniper router. The specification of each device is given in subsections below.

3.1.1 PC

We used a PC with the attributed shown shown in Table 3.1. This PC has been used for router configuration and monitoring purposes.

Table 3.1: PC

CPU	Intel Core2 Duo E6750 2.66 Ghz
RAM	8 GB 1066MHz
HARD	40 GB
NIC	2 x Intel PRO/1000 PT dual port 1 Gbps PCI-Express
Operating System	Linux Ubuntu 8.10 64 bit. Kernel 2.6.27

3.1.2 Agilent N2X Traffic Generator

To generate OSPF traffic, we used an Agilent N2X router tester. This device specifications are shown in Table 3.2. Note that we will pre-compute OSPF messages for this tester to later send to the legacy router. These OSPF messages are based upon the topology and topology changes that we simulate happening in a large OpenFlow network. The OpenFlow controller that would be attached to the legacy router would send these OSPF message to the router.

Table 3.2: Agilent N2X Traffic Generator

Used modules	E7919A 1000Base-X GBIC-RJ45 and E7919A 1000Base-X
Modular chassis	In our testing we used 2 modules stated in the row above
Release Software	N2X Packets 6.4 System Release traffic analyzer software

3.1.3 Juniper Router M7i

The System Under Test (SUT) in our studies is a Juniper router whose specifications as shown in Table 3.3.

Table 3.3: Juniper Router M7i

Modular chassis	4, plus 2 additional fixed FE
Aggregate Half-Duplex Throughput	10Gbps

In the experimental setup we use an Agilent Router Tester for traffic generation as shown in Figure 3.1. The Agilent Router Tester plays the role of an OpenFLow controller and it is connected to the Juniper router. The routing protocol used is OSPF version 2.

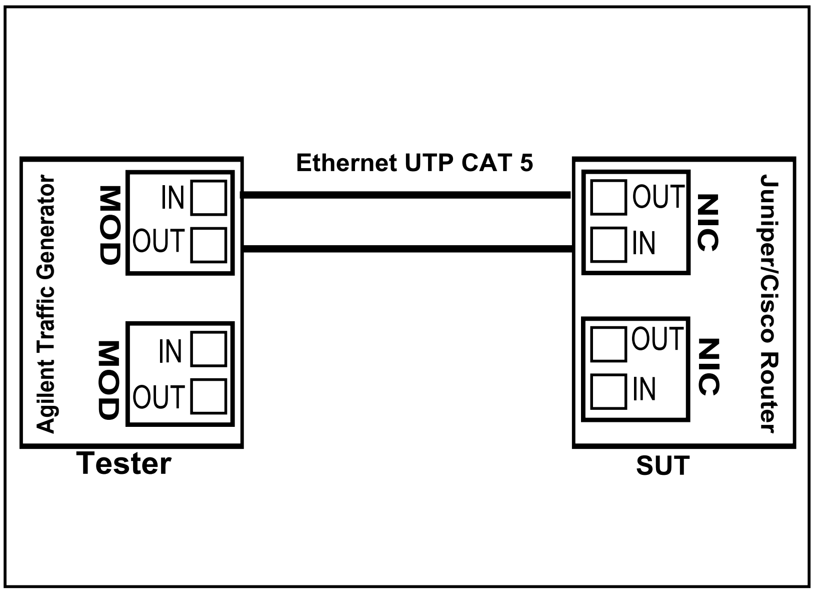

Figure 3.1: Experimental Setup

3.2 Scenarios

Before doing experiments we would like to discuss scenarios which show how an OpenFlow controller exposes the OpenFlow network to the legacy router . In our study we consider two scenarios as follows.

3.2.1 Scenario 1

In the first scenario we have two cases. In case 1 the OpenFlow controller exposes the whole OpenFlow network to the legacy router (see Figure 3.2). The routing protocol between the OpenFlow network and legacy network is OSPF. For every node in OpenFlow network, the controller sends the router an LSA using OSPF. We used Agilent Router Tester to emulate the OpenFlow controller. This tester for generates a large amount of OSPF traffic towards the legacy router. Using this tester we can simulate different routing protocols (OSPF, BGP, RIP) and we can create diffrent network topologies (grid, ring , tree). In our first case we use grid topology. Due to limitation of our tester, it can create a grid with a maximum 20 rows and 20 columns, i.e., a maximum of 20 * 20 = 400 routers in a grid. We can add multiple grids in our routing session to generate more OSPF traffic. Every router in the grid sends a Router LSA towards the legacy router (Juniper). The format of a router LSA is shown in Figure 3.4. In scenario 1 we have second case where the OpenFlow controller represents itself as a single router with a large

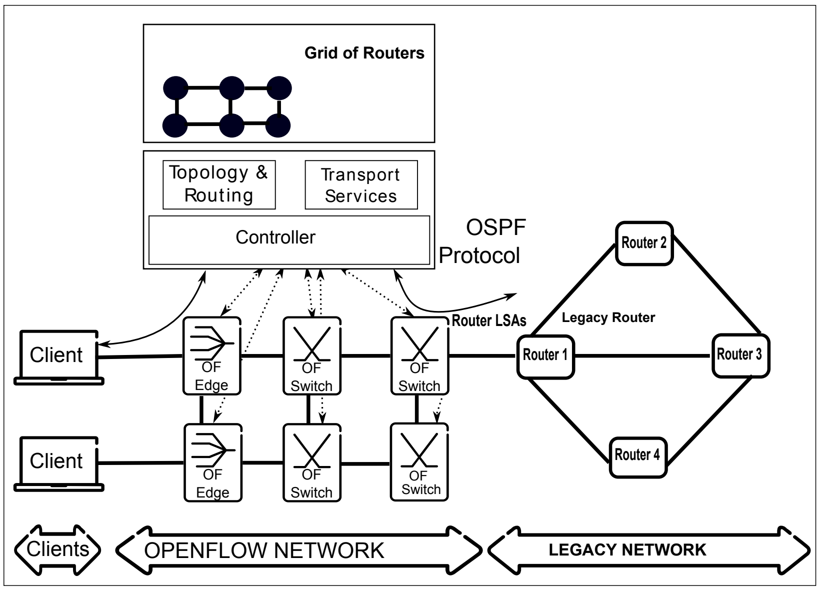

Figure 3.2: Scenario-1 Case:1 Router LSAs

number of interfaces as shown in Figure 3.3. In this case the controller will send a single router LSA with a large number of links attached. A longer explanation of this second case will be given in the section about the experiments.

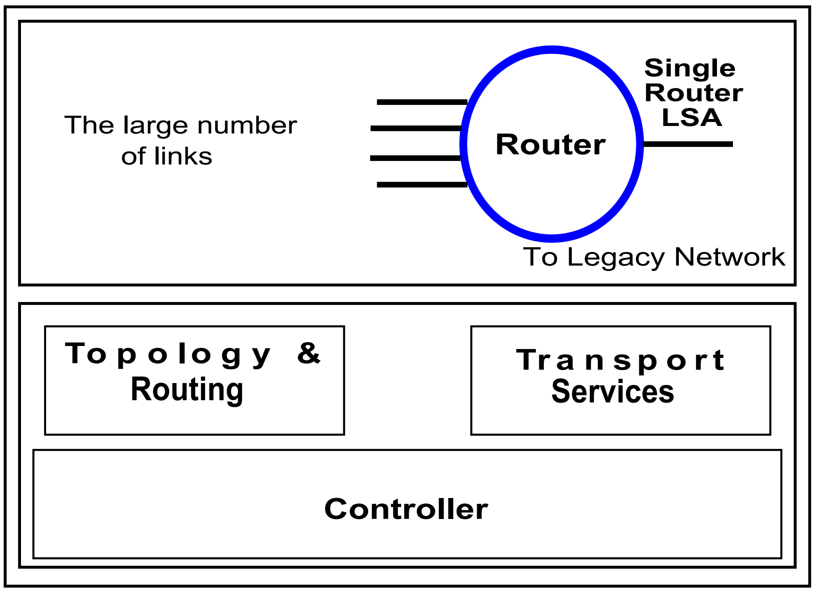

Figure 3.3: Scenario-1 Case(2) Single Router LSA

In scenario 1 in both cases the router LSA is sent to the legacy router, so it is very important to understand the format of the router LSA. In a Route Link state Advertisement (Type 1) the router announces its presence and lists the links to other routers or networks in the same area, together with the metrics to them. Type 1 LSAs are only flooded across the area only. The link-state ID of a type 1 LSA is the originating router's ID. There are other LSA types defined in OSPF (such as Type 2 - Network LSA, Type 3 - Summary LSA, Type 4 - ASBR-Summary LSA, Type 5 - External LSA, Type 6 - Group Membership LSA, and more upto Type 11). In our study we will consider only two LSA types: Type 1 - Router LSA and Type 3 - Summary LSA. Type-3 LSA will be discussed in scenario 2.

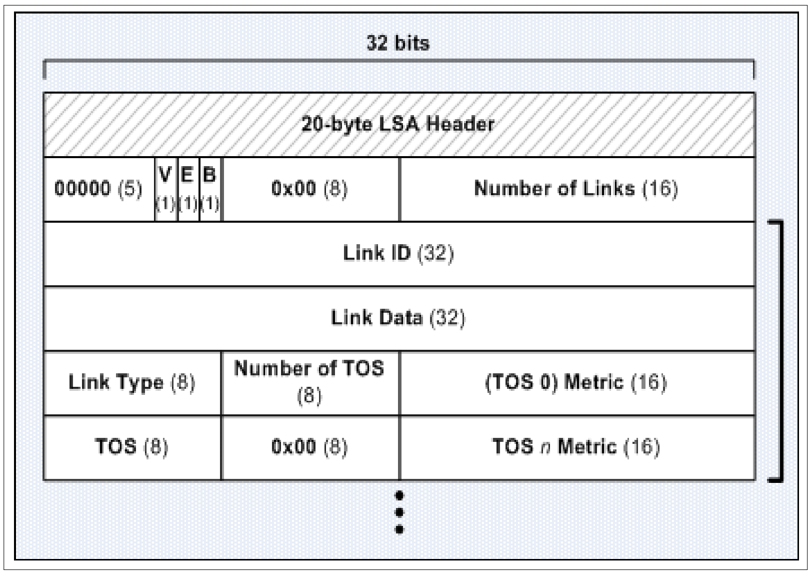

Figure 3.4: Router LSA format

In the experiments (for scenario 1) we are using router LSAs. The reason for the experiments with scenario 1 is that the size of LSA will increase as the number of links increases. In scenario 1 we examine the case where there is either multiple originating source router IDs or a single source router ID. These different router LSAs will enable us is good to analyze the memory and CPU usage of the legacy router. It should be noted that since the Router LSAs are flooded across the area only these Router LSA will not propagate into other areas.

3.2.2 Scenario 2

In the second scenario OpenFlow controller represents itself as an ABR. An ABR takes the information it has learned from one of its attached areas and summarizes this information before sending it out to the other areas it is connected to. Because an ABR sends summary LSAs this helps improve scalability as it removes detailed topology information which is unnecessary for other areas. The result is that a set of because their routing information can be summarized simply as an address prefix and a metric. The summarization process can be configured to remove a lot of detailed address prefixes and replace them with a single summary prefix, this also helps improve scalability. The link-state ID of these summary LSA will be the destination network number for type 3 LSAs. Figure 3.5 shows our second scenario. The format of summary LSA is shown in Figure 3.6

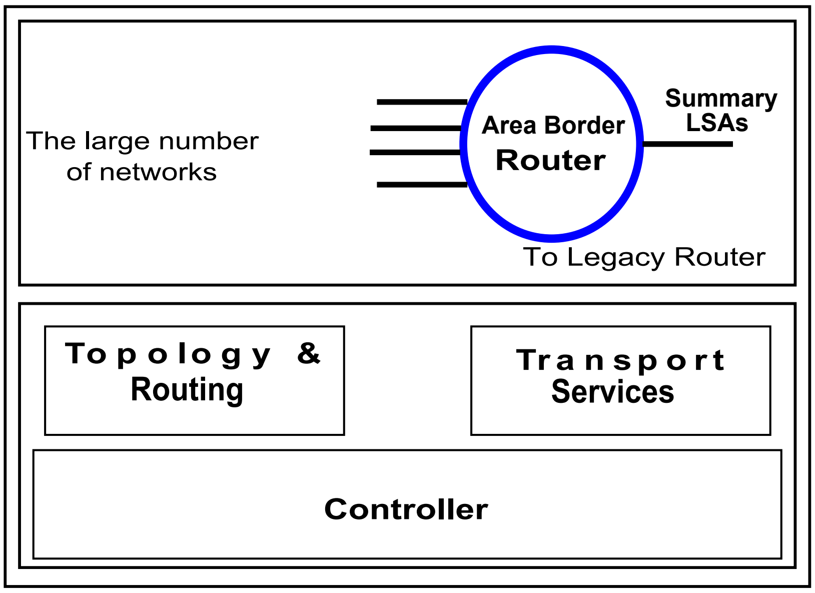

Figure 3.5: Scenario-2 Summary LSA

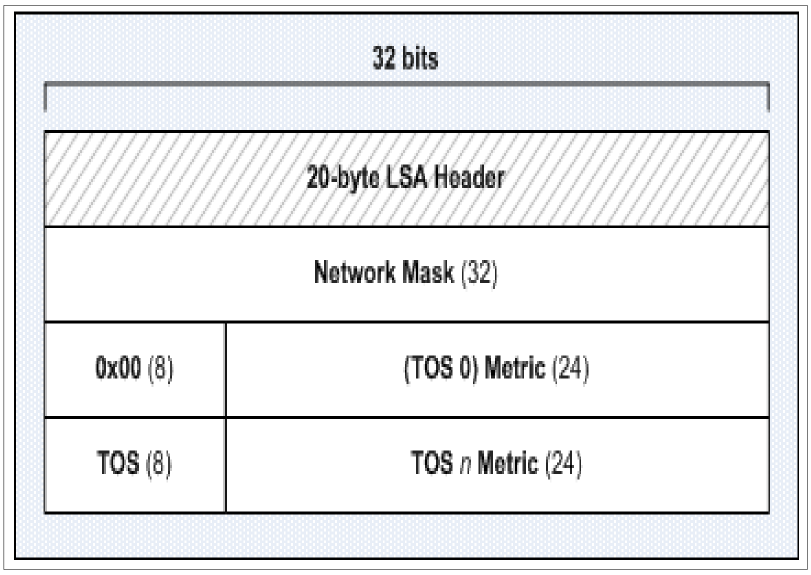

Figure 3.6: Summary LSA format

3.3 Experiments

Using the experimental setup shown in Figure 3.1 a number of experiments are performed. In all of these experiments the Agilent Router Tester acts as if it were the OpenFlow controller. The tester only does traffic generation, thus it does not perform other OpenFlow controller functions. All of these experiments are based on scenario 1 and scenario 2. In all of these experiments we will investigate usage of memory, calculated memory, and the usage of the link's bandwidth (i.e., the link between tester and Juniper router). Note we have not performed experiments for CPU utilization, only theoratical explanation of CPU utilization is presented in this study.

3.3.1 Experiments: Scenario 1 (case 1)

The experimental setup was established for our first scenario. In this case the controller represents itself as many different routers. Using Agilent router tester we can create a grid topology in an OSPF domain. As noted earlier the maximum size of each grid was limited by the tester to 400 routers, so we created a large number of grids. The OSPF database at legacy router after 400 router LSAs will know about 400 routers. We can increase the number of grids to increase the apparent number of routers. In this scenario we assume that Agilent Tester and Juniper router are in the same OSPF area, as router LSAs are only shared within the same area.

3.3.1.1 Router Memory

Before starting experiments we found that the available memory of the juniper router was 745,377,792 bytes. We added 10 grids, each grid have 400 routers. To get information regarding memory usage and details of the OSPF database detail we used Juniper's commands **show process task memory** and **show ospf database detail**. We saved the output of these commands in files for later analysis.

By analyzing the amount of memory used in the legacy router we found that memory usage increase linearly with number of routers. When we had a single grid of 400 routers, we received 400 router LSAs. Out of these 400 LSAs, 324 routers had 9 links (each router), 73 routers had 7 links and 3 routers had 5 links. After we added a new grid with an additional 400 routers then we received 800 router LSAs. The number of routers with 9 links increased to 648, with 7 links increased to 146, and with 5 links increased to 6. The number of routers with 9,7, and 5 links have been increased in the same ratio. As every OSPF LSA has a 24 byte header and every link in router LSA contain 12 bytes, then the size of a router LSA with 1 link will be **24 + 12*1 = 36 bytes**. Similarly the size of single router LSA with 7 links would be **24+12*7=108 bytes**. We have calculated the memory usage for 400, 800,.. 4000 routers, and the size of the router LSAs are shown in Table 3.5. We also notice that the measured memory is considerably 22 times more than calculated memory. The experiments show that as we increase the number of routers in the network the size of OSPF database will increase linearly and neighboring router will require more memory space to store the database. The increase in measured memory is more than the calculated memory due to the OSPF processes (shown in Table 3.4) occupy memory space. However, the explanation for the memory consumed by these processes is outside the scope of this thesis.

Table 3.4: OSPF related processes which use memory

The processes
1-patroot
2-ospf spf linkage
3-ospf lsa topo entry
4-ospf lsa topo link
5-rt tsi
6-itable8 bucket t
7-Timer auto parent re
8-rt metrics
9-ospf rt entry
10-ospf spf entry
11-ospf spf result
12-ospf rt block
13-ospf lsdb entry

The results in Table 3.5 can be represented in the form of a graph as shown in Figure 3.7. The blue squares represent the measured amount of memory. We can see that this increases as the number of router LSAs. The red squares represents the calculated memory (simply based upon the size of the LSA) that also increases linearly along with the number of router LSAs. The experiments shows memory usage at legacy router not only depends on the size of OSPF LSAs upadates but also storage structure of database.

Table 3.5: Router LSAs and measured memory usage

Router LSAs	Measured Memory(Bytes)	Calculated Memory(Bytes)
400	1141400	51044
800	2246560	101972
1200	3373684	152900
1600	4477512	203828
2000	5574256	254756
2400	6672368	305684
2800	7766604	356612
3200	8868896	407540
3600	9954376	458468
4000	10061660	509396

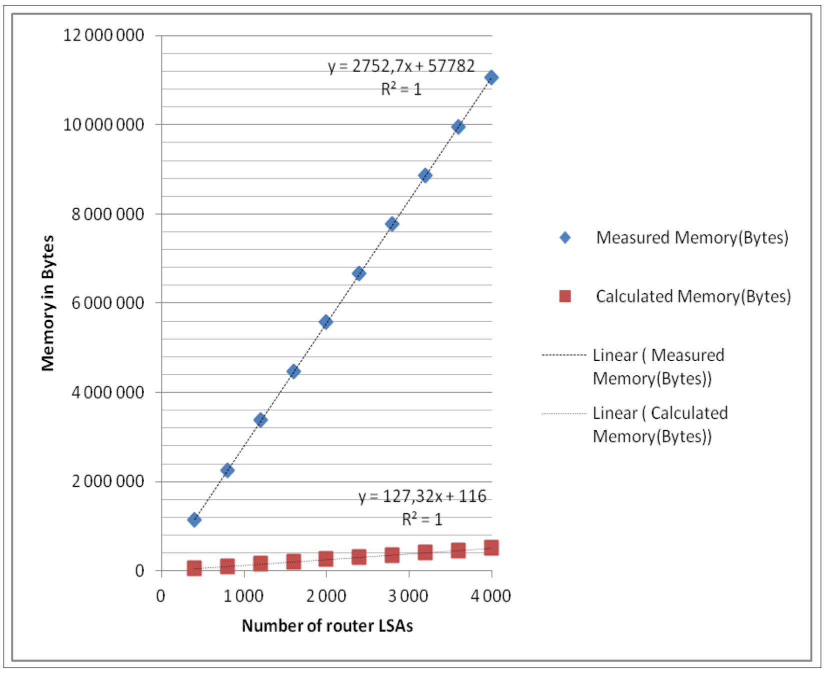

Figure 3.7: Memory usage as a function of the number of router LSAs

3.3.1.2 Link Bandwidth

The amount of link bandwidth consumed increases with number of advertisements as this information must be transferred in order to add it into the OSPF database. A OSPF LS update is sent periodically after 30 minutes (1800 seconds). For 400 Router LSAs the average refresh rate will be (1800/400) 4.5 seconds. As these

LSAs are encapsulated in an IP packet which we assume is transmitted over an Ethernet link to the router. With this information we can compute that the size of IP packet header is 20 bytes to which 18 bytes of Ethernet frame header, and 4 bytes of CRC trailer are added. The transport header's 20 bytes has to be included. The resulting size of an Ethernet frame carrying a router LSA with 9 links will be 20 + 20 + 18 + (24 + 12*9) +4 = 194 Bytes. The results are shown in Table 3.6.

Table 3.6: Router LSAs and link bandwidth usage (Byte/sec)

Router LSAs	Average refresh rate	B/W Usage	Average B/W Usage
400	4.50	43.11	37.77
800	2.25	86.22	75.55
1200	1.50	129.33	113.33
1600	1.12	172.44	151.11
2000	0.90	215.55	188.88
2400	0.75	258.66	226.66
2800	0.64	301.77	264.44
3200	0.56	344.88	302.22
3600	0.50	388.00	340.00
4000	0.45	431.11	377.77

The graphical representation of the above results of the bandwidth usage are shown in Figure 3.8. The red squares shows the maximum bandwidth usage at each step. Maximum bandwidth increases linearly as the number of router LSAs increase. The green triangles shows the average bandwidth usage by OSPF router LSAs. Given that the link data rate is more 1 Gbps or more, the 500 bytes per second of LSA traffic is insignificant for such a link. Our purpose in doing these experiments is to show that the bandwidth of the link which is used will be effected by the size of OSPF routing domain.

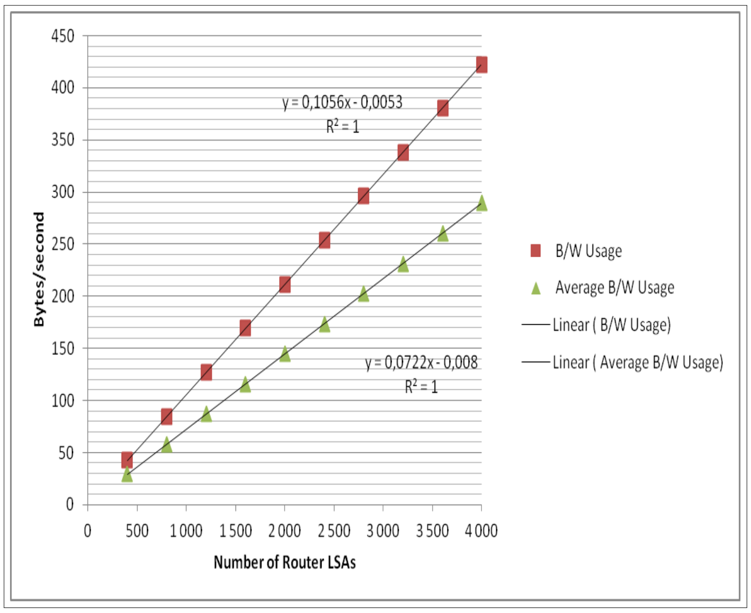

Figure 3.8: Link bandwidth usage versus number of router LSAs

3.3.2 Experiments: Scenario 1 case 2

In second case of the first scenario the OpenFlow controller is acts as a single router and presents OpenFlow network to the legacy router using OSPF . The controller acts as if it were a router with a large number of interface connected to different networks. As each link in the Router LSA is 12 bytes, thus so a router with 10 links will send send a router LSA of size **24 + (12*10) = 144 bytes**. When the size of single router LSA is larger than 1500 bytes the IP fragmentation is required as the MTU limit for IP packet over Ethernet is 1500 bytes. The maximum size of a type 1 OSPF message is 64KB. Given 24 bytes of common LSA header and 12 bytes to represent each link, a type 1 LSA can advertise a maximum of 5331 links[11]. OSPF LSAs are encapsulated in the an IP packet. If LSA size greater than the maximum size of a IP packet then fragmentation of the LSA message is required. The standard IP fragmentation procedure works fine for OSPF. However, the defragmentation procedure will add additional overhead for the legacy router. The experimental results are shown in Table 3.7. These experimental results shows that the measured memory is 2 times more than calculated memory. Note: due to limitations in Agilent tester we can simulate a single router LSA with maximum 500 links.

A graphical representation of these results is shown in Figure 3.9. The blue squares show the calculated memory increases linearly and red squares show that the measured memory also increase linearly with an increase in the number of

Table 3.7: Single Router LSA and memory usage

Number of links	Calculated memory	Measured memory
50	740	1580
100	1340	2220
150	1940	2604
200	2540	4524
250	3140	4662
300	3740	5524
350	4340	8620
400	4940	9630
450	5540	11845
500	6140	12626

links.

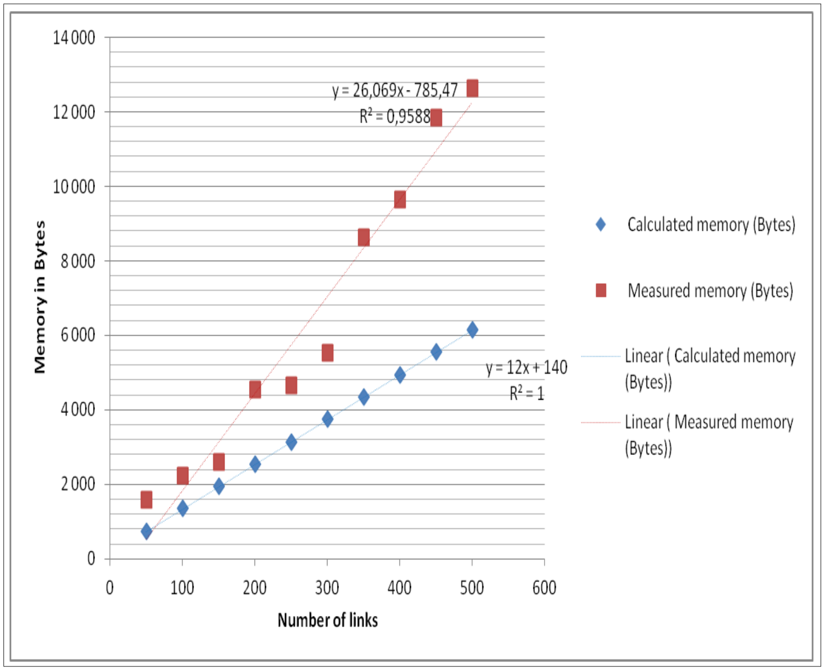

Figure 3.9: Single Router LSA versus memory usage in bytes

In this case links bandwidth usage is insignificant as a OSPF LS update is sent periodically after 30 minutes (1800 seconds). If there is no change in the network topology, an LS update will be sent after 30 minutes that does not effect the bandwidth of the link.

3.3.3 Experiments: Scenario 2

In scenario 2, the controller acts as an ABR of an OSPF area. Type-3 Network-Summary-LSAs are originated by the ABR to advertise the subnets in an area (omitting information about Type-1 and Type-2 LSAs) to neighboring routers outside the area. Type-1 and Type-2 LSAs stay within an area, when an ABR receives a Type-1 or Type-2 LSA, it generates a Type-3 LSA for the **network** learnt via the Type-1 or Type-2 LSA to other areas. Type-3 Network-Summary-LSAs are not being summarized and therefore do not (by default) contain summary routes.

3.3.3.1 Router Memory

The format of network summary LSA is shown in Figure 3.6. It contain 24 bytes of common LSA header and subnet mask of 4 bytes for each network that the ABR advertises to other OSPF areas. The experiments shows that if we have 50 subnets in the area, then the ABR will advertise 50 network summary LSAs. The size of each of these summary LSAs is 28 bytes. These results are shown in Table 3.8. We also notice that the measured memory is 14 times more than calculated memory. Note: due to limitations in Agilent tester we can simulate maximum 500 summary LSAs in a test session.

Table 3.8: Summary LSAs and memory usage

Number of links	Calculated memory	Measured memory
50	1516	24144
100	2916	31136
150	4316	70256
200	5716	86080
250	7116	117032
300	8516	121016
350	9916	125872
400	11316	143872
450	12716	151632
500	14116	186144

A graphical representation of these results is shown in Figure 3.10. In the graph red square indicates the measured memory (as the results of memory detail command), the amount of memory changes as a function of both the number of links and the various OSPF processes that are running. With an increase in number of networks or routers in an area the memory usage also increases. The blue square shows the calculated memory, this increases linearly. The graph shows that the number of network summary LSAs will directly effect the memory of legacy router.

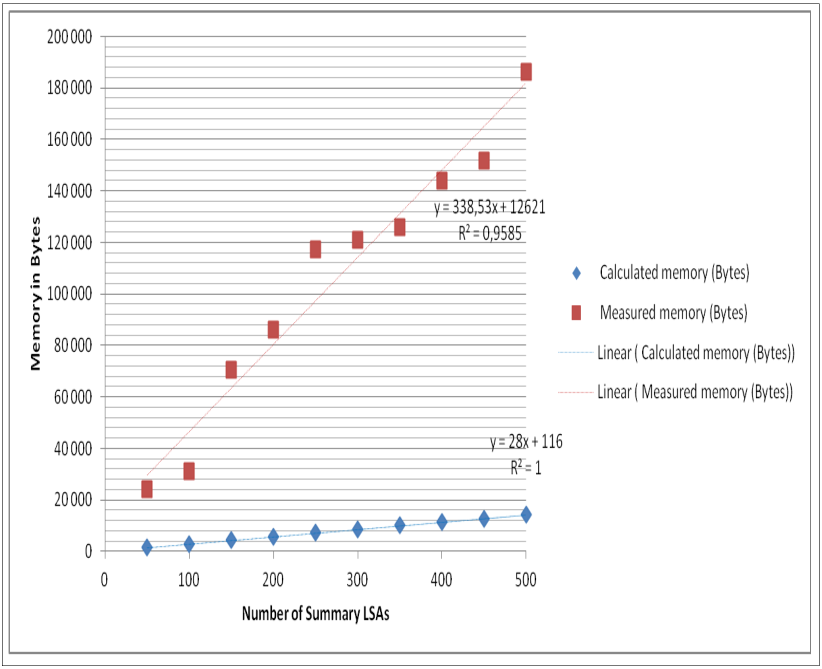

Figure 3.10: Memory usage as a function of the Number of Summary LSAs

3.3.3.2 Link Bandwidth

The usage of bandwidth of the link is also effected with the number LSA updates. As these LSAs are encapsulated in an IP packet which we assume is transmitted over an Ethernet link to the router. With this information we can compute that the size of IP packet header is 20 bytes to which 18 bytes of Ethernet frame header, and 4 bytes of CRC trailer are added. The transport header's 20 bytes are also included. The resulting size of an Ethernet frame carrying a summary LSA will be 20 + 20 + 18 + (24 + 4) + 4 = 90 bytes. The results are shown in Table 3.9.

Table 3.9: Summary LSAs and link bandwidth usage

Number of Summary LSAs	Average refresh rate	B/W Usage
50	36.0	2.5
100	18.0	5.0
150	12.0	7.5
200	9.0	10.0
250	7.2	12.5
300	6.0	15.0
350	5.1	17.5
400	4.5	20.0
450	4.0	22.5
500	3.6	25.0

However, since today's links have such high bandwidths, a few bytes/sec is insignificant for these links. However our main purpose is to show that the increase in LS updates will directly effect the bandwidth of the link. A graphical representation of these results is shown in Figure 3.11.

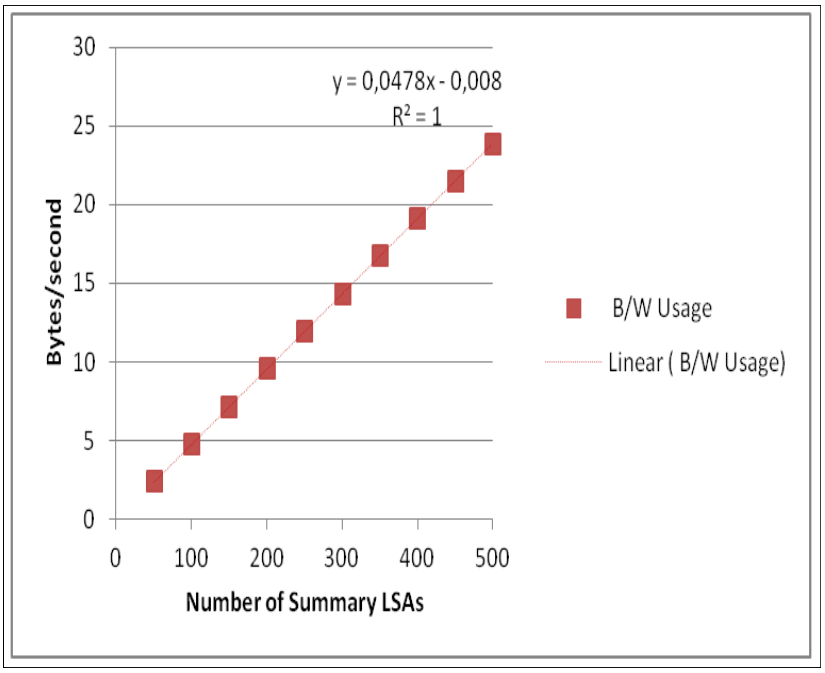

Figure 3.11: Link bandwidth usage as a function of the number of Summary LSAs

The following graph as shown in Figure 3.12 represents combined measurement results with reference to Table 3.7 and Table 3.8. It shows memory usage (measured) of Summary LSA versus Single router LSA. The analysis shows that representation of OpenFlow network towards the legacy router by Summary LSAs is worse than single router LSA by factor of 15x in terms of memory usage.

3.3.4 Router CPU

The OSPF functions that consume CPU processing power are: the Shortest Path First (SPF), the packet processing and flooding. As we know the Dijkstra algorithm is a well-known algorithm for finding a shortest path, OSPF uses this algorithm to build and calculate the shortest path to all known destinations. As we know that the size of routing information update, the frequency of updates, and the lack of good network design are coomon factors that influnce the utilization of CPU. As the size of routing domain grows, the algorithm utilizes more CPU cycle to compute the shortest paths. The Dijkstra calculations are of order **(n* log(n))** where n is the number of nodes in the OSPF domain. Also when link-states changes occur in OSPF domain, the router must recompute the costs using the Dijkstra algoirhtm. If the OSPF routing domain is divided into multiple areas, then overhead on Dijkstra algorithm is reduced which minimize the CPU

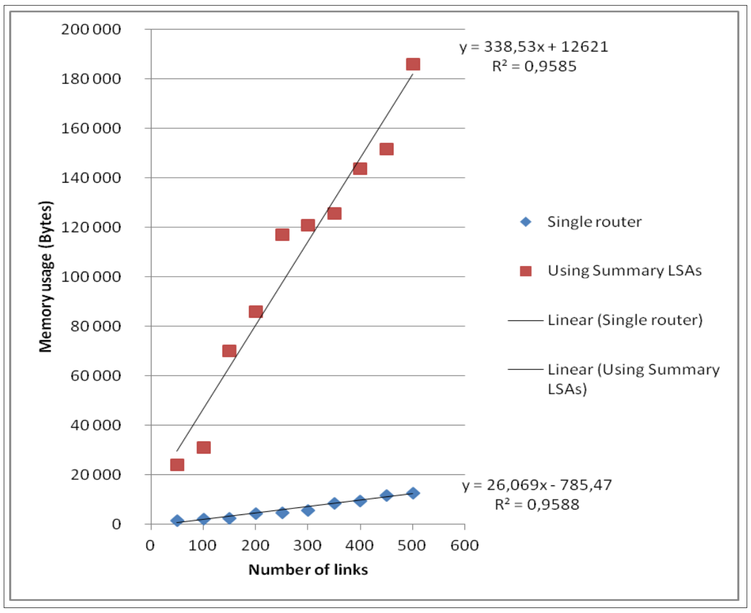

Figure 3.12: Memory usage of Summary LSA versus Single router LSA

utilization[12].

The experiments and discussions in previous sections show that as we increase the size of network domain, three resources were affected: memory, CPU, and link's bandwidth. The OpenFlow controller is responsible for communicating with a legacy neighboring router, the way in which this controller exposes its network directly affects the memory and CPU usage of neighboring router and the bandwidth used. The experiments show that when OpenFlow controller represents itself as a single router (scenario 1 case 2) with a large number of interfaces, these rsources utilization is reduced dramatically. Resource consumption can be reduced by implementing the solutions proposed in Chapter 4.

Chapter 4

Proposed Solution

This chapter suggests two propsed solutions and algorithms for the problem that we discussed in Chapter 3. Section 4.1 proposes solution 1 and section 4.2 suggests a naive solution that can be implemented in SDN environment.

4.1 Solution 1

Figure 4.1 show the proposed solution. The rectangle in the diagram shows the physical node (i.e. OpenFlow Switch (OFS)) and the black dots show instance of the virtual machine. OFS can be directly connected to the controller or be connected by a TCP connection as shown by the dotted lines in Figure 4.1

The controller has a database of all the network that exists in its domain. Also it has a map of physical the topology of the networks. The Virtual Machine Server (VM-Server) can create a virtual machine for each physical node. An intelligent program can be written to divide the map of topology in different OSPF areas, i.e., area 0, area 1, area 2, and so on. Every VM runs the OSPF routing protocol. The areas are allocated according to their physical connectivity. All areas are connected to area 0 by ABRs. As we discussed in OSPF section the ABR is responsible for sending a network summary LSA to area 0, hence all of the nodes in the area do not need to send their LSAs to area 0. Only a single LSA will be send to area 0 if route summarization is configured at ABR. The format of a network summary LSA is shown in Figure 3.6. The network summary LSA includes only prefixes and does not include any other routing information. As we know that each OSPF domain communicates with the other OSPF domains through area 0. Every area know only the network summary of other areas through area 0. If there is any change in the topology of area 1 it does not affect the other areas.

By implementing our proposed model the controller will only send summary

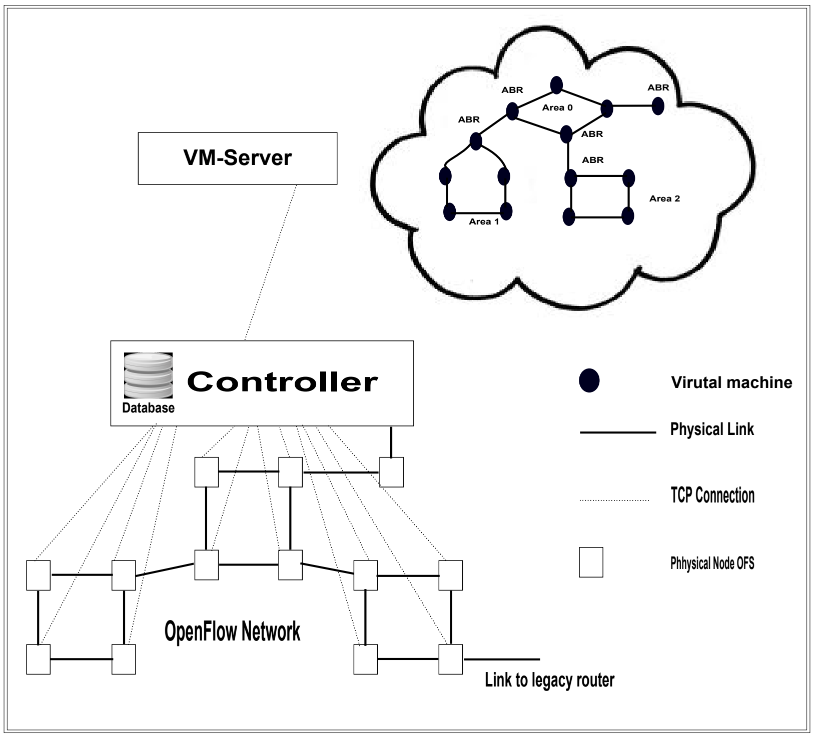

Figure 4.1: Proposed Solution Model 1

LSA to each OSPF area. This will result reduced memory usage, CPU usage and bandwidth usage of link. **Drawback:** The solution we proposed is efficient with respect to the three important resources of the legacy router (memory, CPU, and links bandwidth). However, this solution increases the overhead on the controller, as the controller has to control the VM-Server for each physical node, hence the VM-Server's cost is an overhead on the complete network.

4.2 Naive solution

We first propose a naive solution. According to this model, we introduce Topology Server (T-Server) which is responsible for storing information about all of the OpenFlow networks in the form of a binary tree, in order to support for fast and efficient IP lookup. Figure 4.2 illustrates our proposed solution which we propose that a controller should implement. Suppose the following customer' networks stored in database as shown in Table 4.1.

The T-Server maintains a binary tree as shown in Figure 4.2. In this example there are 6 networks prefixes stored at leaf nodes in Figure 4.3. The nodes with

Table 4.1: Network prefixes stored in database

Network Prefixes
192.168.0.0/19
192.168.32.0/19
192.168.64.0/19
192.168.128.0/19
192.168.160.0/19
192.168.224.0/19

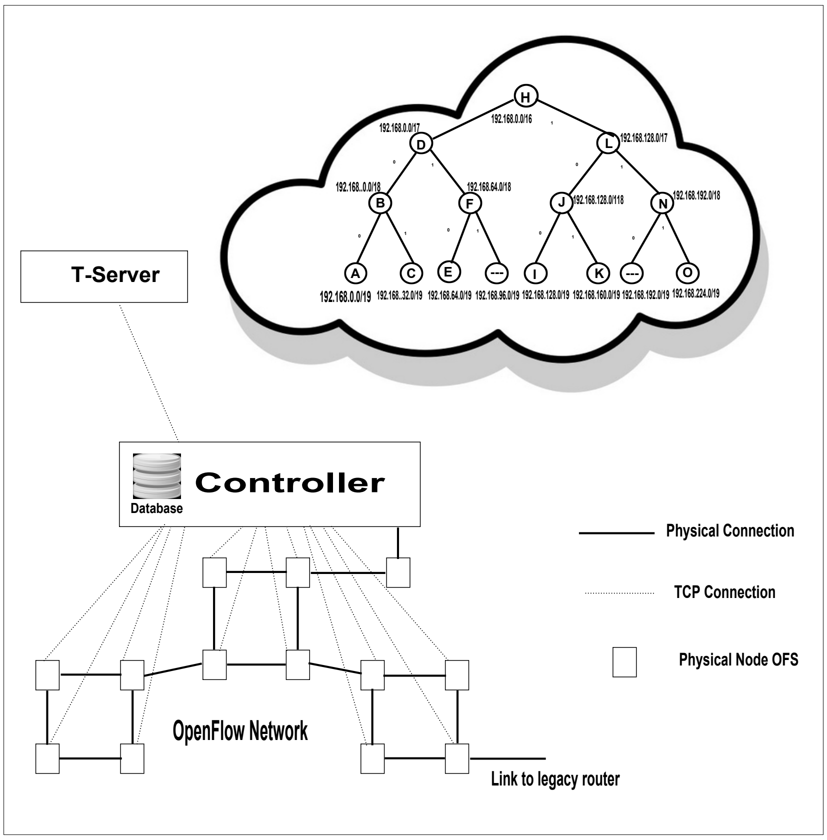

Figure 4.2: Proposed Solution Naive Model

dots indicates empty nodes. Here we will consider two cases, in first case we want to aggregate customers' networks and send this aggregated information to the legacy router. Along with aggregated routes, specific routes are also sent to legacy router.

To create a list prefixes that contains aggregated and specific routes, we

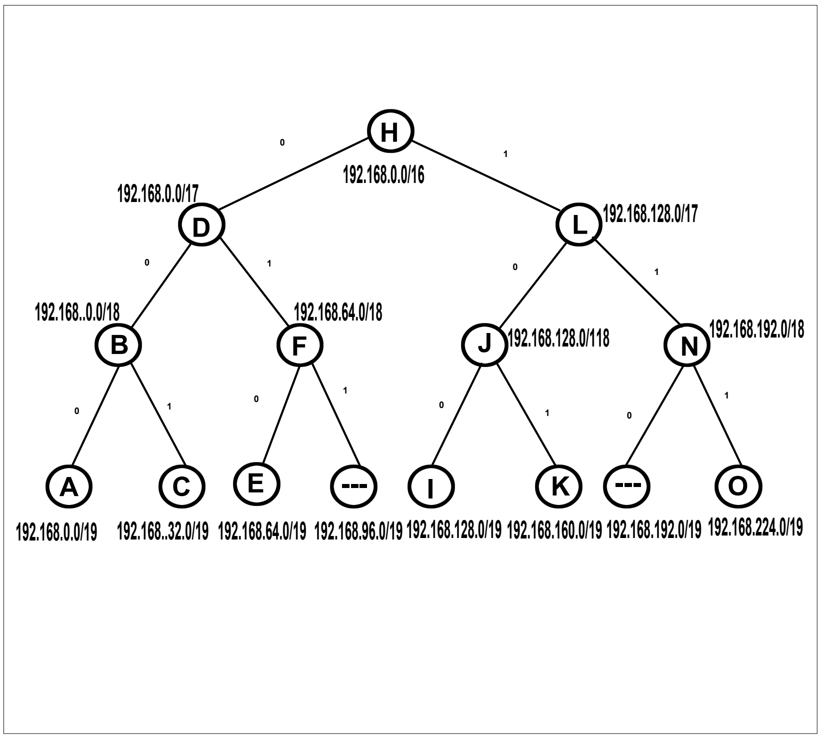

Figure 4.3: Example

performed a recursive Depth First Search (DFS). A post-order implementation of the recursive DFS algorithm is modified to meet our requirements to create an aggregation list. In a post-order traversal, the algorithm traverses the left subtree, right subtree, and then root node. By using this algorithm, at visit of every node we perform some actions to create an aggregation list as shown in the algorithm 1 (described in detail in the next section).

4.2.1 Modified DFS algorithm

We use a recursive DFS algorithm and then modify it, to generate aggregation list.

Algorithm 1 Compute the aggregation list

```
1: Visit left child node if any and store its key value in the aggregation list
2: Visit right child node if any and store its key value in the aggregation list
3: Visit parent node, check if left AND right child node's key values exist in the
   aggregation list, If "yes" remove them and save parent node's key value
```

4.2.1.1 Operations of the modified DFS algorithm

Before the algorithm runs, the aggregation list is empty. As we know that DFS defines a way to traverse the tree structure, here we are using a postorder traversal method and we modify the algorithm for our purpose to calculate the aggregation list. As mentioned in previous section the modified algorithm first traverses the left subtree when it reaches a leaf node, it stores its key value into the list. Here we represent the key values with letters for the purpose of an illustration. In step 1 the aggregation list will contain a A, then when the right child node is visited and C will be stored in the list. After visiting the left and the right child node, the parent node is visited and we check whether its left **and** right child's key values are in the list, if they are both in the list, they are removed and parent node's key value is stored in the aggregation list. This means that A and C will be removed and B will be inserted to the list. By following the same procedure when we come to node F the condition is not fulfilled, so key value of E is not removed and key value of F is not inserted. Later I and K are removed and key value of J is inserted into the list. As a result of this algorithm we will have an aggregation list containing the key values B E J O.

Figure 4.4 shows the step by step procedure to create aggregation list. Here in our example we have 6 networks prefixes in our data base that are stored in a tree structure as shown in Figure 4.3. For simplicity they are represented by the letters A,C,E,I,K, and O, without applying this aggregation algorithm we would have to tell the legacy router about all the prefixes, this would negatively affect its available memory, would use CPU resource, and bandwidth of the link between OFS and legacy router. After applying the aggregation algorithm the number of prefixes are reduced from 6 to 4. As a result we are exposing our network information **without** lying to the legacy router. Our algorithm do aggregation as much as possible to reduce the information that needs to be shared with the legacy neighboring router. Table 4.2 show the final aggregation list.

Table 4.2: Aggregation List

Aggregation List
B
E
J
O

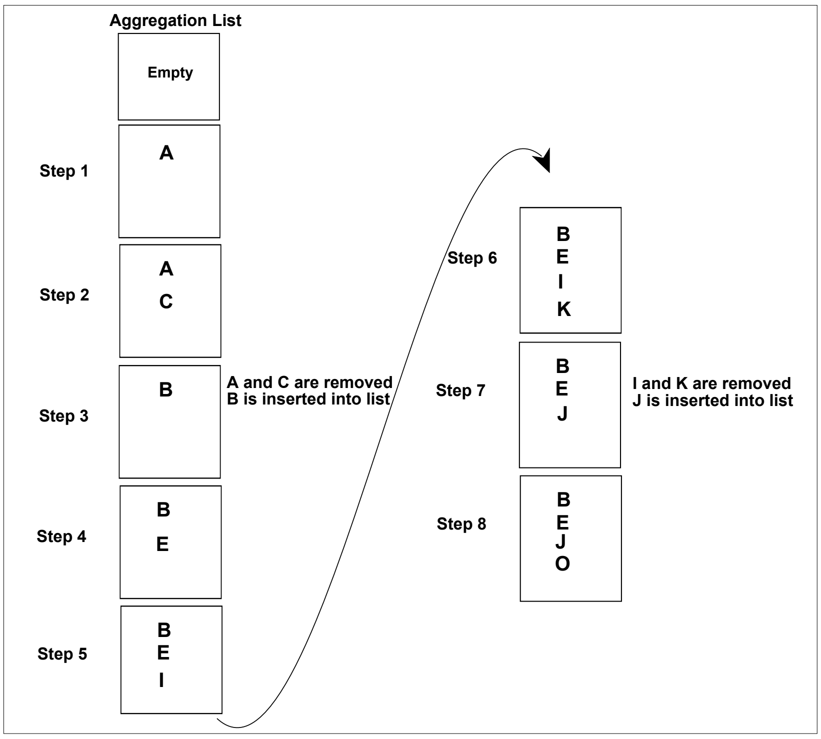

Figure 4.4: Step by step procedure to create the aggregation list

In the second case we want to reduce more routing information between OpenFlow network and legacy router. Here we will be lying about some prefixes that we do not actually have in our database, but we will claim that we have them in our database. Instead of sending 4 network prefixes as in the first case, controller will send only a single network prefix (H) in our example. In this case the controller has to maintain a list of those network prefixes for which it lied about as if it receives any request for a network that it does not atcually have, then there will be a black hole. The OFS that is connected with the legacy router will drop these request packets as it does not know what to do with such a request. The neighboring router will continue sending requests and OFS will drop the packets or forwarding to the controller for instructions as to what to do with this packet. To overcome this problem the controller has to maintain a list of those network prefixes that it does not have in its database. Let us represents the siblings of E and O with letters G and M respectively. We will maintain a list x-list, of those prefixes that we do not actually have in our database, i.e., we need to remember that we lying about them. The x-list can be calculated very easily by using a recursive DFS (Post-order traversal) algorithm, if the parent node is missing left child then the key value of missing node can be calculated easily, it will have the same prefix value as parent node but subnet mask is incremented by 1. Suppose we are missing M then its prefix value will be 192.168.192.0 as this is M's parent node's key value, but and subnet mask will be 19, we notice that this prefix value is same as parent node N, only the subnet mask is incremented by 1. Now consider the right child is missing as represented by G, it can be calculated from parent (F) node's key value. The bit representation of the F prefix is 11000000.10101000.01000000.00000000 (192.168.64.0), its right child value can be calculated by changing the status of bit ON next to the most significant ON bit of the prefix in the parent node (11000000.10101000.01**0**00000.00000000). After changing its status the value will become 11000000.10101000.01**1**00000.00000000 (192.168.96.0) and the subnet mask will be incremented by 1, it will become 19. By using the above method and a recursive DFS (post-order) algorithm, the empty node's key values can be calculated and save into the x-list. In our example the x-list will contain 192.168.96.0/19 and 192.168.192.0/19 represented by G and M respectively. Algorithm 2 will be applied in the T-Server to compute an aggregation list.

Algorithm 2 Compute the aggregation list

```
1: Visit left child node if any and store its key value in the aggregation list
2: Visit right child node if any and store its key value in the aggregation list
3: Visit parent node, check if left OR right child node's key values exist in the
   aggregation list, If "yes" remove them and save parent node's key value
```

After applying algorithm 2 the aggregation list contains **H**. Now the controller will send only an single **(aggregated)** network prefix to the legacy router, the resource consumption of the legacy router is greatly reduced as compare to first case. In this case OFS (connected to legacy router) has to store an extra list that is the list of non-existing networks, this is an additional overhead for the controller and OFS to periodically store and update the x-list and aggregation list. Let us consider what happens when we receive the request for those prefixes **(G,M)** that we do not have in our database. Figure 4.5 illustrates what happens when our border OFS receives a request for the network prefix that it has in the x-list.

For example, the border OFS receives a request for 192.168.96.0/19, then the OFS does not know what to do with this request, so it will inform the controller. Now the controller will communicate with T-Sever and instruct it to do the following actions: remove the prefix (for which a request was received) from the x-list, insert the rest of x-list values in their respective places in the tree and then apply algorithm 1 to calculate the new aggregation list. According to these instructions the T-Serer will remove G from x-list and insert **M** into the binary tree as shown in Figure 4.6. M will be inserted as a left child of node **(N)**. After this algorithm 1 is applied to compute a new aggregation list.

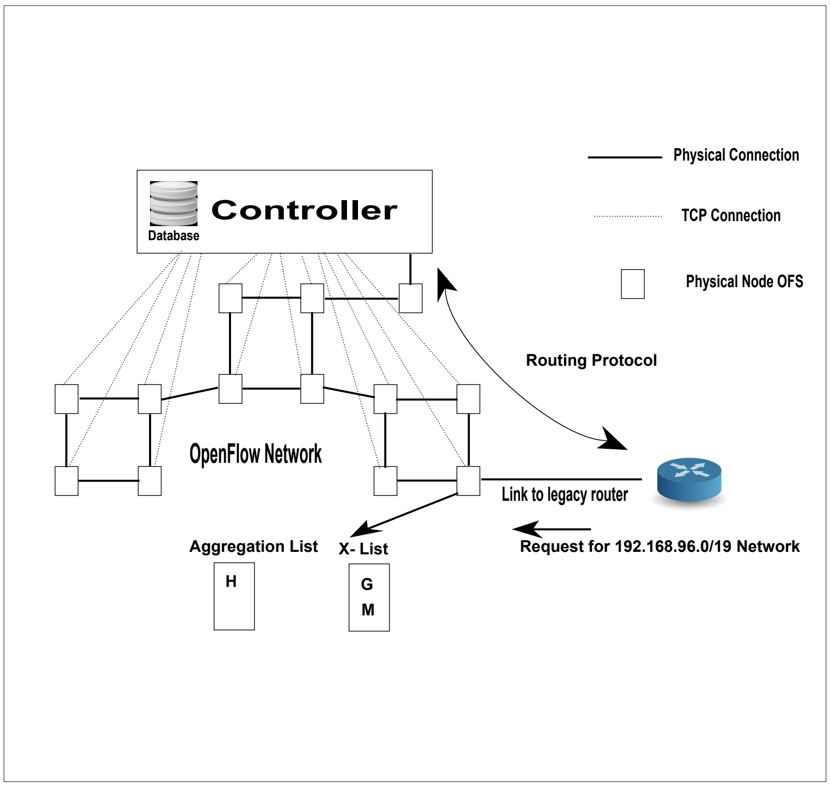

Figure 4.5: Request for non-existing network received

Figure 4.7 shows the step by step procedure to calculate a new aggregation list by applying algorithm 1. We notice that new aggregation list have aggregated and specific network prefixes. After computing this aggregation list, the temporarily inserted values are deleted from the tree, which means that in our case M will be deleted. Overall conclusion of this chapter is that aggregated information will reduce the consumption of resources (memory and CPU) at legacy router to whome OpenFlow controller communicates and bandwidth utilization of the link (between the OpenFlow network and the legacy network) will also be reduced.

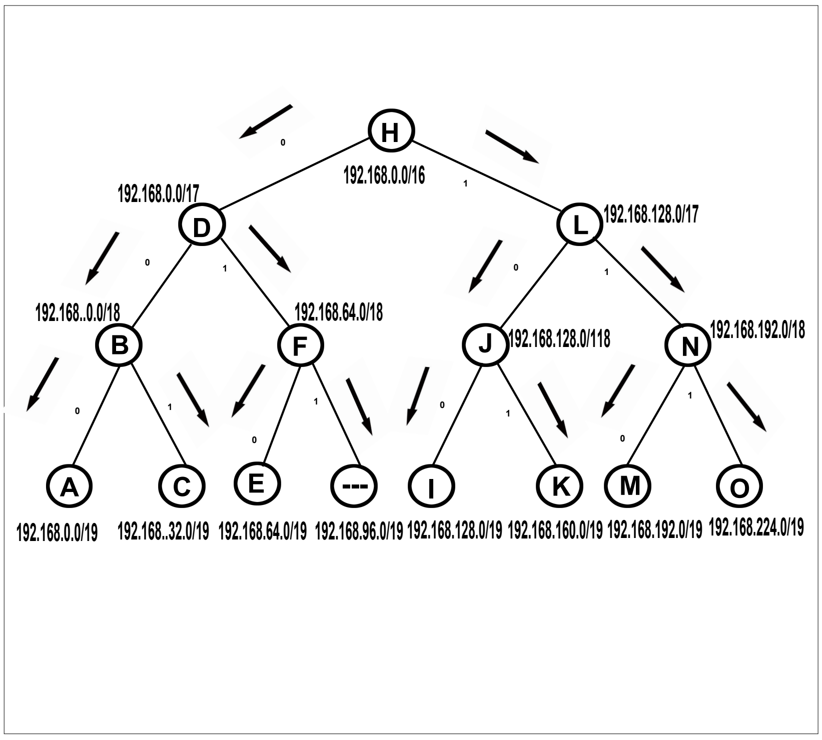

Figure 4.6: Insertion of x-list values in the tree

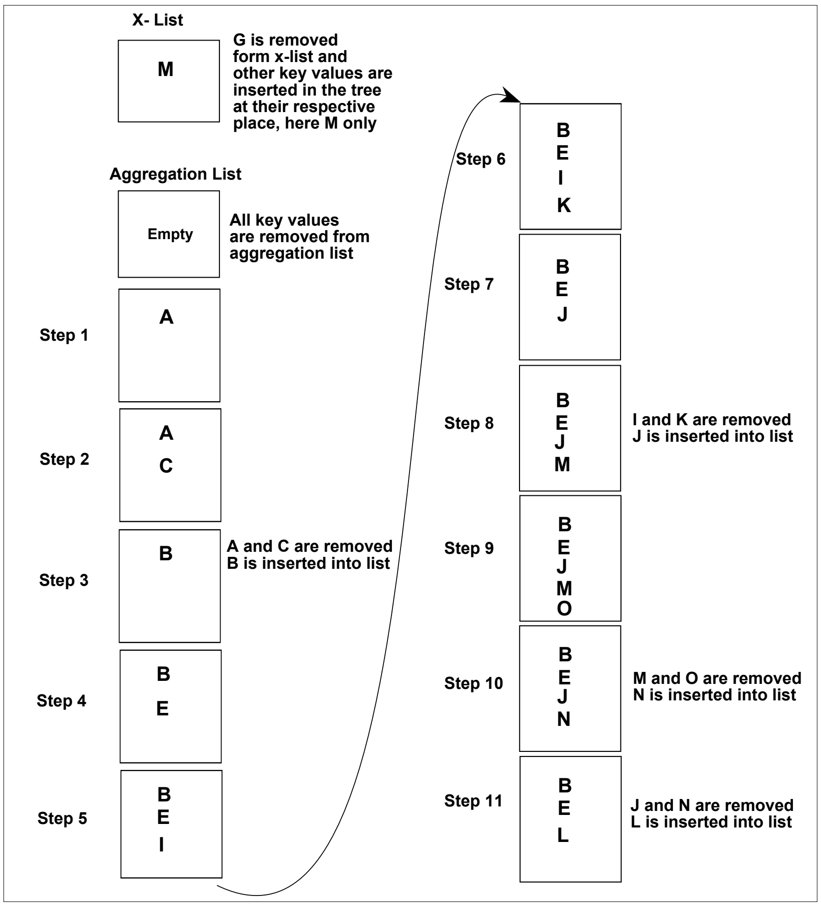
X- List
M
G is removed form x-list and other key values are inserted in the tree at their respective place, here M only
Aggregation List
Empty
All key values are removed from aggregation list
Step 1
A
Step 2
A C
Step 3
B
A and C are removed B is inserted into list
Step 4
B E
Step 5
B E I
Step 6
B E I K
Step 7
B E J
Step 8
B E J M
I and K are removed J is inserted into list
Step 9
B E J M O
Step 10
B E J N
M and O are removed N is inserted into list
Step 11
B E L
J and N are removed L is inserted into list

Figure 4.7: Step by step procedure to create the aggregation list

Chapter 5

Conclusion and Future work

This chapter consist of three sections. Section 5.1 concludes the whole thesis work. Section 5.2 suggests some future work and section 5.3 describes required reflections.

5.1 Conclusions

In this study we presented, what is the difference between traditional routing architecture and Software-defined Networks architecture. We also discussed different possible scenarios that can be used to present OpenFlow networks to the legacy network via OSPF. The experiments and discussions shows that size of OpenFlow network domain directly effects the resources (memory and CPU) of neighboring legacy router. The link's bandwidth is also effected. To overcome these problems we suggested several models for aggregation in an SDN environment, also we developed two algorithms that can automate the route aggregation. Aggregating information from the OpenFlow controller towards the legacy router can reduce the consumption of memory, CPU, and human resources (that would have been expended manually performing route aggregation).

5.2 Future work

The proposed solutions in this study are very innovative but they need improvements in future research. Some of suggested future work are introduced in this section. One of the basic future step for this thesis work is the implementation of design models and algorithms in real environment. Future researcher with programming skills can improve the algorithm. Second suggestion for future work is to pre-calculate all possible aggregation lists for all networks prefixes that we have in x-list. Furthermore we suggest an introduction of a new type of LSA (x-LSA) in

OSPF that contains all network prefixes that are in x-list. Through this x-LSA the neighboring router will be informed about the networks prefixes that we do not have in our database. In this way the legacy router router will not send request for those networks.

5.3 Required reflections

This thesis work presents an automatic way of route aggregation in Software-defined Networks. In this study the introduction of new servers (VM-Server and T-Server) in proposed design models effects CAPEX (Capital expenditure) and OPEX (Operational expenditure) of an ISP. Despite increasing the CAPEX and OPEX for the ISP, the proposed solution can reduce the labor cost as the ISP no longer needs to do aggregation manually and this solution dramatically reduces the consumption of resources (memory and CPU) of the legacy router. As a part of an industrial project, this thesis project is expected to have positive impact on the Software-defined Networks industry. Implementation of the solutions suggested in this thesis will allow automatic route aggregation in SDN. ISPs deploying SDN architecture could benefit from the proposed design models and route aggregation solution.

Bibliography

[1] Anastasius Gavras and Adam Kapovits. Software-defined-networks. http://www.eurescom.eu/news-and-events/eurescommessage/eurescom-messge-2-2012/software-defined-networks.html/, September 2012.

[2] Open Networking Foundation. The new norm for networks. https://www.opennetworking.org/images/stories/downloads/white-papers/wp-sdn-newnorm.pdfl/, September 2012.

[3] Manuel Palacin Mateo. Master thesis, openflow switching performance, school: Politecnico di torino, torino, italy, 2009.

[4] M. Kind, Westphal, A. Gladisch, and S. Topp. SplitArchitecture: Applying the Software Defined Networking Concept to Carrier Networks. In *World Telecommunications Congress (WTC), 2012*, pages 1–6. IEEE, 2012.

[5] Xin Zhao, Yaoqing Liu, Lan Wang, and Beichuan Zhang. On the aggregatability of router forwarding tables. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[6] Dávid Jocha, András Kern, and Kiran Yedavalli. Scalability aspects of centralized control of mpls access/aggregation network.

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[8] D. Simeonidou, R. Nejabati, and S. Azodolmolky. Enabling the future optical internet with openflow: A paradigm shift in providing intelligent optical network services. In *Transparent Optical Networks (ICTON), 2011 13th International Conference on*, pages 1–4. IEEE, 2011.

[9] A. Tavakoli, M. Casado, T. Koponen, and S. Shenker. Applying nox to the datacenter. *Proc. HotNets (October 2009)*, 2009.

[10] Cisco : Internetwork design guide – designing large-scale ip internetworks.

[11] Jeff Doyle. Scaling ospf and is-is, 2005.

[12] John Moy. Rfc 1245: Ospf protocol analysis. 1991.

TRITA-ICT-EX-2013:101