ADAPTIVE ROUTING USING SDN AND NFV

A Thesis
in
Electrical Engineering

Presented to the Faculty of the University
of Missouri-Kansas City in partial fulfillment of
the requirements for the degree

MASTER OF SCIENCE

by

TEJAS BALKRISHNA PARAB

B.E., University of Mumbai, Maharashtra, India, 2014

Kansas City, Missouri
2017

ADAPTIVE ROUTING USING SDN AND NFV

Tejas Balkrishna Parab, Candidate for the Master of Science Degree

University of Missouri-Kansas City, 2017

ABSTRACT

In recent times, the primary focus of every ISP is to deliver high quality of service for multimedia applications. Due to ever-increasing network traffic, it is challenging for ISPs to accomplish optimal network conditions. To achieve high-quality services implementation of network monitoring and QoS routing algorithm is inevitable. Also, several modifications occur while network planning especially in managing physical devices to host network services required to achieve QoS routing. Dynamic routing in physical network hardware is not suitable for cost-effective business models. Therefore, to lower operational and capital expenditure, we propose a technique called adaptive routing managed with the support of SDN (Software Defined Networking and NFV (Network Function Virtualization). We introduce an economical routing model which will assist in meeting the demands of the consumers and at the same time provide significant savings in product cost and usability. To perform adaptive routing, we designed a topology in a virtualized environment. We programmed RINA (Recursive InterNetwork Architecture) for SDN management which acts as a steady northbound API. To implement NFV, we hosted Intrusion detection service(IDS) as a Virtual Network Function (VNF). We created two scenarios where we execute shortest path routing and adaptive routing. To compare these different

routing scenarios, we ran a client-server DASH (Dynamic Adaptive Streaming over HTTP) application. Furthermore, we injected Iperf traffic into the network categorized as unwanted traffic.

Lastly, we performed adaptive routing applying control theoretic technique with the help of SDN and NFV. By comparing two routing scenarios, we observed a notable difference in throughput for DASH application.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Computing and Engineering, have examined a thesis titled “Adaptive Routing Using SDN and NFV,” presented by Tejas Balkrishna Parab, candidate for the Master of Science degree, and hereby certify that in their opinion it is worthy of acceptance.

Supervisory Committee

Deepankar Medhi, Ph.D., Committee Chair
Department of Computer Science & Electrical Engineering

Kenneth Mitchell, Ph.D.
Department of Computer Science & Electrical Engineering

Cory Beard, Ph.D.
Department of Computer Science & Electrical Engineering

CONTENTS

TABLES

ILLUSTRATIONS

ACKNOWLEDGMENT

I would first like to thank my thesis advisor Deepankar Medhi, Ph.D. His guidance helped me in all the time of research and writing of this thesis. Besides my advisor, I would like to thank the rest of my thesis committee: Kenneth Mitchell, Ph.D., and Cory Beard, Ph.D. for their encouragement, insightful comments, and questions. Also, I thank my friends in University of Missouri Kansas City to guide me through all the difficulties. In particular, I am grateful to my friend Marmik Patel for helping me with the software issues. Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them. Thank you.

CHAPTER I

INTRODUCTION

1.1 Overview

A computer network is a complex communication system, which allows many networked devices to exchange information. Computer networking has evolved in last few years following the requirements of the existing technologies. Currently, traditional network architecture implemented on a large scale consists of many hardware computing devices which provide reliability and performance efficiency. Although the existing internet is mainly dependent on conventional network architecture, it may fail to achieve necessities of the future internet. Future internet comprises services related to the consumerization of information technology, cloud services and upcoming concepts like IoT (Internet of Things). All these services will lead to challenges such as the increase in the number of communication devices, high network capacity, elastic scaling of computing and network devices. To meet all the requirements of the future internet, there is a need for various network functions such as network management, network monitoring, and scalability. Traditional network architecture might fail to provide all the requirements because of the increase in Capex (Capital Expenditure) and Opex (Operational Expenditure). Future internet requires more programmability and centralized view of network devices needed for the automation of network services. An innovative approach called Software-Defined Networking is introduced to overcome many challenges in a traditional network.

1.1.1 Software Defined Networking

Computer networks are complex and difficult to manage; software-defined networking helps us to ease management of networking devices. Software-defined networking is an architecture divided into three layers.

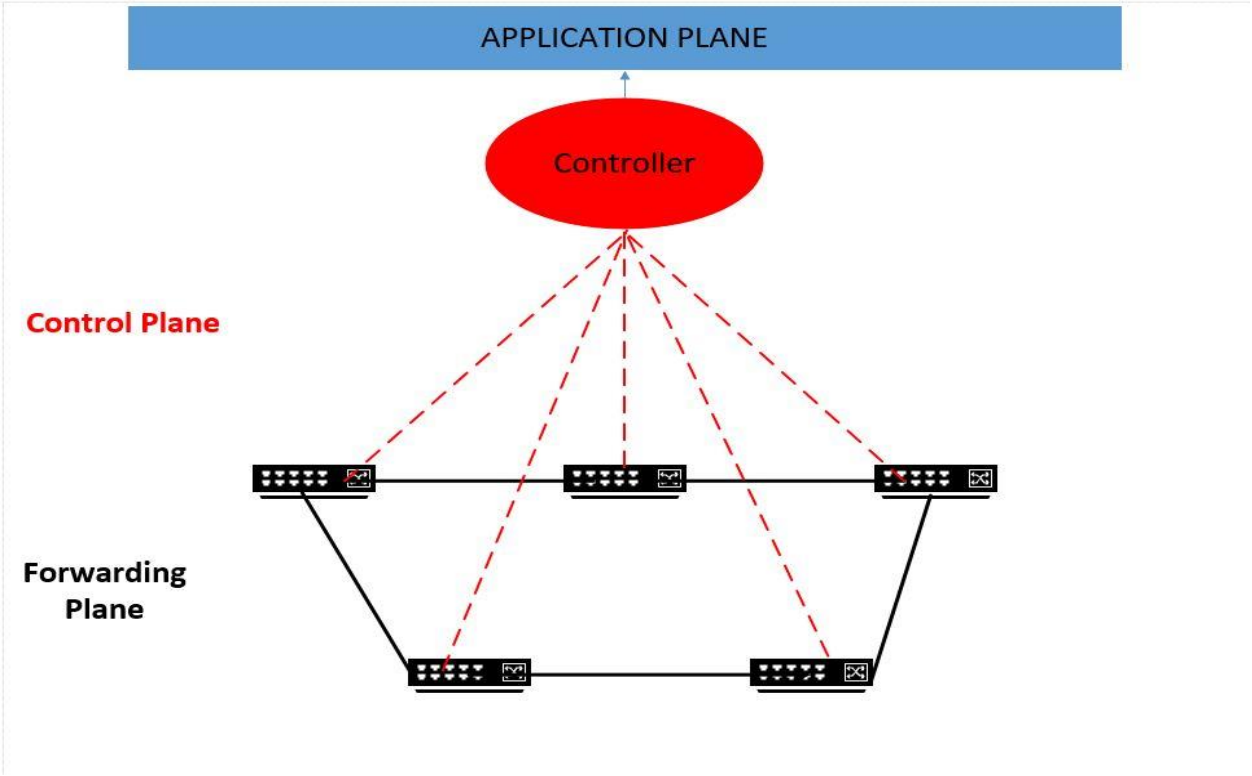

Figure 1: Software Defined Networking Architecture

- Control layer: This layer in SDN is decoupled from networking infrastructure layer. Control layer consists of a controller which provides a centralized view of the network infrastructure. With the help of centralized view provided by the controller, a network can manage, configure underlying devices and program automated SDN policies. SDN controller is the core entity of a network which is used to install flows into network devices.

Controller uses protocols such as OpenFlow, NetConf to install optimal routes in forwarding devices.

- Forwarding layer: The forwarding layer consists of all network devices; hence it is also known as infrastructure layer. Forwarding layer enables data transfer to and from clients through multiple protocols installed by the SDN controller. Forwarding plane consists of network devices such as router or switch. These network devices can perform various tasks on data packets such as dropping, prioritizing, explicit blocking of packets, etc.
- Application layer: Several types of applications and services together form the application plane. In this layer network applications such as firewall, IDS and load balancers can be managed. Open programming nature of SDN architecture can help these applications to provide some information to the SDN controller which will help the network to be more flexible, resilient and well-organized. Applications which directly support operations for the forwarding plane such as routing process within the control plane are not considered as a part of the application layer.

So, all the three planes of SDN architecture interact with each other which enables the network to be more automated. All the three planes of SDN communicate with each other through APIs.

1.1.2 SDN APIs

Application programming interfaces (APIs) in SDN can be open or proprietary. Although the initial goal of SDN was to create open APIs which could interact with devices, services, applications developed by different vendors.

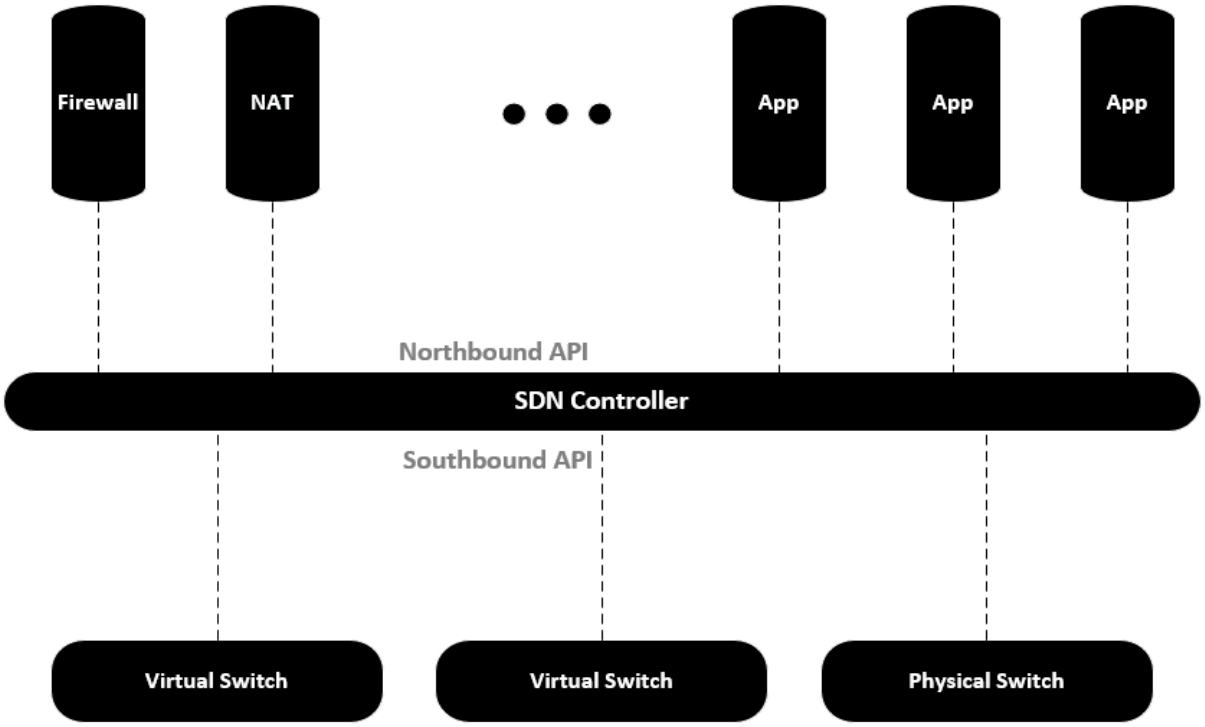

Figure 2: Overview of SDN

Following are the two most important SDN APIs:

- Southbound API: The purpose of the southbound API is to develop a communication channel between the control layer and forwarding layer. Southbound API is established between the controller and the network devices (routers and switches). One of the most commonly used southbound API is OpenFlow protocol. The Open Networking Foundation released OpenFlow in 2011. OpenFlow is layered on the top of Transmission Control Protocol (TCP). OpenFlow protocol is used by the controller to install flow entries internally in a switch. OpenFlow enhances programmability nature of SDN. Due to OpenFlow routing instructions can be periodically installed in network switches; also the specific type of actions and rules can be configured into network devices. OpenFlow

protocol has been designed to make a network more reactive to the real-time traffic demands. There are many southbound APIs other than OpenFlow in the market such as Cisco OpFlex and Network Configuration Protocol(Netconf).

- Northbound API: The purpose of the northbound API is to create a communication link between the controller and the applications over the top of the network. Northbound API is developed to transfer state information of the services from the application layer to control layer. Northbound APIs are not directly linked to the control layer. Northbound API can facilitate new automated innovations depending on the needs of a specific application for the improvement of the network. Currently, there are no standards defined for Northbound API. Developers are creating different northbound APIs according to the requirement of the services or applications. Open network foundation is working on developing northbound API standards for which they have established Northbound Working Group. Northbound API is also seen as a network management component for SDN application plane. Managing SDN application layer under the essential needs of the network will improve network orchestration. With the integration of NFV (Network function virtualization) and SDN, many automated innovations can be established.

1.1.3 Network Function Virtualization

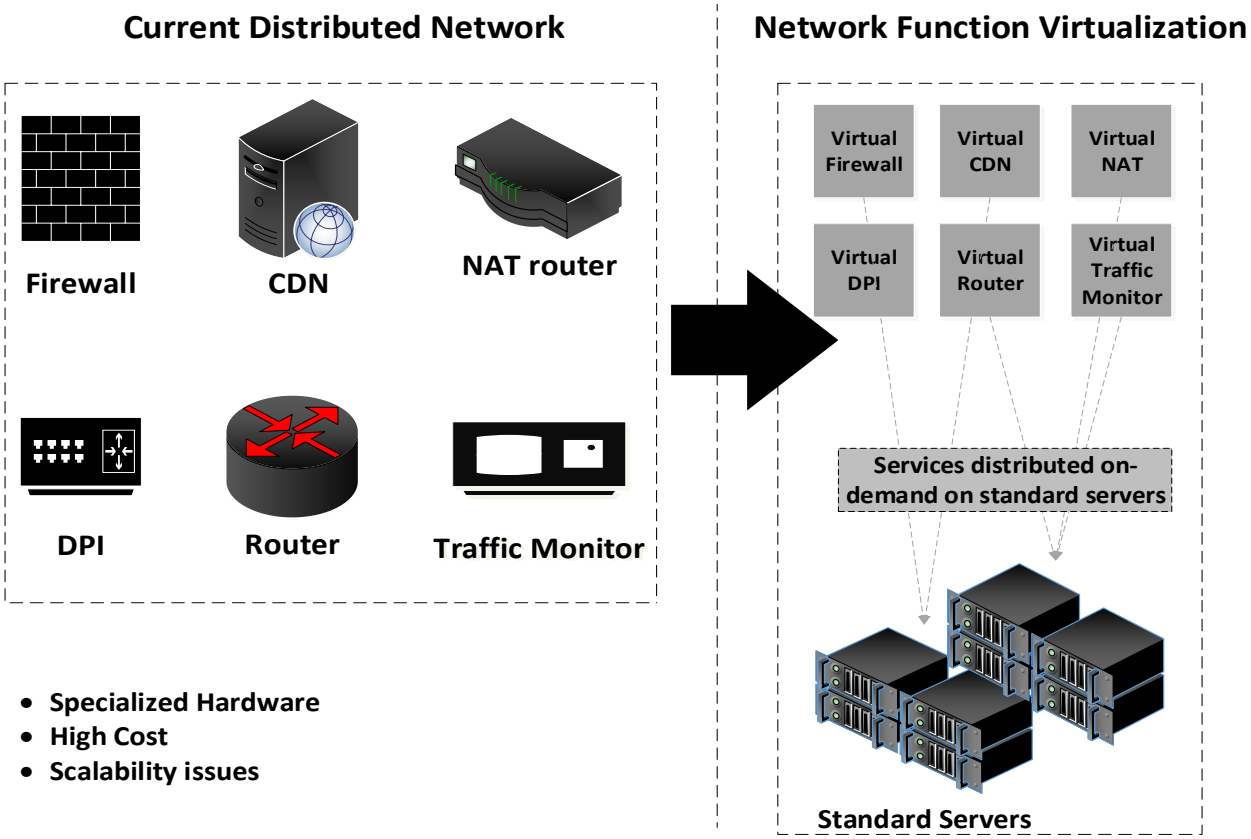

Figure 3: NFV Vision

Network function virtualization is a concept of virtualizing physical network devices which will collectively form a structured block that can be linked together to produce a communication service. The purpose of NFV is to decouple network functions from physical network devices which host services like load balancing, Firewall, CDN; to be hosted on virtual machines. With the implementation of NFV network administrator would not be dependent on hardware devices to build a service chain for communication systems.

Following are the advantages of NFV:

- Flexibility: Physical network devices have limited functions related to computing, memory, storage and networking facilities. The modification of these tasks needs more

time and money. With the help of NFV, administrators can select physical devices which have hardware capacities required to provide optimal network architecture.

- Agility: The capability to configure, deploy, terminate network functions is very agile in NFV's. With NFV, a provider will be able to allocate on-demand resources to improve services for the end-users. On the other hand, in traditional networks allocation and deallocation of network resources will consume more time and money because of the installation of physical hardware resources.
- Scalability and elasticity: Nowadays, service providers must keep up to provide the best network capacity to end users due to rising bandwidth dependent applications and services. Capacity management in traditional networks takes time and planning. NFV can solve this problem by its ability to shrink and expand resources. Network functions(NF's) can provide additional CPU, storage, and bandwidth which can be requested by VIM (virtual infrastructure manager) and allocated to NF's. To alter parameters like CPU, storage, and bandwidth in traditional networks it will require a full physical device replacement.
- Cost reduction: Every service provider tries to lower capital investment in maintenance of network devices. To decrease the budget for system maintenance in IT departments virtualization and software-defined techniques are introduced. With NFV, the overall network Capex (Capital expenditure) and Opex (Operational expenditure) can be reduced.

1.1.4 Motivation and Contribution

Research in Software Defined networking has created solutions to many complex issues which occur in traditional networking. We can find various routing matters related to multimedia communications and one of the critical challenge is QoS routing. We cannot achieve specific goals of QoS (Quality of service) routing using traditional networks such as determination of paths that

satisfy QoS constraints and at the same time determining an alternate path for unwanted traffic. To achieve certain tasks, we need the network to be more flexible and programmable. Software-Defined Networking with the integration of NFV (Network Function Virtualization) can help networks achieve these tasks. To achieve optimal path required for QoS routing, a network should be able to implement adaptive routing. Adaptive routing can help network achieve optimal throughput and desirable QoS parameters. Adaptive routing can also assist in improving link bandwidth utilization which in turn enhances performance for multimedia applications.

To test the integration of SDN-NFV and implement adaptive routing, we created a virtualized testbed using Mininet which helps to set up an SDN network. In this testbed, we created a Client-Server scenario separated with four OVS (OpenFlow Virtual switches). Integration of SDN and NFV will need network orchestration and management. Moreover, the interaction between Virtual network functions (VNF) and the controller for efficient operation of NFV requires network management via northbound API. Therefore, we programmed a northbound API using RINA (Recursive Inter-Networking Architecture) framework. RINA can be used to manage SDN and NFV architecture. Virtual functions categorized into multiple services and applications are hosted on a virtual machine or physical machine. In our case it is a virtual node which is hosting an Intrusion detection system(IDS) called SNORT. SNORT can detect IP packets according to programmed actions sets in the application. By using client-server DASH (Dynamic Adaptive Streaming over HTTP) video application, we inspect the difference in bitrate of the video between shortest path routing and adaptive routing. When two or multiple nodes in a network are assigned same shortest path (hop count), it is considered as a shared path. In this scenario, adaptive routing will provide optimal network path. In the next section, we describe our approach to implement adaptive routing using SDN and NFV.

1.2 Approach

The basic functionality of our approach is explained in Figure 4. For implementing adaptive routing with the integration of SDN and NFV, we are using control theoretic procedure as shown in figure 4. The control theoretic approach was possible due to open framework provided by SDN. We developed a topology with network components, where video segments are sent to the client.

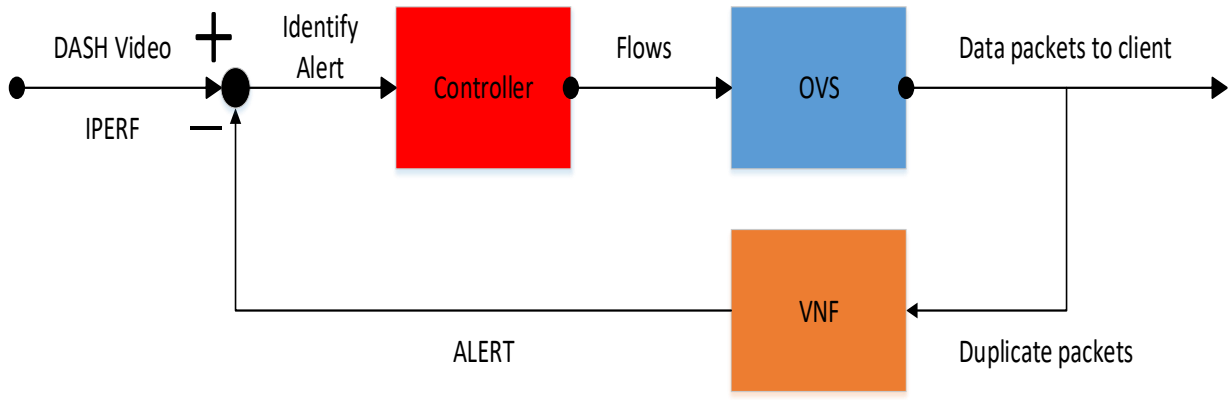

Figure 4: Basic Block Diagram

While video segments are transferred to the client, simultaneously those packets are duplicated and sent to VNF. VNF provides feedback to the controller which is generated by network service hosted on it. In our case, VNF is hosting an IDS service which displays alert with timestamp and IP header information for each packet of DASH video. After the receiving the feedback from VNF, the controller will analyze the feedback and distinguish it as positive (DASH) or negative feedbacks (Iperf traffic) from the identified alerts. The controller will then install forwarding flows in OVS switches which will allow the network to implement adaptive routing. In figure 4, we could visualize network components such as a controller (control layer), OVS switches (forwarding layer) and VNF (application layer) form three layers of SDN. Therefore, these three layers are used to develop control theoretic approach in networking.

The rest of the thesis is organized as follows. After describing related work in chapter 2, we present our method in chapter 3. In chapter 4, we describe the experimental setup, and then we present our results. Lastly, in chapter 6 we conclude our work by stating the limitation of our method and the related future work.

CHAPTER 2

LITERATURE SURVEY

We begin literature survey by discussing implementing SDN and NFV together for accomplishing automated network. We aimed to apply adaptive routing in multimedia communications for eliminating QoS constraints. Software-defined networking supports the idea of openness which leads to programming network devices. Hence, the configuration of southbound and northbound API in SDN allows the functioning of SDN enabled NFV network. While designing SDN and NFV network, services like network management and orchestration are very imperative. Therefore, network management is very crucial for execution of adaptive routing.

RINA, which can be used as network management tool and northbound SDN API allows us to execute control theory in networking. Execution of control theoretic approach to operate SDN and NFV network together is previously done in [1], where RINA is used for managing VNF nodes. In [1], monitoring CPU load of two VNF instances is carried out and load balancing traffic for respective VNF's is executed. Our focus is to implement the similar technique to improve throughput for multimedia communications by accomplishing the idea of adaptive routing. Quite a few papers have been published regarding the development of northbound SDN API. The authors in [9] have developed an NBI architecture which follows principles and guidelines provided by the Open Networking Foundation (ONF). The work in [10], demonstrates NBI design using REST services which is the most common choice for northbound API in SDN. The paper correspondingly talks about issues faced using REST API. An improved version of RESTful API is developed by introducing HTTP content negotiation mechanism which will allow clients to choose various representation formats of fixed resource URIs.

The research in [3] and [4] focuses on QoS provisioning using SDN. The work in both the papers shows the importance of application layer in SDN used for QoS routing. With the implementation of a particular framework for their research, an idea of flexibility provided by SDN application layer is highlighted. QoS routing is achieved by using network monitoring services in [3] and [4]. In this thesis, our foremost objective was to provide maximum link bandwidth to video streaming applications. To achieve similar goals different ideas are implemented in [5] and [13] using NFV as a traffic monitoring system. In [5] optimal route for data transfer is calculated considering QoS parameters and later priority flows are installed in the network. Deployment of NFV as a routing function is carried out in [12] where different routes are established, and packets are classified based on ipv4 and ipv6. Our work is related to [5] where optimal routing function is generated to improve overall throughput using network monitoring. But there is one notable difference; our emphasis is to provide best effort service to a single application (DASH video application) with the implementation of adaptive routing. For our research, we use DASH video application explained in [2].

CHAPTER 3

METHODOLOGY

3.1 Adaptive Routing

In this thesis, we have implemented adaptive routing in a virtualized topology using SDN and NFV. Adaptive routing which is also known as dynamic routing is a method which will establish optimal routing path for data packets based on particular situations. With the help of SDN and NFV implementation of adaptive routing is much easier. Due to the openness of SDN, determination of optimal routing path and directing packets through this optimal path is possible. Adaptive routing can be calculated using many parameters such as hop counts, bandwidth, CPU usage and other networking constraints to provide optimal path. In our testbed, the high priority packets will be directed towards the optimal path (shortest path) and the low priority packets will have an alternate path. The adaptive routing mechanism will be triggered only for the required applications, in our case it is DASH video application. By using virtual network function as a service with the integration of SDN we developed a control theoretic approach which enables Adaptive routing for DASH video application for our topology. To execute this idea management of SDN and NFV is essential. The next portion of the chapter explains the management of NFV and how to use northbound API for SDN.

3.2 Managing NFV using RINA

The application layer of SDN needs a northbound API or network management interface to open a channel between network applications and an SDN controller. Currently, there is no standardize SDN northbound API assigned by the Open Networking Foundation(ONF), so we worked on RINA architecture which can be used as an SDN northbound API and network management interface. RINA (Recursive inter-networking architecture) is a clean slate

architecture. It is developed by IRATI (Investigating RINA as an Alternative to TCP/IP) group. RINA is based on the principle of Inter-process communication (IPC).

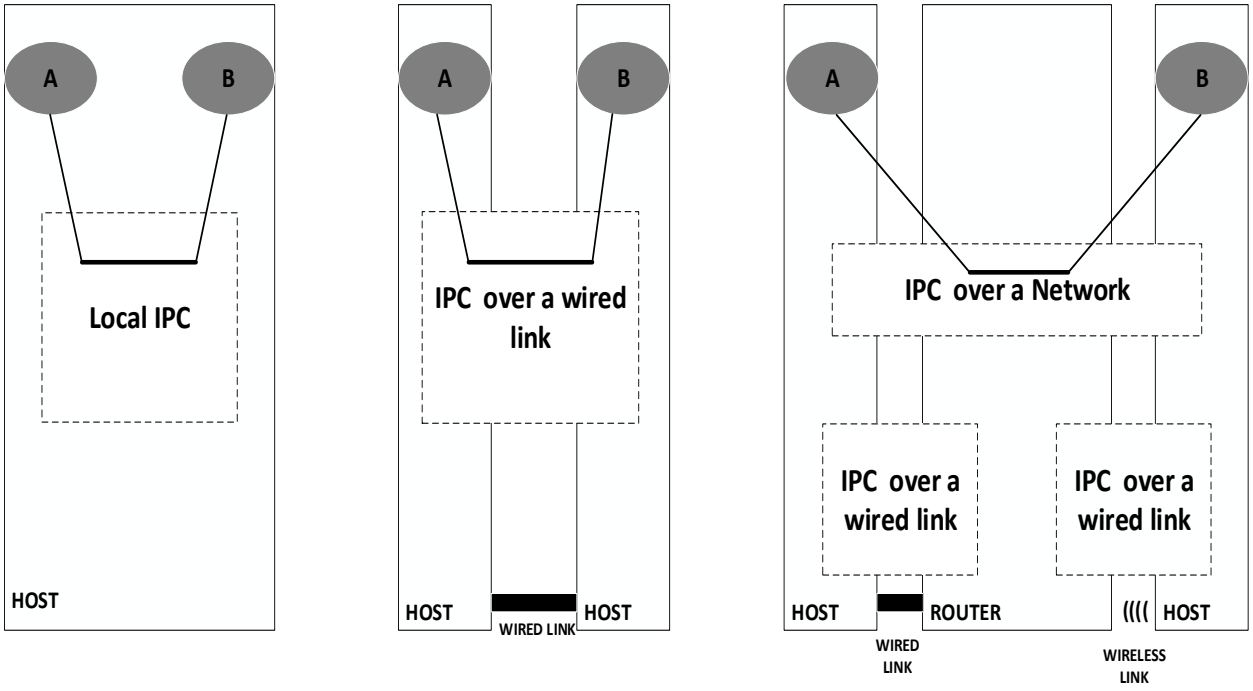

Figure 5: Inter-Process Communication

The primary principle of this architecture is that networking is a single-layered distributed inter-process communication which repeats over different scopes. In Figure 5, we can see that IPC communication can vary for different scenarios. IPC will create a communication channel between processes for every scope. IPC channel can be formed in a single host, multi-host and over a network as shown in Figure 4. A detailed functionality of IPC is shown below in figure 5.

3.2.1 Inter-Process Communication

In figure 6, IPC process is explained briefly in terms of a network topology. In the figure below, Consider all the vertical rectangles as physical machines and solid circles as IPC processes. All these solid circles together form a horizontal building block which is called a Distributed IPC

facility(DIF). The explicit color of the solid circle shows the membership of different DIFs. Adding and removing of IPC processes in a DIF is possible. Black solid lines represent the physical interconnections between devices.

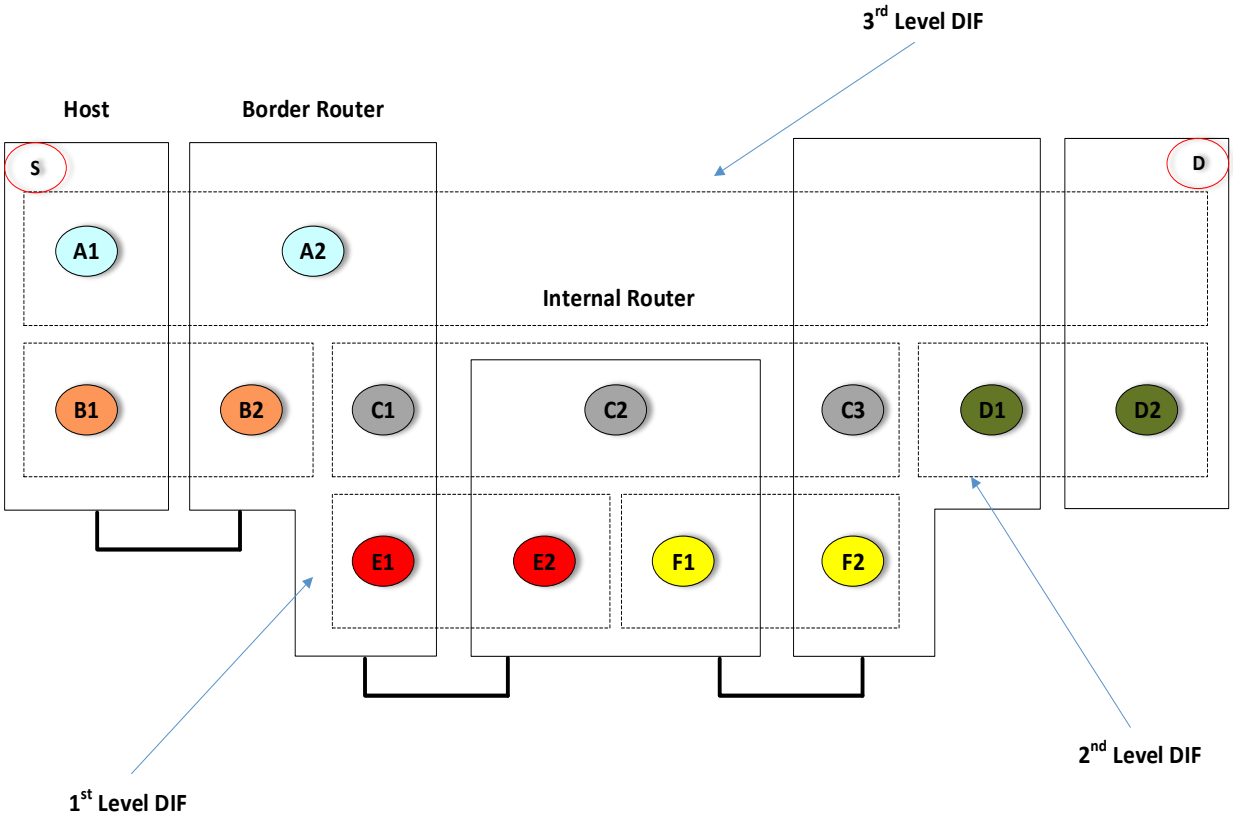

Figure 6: RINA Mechanism

Every DIF can be viewed as a layer of communication. RINA operates on the principle of layered recursive networking; therefore all the DIF's are assembled into one layer. This one layer has many clustered DIF's where required processes repeat over different scope to form a communication channel. For example, consider Figure 6 wherein 1st level DIF will deliver services for 2nd layer DIF. Providing services to the lower DIF's which primarily performs recursive communication is one of the principles of RINA. There is no separate protocol for every single layer, just the configuration and policies can be different depending upon the DIF (Distributed IPC facility). A

collection of DIFs is called Distribution Application facility (DAF) which perform similar tasks and use shared states.

3.2.2 Distributed Application Facility

Figure 6 illustrates the functionality of DAF for different application entities.

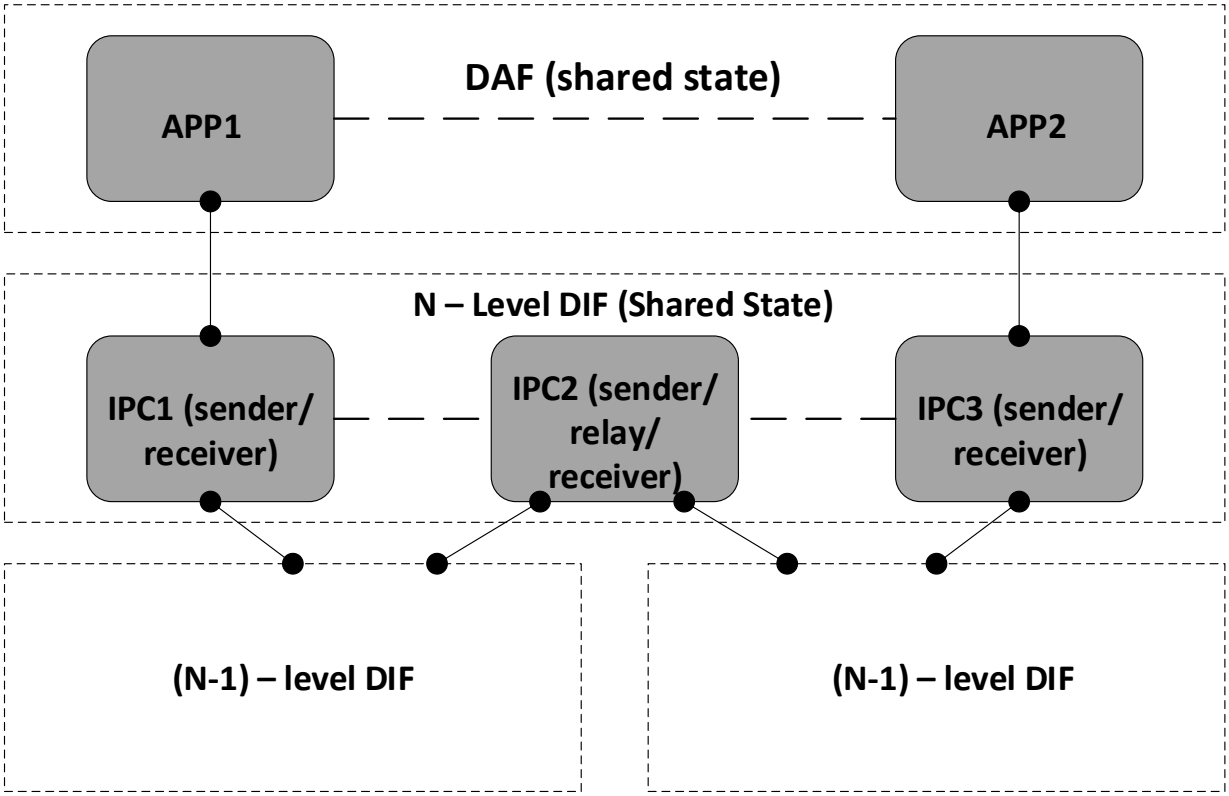

Figure 7: Distributed Application Facility

In figure 7, consider top level DAF (Distributed application facility) which works as a shared state to perform various tasks. DAF consists of different Application entities (App1 and App2), which can execute a particular function on distinct levels such as single host, multi-host or a network level. One can create numerous services such as video streaming, implementing routing algorithms and other network related applications by using DAF. Interaction of different application entities needs low-level data transfer which is provided by internal RINA APIs.

3.2.3 Internal RINA APIs

Figure 8 describes the significant components in the operation of the Application process. RINA requires internal API's for storage, processing application data. Resource information base (RIB) is a database used to store application information. RIB daemon is an essential component for accessing critical data stored in local RIB and a remote RIB which is stored in another application entity. IPC resource manager (IRM) creates an interface to manage underlying IPC's which is considered as low-level DIFs.

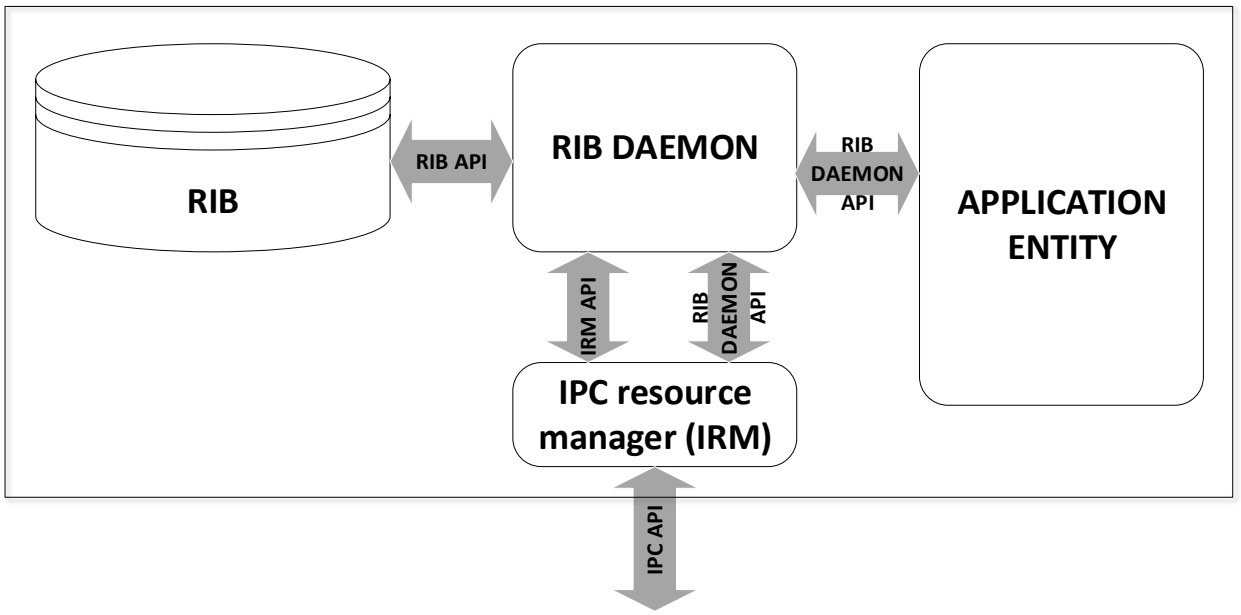

Figure 8: RINA Application APIs

As shown in Figure 8, RINA has two primary APIs which are open for users to build applications and develop new network services.

Following are the two RINA APIs:

- RIB daemon API: This API is based on the principle of subscribing and publishing. The interaction between two application entities can be carried by creating sub/pub event. PUB

event is used to publish useful application information. Corresponding application entity will retrieve this information by creating SUB event.

- IRM API: This API supports the reservation of application connections (flows) to other application processes and transferring messages over existing connections.

We have used these following APIs to create a northbound communication between the controller and NFV service hosted on a virtual network instance. A detailed explanation of implementing RINA API can be seen in Figure 9.

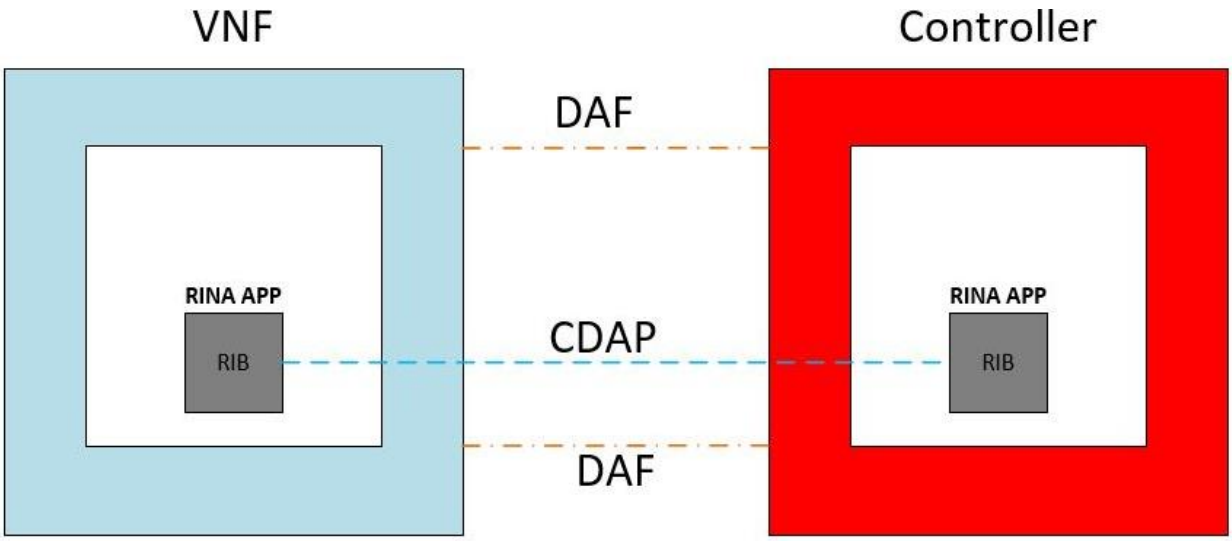

Figure 9: RINA as Northbound API

As shown in figure 9, application entities are created which resides in both controller and VNF (virtual network function). In my testbed, VNF is hosting an IDS (Intrusion detection system) service. An object-based protocol which is called as Common Distributed Application protocol is used for managing two different API components in RINA. To implement NFV, there was a need to create a DAF (Distributed application facility) which generates alert for a specific type of packets. Generation of an alert file can be considered as using it as a service, hence DAF. VNF publishes the information of the DAF (Alert) after creating a PUB event using RIB daemon API.

The corresponding information is subscribed by creating a SUB event by the controller. After every 1 seconds, VNF publishes load information which is subscribed by the controller. Considering figure 7 reference, VNF and controller should be viewed as application entities App1 and App2. SNORT is hosted on the VNF as a IDS service. SNORT plays a vital role in implementing adaptive routing based on the principle of control theory.

3.3 Intrusion Detection System(IDS)

The application layer of SDN can host many network services which are a significant part of an automated network. For our testbed, we installed an open source IDS which can capture packets and create alerts for the defined detection rules. SNORT is an open source firewall which provides network intrusion detection service. It can be used as a packet sniffer which monitors real-time traffic. It is widely used as a TCP/IP traffic analyzer. In our testbed, VNF instance is hosting SNORT application as a network service. For example, consider figure 10 where a snort rule is specified to create alerts for monitoring packets. As shown in figure 10, an alert action is defined for TCP packets which have 10.0.0.4:8001(source IP and port) and 10.0.0.1:any (destination IP and port), here "any" means a random port number. Snort will now generate an alert file with a message “DASH VIDEO DETECTED”.

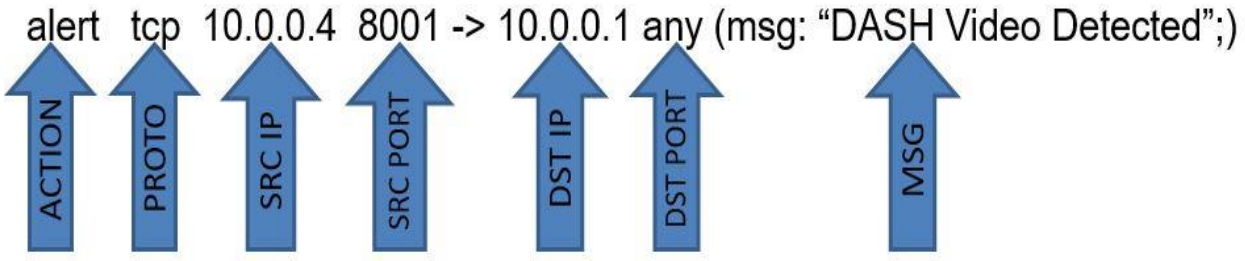

Figure 10: SNORT Rule Format

Below we can see the alert file generate the content:

09/07-23:23:12.848320 [**] [1:10000000:0] DASH VIDEO DETECTED [**] [Priority: 0] {TCP} 10.0.0.4:8001 -> 10.0.0.1:55316

This content information which is generated by VNF(SNORT) instance is transferred to the controller using northbound API. In next chapter, we will see the implementation of NFV, SDN and northbound API(RINA) using control theory approach.

CHAPTER 4

EXPERIMENTAL SETUP

SDN experiments can be performed on many opensource platforms such as mininet, GENI (Global environment for network innovations), ONOS and other platforms. For our research, we used mininet which is a tool to build virtualized network topology.

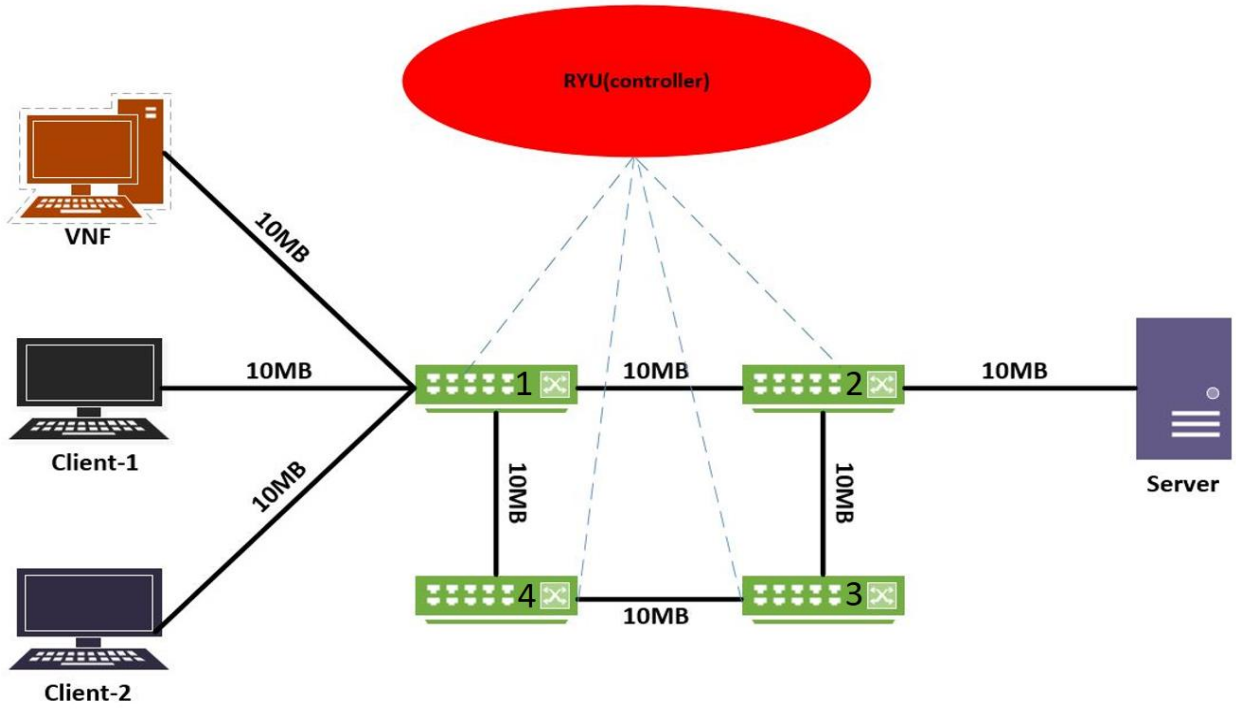

Figure 11: Virtualized Topology

Using mininet, we created a topology shown in Figure 11. Mininet provides functionality for assigning link parameters such as bandwidth and delay. As shown in figure 11, each link has 10MB bandwidth. There are four OVS (OpenFlow virtual switch) switches which are operating on OpenFlow 1.3 version. Ryu controller which has a component-based framework provides a centralized view of the network for managing infrastructure layer of SDN. Ryu controller has a

well-defined southbound API which supports OpenFlow and Netconf. In the topology, there is VNF instance which is hosting SNORT (IDS).

To test adaptive routing, we will run DASH video application from client-1 to server. As shown in Figure 12, DASH (Dynamic Adaptive Streaming over HTTP) is a process of providing high-quality streaming of media content.

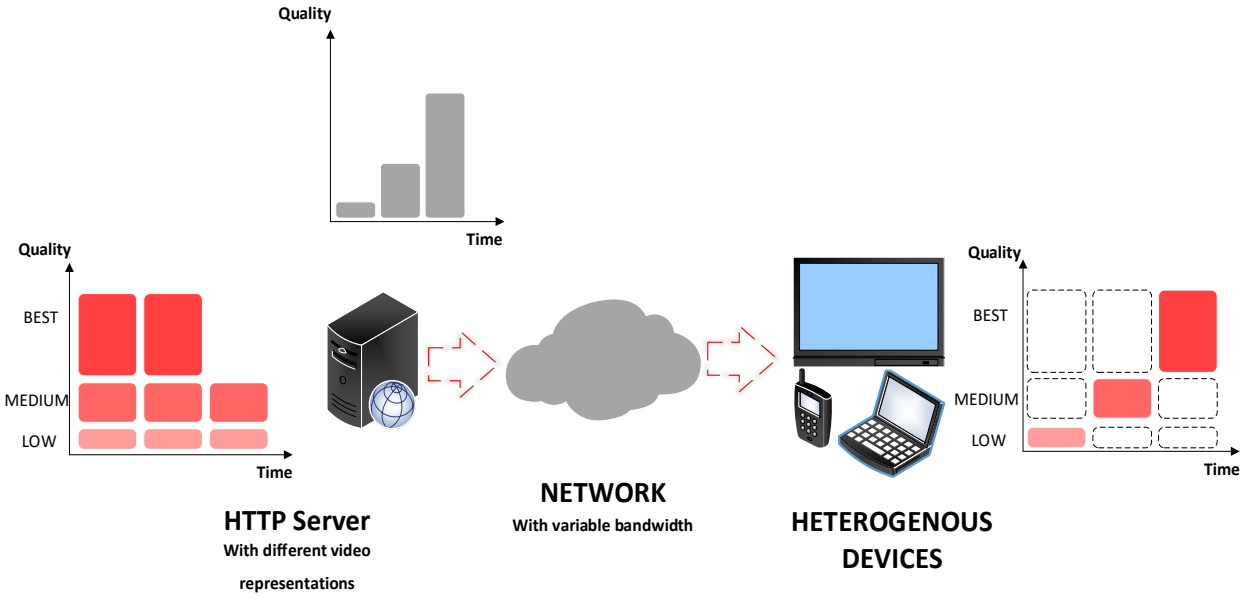

Figure 12: DASH Overview

The HTTP server will host multiple representations of the video. These representations are again divided into smaller segments. Now the client will download subsets of segments depending on the network bandwidth. To distinguish the outcomes of SDN based adaptive routing and traditional routing we monitor the DASH video bitrate. Simultaneously, we have generated extra traffic using Iperf from client-2 to a server. Iperf is a tool to calculate link bandwidth between hosts. It is also used to produce traffic between hosts for research purposes. In next section, we will discuss the different testbed scenarios which will provide a better understanding regarding adaptive routing using control theory.

4.1 Scenario-I

For scenario-I, we have not integrated SDN with NFV which provides a better setup for traditional routing. In figure 13, we see an SDN network with a controller and many networking devices where both are distinguished by control layer and forwarding layer. Here SDN(Ryu) controller which gets a centralized network view uses LLDP (link layer discovery protocol) to calculate the shortest path between nodes. After calculating the shortest path, a controller will generate flows which are installed in OVS switches using OpenFlow protocol. In figure 13, we could see client-1 will take two hops (OVS-1 and OVS-2) to reach a server. Similarly, client-2 will take two hops (OVS-1 and OVS-2) to reach a server. Therefore, Client-1 and Clinet-2 will use same path to reach server. Now we run a DASH client server application from client-1 to server. Simultaneously, by injecting traffic using Iperf from client-2 to a server will force Client-1 and Client-2 traffic to share path.

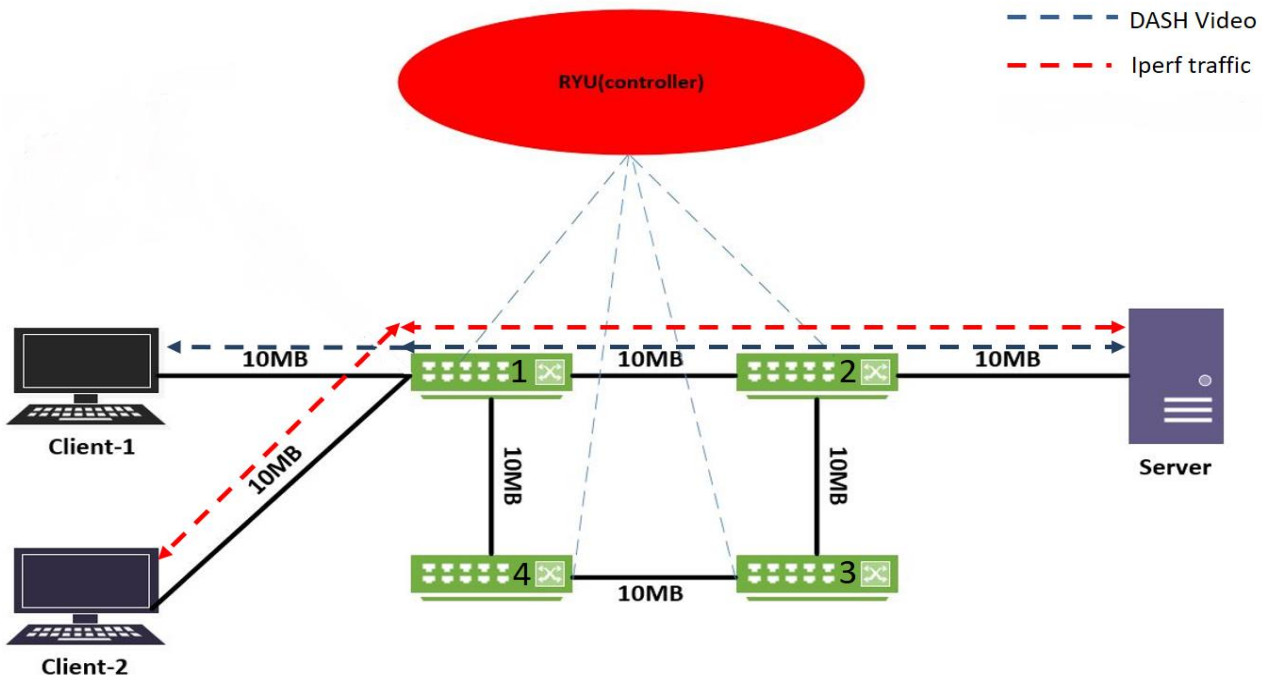

Figure 13: Virtualized Topology(Scenario-I)

As shown in figure 13, blue dotted lines indicate DASH video traffic and red dotted lines indicate Iperf traffic. DASH HTTP server will transfer video segments on the basis of the estimated link bandwidth. Therefore, Iperf traffic will impact the bitrate of every video segment because of the shared link (OVS-1 to OVS-2). Iperf traffic is injected variably from 1MB up to 10MB (link b/w) in a successive manner for the same video. DASH client provides a downloading bitrate for every segment which is monitored and recorded. So, we examined from Scenario-I that a shared link will result in high bandwidth utilization which will eventually decrease the bitrate of DASH video segments. To overcome this problem, adaptive routing is implemented in Scenario-II.

4.2 Scenario-II

For scenario-II, we aimed to implement adaptive routing specifically for DASH video streaming. As shown in Figure 14, integration of SDN and NFV is established on a virtualized platform. We added a VNF instance which provides an IDS service to generate alerts. In this experiment openness of SDN plays a vital role. RINA API is created to manage NFV and transfer all the alerts generated by IDS (VNF) to the controller. As mentioned in chapter 1, we are implementing control theoretic approach to accomplish an automated network. In this section, we will discuss various components and methods for required operation.

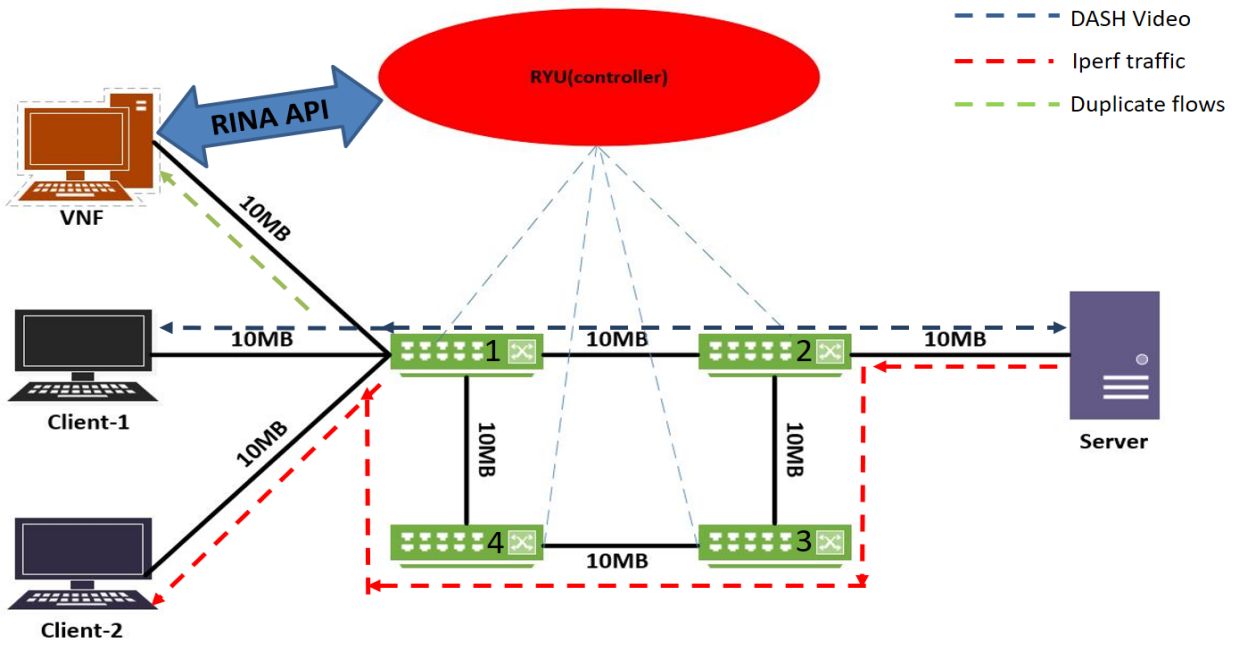

Figure 14: SDN and NFV Topology (Scenario-II)

- **Start RYU controller:** Ryu is executing the OpenFlow13 protocol, i.e., southbound API, to control OVS switches (network devices). It is programmed to configure shortest path routing algorithm in all the OVS switches. After executing shortest path routing, it will wait for IDS information provided by VNF. Depending on the data retrieved from the VNF it will decide to implement adaptive routing on forwarding layer.
- **Start IDS**: SNORT (IDS) is installed on VNF where an alert rule is specified for detection of DASH video packets. As explained in chapter 3, snort rules can be created by specifying IP address, port number, and protocol. Alerts are generated after detection and stored on VNF.
- **Start RINA service:** RINA API processes are installed on VNF and controller machines. As explained in chapter 3, Common distributed application protocol (CDAP) is executed which creates a communication channel between VNF and controller. Now, VNF will start

a service where IDS information (ALERT file) is read and published. The controller will launch a service where the published event is retrieved by creating a subscribe event. Thus, using RINA's sub/pub API communication layer is generated. Every 1 sec, VNF information is updated and transferred to the controller.

- **Start DASH application:** DASH video application is hosted on client-1 and server. DASH server is hosted on port 8001. DASH application starts transferring video segments from client-1 to server. In figure 14, blue dotted lines represent DASH video packets.
- **Generate traffic:** Iperf is a tool to calculate available link bandwidth. It can also be used to generate traffic between nodes. Iperf service is started simultaneously with DASH video application. Iperf version 3 allows us to inject traffic by setting a target bandwidth in Mbit/sec. In our testbed, link b/w is 10MB therefore we generated Iperf TCP traffic which ranges from 1MB up to 10MB and recorded ten sets of video bitrate results for the same video.
- **Workflow of scenario-II:** After starting all the services mentioned above, Ryu controller executes shortest path routing algorithm using LLDP protocol. With the help of SDN controller, specific flows are added in OVS-1 which will duplicate all the traffic coming to client-1 interface and direct it to VNF(IDS) interface. In figure 14, we see a green dotted line going towards VNF which represents duplicate flows of client-1. Snort (VNF) is configured in such a way that it will generate alert file whenever client-1 receives incoming packets from a source port of DASH HTTP server. Therefore, all the incoming packets to DASH client approaching from DASH HTTP server who have a specific port number and source IP (10.0.0.3:8001) are detected by IDS. Detection of video segments is done based on the port number of DASH server. Iperf is running on client-2, but it is using the same

server. Classifying services based on their source port numbers was the only way to distinguish packets which had the same source IP address. As shown in figure 14, VNF transfers alert file created by snort process to the controller through RINA API. At this instant, the importance of northbound API in SDN can be understood. Ryu controller continuously monitors the alert file and performs conditional routing based on the received information. Figure 15, explains the programming flow of the Ryu controller in detail.

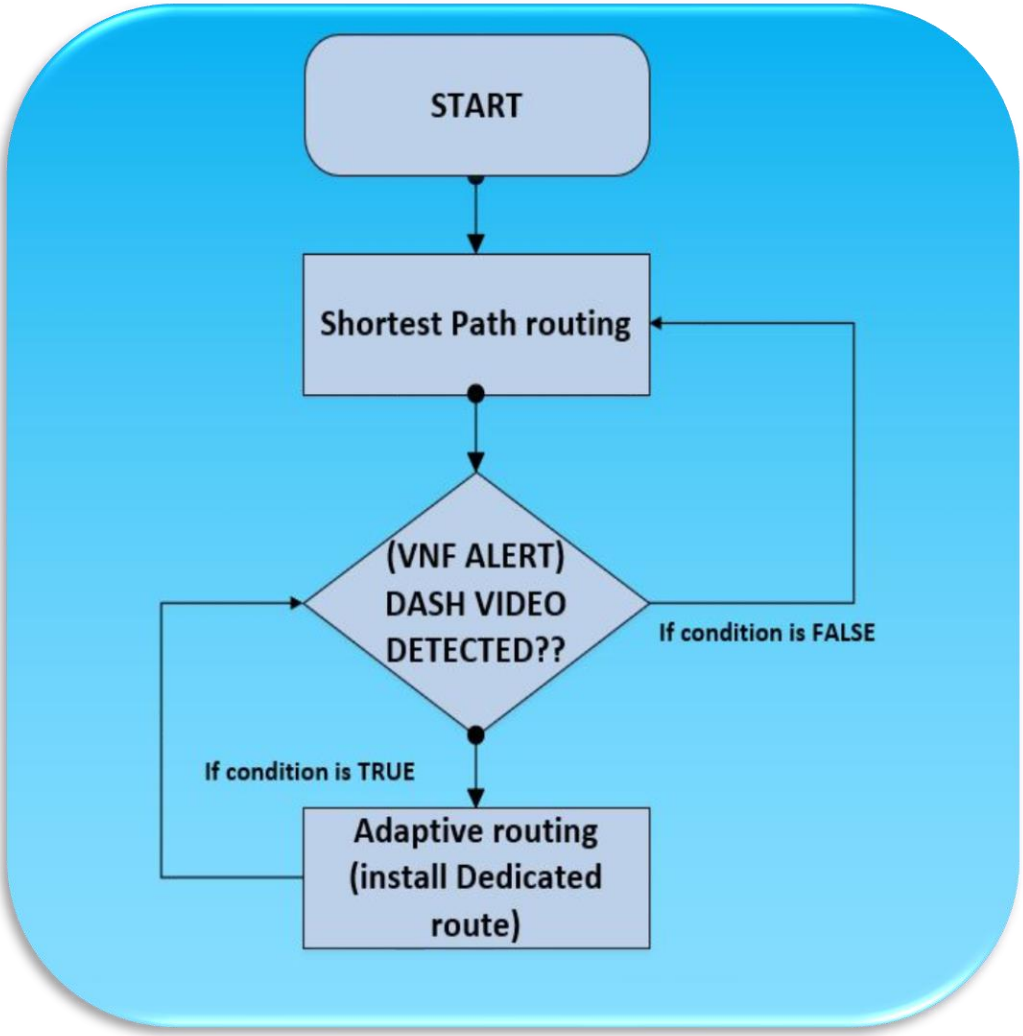

Figure 15: Controller Flow Chart

In figure 15, we see the entire flow of the controller which is programmed using OpenFlow13 protocol. Firstly, it will be implementing shortest path algorithm and then after receiving IDS information execution of adaptive routing takes place. Certain conditions are matched to achieve adaptive routing in the network. In the alert file, timestamp information and an alert message are read by the Ryu controller. The timestamp will indicate the age of alerts. After matching these conditions, source to destination (IP/Port) information provided by the alert file is read. By using this information, shortest path to reach client-1 which is OVS-2 to OVS-1 are masked with TCP source port. Therefore, only packets with TCP source port number of DASH service will use that dedicated route. An alternate route for Iperf traffic is also installed by setting it on a low priority. For the alternative Iperf path, it takes a longer route of four hops. Figure 14 explains the final detail scenario of adaptive routing. The red dotted line in figure 14 represents Iperf traffic which is using alternate path and give priority to DASH video segments. By executing adaptive routing, the throughput of a specific application especially multimedia communication can be improved. After performing the above experiment, both scenarios produced ten sets of video segment bitrates each. These ten sets of video segment bitrates are classified in terms of Iperf bandwidth which ranges from 1MB to 10MB. In next chapter, we analyzed the results obtained through our approach.

CHAPTER 5

RESULTS

Before explaining the results section, we describe the hardware specifications of the physical machine where the virtualized topology (testbed) is designed. The following table illustrates hardware specification used for the experiment:

Table 1: Hardware Specifications

Specification	**SDN network (8 nodes)**
Processor	Intel(R) Core (TM i5-2410M CPU 2.30 GHz)
Memory	8GB
Operating system	Ubuntu 14.04.3 LTS

This chapter includes results obtained by performing experiments explained in chapter 4. The results obtained are classified into three sections as follows:

- **Section 1:** In this segment, we compare results obtained from two scenarios which we explained in chapter 4. We recorded all the video segment bitrate of a video file running on DASH application. Every time variable Iperf traffic from 1MB to 10MB is generated and injected into the network to test a change in video bitrate. To analyze data graphically, we calculated average bitrate of the video for every single targeted bandwidth level.

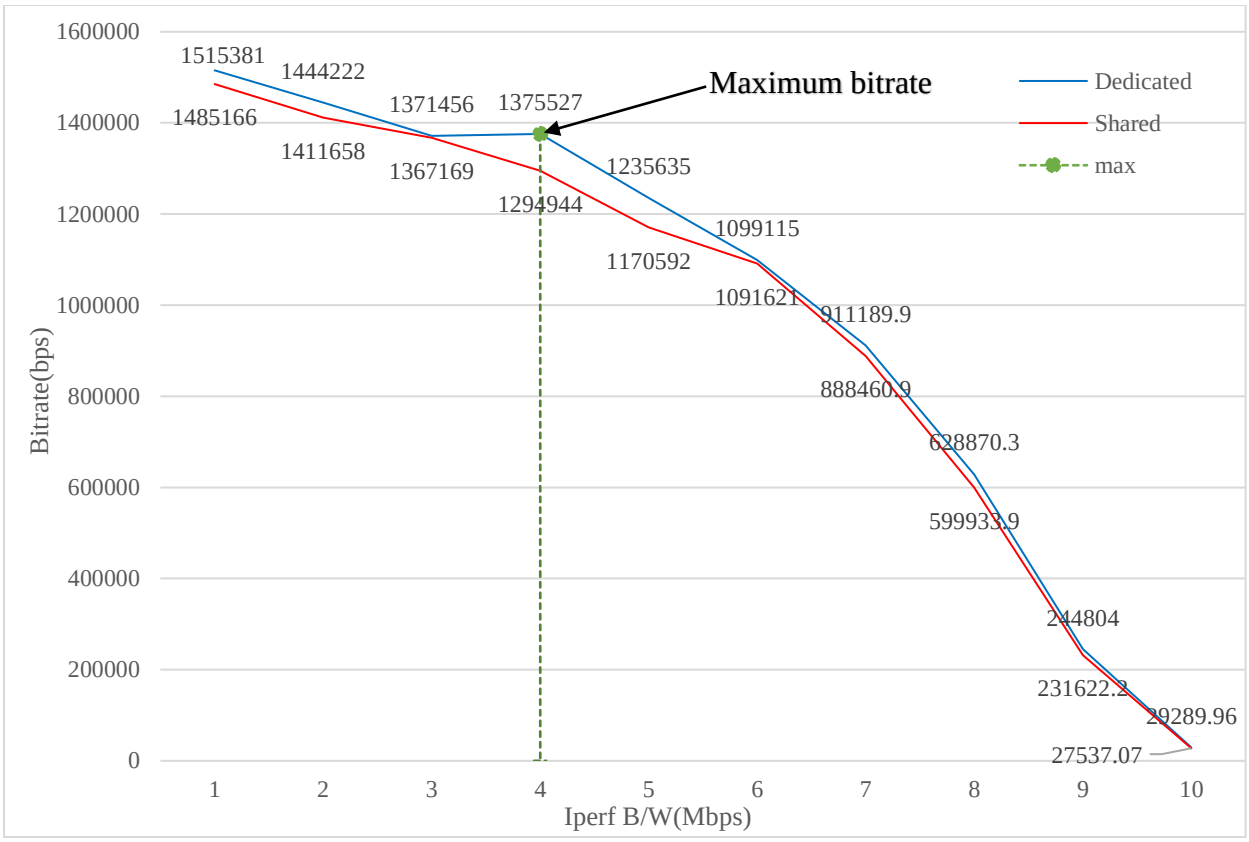

Figure 16: Shared vs Dedicated Routing

- **Section 2:** As shown in Figure 16, the maximum change in bitrate value between shared route (scenario-I) and dedicated route (Scenario-II) was observed when injected Iperf bandwidth is 4MB. Following graph in this section illustrates the graphical representation between No. of segments(video) and bitrate achieved when 4MB external traffic was injected into the network:

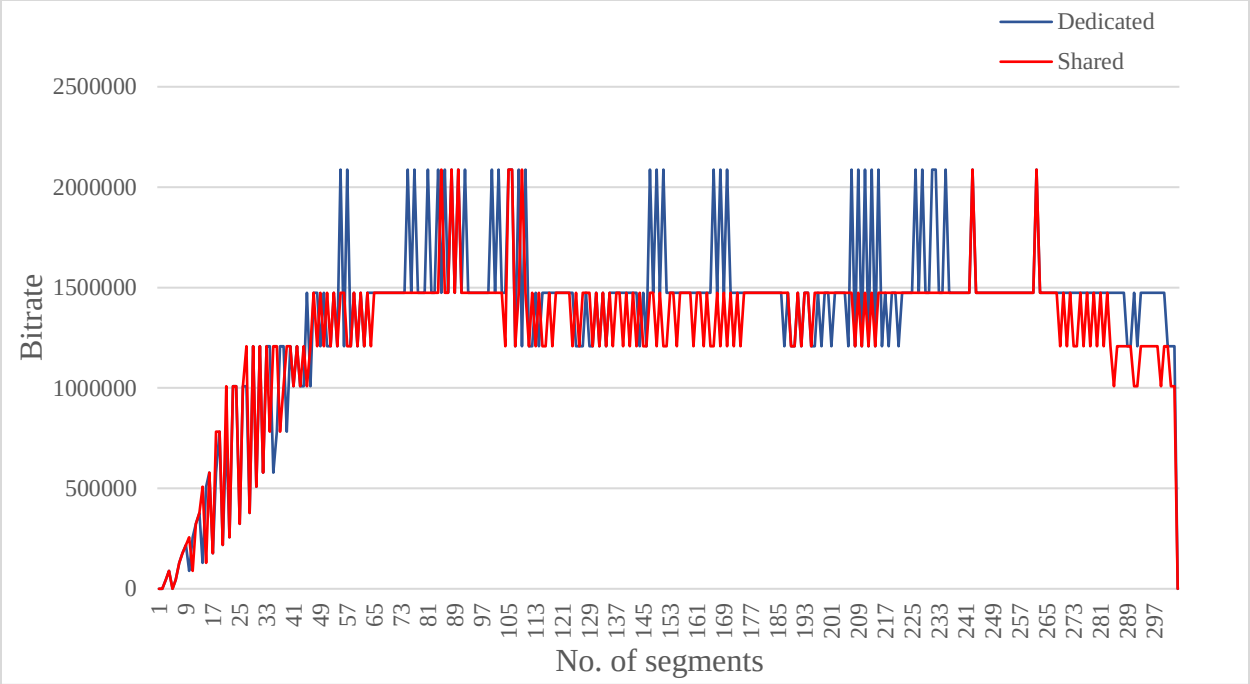

Figure 17: Bitrate vs No. of Segments (With 4MB traffic)

In figure 17, we observe the change in bitrate for both shared path and dedicated path. These results from section-I and section-II show that dedicated route which provides more bandwidth than shared path; transfers video segments at a higher rate. Therefore, the importance of adaptive routing is noticed from these observations.

- **Section 3:** In this segment, we plotted a CDF graph which provides a better perception for comparing throughputs of the dedicated and shared route. In figure 18, we analyze CDF plots for two different performances. According to the graph in figure 18, almost 15% of the segments have bitrate from 1Mbps to 1.5 Mbps.

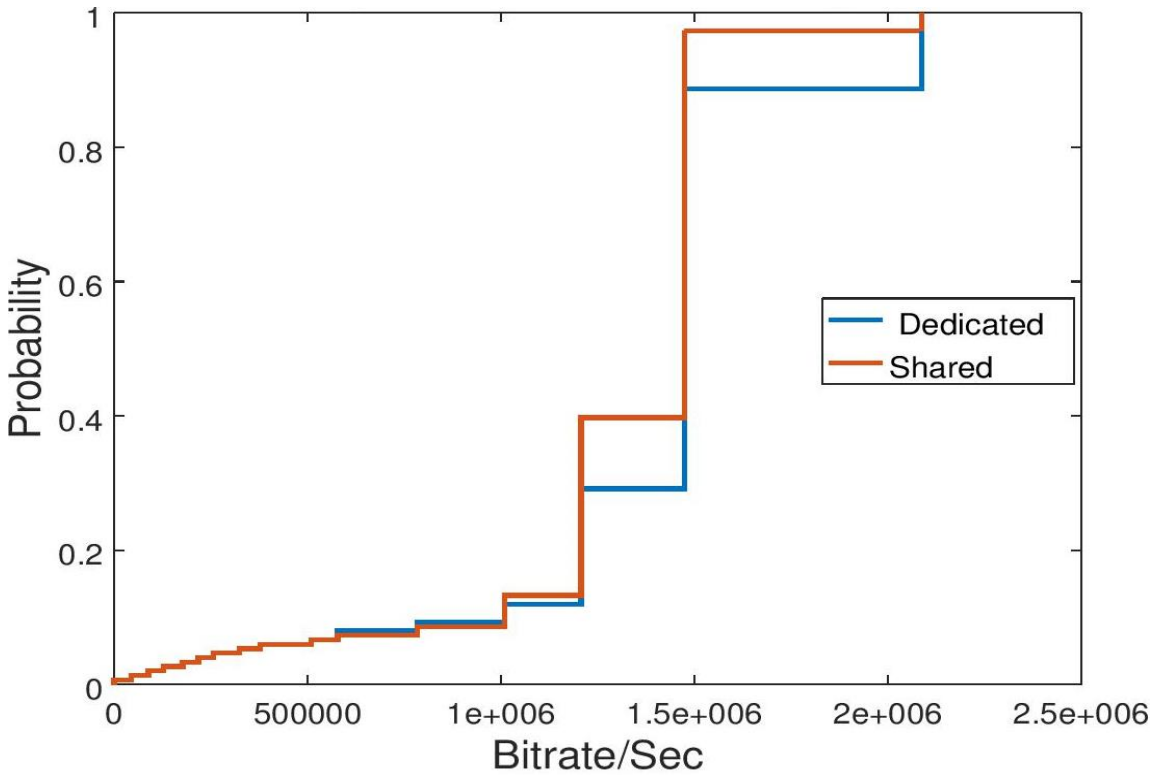

Fig 18: CDF Plot of DASH Video Throughput

Observing the above graph, 40% of video segments which use shared path have bitrate less than 1.3Mbps whereas 35% of video segments which use dedicated path have bitrates less than 1.3Mbps. Also at higher bitrates, we observe that almost 95% of segments for the shared path cannot achieve bitrates more than 1.5 Mbps, the rate is 85% for dedicated path. Therefore 10% of all the segments which used dedicated path have higher bitrate (more than 1.5Mbps) than a shared path.

CHAPTER 6

CONCLUSION

In this thesis, we discussed the challenges faced in traditional networking and how we can resolve it by implementing SDN and NFV. We proposed the idea of Adaptive routing which can be used for many multimedia applications where the primary objective is to provide Quality of Service (QoS). By observing the results, we deduce that implementation of adaptive routing can create an optimal dedicated path for a specific application or service. It also achieves segments of higher bitrate approximately 10% more than that in a shared path. Therefore, overall DASH video throughput will be enhanced by this technique.

Additionally, we saw that using RINA as a network management tool and northbound API allowed us to implement control theoretic approach. Currently, almost all SDN controllers use REST service to build communication between a controller and network applications. REST services have drawbacks which correctly cannot be used as SDN northbound API. REST services do not allow multiple clients with different representation (such as XML, JSON) to receive information from single server. Various complexities are involved for tackling this issue using HTTP 1.1 content negotiation mechanism. Therefore, we opted for RINA which resolves these problems and allows us to implement multiple services using IPC.

Execution of SDN enabled network with the deployment of NFV in the real network will lower Opex and Capex. Moreover, implementing adaptive routing in the real network will be beneficial to all multimedia service providers. Our approach has some limitations while implementing it into the real network. In this thesis, we did not calculate optimal path considering all the QoS parameters. QoS constraints such as delay, CPU load, error rate, etc., can be considered

to achieve optimal route. We have implemented dedicated route for a specific topology, in the real network we must implement it based on an algorithm; Alternate path routing algorithm can be calculated which will direct low priority traffic to an alternate path instead of the shortest optimal path. We believe implementation of Alternate routing with the help of an algorithm will gradually increase throughput in multimedia communication. Lastly, we are using layer three and layer four information to detect segments of DASH video application. In our approach, we explicitly configure IDS to identify packets from coming from DASH server port number. To avoid this, we need to install a mechanism called deep packet inspection in our IDS. It will allow us to detect DASH video segments based on its payload. We conclude by affirming limitations of our work and state the required future work to resolve these challenges:

- Calculating all the QoS constraints like delay, bandwidth, priority, CPU load for optimal path
- Implementing alternate path algorithm for unwanted traffic
- Initializing DPI mode (deep packet inspection) in IDS

REFERENCE LIST

[1] Nabeel Akhtar, Ibrahim Matta, and Yuefeng Wang. Managing NFV using SDN and control theory. In *2016 IEEE/IFIP Network Operations and Management Symposium* (2016), IEEE, pp. 1113-1118.

[2] Parikshit Juluri and Deep Medhi. Cache'n DASH: Efficient Caching for DASH. In *SIGCOMM '15 Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (2015), ACM, pp. 599-600.

[3] Iris Bueno, José Ignacio Aznar, Eduard Escalona, Jordi Ferrer and Joan Antoni García-Espín. An OpenNaaS based SDN Framework for Dynamic QoS Control. In *2013 IEEE SDN for Future Networks and Services* (2013), IEEE, pp. 1-7.

[4] Slavica Tomovic, Neeli Prasad and Igor Radusinovic. SDN Control Framework For QoS Provisioning. In *IEEE 2014 22nd Telecommunications Forum TELFOR* (2014), IEEE, pp. 111-114.

[5] Niels L. M. van Adrichem, Christian Doerr and Fernando A. Kuipers. OpenNetMon: Network Monitoring in OpenFlow Software-Defined Networks. In *2014 IEEE Network Operations and Management Symposium* (2014), IEEE, pp. 1-8.

[6] G. Xilouris, E. Trouva, F. Lobillo, J. M. Soares, J. Carapinha, M. J. McGrath, G. Gardikis, P. Paglierani, E. Pallis, L. Zuccaro, Y. Rebahi and A. Kourtis. T-NOVA: A Marketplace for Virtualized Network Functions. In *2014 European Conference on Networks and Communications* (2014), IEEE, pp. 1-5.

[7] Jon Matias, Jokin Garay, Nerea Toledo, Juanjo Unzilla, and Eduardo Jacob, Toward An SDN-Enabled NFV Architecture. In *IEEE Communications Magazine* (2015), vol.53, IEEE, pp. 187-193.

[8] Raj Jain and Subharthi Paul. Network Virtualization and Software Defined Networking for Cloud Computing: a survey. In *IEEE Communications Magazine* (2013), vol.51, IEEE, pp. 24-31.

[9] Minh Pham and Doan B Hoang. SDN Applications - The Intent-Based Northbound Interface Realisation for Extended Applications. In *IEEE NetSoft Conference and Workshops* (2016), IEEE, pp. 372-377.

[10] Wei Zhou, Li Li, Min Luo and Wu Chou. REST API Design Patterns for SDN Northbound API. In 28th International Conference on Advanced Information Networking and Applications Workshops (2014), IEEE, pp. 358-365.

[11] Nicolas Herbaut, Daniel Negru, Damien Magoni and Pantelis A. Frangoudis. Deploying a Content Delivery Service Function Chain on an SDN-NFV Operator Infrastructure. In International Conference on Telecommunications and Multimedia (2016), IEEE, pp. 1-7.

[12] Josep Batalle, Jordi Ferrer Riera, Eduard Escalona and Joan A. Garcia-Espin. On the Implementation of NFV over an OpenFlow Infrastructure: Routing Function Virtualization. In *2013 IEEE SDN for Future Networks and Services* (2013), IEEE, pp. 1-6.

[13] Yao-Yu Yang, Chao-Tung Yang, Shuo-Tsung Chen, Wei-Hsun Cheng and Fuu-Cheng Jiang. Implementation of Network Traffic Monitor System with SDN. In

2015 IEEE 39th Annual Computer Software and Applications Conference (2015), IEEE, pp. 631-634.

[14] Laxmana Rao Battula. Network Security Function Virtualization(NSFV) towards Cloud Computing with NFV Over Openflow infrastructure: Challenges and Novel Approaches. In *2014 International Conference on Advances in Computing, Communications and Informatics* (2014), IEEE, pp. 1622-1628.

VITA

Tejas Balkrishna Parab was born on June 2, 1992 in Mumbai, Maharashtra, India. He graduated from University of Mumbai in 2014 with B.E. degree in Electronics and Telecommunication.