July 2018

AN EVALUATION OF SDN AND NFV SUPPORT FOR PARALLEL, ALTERNATIVE PROTOCOL STACK OPERATIONS IN FUTURE INTERNETS

Bhushan Suresh

AN EVALUATION OF SDN AND NFV SUPPORT FOR PARALLEL, ALTERNATIVE PROTOCOL STACK OPERATIONS IN FUTURE INTERNETS

A Thesis Presented

by

BHUSHAN SURESH

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

May 2018

Electrical and Computer Engineering

AN EVALUATION OF SDN AND NFV SUPPORT FOR PARALLEL, ALTERNATIVE PROTOCOL STACK OPERATIONS IN FUTURE INTERNETS

A Thesis Presented

by

BHUSHAN SURESH

Approved as to style and content by:

Michael Zink, Chair

David Irwin, Member

Lixin Gao, Member

C.V.Hollot, Department Head
Electrical and Computer Engineering

ACKNOWLEDGEMENTS

I would like to express the deepest appreciation and thanks to my advisor, Professor Michael Zink. His knowledge and leadership has helped me expand my horizon in the field of computer networking. This thesis has been made possible only through his genius and guidance. I will forever be indebted to his support and guidance in my future career. A sincere thanks to Professor David Irwin and Professor Lixin Gao for their invaluable advice and feedback on my research.

A special thanks to Divyashri Bhat for her critical inputs and constructive feedback at every stage of my research. I would also like to acknowledge the CloudLab operations group, in particular, Michael Blodgett, David Johnson and Leigh Stoller. The timely delivery of our work can be attributed to their ability to maintain resources, address our concerns, and resolve issues.

I am grateful to everyone in the ECE department for the knowledge, experience and resources they have shared with me.

Finally, I would like to thank my family, in particular, my loving wife, Shweta for her continued encouragement, patience, and understanding through this effort.

ABSTRACT

AN EVALUATION OF SDN AND NFV SUPPORT FOR PARALLEL, ALTERNATIVE PROTOCOL STACK OPERATIONS IN FUTURE INTERNETS

MAY 2018

BHUSHAN SURESH

M.S, UNIVERSITY OF MASSACHUSETTS, AMHERST

Virtualization on top of high-performance servers has enabled the virtualization of network functions like caching, deep packet inspection, etc. Such Network Function Virtualization (NFV) is used to dynamically adapt to changes in network traffic and application popularity. We demonstrate how the combination of Software Defined Networking (SDN) and NFV can support the parallel operation of different Internet architectures on top of the same physical hardware. We introduce our architecture for this approach in an actual test setup, using CloudLab resources. We start of our evaluation in a small setup where we evaluate the feasibility of the SDN and NFV architecture and incrementally increase the complexity of the setup to run a live video streaming application. We use two vastly different protocol stacks, namely TCP/IP and NDN to demonstrate the capability of our approach. The evaluation of our approach shows that it introduces a new level of flexibility when it comes to operation of different Internet architectures on top of the same physical network and with this flexibility provides the ability to switch between the two protocol stacks depending on the application.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Introduction

For the last forty years the Internet has been based on roughly the same protocol stack, centered around TCP/IP. While there have been modifications to TCP/IP (a series of TCP flavors and a slow but constant change from IPv4 to v6) no completely new protocol stack has seen wide-spread use.

We are currently at an infliction point where several future Internet architectures have been proposed which caused some traction in research and commercial projects alike. One example of such a new architecture is Named Data Networking (NDN) [32], which is an instantiation of the Information Centric Networking (ICN) [13] paradigm. NDN is one among the proposed NSF Future Internet Architectures which also includes eXpressive Internet Architecture (XIA) [4] and MobilityFirst [12] . It introduces a new protocol stack that is incompatible with the traditional one based on TCP/IP. NDN introduces a flexible transport layer that does not conform to the end-to-end connection paradigm of TCP/IP. For this reason, new Internet architectures such as NDN have always been exclusive alternatives. Cisco, one of the leading manufacturers of networking equipment, has recently launched a hybrid ICN solution, *hICN* [1], which is an incremental ICN solution that integrates with existing IP networks. It remains to be seen if *hICN* will be widely adopted.

The goal of our work is to investigate if new technologies like Software Defined Networking (SDN) [21] and Network Function Virtualization (NFV) [5] can be used to support the parallel operation of traditional and future Internets on top of the

same physical hardware. In addition, we investigate the performance impact of such an approach.

One fundamental underlying mechanism that is required to support such an approach is "slicing", which allows isolated virtualization of networking and compute resources. In our vision, such slicing will ultimately allow the parallel operation of many Internets, including the traditional Internet, over one physical infrastructure. While we are far from reaching such complete multi-tenancy, we investigate if an SDN/NFV-based approach has the potential to support our vision.

We present results from an investigation of our SDN/NFV-based approach that allows the parallel operation of different network stacks on top of the same physical hardware. We have decided to use IPv4 [11] and NDN [32] because they are vastly different protocols and the latter presents a future Internet architecture. While we use these two network protocols, our approach is designed to support other transport protocols like IPv6, IPSec, IPx etc.

In the current IP-based Internet, data are requested by providing address information (e.g., the IP address of a server). NDN is different since it requests information based on names and not end-point addresses. For NDN, data does not necessarily have to be served from its original source but can inherently be served from the content store of a router along one of the paths to the server.[1] To achieve this behavior NDN clients initiate an Interest message which identifies the requested content. At each router along the path to the custodian, the pending interest table (PIT) is checked and, if a similar entry exists, the Interest is not further forwarded. It is also checked if the content is already in the router's content store (CS) and if so, the content is directly served from the router.

[1]This is called "custodian" in NDN jargon.

We begin by setting up smaller test topologies on top of which we run NDN and IP applications to evaluate the feasibility of the SDN/NFV architecture and their performance when run in parallel. With the insight we gained from our initial setup, we extend our evaluation to use the example of an adaptive bitrate (ABR) live streaming scenario on a larger topology to demonstrate that our approach supports the parallel operation of IP and NDN. With the aid of popular Quality-of-Experience (QoE) metrics, we evaluate the performance of our approach. Evaluation results demonstrate the flexibility of our approach and indicate how SDN-based traffic engineering can be used to efficiently share resources between IP and NDN.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Software Defined Networks(SDN)

Architectures that combine the deep programmibility of SDN with the flexibility of NFV have gained significant traction in recent years. In [26], Sama et al. present a solution that combines the benefits of SDN and NFV for mobile networks. Juniper's OpenContrail [2] is a popular commercial implementation of IPv4 based NFV using SDN. RouteFlow [22] is another approach that provides a design of NFV using SDN in which the authors evaluate the performance of a Quagga routing engine using OpenFlow. However, these works do not consider the performance implication of co-locating IP and non-IP protocols on the same hardware. In [16], Kanada et al. investigate the performance of a virtualization platform that incorporates both an IP-based and an experimental non-IP protocol on different virtual instances. Unlike our work, they use tunnels for encapsulation of traffic and do not show the benefits of introducing SDN programmability. Flowvisor[25] introduces a virtualization technique that slicing the network through a controller using OpenFlow. With the OpenFlow controller they isolate the topology based on the incoming traffic, isolate CPU utilization by controlling the new flow messages and controller requests. Since this approach provides virtualization through the type of incoming traffic, running two different streaming(IP-based and NDN-based as with ours) applications simultaneously would not be possible.

2.2 OpenFlow Proactive versus Reactive flow rules

OpenFlow [21] is the most well know implementations of SDN in the control plane today, it is been widely accepted and standardized by the Open Networking Foundation. The flow rules that are installed by OpenFlow are of two types: *Reactive flow rules* and *Proactive flow rules*. Reactive flow rules are installed when the switch receives a packet. The OpenFlow agent on the switch checks the flow tables if a flow rule is already in place for this packet if not an $OFPpacket-in$ request is sent to the controller. Based on the OpenFlow controller instructions a new flow rule is created on the flow table. Thus, Reactive flow rules require the controller intervention each time a packet for which no matching on the switch exists. Proactive flow rules works by populating the flow table ahead of time instead of having the controller react to every packet.

2.3 Named Data networks (NDN)

As mentioned in the Sect. 1.1, NDN routing is based on names and not end point addresses. As shown in Fig. 2.1, NDN differs from IP by the fact that it employs content delivery in its network layer. The two protocols are similar but *content chunks* replace IP thus making NDN content centric and IP host centric. NDN caches data on all the intermediate nodes along the path to the requesting user. An NDN node maintains three data structures(as shown in Fig. 2.2 [10]) which are FIB (Forwarding Information Base), PIT (Pending Interest Table) and Content Store (CS). When a client requests for a content though an *Interest Packet*, the CS is checked to see if the content is already cached. If the cache has the requested content, the content is sent back to the client as a *Data Packet* following the backward path of the incoming packet. If the requested content is not present in the CS, the PIT is checked to see if there are any outstanding interest for the same content. If not, an

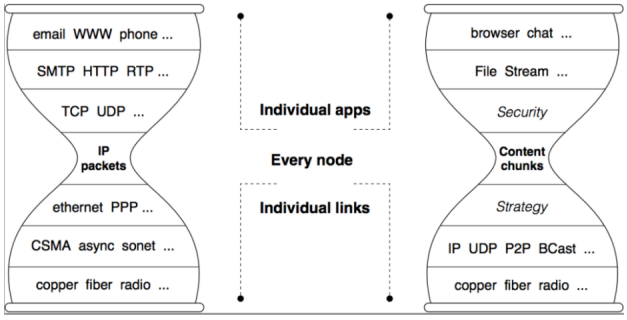


Figure 2.1: IP and NDN Protocol stack comparison

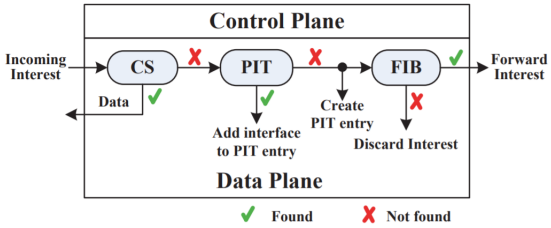


Figure 2.2: Packet forwarding inside an NDN router

entry is created in the PIT and the interest is forwarded out of an interface based on the information in the FIB.

A server-side performance evaluation of NDN has been performed by Marchal et al. in [19]. It is known that NDN in general was designed to help reduce congestion at Internet scale by using an information centric architecture, and by providing each node with a caching capability. Marchal et al. shed light on how NDN's security feature comes at a high cost, as it relies heavily on asymmetric cryptography which affects server performance when NDN data is generated. The authors introduce a new tool, *NDNperf*, to perform an evaluation to show that the time taken for creating NDN packets is a major bottleneck for application performance. They propose and evaluate pragmatic solutions to improve the performance of server-side NDN Data generation leading to significant gains. Our work is different, since we do not focus

on performance issues introduced by a specific protocol or architecture but rather the performance impairment introduced by NFV.

2.4 Network Function Virtualization(NFV)

One of the major goals of NFV is to leverage standard virtualization technology to consolidate network functionalities onto high-end servers [5]. Our architecture combines NFV and SDN since it introduces a substantial flexibility on layer 2 (SDN) and higher layer (3-5) functionality (NFV) as we will illustrate in more detail in Sect. 5.4. We focus on layer 3 NFV, since it allows the parallel operation of vastly different Internet architectures on the same physical network. In order to demonstrate the feasibility of such an approach, we create two virtual machines (VM) which support the traditional TCP/IP protocol and the more recent, state-of-the-art NDN approach. In [28], Sardara et al. introduce a virtualized Linux container platform to incrementally develop and test ICN-based applications. This platform is used by Samain et al. in [27] to compare the performance of ABR video streaming over NDN with current state-of-the-art ABR streaming algorithms over TCP/IP. Although our evaluations include a similar comparison, we additionally provide insight into the effect of introducing SDN traffic engineering in an NFV testbed setup and analyze the performance impact of having multiple transport protocols in such systems. VINI [3], is an approach that lets researchers run a virtual Internet, consisting of a data plane and a control plane, on PlanetLab. On the control plane the architecture provides topology disovery service and a protocol routing suite through XORP[20]. On the data plane they provide a forwarding engine(CLICK [6]) for the packets carried by the overlay (an overlay router). Packets are routed between the data plane and control plane through UDPTunnels. This makes the approach IP-based which would not allow the processing of NDN packets. The paper does not refer to any technique that provides running multiple instances. With our centralized controller we have the ability to

route both IP and NDN packets and also to bring down (or up) virtual instances. The routing protocol that they use is XORP. Quagga provides us the same ability to run different routing protocols and can be run on our VMs. Another virtualization approach by VENICE [14] create a simulation/emulation co-design, employs time dilation for flexibility in emulating target resource properties using the available physical resources, and optimizes resource utilization using dynamic resource reallocation to simulation/emulation elements and dynamic time dilation. To optimize network communication, VENICE uses TUN/TAP devices i.e through IP tunneling. This approach again being IP-based will not enable processing of NDN traffic. VNET/P [17] employs publicly available Palacios Virtual Machine Monitor (VMM) virtualization technique to create a tool on top of the VMM to improve performance of virtual machines hosting HPC applications. The virtualization approach to route packets to a destination on a different machine is to encapsulate the packet and send it via a corresponding UDP or TCP socket. This method would allow us to evaluate NDN's routing performance since routing with this techinque is IP-based.

2.5 CloudLab

CloudLab [24] is a geographically distributed testbed for the development, deployment, and validation of cloud-based services. We decided to use CloudLab for the evaluation of our approach, since it offers a larger number of bare metal nodes, which guarantees 100% performance isolation for the compute resources. The CloudLab infrastructure consists of several different racks of varying compute and storage sizes designed to provide isolated performance and support experiments at-scale. SDN is supported through the deployment of OF switches. CloudLab allows experimenters to reserve bare metal servers on which they can install a variety of either pre-configured or customized OSs. The fact that CloudLab offers bare metal servers and OF switches makes it an ideal testbed for the evaluation of our approach.

CHAPTER 3

ARCHITECTURE

In this section, we present the design of our SDN-NFV based architecture that has the goal to support the parallel operation of different protocol stacks over the same physical hardware. There are two main goals we try to achieve with this architecture. The first one is to create a flexible, software defined, virtualized infrastructure that allows the setup of slices for different protocol stacks. The second goal focuses on dynamic changes of these slices based on load and resource availability.

3.1 Example Scenario

Our approach makes use of three major components. The first one is a *network appliance*, which enables layer 2 switching and has compute capabilities to host VMs on which the protocol stack for layer 3 and above is hosted. The latter enables the appliance to support Network Function Virtualization. For our initial investigations such an appliance is simply realized by a powerful server that supports virtualization well. This server runs OpenVSwitch (OVS) in its host OS and zero, one, or more VMs that run different layer 3 and above protocol stacks. An example for such a network appliance is shown in Fig. 3.1. In its simplest version, this appliance can basically function as a layer 2 switch. This only requires the node to run OVS in the host OS. Running a VM that has, e.g., an IPv4 router installed, turns this node into a router. Adding a second VM that has the NDN Networking Forwarding Daemon (NFD)[1] installed turns this node into an IPv4 *and* NDN router. Additional VMs can

[1] `https://github.com/named-roidata/NFD`

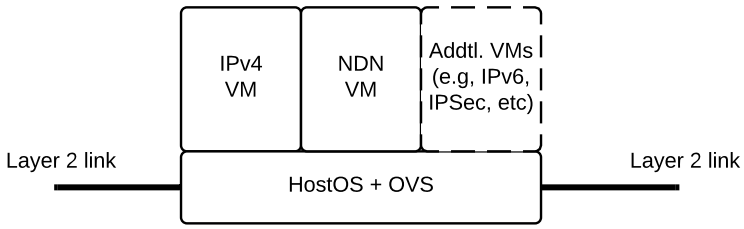


Figure 3.1: Network appliance architecture

be instantiated to support, e.g., IPv6, IPSec, or other protocols. Note, that the main goal of this work is to demonstrate the viability of our approach while investigating its impact on performance. For a fair comparison, we decided to perform our evaluation on a general purpose server but believe that improved performance can be achieved in the future by the combination of SDN switches with compute resources as it has recently been proposed in approaches for Software Defined Exchanges (SDX) [7, 31, 18].

The second component is an *SDN controller* which is connected to the OpenFlow (OF) switches within the appliances. More details of this component are given in Sect. 3.2.

The third component is simply a set of *layer 2 links* that connect the network appliances.

3.2 SDN Support

SDN support allows for a high degree of flexibility in the network configuration and operation. Through SDN a network appliance can assume different functions for different protocol stacks. E.g, the appliance can act as a router for IP traffic and on the other hand simply act as a switch for NDN traffic. For the first instantiation of our approach, SDN functionality is provided by OVS in OF mode. Each OVS is managed by an OF controller. In our case we make use of a centralized controller but the proposed architecture is also designed for distributed controller scenarios. We

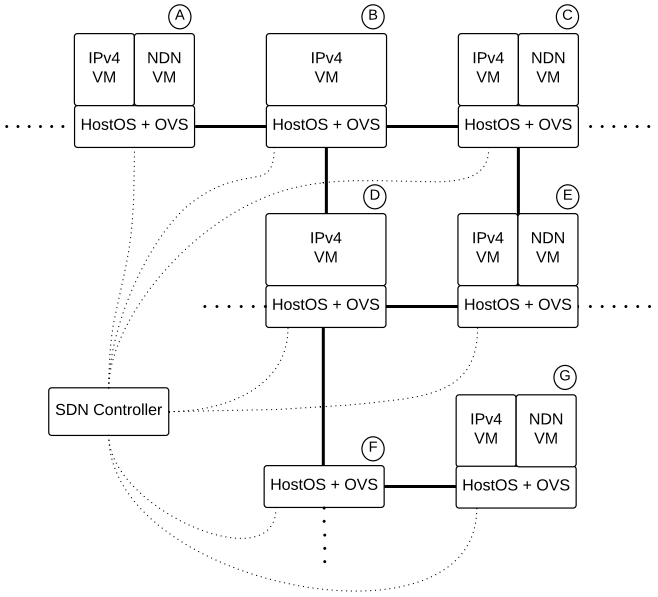
decided to use the Ryu controller[2], since it is open source, well supported, and offers a python API.

3.3 Example Scenarios

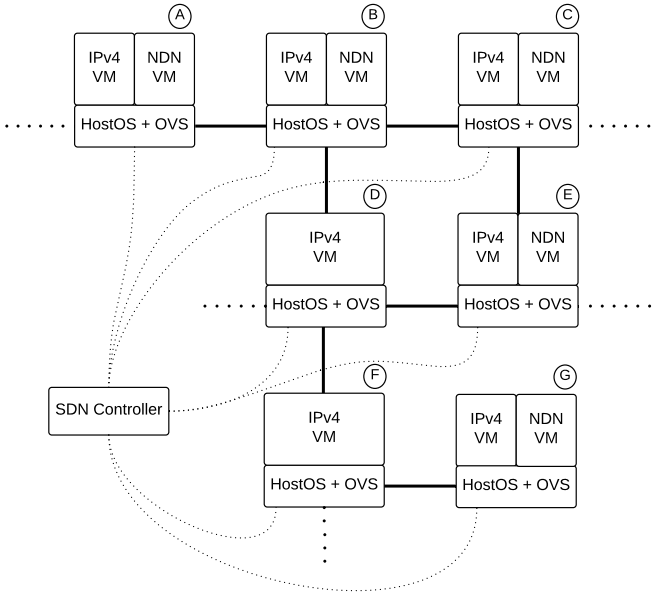
After introducing the individual components of our architecture, we showcase example scenarios to further illustrate the application of our approach in specific network settings. The scenarios is shown in Fig. 3.2, which shows a network that is composed of seven nodes and an SDN controller. Each of the nodes in these scenarios is comprised of a network appliance as shown in Fig. 3.1 and the network links are standard layer 2 links. In addition, each of the network appliances is connected to the centralized SDN controller.

Fig. 3.2(a) shows a scenario where one node acts as a layer 2 switch (F), 2 nodes also support IPv4 (B, D), and 3 nodes support IPv4 and NDN (A, C, E, G). Based on the requirements in this specific scenario switching is only required at node F and no VMs that support higher level protocols are currently active at this node. The forwarding of frames is managed by the SDN controller. For node F, the SDN controller can provide simple learning switch functionality, or any other forwarding algorithm that can be supported by an SDN controller. For nodes B and D, IPv4 routing capability is required in addition to layer 2 switching. This requirement is met by the instantiation of an IPv4 VM on these network appliances. In addition, nodes A and B need to act as layer 2 switches for non-IPv4 based traffic. This functionality is achieved by the SDN controller, which differentiates between IPv4 packets and other layer 3 payload based on the type specified in the Ethernet frame. While frames that contain $EtherType = 0x0800$ are forwarded to the guest IPv4 VM, frames containing

[2] `https://osrg.github.io/ryu/`



(a)



(b)

Figure 3.2: Multi-protocol SDN/NFV approach : (a) Scenario 1 (b) Scenario 2

other EtherType values are forwarded on the layer 2 switch.[3] For nodes A, C, E, and G NDN functionality is required in addition to the IPv4 functionality. This is achieved by instantiating an NDN guest VM on the nodes in addition to the IPv4 VM. In this case, the behavior of the SDN switch also needs to be modified. Here, frames with $EtherType = 0x0800$ (IP) and $EtherType = 0x8624$ (NDN) have to be forwarded to the respective guest VMs. This can be achieved through the appropriate rule set by the OF controller. A detailed description of the SDN functionality and the setup of the networking in the appliances is given in Sect. 4.1.2.

With the second scenario (shown in Fig. 3.2(b)) we illustrate that modifying slices on demand can be easily achieved with this approach. Here, NDN functionality is also required on node B and IPv4 functionality on node F. This can be achieved by simply instantiating the respective VMs on the nodes and implementing the correct rules on the OF controller.

[3]Rules for the forwarding of ARP packets ($EtherType = 0x0806$) are also included in the SDN controller.

CHAPTER 4

EXPERIMENTS

This section describes our experimental setup and scenarios.

4.1 Experimental Setup in CloudLab

In this section, we present the experimental environment we use for the evaluation of our architecture. After introducing the CloudLab testbed, we describe the virtualization and network environment used for the experiments.

4.1.1 Virtualization

We describe the virtualization components used in our setup in the following:

- **Host OS:** For the virtualization on the nodes we make use of the Kernel-based Virtual Machine (KVM) [9]. With KVM Linux becomes a hypervisor on which several guest VMs can run. We have chosen KVM since it supports guest OSs like FreeBSD and NetBSD, which both support a variety of network functions and protocols.

- **OVS:** The Open vSwitch [23] runs on the KVM hypervisor, which supports the bridging between VMs and the physical network the hypervisor is connected to. It is well suited for our approach since it is the default switch in KVM and supports OF.

- **IPv4 VM:** This guest VM runs a vanilla Linux (Ubuntu 14.04) and no additional modification to the OS have to be made since it contains the standard IPv4 protocol suite.

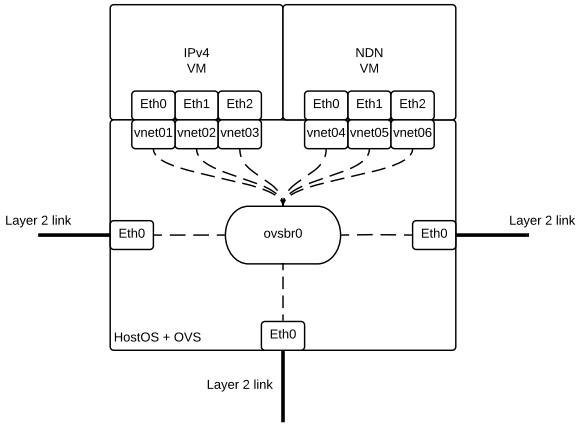


Figure 4.1: Node networking.

- **NDN VM:** This guest VM also runs the same vanilla Linux as the IPv4 VM but NFD had to be installed in addition. NFD is required to enable NDN functionality (e.g., routing, caching, forwarding) at the node. IP routing is disabled on this VM.

4.1.2 Node Internal Networking

Here, we briefly describe the networking setup in the host OS of the nodes that is required to support NFV. Fig. 4.1 shows the network setup for a node interconnected with 3 layer 2 links (example: $Node7$ in Fig 4.4). As mentioned in Sect. 4.1.1, we use OVS to realize the node internal networking and establish the mapping between physical interfaces in the host OS and virtual interfaces in the VMs. E.g., in Fig. 4.1 three physical interfaces have to be mapped to three virtual interfaces per VM. This is realized though a software OF switch, which connects all physical and virtual interfaces. We decided to use an SDN switch instead of a bridge since it allows us to better control the traffic in the node internal virtual network. E.g., IP traffic can be directed to the virtual interfaces of the IP VM instead of broadcasting it to all virtual interfaces. In addition, it will allow for better monitoring (e.g., via OpenFlow port statistics) and traffic isolation.

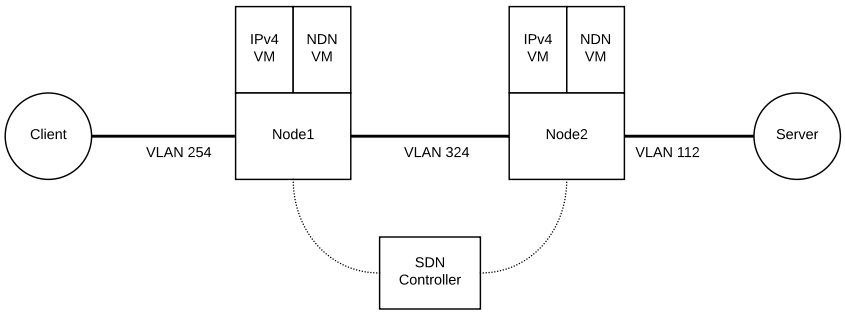


Figure 4.2: Setup 1: Controller to identify NDN and IP packets

4.2 Topology Setup

This section describes our topology setup's we used with each a progression of the previous based on the experimental results.

4.2.1 Setup 1: Identifying NDN and IP packets in the Controller

The aim of this setup was to route NDN and IP packets independently using the flow rules installed by the controller *reactively* (Sect. 2.2). As described in Sect. 5.4, IP packets are identified by ($EtherType = 0x0800$) and NDN packets by ($EtherType = 0x8624$). The CloudLab slice we use for our evaluation is shown in Fig 4.2. It consists of 2 nodes (1-2), a client and a server, and 3 layer 2 links (with each link having a maximum capacity of 10Mbps). In addition, an SDN controller runs on a VM and communicates via the CloudLab control plane with the nodes. Nodes 1 and 2 run both an IP and an NDN VM. The client and the server run both on bare metal servers and generate traffic for the performance evaluation of our architecture. We use $IP\ Ping$ for the generation of IP traffic and $NDNPing$ for the generation of NDN traffic.

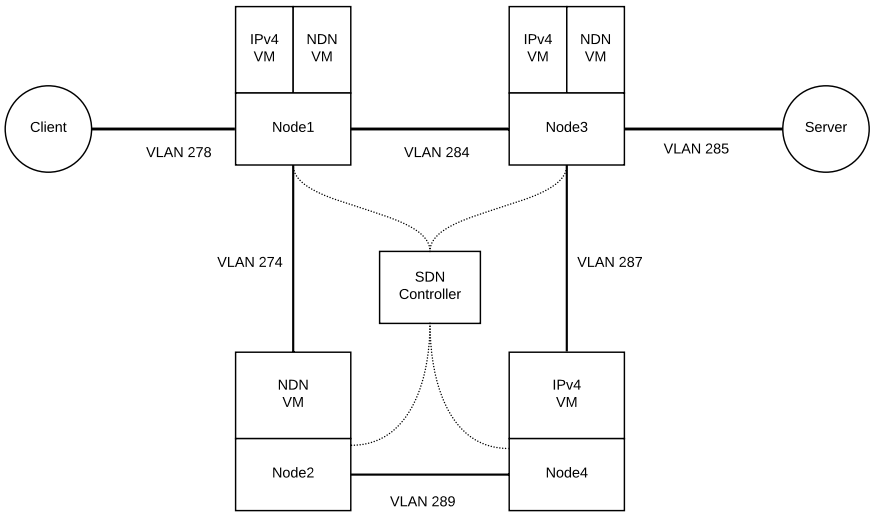


Figure 4.3: Setup 2: Expand the topology to make throughtput measurements

4.2.2 Setup 2: Expand the topology to make throughtput measurements

The CloudLab slice we use for our throughput evaluation is shown in Fig 4.3. It consists of 4 nodes (1-4), a client and a server, and 6 layer 2 links (with each link having a maximum capacity of 10Mbps). Similar to the setup in Fig 4.2 the OF controller runs on a VM and installs flow rules on the nodes *proactively* (Sect. 2.2). Nodes 1 and 3 run both an IP and an NDN VM, while node 2 runs an NDN VM and node 4 runs an IP VM. The client and the server run both on bare metal servers and generate traffic for the performance evaluation of our architecture. We use $iperf$ for the generation of IP traffic and $NDNperf$ and $NDN\ Traffic\ Generator$ for the generation of NDN traffic.

We have chosen this specific type of topology, since it allows us to evaluate different scenarios which requires only very little reconfiguration. E.g., IP traffic can be routed directly via nodes 1 and 3 or on a longer path via nodes 1, 2, 4, and 3 between server and client. This allows us to evaluate how an additional switch (node 2 acts as an Ethernet switch for IP traffic) impacts the performance.

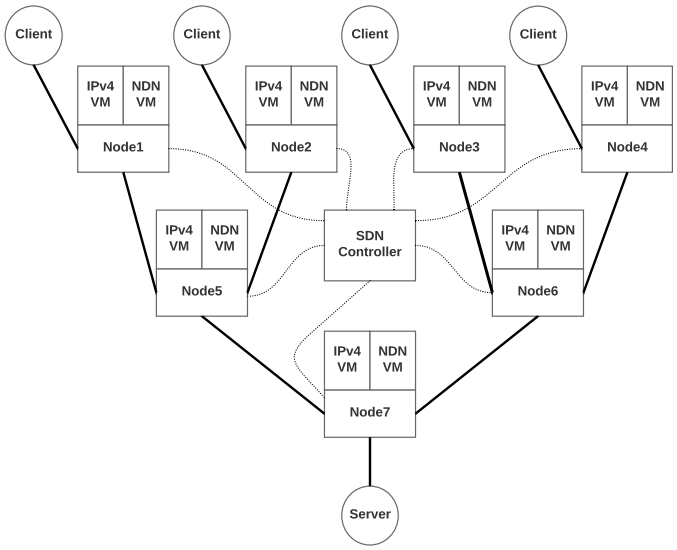


Figure 4.4: Setup 3: Further expand topology to include additional clients

4.2.3 Setup 3: Further expand topology to include additional clients

Similar to the above two setups, the CloudLab slice is shown in Fig 4.4. It consists of 7 nodes (1-7), 4 client PCs and a server, 12 layer 2 links (with each link having a maximum capacity of 10Mbps) and an OF controller runs on a VM. Each of the 7 nodes run both an IPv4 VM instance and an NDN VM instance in parallel. The 4 clients and the server run both on bare metal PCs and generate traffic for the performance evaluation of our architecture.

Having the clients as *baremetal* PCs helps us run multiple client instances and scale the number of clients based on the experimental need. For our experimental evaluation, we consider simultaneously live streaming from 4 clients and 8 clients. To generate IP traffic, the clients initiate $HTTPrequests$ for the video content to the server and to generate NDN traffic we use $NDNperf$ which has the ability to download video content from the server. We have chosen this specific type of topology, since it allows us to evaluate different scenarios which requires very little reconfiguration. E.g., IP traffic is routed through each of the nodes in its path till it reaches

the server. For the NDN's case, we consider a caching based approach where we have the ability to enable or disable the cache, allowing us to evaluate the performance of NDN when all requests are sent to the server and when the cache serves some/most of the requests.

Server: The server node, denoted as *Server* in Fig. 4.4, runs a vanilla Apache2 server that supports HTTP-based segment requests over TCP/IP and an *NDNPerf* server that handles NDN segment requests. Both types of requests are generated by a live video streaming application.

Client: The client nodes, denoted as *Client* in Fig. 4.4, runs the BOLA [30] algorithm, a state-of-the-art buffer-based DASH algorithm, that we incorporate in *AStream*, the open-source Python-based ABR streaming player presented in [15]. In this work, we focus on the BOLA-U algorithm, which maximizes a weighted combination of the bitrate and the smoothness measured in the average rebuffering time.

In order to analyze the effects of bandwidth sharing and congestion, we evaluate each setting for 4 and 8 clients (unless stated otherwise), where all clients simultaneously stream a live video that is 5 minutes long. We also note that the number of clients for each experiment is equally distributed among all the *Client* nodes shown in Fig. 4.4.

4.2.4 Setup 4: Evaluate extendibility on a single SDN/NFV single node

With the above experiments, we implement our architecture with different setups to make a comparative analysis between the IP and NDN case through a live video streaming application. Now, we look to evaluate the scalability of our architecture by running multiple streams between the client and the server with a single SDN/NFV node between the two. The setup shown in Fig 4.5 and Fig 4.6 shows a single node, single client and a single server. In order to analyze the effects of bandwidth sharing

and congestion, we emulate 100 clients simultaneously streaming a live video that is 5 minute and hosted on the server.

We evaluate the following three scenarios:

(a) Scenario 1: Measure the IP-based live video streaming performance with IP on the hypervisor. A similar test is repeated with NDN on the hypervisor.

(b) Scenario 2: Measure the IP-based live video streaming performance with a IPv4 VM on top of the hypervisor. A similar test with NDN VM on top of the hypervisor

(c) Scenario 3: Measure both IP-based and NDN-based live video streaming performance in parallel.

4.2.5 Setup 5: Extend evaluation of extendibility to a larger topology

We extend our evaluation with setup 4 on to a larger topology (Setup 3). The CloudLab slice is shown in Fig 4.4. It consists of 7 nodes (1-7), 4 client PCs and a server, 12 layer 2 links (with each link having a maximum capacity of 10Mbps) and an OF controller runs on a VM. Each of the 7 nodes run both an IPv4 VM instance and an NDN VM instance in parallel. The 4 clients and the server run on bare metal PCs and generate traffic for the performance evaluation of our architecture. We extend our evaluation from Setup 4 by emulating 100 clients on each of the bare metal PCs giving us a total of 400 clients.

Client: The client nodes, denoted as *Client* in Fig. 4.4 runs the BOLA [30] algorithm a state-of-the-art buffer-based DASH algorithm, that we incorporate in *AStream*, the open-source Python-based ABR streaming player presented in [15]. In this work, we focus on the BOLA-U algorithm, which maximizes a weighted combination of the bitrate and the smoothness measured in the average rebuffering time.

Server: The server node, denoted as *Server* in Fig. 4.4 runs a vanilla Apache2 server that supports HTTP-based segment requests over TCP/IP and an *NDNPerf* server that handles NDN segment requests. Both types of requests are generated by a live

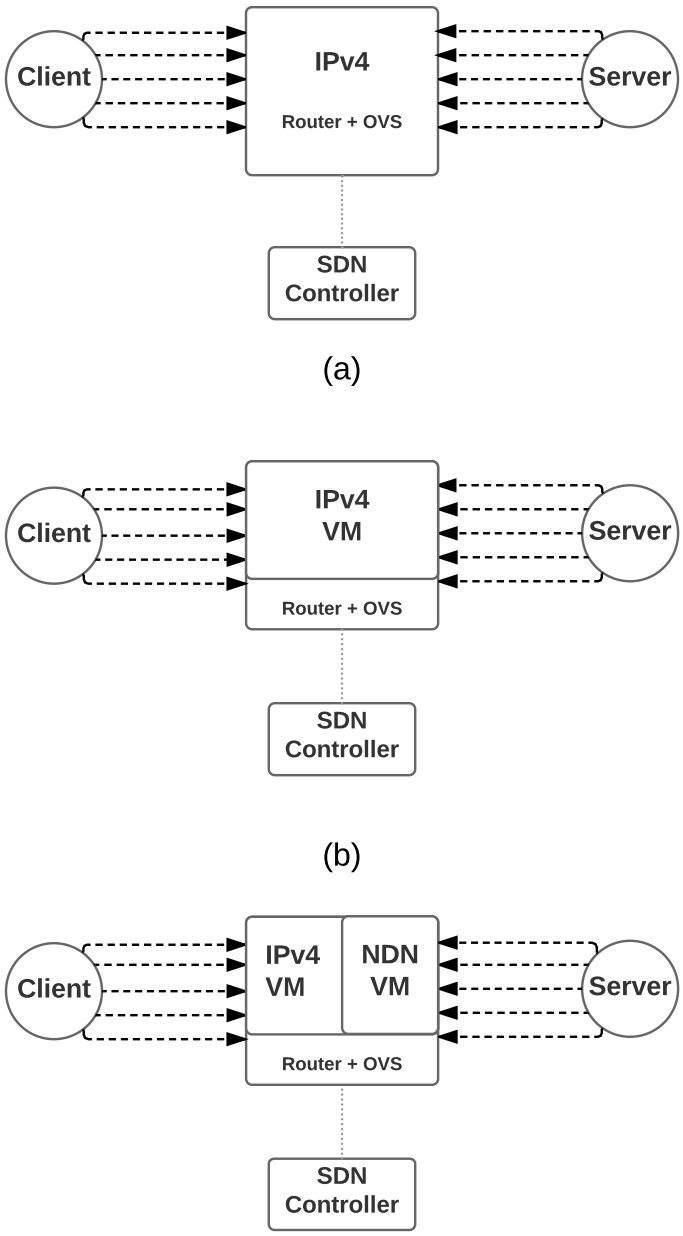
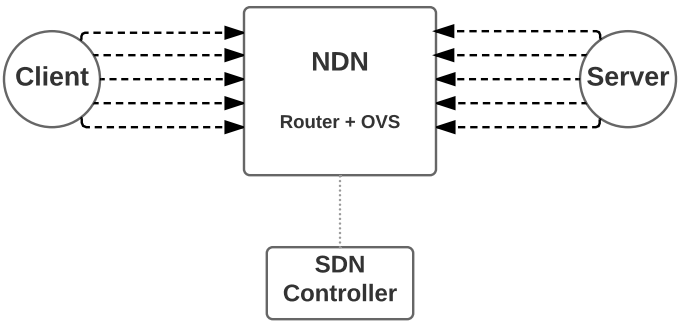
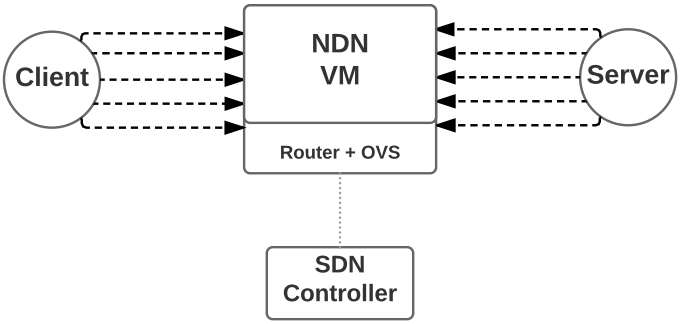


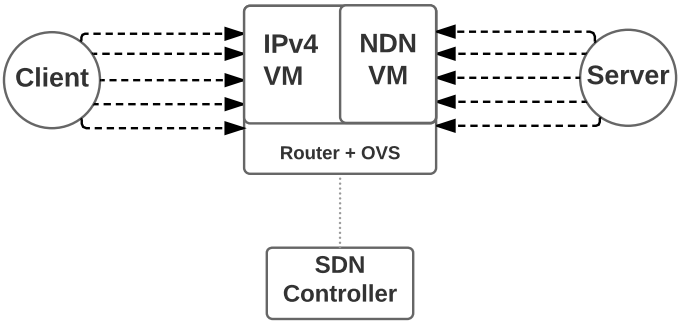
Figure 4.5: IP architecture: (a) Scenario 1 (b) Scenario 2 (c) Scenario 3



(a)



(b)



(c)

Figure 4.6: NDN architecture: (a) Scenario 1 (b) Scenario 2 (c) Scenario 3

video streaming application. Additionally, the server hosts a 5 minute video with 20 different video quality representations.

CHAPTER 5

RESULTS

In this section, we present the results obtained from experiments conducted in the previous section.

5.1 Identifying NDN and IP packets in the Controller

5.1.1 Evaluation Metrics

For this evaluation, we use IP Ping and NDN Ping to evaluate the performance when the controller reactively installs rules. From Fig 5.1 we see that the first IP Ping (denoted as IP Ping with controller) and NDN ping (denoted as NDN Ping with controller) has a longer RTT since there is no flow rule in place and needs the controllers intervention to put a rule in the flow table. The remaining IP and NDN packets have a lower RTT since a rule now is already in place. A comparision between IP and NDN shows that the RTT for NDN Ping is higher than IP Ping. As described in Sect. 2.3, an NDN Router maintains three different data structures: FIB, PIT and CS and forwarding an NDN packet would involve checking these databases. An IP Router forwards packets based on the information in the FIB. Additionally, with the use of SDN, the flow table needs to be checked to route packets based on the flow rules set by the controller. As expected, Fig 5.1 shows the NDN ping has a higher RTT than IP Ping. Since NDN is implemented in user space, the overall performance would be significantly worse than IP (unless caching is used).

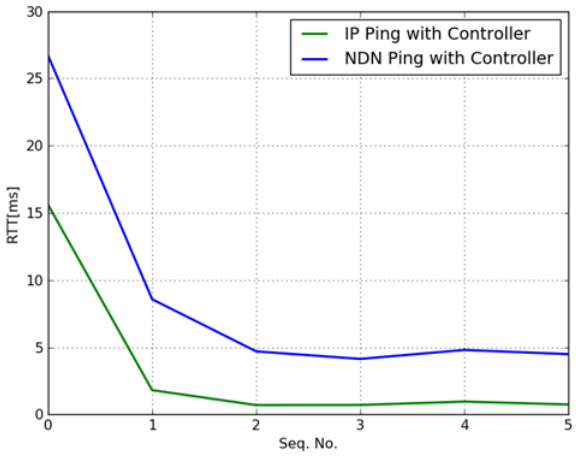


Figure 5.1: Setup 1 - Controller to identify NDN and IP packets

5.2 Expand the topology to make throughtput measurements

5.2.1 Evaluation Metrics

For this initial investigation of our approach, we evaluate the performance metrics of delay and throughput. We focus on these metrics since our initial goal is to study the feasibility of our approach and the performance impact of running network protocols as a virtualized function. Since NDN is conceptually different from IP, we also investigated if certain parameters of the NDN protocol impact its performance.

5.2.2 Evaluation Scenarios

Here, we briefly describe the setup for a series of baseline measurements. The goal of these measurements is to provide metrics for delay and throughput in a non-virtualized topology. The results of these measurements are used to compare them with results from the virtualized, SDN and NFV-enabled topology. For these measurements we set up two additional slices with an identical topology as the one shown for the slice in Fig. 4.3. These slices are different since they only support one Internet architecture (IP or NDN). Thus, OS virtualization, NFV, and SDN are not required.

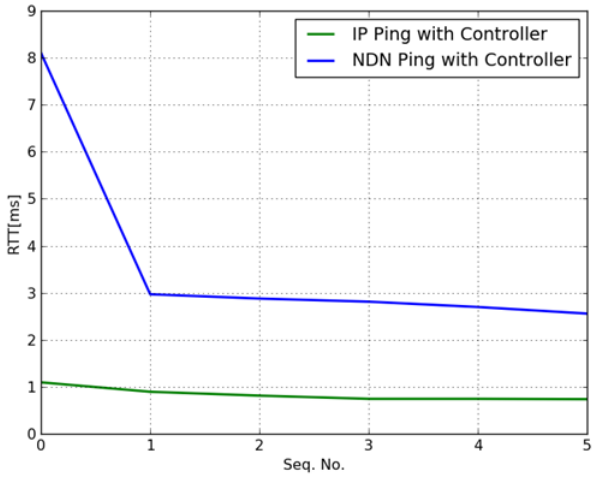


Figure 5.2: Setup 2 - Controller to identify NDN and IP packets

5.2.2.1 Initial Evaluation

We begin the investigation of our approach by using *IP Ping* and *NDN Ping* but this time the controller installs the SDN rules *proactively*. The cases we are considering are when we use NFV setup (denoted as *with NFV*) and when we use a setup without NFV (denoted as *no NFV*). The graph in Fig. 5.2 shows the RTT for the NDN and IP packets are significantly lower than when the rules are installed reactively. In all the future tests, we install flow rules proactively since this helps prevent longer packet processing time.

5.2.2.2 IP-only

For the results presented in this section we make use of a slice that only supports IP (denoted as $non - NFV$) and the slice shown in Fig. 4.3 (denoted as NFV). For this evaluation we use $iperf$ to generate TCP traffic between the server and the client. In this case, traffic is routed on the shortest path ($client$ – $node1$ – $node3$ – $server$) between client and server. The results for the average of 10 runs and the confidence interval are shown in Table 5.1, which indicate that in both cases NFV

and non-NFV, almost the maximum possible throughput can be achieved and NFV does not introduce any performance degradation.

Topology type	Throughput (in Mbps)
Non-NFV (shortest path)	9.67 ± 0.25
NFV (shortest path)	9.69 ± 0.32
Non-NFV (longest path)	9.71 ± 0.34
NFV (longest path)	9.67 ± 0.32

Table 5.1: Throughput for IP measurements.

We perform an additional measurement to investigate if additional nodes on the path between server and client have an impact on performance. In this case, the traffic between client and server is routed along the longest path ($node1$ – $node2$ – $node4$ – $node3$). In the non-NFV case, this increase in hops has no significant impact on the throughput. We obtain similar results for the longest path routing case for the NFV-based topology, as shown in the last row of Table 5.1.

5.2.2.3 NDN-only

For this evaluation we make use of a slice in which NDN is running in the hostOS and no virtualization is used. Traffic is generated with the NDNPerf tool in a variety of settings. The results from this evaluation are shown in Fig. 5.3. We performed these measurements with different settings. First, we used two settings for the size of the ContentStore of the NDN nodes, with one being 0 and the other one 10, to evaluate the impact of caching on the performance of NDN. In addition, we varied the NDN window size – determining how many unacknowledged Interests can be issued by a client – from 1 to 512 (with an exponential increase). The results in Fig. 5.3(a) (No NFV CS=0 and No NFV CS=10) show that NDN achieves a throughput of $\sim 7Mbps$ if the window size is sufficiently large (32 or higher). The results also show that further increasing the size of the window does not have any impact on the

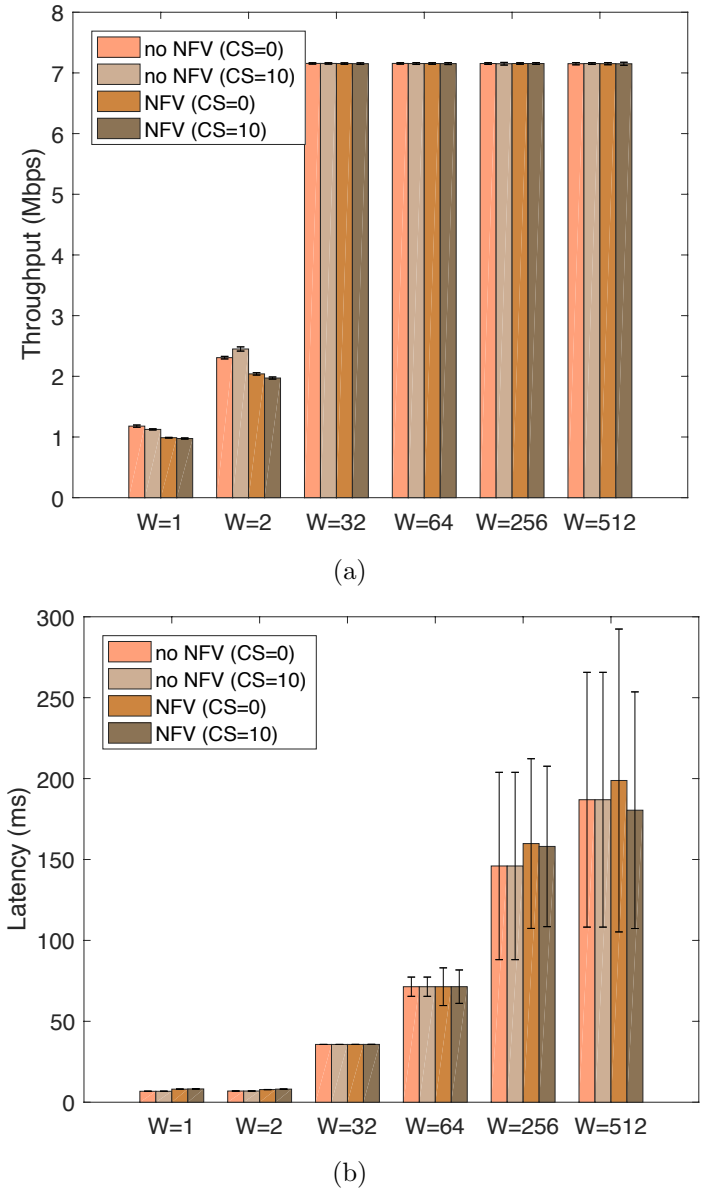


Figure 5.3: Results from NDN throughput and latency measurements. : (a) NDNPerf throughput (b) NDNPerf latency

throughput. Compared to the results for the IP measurement shown in Sect. 5.2.2.2, the throughput for NDN is significantly lower ($\sim 20\%$). We conjecture that the reason for this performance degradation is that NDN is currently implemented in user space and not in the kernel.

To compare the results between the NFV and non-NFV approach we also run the same set of experiments on the topology shown in Fig. 4.3. Fig. 5.3(a) shows that using NFV in the case of NDN does not cause any additional performance loss in terms of throughput. This is similar to the IP case. For the NDN case we also evaluate latency, which is the time between the client issuing an Interest and when the corresponding Content is received by it. The results shown in Fig. 5.3(b) reveal that for large window sizes ($w = 256, 512$) the latency is more variable in the NFV case. With window sizes $w = 256$ and $w = 512$, the number of retransmissions significantly increase resulting in higher latency (total retransmissions = 636 and 1550 respectively)

We perform an additional series of measurements to evaluate if our SDN/NFV approach results in correct behavior of NDN and show the results in Table 5.2. In NDN each router is also a cache and we vary their sizes to observe if this changes the performance according to the expected behavior. The results in the first row of Table 5.2 show results from a measurement where all caches have size 0. The 50 issued Interests probabilistically alternate for two different Contents. We can see that all Interests reach the server and the avg. RTT is the highest of all measurements, since no Content can be cached. In the case of cache size of 1 MB for $Node1$ and $Node3$ (row 2), only $\sim 50\%$ of the Interests reach the server, since half of them can be served from the routers' caches. By further increasing the cache size on $Node1$, $Node2$ and $Node3$ (row 3) shows that number of interests that reach the server are fewer (less than $\sim 50\%$), resulting in reduced RTT time.

CS size for Node1 (MB)	CS size for Node2 (MB)	CS size for Node3 (MB)	Interests sent client	Interests rev'd server	Avg RTT (ms)
0	0	0	50	50	419.59
1	0	1	50	28	267.99
1	10	1	50	22	76.15

Table 5.2: NDN scenarios

5.2.2.4 Concurrent IP and NDN Traffic

	Throughput (Mbps)
NDNperf	2.65 ± 0.65
iperf (TCP)	5.94 ± 2.79

Table 5.3: Parallel NDN and IP Measurement

So far, we have only evaluated scenarios in which either IP or NDN data is transmitted. In this section, we present results from measurements where IP and NDN traffic is transmitted concurrently. Obviously, this can only be performed with the slice shown in Fig. 4.3. In this scenario, $iperf$ and $NDNperf$ run in parallel (on server and client) to generate competing traffic. The ContentStore size is set to 0 on all the NDN routers. Results (average and standard deviation of 10 runs) from these measurements are shown in Table 5.3. The results show that TCP and NDN traffic do not share the maximum link capacity of 10Mbps in a fair manner. We believe that the cause of low throughtput value in the NDN case can be attributed to the fact that the PIT is not fully utilized. With a single client every interest packet is sent to the server. We believe having multiple clients in place would result in interest packets for the same content accumulated in the PIT and resulting in better throughput. For our next evaluation we increase the number of clients to be able to fully utilize the

NDN capabilities and run a real-world application for a better comparative evaluation between NDN and TCP.

5.3 Further expand topology to include additional clients

5.3.1 Evaluation Metrics

For the investigation of our approach, we further expand the topology to include additional clients and evaluate the performance of our architecture with a live video streaming application. The advantages of NDN for efficient delivery of video streams has been widely investigated in works such as [8] and [27], which make ICN-based protocols a likely candidate for the transport of live video. In this work, we particularly look at the emulation of adaptive bitrate based live streaming, where a client selects a video quality bitrate based on the measured download rate history. We chose this application because it makes an interesting case for caching, a functionality that is inherent in NDN and requests for multiple video qualities are potentially distributed among the nodes in the network. Using this application would also help evaluate the advantages in using NDN with other live video streaming applications like Xfinity TV. However, in the traditional IP network case, content source are less diversified possibly resulting in congestion and overall loss in QoE for all video streaming clients in the network. We first introduce the QoE metrics that we use for evaluation of ABR streaming performance before discussing the evaluation setup and results.

Average Quality Bitrate (AQB): One of the objectives of quality adaptation algorithms is to maximize the average quality bitrate of the streamed video. For a comprehensive QoE representation, we need to combine this metric with the *Number of Quality Switches*.

Number of Quality Switches ($\#QS$): This metric is used together with AQB to draw quantitative conclusions about the perceived quality (QoE). For example, for

two streaming sessions having the same AQB, the session with the lower $\#QS$ will be perceived better by the viewer.

5.4 Evaluation Scenarios

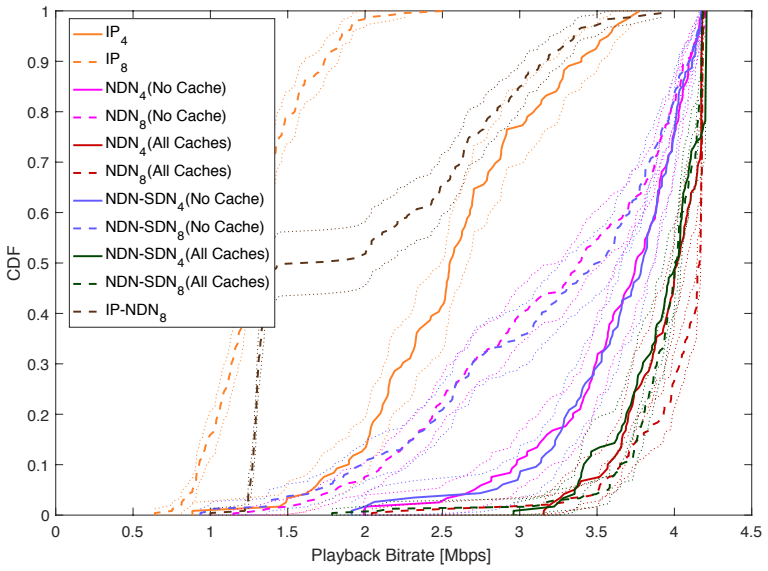
In this section, we show the scenarios that we have considered. Fig. 5.4 presents CDF and CCDF, along with 95% confidence interval, for AQB and $\#QS$ of an ABR live streaming application for various configurations that are repeated 30 times for statistical significance. In the following section, we outline these configurations along with an analysis of the observed results.

5.4.1 IP-based live video streaming

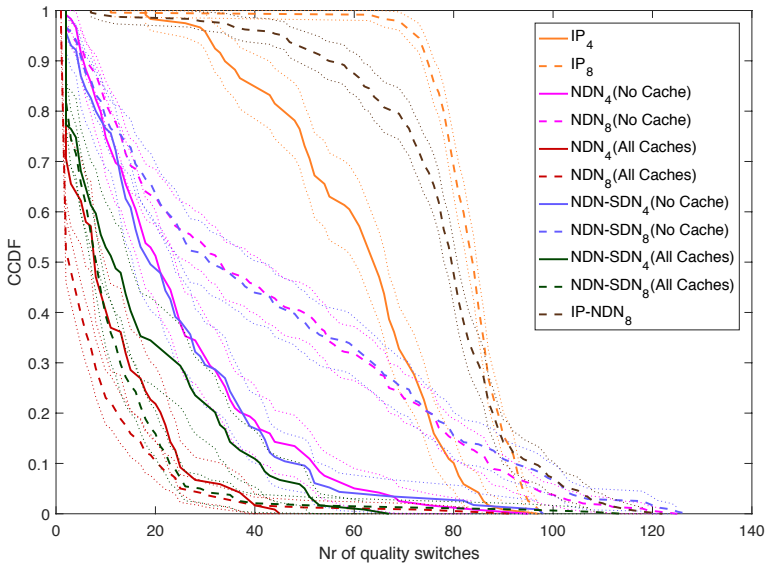
The IP-based measurements are conducted with TCP/IP as the transport layer and the ABR live video streaming application that runs over HTTP (or DASH [29]). We observe that all clients, in both 4 and 8 client cases, obtain a fair share of the bottleneck bandwidth, i.e., each client experiences nearly the same QoE, as expected with TCP-based (we use the default congestion control TCP Cubic) applications. Similarly, the average bitrate, AQB, is halved for the 8 client case as compared to the 4 client case because all content requests are served by a single source, i.e, the *Server* node. The number of quality changes for the 8 client case are the highest where nearly all users experience an average of 80 changes during video playback indicating that the congestion control mechanism in the transport layer leads to a high fluctuation in video quality that is detrimental to the user's viewing experience.

5.4.2 NDN-based live video streaming

In the NDN-based case, the TCP/IP transport layer in *AStream* is replaced by the transport/network layer provided by the *NDNperf* application with a window size of 16 and no congestion control, i.e, the sending rate does not decrease even when interest retransmissions are detected. In spite of this, we observe, from Fig. 5.4, that



(a)



(b)

Figure 5.4: QoE metrics for video playback quality: (a) CDF of Average Playback Bitrate where TCP/IP-based streams suffer the worst quality due to redundancy of requests at the server but NDN streams experience higher average quality and (b) CCDF of Number of quality switches are much higher for TCP/IP than NDN-based streams, which is detrimental to user viewing experience.

all NDN cases (8 client and 4 client, All caches and No cache) show considerable improvement over IP cases. This is due to the fact that NDN incorporates the use of a Pending Interest Table (PIT) and caching, which ensures requests for the same content are cached at routers and content caches, which, in turn, ensure contents are served locally when available. Since video segment requests are distributed across multiple caches in the NDN VMs, the maximum QoE is observed in the NDN-only experiments. This result shows the impact of intrinsic caching in all nodes of the distribution hierarchy and the resulting benefits for a live streaming application.

5.4.3 SDN-based versus NDN-based live video streaming

In this scenario, we evaluate the transport capability of SDN routing for NDN-based live video streaming by disabling the NDN VMs on *Node5* and *Node6*, shown in Fig. 4.4. Here, the two nodes are treated as simple SDN-enabled switches that route NDN traffic according to the rules specified by a centralized OF controller. From Fig. 5.4, we see that for both the 8 client and 4 client case, when NDN is disabled at nodes 5 and 6 a QoE comparable to the case where all nodes run an NDN VM is achieved. A system could thus be comprised of NDN-capable nodes at the edges and SDN-based switches in the core, while still providing good QoE performance for live video streaming.

5.4.4 Concurrent IP-based and NDN-based live video streaming

The goal of this experiment (in which all nodes (1-7) run an NDN VM) is to evaluate the effect of two vastly different protocol stacks (IP and NDN) sharing the same physical substrate. Here, each *Client* node simultaneously runs two *AStream* instantiations; one which uses TCP/IP as the transport layer and another which uses the NDN transport layer API provided by *NDNPerf*. In Fig. 5.4(a), we observe that 50% of the clients have a much smaller variation in average playback rate AQB as compared to the remaining clients. Further analysis, as shown in Table 5.4, reveals

that all 4 IP clients experience a much lower AQB, of 1.34 Mbps as compared to 3.06 Mbps experienced by the 4 NDN clients. We note that the relatively conservative congestion control used by TCP results in the IP video clients backing off from the congestion created by the NDN video clients. In addition, IP packets have to be transmitted end-to-end, which NDN packets can potentially be transmitted from an intermediate node to the clients.

Case	Client 1 (Mbps)	Client 2 (Mbps)	Client 3 (Mbps)	Client 4 (Mbps)	Average bitrate
NDN	3.047	3.090	3.041	3.095	3.06
IP	1.330	1.35	1.33	1.35	1.34

Table 5.4: Concurrent IP-based and NDN-based live video streaming at each client

5.5 Evaluate extendibility with a single SDN/NFV node

5.5.1 Evaluation Metrics

As with the metrics of the previous evaluation (Sect. 5.5.1), the QoE metrics will remain the same. In addition to the average quality bitrate (AQB) and number of quality switches ($\#QS$), we evaluate one additional metric that contributes towards the overall QoE:

Rebuffering Ratio (RB): It is a ratio of the time the user spends in waiting for the video to buffer with respect to the expected/uninterrupted playback time. A zero or lower RB contributes towards a better QoE.

5.5.2 Evaluation Scenarios

In this section, we show the scenarios that we have considered. Fig.5.5 presents CDF and CCDF, along with 95% confidence interval, for AQB and $\#QS$ of an ABR live streaming application for various configurations that are repeated 30 times for

statistical significance. In the following section, we outline these configurations along with an analysis of the observed results.

5.5.2.1 IP and NDN on the host

For this evaluation, we setup IP and NDN on the host as shown figure Fig 4.5(a) and Fig 4.6(a), and run independent IP-based and NDN-based video streaming with 100 clients simultaneously requesting the live video from the server. For the NDN-based streaming, we set the ContentStore (cache) size to 0 MB to make fair comparison with IP-based streaming. The Fig 5.5(a) shows the CDF of the AQB which shows that IP-based streams (denoted as $IP-no-virtualization$) receive the worst quality due to redundancy of requests but NDN streams (denoted as $NDN-no-virtualization$) experience higher average quality. The ability of NDN to accumulate similar interests at the PIT reduces the number of interests that are sent to the server. Fig 5.5(b) shows the CCDF of the number of quality switches($\#QS$) which shows that there are higher number of quality switches in the NDN-based streaming than the IP-based streaming. This can be attributed to the fact that in the case of NDN-based streaming there is the ability to request for higher quality segments resulting in higher number of quality switches. To make a better comparision of the QoE we look at the rebuffering ratio (RB) shown in Fig 5.5(c). From Fig 5.5(c) we see that, IP-based streaming has a longer rebuffering ratio (RB) when compared to the NDN-based evaluation indicating that NDN-based streaming has a better overall QoE than the IP-based streaming.

5.5.2.2 IP and NDN instances running on the host

For this evaluation, as shown in Fig 4.5(b) and Fig 4.6(b) we consider single instances of IP and NDN running on the host machine. These instances are enabled through our SDN/NFV architecture where the controller sets the rules that direct traffic to the VMs. We look at an independent IP-based and NDN-based streaming eval-

uation with IP and NDN instances running on the host respectively. Fig 5.5(a), shows that IP-based streams (denoted as $IP - virtualization$) have a lower quality bitrate when compared to the NDN-based streaming (denoted as $NDN - virtualization$) which experience higher average quality. The ability of NDN to accumulate interests at the PIT means that fewer interests are sent to the server, giving the clients the ability to request for better quality segments and hence resulting in higher AQB. The results also show that our SDN/NFV architecture does not negatively impact the AQB since we get results similar to when IP and NDN are on the host (Fig 4.5(a) and Fig 4.6(a)).

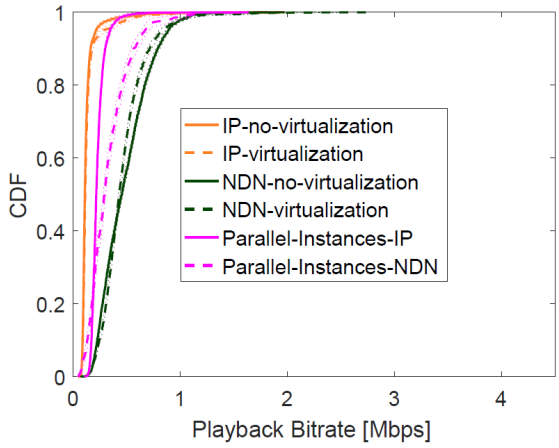
5.5.2.3 Parallel IP and NDN instances running on the host

For this evaluation, we consider parallel IP and NDN instances running on the host as shown in Fig 4.5(c) and Fig 4.6(c). We use the SDN controller to route IP packets to IP VM and NDN packets to the NDN VM. We emulate 50 clients that make IP-based streaming requests and 50 NDN-based streaming requests in parallel. The CDF of the average quality bitrate shows that NDN-based streaming has a slightly higher bitrate as compared to the IP-based streaming. Although the $\#QS$ are higher in the NDN-based streaming, the RB is significantly lower when compared to IP-based streaming. Overall, in NDN-based streaming, the AQB is higher than IP-based streaming with lower RB which leads us to conclude that NDN-based streaming has an overall higher QoE.

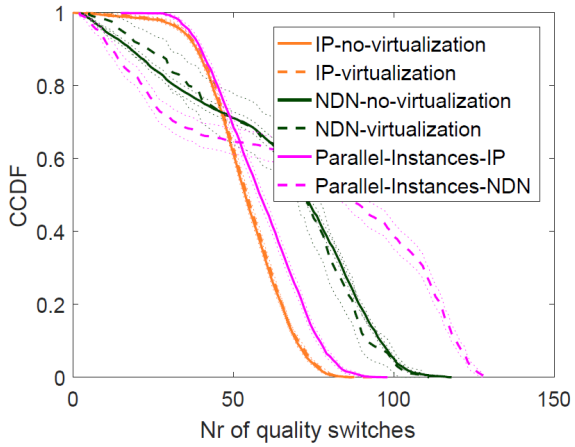
5.6 100 Clients requesting 100 unique videos

5.6.1 Evaluation Metrics

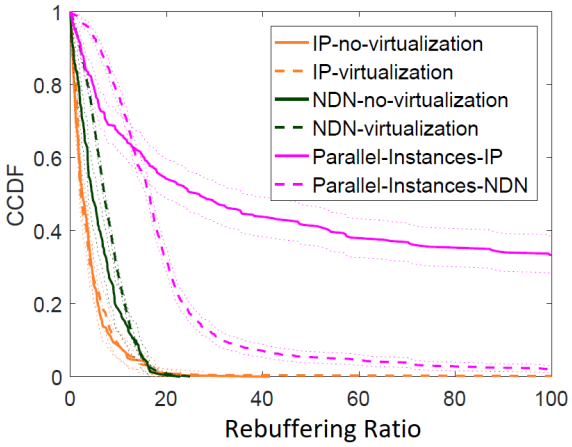
Our goal of this evaluation is to observe the performance of NDN-based and IP based streaming when each client requests for a unique video. Like with the previous evaluation (Setup 4), the setup is shown in Fig 4.5(a) and Fig 4.5(b). We have a single



(a)



(b)



(c)

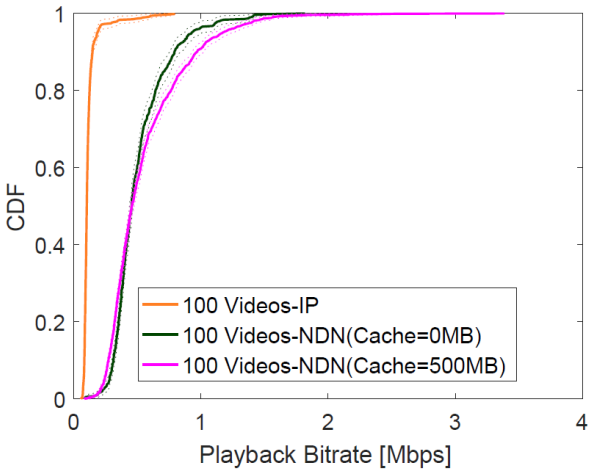
Figure 5.5: Single node evaluation metric with 100 clients: (a) CDF of average quality bitrate (b) CCDF of number of quality switches and (c) CCDF of rebuffering ratio

Server PC which hosts 100 unique 5 minute videos and a *Client* PC on which we emulate 100 clients. We have a single *Router* node between the *Client* and the *Server* on which we alternatively have IP and NDN (no virtualization) to make independent IP-based and NDN-based streaming evaluation. To ensure unique NDN requests for the 100 videos, we update the forwarding table on the *Router* node to add an entry for each of the videos (each video has a unique name). To enable IP requests, on the *Server* we associate each video with a unique IP address and update the forwarding table on the *Router* node with the 100 unique IP addresses. In this manner, a client requests for the video by issuing $HTTP$ requests on the IP address. This setup will thus evaluate the performance of the intermediate *Router* node with the 100 different forwarding rules implemented on it. The evaluation metrics remains the same as the previous setup which are average quality bitrate (AQB), number of quality switches (#QS) and rebuffering ratio (RB).

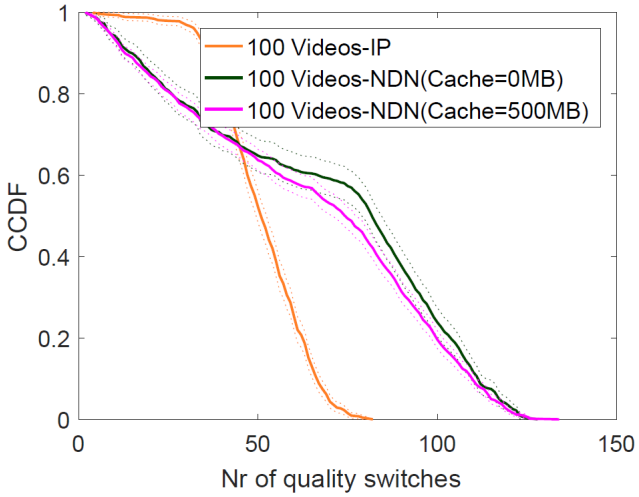
5.6.2 Evaluation Scenario

In this section, we show the scenarios that we have considered. Fig. 5.6 presents CDF and CCDF, along with 95% confidence interval, for AQB and $\#QS$ of an ABR live streaming application for various configurations that are repeated 30 times for statistical significance. In the following section, we outline these configurations along with an analysis of the observed results.

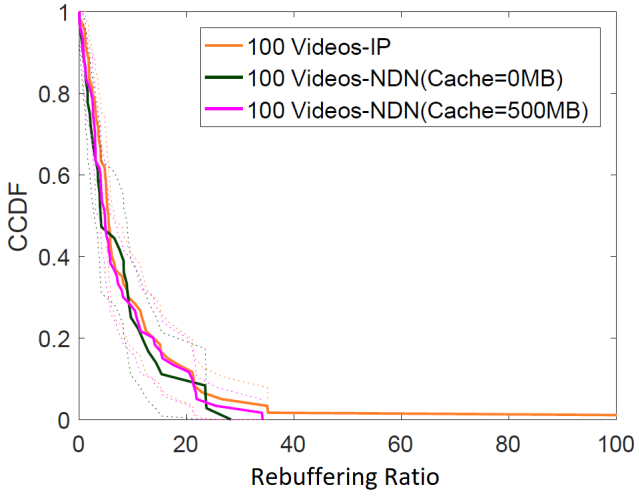
For this evaluation we look at IP and NDN on the host. For the NDN case we analyze how cache based approach will impact the video streaming performance. From Fig 5.6(a), IP-based streams (denoted as 100 $Videos - IP$) suffer the worst quality due to redundancy of requests but NDN streams (denoted as 100 $video - NDN$) experience higher average quality. The improved perfomance in the NDN-based streaming can be attributed to the fact that mutiple requests for the same content are accumulated at the PIT thus reducing the interests sent to the server.



(a)



(b)



(c)

Figure 5.6: Single node evaluation metric with 100 clients and 100 different videos: (a) CDF of average quality bitrate (b) CCDF of number of quality switches and (c) CCDF of rebuffering ratio

When the ContentStore size is set to 500MB with NDN (denoted as 100 *videos-NDN* [Cache=500MB]), we see a slightly higher bitrate value. This increased bitrate value increases the number of quality switches in the NDN cases when compared to the IP-based streaming as shown in Fig 5.6(b). To make a better evaluation of the overall QoE, we analyze the RB shown in Fig 5.6(c) where the IP-based streaming has a slightly higher RB than the NDN-based streaming.

5.7 Extend evaluation of extendibility to a larger topology

5.7.1 Evaluation Metrics

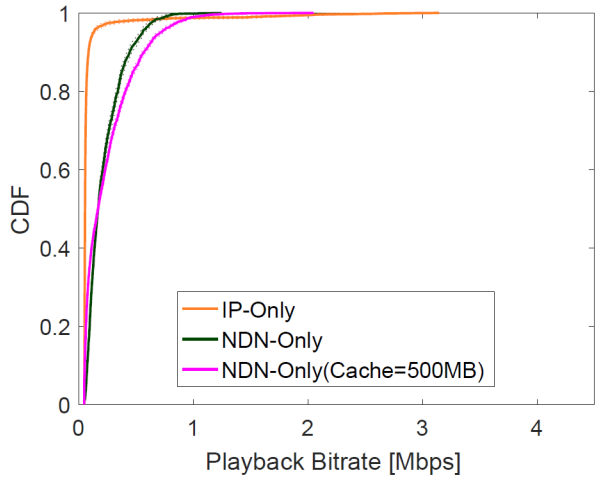
We extend our evaluation to a larger topology as seen in Fig 4.4, by emulating 100 clients on each of the client nodes (1-4) giving a total of 400 clients streaming the video live from the server. As with our previous evaluation, the metrics remain the same which are average quality bitrate (AQB), number of quality switches ($\#QS$) and the rebuffering ratio (RB) all contributing towards the QoE.

5.7.2 Evaluation Scenarios

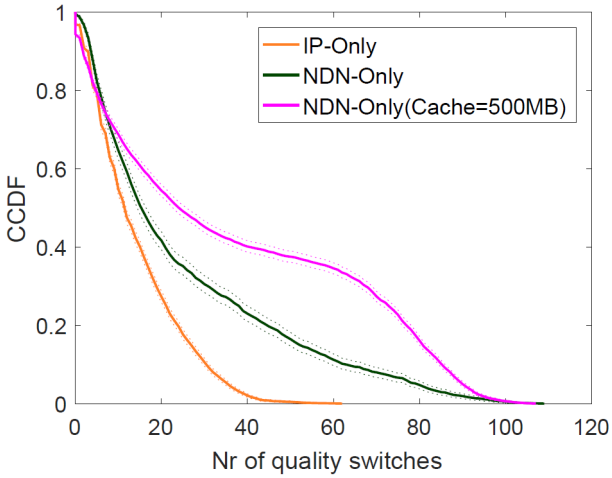
In this section, we show the scenarios that we have considered. Fig. 5.4 presents CDF and CCDF, along with 95% confidence interval, for AQB and $\#QS$ of an ABR live streaming application for various configurations that are repeated 30 times for statistical significance. In the following section, we outline these configurations along with an analysis of the observed results.

5.7.2.1 IP-based live video streaming

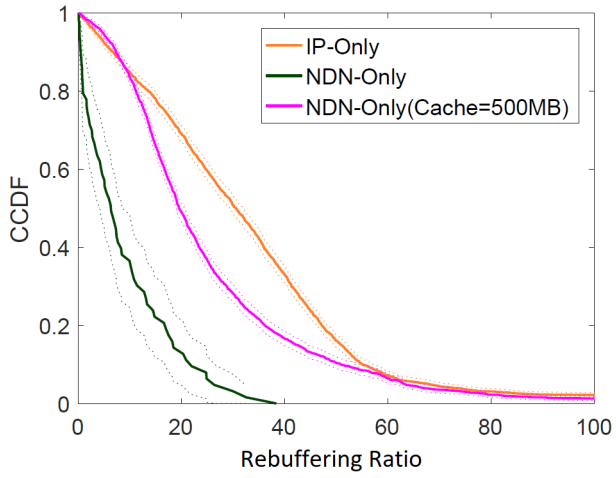
The IP-based measurements are conducted with TCP/IP as the transport layer and the ABR live video streaming application that runs over HTTP. From Fig 5.7(a) (denoted as $IP-only$), we observe that all clients (a total of 400) request lower quality segments which can be attributed to the fact that there are 400 clients simultaneously live streaming the video and all of these segment requests are served by one single



(a)



(b)



(c)

Figure 5.7: Expanded topology evaluation metric with 400 clients and 20 video qualities: (a) CDF of average quality bitrate (b) CCDF of number of quality switches and (c) CCDF of rebuffering ratio

source, the server. With the inablity to request for higher quality segments we expect to see lower number of quality switches in the IP-based streaming which was what we observed in Fig 5.7(b) (denoted as $IP-only$). Similar to our previous evaluations, we look at the RB to make a better evaluation of the overall QoE. From Fig 5.7(c) (denoted as $IP-only$), a high RB is observed.

5.7.2.2 NDN-based live video streaming

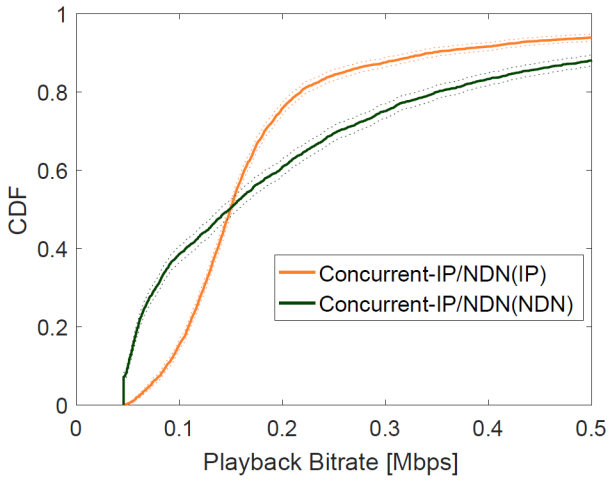
In the NDN-based case, the TCP/IP transport layer in *AStream* is replaced by the transport layer provided by the *NDNperf* application with a window size of 16 and no congestion control, i.e, the sending rate does not decrease even when interest retransmissions are detected. From Fig 5.7(a), with the ContentStore size set to 0 MB (denoted as $NDN-only$) we observe a higher AQB as compared to the IP-based streaming. This is attributed to the fact that with NDN, mutiple interests for the same content are accumulated at the PIT reducing the number of interests that are sent to the server thus providing the clients the ability to request for better quality segments. When we set the ContentStore size to 500 MB (denoted as $NDN-Only$ [Cache=500MB]), we observe a slightly higher AQB as expected since we cache some of the quality segments in the ContentStore. Thus, NDN-based streaming obtain a higher AQB then IP-based streaming. NDN's ability to provide better AQB by requesting for better quality segments results in a higher $\#QS$ when compared to IP as seen from Fig 5.7(b). Like with our previous evaluations, we observe the RB as shown in Fig 5.7(c). We observe a lower RB in the NDN-based streaming when compared to the IP-based streaming which lets us conclude that NDN-based streaming has a better overall QoE than IP-based streaming. From Fig 5.7(c), we also observe that the RB with the Content Size set to 500MB (denoted as $NDN-Only$ [Cache=500MB]) is higher than the NDN case with ContentStore size as 0MB (denoted as $NDN-Only$). Having the cache enabled in the NDN routers provided the clients the ability to not

only request for better quality segments but have these segments served from the nearest cache. The caching policy that is implemented on the ContentStore is LRU (Least Recently Used) which means that the least used entry in the cache is replaced by a newer one. With 20 different video qualities (having a size of 3.4 GB) to choose from and the ContentStore size limited to 500 MB, many of the quality segments are not cached (the least recently used entry is replaced) and the clients request for some of the segments will need to be provided by the server resulting in a higher RB. To prove this hypothesis we reduce the number of qualities and re-run our evaluation as presented in the Sect. 5.8.

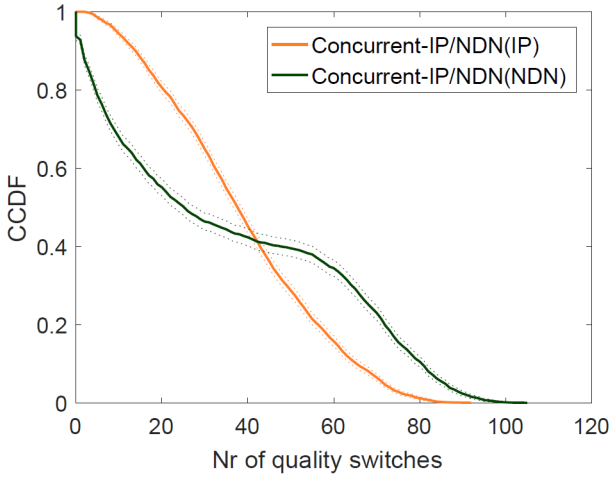
5.7.2.3 Concurrent IP-based and NDN-based live video streaming

The goal of this experiment (in which all nodes (1-7) run an NDN VM) is to evaluate the effect of two vastly different protocol stacks (IP and NDN) sharing the same physical substrate. Here, each *Client* node simultaneously runs two *AStream* instantiations; one which uses TCP/IP as the transport layer and another which uses the NDN transport layer API provided by *NDNPerf*. We have 200 clients that stream over IP and 200 clients that stream over NDN. In the case of NDN, we set the ContentStore size to 0 MB to make comparative analysis with IP-based streaming. In Fig. 5.8, we observe that 60% of the NDN clients have a higher AQB, a lower $\#QS$ and lower RB than the IP clients.

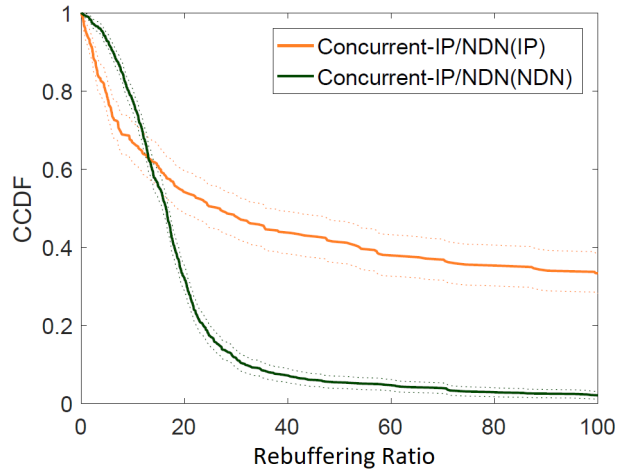
We note that the relatively conservative congestion control used by TCP results in the IP video clients backing off from the congestion created by the NDN video clients. In addition, IP packets have to be transmitted end-to-end, whereas NDN packets can be transmitted from an intermediate node to the clients.



(a)



(b)



(c)

Figure 5.8: Expanded topology evaluation metric with 400 clients and 20 video qualities for concurrent cases: (a) CDF of average quality bitrate (b) CCDF of number of quality switches and (c) CCDF of rebuffering ratio

5.8 Evaluation with frequently requested quality representations

5.8.1 Evaluation Metrics

From our evaluation Sect. 5.7.2.2, we observe a higher RB when the cache is enabled compared to when it is disabled. We believe the number of quality representations contributes to this higher RB. To prove this hypothesis, we reduce the number of quality representations from 20 to 8 and extend our evaluation on the larger topology as seen in Fig 4.4. We emulate 100 clients on each of the client nodes (1-4), giving us a total of 400 clients streaming the video live from the server. The current dataset of all 20 representations hosted on the server amounts to about 3.4 GB. We reduce the size of the dataset to select 8 most requested qualities from our previous evaluations in the Sect. 5.7.2.2. This reduction in the size of the dataset should provide the ContentStore to store most, if not all of the quality representation resulting in a reduction of the RB with the NDN-based streaming(discussed in section Sect. 5.7.2.2)

As with our previous evaluation, the metrics remain the same which are the average quality bitrate (AQB), number of quality switches ($\#QS$) and the rebuffering ratio (RB) all contributing towards the QoE.

5.8.2 Evaluation Scenarios

5.8.2.1 NDN-based live video streaming

The goal of this experiment is to evaluate NDN-based streaming performance with the reduced number of qualities. From Fig 5.9 we observe comparable average quality bitrate when the ContentStore size is set to 0 MB and when its set to 500 MB. From the Fig 5.9(c) (denoted as 8 $Qual - NDN$ $[Cache = 0]$ and 8 $Qual - NDN$ $[Cache = 500MB]$), we observe a significant reduction in the rebuffering ratio (RB) when ContentStore size is 500 MB as compared to the previous evaluation with 20

representations shown in Fig 5.7(c). This can be attributed to the fact that with fewer quality representations, we now have the ability to cache (which has a size limit of 500 MB) most, if not all. With this ability we are able to observe a higher AQB when compared to the previous evaluation with 20 representations shown in Fig 5.7(a). This increase in AQB coincides with the increased $\#QS$ as shown in Fig 5.7(b) which is as expected. Overall with fewer quality representations we have the ability to improve the AQB and reduce the RB. The QoE metric values have been summarized in Table. 5.5 and Table. 5.6.

Case	NDN ContentStore = 0 MB	NDN ContentStore = 500 MB
ABR (Mbps)	0.22	0.24
#QS	25.56	37
RB	9.48	25

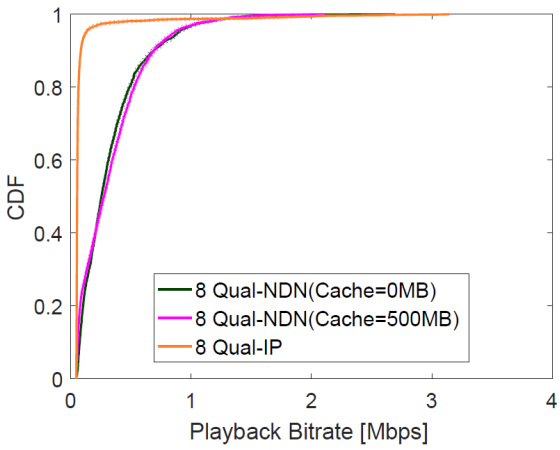
Table 5.5: NDN based streaming with 20 quality representations

Case	NDN ContentStore = 0 MB	NDN ContentStore = 500 MB
ABR (Mbps)	0.32	0.33
#QS	26	47
RB	11.73	9.54

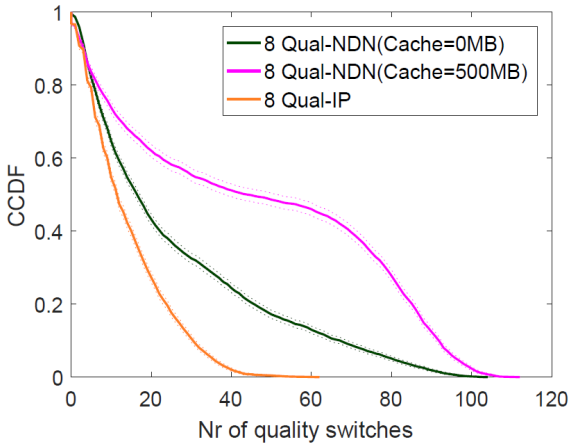
Table 5.6: NDN based streaming with 8 quality representations

5.8.2.2 SDN-based versus NDN-based live video streaming

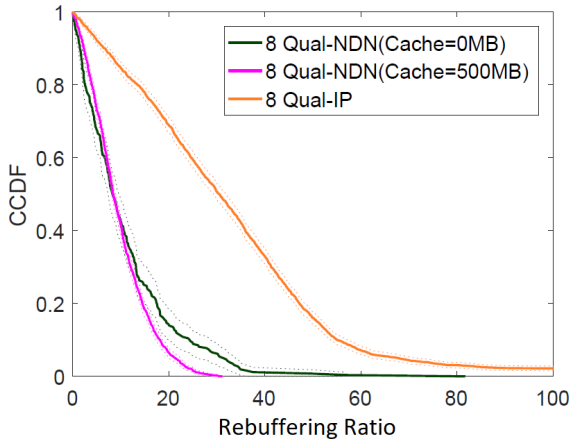
In this scenario, we evaluate the transport capability of SDN routing for NDN-based live video streaming by disabling the NDN VMs on *Node5* and *Node6*, shown in Fig. 4.4. Here, the two nodes are treated as simple SDN-enabled switches that route NDN traffic according to the rules specified by a centralized OF controller. From Fig 5.10, we observe that we get comparable AQB, $\#QS$ and RB, when NDN



(a)



(b)



(c)

Figure 5.9: Expanded topology evaluation metric with 400 clients and 8 video qualities: (a) CDF of average quality bitrate (b) CCDF of number of quality switches and (c) CCDF of rebuffering ratio

is disabled at nodes 5 and 6. A QoE comparable to the case where all nodes run an NDN VM is achieved. A system could, thus, be comprised of NDN-capable nodes at the edges and SDN-based switches in the core, while still providing good QoE performance for live video streaming.

5.9 Streaming with lowest quality representations

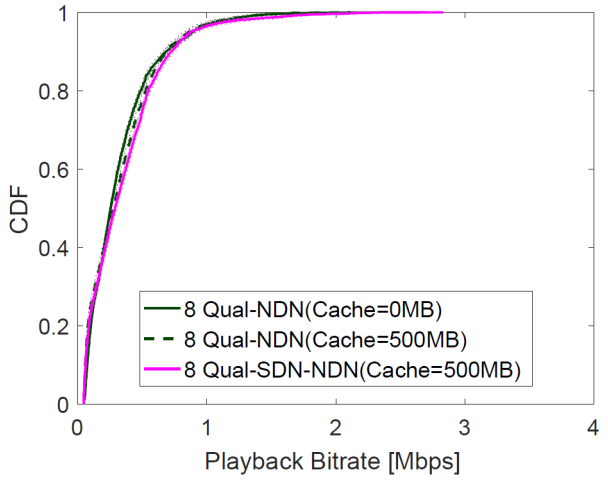
5.9.1 Evaluation Metrics

For our final evaluation on setup Fig. 4.4, we further reduce the number of quality representations from 8 to 2 (lowest two). The goal of this evaluation is to observe QoE specifically with respect to the number of quality switches ($\#QS$) experienced by NDN streaming sessions. From all of our previous evaluations we have observed a higher $\#QS$ with NDN-based streaming than IP-based streaming. With NDN's ability to accumulate multiple interests for similar content in the PIT and having only 2 of the lowest quality represntations to chose from, we hope to observe a lower $\#QS$ in the NDN-based streaming than with IP-based streaming. For this evaluation, we also observe the average quality bitrate (AQB) and the rebuffering ratio (RB).

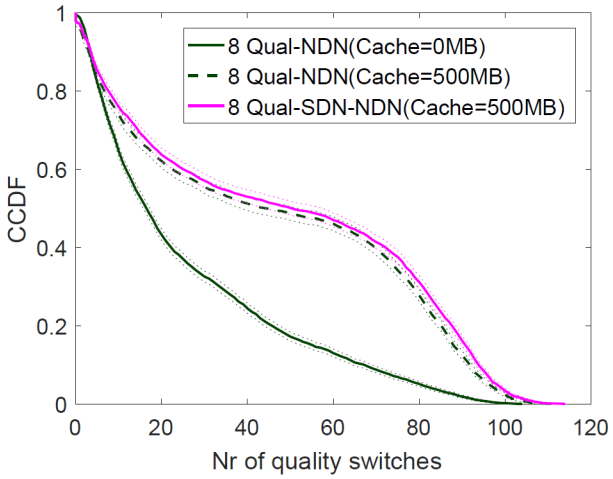
5.9.2 Evaluation Scenario

5.9.2.1 IP and NDN-based live video streaming with lowest quality representations

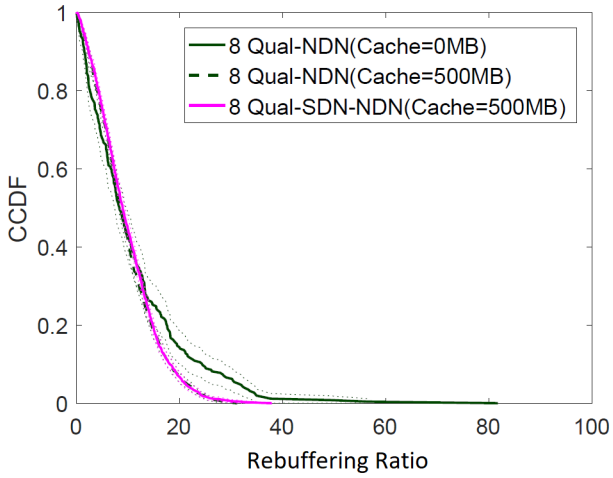
We run independent IP-based and NDN-based evaluation. For the NDN-based streaming we set the ContentStore size to 0 MB to make fair comparison with IP-based streaming. From the graph Fig 5.11, we observe comparable AQB values between the IP and NDN based streaming which is as expected since we have lowest quality representations. With regards to the number of quality switches ($\#QS$), although we just have the lowest 2 quality representations to be requested, NDN-based streaming observes lower quality switches as compared the IP-based streaming.



(a)



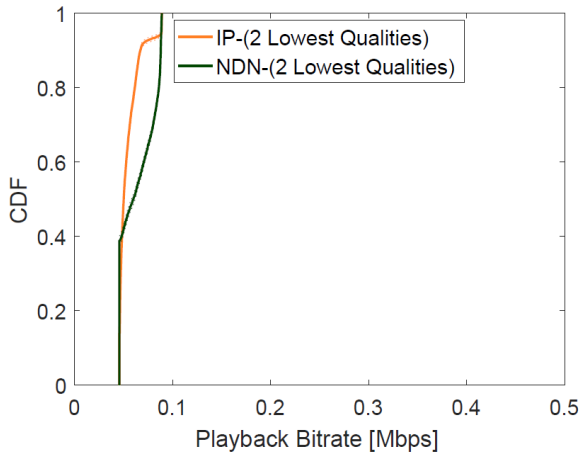
(b)



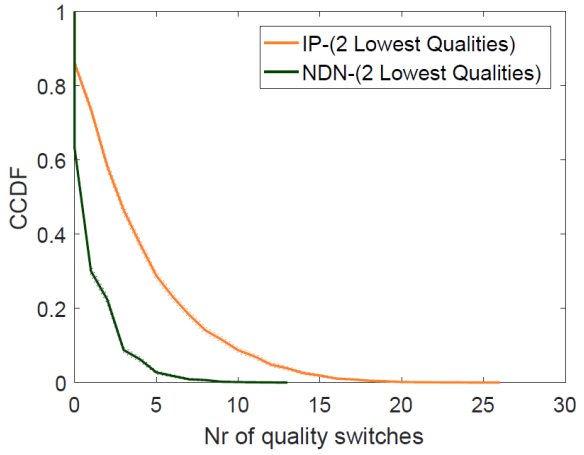
(c)

Figure 5.10: Expanded topology evaluation metric with 400 clients and 8 video qualities for SDN-NDN cases: (a) CDF of average quality bitrate (b) CCDF of number of quality switches and (c) CCDF of rebuffering ratio

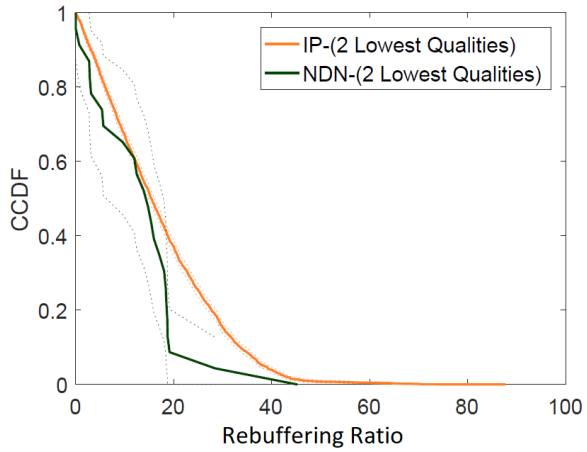
From the RB values, we observe a slightly lower RB with the NDN-based streaming. Overall NDN-based streaming obtains a betters QoE when compared to the IP-based streaming even with the 2 lowest quality representations since IP-based streaming observe higher $\#QS$ and RB.



(a)



(b)



(c)

Figure 5.11: Expanded topology evaluation with 100 clients and 2 lowest quality representations: (a) CDF of average quality bitrate (b) CCDF of number of quality switches and (c) CCDF of rebuffering ratio

CHAPTER 6

CONCLUSION

This work introduces an SDN/NFV-based architecture that allows for the simultaneous operation of different Internet architectures over the same physical infrastructure. We use the CloudLab testbed to perform the evaluation of our approach. Starting with a small setup we show that we can identify and route IP and NDN packets through a centralized controller. With an expanded setup, we then demonstrate that the SDN controller installing rules proactively reduces the overall round trip time. Subsequently, the throughput measurements for the IP and NDN architectures are recorded. We then further expand the setup to run a live video streaming application where we show the advantages of NDN in-terms of the Quality of Experience (QoE). By emulating mutiple clients, starting with 100 and then extending it to 400, we show the extendibility of our SDN/NFV architecture and also show that it does not negatively impact QoE. Through our experimental evaluation with multiple clients, we also show that by having fewer quality representations we can improve the QoE. The flexibility of our approach was demonstrated by the ability to bring down (or up) virtual instances with no impact on the QoE. Overall, results from the evaluation establishes feasibility and flexibility in our approach, providing us the ability to test different protocol stacks inparallel across different test setups. In future work, we plan to evaluate the extendibility of our architecture to determine the sustainibility of the system in the internet as it is today.

BIBLIOGRAPHY

[1] Cisco hicn. https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/ultra-services-platform/mwc17-hicn-video-wp.pdf. Accessed:2017-10-11.

[2] Juniper opencontrail. http://www.opencontrail.org/opencontrail-architecture-documentation/. Accessed:2017-01-16.

[3] Andy Bavier, Nick Feamster, Mark Huang Larry Peterson, and Rexford, Jennifer. Vini veritas: Realistic and controlled network experimentation. In *SIGCOMM '06 Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 3–12.

[4] Ashok Anand, Fahad Dogar, Dongsu Han Boyan Li Hyeontaek Lim Michel Machado Wenfei Wu AdityaAkella David Andersen John Byers Srinivasan Seshan, and Steenkiste, Peter. Xia: An architecture for an evolvable and trustworthy internet. In *ACM Workshop on Hot Topics in Networks (HotNets-X). 2011*, pp. 1–32.

[5] Chiosi, Margaret, and et al. Network functions virtualisation: An introduction, benefits, enablers, challenges & call for action.

[6] Eddie Kohler, Robert Morris, Benjie Chen John Jannotti, and Kaashoek, M. Frans. The click modular router. In *ACM Transactions on Computer Systems (TOCS)*, pp. 263–297.

[7] Gupta, Arpit, Vanbever, Laurent, Shahbaz, Muhammad, Donovan, Sean P., Schlinker, Brandon, Feamster, Nick, Rexford, Jennifer, and Shenker, Scott. Sdx: A software defined internet exchange. In *Proceedings of the ACM SIGCOMM 2014 Conference on SIGCOMM* (2014), ACM.

[8] Gusev, Peter, Wang, Zhehao, Burke, Jeff, Zhang, Lixia, Yoneda, Takahiro, Ohnishi, Ryota, and Muramoto, Eiichi. Real-time streaming data delivery over named data networking. *IEICE Transactions on Communications 99*, 5 (2016), 974–991.

[9] Habib, Irfan. Virtualization with kvm. *Linux J. 2008*, 166 (Feb. 2008).

[10] Huichen Dai, Bin Liu, Yan Chen, and Raychaudhur, Dipankar. On pending interest table in named data networking. In *Architectures for Networking and Communications Systems (ANCS), 2012 ACM/IEEE Symposium*, pp. 1–3.

[11] Internet Engineering Task Force. *RFC 791 Internet Protocol - DARPA Inernet Programm, Protocol Specification*, September 1981.

[12] Ivan Seskar, Kiran Nagaraja, Sam Nelson, and Raychaudhur, Dipankar. Mobilityfirst future internet architecture project. In *AINTEC 11, Asian Internet Engineering Conference, 2011*, pp. 1–3.

[13] Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., and Braynard, R. Networking named content. In *CoNEXT Conference* (2009), pp. 1–12.

[14] Jason Liu, Raju Rangaswami, and Zhao, Ming. Model-driven network emulation with virtual time machine. In *Simulation Conference (WSC), Proceedings of the 2010 Winter*, pp. 1–15.

[15] Juluri, Parikshit, and Medhi, Deep. Cache'n dash: Efficient caching for dash. In *ACM Conference on Special Interest Group on Data Communication*, pp. 599–600.

[16] Kanada, Yasusi, and Nakao, Akihiro. Development of a scalable non-ip/non-ethernet protocol with learning-based forwarding method. In *World Telecommunications Congress (WTC), 2012* (2012), IEEE, pp. 1–6.

[17] Lei Xia, Yuan Tang, Zheng Cui Peter Dinda John Lange Patrick Bridges. Vnet/p: Bridging the cloud and high performance computing through fast overlay networking. In *HPDC '12 - Proceedings of the 21st ACM Symposium on High-Performance Parallel and Distributed Computing*, pp. 259–270.

[18] Mambretti, J., Chen, J., and Yeh, F. Software-defined network exchanges (sdxs) and infrastructure (sdi): Emerging innovations in sdn and sdi interdomain multi-layer services and capabilities. In *2014 International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC)* (Oct 2014), pp. 1–6.

[19] Marchal, Xavier, Cholez, Thibault, and Festor, Olivier. Server-side performance evaluation of ndn. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking* (New York, NY, USA, 2016), ACM-ICN '16, ACM, pp. 148–153.

[20] Mark Handley, Eddie Kohler, Atanu Ghosh Orion Hodson, and Radoslavov, Pavlin. Designing extensible ip router software. In *NSDI'05 Proceedings of the 2nd conference on Symposium on Networked Systems Design Implementation - Volume 2*, pp. 189–202.

[21] McKeown, Nick, Anderson, Tom, Balakrishnan, Hari, Parulkar, Guru, Peterson, Larry, Rexford, Jennifer, Shenker, Scott, and Turner, Jonathan. OpenFlow:

Enabling Innovation in Campus Networks. *ACM SIGCOMM Comput. Commun. Rev. 38*, 2 (Mar. 2008).

[22] Nascimento, Marcelo R, Rothenberg, Christian E, Salvador, Marcos R, Magalhaes, Maurıcio F, Corrêa, CN, and de Lucena, Sidney C. The routeflow approach to ip routing services on software-defined networks. In *Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)* (2011).

[23] Pfaff, Ben, Pettit, Justin, Koponen, Teemu, Jackson, Ethan J., Zhou, Andy, Rajahalme, Jarno, Gross, Jesse, Wang, Alex, Stringer, Jonathan, Shelar, Pravin, Amidon, Keith, and Casado, Martín. The design and implementation of open vswitch. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation* (Berkeley, CA, USA, 2015), NSDI'15, USENIX Association, pp. 117–130.

[24] Ricci, Robert, Eide, Eric, and The CloudLab Team. Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications. *USENIX ;login: 39*, 6 (Dec. 2014).

[25] Rob Sherwood, Glen Gibb, Kok-Kiong Yap Guido Appenzeller Martin Casado Nick McKeown Guru Parulkar. Can the production network be the testbed? In *OSDI'10 Proceedings of the 9th USENIX conference on Operating systems design and implementation*, pp. 365–378.

[26] Sama, M. R., Contreras, L. M., Kaippallimalil, J., Akiyoshi, I., Qian, H., and Ni, H. Software-defined control of the virtualized mobile packet core. *IEEE Communications Magazine 53*, 2 (Feb 2015), 107–115.

[27] Samain, Jacques, Carofiglio, Giovanna, Muscariello, Luca, Papalini, Michele, Sardara, Mauro, Tortelli, Michele, and Rossi, Dario. Dynamic adaptive video

streaming: Towards a systematic comparison of icn and tcp/ip. *IEEE Transactions on Multimedia 19*, 10 (2017), 2166–2181.

[28] Sardara, Mauro, Muscariello, Luca, Augé, Jordan, Enguehard, Marcel, Compagno, Alberto, and Carofiglio, Giovanna. Virtualized icn (vicn): Towards a unified network virtualization framework for icn experimentation. *Proc. ACM ICN* (2017).

[29] Sodagar, I. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *IEEE MultiMedia 18*, 4 (April 2011), 62–67.

[30] Spiteri, Kevin, Urgaonkar, Rahul, and Sitaraman, Ramesh K. Bola: near-optimal bitrate adaptation for online videos. In *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on* (2016), IEEE, pp. 1–9.

[31] Stringer, Jonathan P., Corrêa, Carlos, Bailey, Josh, Pemberton, Dean, Fu, Qiang, Lorier, Christopher, Nelson, Richard, and Rothenberg, Christian Esteve. Cardigan: Deploying a distributed routing fabric. In *19th IEEE Symposium on Computers and Communications (ISCC), June 2014* (2014), IEEE.

[32] Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., Claffy, K., Crowley, P., Papadopoulos, C., Wang, L., and Zhang, B. Named data networking. *SIGCOMM Comput. Commun. Rev. 44*, 3 (July 2014), 66–73.