International Journal of Scientific Engineering and Technology ISSN:2277-1581
Volume No.6 Issue No.1, pp: 23-28 01 Jan. 2017

Performance Evaluation for ICN over Custom-SDN Network

Mareb H. Atiyah, Dr.Ammar D. Jasim
Al.Nahrain University, Baghdad, Iraq
mareb.atiyah@yahoo.com, ammar@coie.nahrainuniv.edu.iq

Abstract: Information-Centric Networking (ICN) and Software-Defined Networking (SDN) are considered a promising networking paradigms in the process of developing Internet architecture. Thus The combination between ICN, which is providing efficient data distribution as a standard, and SDN by providing flexible management environment, leads to a resultant combination to be fully controllable network for efficient data dissemination. In this work, we implemented ICN functionalities by using SDN network. The network design is described based on CCNx as the implementation of ICN, Floodlight as the SDN controller type, Open VSwitch as a virtual SDN switch, and Wrapper which is a middle layer that enables the SDN open flow switch to access the ICN named data packets. Our design provides efficient content delivery with transmission delay lower than that in traditional network, and proved SDN feasibility to implement ICN with higher throughput and acceptable packet loss and jitter in comparison with traditional networks.

Keywords: ICN; SDN; OpenFlow; Wrapper; CCNx; Open VSwitch.

I. Introduction

Most of Internet users are more interested in accessing the desired contents rather than their physical addresses. In traditional TCP/IP network, communication is based on interface addresses of both communicated hosts, Information-Centric Networking (ICN) shifts the communication mode to a content-centric communication paradigm. It supports efficient content delivery by using a name-based routing protocol and placing cache in appropriate nodes of network [1]. ICN deployment is hard due to its extremely different underlying structure. As a result, how to deploy ICN in current Internet architecture, which entirely depends on IP, has become a problem.

Another important new paradigm introduced to support the continuous evolutions of current networking architectures is Software Defined Network (SDN). It decouples the control plane from the data plane. The switch hardware locates the data plane, it preserves the optimized forwarding hardware, whereas the network control plane is centralized into an intelligent programmable authority that is called the controller, which is considered as the brain of the network. The controller provides the intelligence to instruct network switches, to make traffic routing and control through the network infrastructure [2]. The current most popular implementation of the SDN networking paradigm is found in the Open Flow protocol [3], which is first developed in Stanford University in 2008 and is currently under development with the Open Networking Foundation(ONF) [4]. That separation provided by SDN paradigm, is to increase the flexibility of network interconnection equipment control, it also provides openness and programmability that will promote the pace of network innovation. SDN enable the development of new routing and forwarding strategies with just programming. Due to these advantages mentioned above, SDN is generally considered as an applicable solution to ICN deployments. As ICN is a new technology, the most important question is how to implement it in current networks and make use of the provided features. In this work, Software Defined Networking (SDN) architecture is exploited to implement ICN. We used the Wrapper [5] software which make it possible to deploy ICN functionalities over OpenFlow switches. The whole design with its basic components are presented in this paper. Hence the rest of our paper is structured as follows: Section II presents a background on ICN and SDN. In Section III we describe our ICN over SDN approach and explains the design components in detail. Section IV specifically describes our implemented approach using Floodlight SDN controller and CCNx. In Section V we describe the implemented ICN over Traditional network for comparison purposes. In Section VI the evaluation results are introduced with visual graphs. Finally, we conclude our work and open new perspectives for future work in Section VII.

II. Background

In this section the basic idea and description of ICN and SDN are elaborately introduced.

A. Information-Centric Networking

ICN is a novel paradigm to computer networks, which is considered as an approach for content dissemination and retrieval. It focuses on the content objects, so that the network itself become enriched with information about the content. This contradicts the current traditional internet architecture, which depends on IP address, and thus has only the ability to address the hosts. This mutation is provide shifting from host-centric to information centric. The ICN research community objective is to expose content information to the network and let the network figure out where to get the content from and how to process it the best. One of the ICN approach advantages is efficient content distribution, but as lately argued, this advantage is not sufficient to motivate a switch to change its infrastructure. There are many other features, that reinforce the motivation to use the ICN, includes scalable and efficient content distribution, persistent and unique naming, and the presenting of new security model which depends on securing the content itself, rather than the communication channel [6]. However, a number of challenges, that still open and remain without a clarification, related to named based routing are yet to be addressed in order to successfully deploy a content oriented

ISSN:2277-1581
01 Jan. 2017

networking model for future Internet architecture. There are many ICN proposals that have been progressed in the past. All these proposals tried to implement the ICN general idea. For instance, NDN/CCNx [7], PSIRP/ LIPSIN [8], NetInf [9], and DONA [10].

B. Software Defined Networking

In general, it is hard to experiment with new kinds of networks. As these types of networks sometimes employ multiple addressing schemes and include other aspects that is differ from standard one, it is hard to incorporate these new types into existing networks. OpenFlow permits switches, access points and routers from various organization to use the segregation of the control and data planes. The nodes forward data packets that were received based on the rules defined via the controller. If there is no rule for the received data packet, the devices forward these packet to the controller. The controller determines the appropriate action with the packet and sends a new rule to the device so that, it can process future data packets in the same way. The SDN is the technology that adopts the OpenFlow as basic protocol to provide the openness and programmability [11,12]. SDN concept divides the network into three planes which do not match directly with the layers of the OSI reference model. SDN basic structure, that shown in Figure 1, made by Open Network Foundation (ONF) [13]. SDN basically focus on four factors:

- Separation of Control and Data Planes

SDN separate the network devices data plane from control plane. The Data Plane is composite from network elements and its function is providing connectivity. network elements include routers, Ethernet switches and firewalls, with the difference that the forwarding decisions does not autonomously made by the control logic on a local level. Instead, Network elements configuration is provided by the control interface with the Control Plane. To optimize network configuration, the elements frequently sends status updates to a network controller.

- A Centralized Controller

Once the control and data planes are split, it is not necessary to have a distributed control plane. As a result, most SDN realizations migrate an essential portion of network control functionality to a centralized SDN controller, which described as the brain of any SDN network. Every switch in the network is connected to that controller through OpenFlow protocol which allows it to monitor and control each device.

- Open interfaces between the control and the data plane

In the OpenFlow (OF) architecture, there are three main pieces of OpenFlow includes OF controller, OF switch, and OF protocol. An OF switch consists of a number of flow tables in addition to an abstraction layer which communicates with a controller via OpenFlow protocol in a secure manner. Flow tables contain forwarding flow entries, each one match packets to its correspondent flow, then processed and forwarded. These entries consist of many parameters, include i) match fields, which are used for matching the incoming packets, also it uses information of ingress port, packet header, and metadata. ii) counters, which are used to setting up statistic data for each flow, such as the number of received packets, bytes and duration of a particular flow. iii) instructions set used when there is a match; they determine how to process matched packets [14].

- Using External Application to Support the Programmability Feature

In SDN environment, it is possible to program and control network devices, it gives the ability of using software applications, that are developed from hardware, for the purpose of controlling the rules of packet forwarding. In SDN, it is possible to apply network applications that have precise traffic processing and monitoring capabilities. This will provide a dynamic QoS provisioning, access control, and load balancing. it also provides chances to implement many new types of services [15].

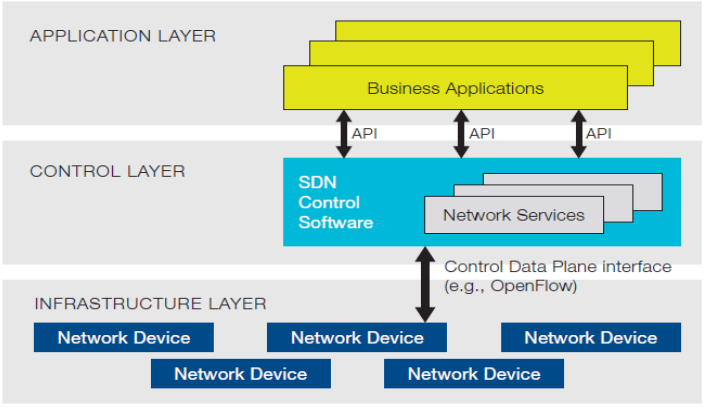


Figure 1: SDN architecture

C. Related Work

Recently, there have been many implementations of combining SDN with ICN. Melazzi and et *al.* proposed content-centric inter-network(CONET) [16] which provide users with remote named resources instead of remote addressed hosts. It supports both overlay and clean-slate already proposed deployment approaches. However, this proposal requires modifications in IP protocol and does not support CCN architecture.

Syrivelis and et al. [17] describe how to pursue a Software Defined Information-Centric Networks and how SDN implementation should be modified to support their content-centric architecture called Blackadder. It simplifies the ICN node and network architecture by using LIPSIN identifiers. But it cannot support name-based caching and efficient content delivery.

Nguyen Nam and et al. [5] proposed Wrapper as an approach to enable Content Centric Networking(CCN)functionalities on software Defined Networking open flow switches. without needing to change the open flow protocol. The evaluation results proved the Wrapper feasibility and low overhead on the network. we used the Wrapper software in our design.

III. ICN over SDN Approach

Our design of ICN-SDN network mainly consists of four components as shown in Figure 2, it includes an Open Flow switch and a controller to build the SDN environment, on the other hand, the CCNx daemon [18] that providing the ICN functionalities, and the Wrapper which runs as a middle layer

ISSN:2277-1581
01 Jan. 2017

between them. The details about each component is provided in the following subsections.

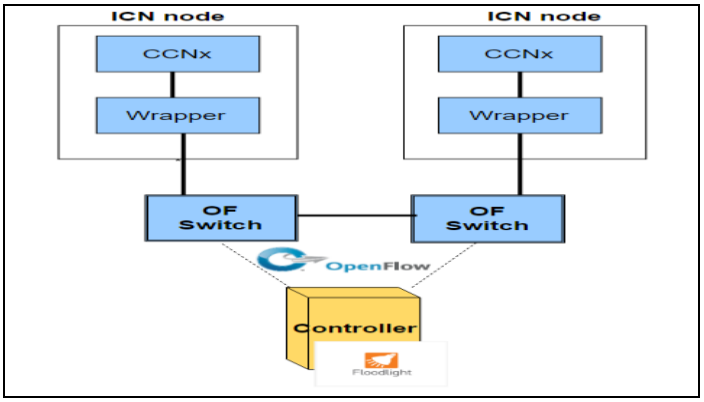


Figure 2: Design components

A. Open Flow Switch

Open VSwitch version 1.0 is used in this work. Which is responsible for packet forwarding. Originally the OpenFlow switch consists of one or more flow tables. The flow table consists of a set of entries called flow entries which provides the forwarding rules for each flow passing in the OpenVSwitch and communicates with the controller using OpenFlow protocol via a secured channel over the network. The matching field of the flow entries matches only the IP packets, and the OpenFlow specification which allowed only to identify the TCP/UDP ports, but it can't analyze the payload of packets, as this needed in our design in case of efficient transport for CCNx packets. Thus, Wrapper software is used to run as a middle layer between the OpenVSwitch and the CCNx daemon.

B. SDN Controller

Floodlight controller is used to implement the ICN over SDN network design of this work. After the process of cloning floodlight controller 1.0 master version is done, floodlight works on port 6653, from its repository on GitHub web site. Floodlight controller programmed by java language, so Java Development Kit (JDK) tools and Another Neat Tool (ANT) must installed first. Then it installed on virtual machine that runs Ubuntu 14.04 LTS as operating system. The taskset utility is used with the controller running command to bind the controller to four threads, thus getting full benefit of them instead of just running on single core.

C. CCNx Daemon

CCNx is the implementation of Content Centric Network(CCN) or ICN. It is sponsored by the Palo Alto Research Center (PARC) and is currently under development [17]. The CCNx 0.8 version is used in this work, and it supports Linux platform. As an ICN reference implementation, the CCNx provides the components and libraries required to build and run applications, that required for our design implementation process. Communication between hosts that runs the CCNx daemon is driven by the consumer. Basically there are two CCNx packet types: Interest sent by hosts who are interested in content; and Data that contains chunks of the content that are requested from the consumer. In order to get Data, the CCNx consumer(requester)broadcasts an Interest packet, which contains content name in the form of URI such as */home/pic/123.jpg* that defines the requirement. CCNx by default listens on port 9695 and it uses UDP to carry ICN packets. In the implementation of our network design CCN_LOCAL_PORT is changed by using specific script, to make a new instance of CCNx on each host.

D. Wrapper Software

In this thesis, Wrapper software cloned from its repository and built using the Debug file inside Ubuntu 14.04 virtual machine, that already have installed CCNx daemon, because it is one of the prerequisites to accomplish Wrapper's work. To run the Wrapper first, the CCNx protocol must be start running, then adding a static UDP route to forward all CCNx packets to the Wrapper, finally adding the row socket interface (i.e. host name and interface like h1-eth0) and the CCNx port number. By default, it takes a name *%C1. M. FACE and* hashed it to IP address equal to *150.53.92.12*. The static routs in the ICN-SDN networks are initiated to ensure that the Interests are destined to the hosts that runs the wrapper software first, and the hosts starts requesting Interests from the server, Simultaneously the wrapper takes the Interests and hashed them to an IP using the hash function. For example, the Interests name with prefix no1 are hashed to IP 101.104.158.124, and the receiver Wrapper port 8888 is used. So that it could be transfer over OF switches and reaches to the Server.

IV. Implementation

There are many available SDN topology types, that include Linear, Single switch topo, tree, and fat tree these topologies have a static structure and can be implemented using multiple number of OF switches and hosts. There is another topology type called custom topology that provides flexible way to create any desired network design with multiple number of switches and hosts. We used a custom topology consists of three OF switches, six hosts and floodlight controller, its architecture shown in Figure 3, using script written in python language. This network is used to implement ICN functionalities, including transfer of named content, across SDN network through the OF switches.

The python script of the custom topology is implemented on virtual machine using mininet emulator. The topology script includes lines of code to remotely connect the OF switches to the floodlight controller using the IP address of the virtual machine that runs the floodlight controller on specific port 6653. To test the network connectivity ccnping tool is used. Host h6 is used to run the ccnpingserver using URI in the form ccnpingserver ccnx:/name/prefix. Hosts h3, h4, and h5 are used to request Interests (named content) from the server by using the command: *ccnping ccnx:/name/prefix*, which uses the same URI that is advertised from the server. A data packet sent from the server to the Wrapper, that is running in hosts h1 and h2, so as to be delivered to the requested host that resides in different or the same OF switch.

ISSN:2277-1581
01 Jan. 2017

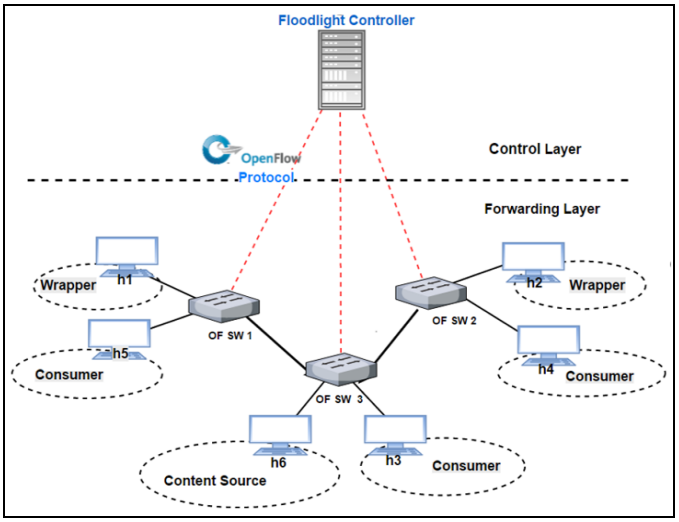


Figure 3: ICN-SDN-Custom Network Architecture

V. ICN-Traditional Network

CCNx is the official implementation of ICN, it builds an IP overlay to transport Interests and Data packets. Thus it can run on traditional IP networks. In this work, a traditional network is built using structure as the same as that is used to build ICN-SDN network, as shown in Figure 4. The purpose of implementing this network is to use all the available network types to build initial ICN implementation. And to demonstrate the difference between the traditional network and the SDN network.

By observing Figure 3 and Figure 4, it's obvious that the SDN had separate the control layer from the forwarding layer. This separation supports the complex communication between ICN nodes when Implementing full ICN functionalities on large scale networks, and resolve the IP problems such as security and traffic explosion.

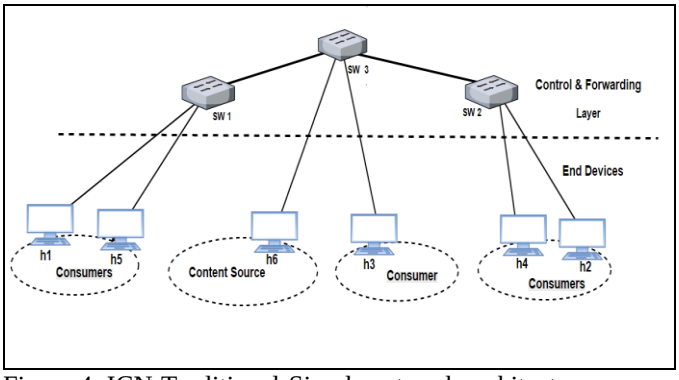


Figure 4: ICN-Traditional-Simple network architecture

VI. Evaluation and Results

To evaluate the performance of the implemented design, we measured the basic performance parameters that includes transmission delay, throughput, packet loss, and jitter. By using different tools. The performance measurements are performed to gather data about the ICN-SDN network, and compare it's results with ICN-Traditional network. The purpose of this evaluation is to test the applicability of the proposed design, and show the benefits from implementing ICN functionalities over SDN rather than implemented it over current traditional network. This evaluation strategy requires building a traditional network copy for the corresponding SDN network and use the same setting for network links and components to get proper results. In the following subsections the performance parameters and their results are presented.

A. Transmission Delay Measurement

We measured the transmission delay for the ICN-SDN network by sending different number of interests from the ccnping client to ccnping server. At first we obtained the results by taking the transmission delay result of the first request. Then, we repeated the test many times until we reached a steady state. The steady state results are presented in Figure 5 in comparison with the first request results.

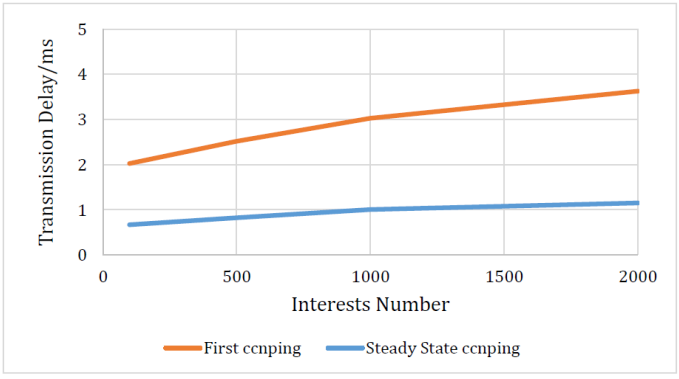


Figure 5: Measured transmission delay for ICN-SDN network

The transmission delay for ICN-traditional network is measured, and presented in comparison with that in ICN-SDN network, as shown in Figure 6 and Figure 7.

For ICN-SDN network, the first request results show additional delay in comparison with that in steady state. This is due to the initial flow setup delay for the SDN scenario, while the topology is still being discovered, there might be additional delay due to the controller's internal representation of the network state, and the controller have to learn all the devices reactively for each packet-in. After the devices have been learned and the controller reaches steady-state, the ping times from a client to server should be consistent. This clear up the decrement in the delay at steady state, as shown in Figure 5.

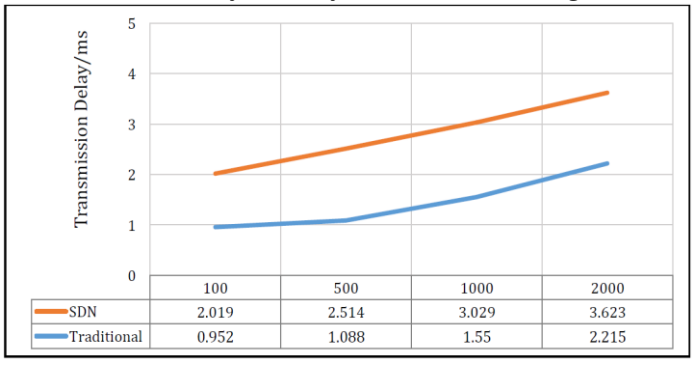
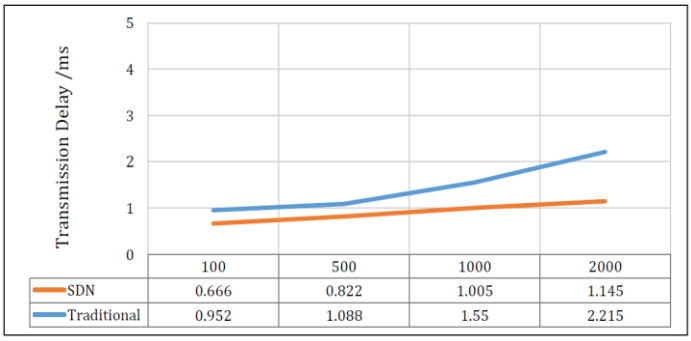


Figure 6: ICN-SDN Network First Request Results Versus ICN-Traditional Network

ISSN:2277-1581
01 Jan. 2017



	100	500	1000	2000
SDN	0.666	0.822	1.005	1.145
Traditional	0.952	1.088	1.55	2.215

Figure 7: ICN-SDN Network Steady State Results Versus ICN-Traditional Network

After comparing the results of the two networks, we found that the ICN-SDN network at first time a round shows additional delay, due to the above reasons, but when it reaches a steady state it's results become better than that in ICN-traditional network. This indicate that SDN network norm dose not adversely affect the network performance, except at the beginning when the network starts the definition process.

B. Throughput Measurement

To measure the throughput for both networks, ICN-SDN and ICN-Traditional network, iperf tool is used. By making a TCP connection between iperf client and server. Then TCP flows are generated by setting the parameters of the iperf tool. Different Mbytes sizes of transferred data, in range 100 Mbps - 1000 Mbps, are used to test the network throughput. The link bandwidth of the implemented networks is 100Mbps. Figure 8 shows the throughput results of the ICN-SDN network versus ICN-traditional network. The results show that the ICN-SDN network throughput is higher than that in traditional network, this is due to the floodlight performance, where the reactive mode in the controller allows a more efficient use of OF switch memory (MAC tables), when OF switch receive a packet without matching flow, the packet is sent to the floodlight controller, which evaluates it, adds the appropriate entries and let the switch continue it's forwarding. In addition, the floodlight controller is set to work in a multithread mode (to make use of the available environment threads, in this work, four threads are used), which enhances the processing speed of the controller, so that, it can handle more processes at the same time and that enhances the network throughput.

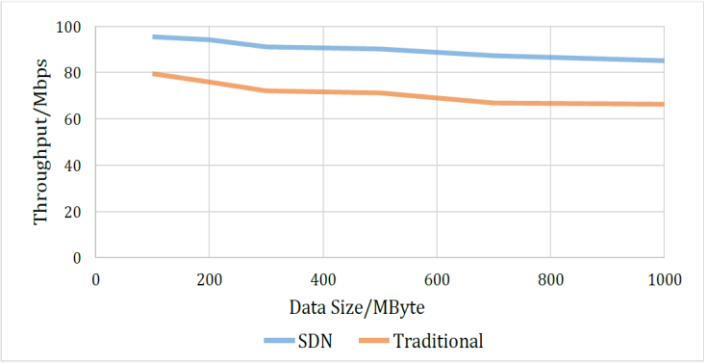


Figure 8: throughput measured for ICN-SDN network versus ICN-traditional network

C. Packet Loss and Jitter Measurement

To measure packet loss and Jitter Iperf is used. By making a UDP connection between iperf client and server. In this test 1 Gbyte is used to test network performance. The results of the tested networks are presented in form of visual graphs including Figure 9 and Figure 10, that presents packet loss and jitter values gathered after run the test for each network. The results of the two networks are so close to each other, and the most important point here is that it does not exceeds the acceptable values (<1% packet loss) of the packet loss and jitter.

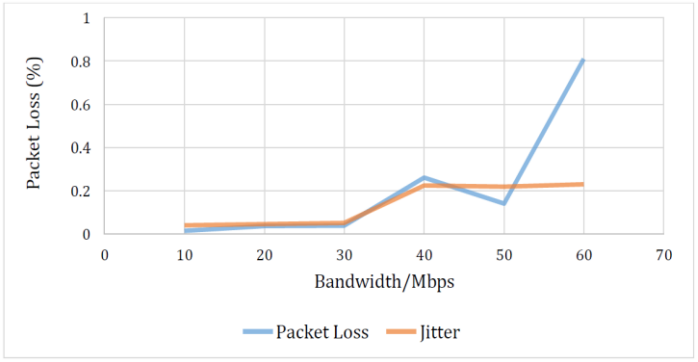


Figure 9: Packet Loss and Jitter Measured for ICN-SDN Network

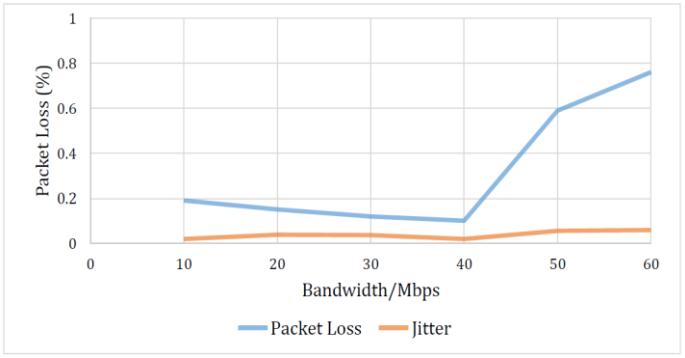


Figure 10: Packet Loss and Jitter Measured for ICN-Traditional Network

VII. Conclusions and Future Work

In this work, we implemented an initial prototype of ICN functionalities using SDN network. The implemented design enables ICN to work in compatible with SDN by adding a middle layer software called Wrapper without the need to do any modification neither in SDN nor in ICN. Since there is no modification required, the use of IP is still available. The other important point, is that the network design exploits the capacity of SDN components for name forwarding and monitoring. Where the Open Flow controller is used to centralize the control of FIB in ICN. Considering the design feasibility, we implemented the basic performance measurement, as a result, the ICN-SDN networks shows an additional transmission delay when the network is run for the first time. But it does not degrade the performance as the SDN network controller reaches a steady state. After that, the delay is enhanced to reach values better than the traditional networks delay values. The ICN-SDN

ISSN:2277-1581
01 Jan. 2017

network throughput results are better than that in traditional network. Finally, Packet loss and jitter results for the SDN networks are so close to that in traditional network, and they don't exceed the acceptable values.

To develop the current implemented work, multiple controllers could be utilizing to manage and control the network collaboratively, adding a module to the controller to do the Wrapper function, and Using other ICN functionalities such as in network caching.

References

i. *Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking", IEEE Trans.Communication Magazine, Vol-50, Issue-7, 2012.*

ii. *B.J. van Asten," Increasing robustness of Software-Defined Networks Fast recovery using link failure detection and optimal routing schemes", Thesis No: PVM 2014-082, Delft University of Technology,2014.*

iii. *Open Network Foundation. (2013, apr) Openflow switch specification openflow version 1.3.2 (wire protocol 0x04). [Online]. Available:* https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.2.pdf

iv. *Open Network Foundation(ONF), SDN architecture overview (version 1.0), [Online]. Available:* https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/SDN-architecture-overview-1.0.pdf

v. *N. Nguyen, D. Saucez, and T. Turletti, "Providing CCN functionalities over OpenFlow switches", HAL Id: hal-00920554, version 1, Inria Sophia Antipolis – Mediterranean, France, 2013.*

vi. *M. Bertrand, P. Truong, J. F. Peltier, Wei You, and G. Simon, "Information-Centric Networking: Current Research Activities and Challenges", Media Networks: Architectures Applications and Standards, CRC press, France, 2012.*

vii. *Named Data Networking project, "NDN Project Overview", [Online]. Available: http://www.named-data.net.*

viii. *P. Jokela, A. Zahemszky, C. E. Rothenberg, S. Arianfar, and P. Nikander, "LIPSIN: Line Speed Publish/Subscribe Inter-networking", Proceedings of the ACM SIGCOMM conference on Data communication (SIGCOMM '09). New York, USA, 2009.*

ix. *Ahlgren, M. D'Ambrosio, C. Dannewitz, A. Eriksson, G. Jovan, "Second NetInf Architecture Description," 4Ward EU FP7 Project, Tech.report D-6.2 v2.0, 2010.*

x. *T. Koponen, A. Ermolinskiy, M. Chawla, K. Hyun Kim, I. Stoica, B. Chun, and S. Shenke, "A Data-Oriented (and Beyond) Network Architecture", conference on Applications, technologies, architectures, and protocols for computer communications (ACM SIGCOMM '07), Kyoto, Japan, 2007.*

xi. *Fei Hu, "Network Innovation through OpenFlow and SDN Principles and Design", 1st edition, CRC Press, Taylor & Francis Group, 2014.*

xii. *A. jasim and D. Hamid," Enhancing the Performance of OpenFlow Network by Using QoS", International Journal of Scientific & Engineering Research, Vol- 7, Issue-5,2016*

xiii. *Open Networking Foundation, "Software-Defined Networking: The New Norm for Networks", USA, 2012.*

xiv. *S. Azodolmolky," Software Defined Networking with OpenFlow", 1st edition, Livery Place, UK, 2013.*

xv. *V. Shukla," Introduction to Software Defined Networking-Openflow & VxLAN", 1st edition, North Charleston, USA, 2013.*

xvi. *Detti, N. Blefari-Melazzi, S. Salsano, and M. Pomposini, "CONET: A Content Centric Inter-Networking Architecture", ACM SIGCOMM Information centric networking (ICN '11), New York, USA, 2011.*

xvii. *Syrivelis, G. Parisis, D. Trossen, P. Flegkas, V. Sourlas, T.Korakis, and L. Tassiulas, "Pursuing a Software Defined Information-centric Network", European Workshop on Software Defined Networking, Darmstadt, Germany ,2012.*

xviii. *CCNx OpenSource project. (2014). [Online]. Available:* http://www.ccnx.org/