Indonesian Journal of Electrical Engineering and Computer Science
Vol. 17, No. 3, March 2020, pp. 1410~1421
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v17.i3.pp1410-1421

A comparative study for bandwidth on demand using ONOS Reactive and Intent forwarding

Fathul Arif Kamarudin[1], Megat Norulazmi Megat Mohamed Noor[2], Fuead Ali[3]
[1]MIIT IoT Research Group, Univesiti Kuala Lumpur, Malaysia
[2,3]Department of Computer Enginering, Univesiti Kuala Lumpur, Malaysia

Article Info

Article history:

Received Jun 11, 2019
Revised Sep 13, 2019
Accepted Sep 27, 2019

Keywords:

Intent forwarding
Performance
Reactive forwarding
Software defined network
Testbed

ABSTRACT

Telco operators need to tailor their networks to be agile, efficient and able manage the operational cost at moderate level. Furthermore, the norm of network volatility practiced by many enterprises open to a new challenge in managing an increasing traffic over the same WAN. For an enterprise, turning toward Bandwidth-on-Demand (BoD) approach promises to deliver extra network capacity when in demand without the complexity of running separate network. However, they need to manually transform these high-level policies into low-level configuration command and thus error-prone. This study was to address the effects of SDN (software-defined network) on the BoD performance towards the differences between the reactive and intent forwarding based on the assessment matrix. The testbed was performed under Mininet simulator which interacted with ONOS (Open Network Operating System). Mesh type topology with a predetermine parameter matrix was used using random topology generator. The expected outcome from this research was to show the effect on the BoD performance under two type of forwarding; reactive and intent based on the assessment matrix. This report contributed towards determining the effectiveness of SDN architecture on BoD performance and to improve future benchmarking on SDN-BoD testbed.

Corresponding Author:

Fathul Arif Kamarudin,
MIIT IoT Research Group,
Universiti Kuala Lumpur,
Bandar Wawasan, 50300 W.P Kuala Lumpur, Malaysia.
Email: fathul.kamarudin@s.unikl.edu.my

1. INTRODUCTION

Bandwidth-on-demand is generally part of the service offered by Telco. The possibility of risks and investment entails when it comes to such service often reduced the opportunity to gain better understanding of its performance. For that, a testbed is a viable method that can be adopted by simulating the required environment envisioned by the R&D of Telco corporation to gain the feedback one may obtain in a real-life situation. The purpose of this study is to provide analysis and discussion on the preliminary data obtained from the early experimental testbed conducted for the research regarding the change-over of network performance through the implementation of certain techniques available in software defined network. In this experiment, the data extracted from the result is based on the comparison between two type of forwarding techniques; intents and reactive method. The reactive forwarding used in the experiment is a host-specific method which is based on entries for every destination host connected to the similar virtual network. The intents method however applies ONOS's intents framework which act as policy-based directives. Both type of forwarding is tested under a list of parameter matrix with specific parameter as-sessments. The experiment is conducted under Ubuntu environment with Mininet and ONOS installed. The testbed consists of a virtual network topology using Mininet that is attached to ONOS controller. From the testbed,

we expect to see changes in the network performance under these two types of forwarding techniques and demonstrate as to why the techniques affect the network performance in such way.

Current network architecture is mainly involved with the integration of various network products and components, also known as network elements to make up a legacy network. The term legacy that associates with network is a common way to describe old network elements that either has been obsolete or has the potential to be one in the future. In terms of service provider perspective, the networks are fundamentally more complex in addition to being multilayer which entails to high availability and performance. Developing the strategies to increase revenue such as Bandwidth on Demand, Pay for Network Features and other services are the responsibilities of Telekom operators [1]. The computer networks however are designed in such a way that it should accommodate huge numbers of network elements; consisting of routers, switches, servers and other types of middleboxes. With the constant struggle of trying to outpace the explosion of Over-the-Top (OTT) service, mobile device and delivery of content across the cloud, a huge challenge present itself for most network service providers to keep up with the exponential growth of internet data traffic. Such concern is put forth by Hakiri [2] stating that the demand for traffic utilization growth is parallel to network architecture in a way that the current network infrastructure capacity to adhere for an explosion of increment in data traffic is limited.

Bandwidth-on-Demand also known as BoD is considered by researchers as one of the ways that Telekom operators can adopt to address the limit of network infrastructure and traffic data growth [1, 3-4]. The concept of Bandwidth on Demand (BOD) has been around for quite a while even though it is lacking as prime focus from Internet service providers (ISPs). Bandwidth-on-Demand is a literal connectivity service that allow the user to request bandwidth on any preferred levels, regardless of the time and location for it to be needed it; thus, providing them the solution to instant demand and scheduled bandwidth. It can be associates with Network as a Service (NaaS) which one of its model functions is to provide personal preference and modification on the user's network usage. Instant demand can be viewed as the ability for network infrastructure to provide the requested bandwidth as real time as possible while still maintaining its connectivity to the user whereas a scheduled bandwidth in BoD service is the option to reserve a desire bandwidth allocation during a specific time and location. According to Omar [5], the way traditional BoD operates can be explained by the following process described in Figure 1 and the layers in SDN architecture as shown in Figure 2.

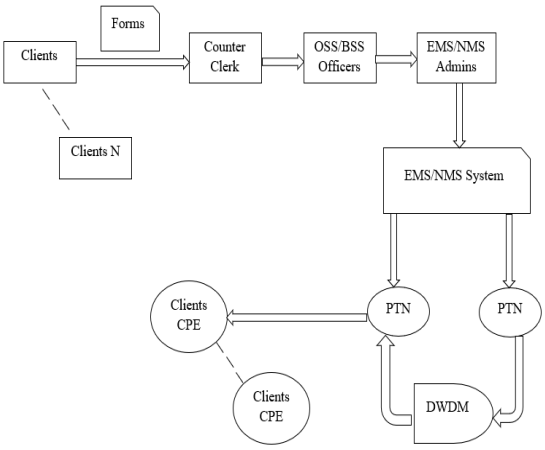

Figure 1. Traditional bandwidth-on-demand service mechanism

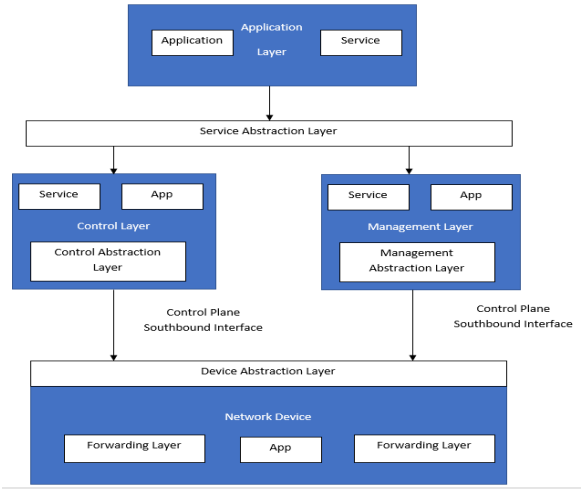

Figure 2. The layers in SDN architecture

Whenever the clients requested any changes on the subscribed bandwidth for their network circuit configuration, they are required to fill form where it will be processed by the OSS/BSS (Operation Support System/ Business Support System) officers. The configuration of network circuit changes is only made by Element Management System (EMS) and Network Management System (EMS) administrators. These administrators have the access to view personal information which related with service subscription such as personal name, company name, bandwidth subscription, start and end date subscription and network information via this application.

One of the important issues that needs to be addressed by ISP regarding the Bandwidth-on-Demand services according to fellow researchers is the over-dependent towards hardware to manage the complexity of network architecture. According to researchers [6], the current operational method employed in Bandwidth-on-Demand is unable to overcome the growth of complexity in network infrastructure. Once the bandwidth demand grows, so does the network infrastructure to withstand the resource require to adhere such service.

ISP is facing the problem with static network infrastructure of unable to properly be dynamic enough for Bandwidth-on-Demand due to the legacy network elements being too saturated in the network infrastructure in which removing it or upgrading it would cost a hefty amount of capital expenditure. ISP can combat this situation by integrating the current mechanism of Bandwidth-on-Demand with a software-driven force that promote flexibility and effective optimization for the control of network complexity [1]. Due to the nature of software, configuration for new changes should be improved, efficient and the time taken for the changes to take place is supposed to be instantaneous and overarching. All these issues need to address, and some solutions have already been provided in a way that theoretically are able to tackle the issues risen by the current implementation of Bandwidth-on-Demand. The challenges and the possible solution for each challenge can be summarize into Table 1 as explained .

Table 1. The Challenges and Possible Solutions of Bandwidth-On-Demand

Bandwidth-on-Demand	
Issues	Possible Solutions
Lack of software-driven control to optimize network complexity [6].	Deploy a centralize controller that masks the internal network control details and acts as programmatic interface to the application [1].
Operational and cost problem for current non-SDN BoD production service affecting the Opex and Capex of an enterprise [7, 8].	A cost-saving approach for designing elastic networks on demand to manage topology and control network resources dynamically [9].
Large data transfers can clog the routers with packets waiting for direction [9].	Open Network Operating System adopt a centralized architecture of scalability and scale-out. Improving scalability and performance[10].

Software-defined networking (SDN) is a methodology in computer networks where it allows for the logical control of the network and enable the possibility of making the computer networks to be programmable and flexible. In its basic, the SDN architecture separate the data plane and the control plane where the software component in SDN is responsible for the control plane of the network; allowing researchers and software developers to install network application between the gap of the planes through its network wide abstraction. In SDN management, vendor-specific interface usage for managing network device individually is replaced with centralized SDN controller. According to the researchers [11], they discussed with the idea of "platform as a service" model for networking. It is considered as a common tendency to separate the infrastructure management from the service management and hiding the underlying physical network and the topology to the user. The authors further mention that the customer is mostly interested in being able to configure policies and defining how packets are handled. The abstraction is where a single router is presented to the main logical controller. Another example includes using names for IP addresses or high-level policies instead of access control configuration files. Figure 2 shows the decoupling of the data plane and the control plane in SDN architecture. The SDN layer is separated into three parts; application, control and device or architecture layers. The main concept of SDN is to combine the separated control layer into one single point of network. What it means is that for this type of architecture, the network device is only required for handling the data layer and forwarding the data packet from one point to another based on SDN controller's decisions. Thus, one specific controller which is directed through application in application layer controls every switch by using application programming interface (API).

The demand for cloud services is increasing drastically [2]. With the rapid expansion of customer demands, the operator is expected to adhere accordingly by taking into account the additional servers, network components, reliable and secure architecture [12]. In particular, the challenges and issues that are of utmost importance in the SDN environment are the following:

a) Scalability: This defines the ability of SDN to handle and process an increasing workload more specifically in the control plane [13].
b) Reliability: The SDN is considered reliable when notifying data delivery failures is in real time. Therefore, SDN controllers must be able to meet real-time requirements for reliable delivery and promptness [14].
c) High Availability: The state in which the services is available constantly for a customer to requests its resource [15].

For ISP to specifically improving Bandwidth-on-Demand, introducing SDN technology into the mechanism is the best approach. Some of the researchers proposed ways to implement the SDN-BoD as shown in the Table 2.

Even though there are many researchers proposed the benefits of SDN implementation in current network service infrastructure, there are lacking proper testing regarding the optimization of the core engine enabling the function of SDN. Even though an experiment of physical testbed in nature can be performed by researchers, the study that can be done to gain feasible data would still be limited as the experiment is inevitably bound to the predetermined number of resources available; blocking the possibility of a scalability. Thus, most of the research so far is based on theoretical models, simulations and emulations. Therefore, there is a need for experimental testbed that can provide all the complex interactions between an observable environment and the subject under test [16]. Another concern raises by Morreale & Anderson [17] is the idea of benchmarking for and SDN-BoD approach. There are many parameters in SDN-BoD approach that influence the optimization of its performance as many as the network elements involved [18].

Table 2. The Implementation Approach of Software-Defined Network

Proposal	Main Purpose	Technique	Angle
A distributed SDN control platform [18, 19]	Improving scalability and performance	Clustering	Scalability Availability
A cost-saving approach for on-demand network in SDN [20,21].	Solving the underlying network design problem	Mixed-integer programming	Elasticity
Using a caching mechanism for data network applications [20,21].	Maintaining flexibility of northbound REST API		Performance

Thus, a suitable environment to present the parameters in a way that can be modified in accordance to the benchmark testing requirement and objectives is needed in which one can be obtain toward the contribution of an experimental testbed [19, 20-21].

2. RESEARCH METHOD

2.1. Study Design

This study is conducted under an experimental testbed using emulated virtual devices to determine the difference of performance in term of a set of assessment matrix between reactive forwarding and intent methods. There are two methods of forwarding available in ONOS. One is called simple reactive forwarding application (onos-app-fwd) and the other one is intent-based simple reactive forwarding (onos-app-ifwd). Reactive forwarding refers to the mechanism used for installing network switch forwarding entries. These entries are installed on request after a sender has sent the packets. The following operations are carried out when a packet enters the input interface. 1) The fields of the packet header are evaluated against the table 0. 2) If no match (no table-miss entry) is included, the package is deleted. 3) If no match exists and a table-miss entry exists, perform the defined table-miss action. 4) If the match is correct, update the counters, run the instructions and forwarded to a table in the pipeline or transmitted from the exit port. Figure 3 shows the flowchart of packet flow in reactive forwarding.

Reactive forwarding can be deployed by installing the application onos-app-fwd in ONOS which is used in the experiment. Intent forwarding is an unchangeable model object that describes application's request to change the network's behavior in ONOS that may describe the network resources, constrains or criteria. Figure 4 shows the flowchart of intent forwarding policy in ONOS. Noted that the colored states are transitional and expected to last briefly while the remaining states are space states where the intent may consume time.

The experiment is performed using Ubuntu 18.04 version under localhost environment. The data for this study is collected using experimental testbed with several parameter matrix as shown in Figure 10 using the predetermine assessment parameter. The parameter matrix is consisted of several switches, links, hosts and type of topology. The parameter matrix is then evaluated by the assessment matrix; throughput, RTT (relay time trip), setup, teardown and Pdup (packet duplicates). The testbed is created using Mininet random topology generator script to generate a mesh type topology with a set of parameter number. Figure 5 shows the overview of the testbed.

The script required the imputation of numbers of switches, links and hosts in that order. For each parameter, a predetermine number is imposed under each test before conducting the actual experiment [22].

The parameter matrix is iterated by 5, 15, 10 for each switch, link and host respectively. Figure 6 shows the flowchart of random generator script.

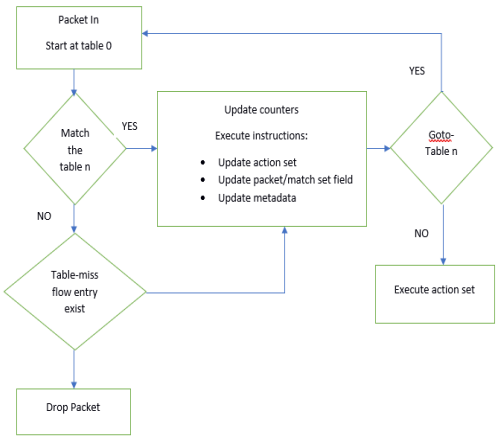

Figure 3. Packets flow in reactive forwarding

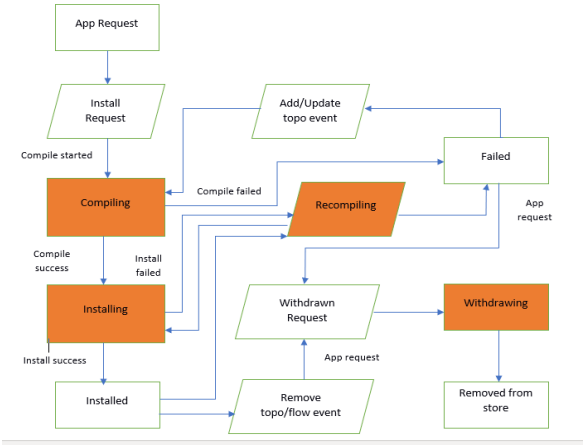

Figure 4. The flow of intents compilation in ONOS

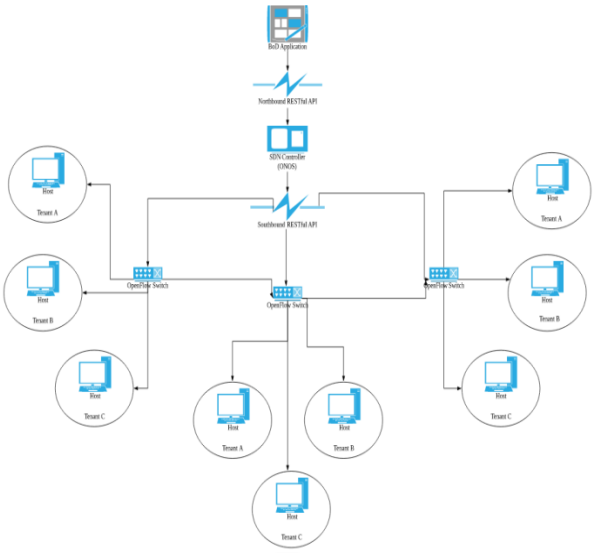

Figure 5. The overview of BoD experimental testbed

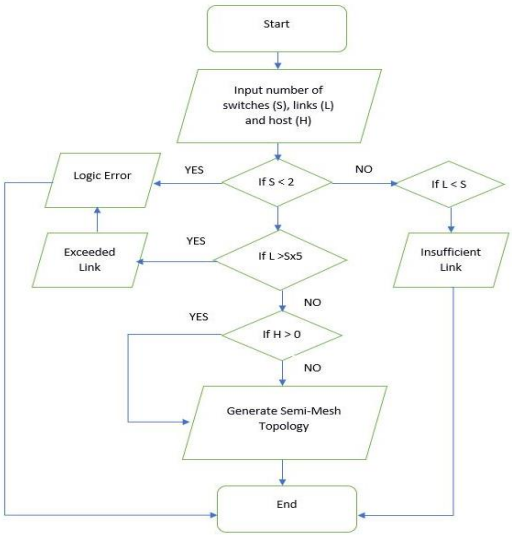

Figure 6. Random topology generator flowchart

The test set is determined by the maximum number of switches which is 25. Once the number of switches reach 25, the host is incremented by 10 and the next test set is conducted. The links is always incremented by 15 according to the incrementation of the switch. The experiment for each technique is considered completed once the number hosts reach 50. It is to be noted that the experiment sets the total maximum number of bandwidths to 50Mbps. This is because the purpose of the experiment is to observe the behavior of the bandwidth with relation to the throughput. Bandwidth can be seen as the total capacity that a link can carry from one point to another whereas throughput is the actual capacity that is successfully achieved from one point to another [23]. This experiment required a virtual separation of network to prevent a host from pinging to each of unrelated group of hosts. This can be achieved by creating a network environment that have a certain number of hosts in their respective tenants. The tenant is intended to act as a separation between assigned host on designated tenant using the technique called network slicing via ONOS. For this experiment, hosts are assumed to be assigned in two tenants; A and B where each of the tenants contain equal half of the number of the hosts. However, this experiment did not showcase a sliced network

environment due to the host able to ping one another regardless of group resulting to void of specific tenant. It is imperative that future experiment implement the function as intended.

3. RESULT AND ANALYSIS

All mininet scripts are placed under one specific folder for easier execution. The scripts are named based on the order of the parameter matrix specification obtained from the random topology generator, e.g. 5-10-10.py (reactive forwarding method for a topology with 5 switches, 10 links and 10 hosts) and i5-10-10.py (intents method for mesh topology with 5 switches, 10 links and 10 hosts). Each of the script contained a set of tests relevant to the purpose of the study. The first test is to obtain the setup time of the topology. The setup time is taken after the first three successful ping which are later averaged. The network connectivity is then tested using the ping all full command. This test is required to determine the RTT of the network via mininet using estimation formula as follow [24]:

$$F\text{^}RTT = \sum\llbracket Avr\_RTT/N\text{^}RTT\rrbracket \quad (1)$$

Where the summation of the average of RTT sample is divided by the total number of RTT occurrence. The result of the final RTT is then averaged. The next test sequence is conducted to determine the throughput of the bandwidth. This is done by using the iperf command from the mininet that specify two hosts in which one act as a server while the other act as the client. The formula for calculating the throughput is as follow [23, 25]:

$$\sum \llbracket Throughput \lessapprox RWIN/RTT\rrbracket \quad (2)$$

Where the RWIN represent the TCP Received Windows to be divided by RTT. The result from this is also averaged. Each of the test is conducted four times to obtain the final average. The test is performed sequentially by following the increasing number of the host using one single command, e.g. 5-10-20, 10-25-20….5-10-30, 10-25-30 and so on.

The experiment started with reactive forwarding method. The mininet is required to be in clean state for every set of tests to free up the port by using the mininet command clean. The intents experiment is commenced once the reactive forwarding experiment is completed. Intents are then being assigned through ONOS via ssh using python scripts to generate intents for every host. The execution command used is similar to the reactive forwarding experiment with the abbreviation of the python named category. The result of each of the experiment is dumped into a text files under one folder as mentioned previously. The result is then recompiled into Excel sheet and the data is sorted into categorical order based on the discussed assessments matrix. All final averaged data is calculated in excel and tabulated.

Table 3 shows a snippet of tabulated data acquired from the reactive forwarding method experiment. The first five column of the table represent its parameter matrix which is consisted of switch, links, hosts, tenants and topology. The first column shows several switches with a starting number of 5 and a maximum number of 25 with the iteration of five in between. The second column shows the number of links required for the formation of mesh type topology. The number is selected based on the successfulness of the random topology generator to generate a semi-mesh type topology. From the test mentioned, the minimum number for the mesh topology to be generated successfully is 25. The first row of the second column which started at 10 is based on the number switches where the first row of the first column started with 5. The maximum number of the links to be generated is 70 with the iteration of 25 in between. The third column shows the number of hosts used in the experiment which is 2. The experiment started with 10 hosts and remained constant throughout the experiment until the first column reaches to number 25. Once the number achieved, the hosts is iterated with 10 while the first and second columns reset back to its initial number which 5 and 10 respectively. The iteration number remained the same as previously mentioned.

Table 3. Result of Reactive Forwarding

Switch	Link	Host	Throughput(mb)	RTT(sec)	Setup(sec)	Teardown(sec)	Pdup
5	10	10	48.33	0.018	0.029	174.963	6
10	25	10	48.35	0.024	0.292	176.072	1796
15	40	10	48.36	0.022	5.040	181.476	64520
20	55	10	48.35	0.027	3.516	189.441	30616
25	70	10	48.35	0.114	3.268	182.404	10658

The placement of the hosts on each switch is determined by the random topology generator which is randomly assigned. The fifth column shows the type of topology used in the experiment. Starting from the forth column to the eighth column, these columns represent the unit of analysis of the assessment matrix; consisting of throughput, RTT, setup, teardown and Pdup. The throughput is measured in megabytes whereas the RTT, setup and teardown are measured in seconds. Except for teardown, the initial data of RTT and setup are in millisecond which later converted into seconds. The Pdup is the number of duplicated packets occurred during the experiment.

Table 4 shows a snippet of the tabulated form of the data obtained from the intents forwarding method experiment. The columns are categorized similar to the reactive forwarding table. Consequently, the purposed and the description of each of the column are the same. From the experiments conducted, the tabulated data of each type of methods can be compared and studied to provide analysis and comprehension on the results obtained. The main focused of the data analysis is to observe the response variables which is consisted of throughput, RTT, setup, teardown and pdup where the supplementary variables which is the parameter matrix is observed. A comparison in a form of graph is generated to determine the relationship between the parameter matrix and the unit analysis of the assessment matrix.

Table 4. Result of Intent Forwarding

Switch	Link	Host	Throughput(mb)	RTT(sec)	Setup(sec)	Teardown(sec)	Pdup
5	10	10	48.37	0.002	0.002	264.444	0
10	25	10	48.37	0.002	0.002	275.992	0
15	40	10	48.36	0.005	0.002	276.458	0
20	55	10	48.37	0.005	0.001	277.116	0
25	70	10	48.41	0.005	0.004	267.345	0

3.1. Data Analysis

In the Figure 7, the throughput of reactive forwarding (RF) and intents forwarding (IF) are being compared. The right axis (vertical) represents the throughput in megabytes (mb), whereas the left axis (horizontal) represent the number of switches, links and hosts with the assigned uppercase letter of S, L and H respectively. Based on the figure, the graph indicates that the throughput of IF yielded the highest amount of achievable throughput which is 48. 51mb out of 50 maximum mb with 5-10-30. The trend shows that whenever the two methods are being compared within the specified number of S, L and H respectively, IF seems to be always better to achieved highest possible throughput. This trend seems to be consistent throughout the duration of the experiment even with a very miniscule difference as showed by the 15-40-10 which is 48.35mb against 48.36mb.

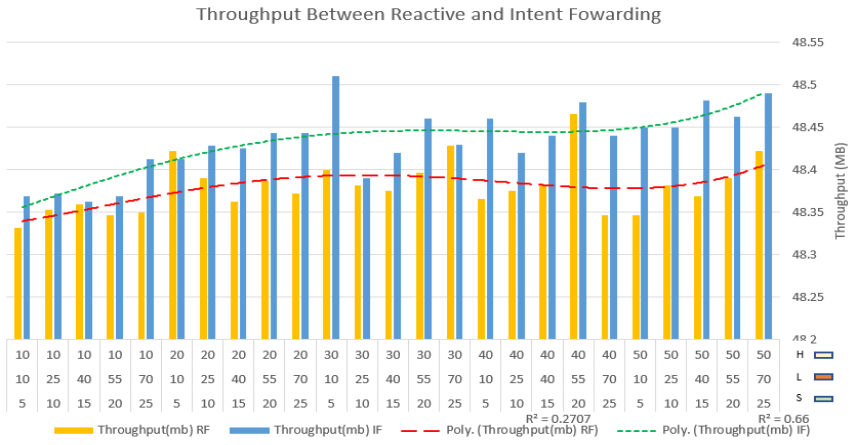

Figure 7. The comparison between reactive and intent forwarding in term of throughput (mb)

As the experiment for each SLH repeated four times before the sum of total averaged of the throughput is taken, this difference showed by the graph could be further widen with more repeatable experiment. The lowest achievable throughput is showed by 5-10-10 RF where it managed 48.33mb followed by 25-70-40 RF and 5-10-50 RF which is both managed 48.34mb respectively. This seems to indicate a

discrepancy between 5-10-10RF, 5-10-50RF and 25-70-40RF. Logically, 5-10-10RF should be higher in achieving maximum throughput as it has the least number of hosts which is 10 compared to 50 hosts with the same number of switches and links even if the difference is in two decimal places. It is the same disparity between 5-10-50RF and 25-70-40RF with a complete difference in term of numbers for each SLH and yet obtained similar throughput. For RF, the highest amount of achievable throughput is 48.46mb under 20-55-40. The chart uses a polynomial fit trend line with the order of 4 to indicate a correlation towards the data obtained from tables.

From the chart, the R^2 of each method is calculated. The calculation shows that the RF method only achieved R^2=0.27 which is weak compare to IF method which is R^2=0.66. Even though the R^2 value of IF is moderate, it is still acceptable. Figure 8 shows the RTT (Round-trip delay time) of reactive forwarding (RF) and intents forwarding (IF) are being compared. The right axis (vertical) represents the RTT in seconds (sec), whereas the left axis (horizontal) represent the number of switches, links and hosts with the assigned uppercase letter of S, L and H respectively. RTT is the term used to describe the total time it takes for a signal to be sent and be acknowledge that the signal has been received. Therefore, it is assumed that the lesser the RTT, the better. Based on Figure 8, it is shown that RF yielded the highest RTT with 0.08 seconds for 20-55-40 followed by the second highest which is 0.07 seconds from 25-70-40.

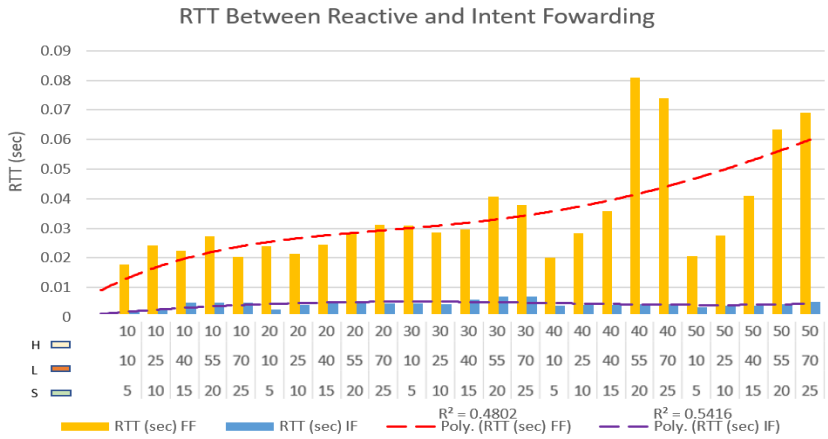

Figure 8. The comparison between reactive and intent forwarding in term of RTT (sec)

The result obtained from IF however, are significantly lower where even the highest RTT for IF is below 0.01 seconds. The highest RTT for IF is 0.007 seconds under both i20-55-30 and i25-70-30 respectively. From the Figure 8, the trend of RTT for RF seems to be steady with an increasing manner. There are some drop of RTT recorded from the RF where the most significant drop happened once the number of switches is reset at 5. Plus, the drop started to show its significant when the number hosts started to reach 40 and above. Therefore, once the switches reset back to the lowest number, the drop happened. The rate of the drop is inversely proportional to the number of hosts. On the other hand, the RTT for IF are vastly different when compare to the RTT of RF. As the result shows, the RTT of IF are all under 0.01 seconds. This shows a great performance in term of response time.

The reason for this is because in RF, the packets are sent in broadcast address. This cause the possibility for the packet to be duplicated as every device discovered in the network are trying to listen. That is why the larger the number destinations needed to be responded, the higher the number of duplicate packets to be produced. This behaviour is observed in RF using Wireshark tool where the duplication of packets is consistently happening. This behaviour significantly affects the RTT due to the filtering of the duplicated packets. Plus, in the experiment, the duplicate packets for the increasing number of hosts managed to reach up to thousands; furthering affecting the RTT of RF. The opposite is observed in IF where the possibility of duplicated to happen are non-existence. The reason why for this outcome is because the intents are consistently being identified by the application that submitted the intent i.e. hosts and its unique intent id generated during the creation of the network topology. After the intents are submitted by the application, it will be sent instantly into a compiling phase before initiating the installing phase to achieve the installed state. That is why a brief moment of time is taken during the first initialisation of network topology as the intents are being compiled and installed. Once the intents are in its installed state, the packet signalling between its destination for the RTT is instantaneous, contributing to the result observed in Figure 8.

Figure 9 shows the setup time (sec) for reactive forwarding (RF) and intent forwarding (IF) are being compared. The left vertical axis represents the setup time seconds whereas the horizontal axis represents the number of switches, links and host; assigned each with the uppercase letter of S, L, H respectively. For this experiment, the setup time is taken to measure the rate of the mesh topology being created based on the influenced of numbers of parameters; switches, links and hosts. The setup time is necessary to be taken into the study because for every performing network topology, the setup time must be fast so that the scalability of the network can be improved. The closer the time of setup with 0.00 second the better. The result would mean the modification of the network or the configuration of the topology can be carried out without effecting the downtime of the network infrastructure.

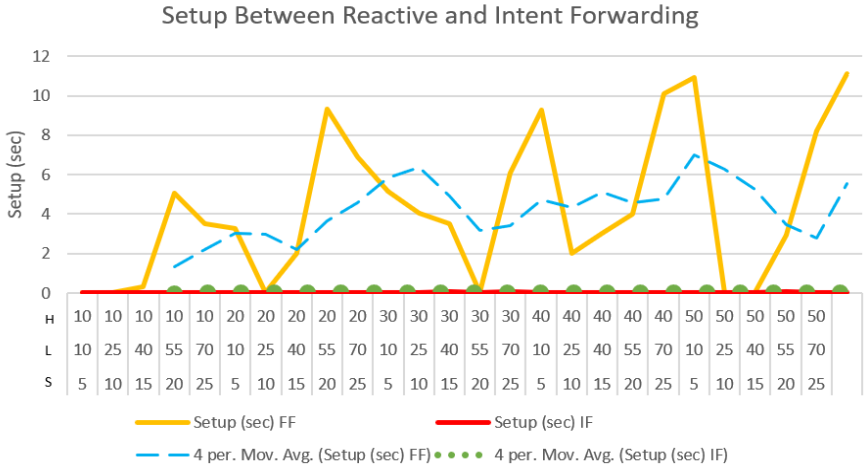

Figure 9. The comparison between reactive and intent forwarding in term of setup (sec)

Based on the Figure 9, the setup time for RF shows a fluttered trend with few spikes as the number of hosts started to increase by 5. The graph shows that the highest peak for RF is 11.09 seconds for 25-70-50. The spike seems to be the highest when the number of links and hosts are at around the number of 55 to 70 for links and 20 to 50 for hosts. The number of switches seems to influence the graph as well as the switches with the number of 25 seems to always associate with highest peak for every set of switches number as described by 15-40-10, 15-40-20, 25-70-30, 25-70-40 and 25-70-50 with the emphasize of 25. The setup time for IF is vastly contrast to that of RF. Based on the Figure 9, the graph for IF shows stabilised trend of setup time with lowest possible value. The lowest recorded time for IF is 0.001 seconds for each i20-55-10, i10-25-40 and i25-70-40. IF setup time constantly hovering between the range of 0.002 to 0.003 seconds with the highest recorded spike of 0.05 seconds. The reason for the major contrast with regards to the setup time of RF and IF is due to the same reason of that stated in RTT.

In RF, the devices used broadcast address to listen to each other devices resulting to the possibility of duplicated packets. Once the duplication of packets occurred, the devices will keep on sending the duplicated packets until a proper connection from the communication of the target device is found. The sending of the duplicated packets usually reached to thousands before receiving the correct packet. The behaviour was observed by using Wireshark during the experiment with RTT and the same behaviour is reflected during the setup time experiment. This in return would lead to severe increased of setup time as the initial point for a setup to be considered as valid is when a proper packet is being received by both targeted devices to establish the first communication. In IF however, the duplicated packets are non-existence due to the consistency of the intents to identify its network resources, i.e. hosts and its intent id; as mentioned during the RTT experiment. Technically the setup time for IF is a lot longer compared to RF if the process for the setup is taken for consideration to the point of the first execution of the topology scripts. This is because before the communication of the devices takes place, the intents need to be compiled and installed which would take more time compared to that of the typical RF method. However, the condition of setup time taken in this experiment is logged during its first successful communication between the devices.

Figure 10 shows the teardown time (sec) for reactive forwarding (RF) and intents forwarding (IF) are being compared. The left vertical axis represents the teardown time in seconds whereas the horizontal axis represents the number of switches, links and host; assigned each with the uppercase letter of S, L, H

respectively. For this experiment, the teardown time is taken to understand how fast the mesh topology created by the script being dissemble for each different method and the relation of the setup and teardown in regard to the scalability. Based on the Figure 10, the teardown time for RF shows a steady trend with minor sudden spikes on two occasions during the experiment.

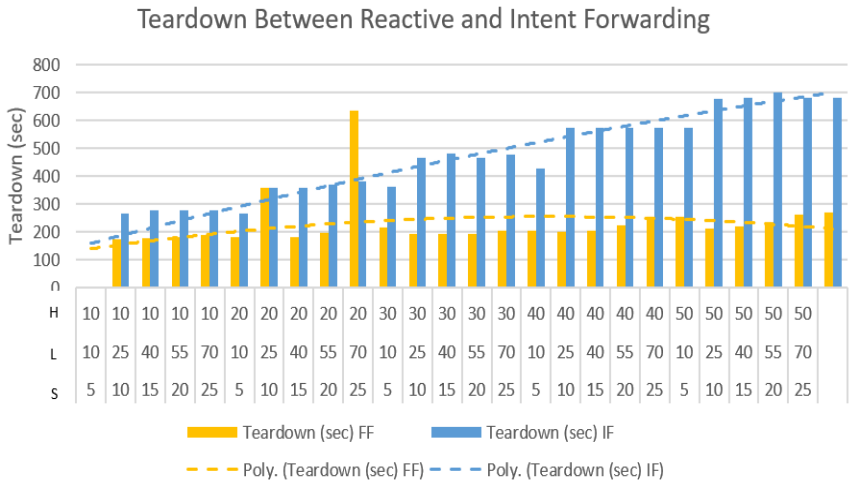

Figure 10. The comparison between reactive and intent forwarding in term of teardown (sec)

Based on Figure 10, the teardown time shows in the graph constantly settled at the range of 170-220 seconds throughout the experiment for RF while spikes occurred at 5-10-20 and 20-55-20 with the value of 358 seconds and 688 seconds respectively. From the graph above, it shows that the range of teardown time for the devices with the number of hosts of 10 and 20 yield around 170 to 190 seconds whereas the devices with the number of hosts of 30,40 and 50 yield around 200 to 260 seconds. The sudden occurrence of the spike seems to be resulted from the unexpected consumption of resources in mininet running on Ubuntu causing the mininet to become unresponsive for a period which long enough to affect the teardown time. On the other hand, the teardown time for IF is higher compared to RF. The teardown of IF shows an upward trend with steady growth. The teardown time seems to be proportional with the number of hosts. As the number of hosts started to increase by 10, the teardown started to add 100 seconds. However, between the transition of the number of hosts, the teardown time of IF seems to be remained in constant. This behaviour of teardown is expected from IF as the intents during the initialisation of network topology needed to be installed and the process consumes times. This in return affected the teardown. Once the intents have been installed, the teardown is always at constant due to the immediate response provided by the intents from the device communication during the setup.

From the results, the difference of throughput for RF and IF are barely negligible due to lack of proper execution for the method of the experimentation. The original idea of conducting the throughput experimentation is to pump the packet simultaneously to stretch the limit of the controller. It is expected that the more host, switches or links would affect the performance of the throughput in both methods. The current version instead determined its throughput by using iperf tool through sequential technique where the recent tested hosts act as client and the server are being tested after the previous test has finished. This in return would not reflect the real situation of the network behavior when the hosts of multiple tenant randomly or simultaneously pump data to the point of affecting the throughput performance. However, for an initial finding, the current experimentation yields enough data to see the small difference of using two types of methods. From the chart, the R^2 of each method is calculated and it is showed that the RF method in term of R value is weak compare to IF method which is moderate. Despite that, during the RTT experiment, the different usage of method showed a significant contrast in the RTT performance. The RF method affect the RTT the most due to duplication of packets occurred almost consistently compared to IF where the intent is instead being used which in return provide persistency in maintaining the network resource. Therefore, when it comes to RTT performance, IF method is superior. Yet when it comes to setup time, the RF technically require less time to setup even though the problem of duplication of packet occurred in which case it would affect the initial time for its device to establish a successful communication. However, the IF method would eventually prevail as it only requires a certain amount of time to install the intents based on

the number of parameters being set to be consistently yield successful communication between the device. In the teardown experiment, even though the IF took the longest time, it is an expected behavior. The number of hosts seems to directly affect the teardown of IF compared to RF where the number of hosts primarily play a negligible role. As a result, even though the teardown of IF is slower compared to RF, the IF's scalability potential is realized. The IF has the trait of possibly allowing a network to be scaled out faster which is sought after by any telecom providers.

4. CONCLUSION

In conclusion, the experiment yielded the initial data for the basis of conducting improved version of experimentation in the future. Regardless of the negligibility found in some of the experiment, the difference between both RF and IF methods are still showed during the experiment. From the data yielded, we can see that IF may play a vital role in improving the scalability of network as showed in the experiments primarily in setup and RTT. The ideal network is to consistently maintain a functioning network infrastructure with the ability to be susceptible with modification and enhancement. Therefore, tearing down a network infrastructure is not profitable compare to adjusting its current state. It is due to that reason that more research needs to be done regarding the scalability of IF based network in to improve the service of network.

ACKNOWLEDGEMENTS

This research study is supported by Universiti Kuala Lumpur Centre of Research & Innovation (CoRI). We thank our esteem colleagues for providing the much-needed insight and expertise which greatly aided the research regardless of the different interpretation may have on this paper.

REFERENCES

[1] Yin, C., Kuo, T. C., Li, T. Y., Chang, M. C., and Liao, B. H. (2014). "*Mediating between OpenFlow and legacy transport networks for bandwidth on-demand Services*". APNOMS 2014 - 16th Asia-Pacific Network Operations and Management. Symposium,6996529.https://doi.org/10.1109/APNOMS.2014.6996529 https://doi.org/10.1145/1721654.1721672

[2] Hakiri, A., Berthou, P., Gokhale, A., \& Abdellatif, S. (2015). "Publish/subscribe-enabled software defined networking for efficient and scalable IoT communications". *IEEE Communications Magazine*, 53(9),48-54. https://doi.org/10.1109/MCOM.2015.7263372

[3] Benzekki, K., El Fergougui, A., \& Elbelrhiti Elalaoui, A. (2016). "Software-defined networking (SDN): a survey". *Security and Communication Networks*, 9(18), 5803-5833. https://doi.org/10.1002/sec.1737

[4] Simon, A. S. (2018). "Bandwidth-on-Demand Motion Control", 26(1), 265-273

[5] Omar, N. (2017). Transport- Software Defined Network T-SDN Specification Requirement for Bandwidth on Demand Service. TMRND, (Julai).

[6] Sadasivarao, A., Syed, S., Pan, P., Liou, C., Monga, I., Guok, C., \& Lake, A. (2013). "*Bursting data between data centers: Case for transport SDN*". Proceedings-IEEE 21st Annual Symposium on High-Performance Interconnects, HOTI 2013, 87–90. https://doi.org/10.1109/HOTI.2013.20

[7] Mendiola, A., Astorga, J., Ortiz, J., Vuleta-Radoiĉić, J., Juszczyk, A., Stamos, K., Higuero, M. (2017). "*Towards an SDN-based bandwidth on demand service for the European research community*". 2017 International Conference on Networked Systems, NetSys 2017. https://doi.org/10.1109/NetSys.2017.7903965

[8] Knoll, T. M. (2014). "*A combined CAPEX and OPEX cost model for LTE networks*". 2014 16th International Telecommunications Network Strategy and Planning Symposium, Networks 2014. https://doi.org/10.1109/NETWKS.2014.6958531

[9] Yassine, A., Shirehjini, A. A. N., \& Shirmohammadi, S. (2016). "Bandwidth On-demand for Multimedia Big Data Transfer across Geo-Distributed Cloud Data Centers". IEEE Transactions on Cloud Computing, X(X), 1. https://doi.org/10.1109/TCC.2016.2617369

[10] OnosProject.org. (2017). "ONOS-A new carrier-grade SDN network operating system designed for high availability", performance, scale-out. Retrieved from https://onosproject.org/

[11] Thyagaturu, A. S., Mercian, A., McGarry, M. P., Reisslein, M., \& Kellerer, W. (2016). "Software Defined Optical Networks (SDONs): A Comprehensive Survey". *IEEE Communications Surveys and Tutorials* (Vol. 18). https://doi.org/10.1109/COMST.2016.2586999

[12] Benzekki, K., Fergougui, A. El, El, A., \& El, B. (2016). "A Secure Cloud Computing Architecture Using Homomorphic Encryption", 7(2), 293-298.

[13] Cai, Z. (2011). "Maestro: Achieving Scalability and Coordination in Centralized Network Control Plane". Rice University, Thesis Degree in Doctor of Philosophy, 102-105.

[14] Shalimov, A., Zuikov, D., Zimarina, D., Pashkov, V., \& Smeliansky, R. (2013). "*Advanced study of SDN/OpenFlow controllers*", Proceedings of the 9th Central \& Eastern European Software Engineering Conference in Russia. Retrieved from https://dl.acm.org/citation.cfm?id=2556621

[15] Armbrust, M., Stoica, I., Zaharia, M., Fox, A., Griffith, R., Joseph, A. D., Rabkin, A. (2010). "A view of cloud computing". *Communications of the ACM*, 53(4), 50. https://doi.org/10.1145/1721654.1721672
[16] Morreale, P., \& Anderson, J. (2014). "Software Defined Networking", (April). https://doi.org/10.1201/b17708
[17] Lian, S., Wang, Y., Han, Y., \& Li, X. (2017). "*BoDNoC: Providing bandwidth-on-demand interconnection for multi-granularity memory systems*". Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC, 738–743. https://doi.org/10.1109/ASPDAC.2017.7858412
[18] Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., \& Lantz B. (2014). "ONOS: Towards an Open", *Distributed SDN OS*, 1-6.
[19] Mueller, J., Wierz, A., \& Vingarzan, D. (2013). "Elastic Network Design and Adaptive Flow Placement in Software Defined Networks".
[20] Zhou, W., Li, L., \& Chou, W. (2014). "*SDN Northbound REST API with Efficient Caches*". Web Services (ICWS), 2014 IEEE International Conference On.
[21] S. González, W. Castellanos, P. Guzmán, P. Arce, and J. C. Guerri, "Simulation and experimental testbed for adaptive video streaming in ad hoc networks," *Ad Hoc Networks*, vol. 52, pp. 89–105, Dec. 2016
[22] D.-D. Chen, Y. C. Lin, and F. Wu, "A design framework for optimizing forming processing parameters based on matrix cellular automaton and neural network-based model predictive control methods," *Appl. Math. Model.*, vol. 76, pp. 918–937, Dec. 2019.
[23] L. Wen, W. Wenbo, J. Xiaojun, and L. Wen, "Optimizing CMT performance by joint predictions of bandwidth and RTT," *J. China Univ. Posts Telecommun.*, vol. 22, no. 4, pp. 81–91, Aug. 2015.
[24] C. Cui, L. Xue, C. H. Chiu, P. Kondikoppa, and S. J. Park, "Exploring parallelism and desynchronization of TCP over high speed networks with tiny buffers," *Comput. Commun.*, vol. 69, pp. 60–68, Sep. 2015.
[25] K. Wada, K. Satsukawa, M. Smith, and T. Akamatsu, "Network throughput under dynamic user equilibrium: Queue spillback, paradox and traffic control," *Transportation Research Part B: Methodological*, Elsevier Ltd, 01-Aug-2018.

BIOGRAPHIES OF AUTHOR

Fathul Arif Kamarudin is a postgraduated student of Universiti Kuala Lumpur under Malaysian Institute Information Technology Internet of Things Research Group. He graduated in Bachelor of Computing in System Security (2017) and currently working as Research Assistant under UniKL-MIIT (2018). Supervised by Dr Megat Norulazmi, Fathul Arif is currently completing his master in Networking.

Megat Norulazmi Megat Mohamed Noor is a Senior Lecturer at the Malaysian Institute of Information System, Universiti Kuala Lumpur. He graduated in Bachelor of Science in Engineering from University of Kagoshima, Japan (1994) and completed a Master of Information Technology at Open University Malaysia (2007). He was awarded a PhD from Universiti Utara Malaysia on 2012 in Information Technology. His recent research interests are Software Defined Network, Network Function Virtualization, Cloud Computing, and IoT. Currently he is the Leader of MIIT's Center of SDN/NFV & IoT. He is an active fellowships member of Okinawa Open Laboratory, Japan

Fuead B Ali is currently a Lecturer at Malaysian Institute of Information Technology, Universiti Kuala Lumpur, since year 2009. Prior to that, he was a Researcher in MIMOS Berhad since 1997. He holds a Bachelor of Science degree in Electrical Engineering from Purdue University, USA.