# SDN Experimental on the PSU Network

Piyawit Tantisarkhornkhet, Warodom Werapun
Department of Computer Engineering,
Faculty of Engineering, Prince of Songkla University
Phuket, Thailand
Email: {5810120060@email.psu.ac.th,
warodom@coe.phuket.psu.ac.th}

Beatrice Paillassa
University of Toulouse, IRIT Laboratory,
INP – ENSEEIHT
Toulouse, France
Email: beatrice.paillassa@enseeiht.fr

***Abstract*— The Prince of Songkla University (PSU) is the top five universities of Thailand that still uses traditional networking. Presently, there are Internet applications that are able to provide dynamic, manageable, and adaptable features such as Software-defined Networking (SDN). SDN is a recent concept of programmable networks that divides the control plane and data plane of all network devices. It can be programmed via an open interface which is interesting to implement SDN because of their various benefits such as centralized network provisioning, lower operating costs etc. In this paper, we propose comparison between traditional PSU network which is defined Static routing (Static routing non-SDN) and SDN which are using programmable algorithm such as Bellman-Ford SDN (BFSDN) unicast, Dijkstra SDN (DSDN) on both unicast and multicast in order to determine worthiness of migration from traditional network to SDN. In part of topology emulator, we have replicated topology by Mininet. Its performance is examined in terms of throughput, latency, jitter, and packet loss.**

***Keywords—Software-defined networking; Unicast; Multicast; OpenFlow; programmable networks; Dijkstra; Bellman-Ford;***

## I. INTRODUCTION

In a traditional network, it is built from different network devices such as hubs, routers, and switches L2/L3 etc. with various protocols implemented on them [1]. Many device types lead to several configuring policies. It makes network administration and performance tuning rather complex. Moreover, the current Internet applications services are more complicate. They are crucial because the Internet is able to handle these new challenges.

Software Define Networking is a new emerging networking approach under network programmable concept. It is an idea to divide the control plane and data plane of all transmission equipment that can be programmed via an open protocol such as ForCES [2], OpenFlow [3], etc. It serves a network manager to be centralized, manage and increase granular security. In a commercial aspect, it allows to reduce operating costs, save hardware and minimize capital expenditures [4]. However, all devices must support SDN protocol likes OpenFlow.

In this paper, we simulate and compare traditional PSU network, BFSDN unicast, DSDN both unicast and multicast in order to explore the value of changing from traditional networking to SDN.

978-1-5090-0629-8/16/$31.00 ©2016 IEEE

The remainder of this paper is structured as follows. In the next section, we present preliminaries on the work, such as Software Defined Networking, Communications and Routing algorithm, Mininet and PSU network topology. In Section III, we illustrate the simulation results. Finally, this paper work is summarized and future work is shown in Section IV.

## II. PRELIMINARIES

### A. *Software Defined Networking*

Software Defined Networking was developed to make innovation convenient and enable facile programmatic control of the network data-path. It isolates controller hardware from forwarding hardware. The SDN structure enables the combination of transmission equipment which is easy policy management.

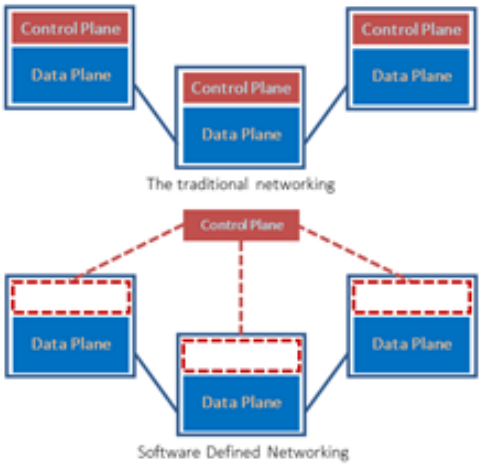


Fig. 1. Traditional network and Software-Defined Network

The SDN architectures, There are ForCES and OpenFlow. They are the basic SDN standard of isolation between the control and data planes. They standardize messages exchange between the planes. In this paper, we focus on OpenFlow architecture because it is popular than ForCES. SDN-OpenFlow architecture composes of application layer, control layer, and infrastructure layer.

- **Application layer or management layer** is the top layer. It consists of network operation tools and user interfaces that management of the network via the control layer by OpenFlow protocol.
- **Control layer** is the middle layer using for control all devices in infrastructure layer via OpenFlow protocol. SDN controller such as NOX/POX [5], Ryu [6], Floodlight [7] etc.
- **Infrastructure layer** is the lowest layer. It includes all devices in traffic forwarding that all devices must support OpenFlow protocol.

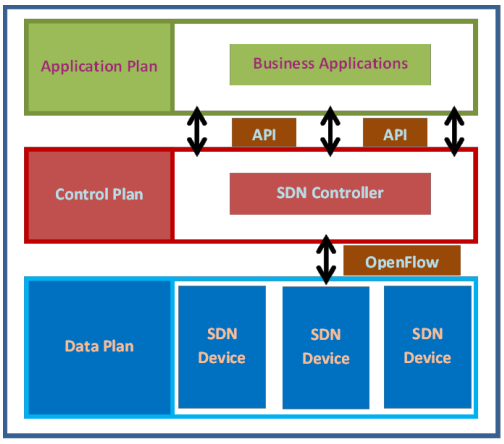


Fig. 2. The SDN architecture

OpenFlow is a flow-based switch specification designed to enable researchers to run experiments in live networks. It is depended on a simple Ethernet flow switch which exposes a standardized interface for adding and removing flow accesses. OpenFlow protocol is an open protocol that is set between the control plane device and the data plane device [8].

## B. *Communications*

In this section, two types of communications as unicast and multicast are reviewed. They have advantages and disadvantages which are depended on situation of communication.

### *1) Unicast*

Unicast is a type of communication where data is sent from one source to one destination. Unicast uses protocol of transport layer such as Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), which are session-based protocols. In this work, we use Dijkstra shortest path and Bellman-Ford shortest path which are popular algorithms [9].

#### *a) Dijkstra's algorithm*

The Dijkstra method is algorithm for finding the optimum path between nodes in a directed graph is G= (V, E), denote: V is the set of nodes and E is the set of edges. Dijkstra's algorithm shows in Fig 3, Primarily, distance at source (dis[s]=0), distance at u (dis[u]=∞) when u≠s, and previous distance at u (pre[u]=null).

Dijkstra's Algorithm

Input: G, s
Output: dis[V], pre[V]

```
1: for each v in G(V)
2:     dis[v] ← ∞
3:     pre[v] ← null
4: insert distance at u node into Q set
5: while (Q!=null)
6:    u← Minimum_distance(Q)
7:    for each v of u
8:       if dis[v] > dis[u]+ew[u,v] then
9:          dis[v]←dis[u]+ew[u,v]
10:         pre[v]←dis[u]
```

Fig. 3. The Dijkstra's Algorithm

In [10] Jehn-Ruey Jiang and team propose Extending Dijkstra's algorithm that adds the node weight analysis in their algorithms.

#### *a) Bellman-Ford's algorithm*

Bellman-Ford method is an algorithm to search for the optimum path between nodes in a directed graph likes Dijkstra's algorithm. However, it is able to analyze negative edge weights. It provides more features than Dijkstra algorithm. In contrast, if Bellman-Ford is used over non-negative weights, its performance is slower than Dijkstra. From Fig. 4, shows Bellman-Ford's algorithm which consists of three main steps as initialize graph, Relaxation of edges and checking negative cycle.

Bellman-ford's Algorithm

Input: G, s
Output: d[V], p[V]

```
1) Initialize graph
1: for each v in V
2:         dis[v] ← ∞
3:         pre[v] ← null
2) Relaxation of edges
1: for i from 1 to IVI-1
2:         for each [u,v] with ew in E
3:                 if dis[u]+ew < dis[v]
4:                         dis[v] ← dis[u]+ew
5:                         pre[v] ← u
3) Checking negative cycle
1: for each ew(u ,v) in E
2:         if d[u]+ew < dis[v]
3:         print "Negative cycle"
```

Fig. 4. Bellman-ford's Algorithm

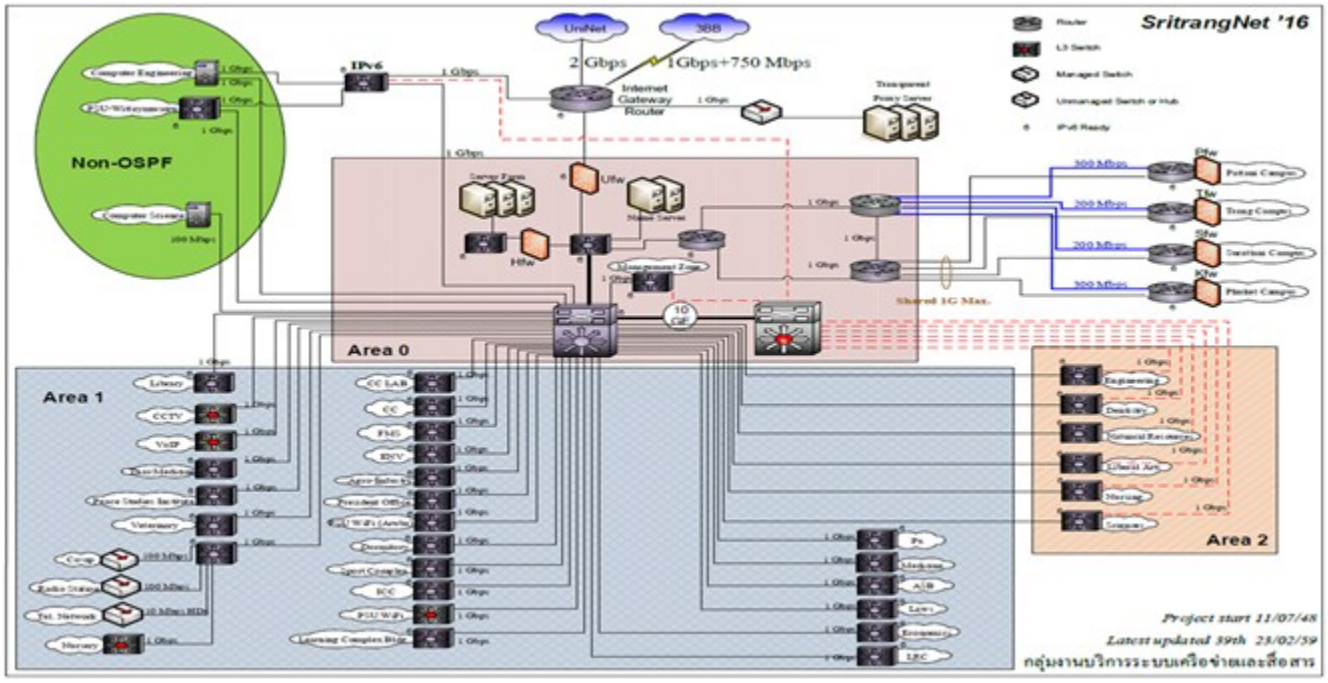


Fig. 5. PSU network topology

*1) Unicast*

Multicast is the communication network ability of sending IP packets to member groups. It can reduce transmission overhead from an application to send copies of data to all group members. An advantage over than unicast is to minimize network traffic and optimize bandwidth where multiple users are accessing the same live video source.

GroupFlow [11] is one SDN-driven approach to IP multicast routing that has already been proposed. It is based on CastFlow [12]. SDN controller of GroupFlow builds on the POX framework. Routing protocol of GroupFlow is Protocol Independent Multicast (PIM) which constructs shared trees centered at a rendezvous-point (RP) router. The PIM, there are two varieties as PIM dense mode and PIM sparse mode only establishes routes to the designated router (DRs) reactively to group joins.

There is some difference between multicast non-SDN and multicast SDN. Multicast SDN can add more membership security to the network which is a normal problem in multicast non-SDN. In SDN, a client joins a multicast group, its membership is assumed to authenticate at a server to a controller. GroupFlow (Fig. 6) adds more membership security in IGMP JOIN phase. GroupFlow multicast model uses Dijkstra's algorithm to compute a shortest path from the multicast source to all multicast members in the network. Each shortest path is added to multicast tree construction.

## C. Tools

Mininet [13] is a topology emulator program that supports SDN based on Linux kernel. It can create network virtual devices as hosts, OpenFlow switches, legacy switches, routers, links, and controllers. It is suitable for researching, testing, and debugging etc. Mininet brings many scripts with user-friendly GUI. In addition, BRITE [14] has been used to create our tested topologies. It supports multiple generation models and can be used as input topologies in Mininet. Although the real network devices would have some cross-traffic, CPU and bus architectures specific to each hardware component that Mininet does not have, Mininet reveals the trend of performance efficiently.

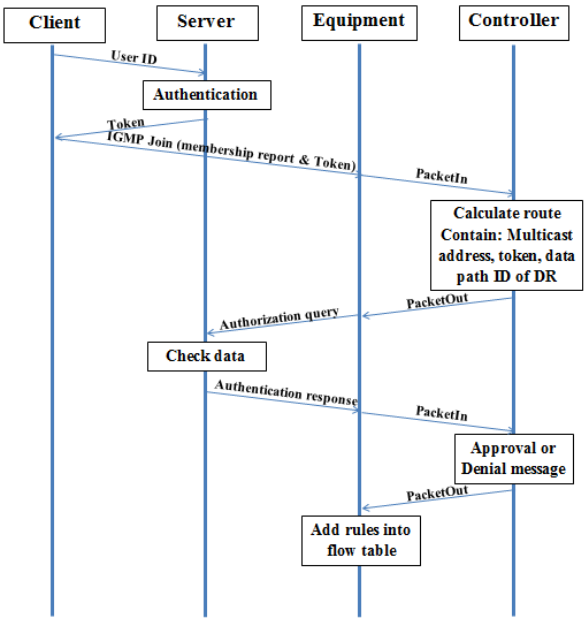


Fig. 6. IGMP Join of GroupFlow model

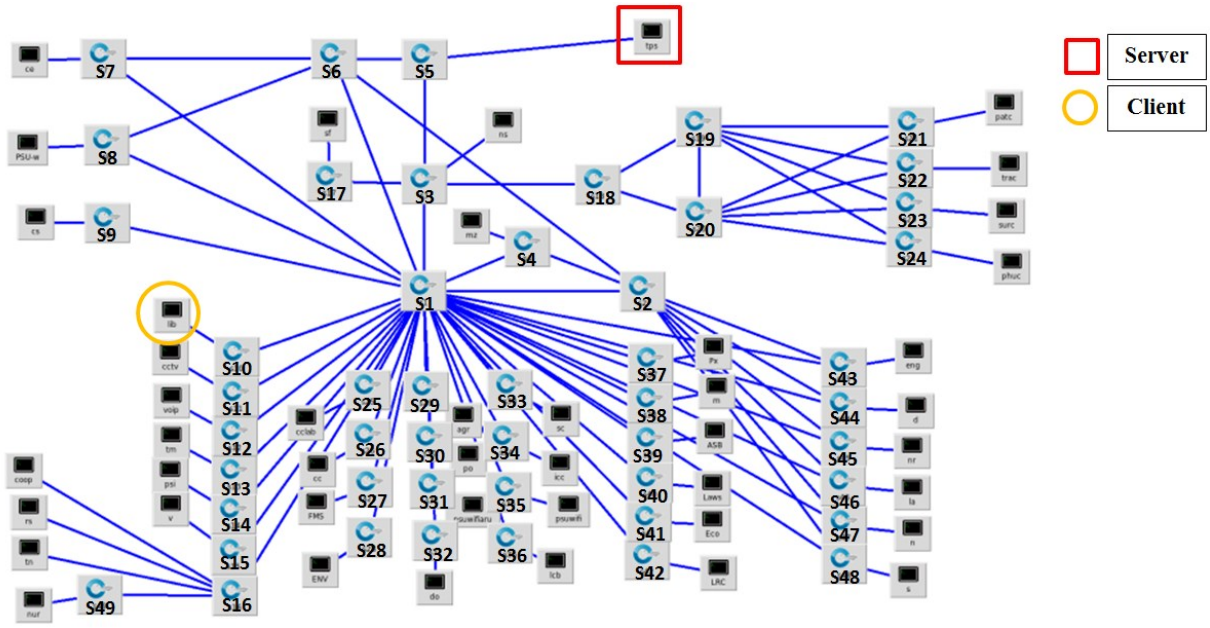


**Fig. 7. PSU Topology on Mininet**

### *D. PSU network topology*

The Prince of Songkla University (PSU) uses traditional network that is the extended star topology. This real existed topology can be a representative of some organizations or companies on a similar scale. Fig. 5 shows PSU topology [15], it consists of different transmission node types such as router, L2 switch, managed switch and unmanaged switch or hub. Most bandwidth between L2 switches is 1 Gbps. We present two previously described transmission types such as unicast and multicast to test network performance since the PSU network topology has various transmission types.

## III. EXPERIMANTAL RESULTS

We set up POX [16] OpenFlow controller and PSU topology in the Mininet while all switches are linked to the controller. Fig. 7 shows a PSU network topology which is simulated from Mininet. It provides most 1 Gbps links between peripheral nodes to the central nodes [15]. For assess the performance of the PSU networking used SDN and Static non-SDN, we use Iperf [17] to test them such as throughput, latency, jitter and packet loss. Experiments are conducted on a single computer with CPU Intel ® core TM i7-4790 3.60 GHz processor and DDR3 8 GB RAM.

TABLE I. SIMULATION SETTINGS

| Parameter | Value |
|---|---|
| Hosts | 45 |
| Nodes | 49 |
| Links | 109 |
| Test times | 100 seconds |
| Controller | POX 0.2.0 support OpenFlow 1.0 |
| Test tools | Mininet |

In the simulation experiments, we measure the performance for Static routing non-SDN and routing algorithm SDN. We use Iperf to create TCP and UDP. We assign one server (circle) and one client (square) because they have many paths such as s10-s1-s3-s5, s10-s1-s6-s5 and s10-s1-s7-s6-s5 etc. Their minimum bandwidth is 1 Gbps.

Fig 8 shows throughput of TCP. The results of Static routing non-SDN, BFSDN unicast and DSDN unicast are 943.25 mbps, 954.50 mbps and 954.69 mbps, respectively. DSDN unicast is slightly better than BFSDN unicast. Static routing non-SDN has lowest throughput. Therefore, BFSDN unicast and DSDN unicast manage bandwidth better than Static routing non-SDN in TCP transmission.

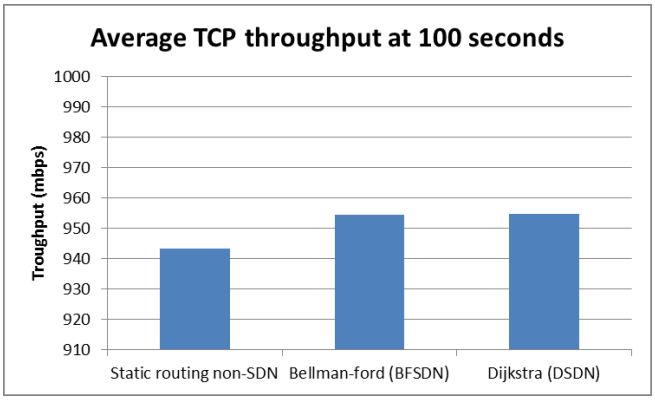


Fig. 8. The average TCP throughput

Fig 9 shows latency of Static routing non-SDN, BFSDN unicast and DSDN unicast by executing Ping command which sends ICMP request messages. Their average latencies are 0.0907 ms, 0.3934 ms and 0.3785 ms respectively. Comparison between algorithms related to SDN, BFSDN unicast latency is

larger than DSDN unicast latency, which is usually caused by the depending inverse throughput. Static routing non-SDN has the lowest latency. Both BFSDN unicast and DSDN unicast on the SDN controller have to take the time calculating the route and install suitable paths on OpenFlow switches. Therefore, the data transfer increases and occurs between nodes in a specific time period for SDN than traditional network.

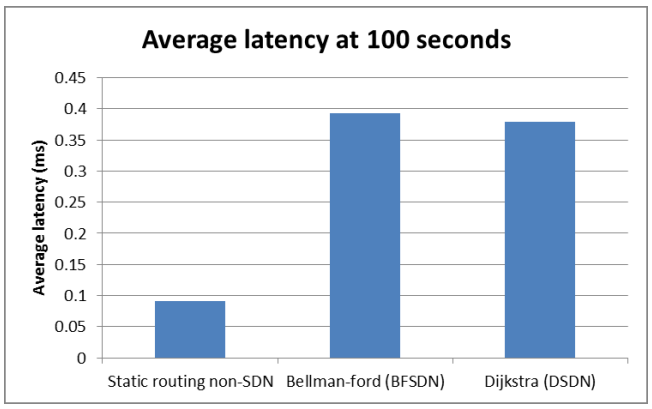


Fig. 9. The average latency

Their initial latency values are 0.9227 ms, 32.8987 ms and 31.7150 ms respectively. When we calculate average latency without an initial value as shown in Fig. 10, their average latency will be 0.0823 ms, 0.0651 ms and 0.0620 ms. Static routing non-SDN has worse latency than others. The initial latency values of SDN affects to average latency. In addition, we examine interval time that SDN has average latency lower than Static routing non-SDN by changing the experiment time from 100 seconds to 2000 seconds. The average latency of DSDN unicast and BFSDN unicast are lower than Static routing non-SDN at 990 seconds and 1619 seconds respectively. However, we do not include them in the figure since it would make the data impossible to read. Therefore, SDN is greater than Static routing non-SDN in several performance metrics on the long term used.

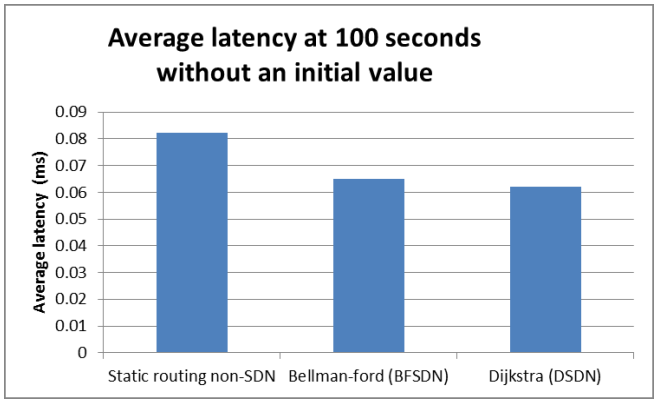


Fig. 10. Average Latency without an initial value

The result of UDP transmission as shown in Fig 11-13, each bandwidth requirement (x-axis) takes 100 seconds for experiment time. Static routing non-SDN, BFSDN unicast, DSDN unicast and Dijkstra multicast SDN (GroupFlow) have been tested for throughput of UDP as shown in Fig 11. All routing are fully well performed until BFSDN unicast began down and stable at 750 mbps, DSDN unicast began down and stable at 800 mbps and Static routing non-SDN and GroupFlow began down and stable at 850 mbps.

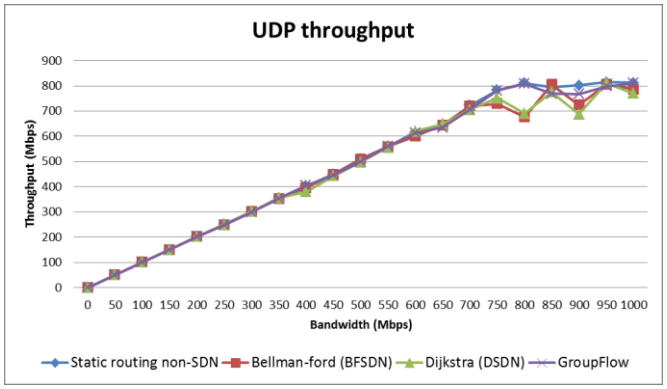


Fig. 11. The UDP throughput

Main factors of jitter and packet loss are depended on bandwidth congestion. At the maximum bandwidth, BFSDN unicast, DSDN unicast and GroupFlow (SDN-related) have high jitter as shown in Fig. 12. Average jitter algorithm are ranked as follow as BFSDN unicast, DSDN unicast and GroupFlow. Static routing non-SDN has the lowest jitter. Therefore, Static routing non-SDN stably runs at fully bandwidth congestion.

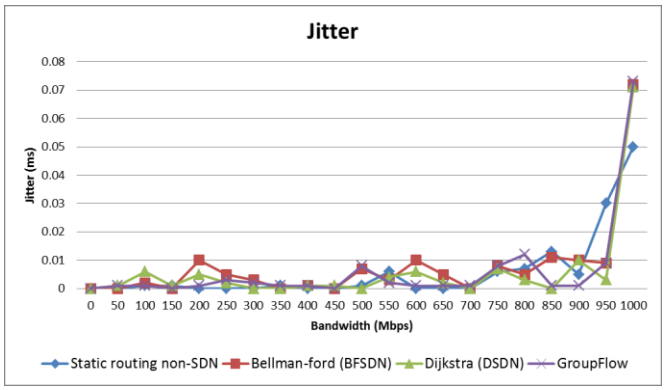


Fig. 12. The UDP jitter

Packet loss is measured to verify the performance of different bandwidth such as jitter. However, it has another factor, called the quality of the network device installation, that affected to the packet loss rate as shown in Fig. 13. The simulation results illustrate that the order of packet loss rate are DSDN unicast, BFSDN unicast, GroupFlow and Static routing non-SDN respectively. Static routing non-SDN slightly swings and has the lowest average packet loss compared with other routing algorithms. BFSDN unicast, DSDN unicast and GroupFlow have the packet loss rate lightly different. They begin to swing a value at 450 mbps and increasing continuously. At high bandwidth usage, we can see different packet loss rate between SDN and Static routing non-SDN obviously since transmission of SDN requires controlled messages by connected switches (a sender side) to a centralize controller while Static routing non-SDN has a control plane

inside a device. Therefore, Static routing non-SDN has an advantage on packet loss for fully bandwidth usage because SDN has more network installation steps due to the separating of the control plane and data plane. However, SDN packet loss percentage is fallen in medium category of phone quality standard [18] which is still adequate.

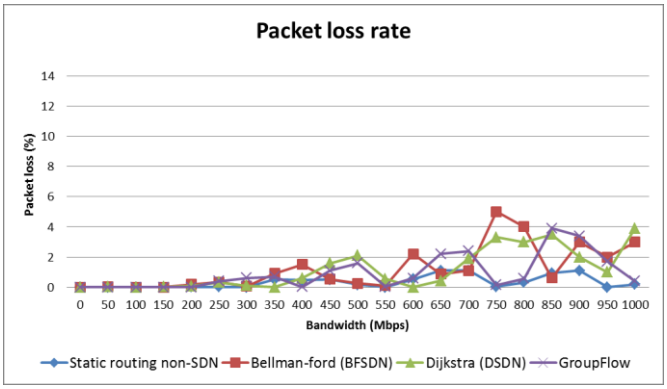


Fig. 13. Packet loss rate

## IV. Conclussion and Future Work

The Software Defined Network is an efficiency programmable network which has more advantages than traditional network such as SDN handles data from a centralize controller, SDN allows reducing deployment configuration, SDN reduces the amount of spending required on infrastructure etc. Thus, SDN is an interesting network system that determines worthiness of migration traditional network.

In this paper, we use POX to implement BFSDN unicast, DSDN unicast, and using GroupFlow multicast model to compare performance analysis with Static routing non-SDN in terms of throughput, latency, jitter, and packet loss under the PSU network topology simulation. Not only SDN benefits previously stated above but also the experiment results show that DSDN unicast and BFSDN unicast have more outperformed than Static routing non-SDN in term of TCP throughput. DSDN and BFSDN unicast have latency lower than Static routing non-SDN when transmission time is over than 990 seconds and 1619 seconds respectively.

Furthermore, we found that the quality decreases at high bandwidth usage since packets are sent to a receiver while controlled messages are required sending to a centralized controller. Thus, SDN has more bandwidth congestion than Static routing non-SDN at same bandwidth requirement. However, SDN offers suitable quality of service requirements such as chat, video streaming, video on demand, video over IP, and video conferencing. The result of this research shows that replacing a traditional network by SDN has benefits. Moreover, we are able to develop various applications which can be used to manage and control network effectively. Eventually, the contribution of this paper is to examine the benefits of migrating from traditional network to SDN-aware network.

For the future work, we plan to conduct simulation experiments for video streaming unicast and multicast transmission over real topologies and build horizontal scaling devices. We are going to simulate edge weights which may be changed during the experimental run and measure overhead of OpenFlow messages at a central controller.

## Acknowledgment

This research has been supported by the Engineering Post-Graduate Scholarship, Faculty of Engineering, Prince of Songkla University, Thailand.

## References

[1] B. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, “A survey of software-defined networking: Past, present, and future of programmable networks,” *Commun. Surv. Tutor. IEEE*, vol. 16, no. 3, pp. 1617–1634, 2014.

[2] A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern, “Forwarding and control element separation (ForCES) protocol specification,” 2010.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: enabling innovation in campus networks,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[4] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-defined networking: A comprehensive survey,” *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[5] “noxrepo (NOX Repo),” *GitHub*. [Online]. Available: https://github.com/noxrepo. [Accessed: 10-Jul-2016].

[6] “osrg/ryu,” *GitHub*. [Online]. Available: https://github.com/osrg/ryu. [Accessed: 10-Jul-2016].

[7] “floodlight/floodlight,” *GitHub*. [Online]. Available: https://github.com/floodlight/floodlight. [Accessed: 10-Jul-2016].

[8] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, “Are we ready for SDN? Implementation challenges for software-defined networks,” *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 36–43, 2013.

[9] S. Upadhaya and G. Devi, “Characterization of QoS based routing algorithms,” *Int. J. Comput. Sci. Emerg. Technol.*, vol. 133, 2010.

[10] J.-R. Jiang, H.-W. Huang, J.-H. Liao, and S.-Y. Chen, “Extending Dijkstra’s shortest path algorithm for software defined networking,” in *Network Operations and Management Symposium (APNOMS), 2014 16th Asia-Pacific*, 2014, pp. 1–4.

[11] “alexcraig/GroupFlow,” *GitHub*. [Online]. Available: https://github.com/alexcraig/GroupFlow. [Accessed: 10-Jul-2016].

[12] “caioviel/CastFlow,” *GitHub*. [Online]. Available: https://github.com/caioviel/CastFlow. [Accessed: 10-Jul-2016].

[13] M. Team, *Mininet: An instant virtual network on your laptop (or other PC)*. 2012.

[14] “BRITE: Boston university Representative Internet Topology gEnerator.” [Online]. Available: http://www.cs.bu.edu/brite/. [Accessed: 10-Jul-2016].

[15] “PSU Network Diagram.” [Online]. Available: http://netserv.cc.psu.ac.th/images/phocadownload/blackbone/psunet.png . [Accessed: 10-Jul-2016].

[16] “noxrepo/pox,” *GitHub*. [Online]. Available: https://github.com/noxrepo/pox. [Accessed: 10-Jul-2016].

[17] “iPerf - The TCP, UDP and SCTP network bandwidth measurement tool.” [Online]. Available: https://iperf.fr/. [Accessed: 10-Jul-2016].

[18] W. Sugeng, J. E. Istiyanto, K. Mustofa, and A. Ashari, “The Impact of QoS Changes towards Network Performance,” *Int. J. Comput. Netw. Commun. Secur.*, vol. 3, no. 2, pp. 48–53, 2015.