Performance Analysis of Single and Multi-controller using OpenFlow Protocol for Software Defined Networks

Layth A Khalil Al Dulaimi 1, R Badlishah Ahmad2, Naimah Yaakob1, Israa Sh Ahmed3 and Mohamed Elshaikh1

1 Embedded Networking and Advance Computing Cluster, School of Computer & Communication Engineering, Universiti Malaysia Perlis (UniMAP) Tingkat 1, Kampus Tetap Pauh, Putra 02600 Arau, Perlis.

2 Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, 21300 Kuala Terengganu, Terengganu, Malaysia.

3 Centre of Excellent Advanced Communication Engineering, School of Computer & Communication Engineering, Universiti Malaysia Perlis (UniMAP), 01000 kangar, Perlis.

laythadnan@studentmail.unimap.edu.my

Abstract. A Software Defined Network (SDN) is a relatively advanced way to implement communication networks. The decision making SDN separates the control layer, which determines where the packets are sent, from the basic infrastructure (also known as the infrastructure layer or data plan) which restores the packets to the specified destination. The newly emerging standard for enactment is the OpenFlow protocol, which contains a standard protocol for communications between the control layer and the infrastructure layer. This study analyzes the impact of OpenFlow single and multi-controllers on the performance of the OpenFlow protocol network. The research was conducted using the discrete event network simulation, omnet++, and involved analyzing the main network metrics including End to End Delay and Throughput. The results indicate that the type of controller used has a clear impact on the performance of the network.

1. Introduction

Widely used protocols in the Internet, known as TCP/IP, are a form of communication, basic contact language and protocols used to connect hosts to the Internet. The TCP/IP protocol is currently the best know network due to the successful development of Internet networks. It is therefore useful to study the behavior and efficiency of this protocol further by taking advantage of a simulation.

The need for new techniques and mechanisms requires the development of a new type of network that can be programmed for different needs and purposes by software applications based on the network operating system. It has now become easier to structure the current network and protocol designs due to the development of a new protocol called OpenFlow. OpenFlow is the first unified communications interface to be positioned between redirection and the layered structure controls of software defined network architecture. SDN is a computer network approach that enables network administrators to manage the services provided by the network by extracting higher-level functionality.

SDN has developed the OpenFlow protocol because it enables flexible control of the network devices that serve as to exchange messages between an OpenFlow controller layer and an OpenFlow switches layer [1]. Furthermore, the procedures for organizing data transfer between network elements occur at different layers in the OpenFlow protocol which can enhance performance efficiency, provide high consistency and extra connectivity, and ensure strict data protection standards.

For any new protocol in the network, there are several methods that can be used to test acceptability, evaluation and performance. One approach is to simulate a range of advantages such as cost creation, scalability, flexibility, accessibility, redundancy (for several purposes) and a simulation speed that is often faster than in real time. Easily adjustable simulation speed can be slower or faster than real time, as required. The method can run on real devices, applications and operating systems and thus offers test results that are highly realistic, however. such experiments are expensive to build. Therefore, it is...
very helpful to use an eraser whereby any possible scenario can be applied and, when it cannot access operation and software systems, it can be changed in any of the systems or applications as needed. Network companies have attempted to deal with distributed environments through the development of new technical solutions for routing protocols. However, there is limited information on the performance of each of these companies and there are no realistic performance comparisons available on a large scale. Consequently, very few performance ratings for OpenFlow protocols and its behavior in various domains using the OMNeT++ have been provided. Simulation tools can provide an appropriate environment for the design, construction and testing of different scenarios for users, offering practical feedback and near-reality when developing real-world systems. This allows system designers to test and determine the validity and efficiency of the design before the system is deployed. A notable feature of the simulation is that it facilitates the evaluation of different network metrics using various verification mechanisms, producing results that cannot be empirically measured on the largest geographically distributed architecture.

2. Related works
This paper focuses on the assessment and analysis of network performance and results using an OMNeT++ simulator. OMNeT++ provides many advantages as a form of network simulation and has been widely used in academic research. It is capable of simulating manufacturing, airports, crowd management and estimating weather. OMNeT++ is scalable whereby all modules are implemented as C++ modules and are linked together in a single process [2]. Furthermore, OMNeT++ can modify parameters such as link bandwidth and delay as well as the configuration of network size, mobility pattern or speed for correcting performance results. When time is a concern in OMNeT++ the performance results need to be repeated for correction and accuracy. OMNeT++ supports the OpenFlow network as an extension of the INET framework including the spanning tree protocol (STP).

Regarding the OpenFlow protocol, the OMNeT++ has many limitations, one of which is that it is not new as the latest version of OpenFlow is 1.4 while, in this paper, OpenFlow switch specification 1.2 will be used [3]. The OMNeT++ uses C++ modules with its simulation engine code as well as all devices and objects, and is a user-level executable program that operates exactly like a ns-3 network simulator [4]. The NOX OpenFlow controller is a user-level program, however OMNeT++ and NOX cannot be compiled and linked together to form a single executable program [5]. OMNeT++ supports several functions such as STP, however STP is not supported by ns-3 as there is no TCP connection between simulated hosts. When using a TCP connection in a real model the results are packet losses and congestion [6].

The simulated OMNeT++ module does not connect to a real OpenFlow controller and thus does not serve as an external entity for measuring the effects of the OpenFlow protocol, whereas Mininet and EstiNet can do so. The network emulator separates namespaces such as Mininet, which reduces the overheads of the system rather than acting as a single simulation process. Mininet emulated hosts use a virtualization approach and can run real applications and exchange information [7]. The network emulator gives unpredictable outcomes that yield different experimental results on each run because the emulated hosts run on CPU speed, current system activities, system load, memory bandwidth, number of emulated hosts and multiplexing over the CPU [7]. Mininet needs to run up a shell process to emulate each host using a user-space or kernel-space (OpenVswitch) to simulate each OpenFlow switch. Thus, Mininet is less scalable than EstiNet, ns-3 and OMNeT++ [8]. Mininet can only be used to study the behavior of the emulated hosts, it cannot be used to study the duration of network/application performance. Mininet GUI can be used to observe the packet playback of a simulation run, where the user needs to write Python code to set up and configure the simulation case. However, OMNeT++ has a GUI, which can be used for the observation of results whereby users need to write C++ code to set up and configure the simulation case. On balance, it is better to use OMNeT++, even it takes time and effort to create simulations, as once modules are created it is much easier to then create new ones [9].
3. Simulation setup for experimentation
The simulation scenarios of the OpenFlow module can be viewed in both a single controller and a multi-controller to analyze the results. This will help determine which model behaves more efficiently and more quickly.

A. Simulation Scenarios
In this assessment, we use the same topology for both single controller and multi controller networks. These represent Malaysian states as shown in Figure 1 and Figure 2.

![Figure 1. Single Controller](image)

This topological diagram of Malaysia contains 14 locations, Representation of state capitals are places where, in the sole observer scenario, the OpenFlow protocol is applied in single control layer scenarios.

Where there is single control for the whole of Malaysia in the OpenFlow protocol, the protocol is connected to the infrastructure layer of each state capital. The capitals of the Malaysian states have been linked according to the calculated distance between themselves in km by using an optical cable for the simulation. The parameters of the simulation program were set up using accurate measurements of the distance between states as shown in Table 1.

| Locations | Distance (km) |
|-----------|--------------|
| ofS_Penang.gate++ <-> DistanceChannel | distance = 47km |
| ofS_Kedah.gate++ <-> DistanceChannel | distance = 171km |
| ofS_Pulau_Pinang.gate++ <-> DistanceChannel | distance = 442km |
| ofS_Terengganu.gate++ <-> DistanceChannel | distance = 360km |
| ofS_Selangor.gate++ <-> DistanceChannel | distance = 367km |
| ofS_Perak.gate++ <-> DistanceChannel | distance = 277km |
| ofS_Pahang.gate++ <-> DistanceChannel | distance = 367km |
| ofS_Selangor.gate++ <-> DistanceChannel | distance = 411km |
| ofS_Putrajaya.gate++ <-> DistanceChannel | distance = 56km |
| ofS_Neheri_Sembilan.gate++ <-> DistanceChannel | distance = 322km |
| ofS_Neheri_Sembilan.gate++ <-> DistanceChannel | distance = 322km |
| ofS_Sarawak.gate++ <-> DistanceChannel | distance = 1038km |
| ofS_Sabah.gate++ <-> DistanceChannel | distance = 734km |
| ofS_Johor.gate++ <-> DistanceChannel | distance = 346km |
| ofS_Kelantan.gate++ <-> DistanceChannel | distance = 410km |

Table 1 Distances between Malaysian states in Single Controller.
Table 2 shows the distances between the Malaysian states in the multiple control scenario where each state is given a control and the states are associated with each other through the control units in the multiple controller layer. Each state has a structure for the SDN and the infrastructure layer is managed by a single control for the same state.

**Table 2 Distances between Malaysian states in the Multi-Controller.**

| Distance between States | Distance (km) |
|-------------------------|---------------|
| Perlis.gate++ <-> DistanceChannel | 47 |
| Kedah.gate++ <-> DistanceChannel | 171 |
| Pinang.gate++ <-> DistanceChannel | 161 |
| Perak.gate++ <-> DistanceChannel | 204 |
| Perak.gate++ <-> DistanceChannel | 346 |
| Kelantan.gate++ <-> DistanceChannel | 161 |
| Selangor.gate++ <-> DistanceChannel | 277 |
| Selangor.gate++ <-> DistanceChannel | 41 |
| Pahang.gate++ <-> DistanceChannel | 32 |
| Pahang.gate++ <-> DistanceChannel | 226 |
| Selangor.gate++ <-> DistanceChannel | 86 |
| Melaka.gate++ <-> DistanceChannel | 225 |
| Johor.gate++ <-> DistanceChannel | 323 |
| Johor.gate++ <-> DistanceChannel | 734 |
| Sarawak.gate++ <-> DistanceChannel | 1038 |
| Kuala Lumpur.gate++ <-> DistanceChannel | 69 |
| Putrajaya.gate++ <-> DistanceChannel | 36 |
| Putrajaya.gate++ <-> DistanceChannel | 56 |
| Putrajaya.gate++ <-> DistanceChannel | 137 |

**B. Performance Metric**

The performance measure of the assessment is the mean End to End Delay for the nodes and Throughput in the investigated networks. In single control, the main server will be in Putrajaya to generate traffic for the network. This will be achieved by changing the size of the message length packet (0.5k, 1k, 1.5k, 2k, 2.5k) and (5k, 10k, 15k, 20k, 25k) for network efficiency in both single control and multi controller cases. The simulation was run for 300 seconds to enable measurements to be taken for the OpenFlow protocol.
4. **Analyzing the performance scenario**

End-to-End Delay refers to the time taken for a packet to be transmitted across a network from source to destination. It is a term commonly used in IP network monitoring, and differs from round-trip time (RTT) in that only the path from source to destination is measured [10]. 1k = 1024 kilobits. Figures 3 and 4 show the End-to-End Delay for networks using a Single controller and a Multi-controller.

![Figure 3. End-to-End Delay for network in Single Controller and Multi Controller](image)

The difference between the use of a single or multi-controller is extremely close, as shown in Figure 3. This behavior illustrates the nature of the network if designed using one of the mechanisms and provides useful information for designers of the network on End-to-end delay.

![Figure 4. End-to-End Delay for network in Single Controller and Multi-Controller](image)

It is also notable that, when traffic is generated in the form (5K, 10K, 15K, 20K, 25K), the behavior and nature of the network are as shown when using the OpenFlow protocol and both mechanisms are used in the simulation program.

Figures 5 and 6 illustrate the Throughput (the ratio of the total amount of data reached by a receiver from a sender to the time it takes for the receiver to obtain the last packet is referred to as Throughput) [10]. Regarding network Throughput, in using a single or multi-controller mechanism, the behavior of the network in a multi-controller design using the OpenFlow controller protocol produces an increase
in Throughput. Figures 5 and 6 show that the use of a multi-controller is therefore more positive in this respect.

Figure 5. Throughput for network in Single Controller and Multi-Controller

Figure 6. Throughput for network in Single Controller and Multi-Controller

5. Conclusion
This paper has introduced network metrics using single or multi-controller technology, followed by a performance evaluation of the OpenFlow protocol. The results indicate that the performance of the OpenFlow protocol in the network is affected by both single or multi-controller technology. The objective of this study is to determine the impact of using a single or multi-controller on more than one module in a network. This provides information for the designers of networks as well as facilitating study of the behavior of an OpenFlow protocol in such scenarios. Future studies can be conducted using these mechanisms for many purposes depending on the actual and virtual need for reality, and the effects can be studied to avoid any costly mistakes when real networks are designed on the ground.
References

[1] Kang, M., Kang, E.Y., Hwang, D.Y., Kim, B.J., Nam, K.H., Shin, M.K. and Choi, J.Y., 2013, March. Formal modeling and verification of SDN-OpenFlow. In Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on (pp. 481-482). IEEE

[2] Wang, S.Y., Chou, C.L. and Yang, C.M., 2013. EstiNet openflow network simulator and emulator. IEEE Communications Magazine, 51(9), pp.110-117.

[3] OpenFlow Switch Consortium, OpenFlow Switch Specification Version 1.0. 0.(2009). Internet: http://www.archive.OpenFlow.org/documents/OpenFlow-spec-v1.0.0.pdf.

[4] Klein, D. and Jarschel, M., 2013, March. An OpenFlow extension for the OMNeT++ INET framework. In Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques (pp. 322-329).

[5] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N. and Shenker, S., 2008. NOX: towards an operating system for networks. ACM SIGCOMM Computer Communication Review, 38(3), pp.105-110.

[6] Fall, K. and Floyd, S., 1996. Simulation-based comparisons of Tahoe, Reno and SACK TCP. ACM SIGCOMM Computer Communication Review, 26(3), pp.5-21.

[7] Lantz, B., Heller, B. and McKeown, N., 2010, October. A network in a laptop: rapid prototyping for software-defined networks. In Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (p. 19).

[8] Wang, S.Y., Chou, C.L. and Yang, C.M., 2013. EstiNet openflow network simulator and emulator. IEEE Communications Magazine, 51(9), pp.110-117.

[9] Chamberlain, T., 2013. Learning OMNeT++. Packt Publishing Ltd.

[10] A. Abdou, A. Matrawy and P. C. van Oorschot, May 2015"Accurate One-Way Delay Estimation With Reduced Client Trustworthiness," in IEEE Communications Letters, vol. 19, no. 5, pp. 735-738,