ANALYSIS OF THE EFFICIENCY OF THE OPENFLOW CONTROLLER IN A NETWORK WITH DIFFERENT LOADS

DOI: 10.36724/2072-8735-2023-17-1-49-54

Denis O. Yakupov,
Povolzhskiy State University of Telecommunications and Informatics, Samara, Russia, **d.yakupov@psuti.ru**

Sergey V. Malakhov,
Povolzhskiy State University of Telecommunications and Informatics, Samara, Russia, **malakhov-sv@psuti.ru**

Manuscript received 11 December 2022;
Accepted 17 January 2023

Keywords: *Software-Defined Network, OpenFlow protocol, OpenDaylight, Cbench, controller productivity*

One of the main ideas of software-defined network is the creation of special software (OpenFlow controller), which allows you to separate the management of existing network equipment (routers and switches) without changing it. This software can run on a separate PC, which is managed by a network administrator. Therefore, this paper examines the performance of the OpenFlow controller. This article describes the OpenFlow OpenDaylight controller, and also defines its place in the network architecture of a software-defined network. The methodology of the conducted research of the experiment includes a description of the special Cbench software. In operation, Cbench was launched in two modes: the delay mode for sending subsequent packets and the maximum data transfer mode for measuring bandwidth. In delay mode, each switch sends one new packet to the emulated stream and waits for a response. After that, it sends the next packet and so on. The delay shows the time it takes for the controller to process an OpenFlow request under low load. In bandwidth mode, each switch sends requests until the buffer is full. Thus, this mode allows you to measure the maximum performance that the controller can handle. According to the presented methodology, experimental samples of the network were collected. The experiments are as follows: 1) when the switch distributes the load to the controllers and, 2) Cbench generates streams for each of the controllers separately. The mode of operation of the Cbench program is also changing. According to the results of the experiments, data is obtained on the number of packets processed by each controller under different conditions.

Information about authors:
Denis O. Yakupov, *Povolzhskiy State University of Telecommunications and Informatics, Assistant of the Department of Software Engineering, Samara, Russia*
Sergey V. Malakhov, *Povolzhskiy State University of Telecommunications and Informatics, Associate Professor of the Department of Software Engineering, Samara, Russia*

Для цитирования:
Якупов Д.О., Малахов С.В. Анализ эффективности контроллера OpenFlow в сети с разной нагрузкой // T-Comm: Телекоммуникации и транспорт. 2023. Том 17. №1. С. 49-54.

For citation:
Yakupov D.O., Malakhov S.V. (2023). Analysis of the efficiency of the OpenFlow controller in a network with different loads. *T-Comm*, vol. 17, no.1, pp. 49-54. (*in Russian*)

Introduction

Currently, network architectures are becoming increasingly popular, which have separate levels of data representation (a comparison with the route table and a transaction of network packets from one port to another is performed) and for management (establishing the optimal route and forwarding data along selected routes). This architecture has many advantages due to an ordered algorithm that makes the data transfer process much easier. Data transmission is controlled by special software, which is a network software controller. The software introduces its own delays for data processing, being on the side of a separate server. Due to this, the task of studying the performance of the controller in software-defined network becomes the most relevant [1].

Trends in the development of modern infocommunication networks, such as the intensive growth of new network applications, the growth of the number and types of new network devices, lead to a significant increase in the volume of transmitted traffic. In this case, it becomes necessary to manage heterogeneous flows and provide - the required level of QoS (Quality of Service) quality of service when ensuring the security of processing the transmission of stream data. This leads to the fact that providers of large networks need to look for new network control mechanisms, with the ability to quickly configure networks.

Software-Defined Networking (SDN) and Network Function Virtualization (NFV) technologies with customizable software provide control over network management, and control is more intelligent and centralized. This makes the network more flexible, programmable, and innovative [2, 3].

The implementation of SDN entails the need to develop adequate models that allow you to quickly obtain accurate estimates of QoS parameters necessary at the stage of network design and further during operation in order to be able to quickly respond to changes in network requirements and network topology modification [4]. It should be noted that the issues of performance and scalability of SDN are still little investigated in terms of building analytical models. Simulation simulations and experiments on real networks have their advantages

Most of the work devoted to the study of the effectiveness of the SDN is aimed at developing simulation models and experimenting on real equipment. [5-7] presents mathematical models for evaluating the performance of controllers and switches under overload conditions. The OpenFlow-based SDN analytical model in [8,9] approximates the data plane as an open Jackson network with a controller modeled as a M/M/1 queue. [10] presents the SDN performance model based on the OpenFlow protocol with several OpenFlow switches, but this model is obtained with the assumption that the processed flows are the simplest, while the SDN controller incoming message processing performance is calculated based on the M/G/1 model. However, it is known that the flows generated by modern applications in the network are not Poisson, which requires considering the real properties (parameters) of traffic in the models used [11, 12].

The works [13, 14] present mathematical models for systems that process non-Poisson flows, but only the parameters of the functioning of individual sections of the SDN, and not the network, are considered.

The development of an analytical model of SDN as a G/G/1 system remains very relevant. The methodology presented in [10] is convenient to use to expand the application processing model in the SDN when servicing non-Poisson flows based on the model of an arbitrary G/G/1 queue.

The following is an analytical model of the OpenFlow-based software-configurable network, since it is the most common, and such a model can be used as a basic model for other communication protocols in the SDN. As a request, a separate package, or a bundle of packages, or a stream is considered.

Packet traffic formed in the form of packet bursts is considered as processed flows, which most closely corresponds to the nature of formation of modern flows [15].

Like the approach, [10] analyzes separately the request receipt process and the procedure for forwarding requests through the switch and the OpenFlow controller, and then the system for forwarding requests to the SDN.

Theoretical information

The controller is a special server running on a special network environment, together with network software. The network environment simplifies the process of developing components, as well as combining them within a single project. It interacts with the equipment and makes it possible to monitor and manage the software-defined network in full. The network environment itself does not control the network, but only allows you to use an interface for special software that already could manage all the network functionality.

Currently, there is a variety of controllers used in software-defined networks. To conduct the experiment, the OpenDaylight controller was chosen.

Previously, network equipment was not controlled centrally – all settings were written individually for each gateway/matchmaker. And it suited almost everyone. However, with the advent of new cloud technologies, needs have changed and new ideas for organizing network management have been needed. One of these ideas was SDN (Software Defined Networking). The idea is that in a network based on it, network management is separate from data devices, which adds another layer of abstraction.

Based on this idea, another one was developed - NFV (Network Functions Virtualization), which closely intersects with the idea of SDN. NFV allows you to virtualize individual network functions that previously could only be implemented physically, for example, a firewall or IDS. That is, physically the user can be in Samara, and the firewall is overseas [16].

And then there is a transition to OpenDaylight. It is a platform for organizing the work of these technologies. Strictly speaking, this is not a ready-made solution, but a framework based on which interested companies and developers can build their products for SDN networks. Before describing OpenDaylight further, you need to get some idea of the structure of such networks. You can conditionally divide an SDN network into three layers:

- at the very top level are applications that are responsible for network and business logic and monitor/monitor network behavior. More complex solutions also combine cloud and NFV applications, and design network traffic according to the requirements of these applications;
- the second layer, in fact, is an abstraction layer - on it there is a separation of network management and data transmission;
- and finally, the last layer is the layer of data transmission devices. It is worth noting that these devices can be both physical and virtual.

OpenDaylight concentrates on the second layer of the SDN network and provides a set of RESTful APIs and an OSGi framework for upper-layer applications ("northern interface"), while implementing C&C protocols for data devices such as OpenFlow, BGP and SNMP ("southern interface"). In addition, it implements various managers - from topology manager to traffic forwarding manager.

The OpenDaylight kernel is written in Java, therefore, it can work on any system where JVM is available. In addition, this framework is extremely flexible and modular, which allows you to use only the capabilities that are necessary for a specific network.

In general, this is a rather promising direction, which, however, makes sense only for a large Enterprise sector, for example, for backbone providers.

Having studied the capabilities of the controller, we can say that OpenDaylight can process more than 500 thousand threads in one second, while saving time for processing one packet about 5 ms [17].

If you do not optimize the generated flows of the system in question, then the use of this architecture will be ineffective. Here, in addition to the OpenFlow controller, there are also third-party components that play an important role in the formation of delays. But, if you perform optimization at the very beginning, then the efficiency of using the OpenFlow controller can be increased significantly.

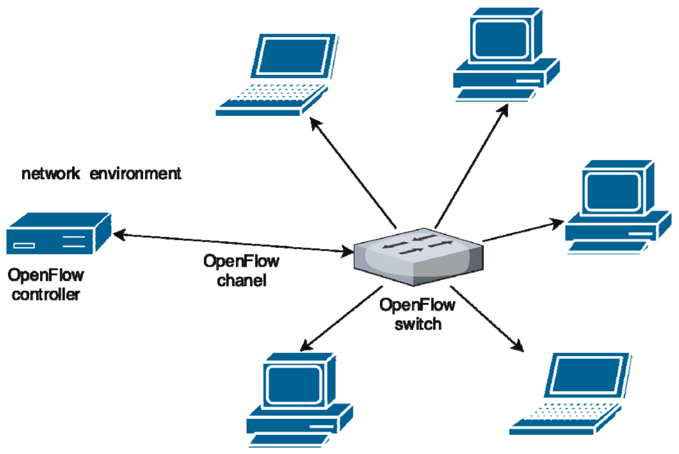

Fig. 1. SDN network architecture

Figure 1 shows a software-defined Open Flow network consisting of an OpenFlow controller, an OpenFlow switch, communicating over a communication channel.

Performance Study Methodology

With the help of special open source Cbench Security Detection software, it is possible to simulate several OpenFlow indicators to measure controller performance and any other parameters, including monitoring tools and maximum performance, detection, and control of incoming traffic, as well as outgoing traffic volumes [18], with a high level of packet verification.

Cbench will be used to measure controller performance. In this case, the number of threads per second that the controller can process will be indicated.

In software-defined network OpenFlow, the controller establishes and distributes traffic through OpenFlow switches. The distribution process may occur statically before the arrival of the packet (active distribution) or dynamically (reactive distribution) as part of the next process. The latter flow parameter is particularly sensitive as the flow slows down the first packet slightly. Once configured, the thread addressing state is cached on the OpenFlow switch so that the process is not repeated for subsequent packets in the same thread. OpenFlow switches also report cache and backup status indefinitely, after a specified timeout, or after a period of inactivity. The installation process, in which both streams are an integral part of the SDN, has been identified as the most likely source of bottlenecks.

The Cbench software measures the various performance parameters associated with the flow setup time. Cbench simulates a configurable number of OpenFlow switches so that they all connect to the same OpenFlow controller. Each simulated switch sends several new message streams to the OpenFlow controller, waits for the corresponding configuration response stream, and records the time difference between request and response.

Cbench supports two operating modes: delay mode and data transfer mode for bandwidth measurement. In delay mode, each switch sends one new packet to the emulated stream and waits for a response. After that, it sends the next packet and so on. The delay shows the time it takes for the controller to process an OpenFlow request under low load.

In bandwidth mode [19], each switch sends requests until the buffer is full. Thus, this mode allows you to measure the maximum performance that the controller can cope with.

Experimental studies

Below are experimental network diagrams, when Huawei S5720 distributes the load to controllers (Fig. 2) and is generated from the Cbench server for each controller OD-1, OD-2, OD-3 and OD-4 its flow (Fig. 3). Description of the main equipment:

1) four servers to install an OpenDaylight (OD) controller on each server;

2) a dedicated ser r for installing Cbench software;

3) Huawei S5720 switch for load sharing;

4) GigabitEthernet communication channels.

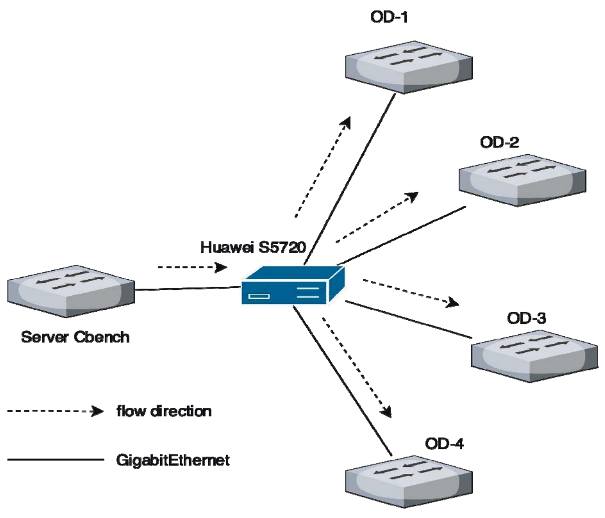

Fig. 2. Experimental network diagram, Cbench server generates one stream

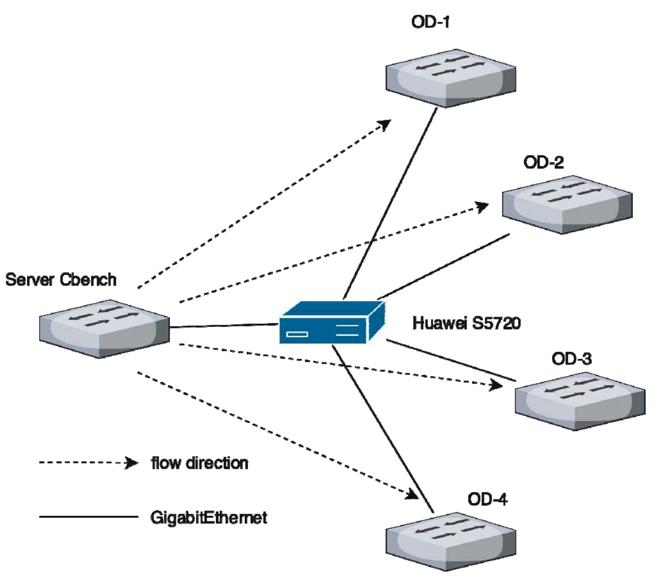

Fig. 3. Experimental network diagram, Cbench server generates four streams for each controller

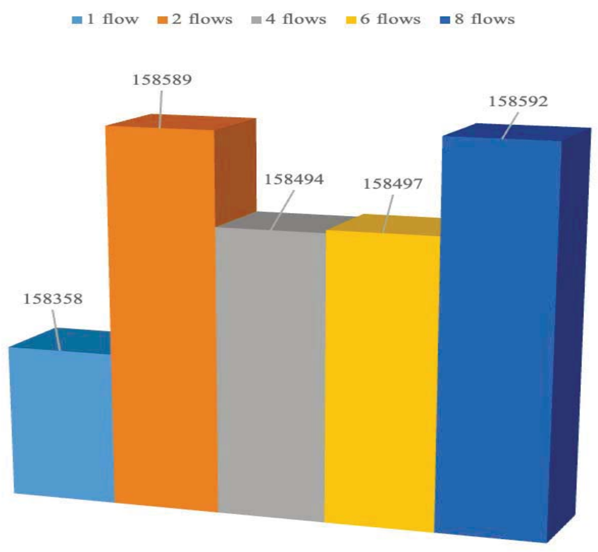

Fig. 4. Network Bandwidth Result for the First Type of Testing

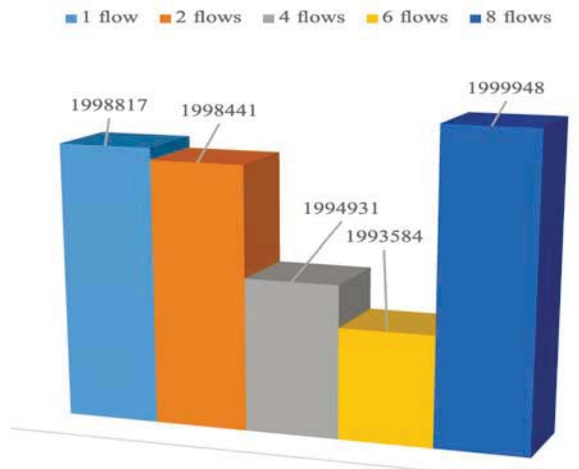

Fig. 5. Network Bandwidth Result for the Second Type of Testing

Description of the experiment:

1. Maximum Performance Experiment. The number of threads is 1, 2, 4, 6 and 8. The stream is generated by Cbench on a dedicated server. The flow passes through the Huawei S5720 switch. On the switch, the load is distributed to 4 OpenDaylight controllers. Cbench runs with the «-t» option with maximum performance.

2. Experiment in delay mode. Cbench starts with the «-i» parameter with a delay between the switches being connected. The remaining parameters, as in the 1st experience.

3. Maximum Performance Experiment. The number of threads is 1, 2, 4, 6 and 8. From the dedicated server, threads are sent to each controller separately. At the same time, Huawei does not perform load balancing. Cbench runs with the «-t» option with maximum performance.

4. Experiment in delay mode. Cbench starts with the «-i» parameter with a delay between the switches being connected. The remaining parameters, as in the 3rd experience.

Results of the experiment

In order to correctly calculate the number of controllers at a certain load on the experimental network [20], you need to know the bandwidth for each controller.

A test result was obtained, which consists of 5 command runs with a certain number of studies. Each launch was carried out on 1, 2, 4, 6, 8 streams. The result shows testing to achieve maximum throughput performance (Figure 4).

The results clearly show that maximum performance is observed where the OpenDaylight controller processes 2 and 8 threads at once.

Test result for the delay time study (Figure 5).

The results clearly show that when sent from the OpenFlow switch to the controllers, the minimum delay is where the OpenDaylight controller processes 4 and 6 threads at once.

Test result when a Cbench object is started on each OpenDaylight controller and the switch does not perform load sharing (Figure 6).

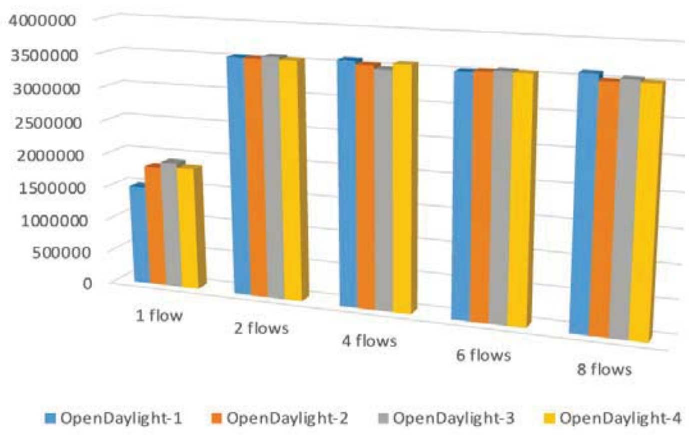

Fig. 6. Network Bandwidth Result for Third Test

As can be seen from the results, the lowest performance is observed when processing a single thread by a group of controllers.

Test result to examine the delay time when the OpenFlow controller does not perform load sharing (Figure 7).

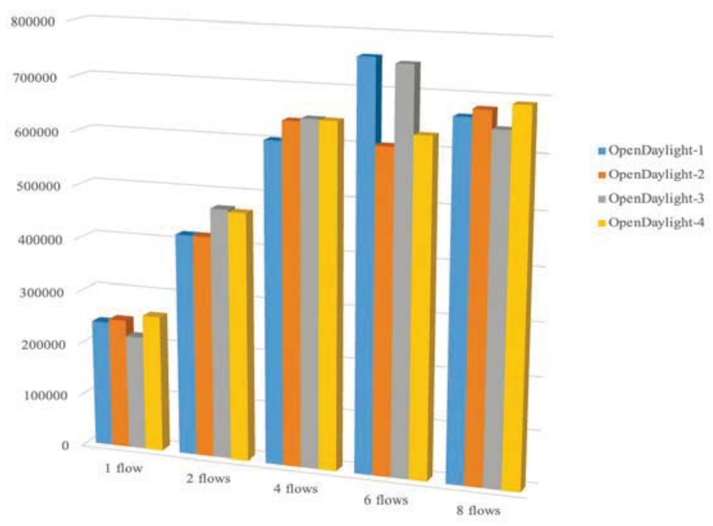

Fig. 7. Network Bandwidth Result for the Fourth Type of Testing

Total, based on the total results for all four types of tests (Figure 8).

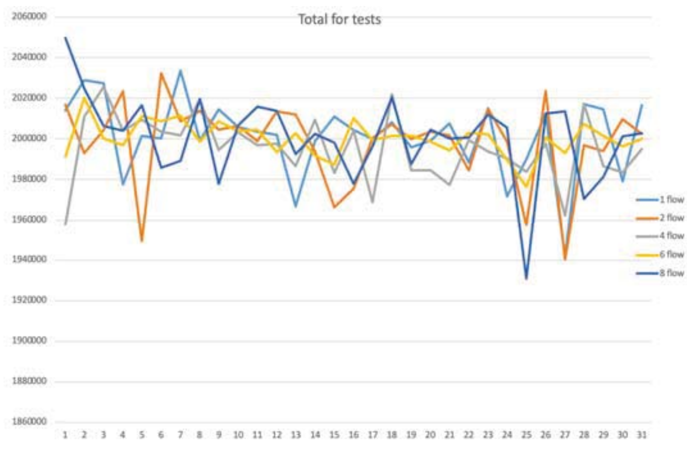

Fig. 8. Total sum of network performance for 31 experiments performed

As you can see from the results of the total, during the first two tests, the greatest performance increase is observed at 8 threads.

Conclusion

After network testing is complete, we can conclude that delays are the most significant aspect to reduce performance. In other words, the relationship between the OpenFlow switch and the OpenDaylight controller should have as little active network equipment as possible and give the highest available speed. This study shows that when the load is distributed to the controllers, there is no significant increase in performance.

When the switch is involved in load balancing: maximum throughput performance is observed where the OpenDaylight controller processes 2 and 8 threads at once, and minimum latency where the OpenDaylight controller processes 4 and 6 threads at once. When the switch does not perform load balancing, the lowest throughput performance is observed when processing one thread by a group of controllers, and the minimum latency of one thread by a group of controllers.

References

1. Habrahabr. Access mode: http://habrahabr.ru/post/169403 (02.11.2022).

2. Feamster N., Rexford J., Zeguraz E. (2013) The road to SDN: An intellectual history of programmable networks. *Networks*. Vol. 11. No. 12. P. 20-40.

3. Astutoz B.N., Mendonca M., Nguyen X.N., Obraczkaz K., Turletti T. (2014) A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys & Tutorials*. Vol. 16. No. 3, pp. 1617-1634.

4. McKeown N., Anderson T., Balakrishnan H., Parulkar G., Peterson L., Rexford J., Shenker S., Turner J. (2008) Openflow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*. Vol. 38. No. 2, pp. 69-74.

5. Bozakov Z., Rizk A. (2013) Taming SDN controllers in heterogeneous hardware environments. *Proceedings of the Second European Workshop on Software Defined Networks (EWSDN)*, pp. 50-55.

6. Azodolmolky S., Wieder P., Yahyapou R. (2013) Performance evaluation of a scalable software-defined networking deployment. *Proceedings of the Second European Workshop on Software Defined Networks (EWSDN),* pp. 68-74.

7. Azodolmolky S., Nejabati R., Pazouki M., Wieder P. (2013) An analytical model for software defined networking: A network calculus-based approach. *Proceedings of the 2013 IEEE Global Communications Conference (GLOBECOM)*, pp. 1397-14021.

8 Jarschel M., Oechsner S., Schlosser D., Pries R., Goll S., Phoe T.G. (2011) Modeling and performance evaluation of an openflow architecture I. *Proceedings of the Twenty-third International Teletraffic Congress (ITC)*, pp. 1-7.

9. Mahmood K., Chilwan A., Osterbo O., Jarschet M. (2015) Modelling of open-flow-based software-defined networks: the multiple node case. *JET Networks*. Vol. 4. No. 5, pp. 278-284.

10. Xiong B., Yang K., Zhao J., Li W., Li K. (2016) Performance evaluation of OpenFlow-based software-defined networks based on queueing model. *Computer Networks*. No. 102, pp. 174-183.

11. O.I. Shelukhin, A.V. Osin, S.M. Smolsky (2008) Self-similarity and fractals // Telecommunications applications. Moscow: Fizmatlit. 368 p.

12. Taggu M.S. (1988) Self-similar processes. In S. Kotz and N. Johnson, editors, Encyclopedia of Statistical Sciences. New York: Wiley. Vol. 8, pp. 352-357.

13. Samuilov K.E., Shalimov I.A., Buzhin I.G., Mironov Y.B. (2018) Model of functioning of telecommunication equipment of software-configurable networks. *Modern information technologies and IT education*. Vol. 14. No. 1, pp. 13-26.

14. Muhizi S., Shamshin G., Muthanna A., Kirichek R., Vladyko A., Koucheryavy A. (2017) Analysis and Performance Evaluation of SD Queue Model. *Lecture Notes in Computer Science*. Vol. 10372, pp 26-37.

15. Okamura H., Dohi T., Trivedi K.S. (2009) Markovian arrival process parameter estimation with group data. *IEEE/ACM Transactions on Networking*. Vol. 17. Iss. 4, pp. 1326-1339.

16. Overview of Promising FOSS Network Projects. Access mode: https://xakep.ru/2014/07/17/promising-foss-projects/ (02.11.2022).

17. Tavakoli A., Casado M., Koponen T. (2009) Applying nox to the datacenter. *Proceedings of HotNets*, issue 8, pp. 42-48.

18. Curtis A., Mogul J., Tourrilhes J., Yalagan-Dula P., Sharma P. (2011) Devoflow: scaling flow management for high-performance networks. *Proceedings of the ACM SIGCOMM*, vol. 1, pp. 127-133.

19. Yu M., Rexford J., Freedman J., Wang J. (2010) Scalable flow-based networking with difane. *Proceedings of the ACM SIGCOMM*, vol. 41, issue 4, pp. 351-362.

20. Rob Sherwood, Kok-Kiong Yap. Cbench: an OpenFlow Controller Benchmarker. Access mode: http://www.openflow.org/wk/ index.php/Oflops. (16.11.2022)

АНАЛИЗ ЭФФЕКТИВНОСТИ КОНТРОЛЛЕРА OPENFLOW В СЕТИ С РАЗНОЙ НАГРУЗКОЙ

Якупов Денис Олегович, *Поволжский государственный университет телекоммуникаций и информатики, г. Самара, Россия,* ***d.yakupov@psuti.ru***

Малахов Сергей Валерьевич, *Поволжский государственный университет телекоммуникаций и информатики, г. Самара, Россия,* ***malakhov-sv@psuti.ru***

Аннотация
Одной из основных идей программно-определяемой сети является создание специального программного обеспечения (контроллера OpenFlow), позволяющего разделить управление существующим сетевым оборудованием (маршрутизаторами и коммутаторами) без его изменения. Это программное обеспечение может работать на отдельном ПК, которым управляет сетевой администратор. Поэтому в этой статье исследуется производительность контроллера OpenFlow. В этой статье описывается контроллер OpenFlow OpenDaylight, а также определяется его место в сетевой архитектуре программно-определяемой сети. Методология проведенного исследования эксперимента включает описание специального программного обеспечения Cbench. В работе Cbench запускался в двух режимах: режим задержки для отправки последующих пакетов и режим максимальной передачи данных для измерения пропускной способности. В режиме задержки каждый коммутатор отправляет один новый пакет в эмулируемый поток и ожидает ответа. После этого он отправляет следующий пакет и так далее. Задержка показывает время, которое требуется контроллеру для обработки запроса OpenFlow при низкой нагрузке. В режиме полосы пропускания каждый коммутатор отправляет запросы до тех пор, пока не заполнится буфер. Таким образом, этот режим позволяет измерить максимальную производительность, с которой может справиться контроллер. По представленной методике были собраны экспериментальные образцы сети. Эксперименты, следующие: 1) когда коммутатор распределяет нагрузку на контроллеры и, 2) Cbench генерирует потоки для каждого из контроллеров отдельно. Меняется и режим работы программы Cbench. По результатам экспериментов получают данные о количестве пакетов, обработанных каждым контроллером в разных условиях.

Ключевые слова: *программно-определяемая сеть, протокол OpenFlow, OpenDaylight, Cbench, производительность контроллера.*

Литература

1. Habrahabr [Electronic resource]. Access mode: http://habrahabr.ru/post/169403 (02.11.2022).
2. *Feamster N., Rexford J., Zeguraz E.* The road to SDN: An intellectual history of programmable networks // Networks. 2013. Vol. 11. No. 12. P. 20-40.
3. *Astutoz B.N., Mendonca M., Nguyen X.N., Obraczkaz K., Turletti T.* A survey of software-defined networking: Past, present, and future of programmable networks // IEEE Communications Surveys & Tutorials. 2014. Vol. 16. No. 3, pp. 1617-1634.
4. *McKeown N., Anderson T., Balakrishnan H., Parulkar G., Peterson L., Rexford J., Shenker S., Turner J.* Openflow: Enabling innovation in campus networks // ACM SIGCOMM Computer Communication Review. 2008. Vol. 38. No. 2, pp. 69-74.
5. *Bozakov Z., Rizk A.* Taming SDN controllers in heterogeneous hardware environments // Proceedings of the Second European Workshop on Software Defined Networks (EWSDN), 2013, pp. 50-55.
6. Azodolmolky S., Wieder P., Yahyapou R. Performance evaluation of a scalable software-defined networking deployment // Proceedings of the Second European Workshop on Software Defined Networks (EWSDN), 2013, pp. 68-74.
7. *Azodolmolky S., Nejabati R., Pazouki M., Wieder P.* An analytical model for software defined networking: A network calculus-based approach // Proceedings of the 2013 IEEE Global Communications Conference (GLOBECOM), 2013, pp. 1397-14021.
8. *Jarschel M., Oechsner S., Schlosser D., Pries R., Goll S., Phoe T.G.* Modeling and performance evaluation of an openflow architecture I Proceedings of the Twenty-third International Teletraffic Congress (ITC), 2011, pp. 1-7.
9. *Mahmood K., Chilwan A., Osterbo O., Jarschet M.* Modelling of open-flow-based software-defined networks: the multiple node case // JET Networks. 2015. Vol. 4. No. 5. pp. 278-284.
10. *Xiong B., Yang K., Zhao J., Li W., Li K.* Performance evaluation of OpenFlow-based software-defined networks based on queueing model // Computer Networks. 2016. No. 102, pp. 174-183.
11. *O.I. Shelukhin, A.V. Osin, S.M. Smolsky*, Self-similarity and fractals // Telecommunications applications. M.: Fizmatlit, 2008. 368 p.
12. *Taggu M.S.* Self-similar processes // In S. Kotz and N. Johnson, editors, Encyclopedia of Statistical Sciences. New York: Wiley. 1988. Vol. 8, pp. 352-357.
13. *Samuilov K.E., Shalimov I.A., Buzhin I.G., Mironov Y.B.* Model of functioning of telecommunication equipment of software-configurable networks//Modern information technologies and IT education. 2018. Vol. 14. № 1, pp. 13-26.
14. *Muhizi S., Shamshin G., Muthanna A., Kirichek R., Vladyko A., Koucheryavy A.* Analysis and Performance Evaluation of SD Queue Model // Lecture Notes in Computer Science. 2017. Vol. 10372, pp 26-37.
15. *Okamura H., Dohi T., Trivedi K.S.* Markovian arrival process parameter estimation with group data // IEEE/ACM Transactions on Networking. 2009. Vol. 17. Iss. 4, pp. 1326-1339.
16. Overview of Promising FOSS Network Projects [Electronic resource]. https://xakep.ru/2014/07/17/promising-foss-projects/ (02.11.2022).
17. *Tavakoli A., Casado M., Koponen T.* Applying nox to the datacenter // Proceedings of HotNets, 2009, issue 8, pp. 42-48.
18. *Curtis A., Mogul J., Tourrilhes J., Yalagan-Dula P., Sharma P.* Devoflow: scaling ?ow management for high-performance networks // Proceedings of the ACM SIGCOMM, 2011, vol. 1, pp. 127-133.
19. *Yu M., Rexford J., Freedman J., Wang J.* Scalable flow-based networking with difane // Proceedings of the ACM SIGCOMM, 2010, vol. 41, issue 4, pp. 351-362.
20. *Rob Sherwood, Kok-Kiong Yap.* Cbench: an OpenFlow Controller Benchmarker [Electronic resource]. Access mode: http://www.openflow.org/wk/index.php/Oflops. (16.11.2022).

Информация об авторах:
Якупов Денис Олегович, *Поволжский государственный университет телекоммуникаций и информатики, ассистент кафедры ПрИ, г. Самара, Россия*
Малахов Сергей Валерьевич, *Поволжский государственный университет телекоммуникаций и информатики, доцент кафедры ПрИ, к.т.н., г. Самара, Россия*