A Context-Aware Traffic Engineering Model for Software-Defined Networks

Phuong T. Nguyen[1(✉)], Hong Anh Le[2], and Thomas Zinner[3]

[1] Research and Development Center, Duy Tan University, Da Nang, Vietnam
phuong.nguyen@duytan.edu.vn
[2] Hanoi University of Mining and Geology, Hanoi, Vietnam
lehonganh@humg.edu.vn
[3] Lehrstuhl für Informatik III, Universität Würzburg, Würzburg, Germany
zinner@informatik.uni-wuerzburg.de

Abstract. Software-Defined Networking is a novel paradigm, based on the separation of the data plane from the control plane. It facilitates direct access to the forwarding plane of a network switch or router over the network. Though it has a lot advantages, the SDN technology leaves considerable room for improvement. Research problems like efficient techniques for customization and optimization for SDN networks are under investigation. This paper aims at proposing a model for traffic engineering in SDN-based networks.

Keywords: Software-Defined Networking · Traffic Engineering · Context-Aware Systems

1 Introduction

Performing experiments in production networks with legacy switches had been a costly and arduous task for a long time, until the Software-Defined Networking approach (SDN) appeared. SDN is a novel paradigm, based on the separation of the data plane from the control plane. While the former remains in switches, the latter is ported to a programmable controller which can either be a physical computer or a virtual machine. By this way, SDN gives researchers the flexibility in working with networks, it allows to perform their own experiments on network devices. The emergence of SDN provides users with a convenient way to customize network applications, without intervening the internal design of commodity switches [5],[6]. Though its technical aspects are still under development, SDN has found its way going into other formulation, SDN is widely adapted by network, content, and datacenter providers.

OpenFlow is a protocol for the communication between controllers and OpenFlow switches. It provides direct access to the forwarding plane of a switch or router. OpenFlow is widely accepted and considered to be the most notable deployment of SDN [5]. The protocol has recently received a growing attention both from academe and industry. As of June 2014, there are more than 150

P.C. Vinh et al. (Eds.): ICTCC 2014, LNICST 144, pp. 73–82, 2015.
DOI: 10.1007/978-3-319-15392-6_8

members registered to the Open Networking Foundation - the consortium for the development and standardization of SDN.

Despite a myriad of advantages, the SDN technology leaves considerable room for improvement. Research problems like efficient techniques for customization and optimization for SDN networks are under investigation. Our paper aims at introducing state of the art of SDN. Afterward, it is going to present a proposal for a traffic engineering model in SDN-based networks. The paper is organized as follows. In Section 2 we present the reader an overview of SDN functionalities. Recent developments on SDN performance are reviewed in Section 3. A running example is introduced in Section 4. Section 5 highlights the motivations, research objectives as well as our proposed solution. Finally, Section 6 concludes the paper.

2 SDN Functionalities

An SDN switch may hold a number of flow tables, each of them stores forwarding rules. A flow entry consists of three components: headers, actions, and statistics. The flow tables of a switch are used as the base for manipulating packets. Figure 1 illustrates how the first packet of a new flow is processed at an SDN switch.

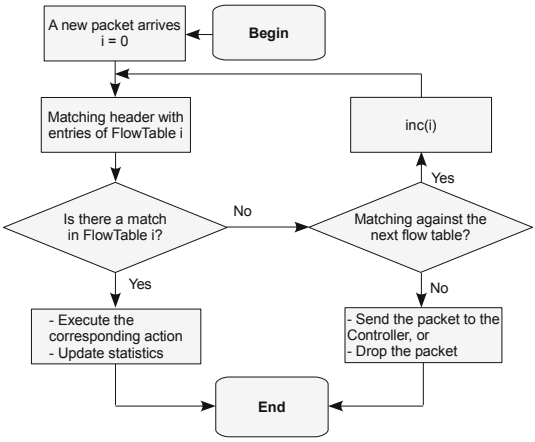

Fig. 1. Processing for the first packet of a flow

Upon the arrival of the packet, the packet's header is compared with the rules in the first flow table. If there is a match, the corresponding actions are executed and statistics are updated. The packet can also be discarded depending on the rules defined at the switch. In contrast, if there is no match, the header is compared against rules in the next tables or the packet is encapsulated and forwarded towards the controller. This is left at programmer's discretion. When

receiving the packet, the controller may create a new rule and sends back to the switch which in turn updates the new rule to the flow table.

Figure 2 shows the fields for matching defined by the OpenFlow standard.

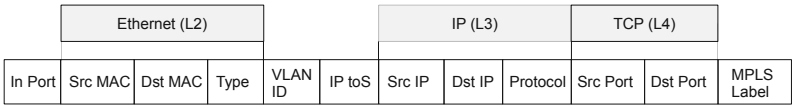

Fig. 2. Fields for matching [17],[18]

The following components are essential for an SDN implementation:

- The data plane is a set of flow tables and the actions corresponding to the table entries.
- The control plane is the controller that manages the flow tables through a pre-defined protocol.
- A flow is normally a group of consecutive packets sharing same features.

Using SDN's centralized intelligence, an administrator can exploit some advantages. SDN is used to route flow at Layer 2, Layer 3, and Layer 4 separately or concurrently according to packet's header information. In conventional networks, to configure network flows, an administrator needs to manipulate network devices separately using CLI (Command Line Interface). This poses a great difficulty, especially for networks with heterogeneous hardware units. In addition, it is almost impossible to program the network so that it can self configure during operation; any re-configuration needs to be manually done by the administrator. SDN helps the administrator do his work in a smoother way. It is not necessary for the administrator to know the specification of each hardware. He can configure a wide range of hardwares from different vendors using a single language.

3 Technical Issues

One might argue that SDN is no more useful if the time for processing incoming packets is longer that that of a legacy switch. It is, therefore, necessary to investigate the performance of SDN-based switches. The performance of data plane and control plane has received much interest from the research community. This section gives an overview of some notable studies on the problem.

It is worth noting that controller and switch - representing the control plane and the data plane - are the integral parts of the SDN approach. Correspondingly, performance evaluations have been done pertaining to these aspects. Parameters regarding response time, throughput of controller implementation as well as OpenFlow switches have been thoughtfully investigated. In [6] a performance evaluation for the data plane is presented. In this paper, the performance the of OpenFlow data plane is compared to that of IP routing and switching. It has

been shown that, the performance of the data plane is comparable with that of the two technologies. Similarly, in [8] the authors introduce a performance comparison for an OpenFlow switch implementation and a native router. The two studies, [6] and [8], show that the efficiency of the forwarding plane of OpenFlow switches is similar to that of commodity switches.

Investigating performance of control plane helps promote an understanding of the feasibility, scalability, and robustness of the SDN concept. This helps network administrators know the number of controllers launched to handle the network. In addition, this helps further provision suitable resources. There are lots of studies conducted to investigate performance of control plane for SDN implementations. A performance comparison for different OpenFlow controllers is presented in [7]. Alongside some existing controllers, the authors propose NOX-MT, an enhancement of NOX with multi-threads. The performance tests are then conducted on the four OpenFlow implementations: NOX, NOX-MT, Maestro, and Beacon. The measurement metrics are controller's throughput and response time. The experimental results demonstrate that NOX-MT outperforms the other implementations.

In [5] the authors conduct performance tests to evaluate the performance of the control plane and the data plane of three OpenFlow switches: Open vSwitch, Pronto 3290 and NetFPGA OpenFlow switch. The experiments aim to measure the delay time at each OpenFlow switch for different packet sizes. The experimental results show that the NetFPGA OpenFlow switch has the lowest delay time compared to those of Open vSwitch and Pronto 3290. The Open vSwitch needs much more time to process a packet since it frequently accesses memory. In addition to the performance measurement, the authors also propose a simple model of an OpenFlow architecture based on queueing theory. In their approach, the performance of a controller is measured with the following parameters: the delay time at controllers and switches, the probability a packet is dropped given that the controller is out of service. The model has the advantage of swiftly delivering results but it has also some limitations. Despite the fact that an OpenFlow controller can host a number of switches, the model allows only one switch for a controller. In addition, only TCP traffic is considered and UDP traffic is missing.

A model for flexible benchmarking OpenFlow controllers has been proposed in [2]. Along with the model a software tool was implemented. The benchmark allows the emulation of scenarios and topologies, it helps build performance evaluation scenarios using several virtual switches. The performance of the benchmark is compared with that of Cbench. The experimental results show that the tool produces comparable results to those of Cbench. In addition, the benchmark is able to provide more performance statistics, i.e. round trip time, number of sent/received packets per second, and the number of outstanding packets per switch. It is also possible to examine whether the controller treats switches equally or not. This helps study further the performance characteristics of different OpenFlow controller implementations.

A model for the optimization of flow scheduling for datacenter networks is presented in [9]. The system, named Hedera can schedule in an adaptive fashion, in order to utilize network resources more efficiently. From a global view, the system can monitor flow and based on the information collected calculates a more efficient path to redirect flows. Similarly, with the model proposed in [11], [12], an OpenFlow system can detect traffic congestion of a virtual link and migrate flows away from the congested links.

In summary, different technical issues of OpenFlow have been thoroughly examined in many studies. Important parameters relating to the performance of the OpenFlow concept have been identified. A number of different performance benchmarks are already implemented to measure performance of OpenFlow network
components. The performance of data plane of an OpenFlow switch is close to that of a legacy switch while the performance of control plane depends on controller implementation. Some studies have taken the first steps towards network traffic tailoring based on network's conditions. However, the issue of using up-to-date performance information to control network is yet at an early stage and remains an open research problem.

4 Running Example: Datacenter

Network technologies prior to SDN supported a certain level of virtualization. Nevertheless, customizing traffics to meet user's demands was not only a costly but also daunting task, since technologies did not offer a convenient way to do. SDN has been applied in datacenters and helps eliminate the limitations. The SDN architecture facilitates flexible control of the whole network. Through the separation of control plane and data plane, it is possible to route network traffic with regard to the content of flows.

SDN provides a better way to utilize resources while hiding the underlying physical networks. It allows for decoupling the logical layer and physical layer to create virtual networks working on a shared physical network infrastructure. Switches are responsible for basic data forwarding while the control functionalities are handled at a virtual machine/programmable server. As a result, administrators are able to add new resources without needing to re-configure the existing devices. In network virtualization, a slice is defined as a set of flows and formed based on pooling of different network resources. The SDN scheme helps manage network slices and automate network management. This facilitates the development of a multi-tenant network environment. Each slice is handled by a single, logical controller and runs through multiple switches. Slicing enables flexible connection of servers, storage equipments and switches. Users are able to independently customize their own load and policies [19].

In datacenters, a common task is to move a large amount of data from a location to another location. Using SDN, one can configure network based on packet header at execution time to efficiently exploit existing bandwidth. To see how SDN helps optimize network resources and bandwidth, we consider a datacenter as depicted in Figure 3 and Figure 4 where a caching scheme is applied.

In this scenario, file server FS is used to provide data for the whole network and CS works as a cache server. Big volume data is frequently transferred from FS to CS. Another data stream with lower bandwidth requirement flows continuously from $SW1 \rightarrow SW3$. Figure 3 shows how a datacenter in legacy network operates. All traffics including that from FS to CS are routed through the high bandwidth channel. The low bandwidth channel is inadvertently left free.

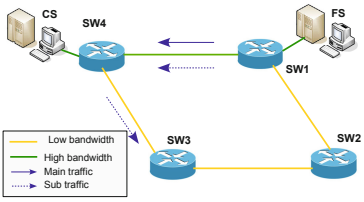

Fig. 3. Data transmission in legacy networks

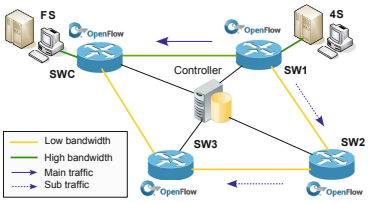

Fig. 4. Data transmission in SDN networks

The SDN paradigm provides a productive way to manage the transmission. In such a network the cache station CS is about to serve a number of nodes and therefore should be deemed of importance. Flows from the file server to the cache server are given a certain level of priority. In Figure 4 the same network topology is deployed with SDN switches superseding legacy switches. A controller is added to handle all the switches. Flows are distinguished by their features, e.g. the MAC addresses of CS and FS. Since the controller has a global view of the network and knows the network topology, it is able to customize flows to maximize network utilization. The controller assigns the highest bandwidth channel to flows from FS to CS, i.e. $SW1 \rightarrow SW4$, whilst moving out subordinate traffics to the low bandwidth channel, i.e. $SW1 \rightarrow SW2 \rightarrow SW3$. Once the data has been completely transmitted, the controller allocates the released traffic to other flows. By doing this, flows are manipulated to tailor network traffic. As a result, network bandwidth is more efficiently utilized.

Implementing the SDN concept in a datacenter brings benefits. However, from our perspective, there are still issues that need to be addressed. Considering the above mentioned scenario, some questions might arise. For example, what would happen if the channel from $SW1 \rightarrow SW2 \rightarrow SW3 \rightarrow SW4$ has been beefed up and got a higher bandwidth than that between $SW1 \rightarrow SW4$? The controller program is upgraded, does its processing ability affect the overall performance? Can the controller be aware of changes if an SDN switch with greater processing capability has been added? There is a need to adaptively react to changes happening in the surrounding environment. We consider these issues as our research problems.

5 A Proposed Model for the Adaptive Control of an SDN-Based Datacenter

5.1 Motivations

SDN ushers in a new era of the network technology. Nevertheless, the concept is still in its infancy. There is a potential of souping up network applications using SDN/OpenFlow. A main question is how to further exploit the centralized intelligence of SDN to increase network utilization. It is also necessary to further facilitate the cooperation between controllers and switches.

The survey in Section 3 implies that the performance of an SDN network is substantially dependent on the controller's performance and switch's status. An SDN/OpenFlow controller might be suitable for a specific type of applications than for an other. In a network where there is the presence of several SDN switches and corresponding controllers, a change in controller's implementation may produce adverse or beneficial effects on to the network. Applications themselves also have influence on the system, their behaviours can possibly place a burden on the performance over the course of time. **Given the circumstances, the regular monitoring of controllers, switches, flows, and application behaviours is intrinsic to a good performance of the whole system**.

So far, several researches have been conducted to study different aspects of the SDN approach. **However, comparatively little of them has addressed the issue of utilizing the performance metrics for controlling SDN networks**.

5.2 Goals

From our perspective, a control model for SDN networks which has the ability to deal with changes or perturbations occuring at execution time is meaningful. Based on investigations, our work aims at developing a management model for OpenFlow networks which performs operations according to the performance information of the underlying network. The proposed model is expected to facilitate traffic engineering techniques for OpenFlow networks. It will, therefore, pave the way for further developments in the SDN domain. The specific aims of our work are as follows.

Aim #1: To propose techniques for employing flow statistics, information about switches, application, and controllers to control OpenFlow networks.

- Hypothesis #1: Information about flows, switches and controllers as well as bandwidth demand from applications is beneficial to the autonomous reaction to changes or disruptions happening in the network.
- Expected Result #1: To develop a traffic engineering model that exploits information about the conditions of the surrounding environment as the input.

Aim #2: To validate the efficiency of the proposed model in an SDN-based system - a datacenter.

– Hypothesis #2: The prototype helps the datacenter efficiently utilize network resources and deal with environmental stimuli.
– Expected Result #2: To realize the model by building a software prototype based on the OpenFlow standard and to deploy test scenarios in a datacenter, either a real system or simulation.

Our work aims at proposing and evaluating a traffic engineering model for SDN-based datacenters. The traffic engineering model is realized using the OpenFlow standard. Figure 5 displays an abstract view of the prospective model. The succeeding sections will give a brief introduction to the model.

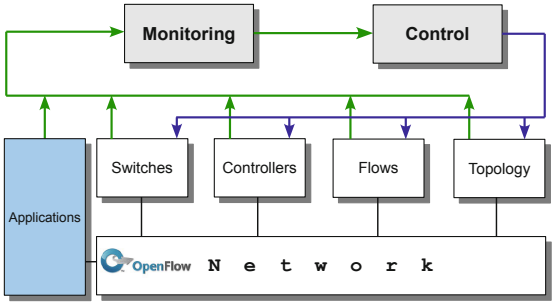

Fig. 5. An abstract view of the proposed model

5.3 Performance Monitoring

The proposed architecture consists of two main modules: Monitoring and Control as shown in Figure 5. The monitoring of controllers, switches, and flows contributes towards the optimization of the overall performance. In the first place, it is necessary to identify the factors that best represent the performance of an OpenFlow network. The existing studies on the performance of OpenFlow implementations provide a comprehensive analysis that can be used to derive a performance model suited to the requirements. The model will either exploit existing techniques or propose novel methods for the extraction of OpenFlow network's features.

The aim of performance monitoring is to provide the metrics reflecting network conditions. This module frequently communicates with switches, controllers, and flows to get up-to-date information about the network situation.

5.4 Topology Discovery

Information about network topology helps the control module calculate feasible paths for flows. The monitoring module needs to collect topology information to maintain a view of the network topology.

Each OpenFlow switch is programmed with some predefined configuration. The controller programs a rule in the new switch when it connects to the network.

The switch periodically sends data packets to all ports and waits for responses. This aims at testing the availability of neighbour switches as well as measuring sojourn time. The information is collected by the monitoring modul which then supplies the controller with status of links and transfer time. Once the controller collects enough information about network topology from all switches, it is able to construct network topology.

5.5 Network Control

Information collected by the monitoring module serves as the input for the control module (cf Figure 5). This includes information from flows, switches, controllers and applications. Switches report collision or disruption occurring in the constituent network segments. The statistics collected from the flow tables provide a view of the flows. Each application sends up-to-date information about bandwidth demand. The capacity of the existing communication channels will also be frequently reported from the switches.

The control module receives the information and conducts adequate countermeasures. It orders the controller to move or re-distribute flows to avoid collision and disruption. New routes are then calculated and programmed into related switches. The changes enable efficient bandwidth distribution and result in tailoring bandwidth demand and channel capacity. This contributes towards the efficient utilization of network resources and optimization of overall performance.

6 Conclusions and Future Work

In this paper we have introduced our proposal for a traffic engineering model for Software-Defined Networks. To turn the proposals into realization, we are working towards a software prototype for the management of OpenFlow networks based on an existing OpenFlow controller implementation, e.g. NOX, Maestro, Floodlight, etc. The software prototype operates as an overlay between the OpenFlow architecture and its applications. Afterwards, to validate the efficiency of the proposed model, its features are going to be investigated. This is done by deploying the software prototype in the selected use case datacenter. A testbed will be built and the test infrastructure should emulate the activities of a datacenter. Simulation might be necessary given that the available resources are not sufficient to perform tests for a large scale. Experiments will be conducted to measure network throughput, transfer time under different system configurations. The evaluations aim at examining the performance, feasibility, and scalability of the proposed approach in the datacenter. It is expected that the deployment of the software prototype will help the datacenter utilize network resources efficiently.

References

1. Curtis, A.R., Mogul, J.C., Tourrilhes, T., Yalag, P., Sharma, P., Banerjee, S.: Devoflow: Scaling flow management for high-performance networks. ACM SIGCOMM (2011)

2. Jarschel, M., Lehrieder, F., Magyari, Z., Pries, R.: A Flexible OpenFlow-Controller Benchmark. In: European Workshop on Software Defined Networks (2012)
3. Rotsos, C., Sarrar, N., Uhlig, S., Sherwood, R., Moore, A.W.: OFLOPS: An open framework for openflow switch evaluation. In: Taft, N., Ricciato, F. (eds.) PAM 2012. LNCS, vol. 7192, pp. 85–95. Springer, Heidelberg (2012)
4. Heller, B., Sherwood, R., McKeown, N.: The controller placement problem. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks (HotSDN 2012) (2012)
5. Jarschel, M., Oechsner, S., Schlosser, D., Pries, R., Goll, S., Tran-Gia, P.: Modeling and performance evaluation of an OpenFlow architecture. In: Proceedings of the 23rd International Teletraffic Congress (2011)
6. Bianco, A., Birke, R., Giraudo, L., Palacin, M.: OpenFlow switching: data plane performance. In: Proceedings of IEEE International Conference on Communications (2010)
7. Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., Sherwood, R.: On controller performance in software-defined networks. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, Hot-ICE 2012 (2012)
8. Moreira, et al.: Packet forwarding using openflow. In: First Workshop on Network Virtualizaton and Intelligence for Future Internet, WNetVirt 2010 (2010)
9. Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N., Vahdat, A.: Hedera: dynamic flow scheduling for data center networks. In: Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, NSDI 2010 (2010)
10. Open Networking Foundation. https://www.opennetworking.org/membership/members (accessed June 20, 2014)
11. Mattos, et al.: OMNI: OpenFlow management infrastructure. In: Proceedings of the 2nd IFIP International Conference Network of the Future, NoF 2011 (2011)
12. Fernandes, et al.: Multinetwork control using openflow. In: First Workshop on Network Virtualizaton and Intelligence for Future Internet (2010)
13. Yu, M.: Scalable Management of Enterprise and Data-Center Networks. PhD Dissertation (2011)
14. Sherwood, et al.: FlowVisor: A Network Virtualization Layer. OpenFlow Switch (2009)
15. Pisa, P.S., Fernandes, N.C., Carvalho, H.E.T., Moreira, M.D.D., Campista, M.E.M., Costa, L.H.M.K., Duarte, O.C.M.B.: OpenFlow and xen-based virtual network migration. In: Pont, A., Pujolle, G., Raghavan, S.V. (eds.) WCITD 2010. IFIP AICT, vol. 327, pp. 170–181. Springer, Heidelberg (2010)
16. Rotsos, et al.: Cost, performance & flexibility in openflow: pick three. In: Proceedings of IEEE International Conference on Communications (2012)
17. Simeonidou, D., Nejabati, R., Azodolmolky, S.: Enabling the Future Optical Internet with OpenFlow: A Paradigm Shift in Providing Intelligent Optical Network Services
18. McKeown, et al.: OpenFlow: Enabling Innovation in Campus Networks. SIGCOMM Comput. Commun, Rev. (2008)
19. Google Inc.: Inter-Datacenter WAN with centralized TE using SDN and OpenFlow