Load Balancing in OpenFlow-enabled Switches for Wireless Access Traffic Aggregation

Hakan Selvi*, Engin Zeydan†, Gürkan Gür*, and Fatih Alagöz*
*SATLAB, Dept. of Computer Engineering, Bogazici University, 34342, Bebek, Istanbul, TR
†Turk Telekom, Umraniye, Istanbul, TR
Email: {hakan.selvi, gurgurka, alagoz}@boun.edu.tr, engin.zeydan@turktelekom.com

***Abstract*—The immense development in mobile devices and the proliferation of ubiquitous cloud-based services have led to a surge in the utilization of remote data centers. These dramatic advances in wireless based services induce a huge traffic flow within the network. However, consequent high load on the network devices may lead to faults and thus an adverse decrease in the user experience. On the other hand, balancing the aggregated multi-flow traffic on ports of network switches is a challenging issue. This work presents a load balancing algorithm implemented in OpenDaylight controller for managing the load on the switch ports by utilizing Software-Defined Networking (SDN) architecture. The switch acts a connector which aggregates traffic from wireless access nodes. An experiment design is constructed with hardware-based OpenFlow-enabled switch and connected wireless hosts. According to the load on the ports, the developed scheme dynamically load-balances the incoming traffic towards the backhaul connection. It uses OpenFlow for gathering statistics and flow rule installation on the switch in an adaptive manner.**

I. Introduction

The emerging services and systems in the ICT domain leverage powerful user devices for improving user experience and cloud-based back-end for processing-intensive tasks. The user devices in these environments are typically connected to the Internet through wireless access networks. Since the need for computation power and demand for novel services burgeon, there is a huge traffic in the network that poses significant resource and management challenges. The excessive load on the network links or the network devices needs to be balanced for increasing the reliability and preventing possible failures. This is also crucial for improved user experience regarding network-based services. One of the most important approaches for balancing the traffic load is a dedicated middle-box deployed within the network. Since these hardware devices are specialized and it is difficult to relocate them in order to keep up with the dynamic changes within the network, this approach does not provide the desired flexibility and scalability.

As a result of all these challenges and barriers, there is an ongoing research for providing simple yet effective mechanisms in network management and administration. Software-Defined Networking (SDN) is a recent research topic that is promising for addressing such issues [1]. SDN decouples control and data layers by implementing the control mechanism within network-resident software apparatus. The logically centralized controller can have a general view of the network and leverages the dynamic management of the network [2].

978-1-5090-0223-8/16/$31.00 © 2016 IEEE

The current de-facto standard for implementing communications between SDN controllers and switches is the OpenFlow protocol [3]. OpenFlow-enabled switches do not have any control mechanism and needs to communicate with the controller for deciding on network functions. This communication should be provided in a secure manner and it is better to be standardized. The controller may send flow rules to be installed on the flow tables at the switches and switches may forward packets to be processed by the controller for further action if necessary. For network monitoring and management, OpenFlow provides a mechanism for collecting statistics from the underlying infrastructure. Since load-balancing algorithms require load information on the links or the ports, OpenFlow is a natural enabler for implementing load-balancing in IP networks using SDN-enabled switches.

This work focuses on balancing loads on the ports of an SDN switch that aggregates traffic from a wireless access network (i.e. wireless access nodes). In order to achieve this objective, an algorithm is implemented in OpenDaylight controller, which is an open source controller for SDN switches using OpenFlow protocol [1]. The remainder of this study is organized as follows. Section II presents the implemented algorithm and the experiment testbed. Section III discusses the results and concludes the study along with research directions.

II. Load Balancing of OpenFlow-enabled Switch Ports for Wireless Traffic Aggregation

In this section, we present the details of developed algorithm and the experiment environment constructed for testing and demonstrating the load balancing scheme.

A. Load-Balancing Algorithm Design

The load balancing algorithm is implemented for OpenDaylight that is one of the most actively-used and supported controllers for SDN research. The algorithm works in an effective manner and collects the required statistics related to traffic load from the switch ports and installs the flow rules on the switches that is needed for forwarding a portion of the traffic to balance the load among the ports that have access to the backhaul.

The algorithm operation relies on monitoring and taking counter-actions for load balancing and is described below:

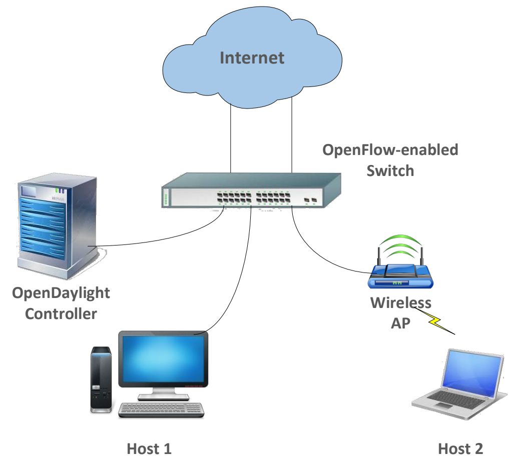

Fig. 1. The general view of the demo environment.

1. Initial flow rules are installed when the controller establishes the connection with the switch.
2. The controller sends a request for statistics to the switch each T_s by using northbound interface provided by OpenDaylight to gather the received byte count information of the port that is used for accessing to the backhaul (for Internet access).
3. When the algorithm detects that the byte count of that port exceeds threshold R_t, it generates new flow rules to be installed on the switch. These rules forward the traffic of one of the hosts to the other port that also has WAN connection.
4. The algorithm continuously sends statistics request each T_s after the load balancing action for further operations.

where T_s=1 sec and R_t= 8 Mbit/sec.

For evaluating the performance and demonstrating the algorithm, an experiment environment is designed and constructed which is presented in Section II.B.

B. Experiment Design

To examine the results and the performance of our algorithm, we set up a test system with a hardware-based OpenFlow switch and run experiments in this environment. The controller is selected as OpenDaylight - Helium version. The SDN switch is an HP 3800-48G-4XG which provides 48-ports. It supports both OpenFlow v1.0 and v1.3 but since the controller version does not support v1.3, we configured the switch to support v1.0 [4].

As mentioned before, there are 2 hosts used for the experiment. One of them is directly connected to switch with an Ethernet cable. On the other hand, the second host is connected to TP-Link AC1750 wireless AP (WAP) and this WAP is connected to the switch with an Ethernet cable. These hosts can access to the Internet and request for various services to generate traffic within the network. Host 1 is connected to the Port 13, WAP is connected to the Port 11 and controller is connected to the Port 1. Moreover, there are Internet connections available in the Port 5 and 7.

Initially, both hosts access to the Internet via Port 5 by matching with the initial flow rules installed by the algorithm. The controller gathers statistics from Port 5 each second and checks whether the load (received byte count) on that port exceeds the threshold mentioned in the previous subsection. When the load threshold is exceeded, the algorithm installs particular flow rules to forward the traffic destined to and sourced by one of the hosts to Port 7 in order to balance the load on the hardware. The experiment environment is presented in Figure 1.

III. Results and Conclusion

OpenDaylight provides a web interface which presents general information related to the network and devices such as flow rules, port statistics and connected hosts. After controller establishes the connection with the switch, the initial flow rules are installed on the switch and it is observed on the web interface.

Both hosts access the Internet and request for various services. These requests and corresponding responses generate a traffic flow on the switch. The algorithm checks for the load information on Port 5. When the received byte count exceeds 8 Mbit/sec, the algorithm installs new flow rules on the switch through OpenFlow messages. All the former rules are removed in order to set port statistics to zero since OpenFlow 1.0 does not provide required methods for achieving the same objective. After the new configurations, it is observed that the Internet connection on the hosts is still available and the load on the Port 7 starts to increase because some of the packets match with the new flow rules and forwarded to the Port 7.

Since this is the initial deployment and OpenFlow 1.0 version does not provide a flexible environment, it is challenging to work on real hardware rather than a simulation. As future work, more than one OpenFlow-enabled switch will be used for the experiments and newer version of OpenDaylight will be utilized for supporting the latest OpenFlow features. Moreover, novel use cases will be implemented in this environment to better evaluate the potential of the algorithm for further improvements.

Acknowledgment

This work is supported in part by TUBITAK TEYDEB 1509 program under grant number 9120067.

References

[1] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, T. Turletti *et al.*, "A survey of software-defined networking: Past, present, and future of programmable networks," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 3, pp. 1617–1634, 2014.

[2] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 7–12.

[3] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using openflow: A survey," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 1, pp. 493–512, 2014.

[4] Open Networking Foundation (ONF), "Openflow specifications," 2016. [Online]. Available: https://www.opennetworking.org/sdn-resources/openflow/, January 2016