Software Defined Networking and Services

Jonathan Lim
University of California, Santa Cruz
California, USA
jflim@ucsc.edu

Katia Obraczka
University of California, Santa Cruz
California, USA
katia@soe.ucsc.edu

***Abstract*— Software-defined networking (SDN) is a new paradigm that aim at making networks programmable by decoupling network control from the data forwarding hardware. OpenFlow, a notable SDN protocol, provides an API that lets a logically centralized control element, called a network operating system or "controller", to organize OpenFlow-enabled switches. Such abstractions promote network innovation by permitting finer-grained control and simplified service deployment over the underlying data plane. In our work, we examine how SDN can be used to provide customized services to users. In order to test and evaluate the performance of such services, we create an OpenFlow testbed consisting of OpenFlow-enabled network elements.**

Keywords—networks; OpenFlow; management; testbed; experimentation

I. INTRODUCTION

Networks play a major role in our lives today. Networks carry lots of traffic and are critical to the success of businesses and institutions. As networks continue to increase in size and complexity, managing networks will be harder. Software-defined networking (SDN) is an architecture that makes networks programmable. Two main ideas associated with SDN is the separation of the data plane from the control plane and the ability to create higher level abstractions. With SDN, network operators can have better control of their networks. SDN will also reorganize the way networks are built upon today so that protocols can be run as applications on top of a network operating system. The aim of this research is to construct an OpenFlow testbed for evaluating the performance of such network services. To experiment with programmable networks, we implemented a simple multipath forwarding algorithm on the controller.

II. BACKGROUND

Prof. Scott Shenker notes that networks have become part of the critical infrastructure of business, schools and homes and "this success is a blessing and a curse" [1]. While network research becomes more relevant, new ideas will make smaller impacts. Any new idea requires successful test results on real-life networks to convince wide-spread deployment of the idea. Since networks today are large and heavily used, many new ideas aren't tested on real-life networks because to since the risk of failure outweighs foreseeable gains. Any mistake can cause networks to crash and cost businesses lots of money. Also, many switches and router manufacturers design closed systems so people cannot implement their own protocols onto their own devices. The lack of new ideas impacting networks leads many to say that the network infrastructure is "ossified" [1]. To counter this problem, Stanford researchers developed OpenFlow [1], an open protocol that allows network operators and developers to partition network traffic on network devices and to insert flows into the proprietary devices while not exposing the inner workings of the devices. OpenFlow provides the network operating system with a standardized way to talk to the devices in the network, regardless of the manufacturer/model.

III. SOFTWARE-DEFINED NETWORKS

In a non-SDN network view, the data plane and the control plane are both in each device. The network is distributed where each device manages its own state and each device has its own operating system and specialized forwarding hardware. A device from one manufacturer might have a different configuration process from a device from a different manufacturer.

In a SDN network view, the data plane and the control plane are separated. OpenFlow switches become simple packet forwarding hardware that perform actions when incoming packets match rules of inserted flows. The controller is logically centralized and will control the network state [2]. The controller will insert and delete flows from the flow tables of the OpenFlow switches using the OpenFlow protocol. Software-defined networking will allow people to customize network services for their needs and to exert finer-grained control over their networks.

This work was supported by the UCSC SURF-IT 2012 Research Experience for Undergraduates Site, NSF grant CNS-1156606.

IV. TESTBED

The testbed consists of three TP-Link TL-WR1043ND gigabit routers, two desktop computers, and a laptop running Floodlight [3] as the controller. The network elements are linked together as shown in Figure 1. As mentioned previously, the control and the data elements are separated. The control links are those that connect the controller to hub and the hub to switche and the data links are the remaining links. One way to turn a commercial switch to an OpenFlow switch is by using Pantou and running OpenFlow as an application on top of OpenWRT[4], an operating system for embedded devices. OpenWRT allows users and developers to choose packages and customize their devices in ways that may not have been possible before. We used a pre-compiled binary image for OpenFlow 1.0 with the TL-WR1043ND, which is offered on the OpenFlow website. For our purposes, three configuration files (network, OpenFlow, wireless) need to be filled in in order for the testbed to function. In the network file, we used the default network configuration to set five wired ports. In the OpenFlow configuration file, some ports of TL-WR1043ND are designated to be OpenFlow ports so that the controller can use them to send flows on. The controller's IP address is also specified along with relevant information such as mode and datapath ID. The wireless file is configured for a simple WLAN and will be modified in the future as it has not been used in the testbed yet.

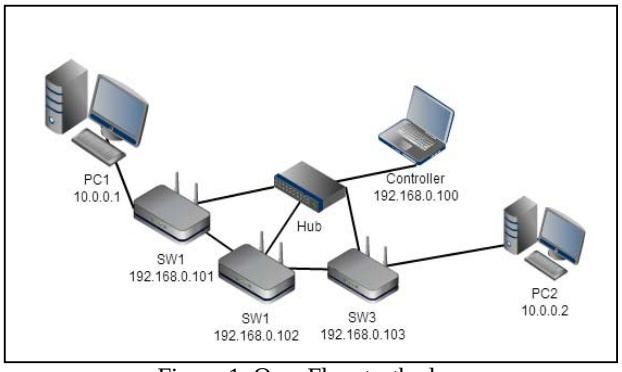

Figure 1: OpenFlow testbed

V. MULTIPATH FORWARDING

The multipath forwarding algorithm was implemented in the forwarding module of Floodlight. It runs on top of the controller similar to an application. We were motivated to compute multiple paths for a given source/destination pair to partially load balance on the network. At the time of this paper, the Floodlight forwarding module computes the shortest path based on hop count. By inserting a flow with multiple paths, network traffic can be shared on multiple paths. The multiple path selection algorithm used was the Multipath Dijkstra algorithm [5]. The algorithm is similar to the Dijkstra algorithm except it adjusts the weights of each link depending on whether the link was selected for a previous path or not. Different values can influence the number of different paths found.

VI. CONCLUSION AND FUTURE WORK

The testbed functions correctly when it is directly connected with cables. Setting up the testbed wirelessly will allow us to connect with other testbeds such as the larger OpenFlow testbed at INRIA Sophia Antipolis and expand our testing environment greatly. We also plan to deploy additional software-based switches into our testbed. The multipath forwarding function was intended to calculate multiple paths based on different cost metrics but we were not able to collect enough metrics to implement the idea further. As future work, we plan to explore ways to customize services further which will enable the controller to handle host application needs. We plan to export an API that allows host-to-controller communication for transmitting relevant information. This includes determining the appropriate cost metric to use and other options such as multiple v. single path choices.

ACKNOWLEDGMENT

I would like to thank Marc Mendonca for mentoring and answering questions I had regarding SDN and OpenFlow. I would also like to thank the Floodlight development for answering questions regarding controllers.

REFERENCES

[1] N. McKeown, et al., "OpenFlow: Enabling Innovation in Campus Networks", ACM SIGCOMM CCR, Vol. 38, Issue 2, March 2008.

[2] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, and N. McKeown. NOX: Towards an Operating System for Networks. ACM SIGCOMM CCR, July 2008.

[3] Floodlight OpenFlow Controller. http://floodlight.openflowhub.org/.

[4] OpenWRT community. The OpenWRT Project Homepage, 2008. http://openwrt.org/.

[5] J. Yi, A. Adnane, S. David, and B. Parrein, "Multipath optimized link state routing for mobile ad hoc networks", Ad Hoc Netw. (2010).