PROJECT: NETWORK OPTIMIZATION USING SDN

REPORT

INTRODUCTION

The goal of this project is to analyze and optimize the network flow for two specific protocols namely SSH and HTTP using SDN. This goal was achieved by using Mininet to emulate a network topology consisting of open vswitches, hosts and use POX Open flow controller to install rules on those Open vswitches

MEASUREMENT SUITES

In order to assert our optimization is valid for the two protocols, measurement suits were created for HTTP and SSH protocol. These measurement suits were python application consisting of server and client written in such a way that resembles the exchange of packet been done in a real protocol.

HTTP PROTOCOL

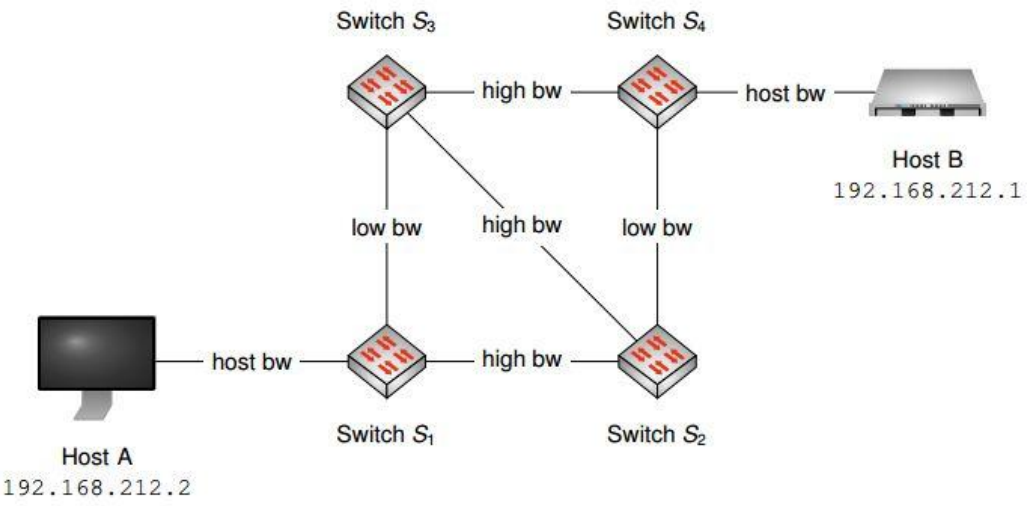
The total number of HTTP GET requested by httpclient to the httpserver during a single protocol run is 3. The maximum size of a packet during one protocol run is 8 KiB. The packet sent by httpserver range from 1 KiB to 8 KiB. Average duration for one protocol run of httpserver and httpclient on local host (127.0.0.1) is around 4 milliseconds.

SSH PROTOCOL

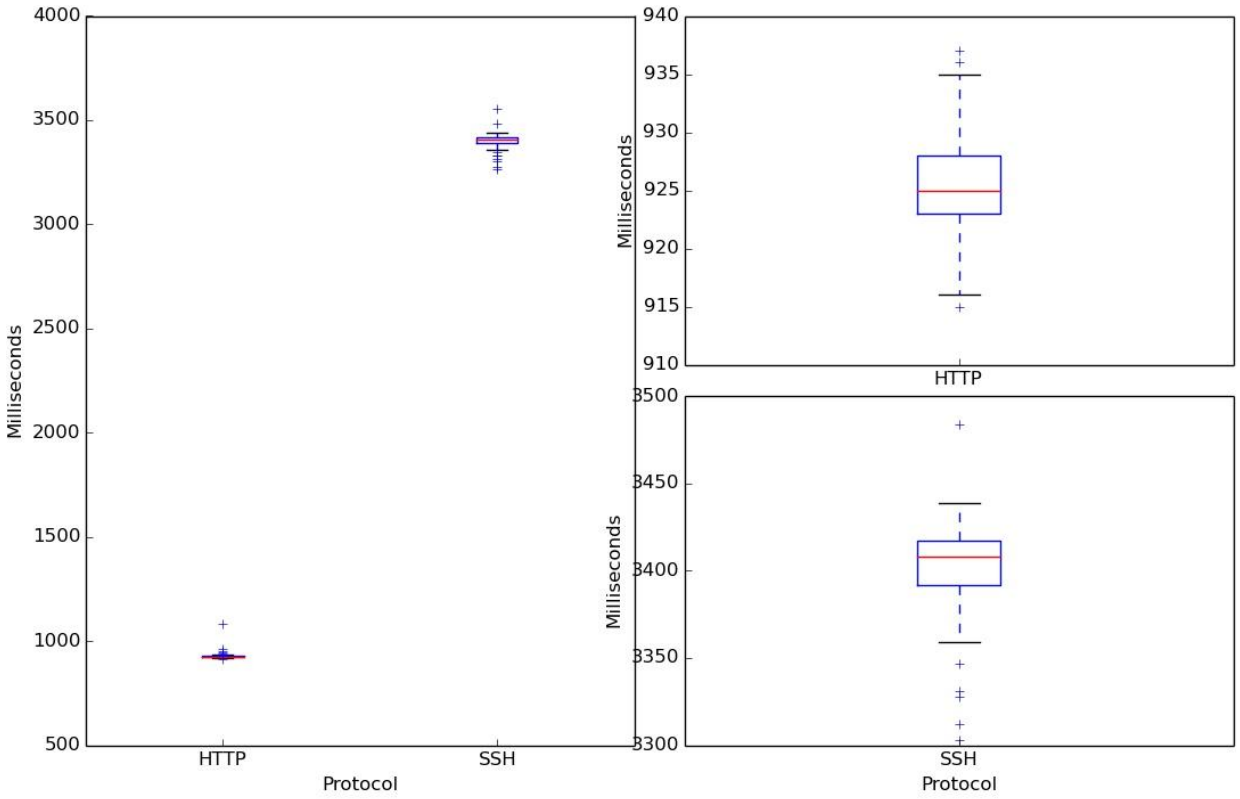
The total number of commands sent by sshclient to the sshserver during a single protocol run is 10. The maximum size of a packet during one protocol run is 300 Bytes and they are usually within the range of 10-300 Bytes from server and 4-20 Bytes from the client. Average duration for one protocol run of ssherver and sshclient on local host (127.0.0.1) is around 212 milliseconds.

MININET TOPOLOGY

The following Mininet topology was created as shown in below figure. 'Host bw' link was configured to have bandwidth of 1 Mbit/s and had delay of 1 ms. 'High bw' link was configured with bandwidth of 512kbit/s and delay of 20ms. Similarly 'low bw' link was configured with bandwidth of 32kbit/s and delay of 1ms. In order to avoid the packets to infinitively loop around the network, flooding of packets was disabled between links (s3, s4) and (s1, s2).



The following box plot shows the results after running the measurement suite on the above topology with server running on Host B and client running on Host A.



From the above box plot we can see that the average protocol duration for HTTP is around 925 milliseconds and for SSH is around 3410 milliseconds.

CONTROLLING TRAFFIC FLOW USING POX CONTROLLER

To further continue on the analysis the given topology was divided into two paths one with High Bandwidth and the other with Low latency. Custom pox controller was written to separate the flows of the two protocol in two different path.

High Bandwidth path from Host A to Host B:

Host A →S1 →S2 →S3 →S4 →Host B

Low Latency path from Host A to Host B:

Host A → S1 →S3 →S4 →Host B (Or) Host A →S1 →S2 →S4 →Host B

ANALYSIS

We can see from the Measurement Suites that for one protocol run the packets exchanged in HTTP are larger (in range of KiBs) compared to that of SSH (in range of Bytes), but the number of packets exchanged in SSH (around 20 packets) is larger compared to number of packets exchanged in HTTP (around 6 packets). In other words, HTTP results in less number of packets of larger size being exchanged and SSH results in lot of packets of smaller size being exchanged.

HTTP MEASUREMENTS

Since HTTP exchanges less number of packets of larger size, duration of one protocol run for HTTP on High Band width path is less compared to that of Low Latency path. This is because even though we have a low latency link when the bandwidth of the link is not too high it takes longer time for the packets to reach destination because of their large size. The average time taken in milliseconds for one protocol run on High Band width path is between 880 - 900 ms and on Low latency path is between 4040 - 4050 ms. The results are captured in Fig 1.

SSH MEASUREMENTS

Since SSH exchanges more number of packets of less size, the duration of one protocol run for SHH on Low latency path is less compared to that of a High bandwidth path. This can also be reasoned the same way as above. Even though we have a High bandwidth link when the latency of the link is not too low, it takes longer time for the protocol run because of the number of packets being exchanged. The average time taken in milliseonds for one protocol run on Low latency path is between 1470 - 1500 ms and on High bandwidth path is between 3370 - 3400 ms. The results are captured in Fig 2.

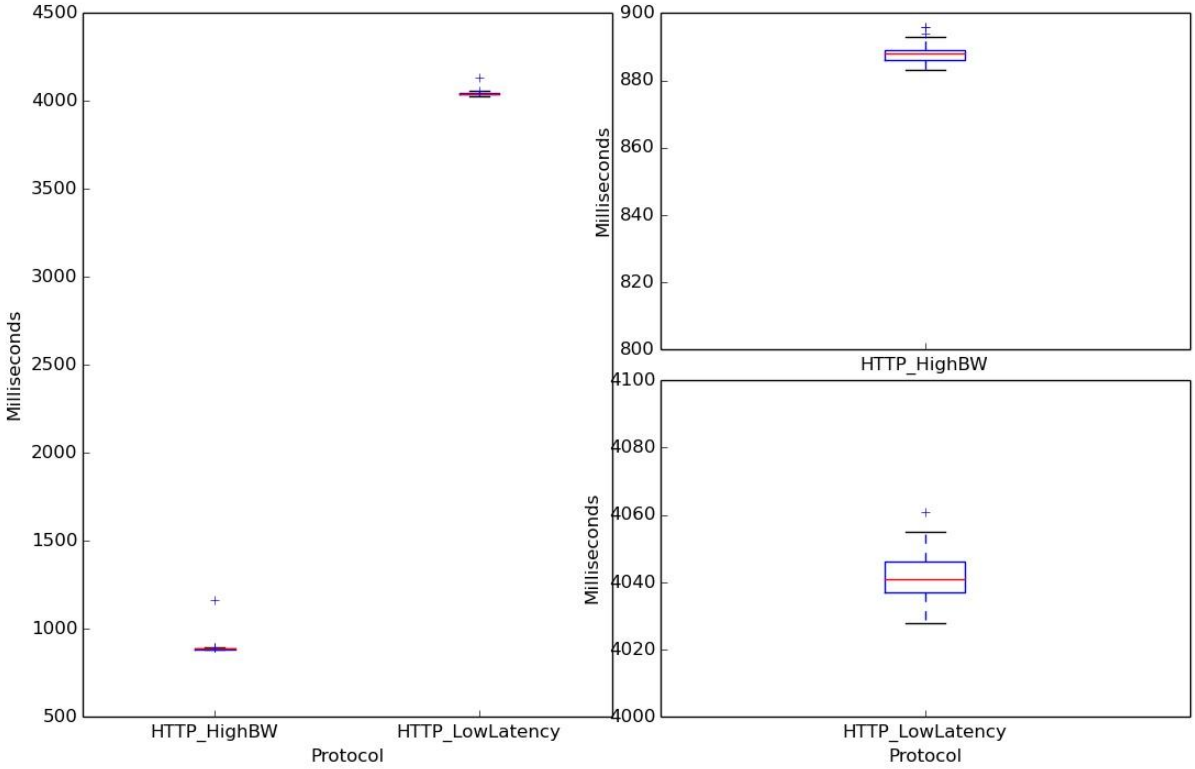


Fig 1. HTTP on High Bandwidth and Low Latency

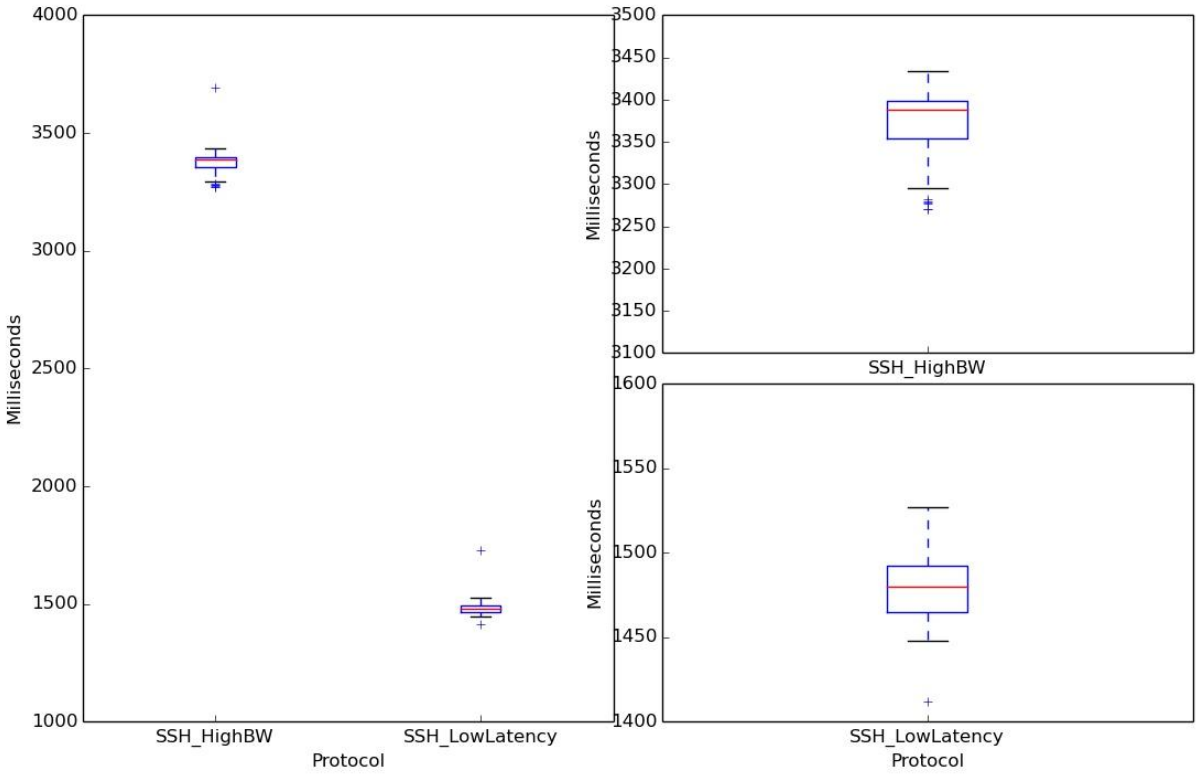
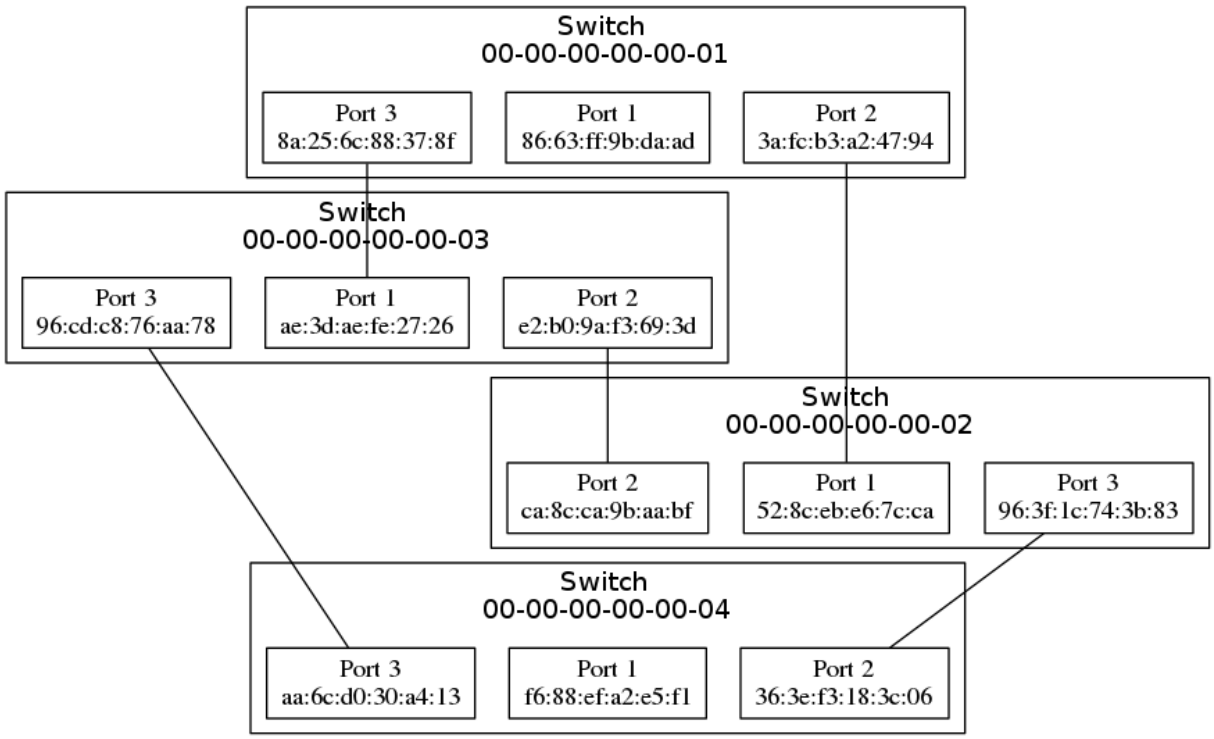


Fig 2. SSH on High Bandwidth and Low Latency

TOPOLOGY DISCOVERY

As a sub task, another generic pox controller was written to discover any given Mininet topology. Switches present in the topology were identified during the connection up event. The links between the switches were identified using LLDP mechanism. The controller instructs each switch to sends LLDP packets through each of its ports. The switches connected to other end of the link received this LLDP packets and triggered packet in event in controller. This mechanism was used in the controller to discover the link between any two switches. The discovery of switches and links were stored in respective data structures and later on request was exported in dot file format which can be drawn using graphviz library.

The following diagram was drawn using graphviz after discover of the problem topology.



FEEDBACK

HOW MUCH TIME DID YOU INVEST INTO THIS PROJECT?

I spent around 50 to 60 hours for this project. This time includes learning, analysis, experimenting and coding.

WAS IT TOO EASY/DIFFICULT OR JUST RIGHT?

I think it was just right for a Masters course. Since I was new to python, it was slightly time consuming to learn python programming initially.

WHAT DID YOU LIKE/DISLIKE ABOUT THE PROJECT?

I liked coding in python, learning SDN and Mininet concepts. Among the programing tasks, I liked designing the controller to discover the network topology and providing the output in graphical representation. I liked the general idea of using python libraries to display the results in a graphical format.

There is a huge delay between submission of the project and getting the grades. As each sub tasks of this project are interrelated if the previous project submission has some code defect the next submission would have the same code. Giving early feedbacks can avoid this problem.

WHAT WOULD YOU DO DIFFERENT NOW?

I would give bandwidth and latency of the links more importance during analysis of any protocol in computer networks.

I would prefer python as my programming language for implementing network related programs as I feel it is easy to learn and use.

I would also document my work. For example, documenting the usage of server and client if they have a dependent external library. I lost grades for problem 2 because my ssh application required external library and I didn't not document them during submission.

WHAT WOULD YOU TELL PEOPLE WHO STILL HAD TO DO THIS ASSIGNMENT?

I would tell this project is great start for understanding the real time working of switches in a network. Working concepts of a protocol read in theory can be practically experienced in this project. Also understanding concepts of bandwidth, latency and how the network links can be effectively used for protocols with different properties.

If you are new to python, I would also say this project would be a great start to learn python programming language.