Implementation and analysis of network functions in a virtualized topology using RYU controller
Dr. Nader F. Mir, Pinak N. Karadkar
Department of Electrical Engineering, San Jose State university, San Jose, California 95192.

Introduction
Virtualization has been studied and developed to change the way networks operate by enhancing standard IT technologies to consolidate network component types in large servers, storage devices and switches. In traditional network establishments such as routers, firewalls and content delivery appliances, each function was implemented by a certain set of software and hardware. However, the software and hardware for such networks was interdependent, viz., they cannot be separated easily.

The primary goal of this project is to implement a routing functionality on a virtualized network topology using simulation software Mininet and OpenFlow controller RYU. The parameters that would be analyzed are source and destination IP address, link capacity, bandwidth usage, host reachability, etc. In addition to this, security features such as firewall implementation and packet filtering is also discussed extensively. As an application, this project may also be integrated with a web GUI to ease the reading of all the aforementioned parameters.

Methodology
The traffic that the hosts generate passed through the virtual switches in mininet. RYU controller extracts this information using the code that was developed for the project in Python. RYU is an open source controller and is freely available on github using any of the following commands.

% pip install ryu
For installing from source code;
% git clone git://github.com/osrg/ryu.git
% cd ryu; python ./setup.py install

This topology can be implemented or created by using the Mininet command. As mininet works on a single Linux operating system, the command is preceded by a $ for shell prompt and # for shell prompt. Mininet precedes mininet command that will be entered in the shell prompt. The sudo mn command creates a mininet topology. The commands that are concatenated to it are external features to be added. The topology for the command that we use in this network implementation looks like this:

```bash
sudo mn --topo single, < number of hosts> --mac --switch ovsk
```

The firewall execution is displayed at the successful output of the firewall enable command. While the result is displayed, the parameters can be observed that include the details of firewall running after which the rules will be added to the firewall.

Rules can range in many ways

The packets that were monitored using the Wireshark tool and the packet generation using iperf utility tool are also focused and analyzed in this section. Graphical analyses are more accurate and lets the user understand the exact parameters and their performance evaluation and their role in the project. This section plays a very important role in the conclusion of the project.

Results
The packets that were monitored using the Wireshark tool and the packet generation using iperf utility tool are also focused and analyzed in this section. Graphical analyses are more accurate and lets the user understand the exact parameters and their performance evaluation and their role in the project. This section plays a very important role in the conclusion of the project.

The packets that were monitored using the Wireshark tool and the packet generation using iperf utility tool are also focused and analyzed in this section. Graphical analyses are more accurate and lets the user understand the exact parameters and their performance evaluation and their role in the project. This section plays a very important role in the conclusion of the project.

The figure below shows the CPU utilization of the designed system while implementing the firewall functionality.

```plaintext
4.0 µs - Means that the processor is utilizing or spending 4% of its time running user space processes (does not belong to the kernel). Most of the times, when the CPU is not idle, it spends its time running user space programs. [19]
```

Conclusions
The main focus of the project was to observe and analyze how security really works in present infrastructures and how it can be implemented on a virtual environment. The following conclusions can be drawn after the implementation:

1. CPU utilization: The amount of CPU time used in the designed system is much lesser than the hardware equivalent of firewall routers or packet filters.
2. Invoking RYU and extracting data: RYU, which also follows the OpenFlow protocol scheme is successfully invoked and establishes connection with the Open vSwitch in order to extract the data from the flow tables of the switches. The controller successfully manages and controls the switches and helps in the network functions to achieve their tasks. A python module is implemented to perform certain functionalities. This module allows the controller to fetch flow table data from the switches.

Key References
3. IEEE Computer Society

Acknowledgements
The author would like to take this opportunity to thank Prof. Nader F. Mir for his valuable guidance and inputs that helped in the successful completion of the project.