SDN Based Monitoring and Filtering Application

Prof. Chao-Li Tarng, Anudeep Reddy Ramashayam, Sneha Viswalingam
Department of Electrical Engineering, San Jose State university, San Jose, California 95112.

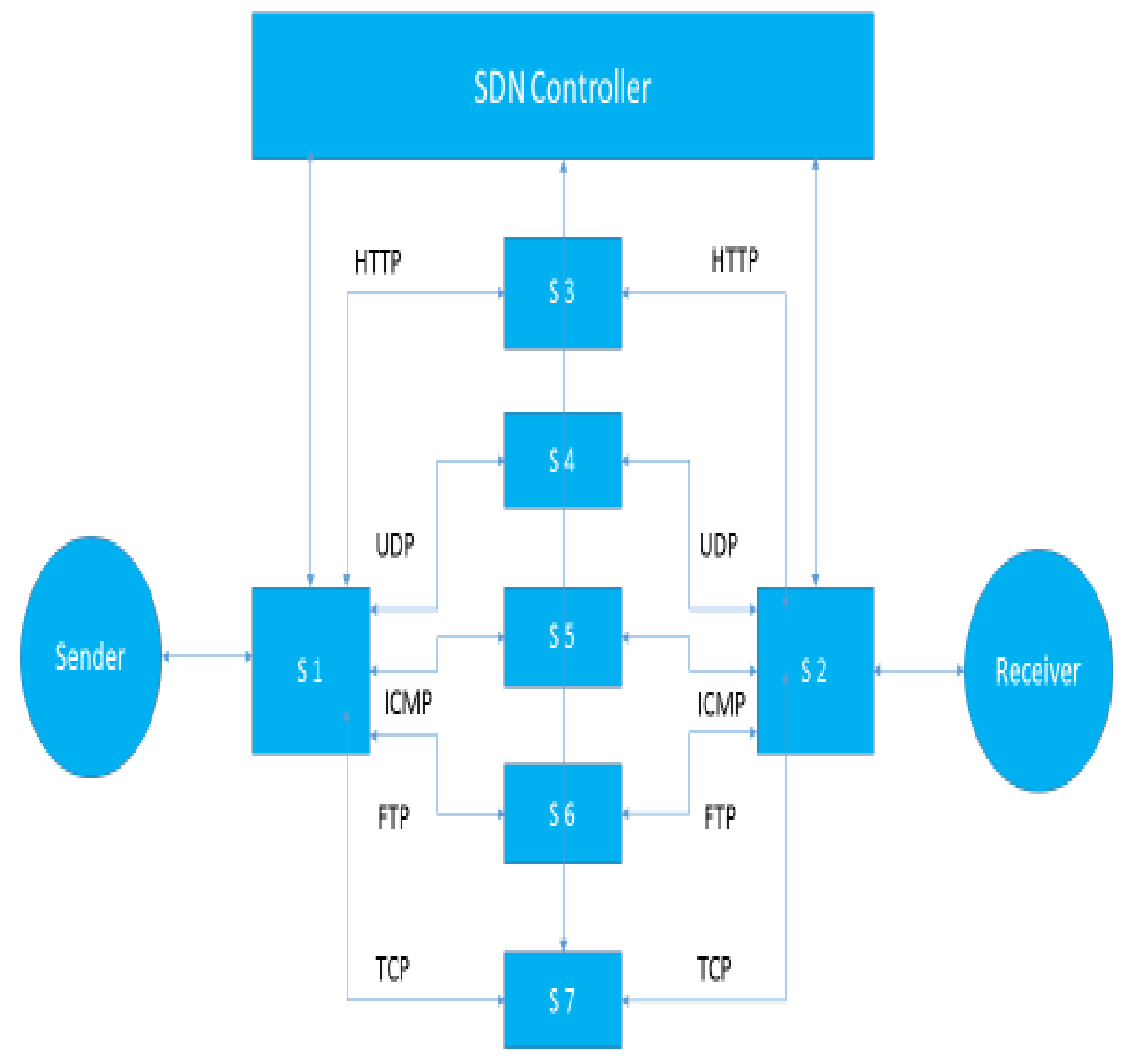
Introduction

Traditional networks were designed to forward packets from source to destination using the shortest route possible. Routers and switches were mostly agnostic to the applications being served by the network. Today's networks are forced to be application-aware, to improve user experience, provide service differentiation, and reduce operational costs. Software-defined network (SDN) architecture allows service providers to build networks with increased application awareness, which can be built into the network by developing SDN controller applications that keep track of application-level characteristics and use that intelligence to provision flow into the network switches.

The main objective of this project is to build an application of monitoring and filtering that's based on Software Defined Networking (SDN). This application uses a controller to observe the headers of IP Packets in the flow and differentiates them with their protocol and sends those flows along their respective switches that are already defined.

Design

Network Architecture



Implementation

We have created different paths for TCP, UDP and ICMP traffic. We have further classified TCP traffic in to HTTP and FTP traffic. For each different type of traffic, we have assigned a dedicated path.

When the switch forwards the first packet in the flow it receives to the controller, the controller checks the Ethernet packet for IP content. If it's indeed an IP packet, the controller then checks the protocol header to determine if it's a TCP, UDP or an ICMP packet. If the packet is a TCP packet, then it further determines if the packet is HTTP or FTP by inspecting the port number. After the controller determines the type of packet, it instructs the switch to send the packet along a specific path/route. Only the first packet from a flow is sent to the controller, as in Pyretic/POX. Thereafter, a rule is installed on the switch for the flow. This allows the flow to be dealt with by the switch itself, saving the controller from having to deal with too many packets.

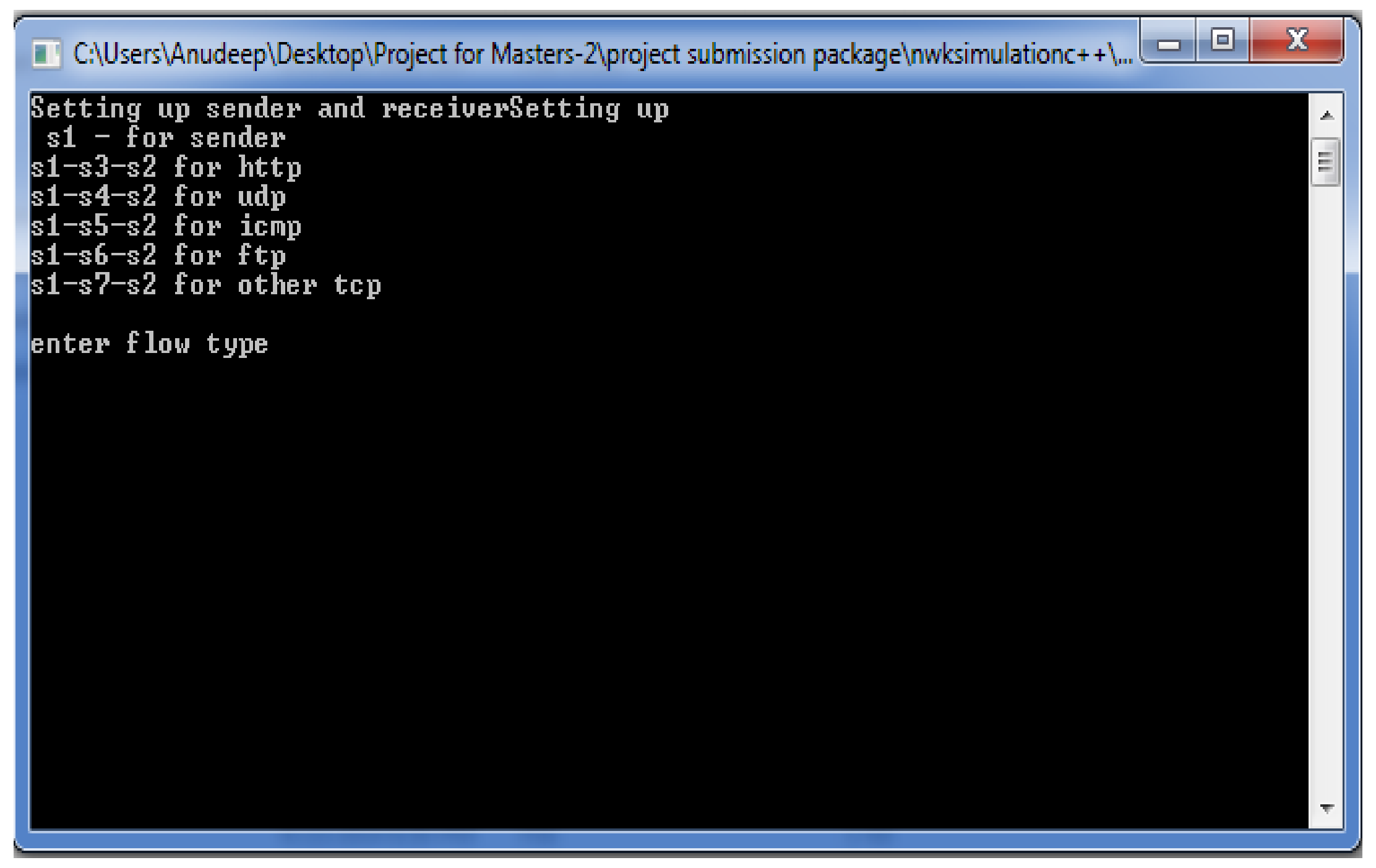
According to our topology:

- HTTP packets are routed along S3.
- FTP packets are routed along S6.
- Other TCP packets are routed along S7.
- UDP packets are routed along S4.
- ICMP packets are routed along S5.

We identify different packets as being TCP/UDP using the identifier in the IP header for the transport protocol, after IP packets themselves are identified using an identifier in the Ethernet header. The applications are identified using the server port being contacted by the client. This can be termed as Deep Packet Inspection method.

Results

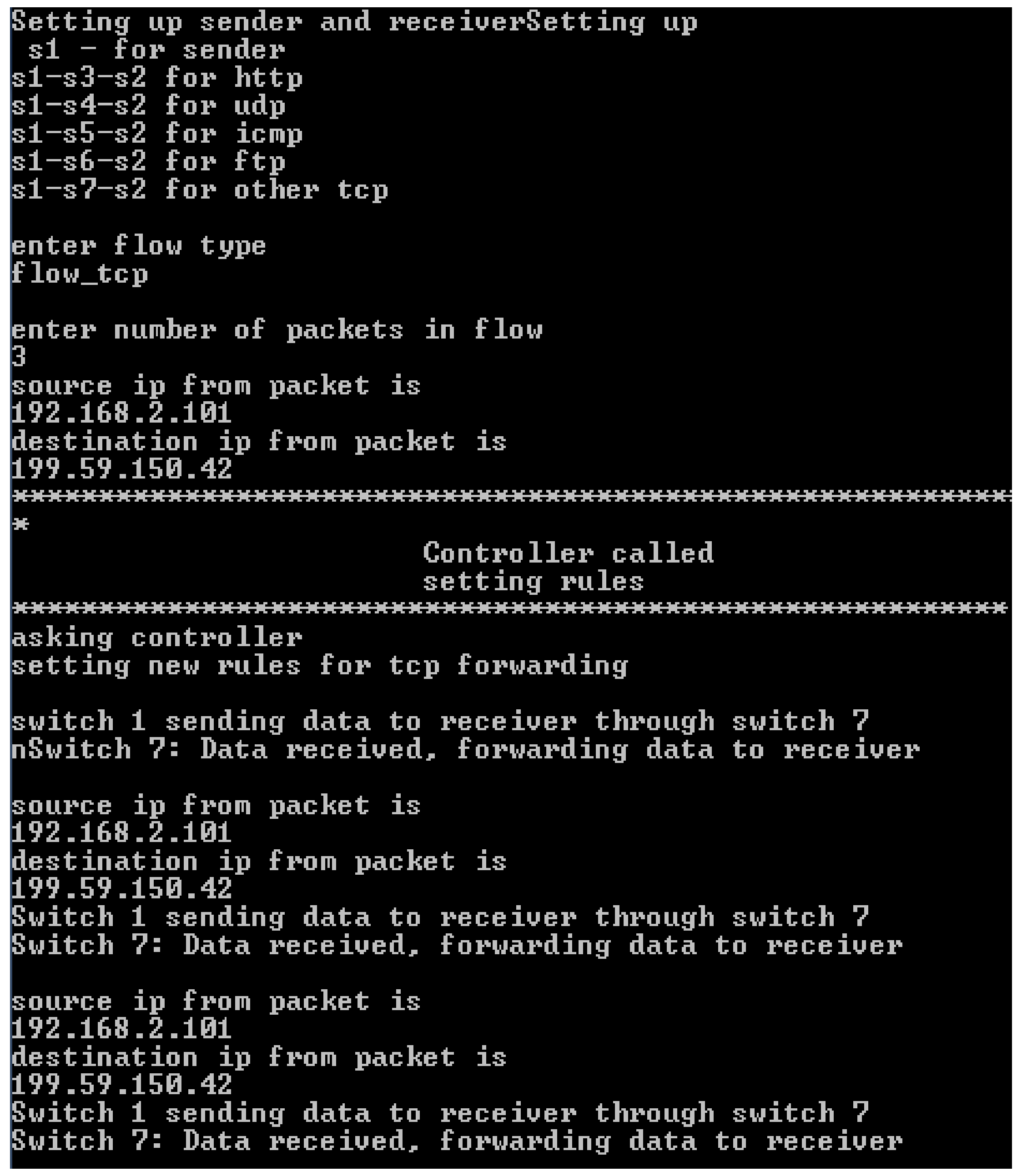
Home screen of our application is as follows:


C:\Users\Anudeep\Desktop\Project for Masters-2\project submission package\nwksimulationc++\...

```
Setting up sender and receiverSetting up
 s1 - for sender
s1-s3-s2 for http
s1-s4-s2 for udp
s1-s5-s2 for icmp
s1-s6-s2 for ftp
s1-s7-s2 for other tcp

enter flow type
```

When we enter a flow type like TCP and number of packets like 3:



```
Setting up sender and receiverSetting up
 s1 - for sender
s1-s3-s2 for http
s1-s4-s2 for udp
s1-s5-s2 for icmp
s1-s6-s2 for ftp
s1-s7-s2 for other tcp

enter flow type
flow_tcp

enter number of packets in flow
3
source ip from packet is
192.168.2.101
destination ip from packet is
199.59.150.42
********************************************************************************
*
                              Controller called
                              setting rules
********************************************************************************
asking controller
setting new rules for tcp forwarding

switch 1 sending data to receiver through switch 7
nSwitch 7: Data received, forwarding data to receiver

source ip from packet is
192.168.2.101
destination ip from packet is
199.59.150.42
Switch 1 sending data to receiver through switch 7
Switch 7: Data received, forwarding data to receiver

source ip from packet is
192.168.2.101
destination ip from packet is
199.59.150.42
Switch 1 sending data to receiver through switch 7
Switch 7: Data received, forwarding data to receiver
```

Only the first packet is been sent to the controller and the controller sets up a rule back on switch observing that first packet and the rest of the flow need not rely on controller that maintains some load off from controller. Any number of packets can be sent in the flow. It's the same functionality for rest of the flows as like shown above for TCP.

Advantages of using SDN

Traditional routing techniques are not application-aware. The application-deliver infrastructure in today's internet is loosely-coupled with the packet forwarding infrastructure, which leads to duplication of functions in the network. This, in turn, impacts user experience, and increases operational costs. The request routing techniques in use today are –

- Policy-Based Routing (PBR)
- BGP Route Advertisement
- Use of a Centralized Request Router

They have many limitations that results in unused bandwidth, increase in CAPEX and Latency not considered by L3 routers when routing the requests.
Software-defined network (SDN) architecture allows service providers to build networks with increased application awareness, which can be built into the network by developing SDN controller applications that keep track of application-level characteristics and use that intelligence to provision flow into the network switches.

Summary

SDN technology opens up an array of opportunities for network service providers. It helps them roll out monetizable services in their networks faster without having to depend on network equipment manufacturers. Application-aware routing architecture using SDN allows service providers to lower CAPEX/OPEX, and improve the overall end-user experience along with many other advantages. We could successfully implement the required architecture in C++ by re-developing the controller and getting the required results.

Key References

1. Application-Aware Routing in Software Defined Networks by Saro Velrajan, Director of Technology, Aricent. http://www.aricent.com/sites/default/files/pdfs/Aricent_Whitepaper_-_Application_Aware_Routing_in_SDN.pdf
2. Big Tap Data Sheet by Big Switch Networks: http://bigswitch.com/sites/default/files/sdnresources/bigtapmfdsenfeb2014_0.pdf
3. Kim, H., & Feamster, N. (2013). Improving network management with software defined networking. *Communications Magazine, IEEE*, *51*(2), 114-119.

Acknowledgements

We would like to thank Professor Chao-Li Tarng, our project advisor, for all his support throughout the project cycle. He has been extremely helpful whenever we needed him. We also thank Prof. Nader Mir, our project co-advisor for extending his support during project report validations. We also like to thank Prof. Thuy T. Le and Prof. Morris Jones for helping us whenever we needed any information.