Programmable Data Plane Using P4 Language

Naga Ganesh Kurapati, Rakesh Musalay, Ramanand Shankarling

Computer Engineering

University of Massachusetts Lowell

Abstract— **Software Defined Networking gives network engineers and operators programmatic control over their networks. In SDN, the control plane is separated from the forwarding plane. The idea is to use P4, a domain specific programming language to design packet encapsulate/de-capsulate modules that can process packet headers and identify specific type of packets, following which further actions can be defined.**

Keywords— **P4, SDN, Parse, Video stream, Packets**

I. INTRODUCTION

P4 is a declarative language for telling forwarding-plane devices (switches, NICs, firewalls, filters, etc.) how to process packets. Essentially, P4 aims to bring in the benefits of software engineering (composing programs, debugging, code coverage, provable behaviour, model checking, etc.) in the design of network systems. P4 program describes the way in which the packets need to be processed by the systems and tell the forwarding elements what to do.

This project will use P4, to design packet encapsulate/de-capsulate modules that can process packet headers and perform the following tasks

a. Identify OpenSSL flows
b. Identify video streams from/to a streaming server
c. Identify YouTube streams

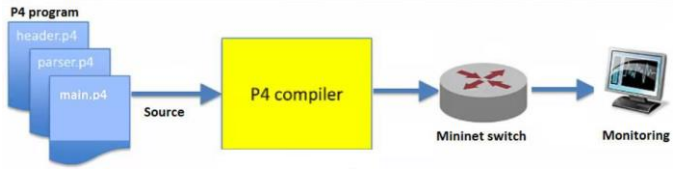

Fig. 1 Block diagram of implementation.

II. TOOLS

A brief account of various tools and software used in the implementation of this project is given below.

A. *Mininet*

Mininet is a network emulator which creates a network of virtual hosts, switches, controllers, and links. Mininet hosts run standard Linux network software, and its switches support OpenFlow for highly flexible custom routing and Software-Defined Networking. Oracle VM VirtualBox software is used to set up Linux VM which includes the Mininet and other useful tools preinstalled. Also, installed other necessary tools like Putty – for secure shell connection and XMing – for graphical terminal of Linux OS in windows machine.

B. *Streaming server – Mist*

MistServer is a opensource streaming media server that works in any streaming environment. The MistServer supports various protocols and formats to stream video packets among which RTMP, HTTP are free and default for the opensource version. In our project, MistServer is used to as a streaming source for the video packets and is installed in a ubuntu virtual machine. The configuration settings and the video stream uploads are done using a web interface which runs in the local host of the virtual machine.

C. *TCPReplay suite*

TCPReplay is a suite of BSD GPLv3 licensed tools for UNIX (and Win32 under Cygwin) operating systems which gives you the ability to use previously captured traffic in libpcap format to test a variety of network devices. It allows you to classify traffic as client or server, rewrite Layer 2, 3 and 4 headers and finally replay the traffic back onto the network and through other devices such as switches, routers, firewalls, NIDS and IPS's.

D. *Scapy*

Scapy is a packet manipulation tool for computer networks, written in Python. It can forge or decode packets, send them on the wire, capture them, and match requests and replies. It can also handle tasks like scanning, tracerouting, probing, unit tests, attacks, and network discovery.

E. *P4 compiler*

P4 is a programming language designed to allow programming of packet forwarding planes. P4 compiler is used to translate specifications of logical packet processing specifications from P4 to low-level switch configurations.

III. ENVIRONMENT SET UP

The steps involved to set up the environment for the project using the tools described earlier is given below.

Mininet is used to set up a simple topology containing a switch S1 and two hosts H1 and H2.

A. *Set up on host H1*

The Mist Controller is started on host H1. This will enable the host H1 to start streaming the video packets from the Mist server for the configured protocol.

Tcprewrite tool is used in the project to edit the source/destination MAC and IP addresses for YouTube QUIC and SSL packets. The source and destination IP addresses are changed to the IP addresses of the host H1 and host H2 in the mininet topology.

Scapy is started host H1. It is used to replay the pcap file for QUIC and SSL packets which contain modified source and destination addresses.

B. *Set up on host H2*

VLC media player is setup on the Host H2 to play the video packets sent by the streaming server. The RTMP video URL is taken from the MistServer and provided to the VLC media player. On play, the video is decoded and played on the VLC player in the RTMP format.

Wireshark is used to analyse the input packet stream from the host H1 through the switch S1. The packets are analysed and stored as pcap files for analytics.

C. *Set up on switch S1*

The switch S1 has P4 compiler installed. S1 intercepts the packets before routing them to host H2. The P4 program designed will parse the incoming stream of packets and is able filter specific type of packets as per requirement. The program can be modified to define actions for filtered packets.

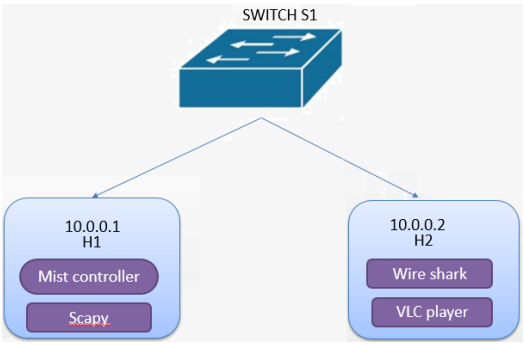

Fig. 2 Mininet topology with environment set up.

IV. DESIGN

In a generic description, P4 program consists of following key components.

- Headers describe sequence and structure of a series of fields.
- Parsers specify how to identify headers and valid header sequences within packets.
- Tables perform packet processing using fields for match.
- Actions support for construction of complex actions from simpler protocol-independent primitives.
- Control programs determine the order of match and action tables that are applied to a packet.

The program designed consists of following files.

A. Header file

- The requirement of the project is to identify streaming packets from YouTube (QUIC protocol) which uses UDP packets. TCP protocol is used in RTMP and SSL packets.
- The header file contains header definitions for TCP and UDP packets. The header definitions provide information on length of different pre-defined fields for specified protocol packets.

B. Parser file

- The parser program parses the incoming stream of packets on the ingress pipeline. Parser decapsulates the packets by examining header types and would be able to differentiate between UDP and TCP packets.

C. Main file

- The main program consists of match conditions defined for QUIC, RTMP and SSL packets.
- The actions for each of these packets are also defined. The action block is executed whenever corresponding match is found.
- The program will be to parse the incoming packets and check for match conditions. For simplicity, in our project, action defined is drop for these filtered packets.
- Also, counters are defined for each of the mentioned protocols. Whenever, match is found, corresponding counter value is incremented and stored in the table.

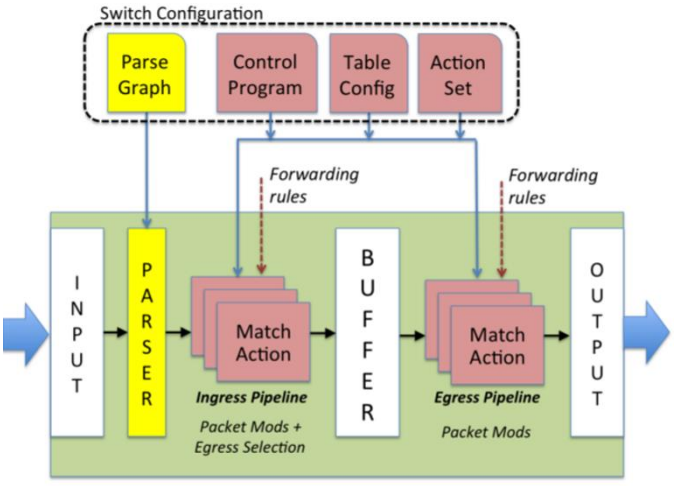

Fig. 3 Abstract forwarding model.

V. Evaluation

A. Identification of SSL flows
- The SSL packet flows are captured using wireshark after opening an SSH connection from SSH client to SSH server used for demo purposes.
- The source and destination IP addresses of the SSL flows are edited to be of host H1 and H2 respectively, using TCP rewrite and are stored in pcap file to be used by Scapy.
- Scapy is used for sending all the modified SSL flows from host H1 to the network interface of switch S1.
- By default, the SSL packets would be able to reach H2 through switch S1. This can be checked by running Wireshark on the host H2.
- Now, to filter SSL packets at S1, we will run bash file which adds the entries for match+action table for SSL packets.
- Again, we will use Scapy to send SSL packets from H1 to H2. But, this time S1 will be able to identify the SSL packets and drop them.
- This can be verified with wireshark on host H2 which will not receive any SSL packets as they are filtered at the switch S1.

B. Identification of video streams from MistServer
- The RTMP packets are streamed directly form Mist Controller running in the host H1. The video to be streamed is uploaded to the MistServer and RTMP is chosen as the streaming format.
- The streamed packets are captured and rendered at the host H2 by running the VLC media player. Once the link of the RTMP video stream is provided in the VLC stream player, the video will start to play in the VLC console.
- Now, to filter RTMP packets at S1, we will run bash file which adds the entries for match+action table for RTMP packets.
- Once the entries are added, the switch S1 will identify RTMP packets and drop them (since action defined is drop).
- This is verified at H2 as the streaming gets stopped and video play on VLC is halted.

C. Identification of YouTube streams (QUIC)
- The YouTube packets, which use QUIC protocol for streaming, are recorded in wireshark from the source www.youtube.com for a sample video.
- The source and destination IP addresses of the QUIC packets are edited to be of host H1 and H2 respectively, using TCP rewrite and are stored in pcap file to be used by Scapy.
- Scapy is used for sending all the modified QUIC packets from host H1 to host H2.
- By default, the SSL packets would be able to reach H2 through switch S1. This can be checked by running Wireshark on the host H2.
- Now, to filter QUIC packets at S1, we will run bash file which adds the entries for match+action table for QUIC packets.
- Again, we will use Scapy to send QUIC packets from H1 to H2. But, this time S1 will be able to identify the QUIC packets and drop them.
- This can be verified with wireshark on host H2 which will not receive any QUIC packets as they are filtered at the switch S1.

VI. Conclusion

- The project implementation is successful. The designed program uses P4 language to parse and identify RTMP, SSL and QUIC protocol packets, and action defined is to drop the filtered packets.
- Also, the program implements counter to keep record of number of each of the packet types.
- However, the challenge was to visualize and read the counter values from the table.
- Documentation and detailed information on our implementation can be found on below link: "ganeshkurapati/Programmable-Data-Plane-using-P4-Language," GitHub. [Online]. Available: https://github.com/ganeshkurapati/Programmable-Data-Plane-using-P4-Language.

VII. Future Work

- The counter data needs to read which can used for traffic analysis and implement reporting of statistics.
- Since, the project is able identify specific type of packets, the same with combination of controller, could be extended to implement contextual routing.

Acknowledgment

We wish to acknowledge and thank Prof Yan Luo, Department of Electrical and Computer Engineering, University of Massachusetts Lowell, for his able guidance in implementation of our project.

References

[1] "Let's get started | P4." [Online]. Available: http://p4.org/p4/lets-get-started/. [Accessed: 04-May-2017].
[2] "Mininet: An Instant Virtual Network on your Laptop (or other PC) - Mininet." [Online]. Available: http://mininet.org/. [Accessed: 04-May-2017].
[3] "mininet/mininet," GitHub. [Online]. Available: https://github.com/mininet/mininet. [Accessed: 04-May-2017].

[4] “MistServer - Documentation, guides and manuals.” [Online]. Available: https://mistserver.org/documentation. [Accessed: 04-May-2017].
[5] “MistServer - Simple, smart and stable media streaming server software.” [Online]. Available: https://mistserver.org/. [Accessed: 04-May-2017].
[6] “p4language,” GitHub. [Online]. Available: https://github.com/p4lang. [Accessed: 04-May-2017].
[7] P. Bosshart et al., “P4: Programming Protocol-independent Packet Processors,” SIGCOMM Comput. Commun. Rev., vol. 44, no. 3, pp. 87–95, Jul. 2014.
[8] “phaethon/scapy,” GitHub. [Online]. Available: https://github.com/phaethon/scapy. [Accessed: 04-May-2017].
[9] “Tcpreplay.” [Online]. Available: http://tcpreplay.synfin.net/. [Accessed: 04-May-2017].
[10] Antonio Capone, “Tutorial on SDN data plane evolution,” 11:45:24 UTC.
[11] “Welcome to Scapy’s documentation! — Scapy v2.1.1-dev documentation.” [Online]. Available: http://www.secdev.org/projects/scapy/doc/. [Accessed: 04-May-2017].