Implementing a Packet Filter using a P4 Programmable Switch

Andrew Smith

Advisors: Ali Mazloum, Jose Gomez, Jorge Crichigno

Integrated Information Technology Department, University of South Carolina, Columbia, South Carolina

Abstract

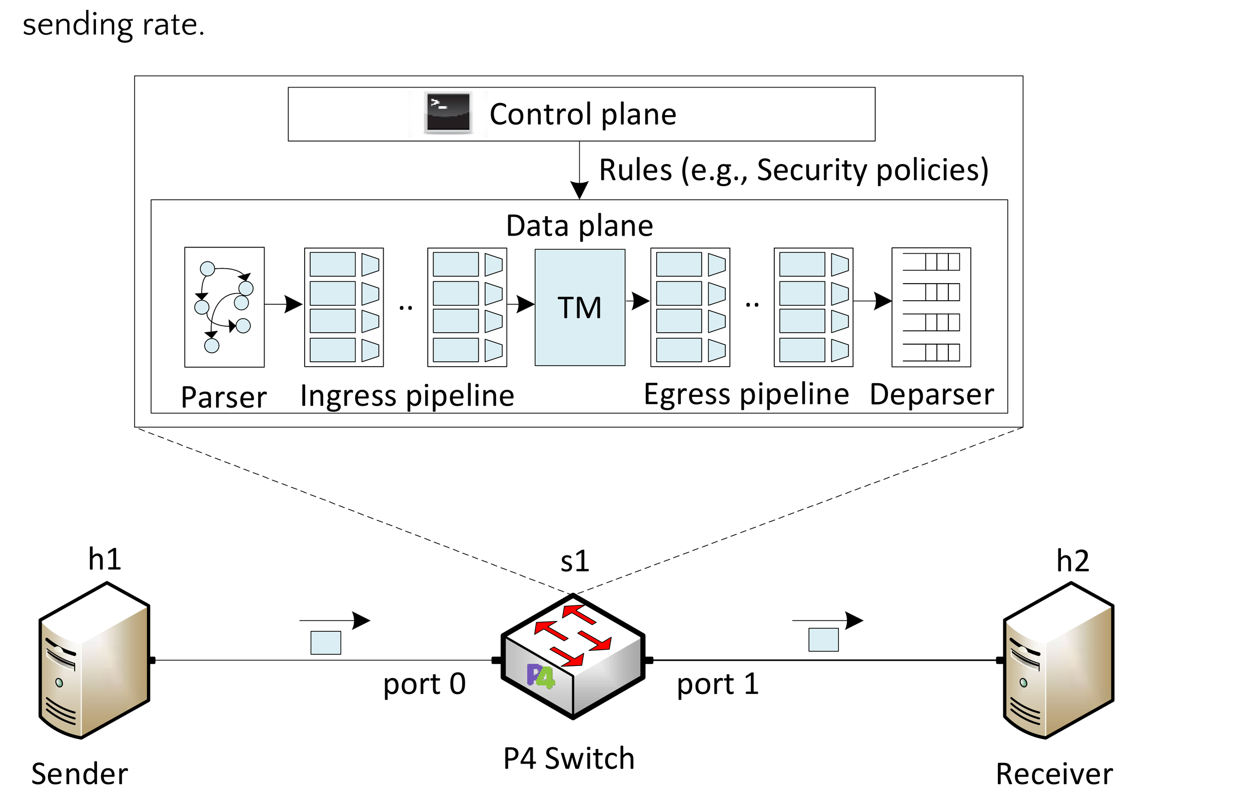
- This project presents a packet filter implemented using a P4 programmable switch.
- P4 is a programming language to describe the behavior of the data plane.
- The data plane is structured as a pipeline that processes a stream of bits.
- With P4, the programmer specifies how the pipeline will manipulate the information contained in packet headers to make decisions.
- In this project, a P4 programmable switch inspects the content of packet headers to decide whether to drop or allow them to pass.
- This decision is based on predefined rules that the network administrator established as security policies.
- The P4 programmable switch will implement a counter to store the number of times a packet is dropped based on these predefined rules.
- Results show that P4 facilitates implementing a packet filter that allows the network administrator to configure security policies.
- Moreover, this project displays that P4 facilitates implementing counters to record statistics of interest at runtime.

Project Description

- A packet filter is a network device that examines each datagram in isolation and determines whether the datagram should be allowed to pass or dropped based on administrator-specific rules.
- Filtering decisions are typically based on:
  - IP source or destination address.
  - Protocol type in IP datagram field: TCP, UDP, ICMP, and others.
  - TCP or UDP source and destination port.
  - TCP flag bits: SYN, ACK, and other flags.
  - ICMP message type.
  - Different rules for datagrams leaving and entering the network.
  - Different rules for the different router interfaces.
- This project aims at implementing a packet filter on a programmable switch using the P4 language, as well as a counter to record the number of disallowed packets.
- The packet filter will enable the network administrator to count and block packets based on physical ingress and/or egress interfaces, IP source or destination address, protocol type in the IP datagram field (TCP, UDP, ICMP), and TCP or UDP source and destination port.

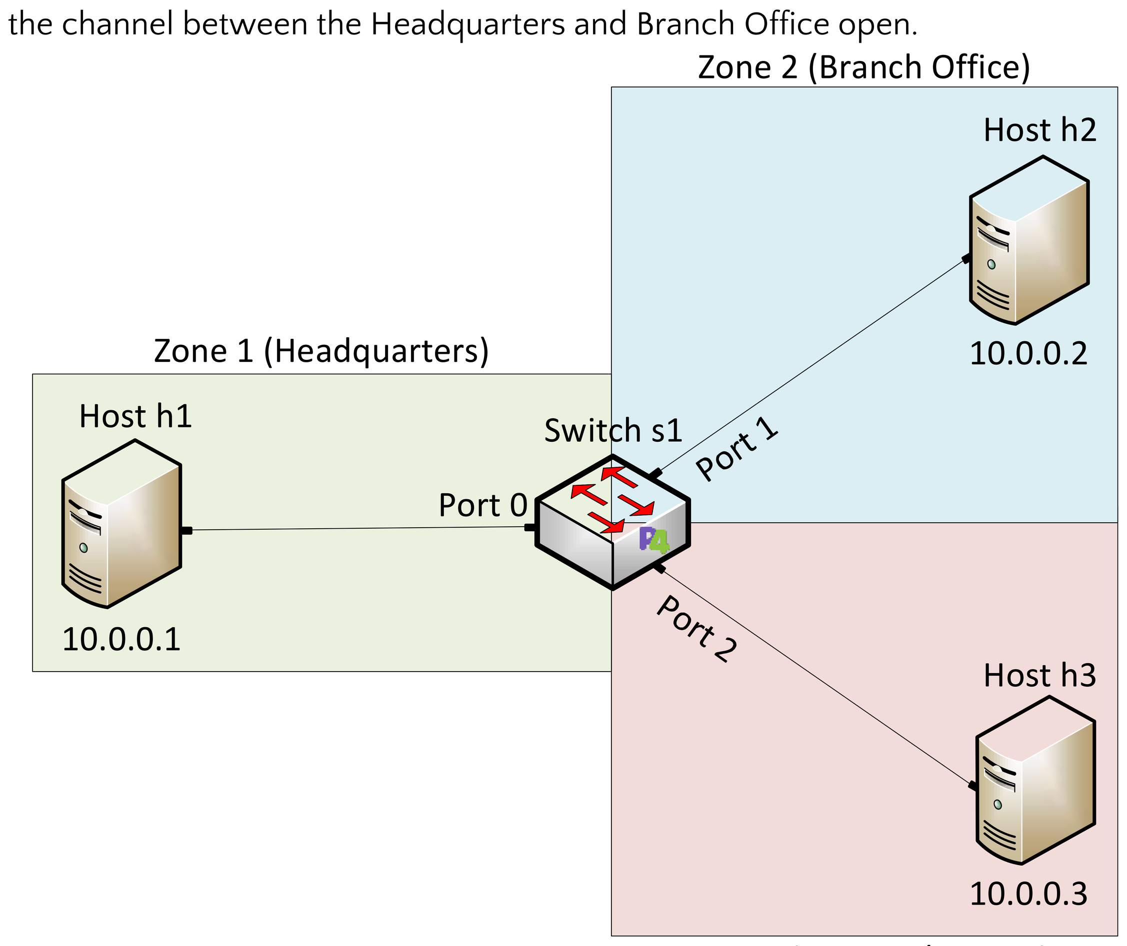
Background on P4 programmable switches

- P4 programmable data planes emerge as a natural evolution of Software-Defined Networking (SDN).
- In the SDN context, the software describes how packets are processed, conceived, tested, and deployed in a much shorter time span by operators, engineers, researchers, and practitioners in general.
- SDN fostered significant advances by separating the switch into two logical components: the control and data planes.
- The control plane implements the switch intelligence, for instance, computing the states of a routing protocol (e.g., BGP, OSPF), running a machine learning algorithm (e.g., classifiers), and processing digests from the data plane.
- The data plane governs the forwarding behavior of a P4 switch by manipulating packets at line rate.
- This project uses the V1 model, a P4 programming model comprising a programmable parser, an ingress pipeline, an egress pipeline, a deparser, and a non-programmable component, the traffic manager (TM).
- The parser extracts the information from packet headers so that the other following stages can make decisions.
- The ingress and egress pipelines execute actions with match-action tables.
- Examples of actions in the data plane can be modifying the destination IP address and decrementing the time-to-live (TTL) field in the IPv4 header.
- The deparser reassembles and emits the packet processed by the previous stages.
- The traffic manager handles operations related to the switch's queue and the sending rate.



Test System

- This project implements a packet filter and direct counter using the behavioral model version 2 (BMv2) software switch that implements the V1 model.
- The topology comprises three hosts and a P4 switch that acts as the packet filter.
- Host h1 represents a device in a company's headquarters (Zone 3), host h2 is a device in a branch office (Zone 2), and host h3 represents a device that is not managed by the company (Zone 3).
- Packets going from host h1 to host h2 and vice versa are subject to different security policies than packets going to host h3.
- Switch s1 leverages match-action tables to drop and count packets based on the physical ingress port and the protocol in the IP datagram field(e.g., TCP, UDP, ICMP).
- The P4 program implemented in switch S1 blocks traffic from port 2 and leaves the channel between the Headquarters and Branch Office open.



Experimentation

- For simplicity, full connectivity will be allowed by default using the *forwarding* match-action table:

Table name: forwarding.

Rule #	Key (Dst. IP)	Action	Action data (egress port)
1	10.0.0.1	forward	0
2	10.0.0.2	forward	1
3	10.0.0.3	forward	2

- First implement a match-action table to simultaneously block and count a packet when called. This is done in the following steps.
  - Establish a direct counter to count on a per-packet basis:

```
direct_counter(CounterType.packets) blocked_counter;
```

  - Create the table *block_counter* and supply it with the above counter and default action of drop:

Table name: block_counter.

Rule #	Key	Action	Action data
default		drop	

  - This table will be called via the Ingress apply logic whenever a packet hits on another specific table. Thus, if a hit occurs on the other table, *block_counter* will iterate, blocking and counting the packet. Otherwise, forwarding will proceed.
- The following scenarios were implemented using match-action tables to test the packet filter:
- Scenario 1: Dropping packets from physical ingress port 2 (Danger).
  - The table *ingress_port_ACL* is populated with the following rules:

Table name: ingress_port_ACL.

Rule #	Key (In. Port)	Action	Action data
1	2	NoAction	

  - Packets that entering through switch port 2 will hit on *ingress_port_ACL*.
- Scenario 2: Dropping packets based on protocol type in IP datagram field.
  - The table *datagram_field_ACL* is populated with the following rules:

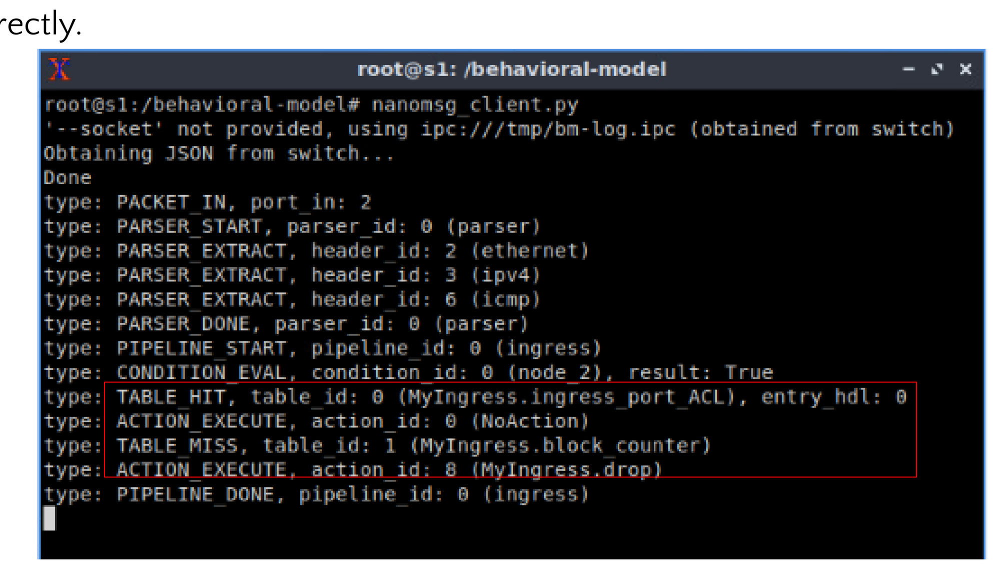
Table name: datagram_field_ACL.

Rule #	Key (protocol)	Action	Action data
1	0x06	NoAction	
2	0x11	NoAction	
3	0x01	NoAction	

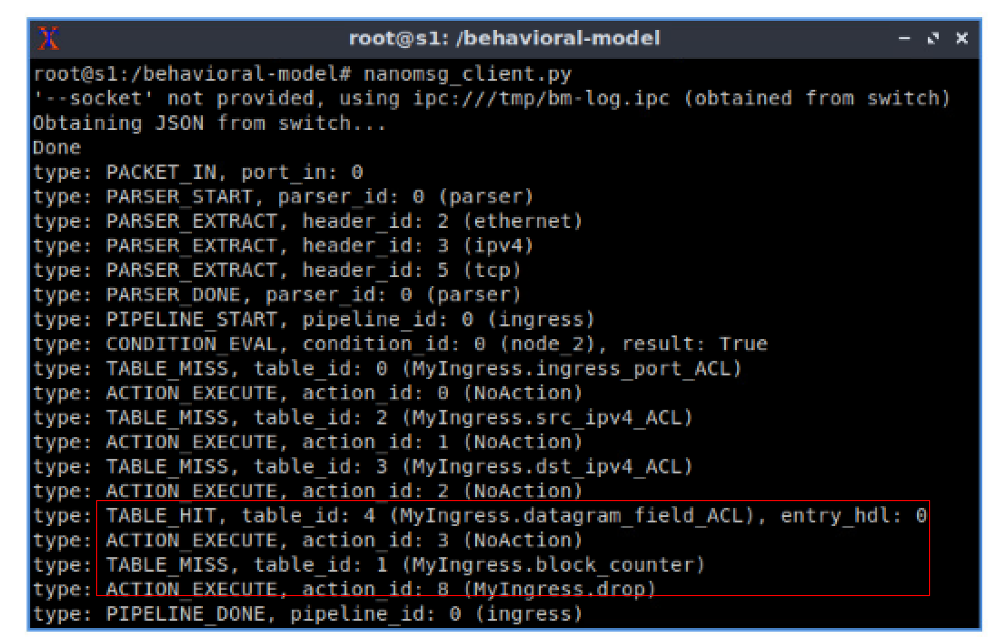
  - The table matches for the protocol numbers, written in hexadecimal. TCP (0x06), UDP (0x11), ICMP (0x01). Thus, all TCP, UDP, and ICMP packets will hit on *datagram_field_ACL*.
- Note: the tables that define the desired packet trait to be dropped (e.g., *ingress_port_ACL* and *datagram_field_ACL*) do not have actions. This is because the apply logic will call the block_counter when a packet hits on either table, allowing the packet to be dropped and counted

Results

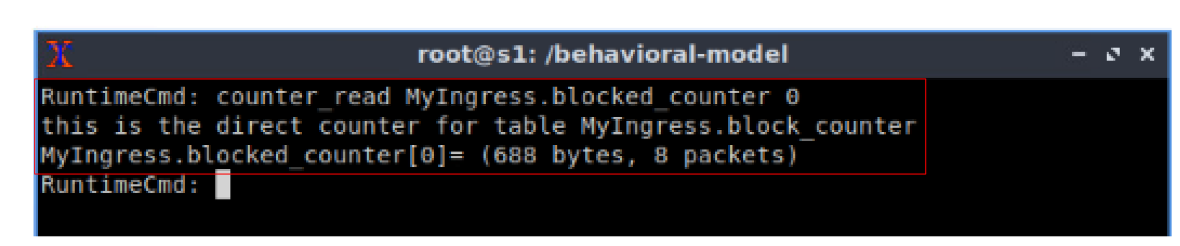
- Results show that packets were successfully filtered.
- The *ping* command was used to verify the first scenario.
- Packets originating from physical port 2 (Danger) were dropped.
- The *nanolog* tool also corroborated that the match-action table was applied correctly.



- In the second scenario, Host h2 was configured with a simple HTTP server, and Host h1 used the *wget* command to try to connect over tcp.
- The *nanolog* tool displayed that request packets using TCP were dropped.



- Finally, the counter was verified using the P4 runtime CLI.
- The output confirmed that 4 ping packets were dropped and counted and 4 *wget* attempts were dropped and counted.



Lessons Learned

- Learned how to implement a packet filter using P4.
- Leveraged match-action tables to implement security policies.
- Implemented a direct counter to record the number of blocked packets.
- Validated the implementation of the security policies in the Netlab environment.
- Understood the flexibility of P4 programmable switches in implementing security features.
- Understood the ability of P4 programmable switches to track and store useful data at line rate.

Conclusion

- This project implemented a packet filter and direct counter using the P4 programming language.
- P4 provides the tools to define how packets are processed in the data plane.
- With P4, the programmer can implement custom security policies.
- Match-action tables are valuable constructs to perform actions on a per-packet basis.
- P4 offers pathways to export important statistics (e.g., counters)
- Future works can include more complex packet processing and statistic gathering using other constructs available in P4.

Acknowledgement

- This work was supported by the Office of Naval Research, under Award N00014-23-1-2245.