Implementing a Packet Filter using a P4 Programmable Switch

Caroline Boozer, Anaia Prather, Camila Pereira

Advisors: Ali Mazloum, Jose Gomez, Jorge Crichigno

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

Abstract

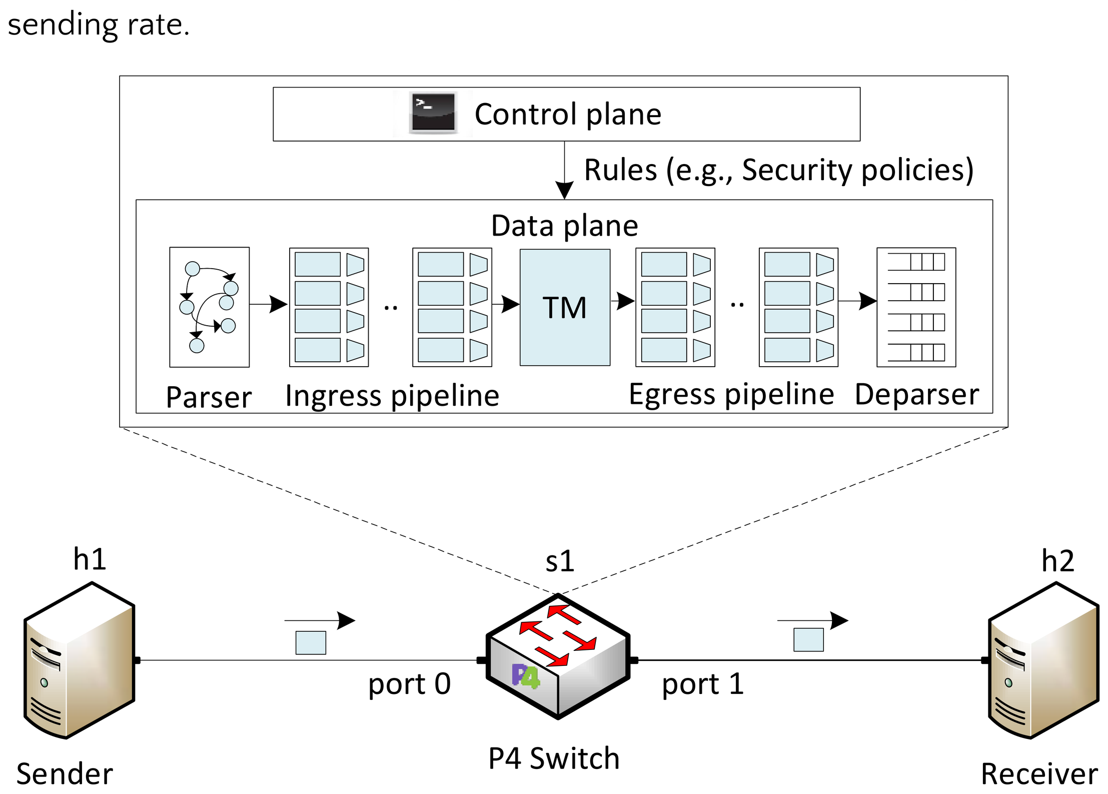
- This project presents a packet filter implemented using a P4 programmable switch.
- P4 is a programming language to describe the behavior of the data plane.
- The data plane is structured as a pipeline that processes a stream of bits.
- With P4, the programmer specifies how the pipeline will manipulate 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.
- Results show that P4 facilitates implementing a packet filter that allows the network administrator to configure security policies.
- Moreover, this project implements the concepts of security zones, which consists of applying different security policies for each switch's interface.

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.
- The packet filter will enable the network administrator to 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.

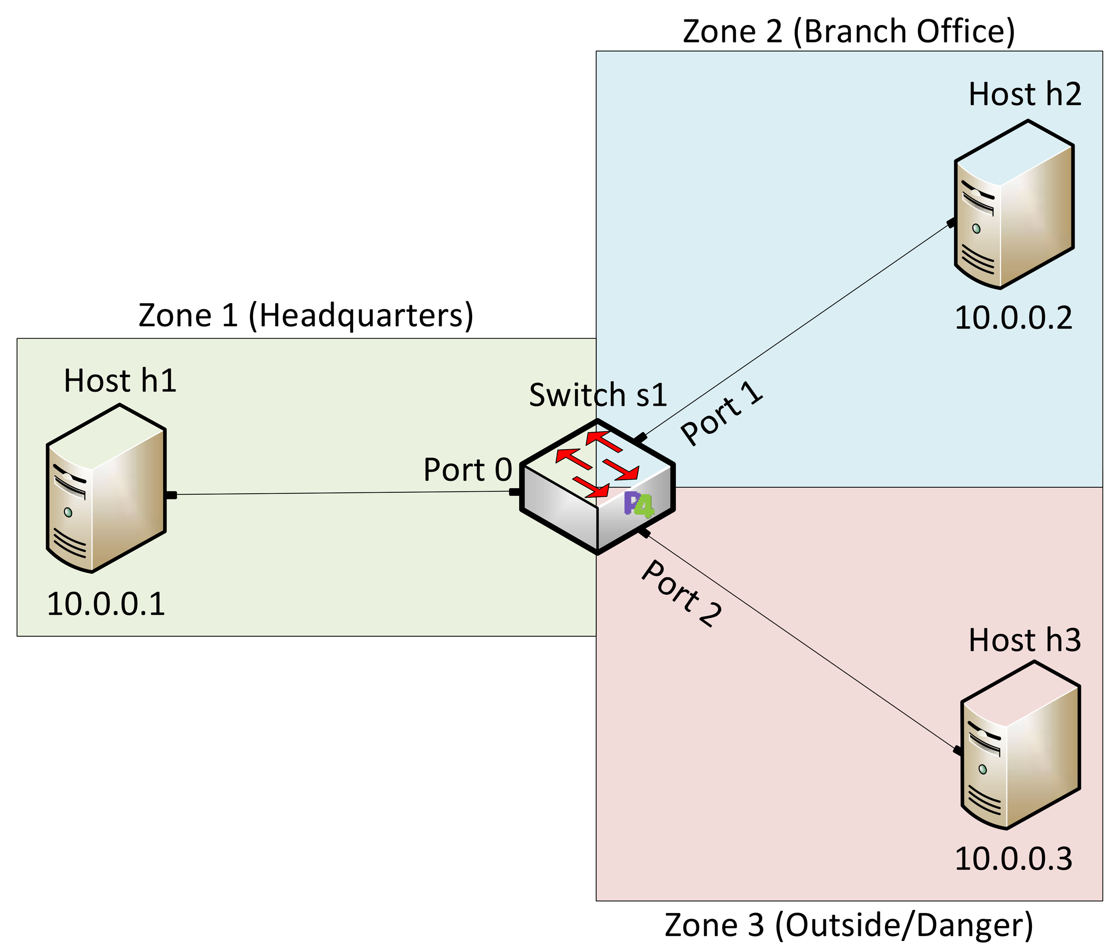
Background on P4 programmable switches

- 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 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 forward or drop packets based on the destination IPv4 address, the destination port, the transport protocol (e.g., TCP, UDP), and ICMP requests.
- The P4 program implemented in switch S1 allows ICMP requests from host h2 but denies those from host h3.



Experimentation

- The following scenarios were implemented using match-action tables to test the packet filter:
- Scenario 1: Filtering packets based on the destination IP address.
  - The table *forwarding* is populated with the following rules:

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	drop	

  - These rules forward packets with destination IP addresses 10.0.0.1 and 10.0.0.2 (rules 1 and 2) but drops packets with destination IP address 10.0.0.3 (i.e., rule 3).
  - Scenario 2: Dropping segments going to the TCP port 80.
    - This scenario requires two match-action tables: *filter_TCP_dstPort* and *forwarding*.
    - The match-action table *filter_TCP_dstPort* drops packets going to port 80, whereas the match-action table *forwarding* forwards packets to their respective destination IP address.

Table name: filter_TCP_dstPort.

Rule #	Key (Dst. Port)	Action	Action data
1	80	drop	

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

  - Scenario 3: Restricting ICMP requests coming from a specific security zone.
    - Two match-action tables implement this filter: *filter_ICMP_protocol* and forwarding.
    - ICMP requests from Zone 3 (Danger) to Zone 1 (Headquarters) are blocked, whereas requests from Zone 2 (Branch Office) to Zone 1 are allowed.

Table name: filter_ICMP_protocol

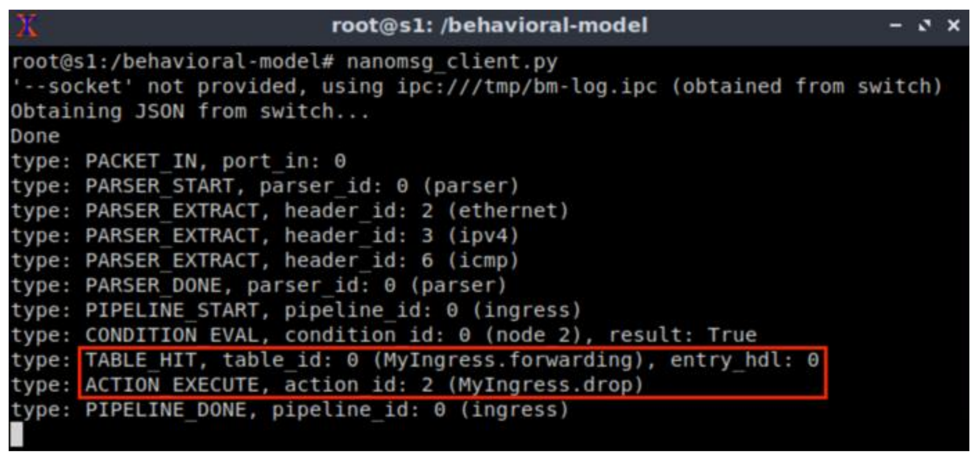
Rule #	Key (ICMP protocol)	Action	Action data (ingress port)
1	8	drop	2

Table name: forwarding.

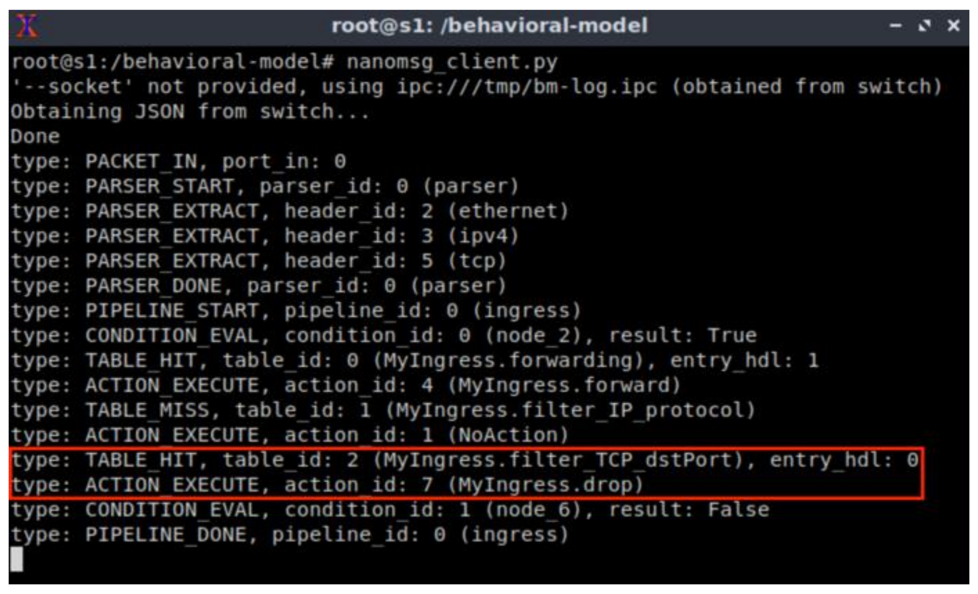
Rule #	Key (Dst. IP)	Action	Action data (egress port)
1	10.0.0.1	forward	0
2	10.0.0.2	forward	1
3	10.0.0.3	forward	2

Results

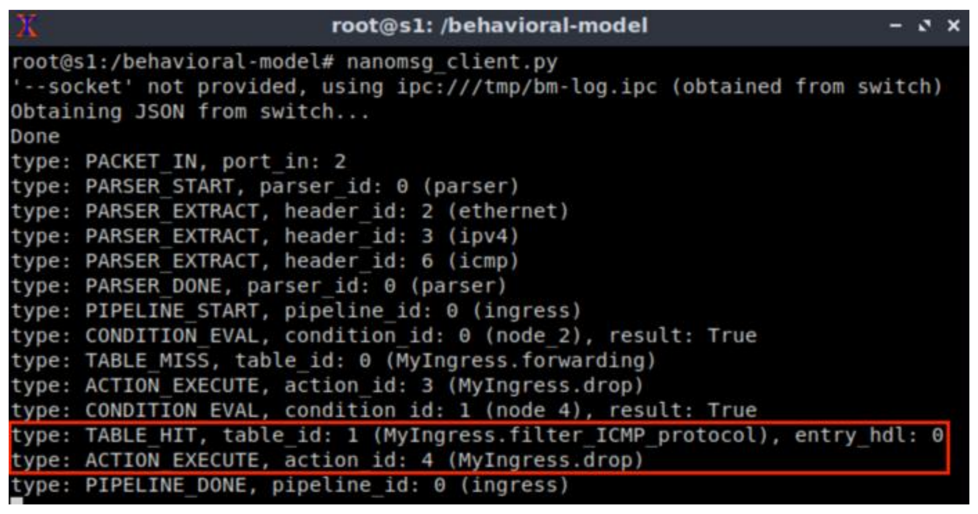
- Results show that packets were successfully filtered.
- The *ping* command was used to verify the first scenario.
- Packets with destination IP address 10.0.0.3 were dropped.
- The *nanolog* tool also corroborated that the match-action table was applied correctly.



- In the second scenario, the sender used the *hping3* tool to create a TCP packet.
- The *nanolog* tool displayed that packets going to port 80 were dropped.



- Finally, the third scenario was tested using the *ping* tool.
- The output confirmed that packets host h3 could not send ICMP requests to host h1.



Lessons Learned

- Learned how to implement a packet filter using P4.
- Leveraged match-action tables to implement security policies.
- Applied the concept of security zones using a P4 switch.
- Validated the implementation of the security policies in the Netlab environment.
- Understood the flexibility of P4 programmable switches in implementing security features.

Conclusion

- This project implemented a packet filter 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.
- Future works can include more complex packet processing using other constructs available in P4.

Acknowledgement

- This work was supported by the Office of Naval Research (ONR), grant N00014-20-1-2797: "Enhancing the Preparation of Next-generation Cyber Professionals"