CSE567-PA02

Building a programmable data plane with P4 & P4Runtime

Submission deadline: 13st March 2022

Total: 16 points

In this assignment, you will gain hands-on experience in writing custom programs for the data plane switches using the P4 language. You will build a custom virtual network using mininet that provides a P4-based software switch, bmv2, which is based on the v1model's simple_switch_grpc target. The bmv2 target switch can be managed statically using the thrift API. The bmv2 switch can also be dynamically managed via the control plane using the P4Runtime API.

In the next few sections, you will set up your machine with the VM image, test few tutorial examples with template code (you write the remainder code), and develop your custom application.

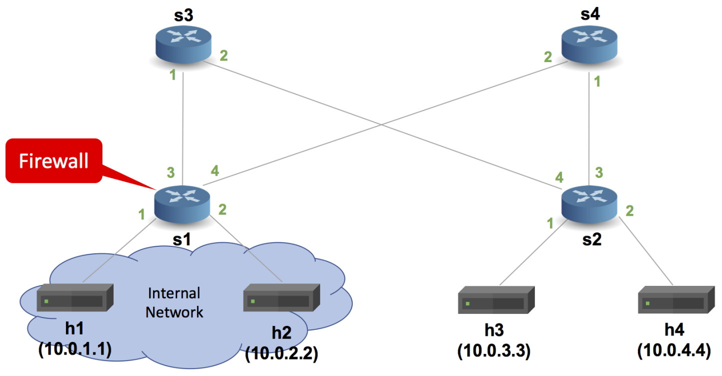
Setting up your machine

1. Install virtual box
    - Download and install the appropriate executable for Windows / Ubuntu and follow the instructions
2. Download and setup the VM from the following link. This VM contains mininet, bmv2 switch tools (P4, P4Runtime), and other necessary tools such as wireshark.

Problem statement

The tutorial exercise applications are already available in the VM in the "tutorials" folder and are publicly available here. You have already understood writing basic programs ("basic" and "basic_tunnel") in the class. These programs used table data structure and for every incoming packet, the switch performed table match-action processing. We will now look at use cases that comprise of: (1) switch stores application state, (2) switch/application state is tagged to the packet, (3) packet copies are created within the switch, and (4) the switch data structures are dynamically configured.

Q.1. Implement a stateful firewall on switch s1 (shown in figure) that functions as follows.



- Hosts h1 and h2 are on the internal network and can always connect to one another.
- Hosts h1 and h2 can freely connect to h3 and h4 on the external network.

- Hosts h3 and h4 can only reply to connections once they have been established from either h1 or h2, but cannot initiate new connections to hosts on the internal network.

Read the complete problem description here.

a) Explain the concept of bloom filter and how is it relevant to stateful firewall. **(1 point)**
b) Complete the to-do tasks in the given P4 program. Test your solution with traffic between h1 and h3 (iperf) and show the output. **(1 point)**
c) This stateful firewall is implemented 100% in the dataplane using a simple bloom filter. A bloom filter is used for testing membership, i.e., test if a particular key exists, but can have false positives. Thus there is some probability that would let unwanted flows to pass through. What is an alternative solution to provide 100% assurance that unwanted flows are blocked? Does the alternate solution have any limitation? **(1 point)**

Q.2. Implement a P4 program within the data plane to monitor that enables a host to monitor the utilization of all links in the network. The host generates a source routed probe packet with probe headers for each intermediate switch. You have to program each switch to monitor the following link statistics in the probe headers.

- Total number of bytes sent from the egress port
- The inter-packet interval

Read the complete problem description here.

a) Implement the link monitoring logic. Spend decent amount of time in understanding the parser. The measured link utilizations should agree with what iperf reports; explain and add snapshots to the report. **(1 point)**
b) In this program register arrays are used to store byte counts for egress ports. Can we use P4 counters instead? Why or why not? **(1 point)**
c) Is there an alternative solution to solve this problem? Explain. **(1 point)**

Q.3. Implement a P4 program to multicast incoming packets at layer 2, i.e., send the copy of the packet to all the MAC addresses that belong to the same multicast group. You will learn how to create multicast groups via the control plane.

Read the complete problem description here.

a) Implement the multicast logic. Report the snapshots with **(1 point)**
   i) h1, h2, h3 as part of the multicast group
   ii) h1, h2, h3, h4 as part of the multicast group
b) Update the solution to work with two different multicast groups **(2 points)**
   i) h1, h2, h3, h4 belong to multicast group 1
   ii) h5, h6, h7, h8 belong to multicast group 2 (You will have to update the topology to have 8 hosts connected to the switch)
   Report the working along with snapshots.

Q.4. Implement a P4 program for load balancing based on a simple version of Equal-Cost Multipath Forwarding (ECMP). ECMP load balancing distributes traffic more evenly by installing entries for multiple best paths to the switch's forwarding layer and using load balancing algorithm to identify flows and distribute them to different paths. If number of flows is large this algorithm helps in distributing the traffic along with other unused paths in order to reduce the congestion. ECMP implementation enables us to evenly spread flows across the network leading to best utilization of multiple links towards the destination.

- The switch you will implement will use two tables to forward packets to one of two destination hosts at random.
    - The first table will use a hash function (applied to a 5-tuple consisting of the source and destination IP addresses, IP protocol, and source and destination TCP ports) to select one of two hosts.
    - The second table will use the computed hash value to forward the packet to the selected host.

Read the complete problem description here.

a) Complete the load balancing code. Send several messages to demonstrate that load balancing is working; report it as a snapshot **(1 point)**

b) The current topology has two routes from h1 to h2/h3 and two servers (h2 and h3). Considering adding one more path, i.e., switch s4 connected to switch s1 and h4 so we have three servers now, h2, h3, h4.
   a) What all changes are required in the existing code? **(1 point)**
   b) Implement these changes and demonstrate load balancing between the three servers. **(1 point)**

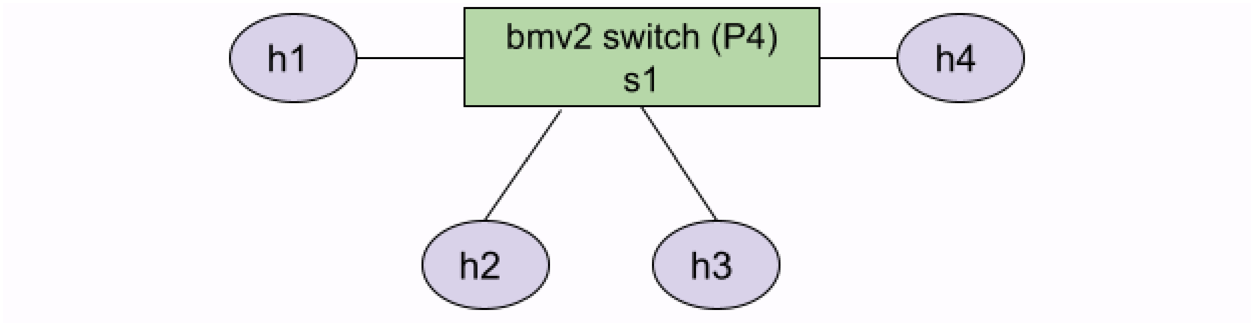
Q.5. This program template is NOT available in the tutorial exercise. Implement a P4 program that sends an telemetry packet to the controller node "h1" as follows.

a) Build the custom network topology as shown in the figure. **(1 point)**

b) Our telemetry packet is a typical UDP packet with payload containing **(1 point)**
   a) the source port byte count and
   b) the port id.

c) Generate the telemetry packet after every 10th incoming packet at each switch port, i.e., the 4 ports should be monitored individually and a telemetry packet for port "x" is generated after every 10th incoming packet at port "x". Send the egress port byte count for every 10th packet to the host h1. **(1 point)**

d) Consider sending simple UDP packets and demonstrate the solution. You will have to use send and receive python programs **(1 point)**



What to submit:
Submit the following documents in one folder. The naming convention for the folder is
<roll-no>PA01CSE567

1. A short write-up with snapshots followed by an explanation of your observations (if any)
2. Include the code for each question.