Welcome to P4 Programming tutorial session

Introduction of the lab environment(Jupyterhub)

Each student is given a credential to login to the Jupyterhub (https://training.oftein.fsktm.um.edu.my) When you login, a jupyter notebook docker container will spawned for you.

Lab content

For the first part, we will be looking at what a P4 program. Next, we will have a hands on lab on how to setup network switches, and use our P4 program as the data plane program.

1. P4 program overview
1. Tutorial for P4 switches Mininet

P4 program overview

Introduction

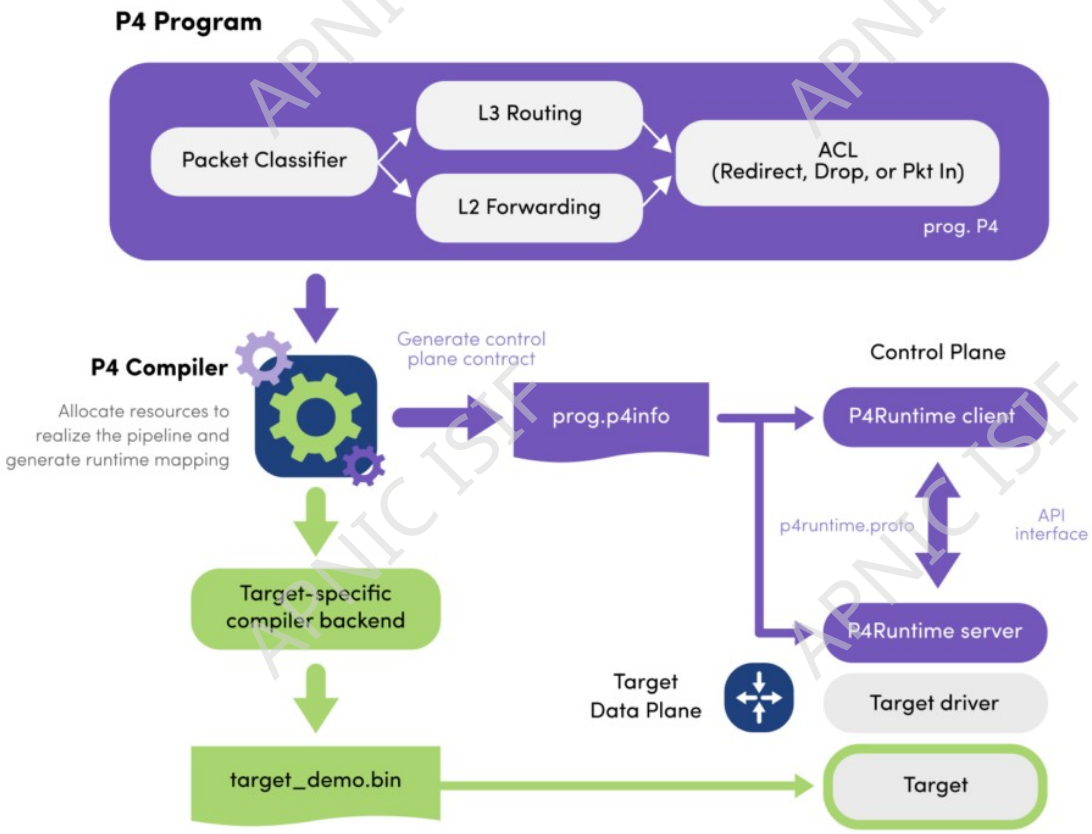
P4 program = target-independent code, you write a P4 program in P4 language.

P4 compiler = compile your code into target-dependent files, and a P4Runtime Info file(.p4info)

Hence, with P4 Compiler, P4 can run in any devices(a software switch, tofino, etc) that is supported.

P4Runtime = communication protocol between control plane and data plane; switch(P4Runtime server), controller(P4Runtime client)

P4Runtime Info = with this file, controller able to access/insert/program the data plane(the switch) tables and update/delete the entries.



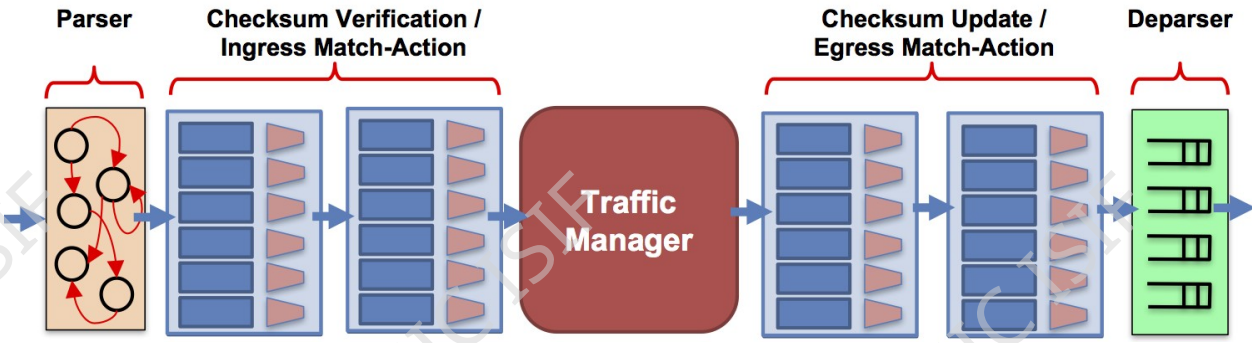
P4 Program

P4 Architecture

A programming model, in general provide a logical view of the processing pipeline.

The **V1Model** is the architecture commonly used with the BMv2 Simple Switch. It defines the programmable structure of the pipeline. There is a direct relationship between the blocks.

V1Model architecture



Example of P4 Program

```
/* -*- P4_16 -*- */
#include <core.p4>
#include <v1model.p4>

const bit<16> TYPE_IPV4 = 0x800;

/*************************************************************************
****
********************** H E A D E R S
*************************************
*************************************************************************
***/

typedef bit<9>  egressSpec_t;
typedef bit<48> macAddr_t;
typedef bit<32> ip4Addr_t;

header ethernet_t {
    macAddr_t dstAddr;
    macAddr_t srcAddr;
    bit<16>   etherType;
}

header ipv4_t {
    bit<4>    version;
    bit<4>    ihl;
    bit<8>    diffserv;
```

```
    bit<16>   totalLen;
    bit<16>   identification;
    bit<3>    flags;
    bit<13>   fragOffset;
    bit<8>    ttl;
    bit<8>    protocol;
    bit<16>   hdrChecksum;
    ip4Addr_t srcAddr;
    ip4Addr_t dstAddr;
}

struct metadata {
    /* empty */
}

struct headers {
    ethernet_t   ethernet;
    ipv4_t       ipv4;
}

/*************************************************************************
****
*********************** P A R S E R
*************************************
*************************************************************************
***/

parser MyParser(packet_in packet,
                out headers hdr,
                inout metadata meta,
                inout standard_metadata_t standard_metadata) {

    state start {
        transition parse_ethernet;
    }

    state parse_ethernet {
        packet.extract(hdr.ethernet);
        transition select(hdr.ethernet.etherType) {
            TYPE_IPV4: parse_ipv4;
            default: accept;
        }
    }

    state parse_ipv4 {
        packet.extract(hdr.ipv4);
        transition accept;
    }

}
```

```
/*************************************************************************
****
************   C H E C K S U M   V E R I F I C A T I O N
*************
*************************************************************************
***/

control MyVerifyChecksum(inout headers hdr, inout metadata meta) {
    apply {  }
}


/*************************************************************************
****
*************  I N G R E S S   P R O C E S S I N G
*******************
*************************************************************************
***/

control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t standard_metadata) {
    action drop() {
        mark_to_drop(standard_metadata);
    }

    action ipv4_forward(macAddr_t dstAddr, egressSpec_t port) {
        standard_metadata.egress_spec = port;
        hdr.ethernet.srcAddr = hdr.ethernet.dstAddr;
        hdr.ethernet.dstAddr = dstAddr;
        hdr.ipv4.ttl = hdr.ipv4.ttl - 1;
    }

    table ipv4_lpm {
        key = {
            hdr.ipv4.dstAddr: lpm;
        }
        actions = {
            ipv4_forward;
            drop;
            NoAction;
        }
        size = 1024;
        default_action = drop();
    }

    apply {
        if (hdr.ipv4.isValid()) {
            ipv4_lpm.apply();
```

```
        }
    }
}

/*************************************************************************
****
***************  E G R E S S   P R O C E S S I N G
*******************
*************************************************************************
***/

control MyEgress(inout headers hdr,
                 inout metadata meta,
                 inout standard_metadata_t standard_metadata) {
    apply {  }
}

/*************************************************************************
****
*************   C H E C K S U M    C O M P U T A T I O N
**************
*************************************************************************
***/

control MyComputeChecksum(inout headers  hdr, inout metadata meta) {
     apply {
        update_checksum(
        hdr.ipv4.isValid(),
            { hdr.ipv4.version,
              hdr.ipv4.ihl,
              hdr.ipv4.diffserv,
              hdr.ipv4.totalLen,
              hdr.ipv4.identification,
              hdr.ipv4.flags,
              hdr.ipv4.fragOffset,
              hdr.ipv4.ttl,
              hdr.ipv4.protocol,
              hdr.ipv4.srcAddr,
              hdr.ipv4.dstAddr },
            hdr.ipv4.hdrChecksum,
            HashAlgorithm.csum16);
    }
}

/*************************************************************************
****
***********************  D E P A R S E R
*******************************
*************************************************************************
***/
```

```
control MyDeparser(packet_out packet, in headers hdr) {
    apply {
        packet.emit(hdr.ethernet);
        packet.emit(hdr.ipv4);
    }
}

/*************************************************************************
****
***********************  S W I T C H  ********************************
*************************************************************************
***/

V1Switch(
MyParser(),
MyVerifyChecksum(),
MyIngress(),
MyEgress(),
MyComputeChecksum(),
MyDeparser()
) main;
```

Tutorial for P4 switches Mininet

What is Mininet

Mininet is a network emulator, or perhaps more precisely a network emulation orchestration system. It runs a collection of end-hosts, switches, routers, and links on a single Linux kernel. It uses lightweight virtualization to make a single system look like a complete network, running the same kernel, system, and user code. A Mininet host behaves just like a real machine; you can ssh into it (if you start up sshd and bridge the network to your host) and run arbitrary programs (including anything that is installed on the underlying Linux system.) The programs you run can send packets through what seems like a real Ethernet interface, with a given link speed and delay. Packets get processed by what looks like a real Ethernet switch, router, or middlebox, with a given amount of queueing. When two programs, like an iperf client and server, communicate through Mininet, the measured performance should match that of two (slower) native machines.

In short, Mininet's virtual hosts, switches, links, and controllers are the real thing – they are just created using software rather than hardware – and for the most part their behavior is similar to discrete hardware elements. It is usually possible to create a Mininet network that resembles a hardware network, or a hardware network that resembles a Mininet network, and to run the same binary code and applications on either platform.

What is BMV2

This is the second version of the reference P4 software switch, nicknamed bmv2 (for behavioral model version 2). The software switch is written in C++17. It takes as input a JSON file generated from your P4 program by a P4 compiler and interprets it to implement the packet-processing behavior specified by that P4 program.

bmv2 is not meant to be a production-grade software switch. It is meant to be used as a tool for developing, testing and debugging P4 data planes and control plane software written for them. As such, the performance of bmv2 - in terms of throughput and latency - is significantly less than that of a production-grade software switch like Open vSwitch. For more information about the performance of bmv2, refer to this document.

Lab 1 : Creating mininet topology

In this section, we will be creating a **linear topology with 2 switches and 2 hosts** with mininet

1.1 Create a topology in mininet

```
sudo mn --custom new_bmv2.py --switch simple_switch_grpc --host onoshost --controller none --topo linear,2
```

Sample Output

```
*** Cleanup complete.
(base) jovyan@eac5f8113fdb:~/work$ sudo mn --custom new_bmv2.py --switch simple_switch_grpc --host onoshost --controller none --topo linear,2
*** Error setting resource limits. Mininet's performance may be affected.
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s2) (s2, s1)
*** Configuring hosts
h1 h2
*** Starting controller

*** Starting 2 switches
s1 .. ⚡simple_switch_grpc @ 50001
s2 .. ⚡simple_switch_grpc @ 50002

*** Starting CLI:
mininet>
```

1.2 Display the listening port to verify that the switch spin up is listening on the port

```
netstat -tulpn
```

Sample Output

```
(base) jovyan@eac5f8113fdb:~/work$ netstat -tulpn | grep 5000
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp6       0      0 :::50001                :::*                    LISTEN      -
tcp6       0      0 :::50002                :::*                    LISTEN      -
(base) jovyan@eac5f8113fdb:~/work$
```

1.3 Check the running process

```
ps aux | grep simple_switch_grpc
```

Sample Output

```
(base) jovyan@eac5f8113fdb:~/work$ ps aux | grep simple_switch_grpc
root        2951  0.0  0.0  10680  5176 pts/0    S+   07:01   0:00 sudo mn --custom new_bmv2.py --switch simple_switch_grpc --host onoshost --controller none --topo linear,2
root        2952  0.0  0.0  10680   820 pts/1    Ss   07:01   0:00 sudo mn --custom new_bmv2.py --switch simple_switch_grpc --host onoshost --controller none --topo linear,2
root        2953  0.2  0.0 172408 18912 pts/1    Sl+  07:01   0:00 /opt/conda/bin/python /opt/conda/bin/mn --custom new_bmv2.py --switch simple_switch_grpc --host onoshost --controlle
root        2997  0.1  0.0 335028 27472 ?        Ssl  07:01   0:00 simple_switch_grpc --device-id 1 -i 1@s1-eth1 -i 2@s1-eth2 --log-console -Lwarn --no-p4 -- --cpu-port 255 --grpc-ser
root        3004  0.1  0.0 335028 27296 ?        Ssl  07:01   0:00 simple_switch_grpc --device-id 1 -i 1@s2-eth1 -i 2@s2-eth2 --log-console -Lwarn --no-p4 -- --cpu-port 255 --grpc-ser
jovyan      3130  0.0  0.0   6608  2344 pts/7    S+   07:02   0:00 grep --color=auto simple_switch_grpc
(base) jovyan@eac5f8113fdb:~/work$
```

1.4 Query mininet information

From mininet terminal(mininet>), query the host mac address

```
Mininet> h1 ifconfig | grep ether
Mininet> h2 ifconfig | grep ether
```

Add static arp entry for the hosts

```
Mininet> h1 arp -s 10.0.0.2 xx:xx:xx:xx:xx:xx
Mininet> h2 arp -s 10.0.0.1 yy:yy:yy:yy:yy:yy
```

Query the links information

```
Mininet> links
```

Sample output

```
mininet> links
h1-eth0<->s1-eth1 (OK OK)
h2-eth0<->s2-eth1 (OK OK)
s2-eth2<->s1-eth2 (OK OK)
mininet> h1 ifconfig | grep ether
        ether e6:fa:21:8e:96:7a  txqueuelen 1000  (Ethernet)
mininet> h2 ifconfig | grep ether
        ether c6:3d:bc:84:f2:75  txqueuelen 1000  (Ethernet)
mininet> h1 arp -s 10.0.0.2 c6:3d:bc:84:f2:75
mininet> h2 arp -s 10.0.0.1 e6:fa:21:8e:96:7a
```

1.5 Using P4RuntimeShell, connect to the Switch

Open a new terminal, connect to Switch1

```
connect --grpc-addr 127.0.0.1:50001 --device-id 1 --election-id 0,1 --config p4info.txt,basic.json
```

Open a new terminal, connect to Switch2

```
connect --grpc-addr 127.0.0.1:50002 --device-id 1 --election-id 0,1 --config p4info.txt,basic.json
```

Sample Output

```
(base) jovyan@9c51efe019eb:~/work$ connect --grpc-addr 127.0.0.1:50002 --device-id 1 --election-id 0,1 --config p4info.txt,basic.json
*** Welcome to the IPython shell for P4Runtime ***
P4Runtime sh >>>

P4Runtime sh >>>
```

1.6 Insert table entry to BMV2 Switches

From Switch1 P4RuntimeShell

```
te=table_entry["MyIngress.ipv4_lpm"](action="MyIngress.ipv4_forward")
te.match["hdr.ipv4.dstAddr"] = "10.0.0.2/32"
te.action["port"] = "2"
```

```
te.action["dstAddr"] = "xx:xx:xx:xx:xx:xx"
te.insert()
te=table_entry["MyIngress.ipv4_lpm"](action="MyIngress.ipv4_forward")
te.match["hdr.ipv4.dstAddr"] = "10.0.0.1/32"
te.action["port"] = "1"
te.action["dstAddr"] = "yy:yy:yy:yy:yy:yy"
te.insert()
```

From Switch2 P4RuntimeShell:

```
te=table_entry["MyIngress.ipv4_lpm"](action="MyIngress.ipv4_forward")
te.match["hdr.ipv4.dstAddr"] = "10.0.0.2/32"
te.action["port"] = "1"
te.action["dstAddr"] = "xx:xx:xx:xx:xx:xx"
te.insert()
te=table_entry["MyIngress.ipv4_lpm"](action="MyIngress.ipv4_forward")
te.match["hdr.ipv4.dstAddr"] = "10.0.0.1/32"
te.action["port"] = "2"
te.action["dstAddr"] = "yy:yy:yy:yy:yy:yy"
te.insert()
```

Sample Output

```
(base) jovyan@9c51efe019eb:~/work$ connect --grpc-addr 127.0.0.1:50002 --device-id 1 --election-id 0,1 --config p4info.txt,basic.json
*** Welcome to the IPython shell for P4Runtime ***
P4Runtime sh >>> te=table_entry["MyIngress.ipv4_lpm"](action="MyIngress.ipv4_forward")
           ...: te.match["hdr.ipv4.dstAddr"] = "10.0.0.2/32"
           ...: te.action["port"] = "1"
           ...: te.action["dstAddr"] = "c6:3d:bc:84:f2:75"
           ...: te.insert()
           ...: te=table_entry["MyIngress.ipv4_lpm"](action="MyIngress.ipv4_forward")
           ...: te.match["hdr.ipv4.dstAddr"] = "10.0.0.1/32"
           ...: te.action["port"] = "2"
           ...: te.action["dstAddr"] = "e6:fa:21:8e:96:7a"
           ...: te.insert()
field_id: 1
lpm {
  value: "\n\000\000\002"
  prefix_len: 32
}

param_id: 2
value: "\001"

param_id: 1
value: "\306=\274\204\362u"

field_id: 1
lpm {
  value: "\n\000\000\001"
  prefix_len: 32
}

param_id: 2
value: "\002"

param_id: 1
value: "\346\372!\216\226z"
```

1.7 Verifying ping between hosts

From mininet> terminal, perform ping test

```
Mininet> h1 ping h2
```

Sample Output

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=62 time=1.85 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=62 time=1.62 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=62 time=1.72 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=62 time=1.68 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=62 time=1.52 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4008ms
rtt min/avg/max/mdev = 1.517/1.676/1.852/0.111 ms
```