Runtime Controller, Checksum Calculation, Deparser

Jorge Crichigno[1], Mariam Kiran[2]
[1]University of South Carolina, [2]ESnet

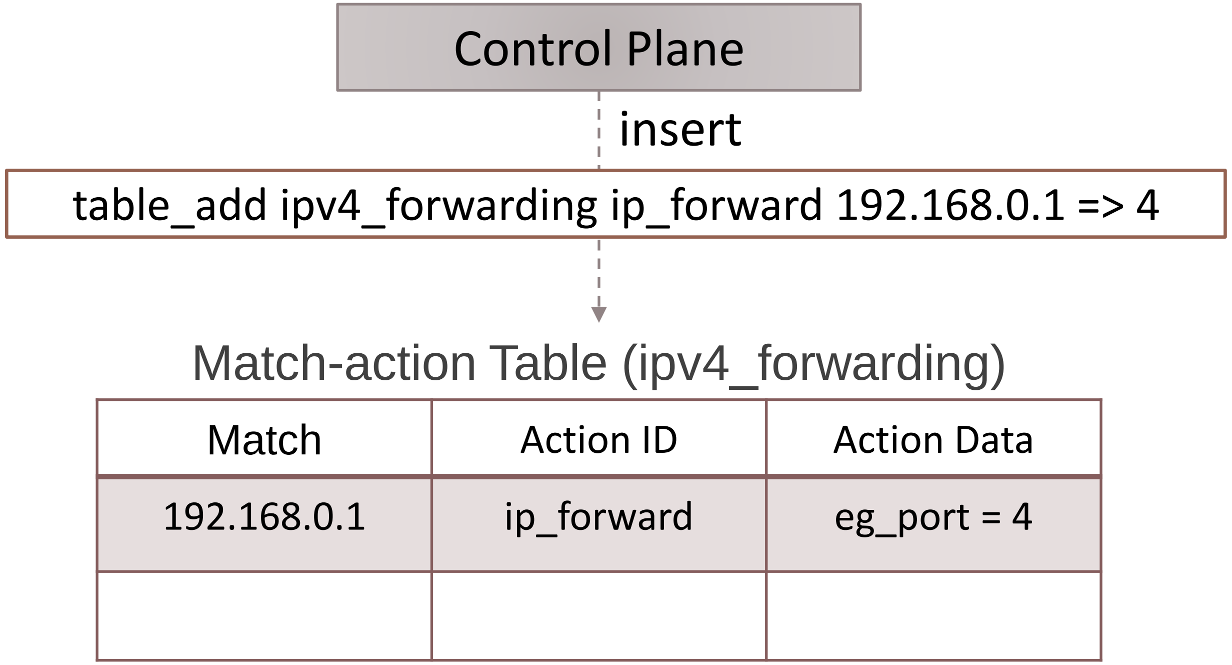
Lab Assistants: Elie Kfoury, Ali AlSabeh, Jose Gomez
University of South Carolina

WASTC 2022 virtual Faculty Development Weeks (vFDW)
June 15, 2022

Runtime Controller

Control Plane

- The match-action tables are empty by default
- The control plane populates the tables with entries
- The control plane can insert, remove, and update table entries



Control Plane

insert

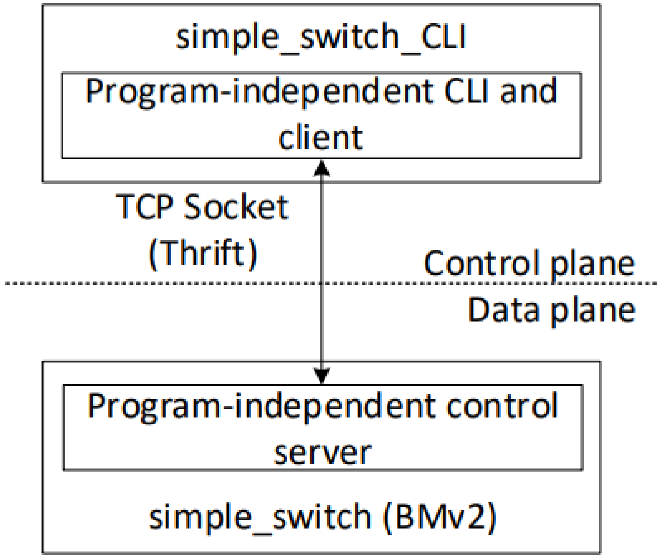
```
table_add ipv4_forwarding ip_forward 192.168.0.1 => 4
```

Match-action Table (ipv4_forwarding)

Match	Action ID	Action Data
192.168.0.1	ip_forward	eg_port = 4

Runtime Environment

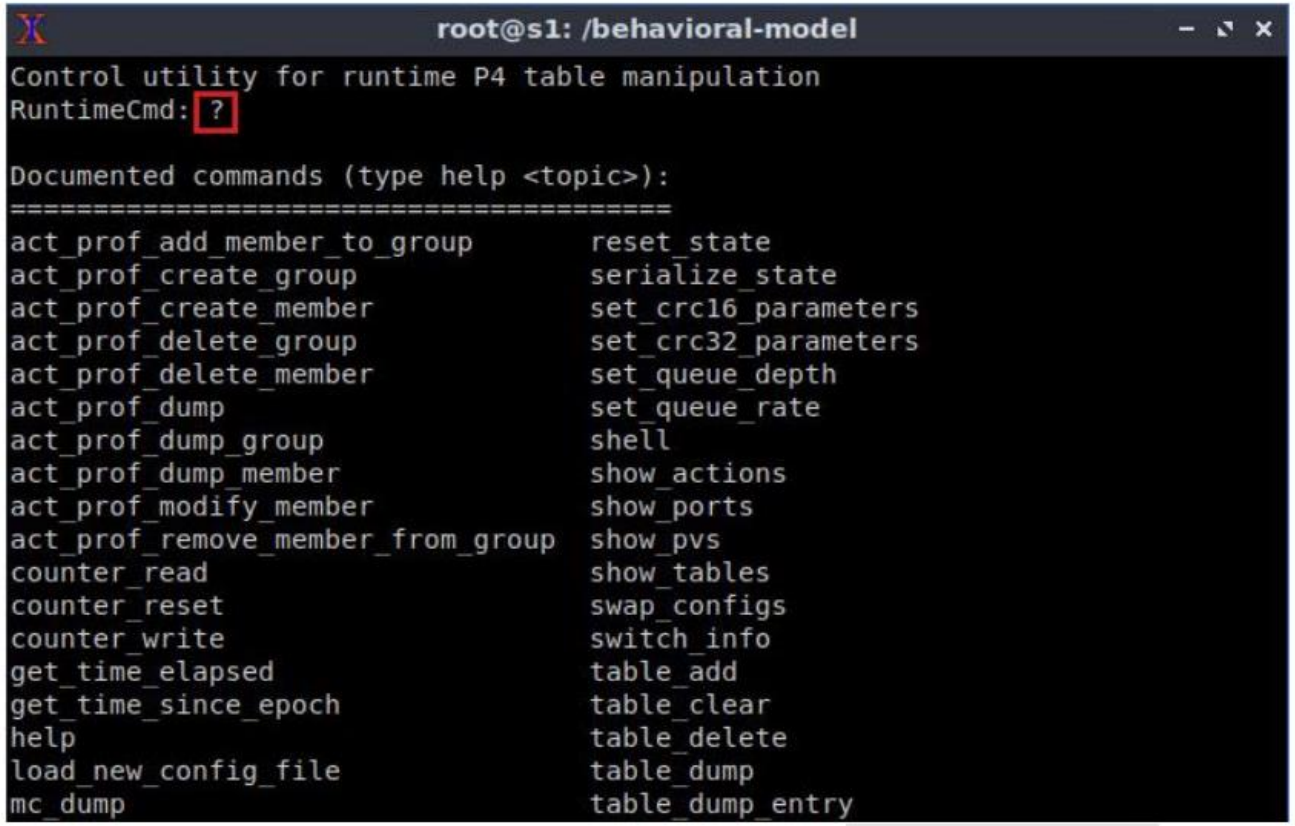
- The simple_switch_CLI tool is used to populate the tables in this lab series
- This tool includes a program-independent CLI and a Thrift[1] client
- It connects to a Thrift control server residing on the switch



[1] Thrift is an interface definition language and binary communication protocol used for defining and creating services

Runtime Environment

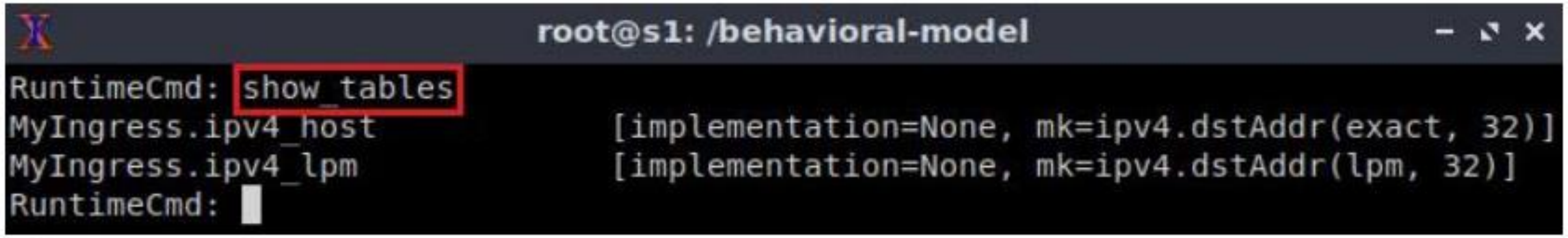
- The simple_switch_CLI is similar to other CLIs (e.g., Cisco IOS CLI) and offers a variety of commands



```
Control utility for runtime P4 table manipulation
RuntimeCmd: ?

Documented commands (type help <topic>):
========================================
act_prof_add_member_to_group       reset_state
act_prof_create_group              serialize_state
act_prof_create_member             set_crc16_parameters
act_prof_delete_group              set_crc32_parameters
act_prof_delete_member             set_queue_depth
act_prof_dump                      set_queue_rate
act_prof_dump_group                shell
act_prof_dump_member               show_actions
act_prof_modify_member             show_ports
act_prof_remove_member_from_group  show_pvs
counter_read                       show_tables
counter_reset                      swap_configs
counter_write                      switch_info
get_time_elapsed                   table_add
get_time_since_epoch               table_clear
help                               table_delete
load_new_config_file               table_dump
mc_dump                            table_dump_entry
```

Runtime Environment

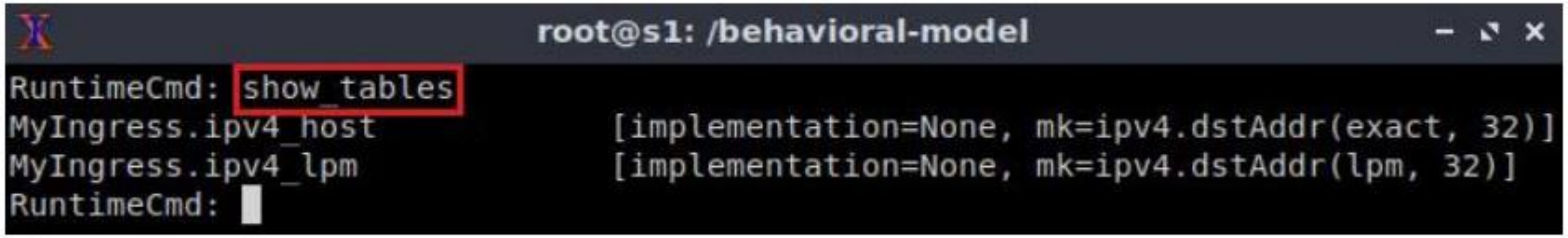
- The simple_switch_CLI is similar to other CLIs (e.g., Cisco IOS CLI) and offers a variety of commands



```
RuntimeCmd: show_tables
MyIngress.ipv4_host            [implementation=None, mk=ipv4.dstAddr(exact, 32)]
MyIngress.ipv4_lpm             [implementation=None, mk=ipv4.dstAddr(lpm, 32)]
RuntimeCmd: 
```

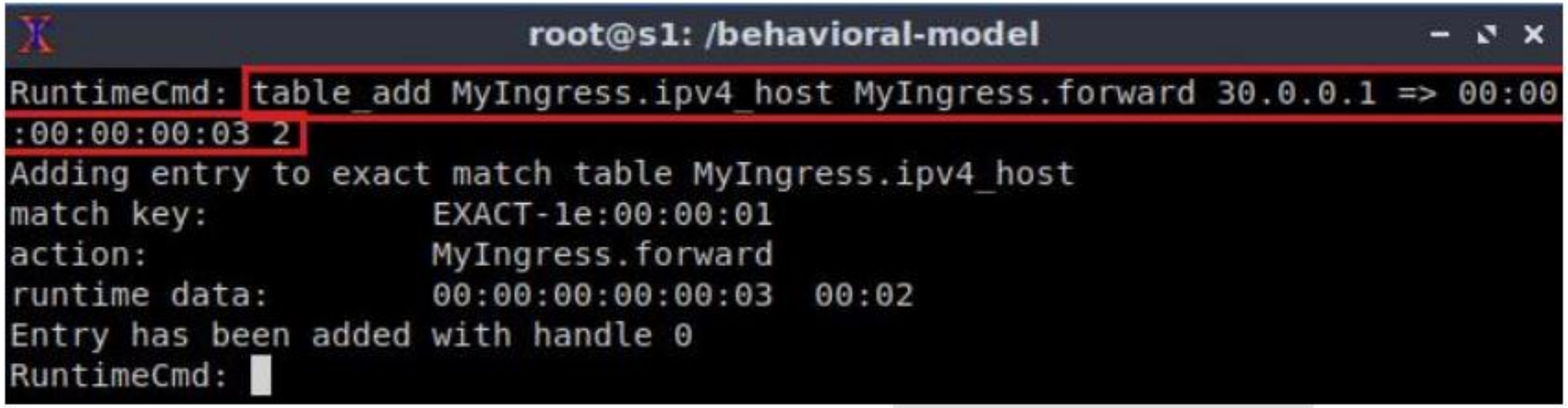
Runtime Environment

- The simple_switch_CLI is similar to other CLIs (e.g., Cisco IOS CLI) and offers a variety of commands



root@s1: /behavioral-model

```
RuntimeCmd: show_tables
MyIngress.ipv4_host            [implementation=None, mk=ipv4.dstAddr(exact, 32)]
MyIngress.ipv4_lpm             [implementation=None, mk=ipv4.dstAddr(lpm, 32)]
RuntimeCmd:
```

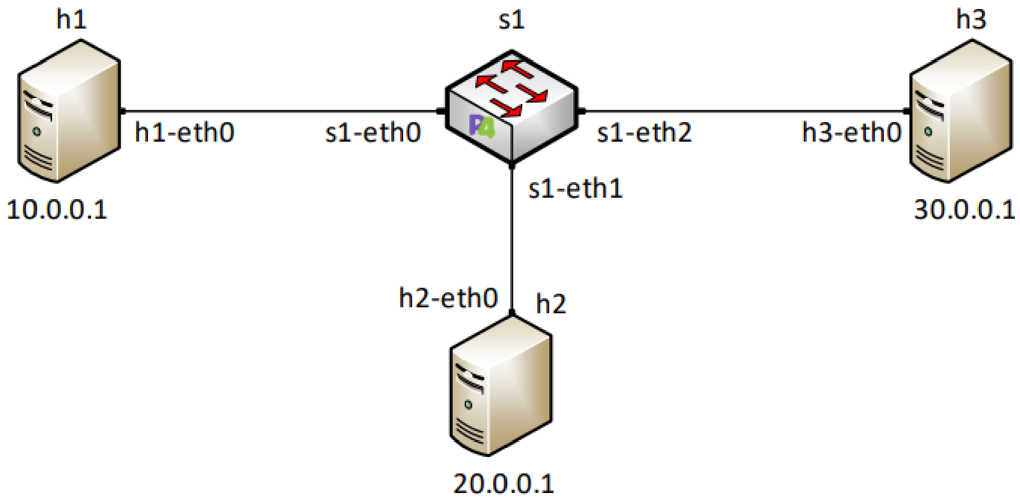


root@s1: /behavioral-model

```
RuntimeCmd: table_add MyIngress.ipv4_host MyIngress.forward 30.0.0.1 => 00:00
:00:00:00:03 2
Adding entry to exact match table MyIngress.ipv4_host
match key:           EXACT-1e:00:00:01
action:              MyIngress.forward
runtime data:        00:00:00:00:00:03   00:02
Entry has been added with handle 0
RuntimeCmd:
```

Lab 7 Topology and Objectives

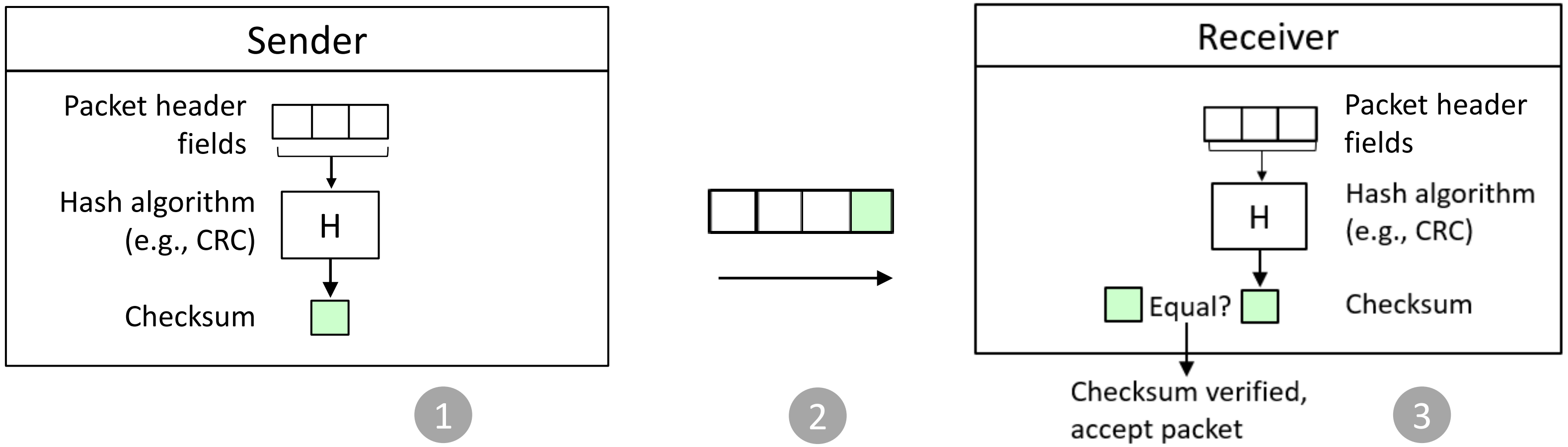
- The topology consists of three hosts: h1, h2, and h3; one P4 switch: s1
- The P4 program is already provided; no P4 programming is needed in this lab
- The objectives are
  - Navigating the simple_switch_CLI tool
  - Displaying ports, tables, and actions
  - Inserting, updating, and deleting table entries



Checksums

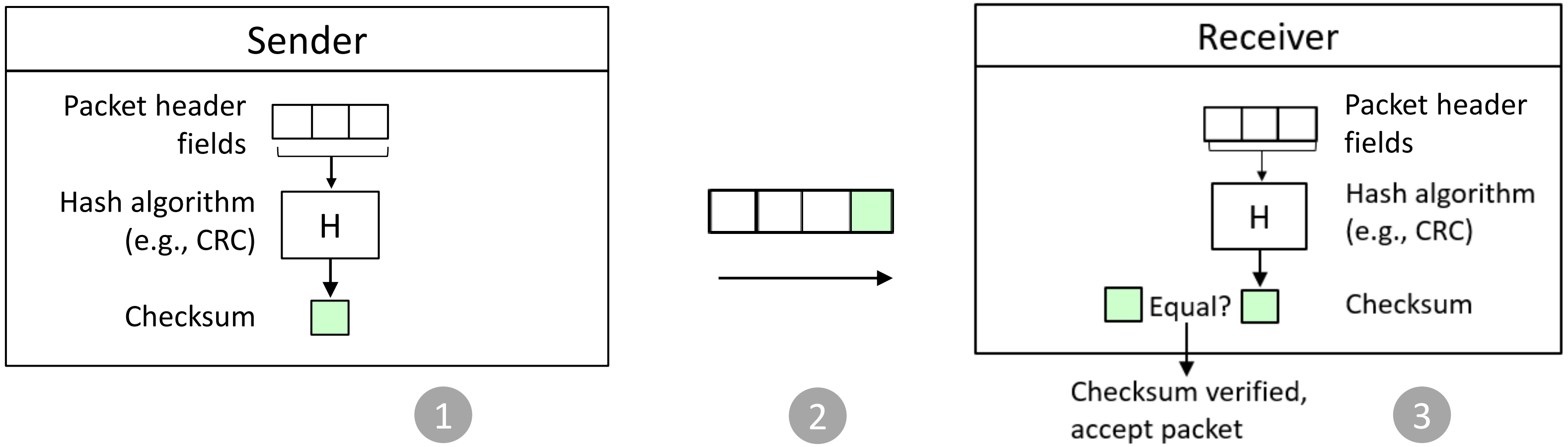
Checksums

- Several protocols use checksums to validate the integrity of the packet headers
- A checksum is a small value computed with a checksum algorithm; e.g., CRC16


Sender
Packet header fields
Hash algorithm (e.g., CRC)
H
Checksum
1
2
Receiver
Packet header fields
Hash algorithm (e.g., CRC)
H
Equal?
Checksum
Checksum verified, accept packet
3

Checksums

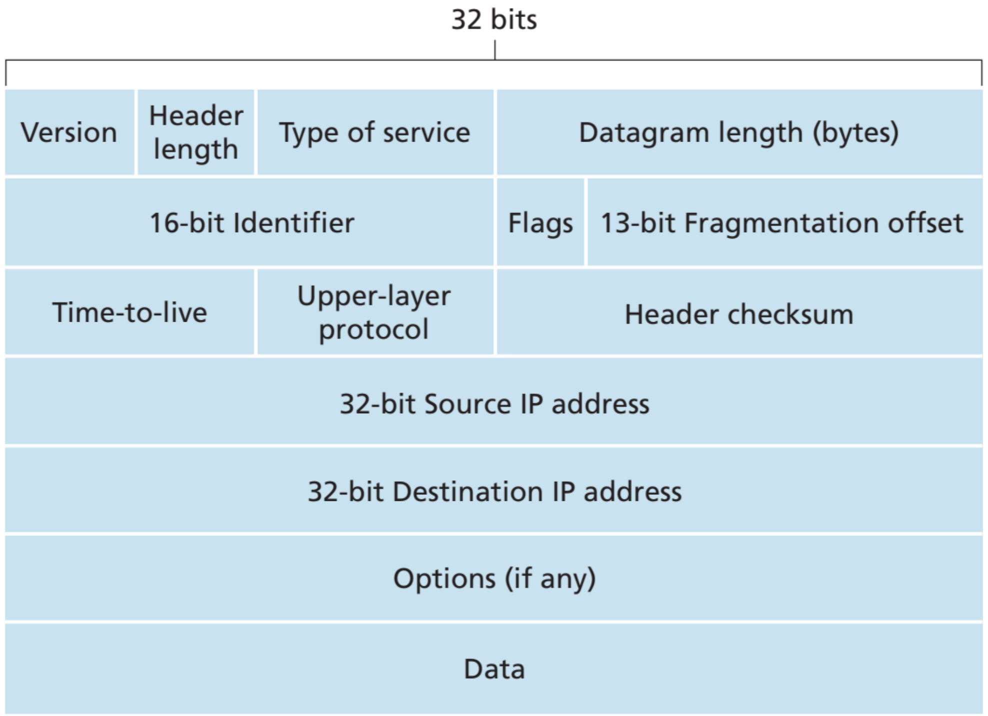
- Several protocols use checksums to validate the integrity of the packet headers
- A checksum is a small value computed with a checksum algorithm; e.g., CRC16
- No built-in constructs in P4$_{16}$; instead, they are expressed as externs (provided by specific libraries)
  - ➢ Externs enable the programmer to use specialized computation provided by the platform


Sender
Packet header fields
Hash algorithm (e.g., CRC)
H
Checksum
1
2
Receiver
Packet header fields
Hash algorithm (e.g., CRC)
H
Equal?
Checksum
Checksum verified, accept packet
3

Checksums

- Several protocols use checksums to validate the integrity of the packet headers
- A checksum is a small value computed with a checksum algorithm; e.g., CRC16
- No built-in constructs in P4$_{16}$; instead, they are expressed as externs (provided by specific libraries)

IP header



32 bits

Version | Header length | Type of service | Datagram length (bytes)

16-bit Identifier | Flags | 13-bit Fragmentation offset

Time-to-live | Upper-layer protocol | Header checksum

32-bit Source IP address

32-bit Destination IP address

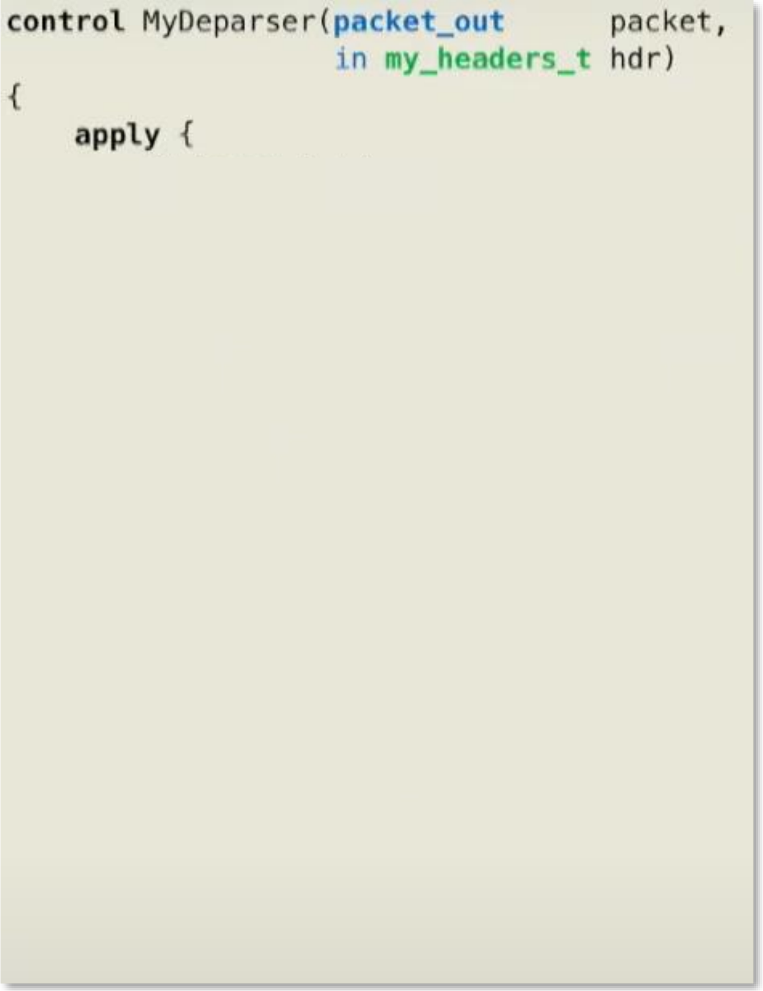
Options (if any)

Data

Deparser

Deparser

- Assembles the headers back into a well-formed packet
- Expressed as a control function (no need for another construct)
- Output parameter is a `packet_out` extern (defined in core.p4)

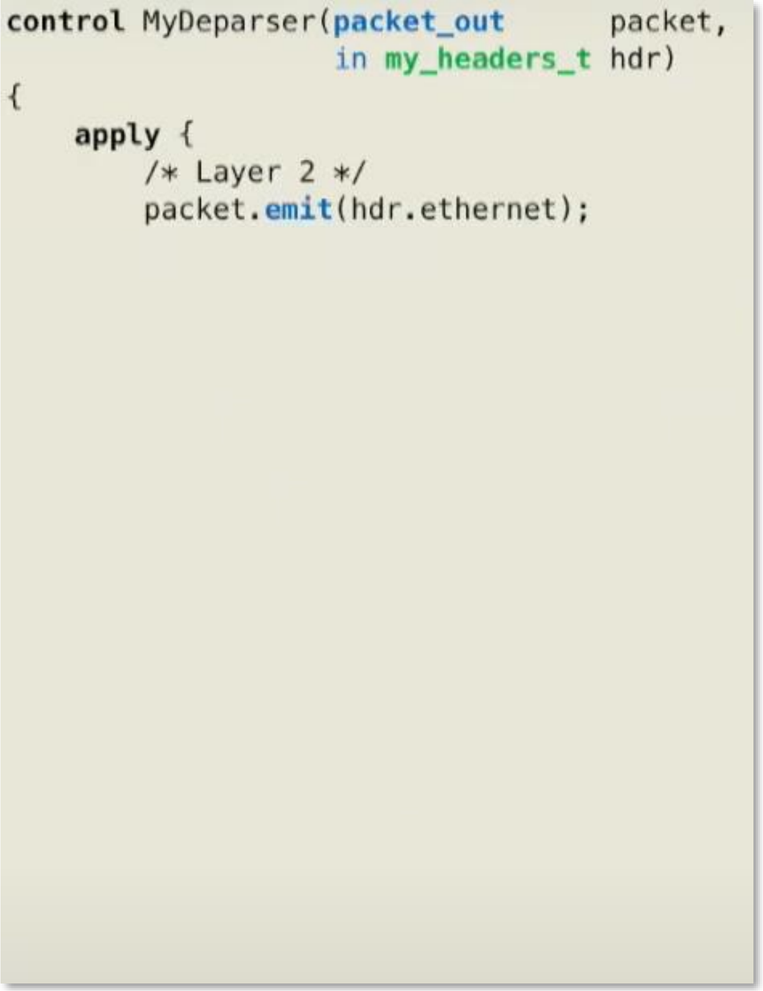


```
control MyDeparser(packet_out      packet,
                   in my_headers_t hdr)
{
    apply {
```

Example from “Introduction to P4$_{16}$ - Part 2”, Vladimir Gurevich.” Online: https://tinyurl.com/23r3nzj9

Deparser

- Assembles the headers back into a well-formed packet
- Expressed as a control function (no need for another construct)
- Output parameter is a `packet_out` extern (defined in core.p4)
- The `emit` method serializes header, if valid

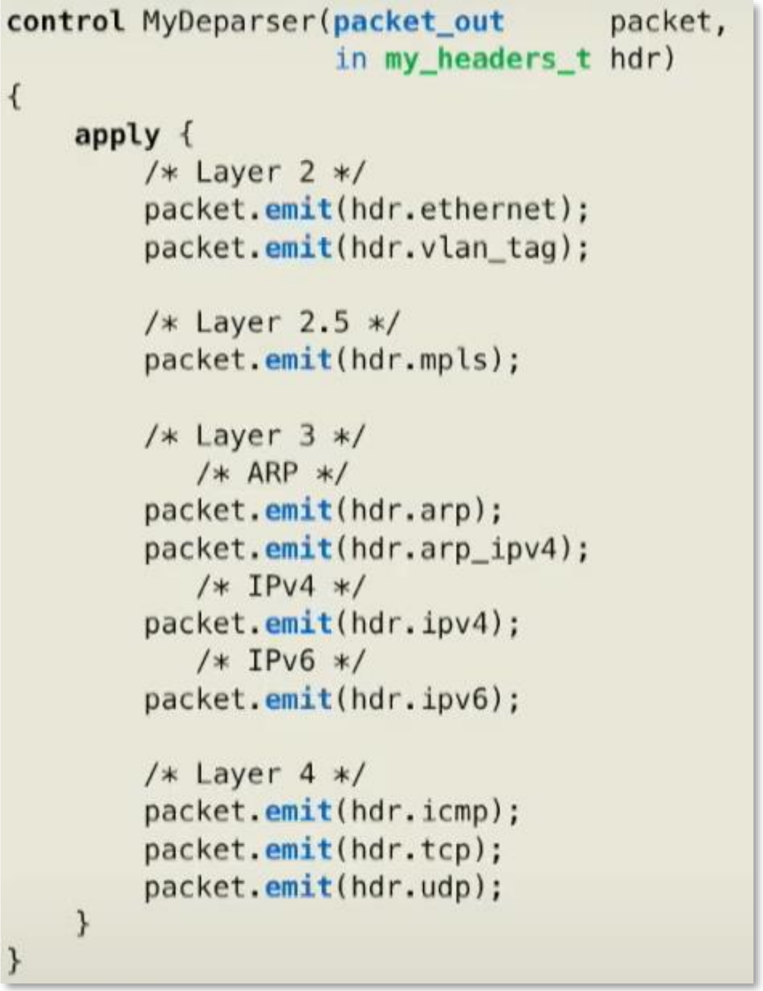


```
control MyDeparser(packet_out      packet,
                   in my_headers_t hdr)
{
    apply {
        /* Layer 2 */
        packet.emit(hdr.ethernet);
```

Example from “Introduction to P4$_{16}$ - Part 2”, Vladimir Gurevich.” Online: https://tinyurl.com/23r3nzj9

Deparser

- Assembles the headers back into a well-formed packet
- Expressed as a control function (no need for another construct)
- Output parameter is a `packet_out` extern (defined in core.p4)
- The `emit` method serializes header, if valid
- If the header is not valid or not available, then the statement has no effect

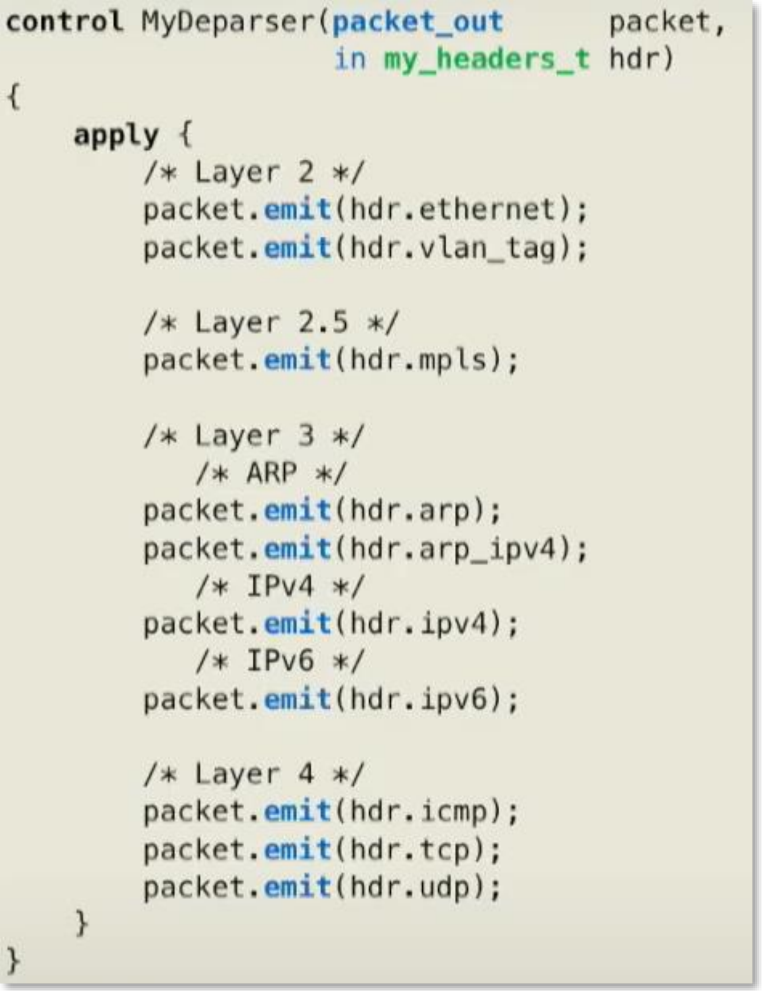


```
control MyDeparser(packet_out       packet,
                   in my_headers_t hdr)
{
    apply {
        /* Layer 2 */
        packet.emit(hdr.ethernet);
        packet.emit(hdr.vlan_tag);

        /* Layer 2.5 */
        packet.emit(hdr.mpls);

        /* Layer 3 */
           /* ARP */
        packet.emit(hdr.arp);
        packet.emit(hdr.arp_ipv4);
           /* IPv4 */
        packet.emit(hdr.ipv4);
           /* IPv6 */
        packet.emit(hdr.ipv6);

        /* Layer 4 */
        packet.emit(hdr.icmp);
        packet.emit(hdr.tcp);
        packet.emit(hdr.udp);
    }
}
```

Example from “Introduction to P4$_{16}$ - Part 2”, Vladimir Gurevich.” Online: https://tinyurl.com/23r3nzj9

Deparser

- Assembles the headers back into a well-formed packet
- Expressed as a control function (no need for another construct)
- Output parameter is a `packet_out` extern (defined in core.p4)
- The `emit` method serializes header, if valid
- If the header is not valid or not available, then the statement has no effect
- The deparser is decoupled from the parser
- The deparser can have conditional statements (as in other control blocks)

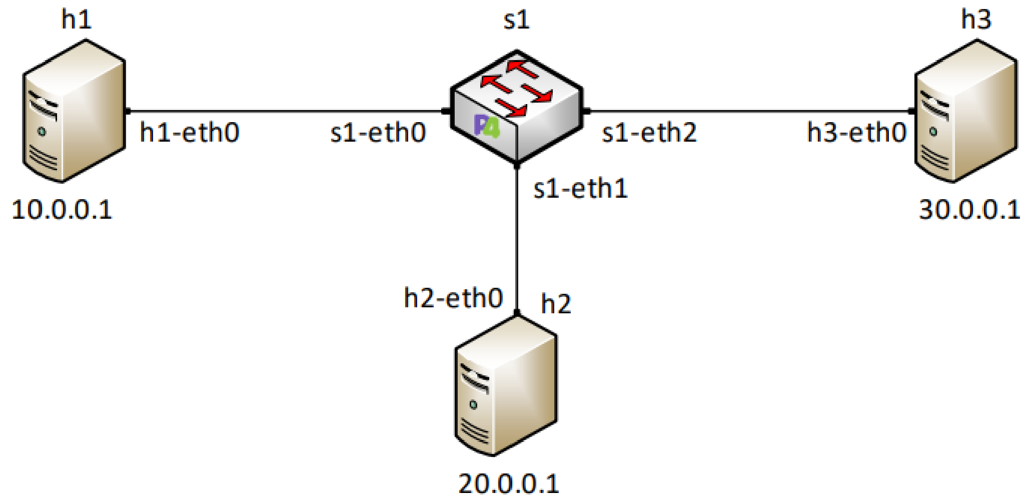


```
control MyDeparser(packet_out      packet,
                   in my_headers_t hdr)
{
    apply {
        /* Layer 2 */
        packet.emit(hdr.ethernet);
        packet.emit(hdr.vlan_tag);

        /* Layer 2.5 */
        packet.emit(hdr.mpls);

        /* Layer 3 */
           /* ARP */
        packet.emit(hdr.arp);
        packet.emit(hdr.arp_ipv4);
           /* IPv4 */
        packet.emit(hdr.ipv4);
           /* IPv6 */
        packet.emit(hdr.ipv6);

        /* Layer 4 */
        packet.emit(hdr.icmp);
        packet.emit(hdr.tcp);
        packet.emit(hdr.udp);
    }
}
```

Example from “Introduction to $P4_{16}$ - Part 2”, Vladimir Gurevich.” Online: https://tinyurl.com/23r3nzj9

Lab 8 Topology and Objectives

- The topology consists of three hosts: h1, h2, and h3; one P4 switch: s1
- The P4 program modifies the headers of the packet
- The P4 program recomputes the checksum of the updated headers
- The objectives are
  - ➢ Validating and implementing checksums
  - ➢ Understanding and implementing a deparser


h1
s1
h3
h1-eth0
s1-eth0
s1-eth2
h3-eth0
s1-eth1
10.0.0.1
30.0.0.1
h2-eth0
h2
20.0.0.1