

Runtime Controller, Checksum Calculation, Deparser

Jorge Crichigno

College of Engineering and Computing, University of South Carolina

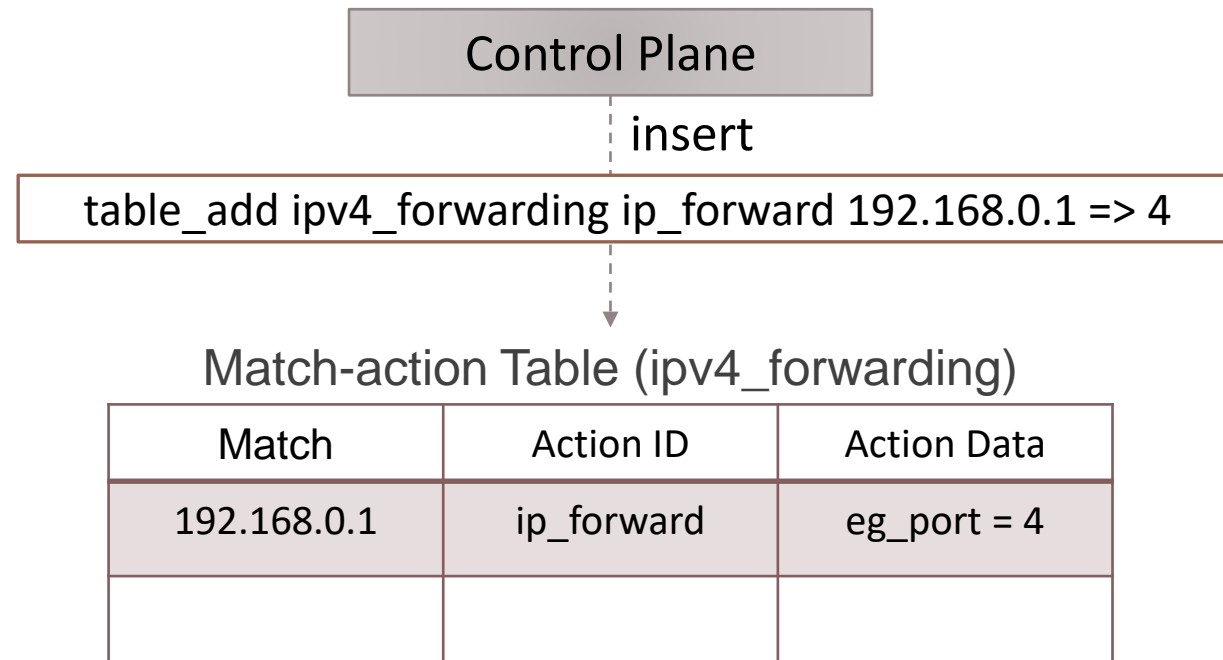
A Hands-on Tutorial on P4 Programmable Data Planes

Tuesday March 7, 2023

Runtime Controller

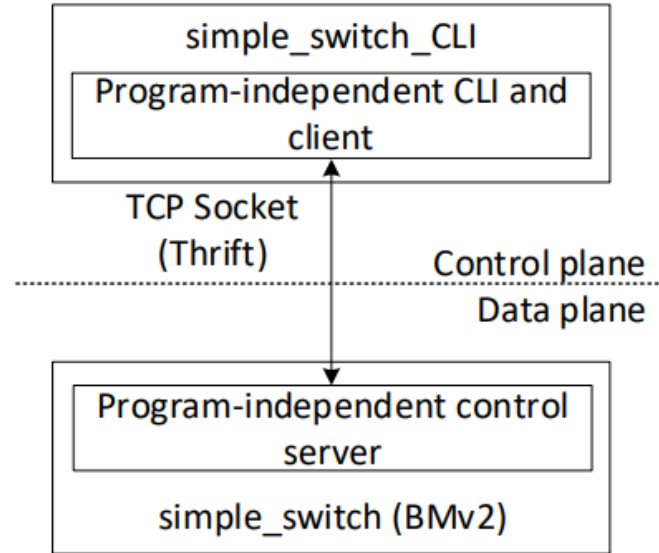
Control Plane

- 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



Runtime Environment

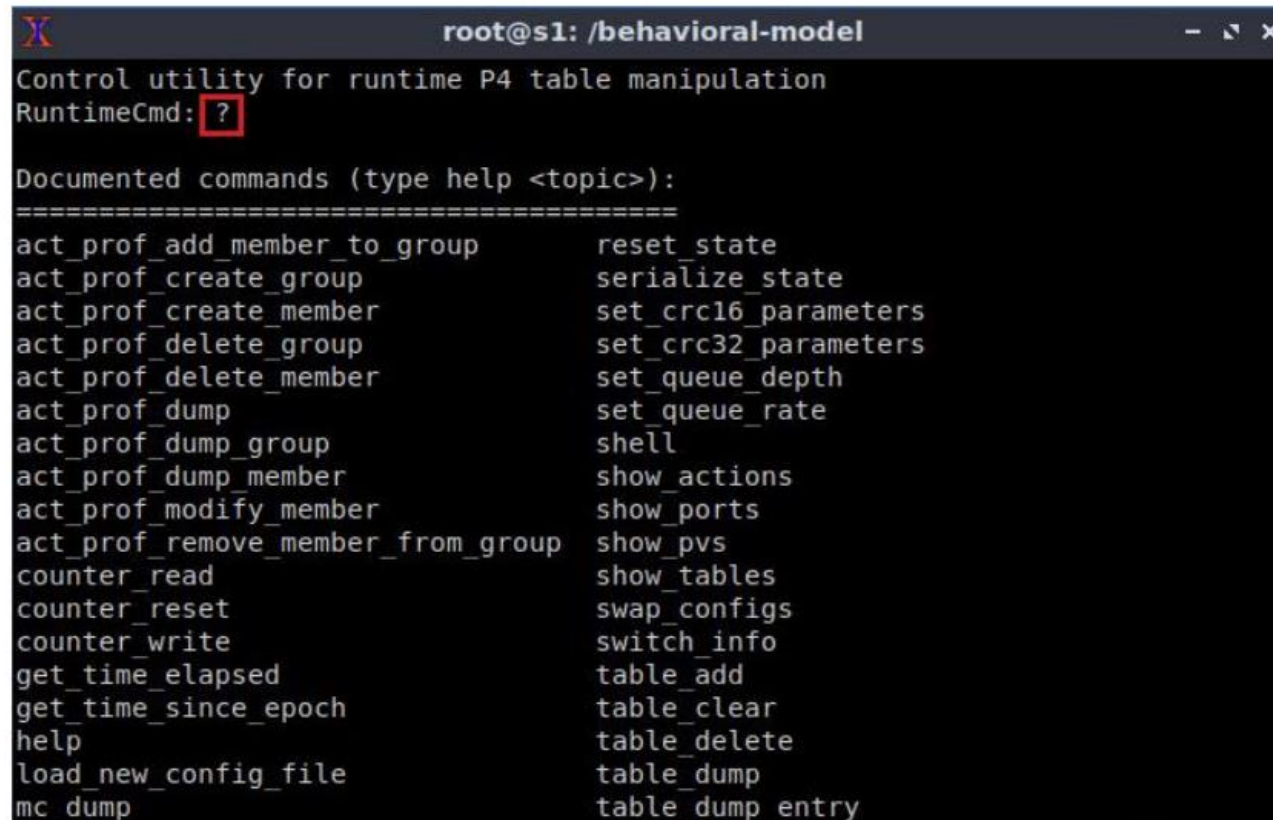
- The `simple_switch_CLI` tool is used to populate the tables in this lab series
- This tool includes a program-independent CLI and a Thrift¹ client
- It connects to a Thrift control server residing on the switch



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



```
root@s1: /behavioral-model
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

```
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: █
```

Runtime Environment

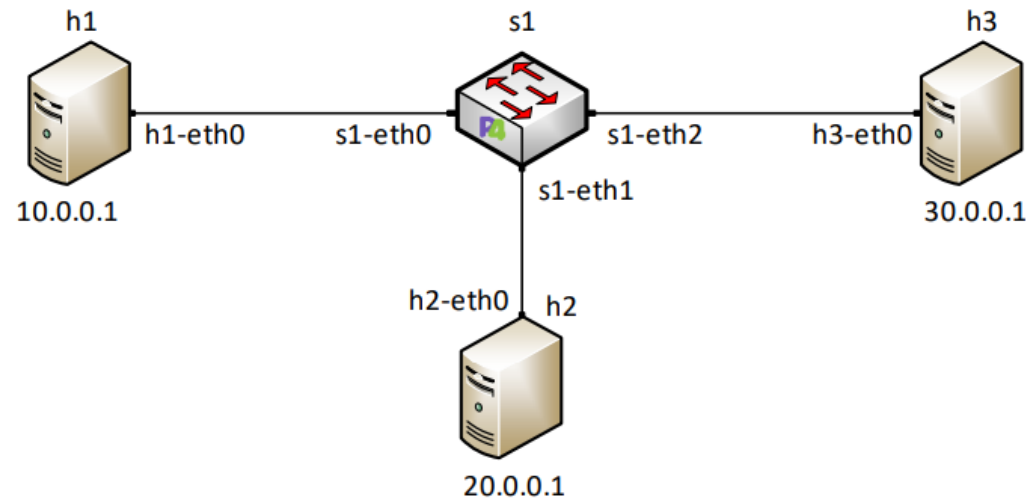
- The simple_switch_CLI is similar to other CLIs (e.g., Cisco IOS CLI) and offers a variety of commands

```
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 => 00:00:00:00:03 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

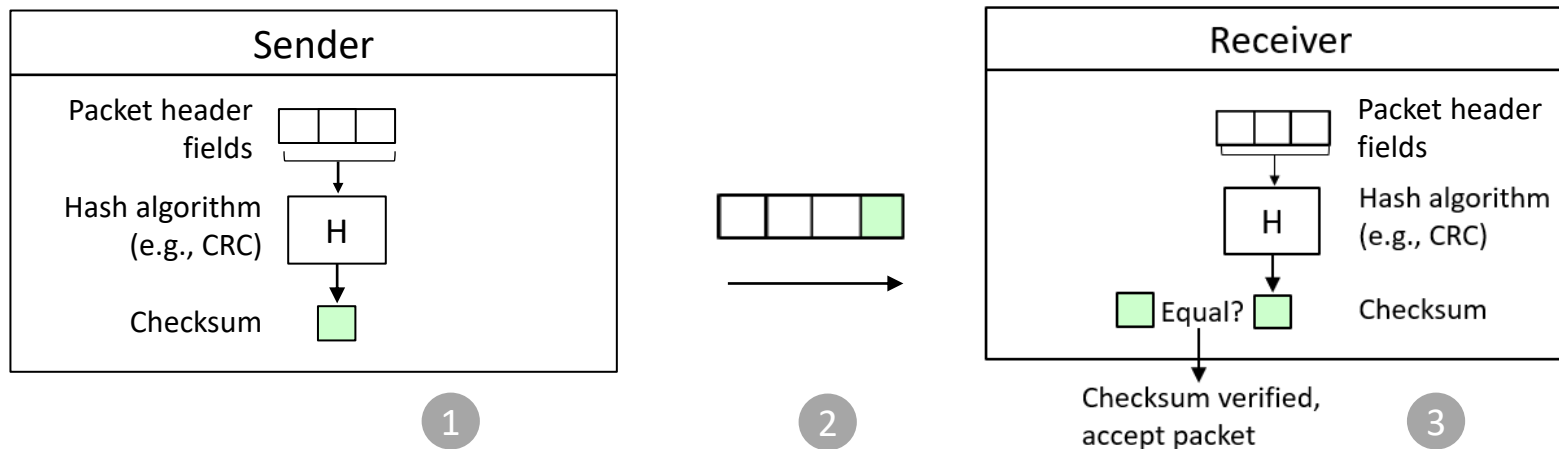
- The topology consists of three hosts: h1, h2, and h3; one P4 switch: s1
- The P4 program is already provided; no P4 programming is needed in this lab
- The objectives are
 - Navigating the simple_switch_CLI tool
 - Displaying ports, tables, and actions
 - Inserting, updating, and deleting table entries



Checksums

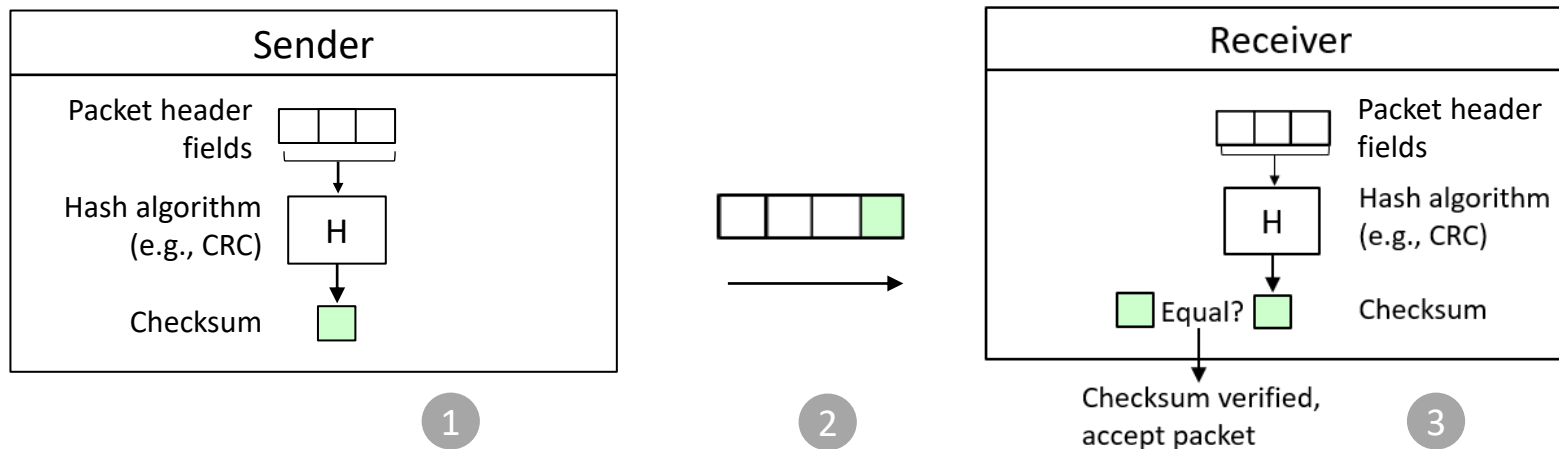
Checksums

- 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



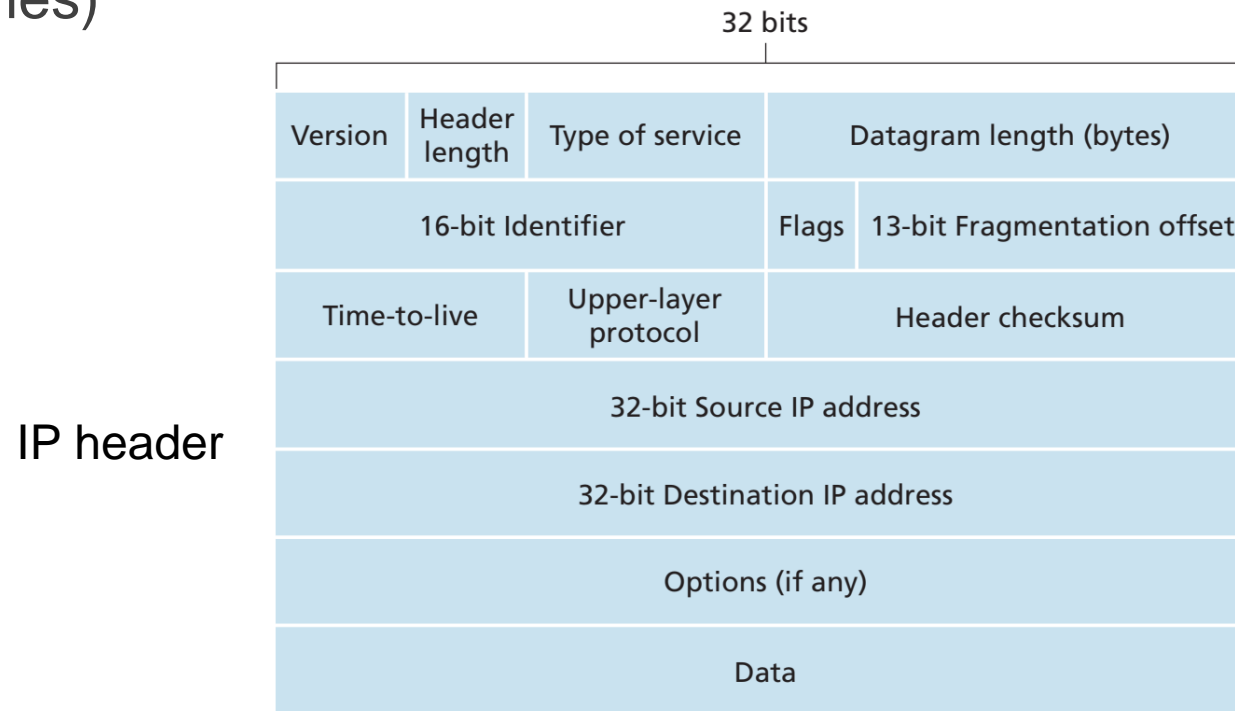
Checksums

- 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 P4₁₆; instead, they are expressed as externs (provided by specific libraries)
 - Externs enable the programmer to use specialized computation provided by the platform



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 P4₁₆; instead, they are expressed as externs (provided by specific libraries)



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 P4₁₆ - Part 2”, Vladimir Gurevich.” Online: <https://tinyurl.com/23r3nzi9>

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 P4₁₆ - Part 2”, Vladimir Gurevich.” Online: <https://tinyurl.com/23r3nzi9>

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 P4₁₆ - Part 2", Vladimir Gurevich." Online: <https://tinyurl.com/23r3nzi9>

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 P4₁₆ - Part 2”, Vladimir Gurevich.” Online: <https://tinyurl.com/23r3nzi9>

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

