Runtime Controller, Checksum Calculation, Deparser

Jorge Crichigno

College of Engineering and Computing, University of South Carolina

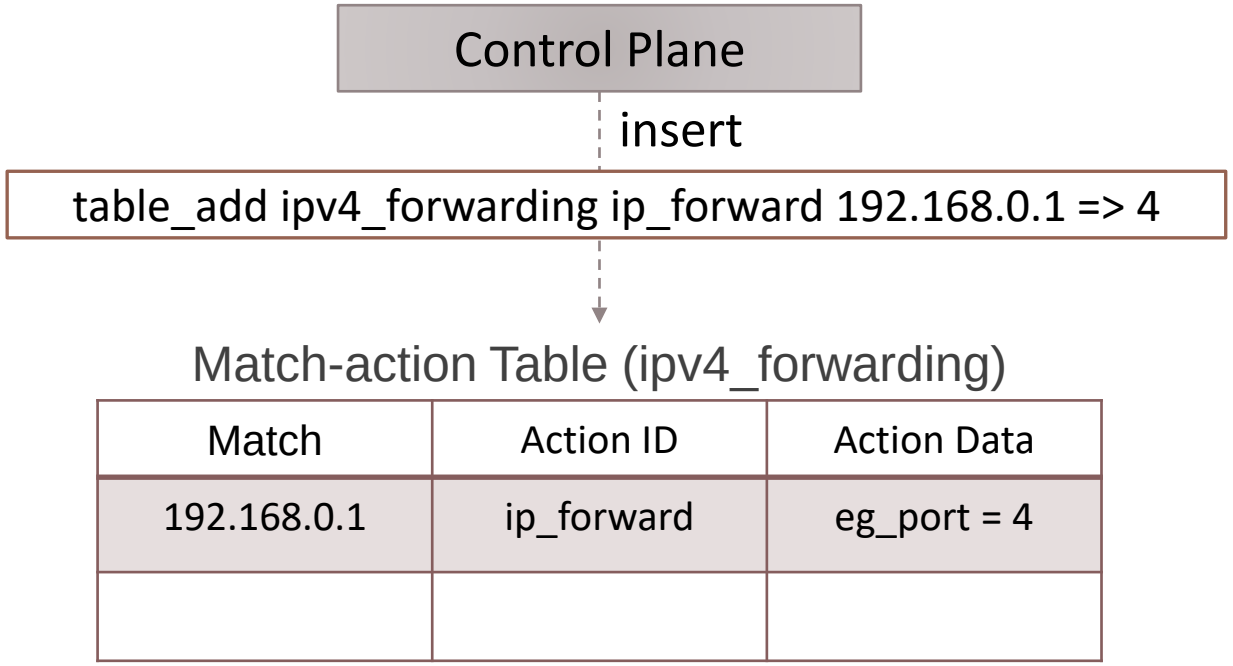
A Hands-on Tutorial on P4 Programmable Data Planes

Tuesday March 7, 2023

Runtime Controller

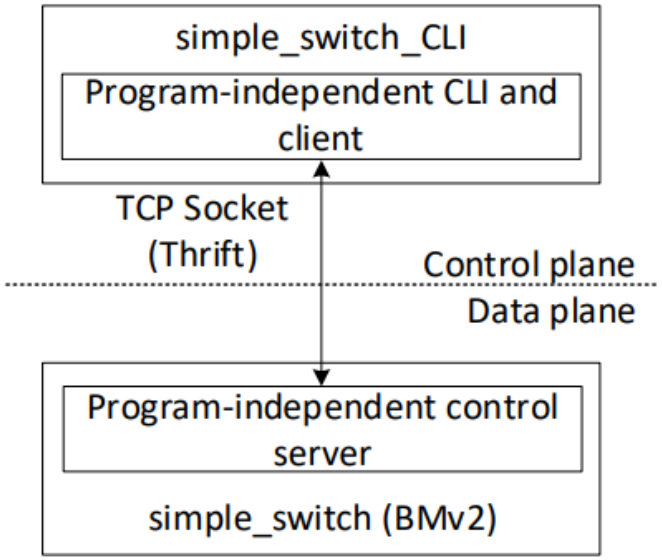
Control Plane

- 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

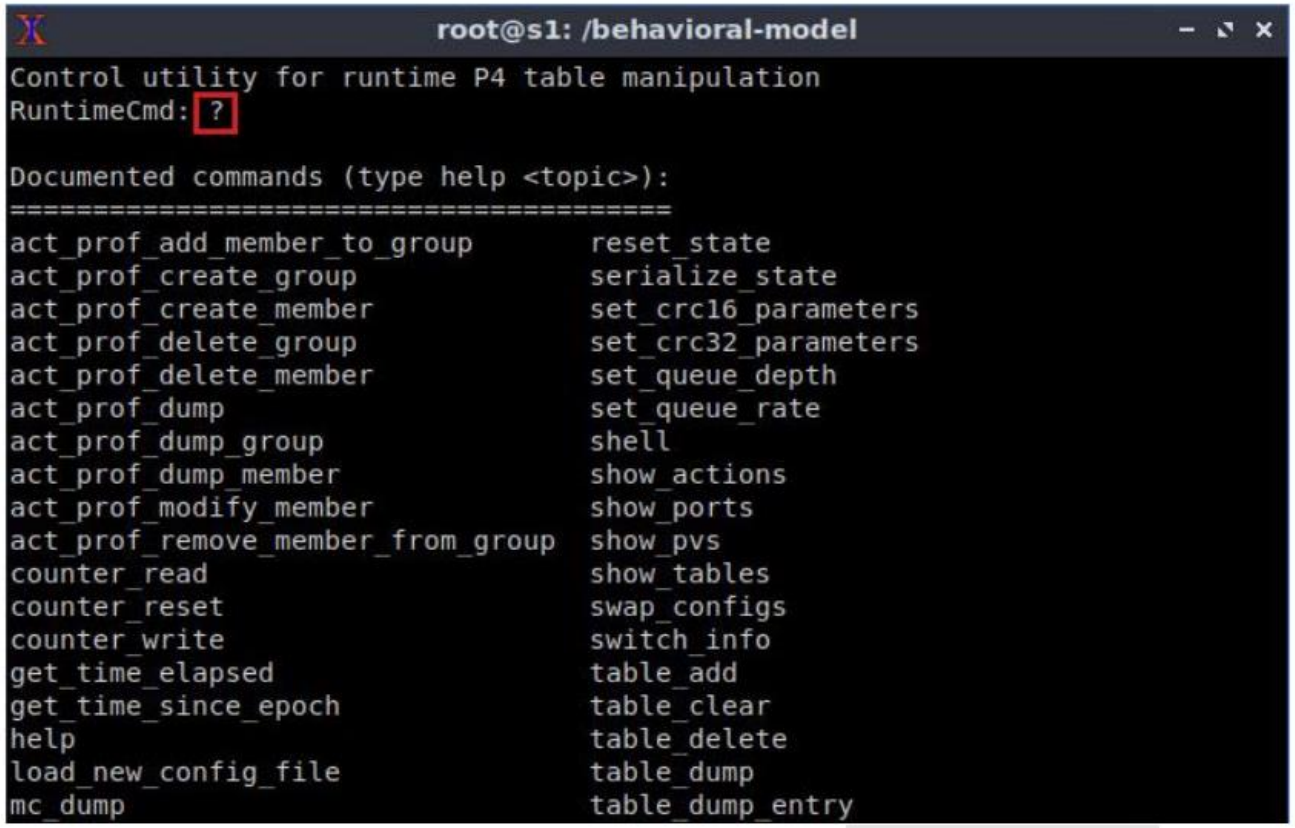
- The simple_switch_CLI tool is used to populate the tables in this lab series
- This tool includes a program-independent CLI and a Thrift[1] client
- It connects to a Thrift control server residing on the switch



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

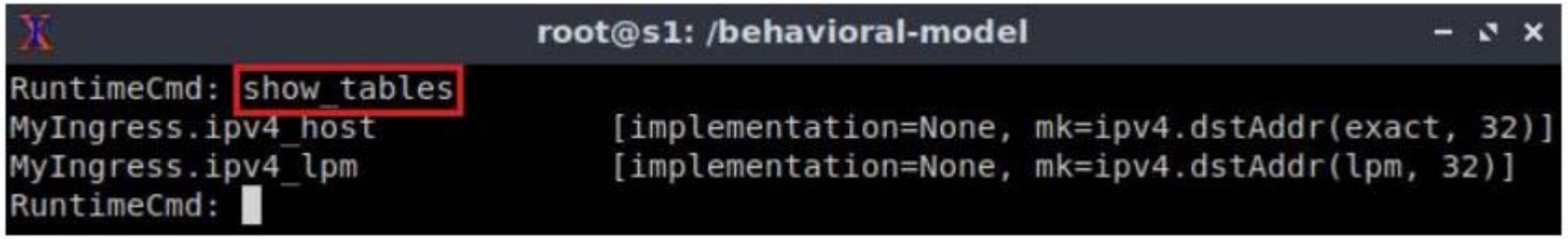
Runtime Environment

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



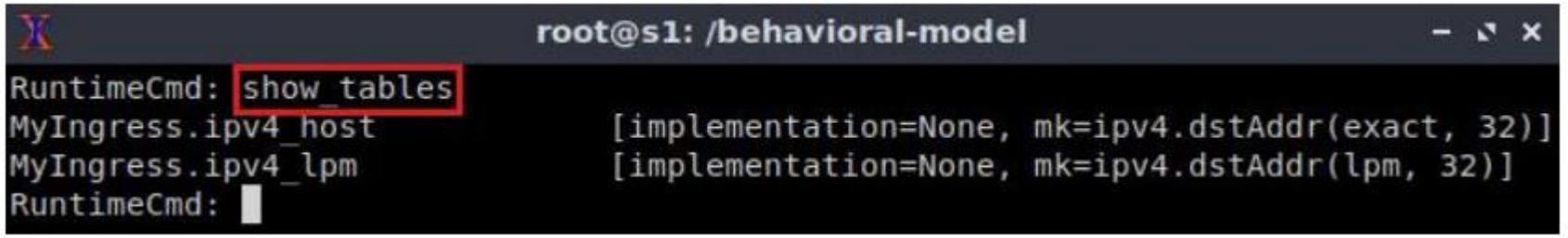
Runtime Environment

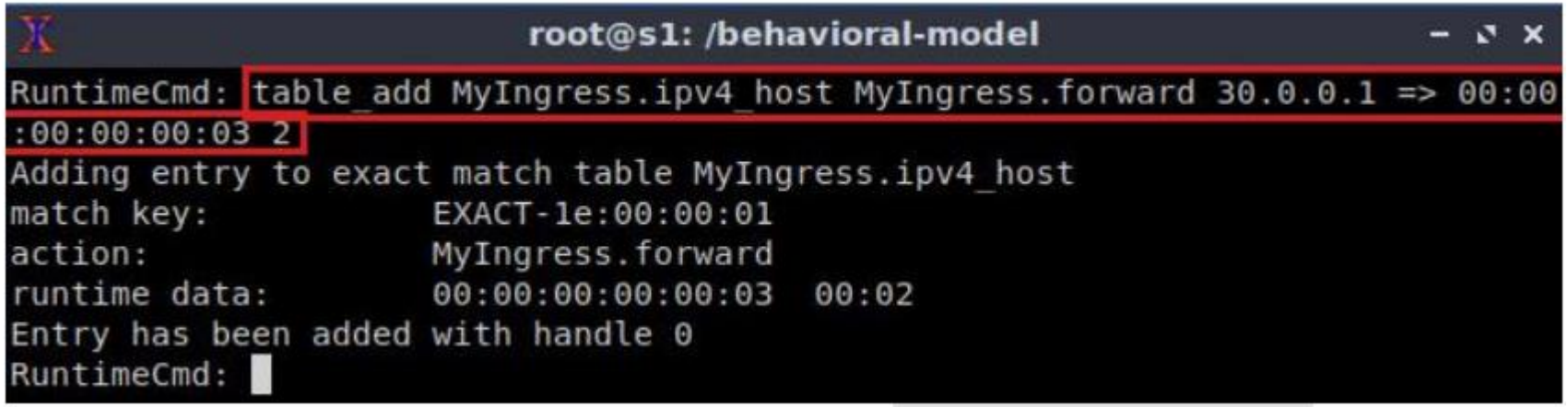
- The simple_switch_CLI is similar to other CLIs (e.g., Cisco IOS CLI) and offers a variety of commands



Runtime Environment

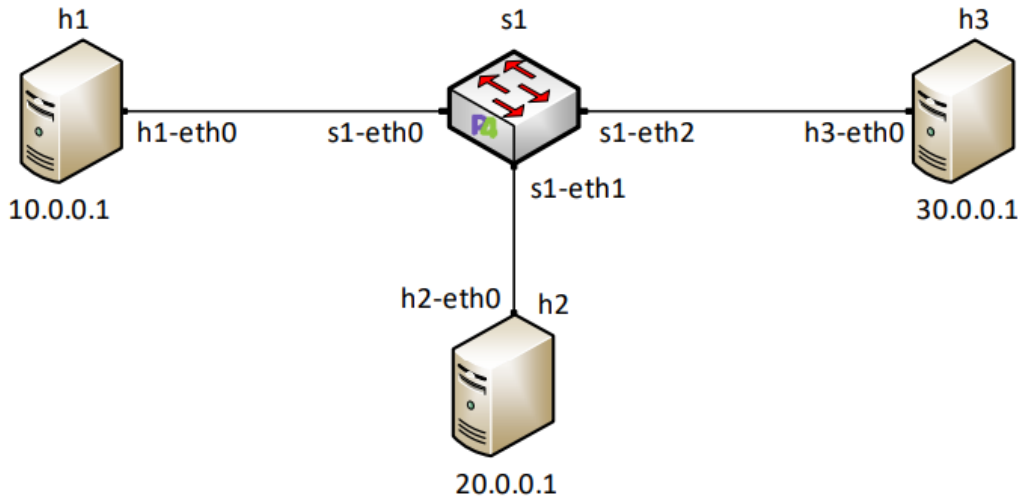
- The simple_switch_CLI is similar to other CLIs (e.g., Cisco IOS CLI) and offers a variety of commands





Lab 7 Topology and Objectives

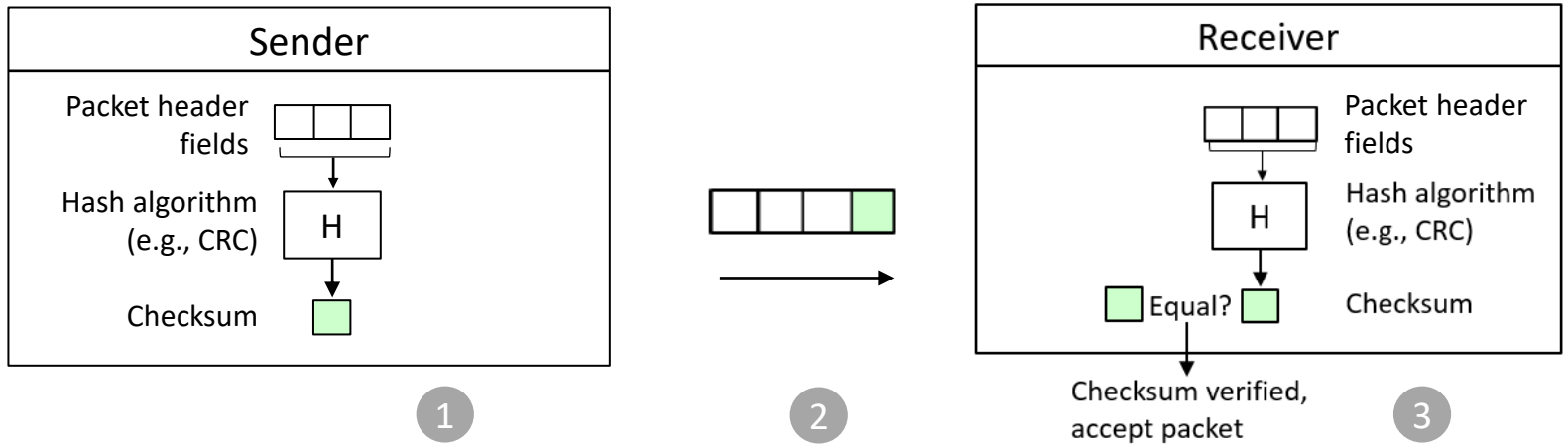
- The topology consists of three hosts: h1, h2, and h3; one P4 switch: s1
- The P4 program is already provided; no P4 programming is needed in this lab
- The objectives are
  - Navigating the simple_switch_CLI tool
  - Displaying ports, tables, and actions
  - Inserting, updating, and deleting table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



Checksums

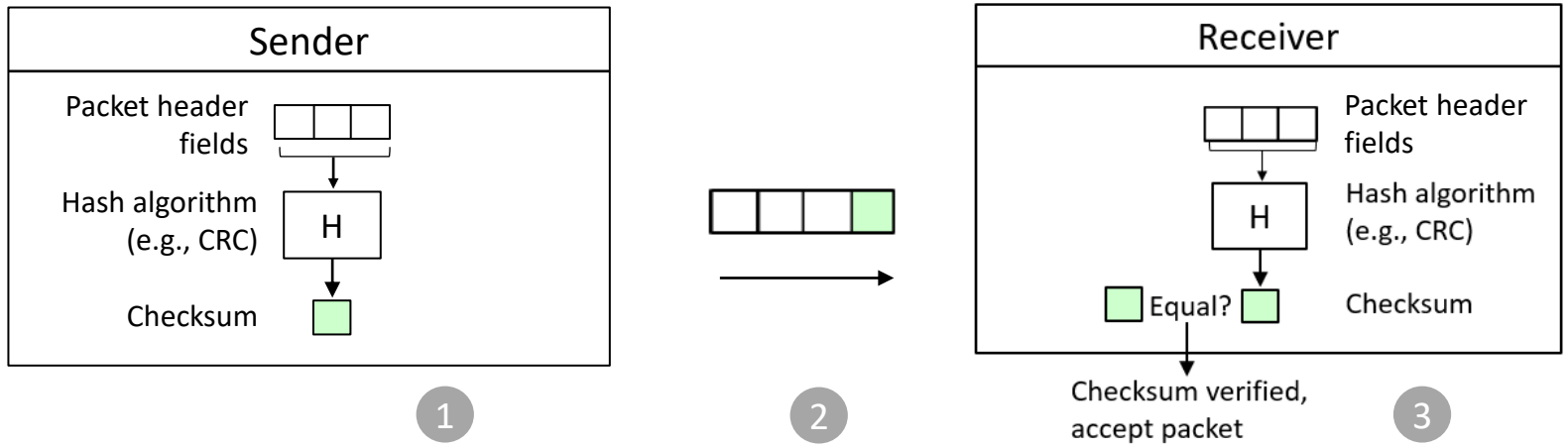
Checksums

- 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

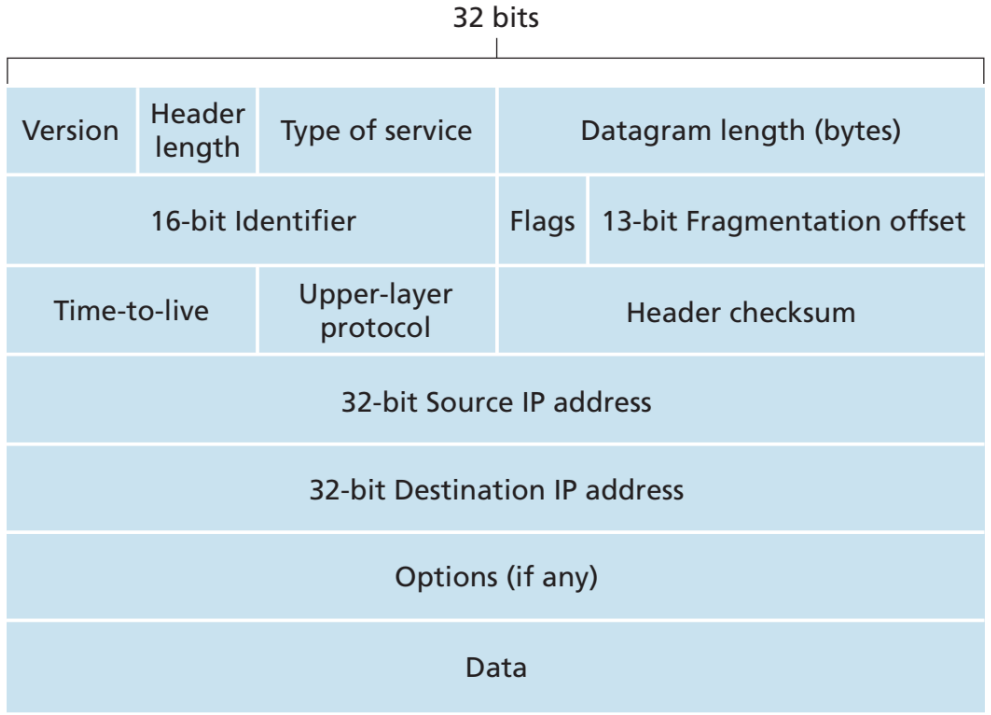
- Several protocols use checksums to validate the integrity of the packet headers
- A checksum is a small value computed with a checksum algorithm; e.g., CRC16
- No built-in constructs in $P4_{16}$; instead, they are expressed as externs (provided by specific libraries)
  - Externs enable the programmer 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 $P4_{16}$; instead, they are expressed as externs (provided by specific libraries)

IP header



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