Standard Metadata, Counters, and Meters

Jorge Crichigno

College of Engineering and Computing, University of South Carolina

A Hands-on Tutorial on P4 Programmable Data Planes

Wednesday March 8, 2023

Standard Metadata

Standard Metadata

- Metadata is state associated with each packet

- It can be treated like a set of variables associated with each packet, read and written by actions executed by tables

- Some metadata has special significance to the operation of the switch
  - ➢ This metadata is called Intrinsic Metadata, because it has intrinsic semantics to the operation of the machine[1]

1. The P4 Language Specification. Online: https://tinyurl.com/4zkwjp4b

V1 Model Standard Metadata

- Metadata V1 model

```
struct standard_metadata_t {
    bit<9>  ingress_port;
    bit<9>  egress_spec;
    bit<9>  egress_port;
    bit<32> clone_spec;
    bit<32> instance_type;
    bit<1>  drop;
    bit<16> recirculate_port;
    bit<32> packet_length;
    bit<32> enq_timestamp;
    bit<19> enq_qdepth;
    bit<32> deq_timedelta;
    bit<19> deq_qdepth;
    bit<48> ingress_global_timestamp;
    bit<32> lf_field_list;
    bit<16> mcast_grp;
    bit<1>  resubmit_flag;
    bit<16> egress_rid;
    bit<1>  checksum_error;
}
```
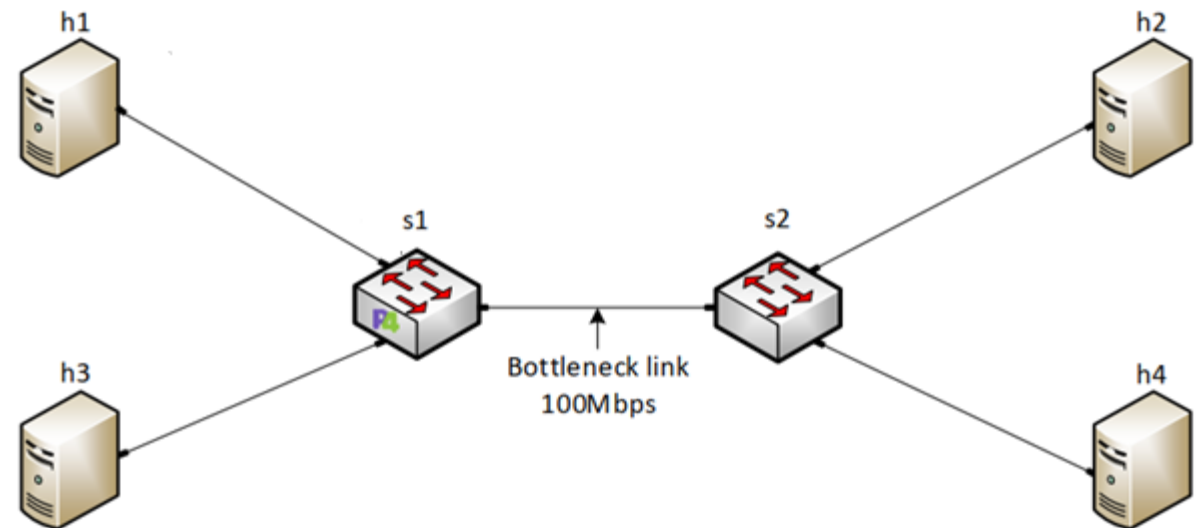
- `ingress_port`
  - ➤ port on which the packet arrived

- `egress_spec`
  - ➤ egress intended port set during the ingress pipeline

- `ingress_global_timestamp`
  - ➤ a timestamp, in microseconds, set when the packet shows up on ingress

- `egress_global_timestamp`
  - ➤ a timestamp, in microseconds, set when the packet starts egress processing

- `enq_qdepth`
  - ➤ depth of queue when the packet was first enqueued, in number of packets

1. The P4 Language Specification. Online: https://tinyurl.com/4zkwjp4b

V1 Model Standard Metadata

- Metadata V1 model

```
struct standard_metadata_t {
    bit<9>  ingress_port;
    bit<9>  egress_spec;
    bit<9>  egress_port;
    bit<32> clone_spec;
    bit<32> instance_type;
    bit<1>  drop;
    bit<16> recirculate_port;
    bit<32> packet_length;
    bit<32> enq_timestamp;
    bit<19> enq_qdepth;
    bit<32> deq_timedelta;
    bit<19> deq_qdepth;
    bit<48> ingress_global_timestamp;
    bit<32> lf_field_list;
    bit<16> mcast_grp;
    bit<1>  resubmit_flag;
    bit<16> egress_rid;
    bit<1>  checksum_error;
}
```

Application: compute the time the packet is waiting in the queue (Traffic Manager)

h1

h2

s1

s2

h3

Bottleneck link
100Mbps
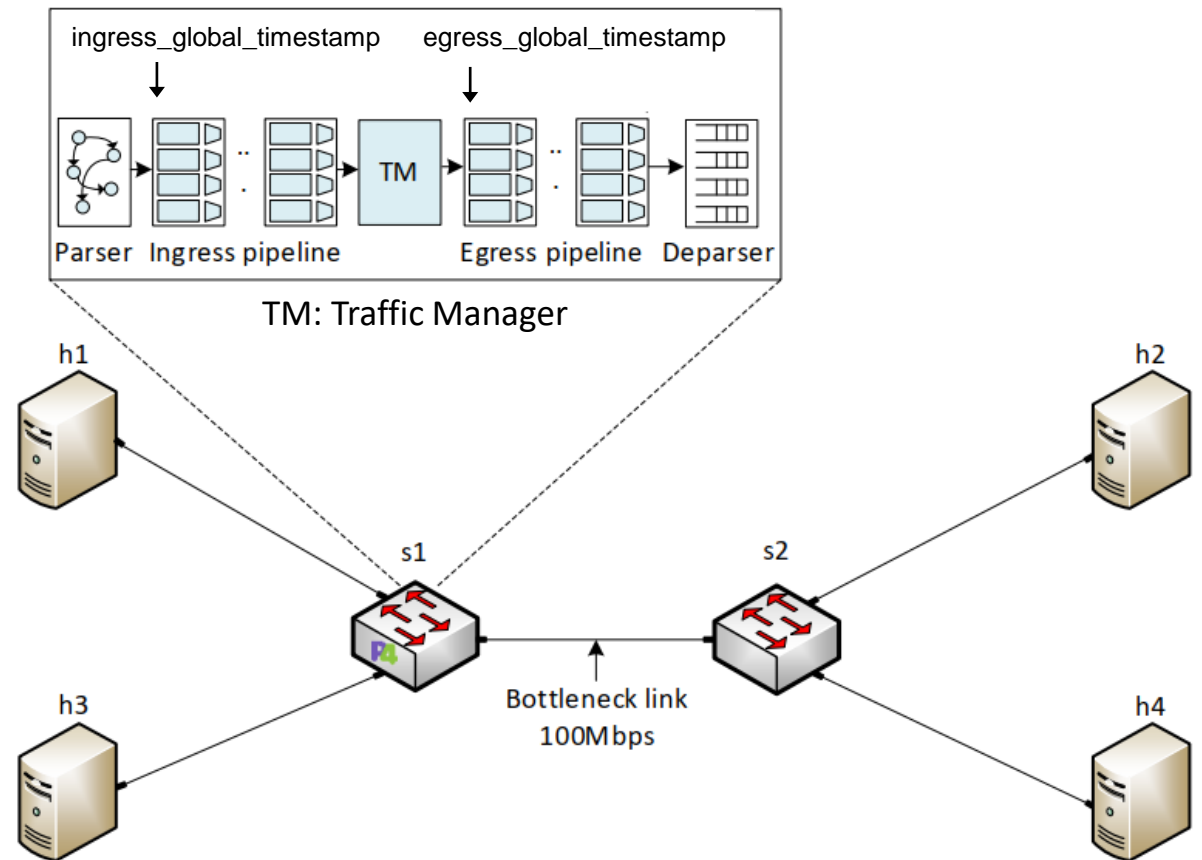
h4

V1 Model Standard Metadata

- Metadata V1 model

```
struct standard_metadata_t {
    bit<9> ingress_port;
    bit<9> egress_spec;
    bit<9> egress_port;
    bit<32> clone_spec;
    bit<32> instance_type;
    bit<1> drop;
    bit<16> recirculate_port;
    bit<32> packet_length;
    bit<32> enq_timestamp;
    bit<19> enq_qdepth;
    bit<32> deq_timedelta;
    bit<19> deq_qdepth;
    bit<48> ingress_global_timestamp;
    bit<32> lf_field_list;
    bit<16> mcast_grp;
    bit<1> resubmit_flag;
    bit<16> egress_rid;
    bit<1> checksum_error;
}
```

Application: compute the time the packet is waiting in the queue (Traffic Manager)



TM: Traffic Manager

# Lab 5 Topology and Objectives

- The topology consists of four hosts: h1, h2, h3, and h4; one P4 switch: s1; and one legacy switch: s2

- The objectives are
  - ➢ Understanding the V1Model standard metadata
  - ➢ Defining custom headers
  - ➢ Using custom headers to monitor the switch's queue
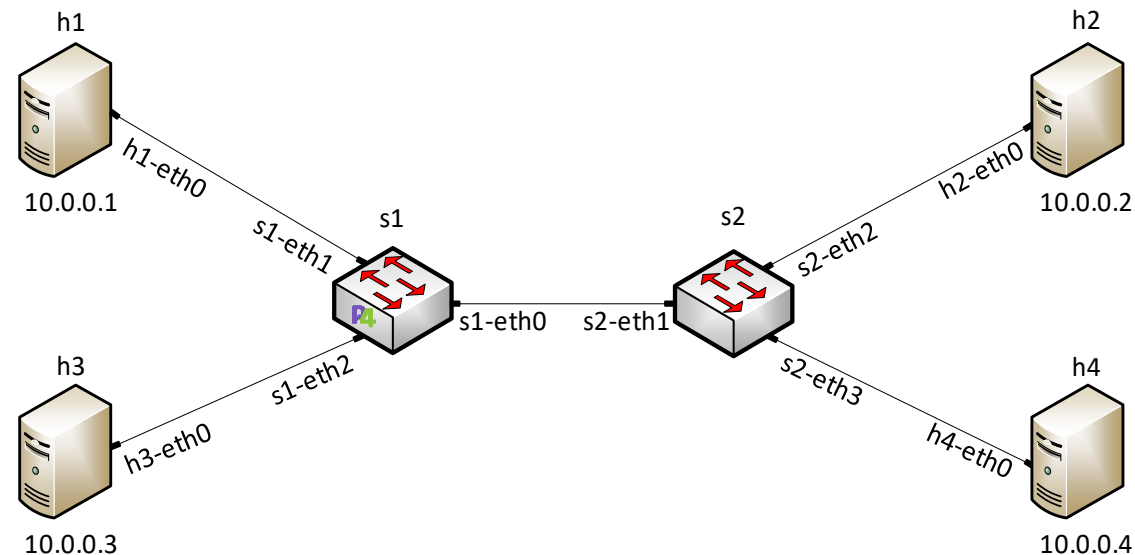
# Counters

# Stateless and Stateful Objects

- Stateless objects (transient) do not preserve the state between packets
  - Metadata (variables)
  - Packet headers
- Stateful objects (persistent) preserve state between packets
  - Tables
  - Counters
  - Meters ⎤ Referred to as **stateful memories** in the P4 Language Specification[1]
  - Registers ⎦
- Stateful memories require resources on the target and hence are managed by the compiler

1. P4 Language Specification, Online: https://tinyurl.com/4zkwjp4b.

# P4 Counters

- Counters are a mechanism for keeping statistics
- A P4 program (data plane) can update counter values but cannot read them
- The control plane can read counter values and use them for other control applications
- Counters only support packet counters, byte counters, or a combination of both
- There are two types of counters: direct and indirect

# P4 Direct Counters

- Direct counters are associated to a match-action table –effectively extend the table

# P4 Direct Counters

- Direct counters are associated to a match-action table –effectively extend the table
- Example:
  - ➢ instantiate a counter as a counter of packets and bytes

```
 1: control MyIngress(inout header hdr,
 2:                    inout metadata meta,
 3:                    inout standard_metadata_t standard_metadata){
 4: direct_counter(counterType.packets_and_bytes) my_direct_counter;
 5:
 6: action forward(egressSpect_t port){
 7:     standard_meadata.egress_spec = port;
 8: }
 9:
10: action drop(){
11:     mark_to_drop(standard_metadata);
12: }
13:
14: table forwarding {
15:     key = {
16:         hdr.ipv4.dstAddr : exact;
17:      }
18:     actions = {
19:         forward;
20:         drop;
21:         NoAction;
22:     }
23:     size = 32;
24:     default_action = drop();
25:     counters = my_direct_counter;
26: }
27: apply {
28:     if(hdr.ipv4.isValid()){
29:         forwarding.apply();
30:     }
31: }
```

P4 Direct Counters

- Direct counters are associated to a match-action table –effectively extend the table
- Example:
  - ➢ instantiate a counter as a counter of packets and bytes
  - ➢ specify the counter as a property of the table of interest

```
 1: control MyIngress(inout header hdr,
 2:                    inout metadata meta,
 3:                    inout standard_metadata_t standard_metadata){
 3:
 4: direct_counter(counterType.packets_and_bytes) my_direct_counter;
 5:
 6: action forward(egressSpect_t port){
 7:      standard_meadata.egress_spec = port;
 8: }
 9:
10: action drop(){
11:      mark_to_drop(standard_metadata);
12: }
13:
14: table forwarding {
15:      key = {
16:          hdr.ipv4.dstAddr : exact;
17:       }
18:      actions = {
19:          forward;
20:          drop;
21:          NoAction;
22:      }
23:      size = 32;
24:      default_action = drop();
25:      counters = my_direct_counter;
26: }
27: apply {
28:      if(hdr.ipv4.isValid()){
29:          forwarding.apply();
30:      }
31: }
```
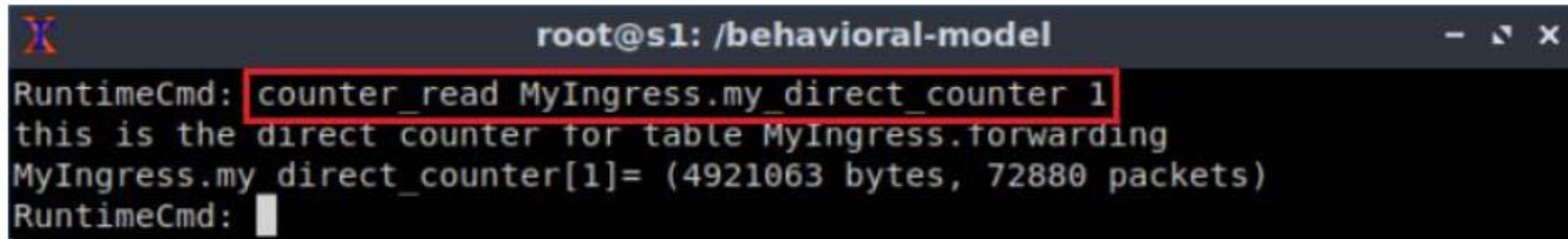
P4 Direct Counters

- A single instantiation of a direct counter always contains as many independent counter values as the number of entries in the associated table

```
 1: control MyIngress(inout header hdr,
 2:                    inout metadata meta,
 3:                    inout standard_metadata_t standard_metadata){
 3:
 4: direct_counter(counterType.packets_and_bytes) my_direct_counter;
 5:
 6: action forward(egressSpect_t port){
 7:         standard_meadata.egress_spec = port;
 8: }
 9:
10: action drop(){
11:         mark_to_drop(standard_metadata);
12: }
13:
14: table forwarding {
15:         key = {
16:             hdr.ipv4.dstAddr : exact;
17:         }
18:         actions = {
19:             forward;
20:             drop;
21:             NoAction;
22:         }
23:         size = 32;
24:         default_action = drop();
25:         counters = my_direct_counter;
26: }
27: apply {
28:         if(hdr.ipv4.isValid()){
29:             forwarding.apply();
30:         }
31: }
```

forwarding

my_direct_counter

Key	Action	Action Data	Idx.	Count	
				Packets	Bytes
10.0.0.1	forward	egress port = 1	0	0	0
10.0.0.2	forward	egress port = 2	1	71	106,500
10.0.0.3	forward	egress port = 3	2	23	34,500
10.0.0.4	forward	egress port = 4	3	52	78,000
10.0.0.5	forward	egress port = 0	4	84	126,000
10.0.0.6	forward	egress port = 0	5	11	16,500
10.0.0.7	forward	egress port = 0	6	0	0
10.0.0.8	forward	egress port = 0	7	37	55,500
⋮	⋮	⋮	⋮	⋮	⋮
10.0.0.32	drop	egress port = 0	31	49	73,500

P4 Direct Counters

- The control plane can read the counters
- E.g., the following command for `MyIngress.my_direct_counter` indicates that the counter associated with entry 1 counted 4,921,063 bytes and 72,880 packets

P4 Indirect Counters

- Indirect counters are independent counters that can be referred to specific entries or group of entries in a match-action table
- E.g., there is a big table, but only a few counters are needed (few entries)
- The code must specify the number of independent counters (array size)

P4 Indirect Counters

- Example: instantiate a counter as an array of 3 elements, to count packets and bytes

```
 1: control MyIngress(inout header hdr,
 2:                    inout metadata meta,
 3:                    inout standard_metadata_t standard_metadata){
 4: counter(3,counterType.packets_and_bytes) my_indirect_counter;
 5:
 6: action forward(egressSpect_t port, bit<32> index){
 7:      standard_meadata.egress_spec = port;
 8:      my_indirect_counter.count(index);
 9: }
10: action drop(){
11:      mark_to_drop(standard_metadata);
12: }
13:
14: table forwarding {
15:      key = {
16:          hdr.ipv4.dstAddr : exact;
17:       }
18:      actions = {
19:          forward;
20:          drop;
21:          NoAction;
22:      }
23:      size = 32;
24:      default_action = drop();
25: }
26:
27: apply {
28:      if(hdr.ipv4.isValid()){
29:          forwarding.apply();
30:      }
31: }
```
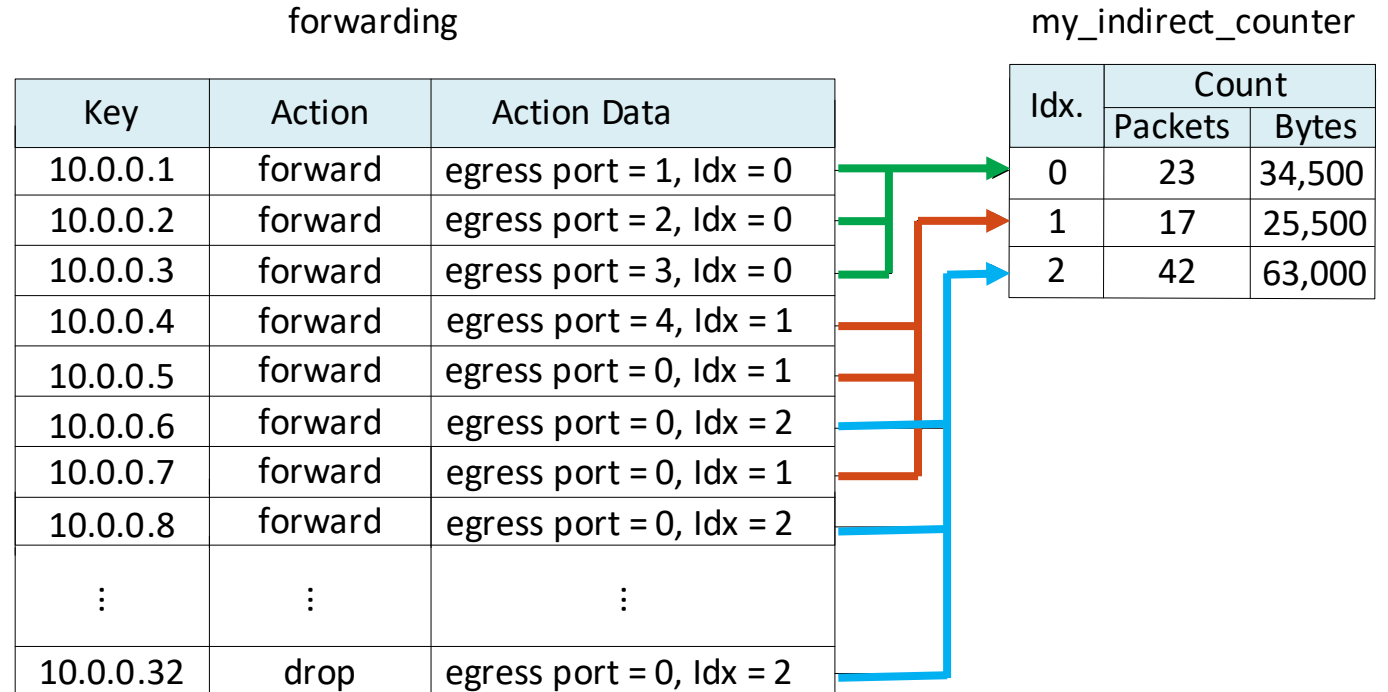
P4 Indirect Counters

- Example: instantiate a counter as an array of 3 elements, to count packets and bytes
- The `count` method is used to increment the value

```
1: control MyIngress(inout header hdr,
2:                    inout metadata meta,
3:                    inout standard_metadata_t standard_metadata){
3:
4: counter(3,counterType.packets_and_bytes) my_indirect_counter;
5:
6: action forward(egressSpect_t port, bit<32> index){
7:     standard meadata.egress spec = port;
8:     my_indirect_counter.count(index);
9: }
10: action drop(){
11:     mark_to_drop(standard_metadata);
12: }
13:
14: table forwarding {
15:     key = {
16:         hdr.ipv4.dstAddr : exact;
17:     }
18:     actions = {
19:         forward;
20:         drop;
21:         NoAction;
22:     }
23:     size = 32;
24:     default_action = drop();
25: }
26:
27: apply {
28:     if(hdr.ipv4.isValid()){
29:         forwarding.apply();
30:     }
31: }
```

P4 Indirect Counters

- Example: count packets/bytes routed by routes 1-3; routes 4, 5, 7; and routes 6, 8, 32
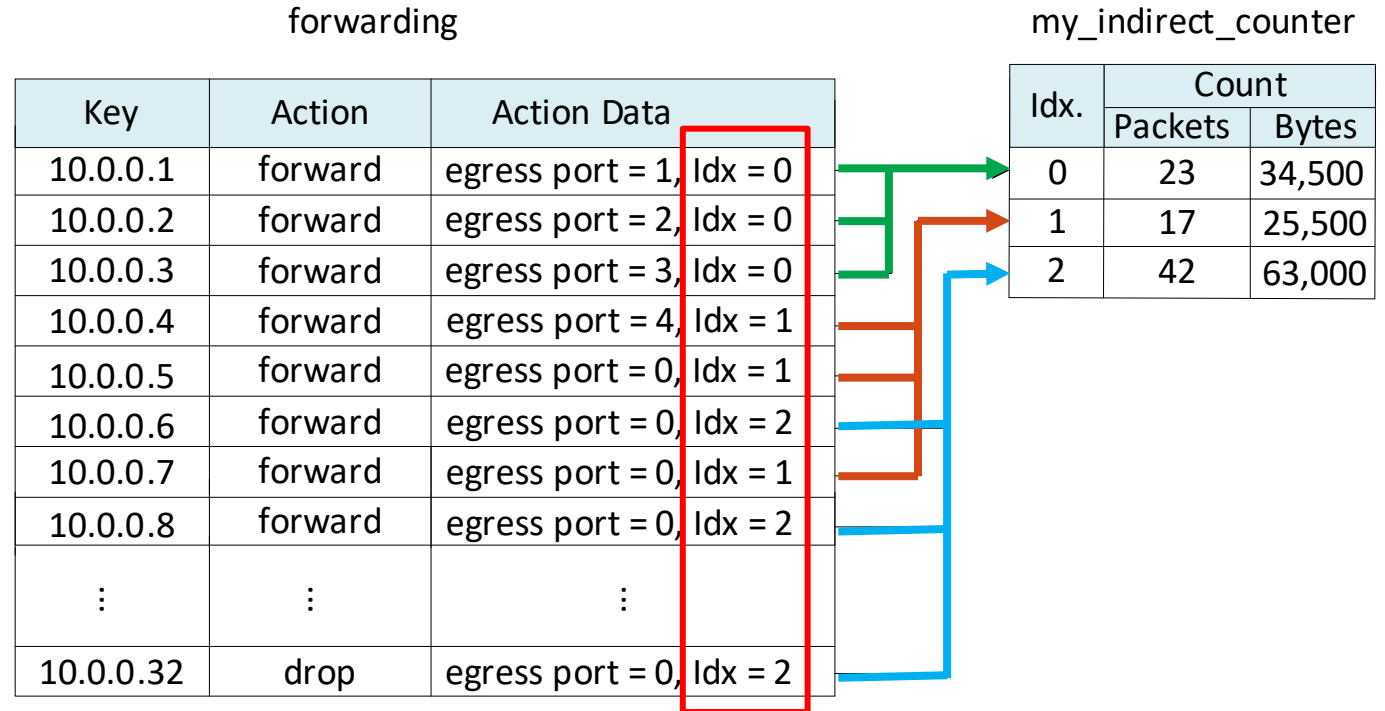
```
1: control MyIngress(inout header hdr,
2:                    inout metadata meta,
3:                    inout standard_metadata_t standard_metadata){
3:
4: counter(3,counterType.packets_and_bytes) my_indirect_counter;
5:
6: action forward(egressSpect_t port, bit<32> index){
7:         standard_meadata.egress_spec = port;
8:         my_indirect_counter.count(index);
9: }
10: action drop(){
11:         mark_to_drop(standard_metadata);
12: }
13:
14: table forwarding {
15:         key = {
16:             hdr.ipv4.dstAddr : exact;
17:         }
18:         actions = {
19:             forward;
20:             drop;
21:             NoAction;
22:         }
23:         size = 32;
24:         default_action = drop();
25: }
26:
27: apply {
28:         if(hdr.ipv4.isValid()){
29:             forwarding.apply();
30:         }
31: }
```

forwarding

Key	Action	Action Data
10.0.0.1	forward	egress port = 1, Idx = 0
10.0.0.2	forward	egress port = 2, Idx = 0
10.0.0.3	forward	egress port = 3, Idx = 0
10.0.0.4	forward	egress port = 4, Idx = 1
10.0.0.5	forward	egress port = 0, Idx = 1
10.0.0.6	forward	egress port = 0, Idx = 2
10.0.0.7	forward	egress port = 0, Idx = 1
10.0.0.8	forward	egress port = 0, Idx = 2
⋮	⋮	⋮
10.0.0.32	drop	egress port = 0, Idx = 2

my_indirect_counter

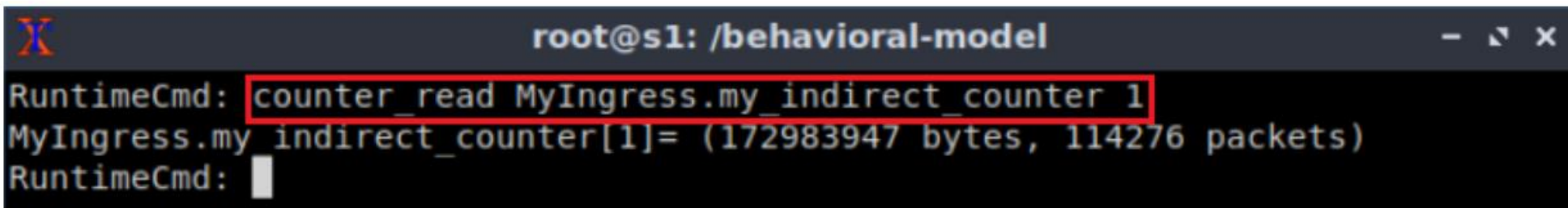| Idx.	Count	
	Packets	Bytes
0	23	34,500
1	17	25,500
2	42	63,000

P4 Indirect Counters

- Example: count packets/bytes routed by routes 1-3; routes 4, 5, 7; and routes 6, 8, 32
- Note that the index used to increment the counter is retrieved from the action data
  - The index can also be computed, as needed by the programmer

```
1: control MyIngress(inout header hdr,
2:                    inout metadata meta,
3:                    inout standard_metadata_t standard_metadata){
3:
4: counter(3,counterType.packets_and_bytes) my_indirect_counter;
5:
6: action forward(egressSpect_t port, bit<32> index){
7:        standard meadata.egress spec = port;
8:        my_indirect_counter.count(index);
9: }
10: action drop(){
11:        mark_to_drop(standard_metadata);
12: }
13:
14: table forwarding {
15:        key = {
16:            hdr.ipv4.dstAddr : exact;
17:        }
18:        actions = {
19:            forward;
20:            drop;
21:            NoAction;
22:        }
23:        size = 32;
24:        default_action = drop();
25: }
26:
27: apply {
28:        if(hdr.ipv4.isValid()){
29:            forwarding.apply();
30:        }
31: }
```

forwarding

Key	Action	Action Data
10.0.0.1	forward	egress port = 1, Idx = 0
10.0.0.2	forward	egress port = 2, Idx = 0
10.0.0.3	forward	egress port = 3, Idx = 0
10.0.0.4	forward	egress port = 4, Idx = 1
10.0.0.5	forward	egress port = 0, Idx = 1
10.0.0.6	forward	egress port = 0, Idx = 2
10.0.0.7	forward	egress port = 0, Idx = 1
10.0.0.8	forward	egress port = 0, Idx = 2
⋮	⋮	⋮
10.0.0.32	drop	egress port = 0, Idx = 2

my_indirect_counter

Idx.	Count	
	Packets	Bytes
0	23	34,500
1	17	25,500
2	42	63,000

P4 Indirect Counters

- The control plane can read the counters
- E.g., the following command for `MyIngress.my_indirect_counter` indicates that the counter associated with entry 1 counted 172,983,947bytes and 114,276 packets