Counters and Registers in P4

Fábio L. Verdi
Leandro Almeida

Advanced Topics in Computer Networks

19 de outubro de 2022

Slides based on the presentation done by Vladimir Gurevich (Intel Principal Engineer) at the P4 Developers Day (P4 D2), May 2017

Agenda

- Goals
- Types of objects in P4
- Counters
- Registers
- Control Plane access to counters and registers
- Other types of externs
- Task

Goals of this class

- Present the fundamentals about the persistence (state) of information in the data plane;
- Present the concepts of counters and registers in the context of the P4 language;
- Demonstrate use cases of counters and registers in P4;
- Carry out a practical activity contemplating the concepts of accountants and registers.

Types of objects in P4

Stateless objects

- Restarted for each packet;
- Exemples: variables (metadata), headers, packet_in, packet_out.

Stateful objects

- Keep the state among packets;
- Exemples: Tables, Externs (Counters, Meters, Registers, ...)

Types of objects

Stateful Objects

Tabela 1: Operations - Tables and Externs (Counters e Registers)

Object	Data Plane		Control Plane	
	Read	Write	Read	Write
Table	apply()	–	✓	✓
Counter	–	count()	✓	✓
Register	read()	write()	✓	✓

Counters

Basic concepts

- Are objects (externs) defined by the architecture used for incremental counting of packets, bytes, or both;
- Are allocated similarly to arrays in general-purpose languages;
- Cannot be read from the data plane;
- Can be read via the control plane.

Types of counters int the V1model

- packets
- bytes
- packets_and_bytes

Counters

```
3   control MyIngress(inout headers hdr,
4                     inout metadata meta,
5                     inout standard_metadata_t standard_metadata) {
6
7       /* Declaração do contador */
8       counter(24, CounterType.packets) forwardingPkts; // Contador de pacotes encaminhados por porta
9
10      action ipv4_forward(macAddr_v dstAddr, egressSpec_v port) {
11          standard_metadata.egress_spec = port;
12          hdr.ethernet.srcAddr = hdr.ethernet.dstAddr;
13          hdr.ethernet.dstAddr = dstAddr;
14          hdr.ipv4.ttl = hdr.ipv4.ttl - 1;
15          forwardingPkts.count((bit<32>)standard_metadata.egress_spec);
16      }
17
18      table ipv4_lpm {
19          key = {
20              hdr.ipv4.dstAddr: lpm;
21          }
22          actions = {
23              ipv4_forward;
24              drop;
25              NoAction;
26          }
27          size = 1024;
28          default_action = NoAction();
29      }
```

Figura 1: Example of a Counter in P4.

Registers

Basic concepts

- Are objects (externs) defined by the architecture used to store data;
- Are allocated similarly to arrays in general-purpose languages;
- Can be read/written in the data plane;
- Can be read/written via the control plane.

Registers

```
register<bit<48>>(16384) last_seen;

action get_inter_packet_gap(out bit<48> interval,
                            bit<14> flow_id)
{
    bit<48> last_pkt_ts;

    /* Get the time the previous packet was seen */
    last_seen.read((bit<32>) flow_id,
                   last_pkt_ts);

    /* Calculate the time interval */
    interval = standard_metadata.ingress_global_timestamp -
               last_pkt_ts;

    /* Update the register with the new timestamp */
    last_seen.write((bit<32>) flow_id,
                    standard_metadata.ingress_global_timestamp);
}
```

Figura 2: Example of a Register in P4.

Access via control plane

Runtime control API - CLI

- Use it through the command `simple_switch_CLI`

Counter

```
RuntimeCmd:  counter_read MyIngress.c 1
MyIngress.c[1]= BmCounterValue(packets=1, bytes=59)
```

Register

```
RuntimeCmd:  register_read MyIngress.reg 1
MyIngress.reg[1]= 15
```

Other types of externs

- Meters (Traffic color);
- Direct Counters (extending Tables);
- Stateless externs (hash, random, Checksum16)

Monitoring task

Task 1 - Counter

Rewrite the basic forwarding activity, adding a counter (positioned in Ingress) of packets and bytes per input port.

Task 2 - Register

Rewrite the basic forwarding activity, adding a register (positioned in Egress) to store queue occupancy per output port. (Note: use the metadata `standard_metadata.enq_qdepth`).

Happy Hacking! ☺

Doubts?

verdi@ufscar.br
leandro.almeida@estudante.ufscar.br