P4 Expert Roundtable Series
April 28-29, 2020
Hosted by: ONF

Sponsored By
Google intel STORDIS The Open Networking Expert

Realizing Source Routed Multicast Using Mellanox’s Programmable Hardware Switches

Matty Kadosh[1], Yonatan Piasetzky[1], Barak Gafni[1], Lalith Suresh[2], Muhammad Shahbaz[3], Sujata Banerjee[2]

[1]*Mellanox*, [2]*VMware*, [3]*Stanford*

Group Communication in Public Clouds

10,000s of tenants

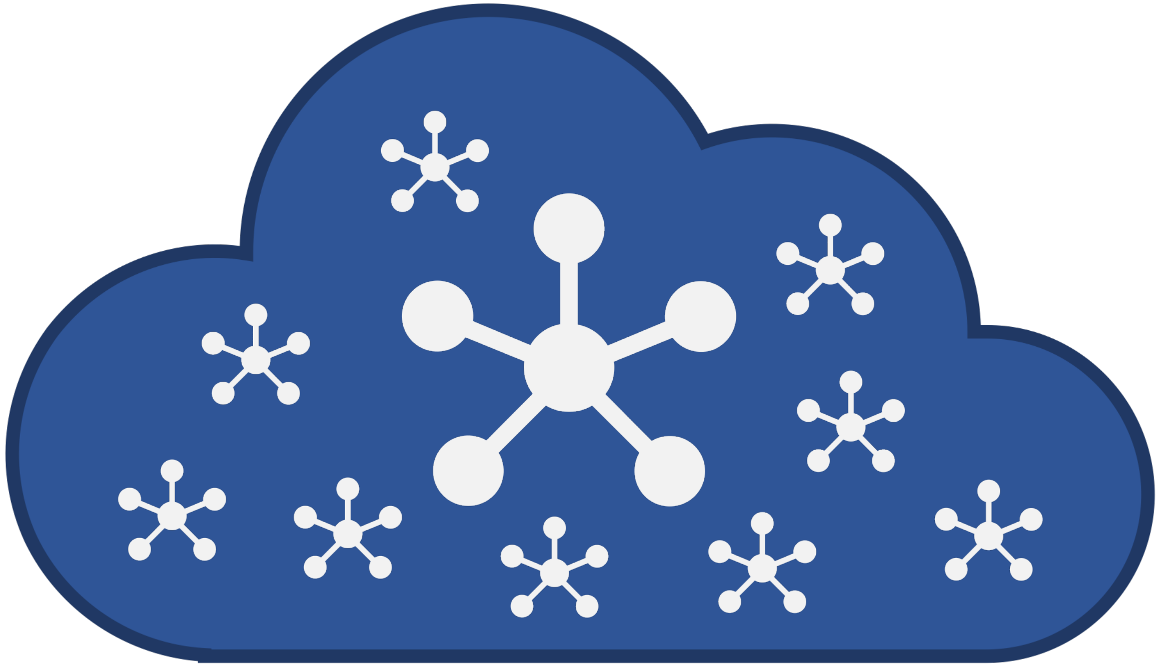
100s of workloads

Replication
for databases and state machines

Publish-Subscribe
like ZeroMQ and RabbitMQ

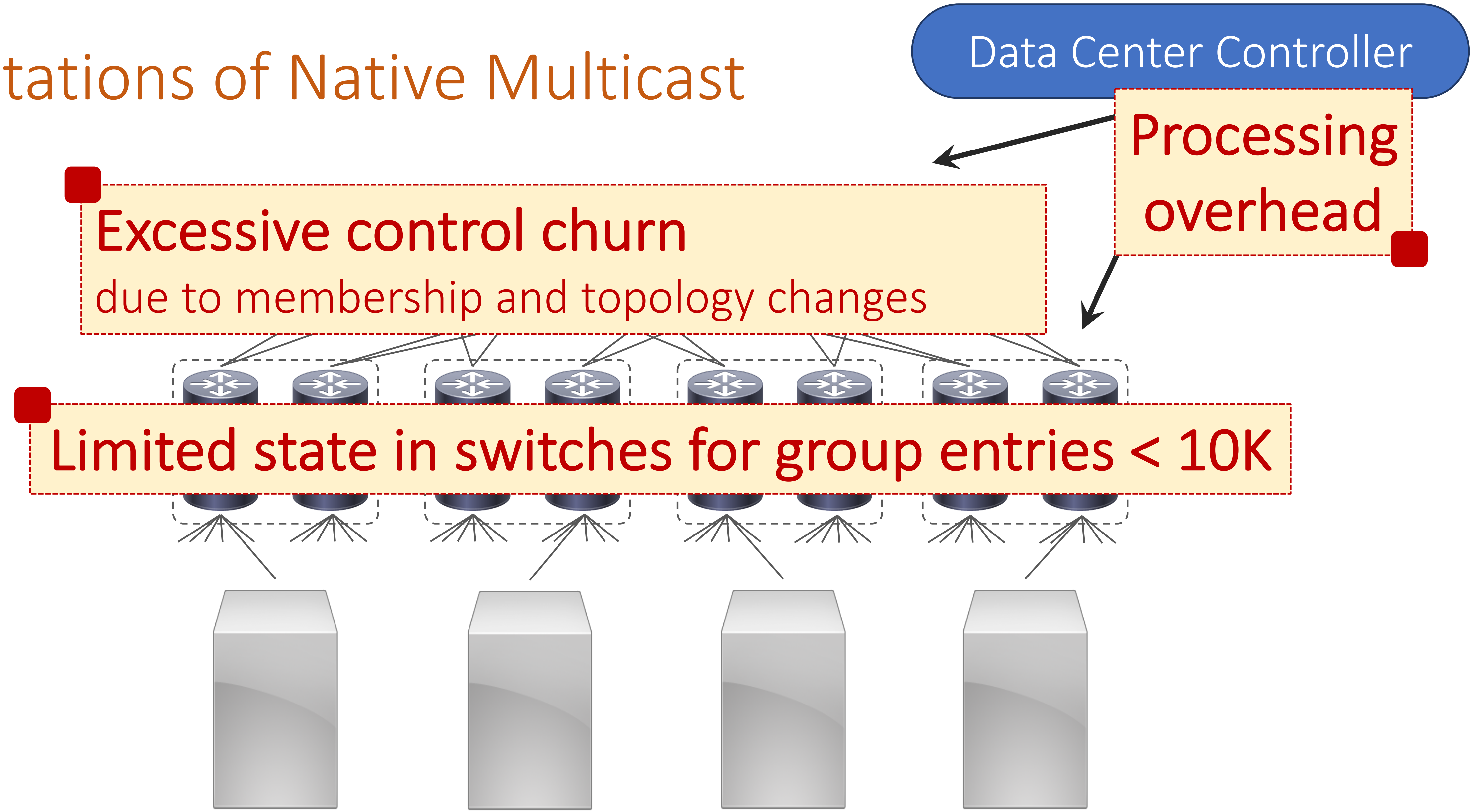
Infrastructure Apps
like VMware NSX and OpenStack

Millions of distinct **group** communications

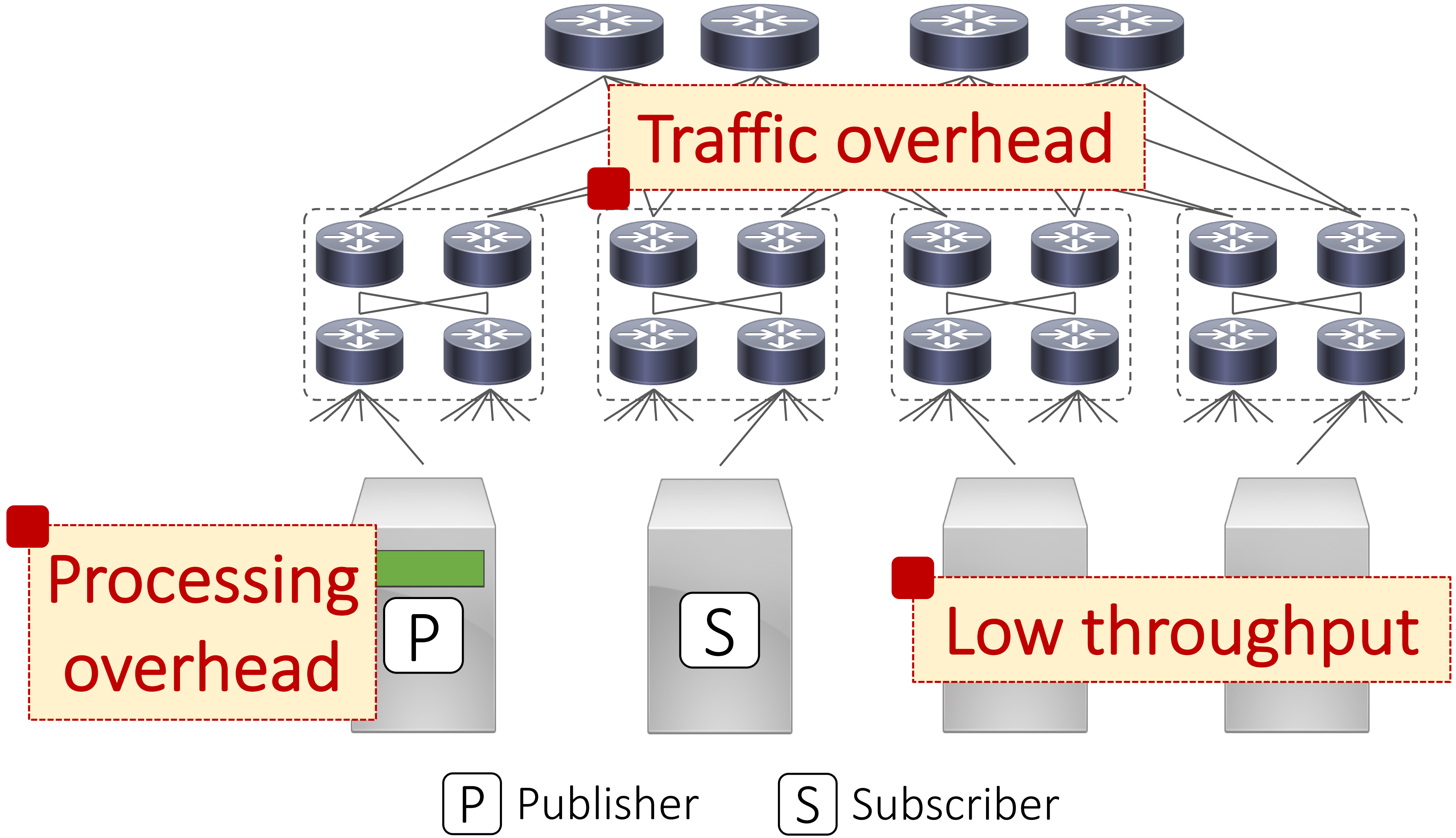


amazon Google Microsoft

Limitations of Native Multicast

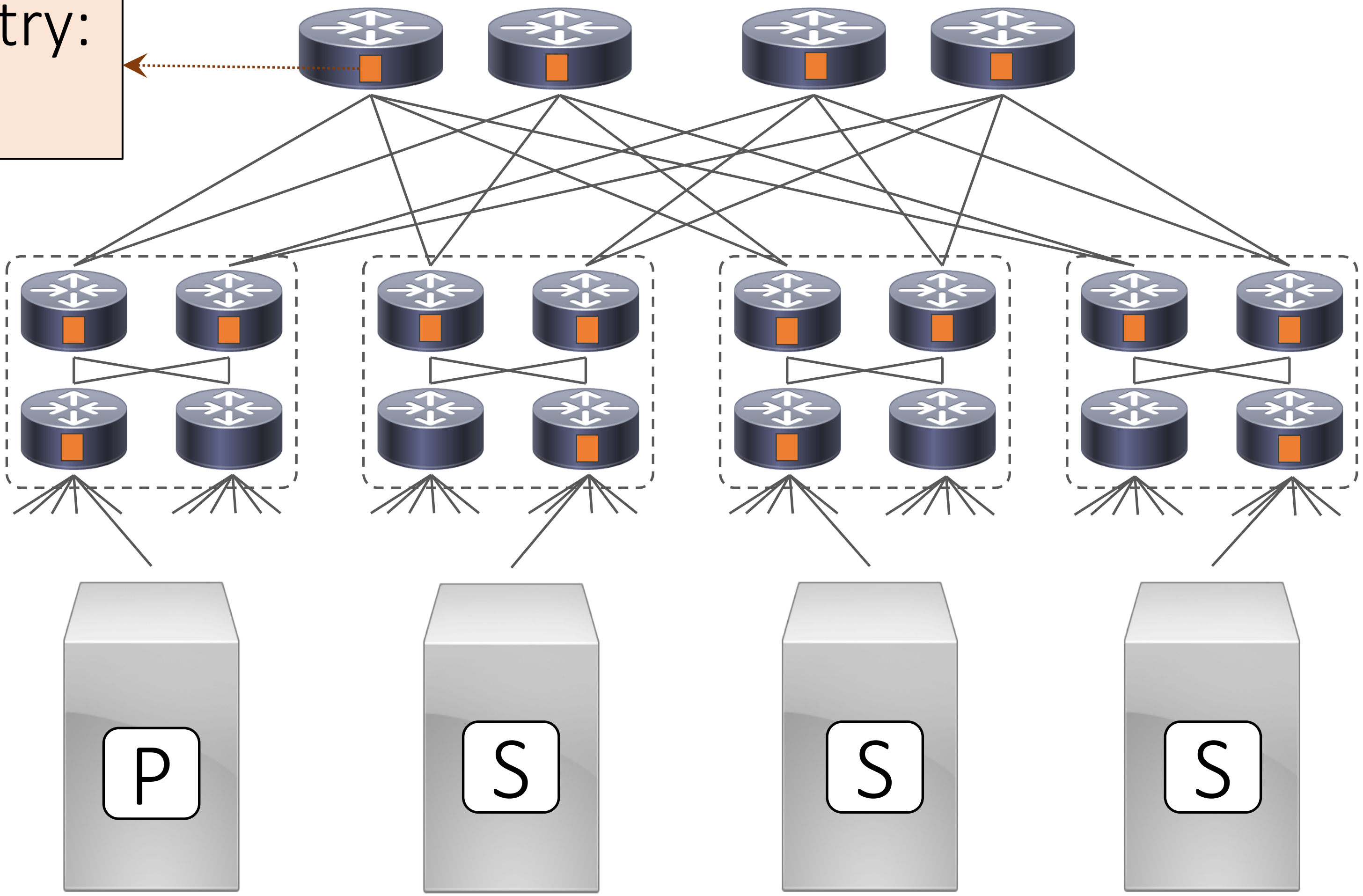


Restricted to Application-Level Multicast



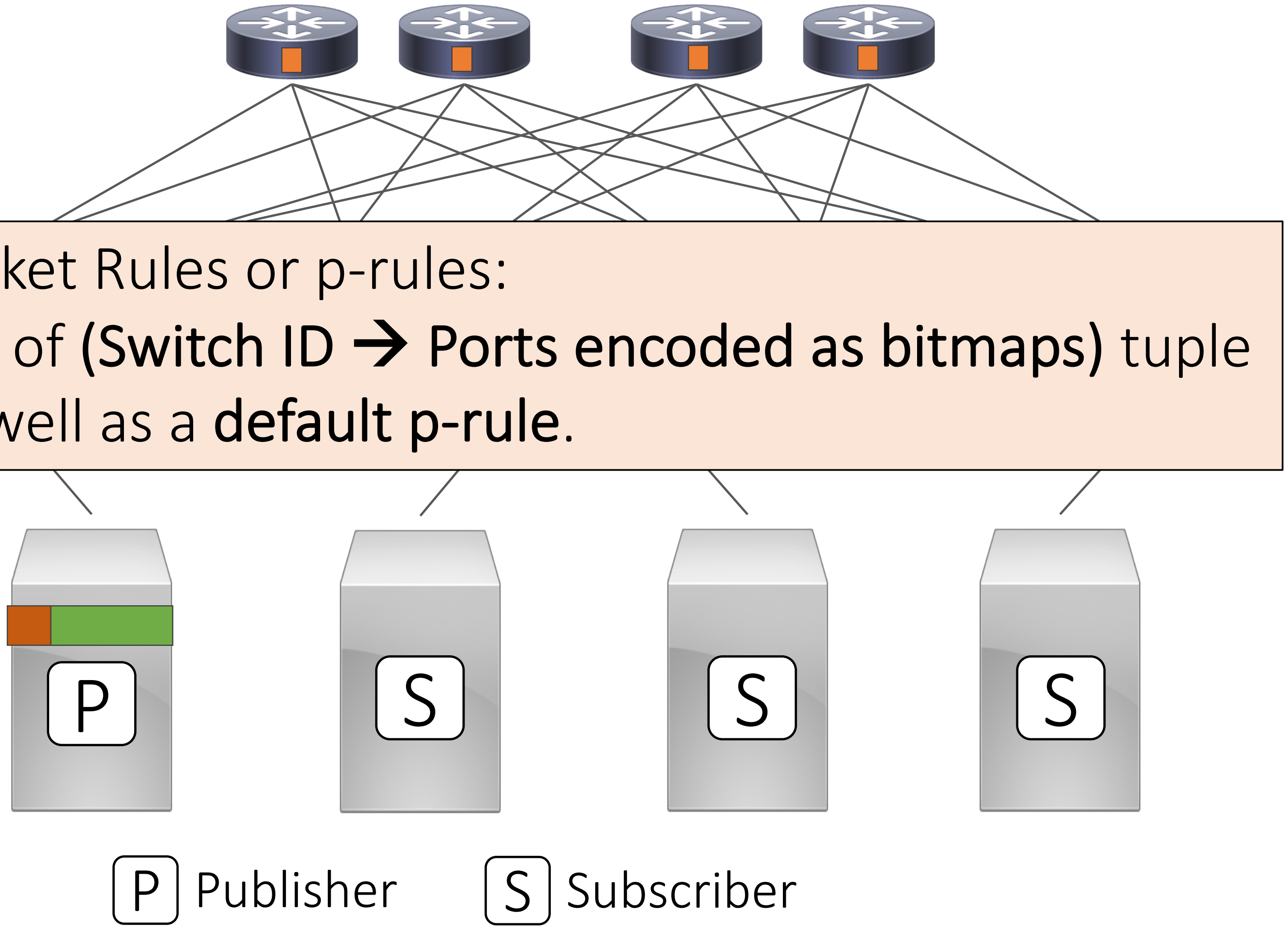
Elmo: Source Routed Multicast → SIGCOMM'19

Group Table Entry:
ID → Ports



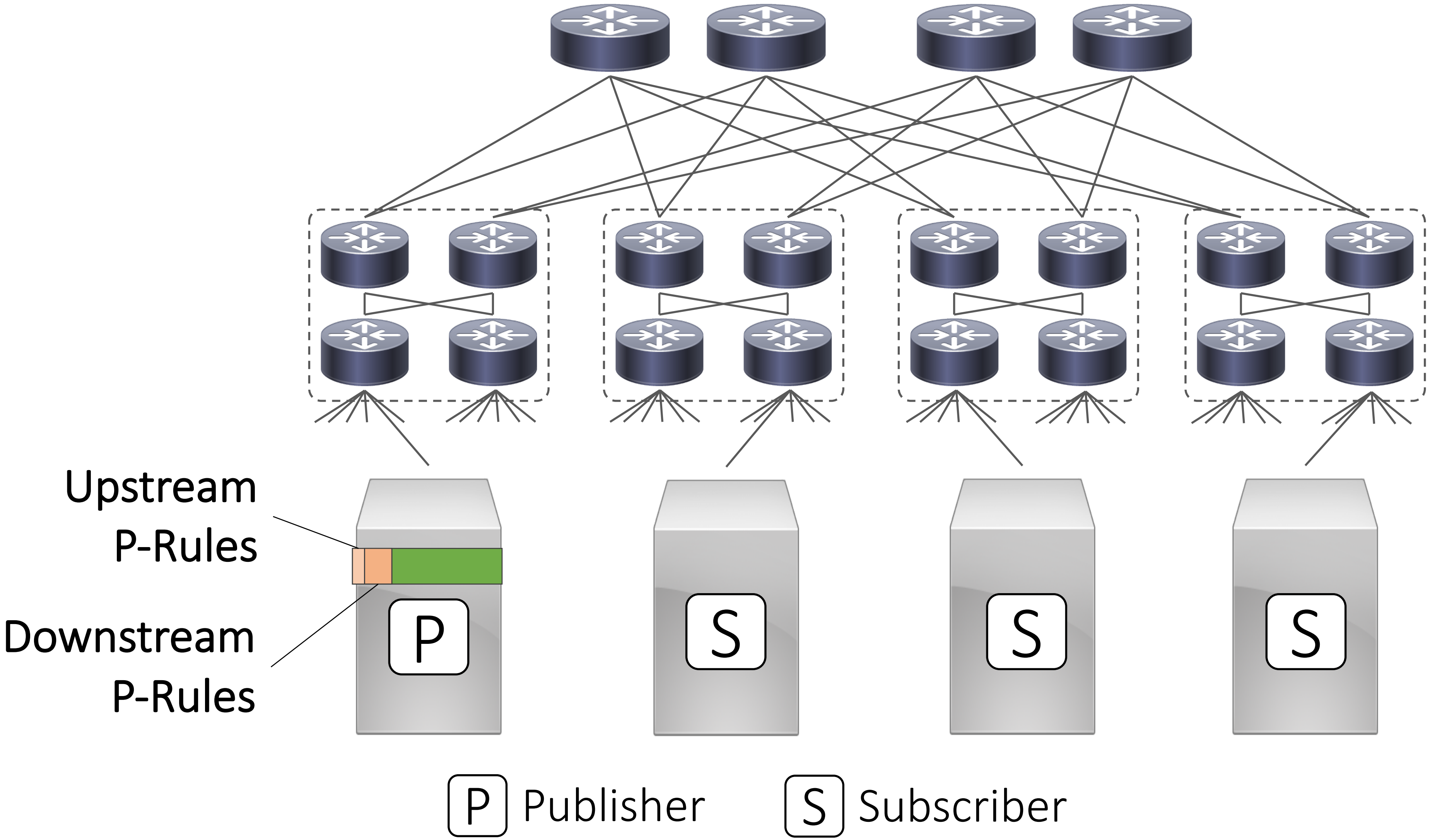
P Publisher S Subscriber

Elmo: Source Routed Multicast

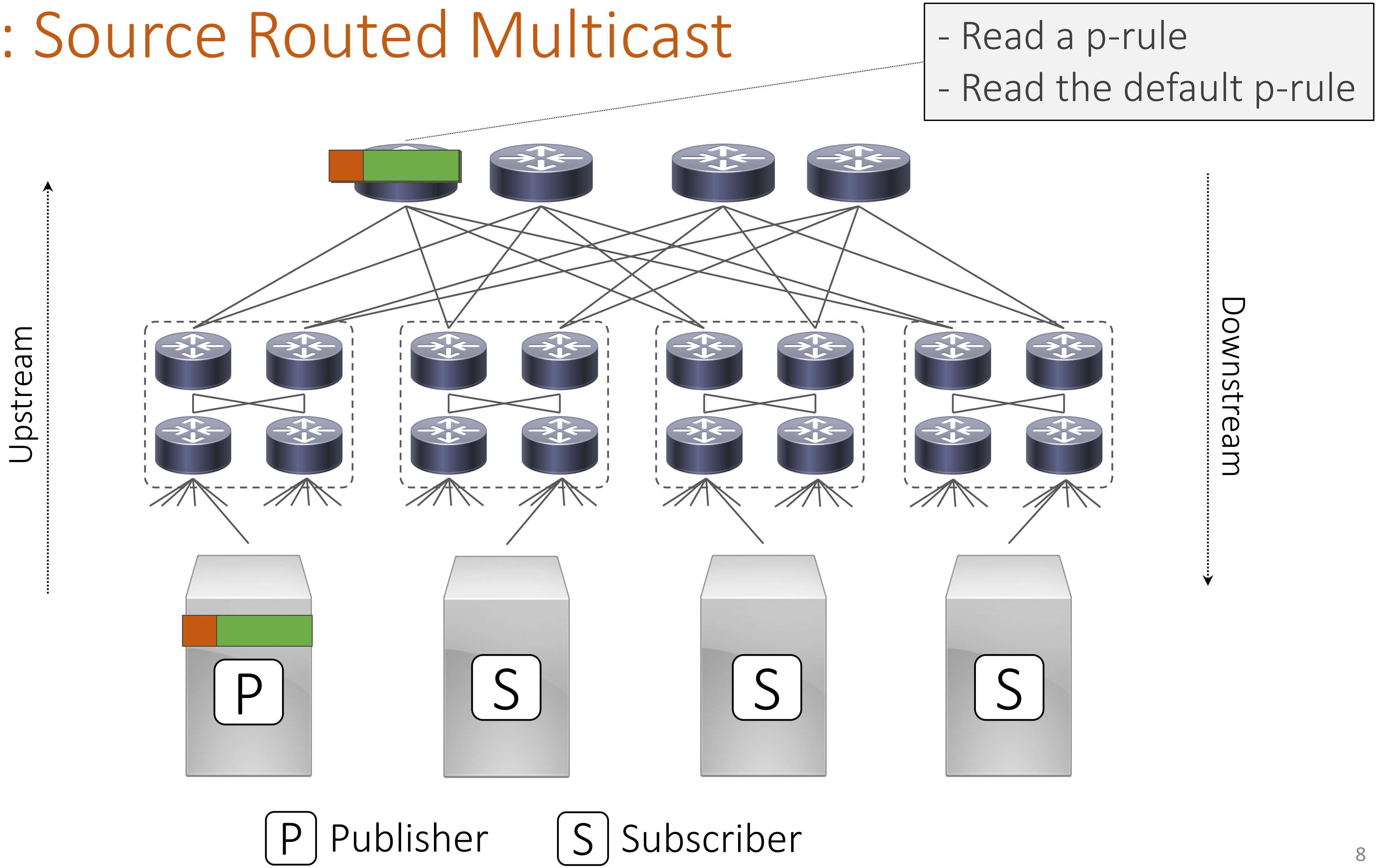


Packet Rules or p-rules:
List of **(Switch ID → Ports encoded as bitmaps)** tuple as well as a **default p-rule**.

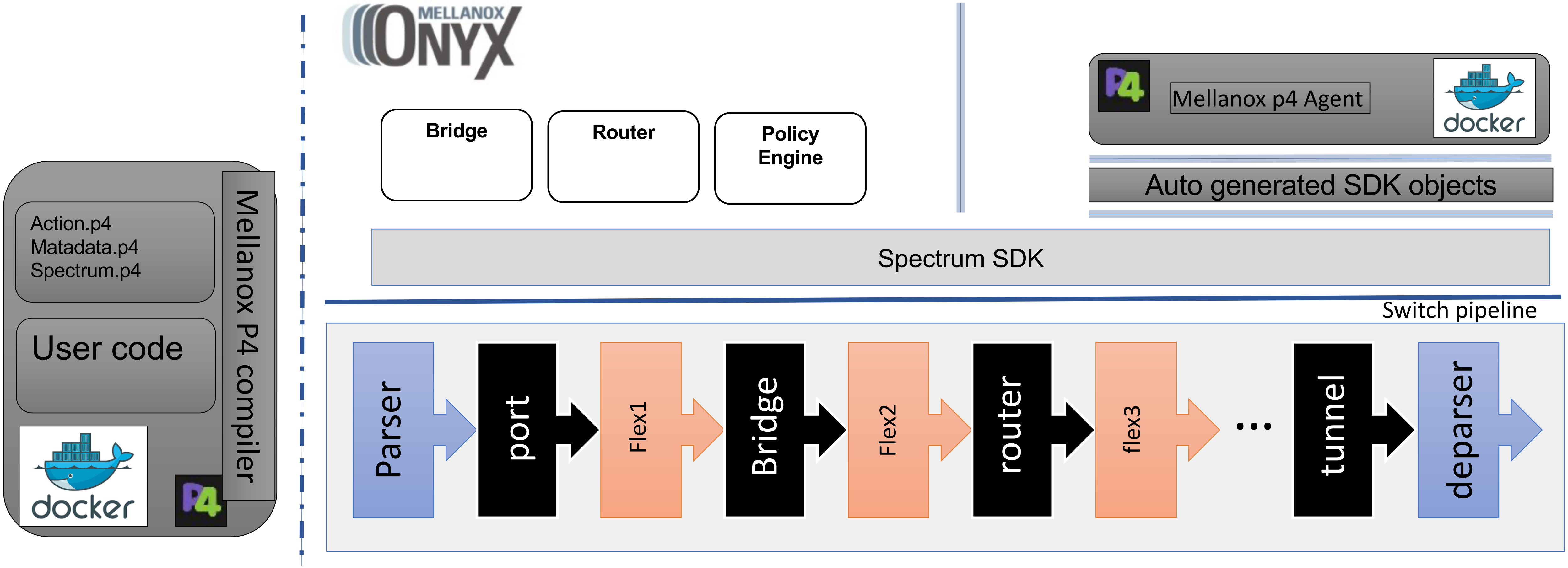
Elmo: Source Routed Multicast



Elmo: Source Routed Multicast

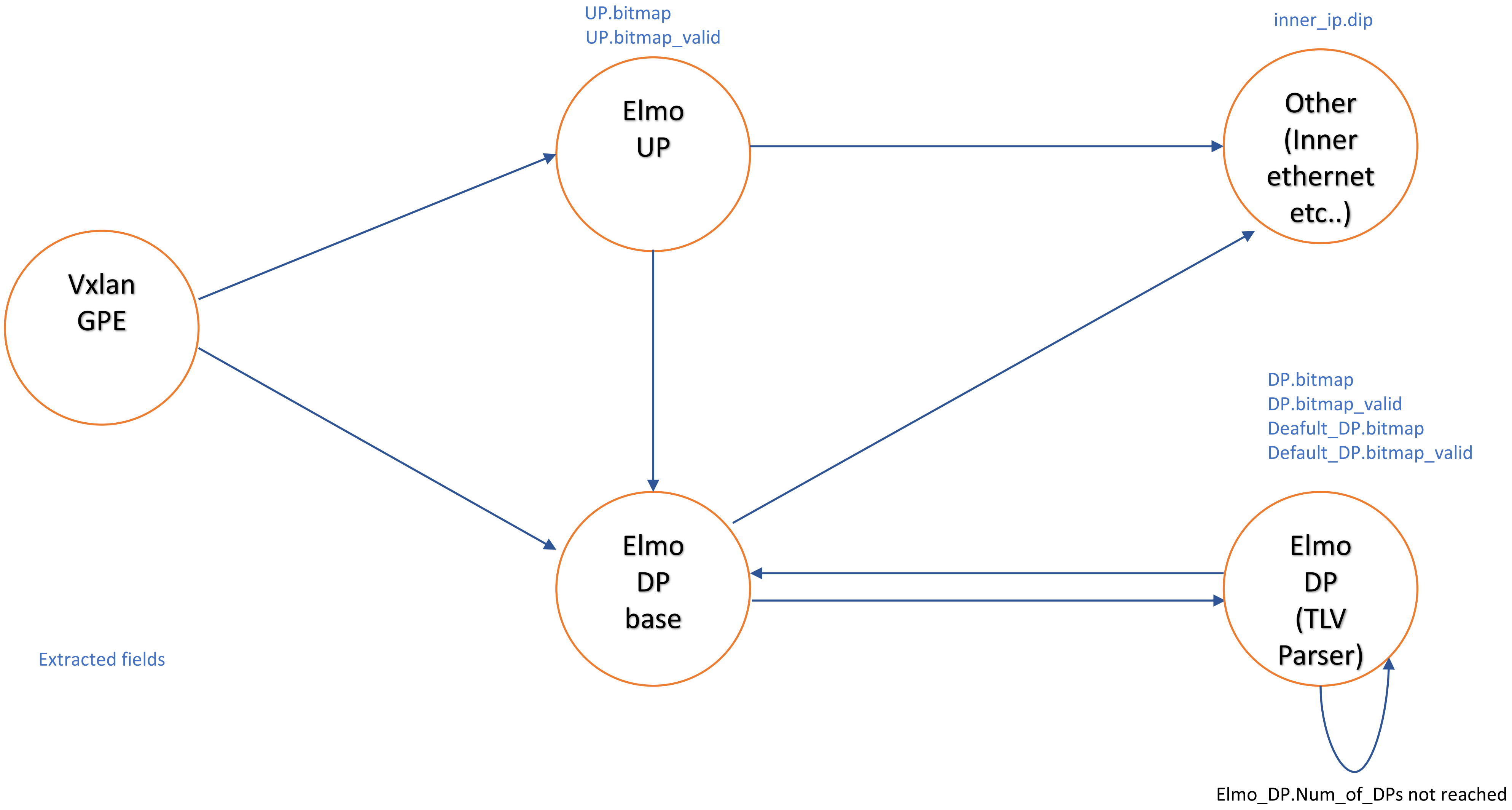


Mellanox Programmable switch model



- Hybrid – Integration between legacy (switch router) and programmable pipeline
- NOS (ONYX / SONiC) and user applications run in parallel

Parser state machine



Options parser

- Options are common among network protocols (IPv4, TCP, etc..)
- Options follows some common structure
  - Base header has a known length
  - Total header length (computed)
  - Total options length
  - Options are built in a TLV fashion:
    - Type (self-indicator)
    - Length (some granularity)
    - Type and Length fields are fixed
  - This structure mainly exists to support unknown options
  - State transition is defined in the base header
- In Elmo:
  - Downstream P-rules are options
  - Unknown switch ID
  - Default p-rule – common Switch ID

```
Elmo downstream header (DP base):
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Num of DPs  |   next_proto  |            Reserved           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Elmo downstream P-Rule (DP):
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Reserved   |                   Switch ID                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Bitmap                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Single switch Functionality

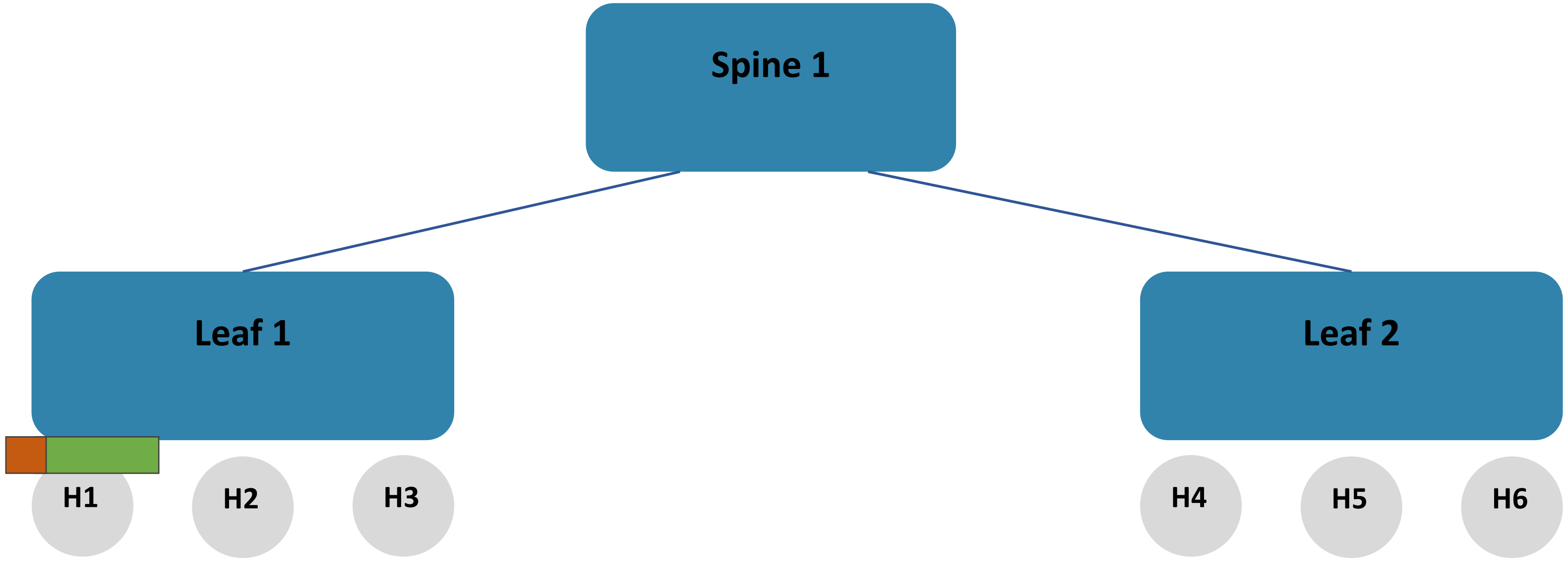
- Upstream bitmap
- Downstream bitmap
- Default p-rule
  - Increase scale on the expense of excess traffic
- Normal forwarding by the legacy pipeline

```
action set_egress_ports_dp() {
    metadata.egress_ports = headers.elmo_downstream_p_rule.bitmap;
}
```

```
table initial_bitmap_table {
    key = {
        headers.elmo_upstream_p_rule.isValid()   : exact;
        headers.elmo_downstream_p_rule.isValid() : ternary;
    }
```

```
apply
{
    if (initial_bitmap_table.apply().miss) {
        if (s_rule_table.apply().miss) {
            default_bitmap_table.apply();
        }
    }
}
```

Demo



- Mcast groups:

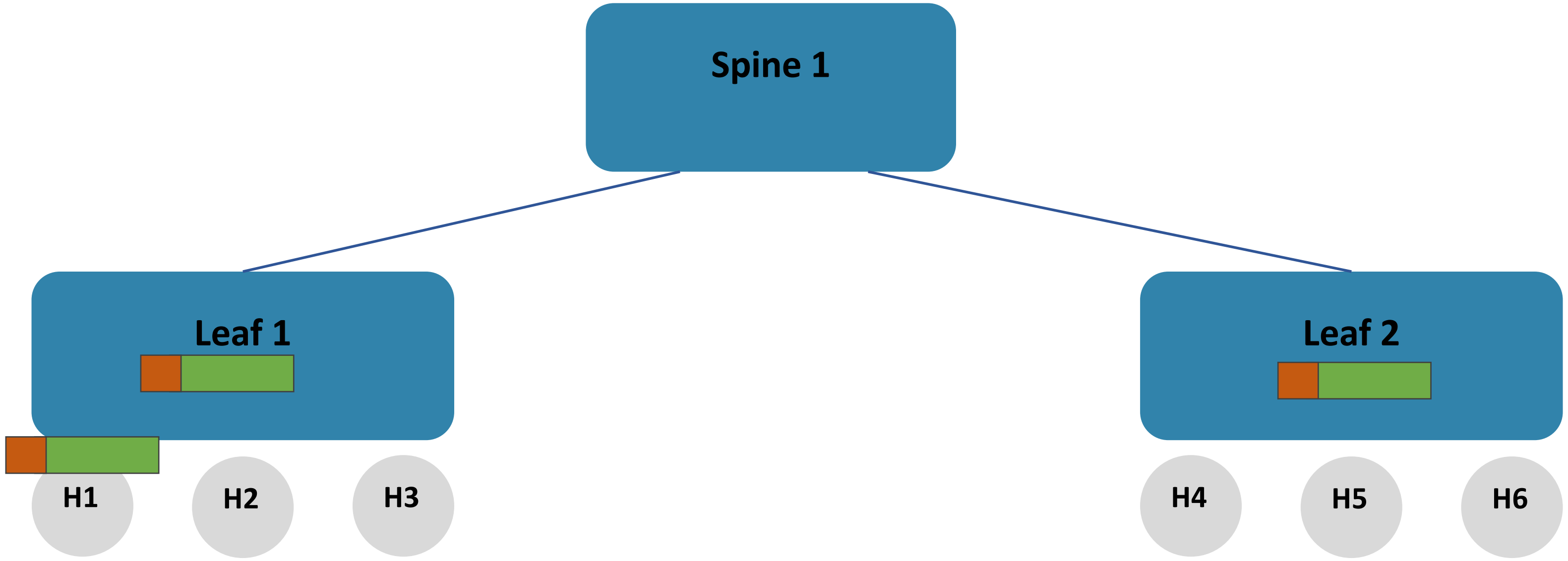
1. H1 transmits to H2:
   UP bitmap : [000, 010]
   no DP header

2. H1 transmits to H3, H4 and H6:
   UP bitmap: [001, 001]
   DP: [S1: 001, L2: 101]

3. H1 transmits to H2, H4, H5:
   UP bitmap: [001, 010]
   DP:[S1: 001, L2: 110]

4. H1 transmits to H5, H6:
   UP bitmap: [001, 000]
   DP: [S1: 001, default: 011]

Demo



- Mcast groups:

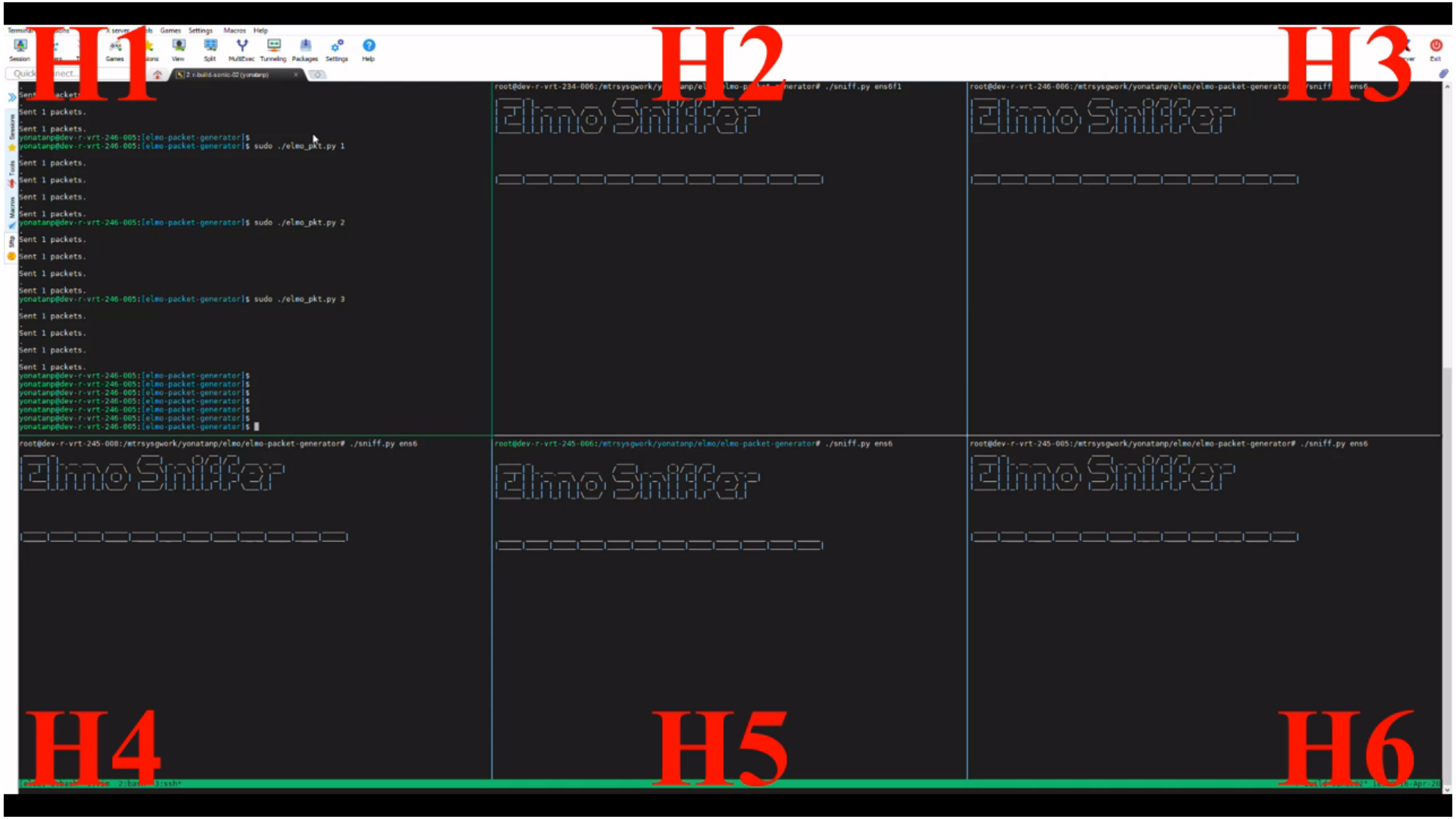
1. H1 transmits to H2:
   UP bitmap : [000, 010]
   no DP header

2. H1 transmits to H3, H4 and H6:
   UP bitmap: [001, 001]
   DP: [S1: 001, L2: 101]

3. H1 transmits to H2, H4, H5:
   UP bitmap: [001, 010]
   DP:[S1: 001, L2: 110]

4. H1 transmits to H5, H6:
   UP bitmap: [001, 000]
   DP: [S1: 001, default: 011]

Demo



H1
H2
H3
H4
H5
H6
Elmo Sniffer
sudo ./elmo_pkt.py 1
sudo ./elmo_pkt.py 2
sudo ./elmo_pkt.py 3
Sent 1 packets.
./sniff.py ens6

Challenges

- In the following slides, we'll share our experience from this work.
- Challenges encountered during this work:
  - Multicast
  - Options parsing
  - Extraction

Multicast

- Multicast is not handled by the PSA model (extern)
- Hard for stateless switch multicast
- This work - directly expose MC bitmap to the dataplane:
  - metadata.egress_ports = headers.elmo_downstream_default_p_rule.bitmap;
- Multicast group table can be easily supported
- Hybrid architecture - support non-physical ports as well (e.g., router interface etc..)

Options parsing

- Options current implementation in P4
  - possible but not trivial
  - Not easily offloadable
- Common use case
  - Worthwhile to have standard fashion of defining
  - Easily HW offloaded by the different vendors.
- Build a sub-parser prototype which follows the observed structure

Extractions

- Extract methods act on entire headers –consumes them and advances the cursor
- It is further assumed that HW will extract all the fields of the accepted header.
  - Could be costly
- What if you require a subset of the fielc
- May prevent HW optimizing by selectiv
  - Dynamically loaded control
- Advanced field extraction features like v

```
Elmo downstream header (DP base):
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Num of DPs  |   next_proto  |            Reserved           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Elmo downstream P-Rule (DP):
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Reserved   |                   Switch ID                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Bitmap                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Extraction - proposal

- We implemented field extraction primitives on our architecture:
  - Void extract_field<T>(out T headerLvalue.field);
  - void extract_field<T>(out T headerLvalue.field, in bit<32> variableFieldOffset);
  - void extract_field<T>(out T headerLvalue.field, in bit<32> variableFieldSizeInBits, in bit<32> variableFieldOffset);

  - Extract a single field and advance the cursor,
    - Adds to current header primitives (not replace)

  - Useful also for:
    - variable offset fields
    - more than one variable length field in a header

- Another option - Usage analysis in the compiler backend
  - Sufficient for monolithic P4 executables

  - Problematic for target architectures which allow dynamic insertion of control pipelines (which share the same parser)

Conclusions

- Elmo compactly encodes multicast policy inside packets
- Designed for multi-tenant data centers scales
- Demonstrated, for the first time, Elmo implementation with wire speed performance using hybrid programmable dataplane
- All legacy forwarding and control plane is intact

Expert Roundtable Series
April 28-29, 2020

Hosted by:
ONF

Sponsored By

Google intel STORDIS
The Open Networking Expert

Thank You

Matty Kadosh (Mellanox), *mattyk@mellanox.com*
Lalith Suresh (VMware), lsuresh@vmware.com

Acknowledgment

Jen Rexford[1]
Nick Feamster[2]
Ori Rottenstreich[3, 5]
Mukesh Hira[4], Mihai Budiu[4], Ben Pfaff[4]
Alan Lo[5], Aviv Kfir[5], Jose Yallouz[5]
[1] Princeton [2] U. Chicago [3] Technion [4] VMware [5] Mellanox