Lecture 11: In-Network Computing

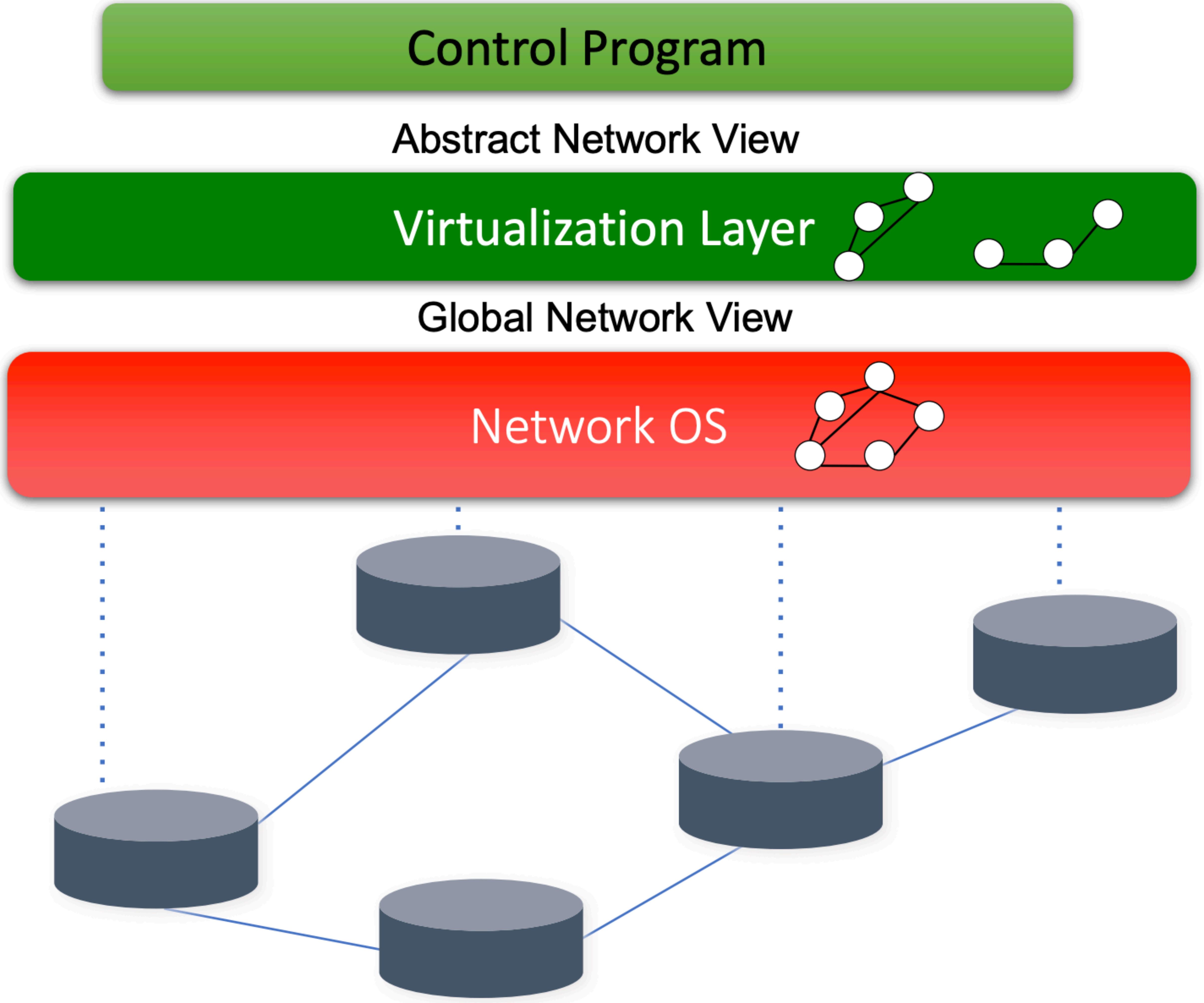
CS 234 / NetSys 210: Advanced Computer Networks

Sangeetha Abdu Jyothi

UCIrvine

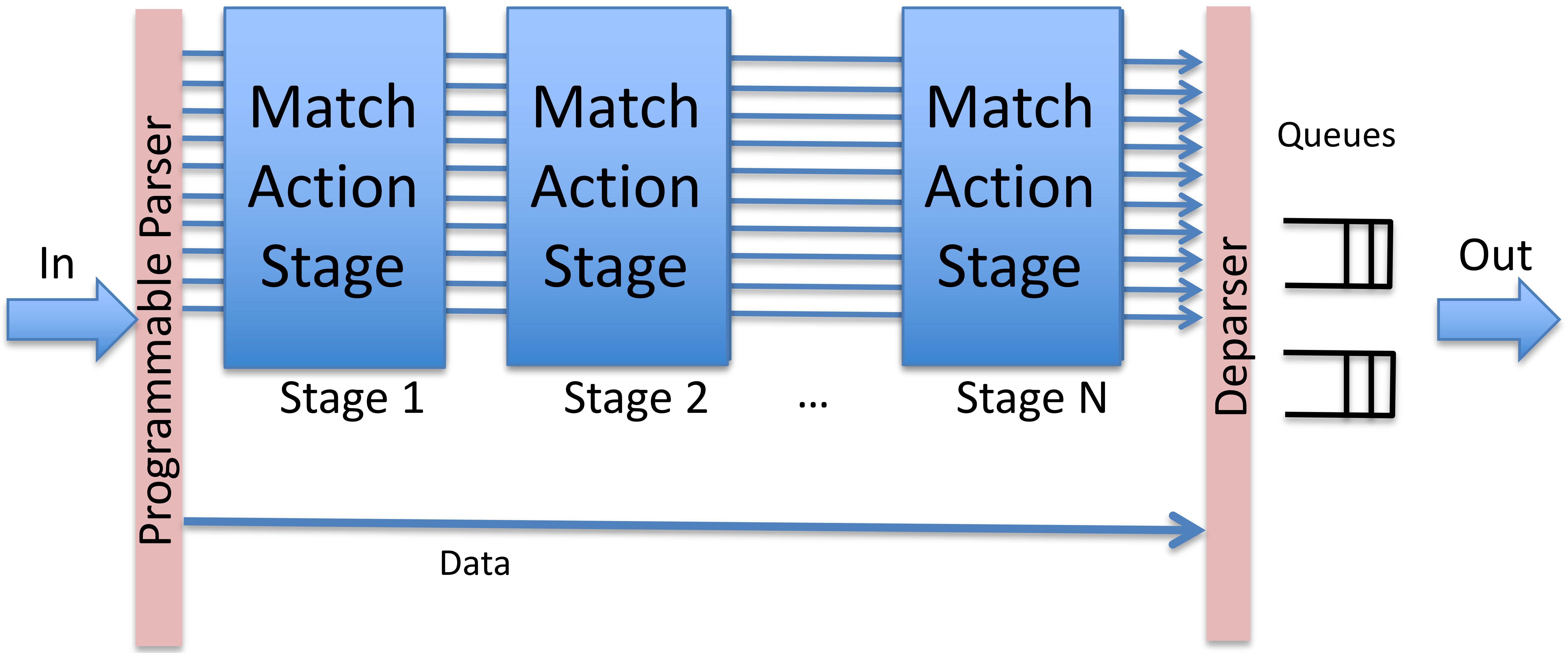
This lecture uses material from Muhammad Shahbaz's talk, Radhika Mittal's CS598HPN and SwitchML talk

Recap: SDN



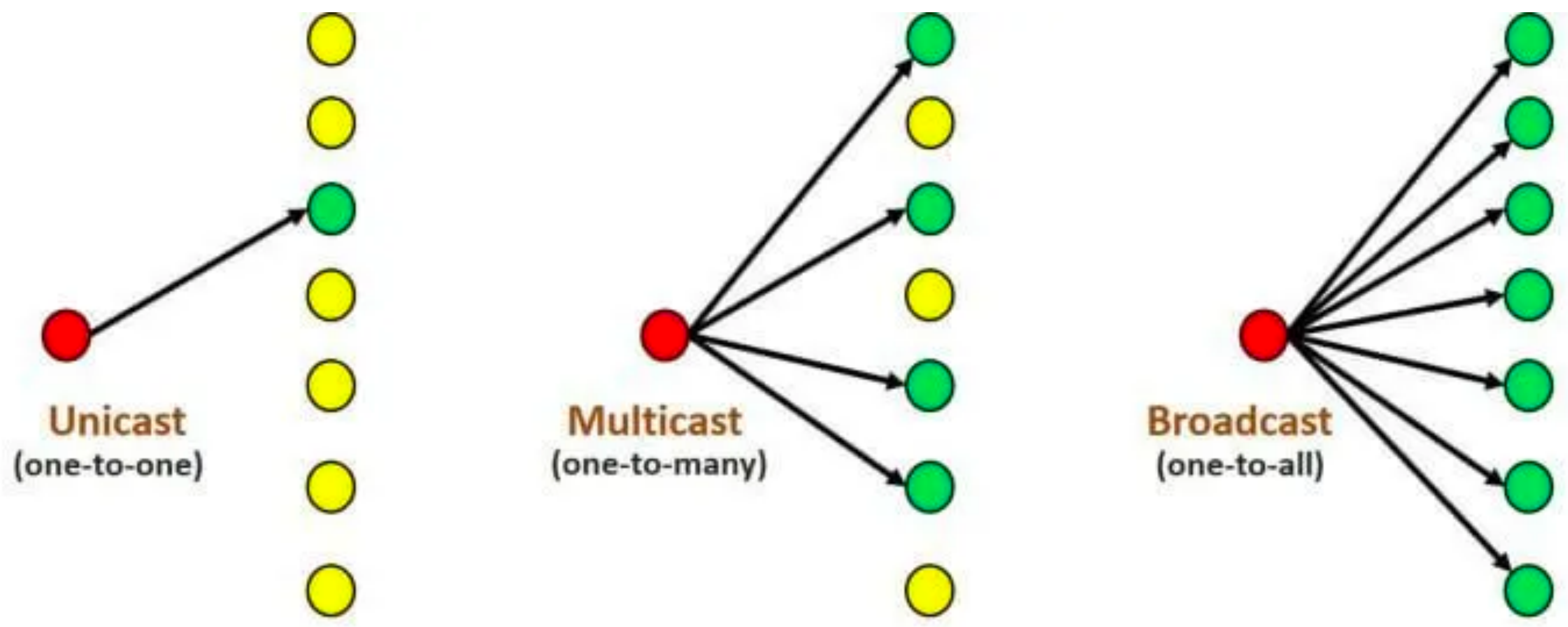
Control Program
Abstract Network View
Virtualization Layer
Global Network View
Network OS

Recap: Programmable Networking Hardware

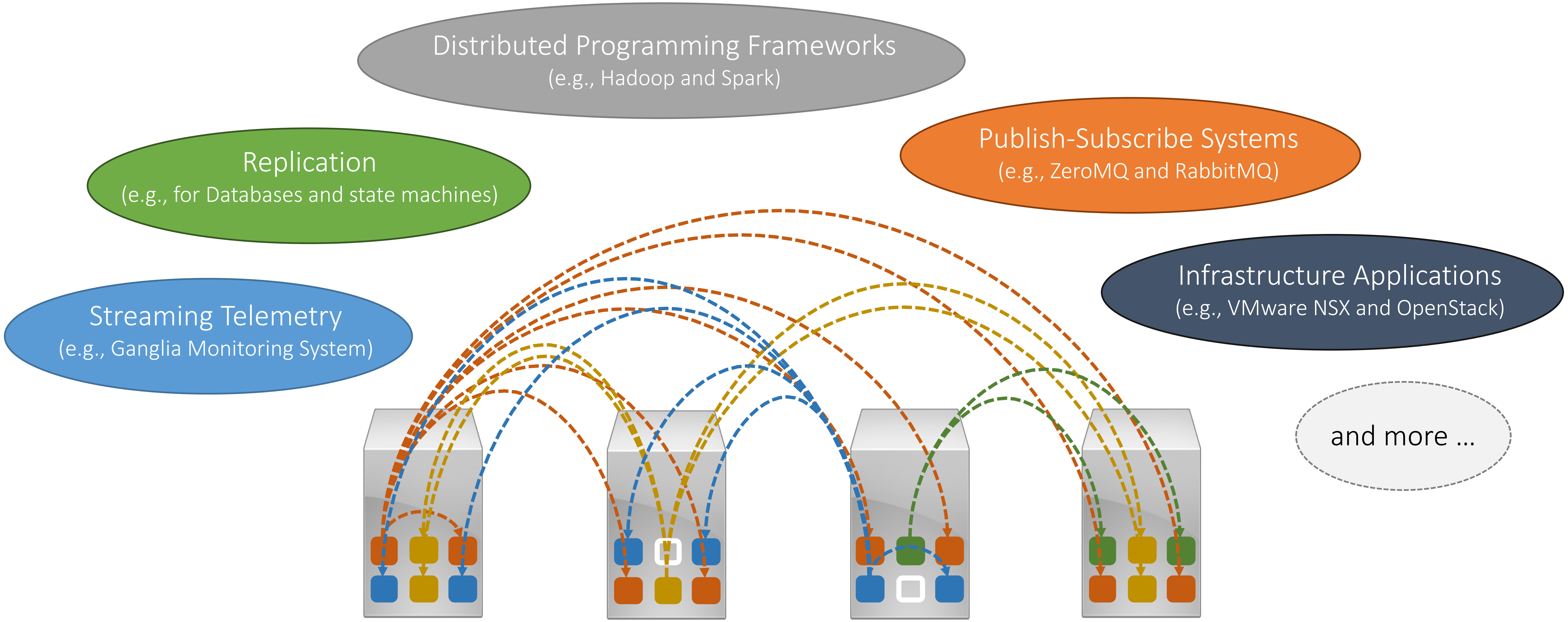


Elmo: Source-Routed Multicast for Public Clouds

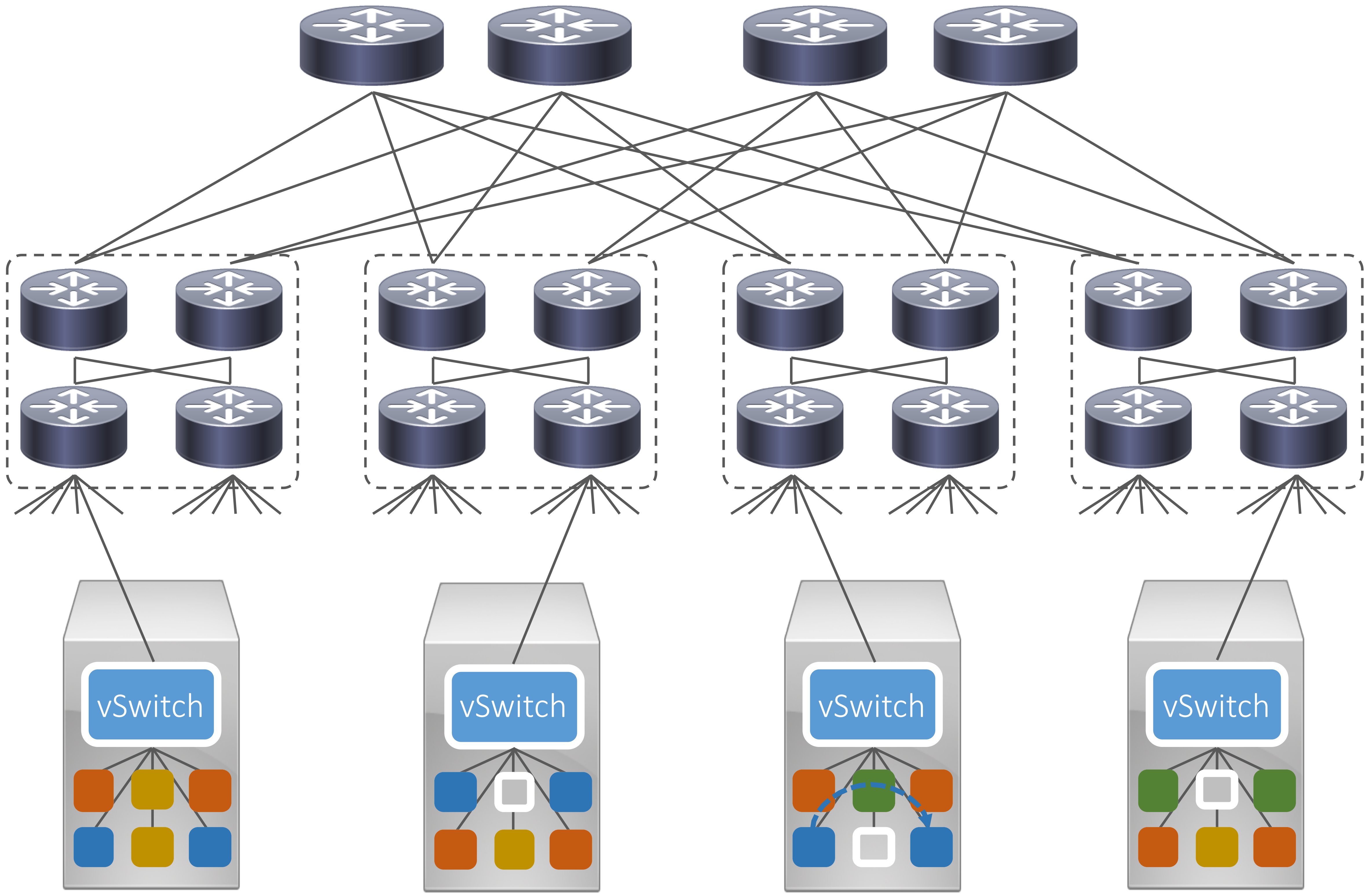
Unicast vs. Multicast vs. Broadcast

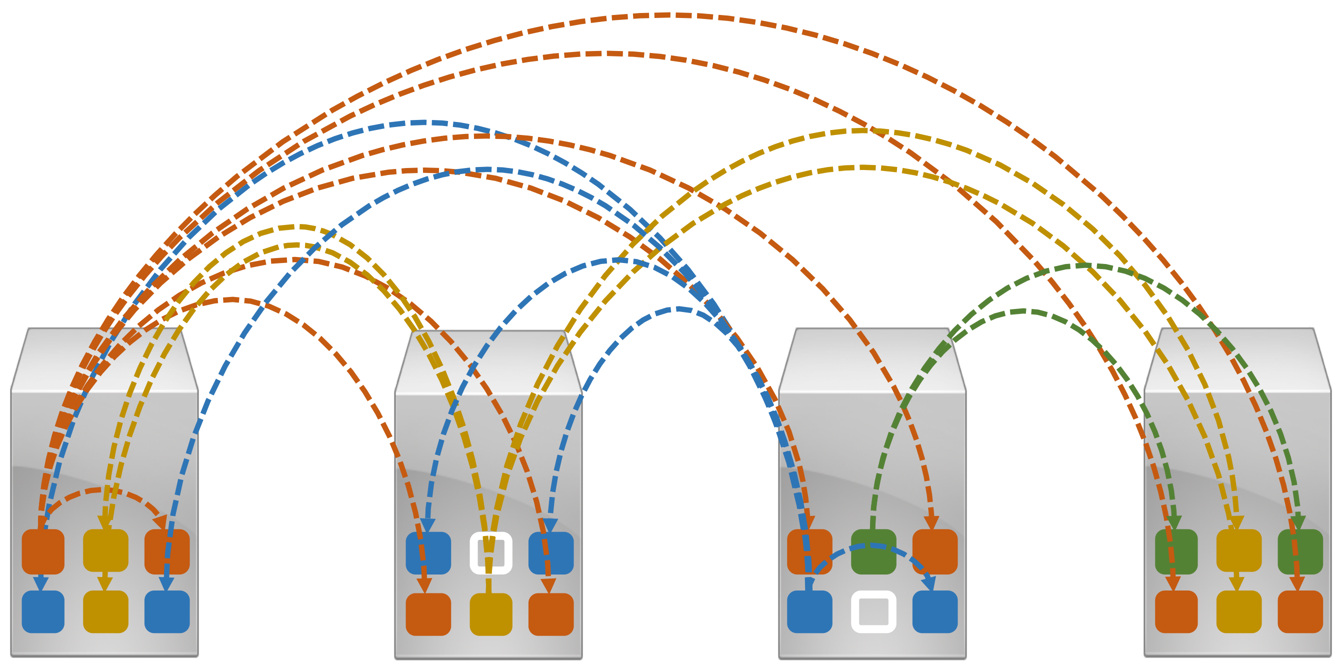


One to Many Communication Pattern in Cloud

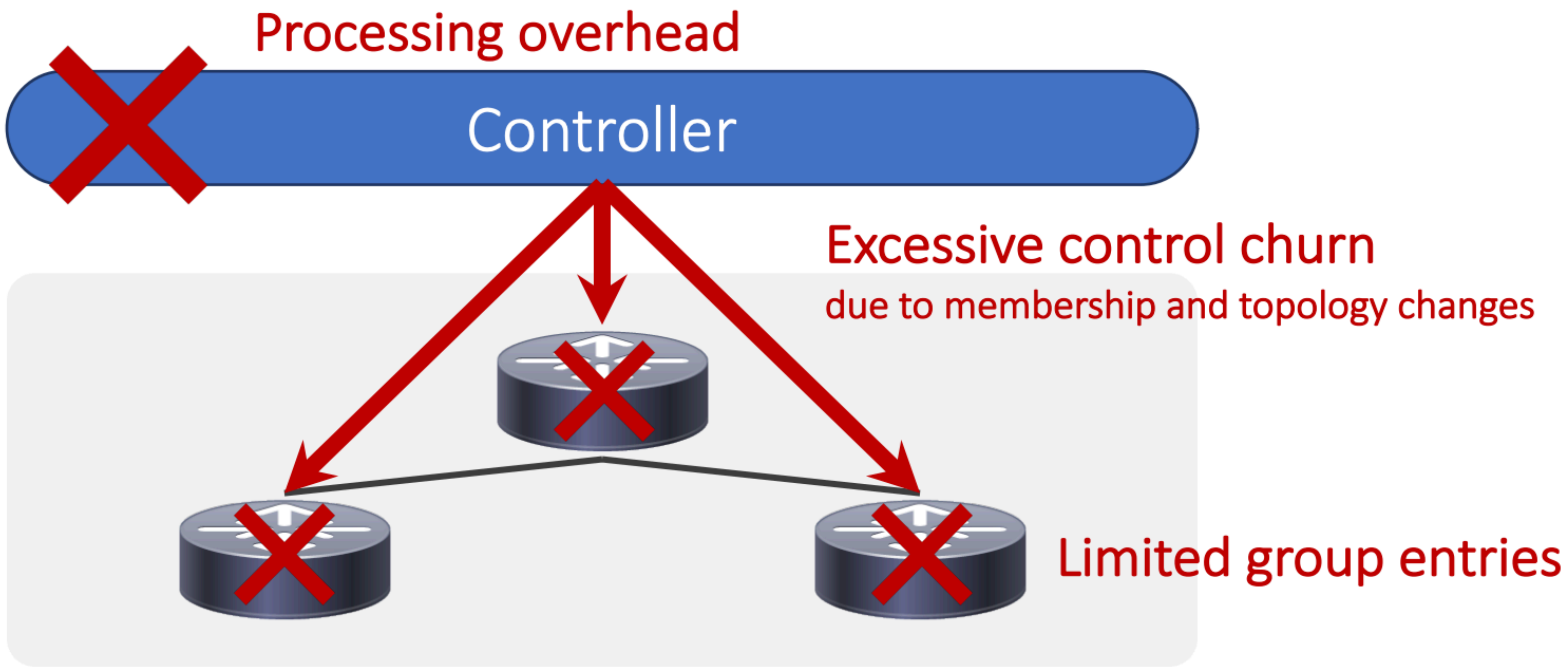


One to Many Communication in Cloud

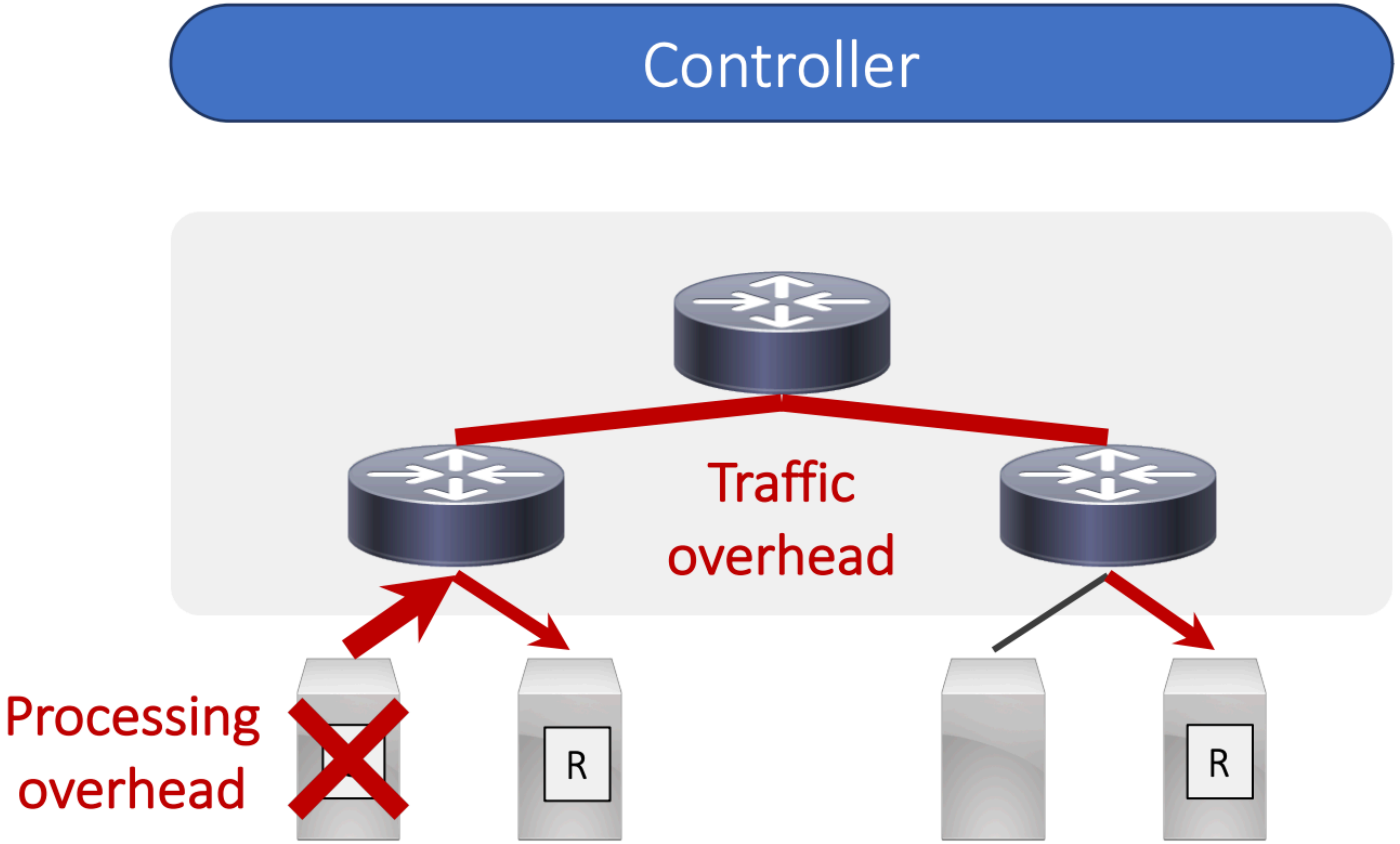




Limitations of Native Multicast

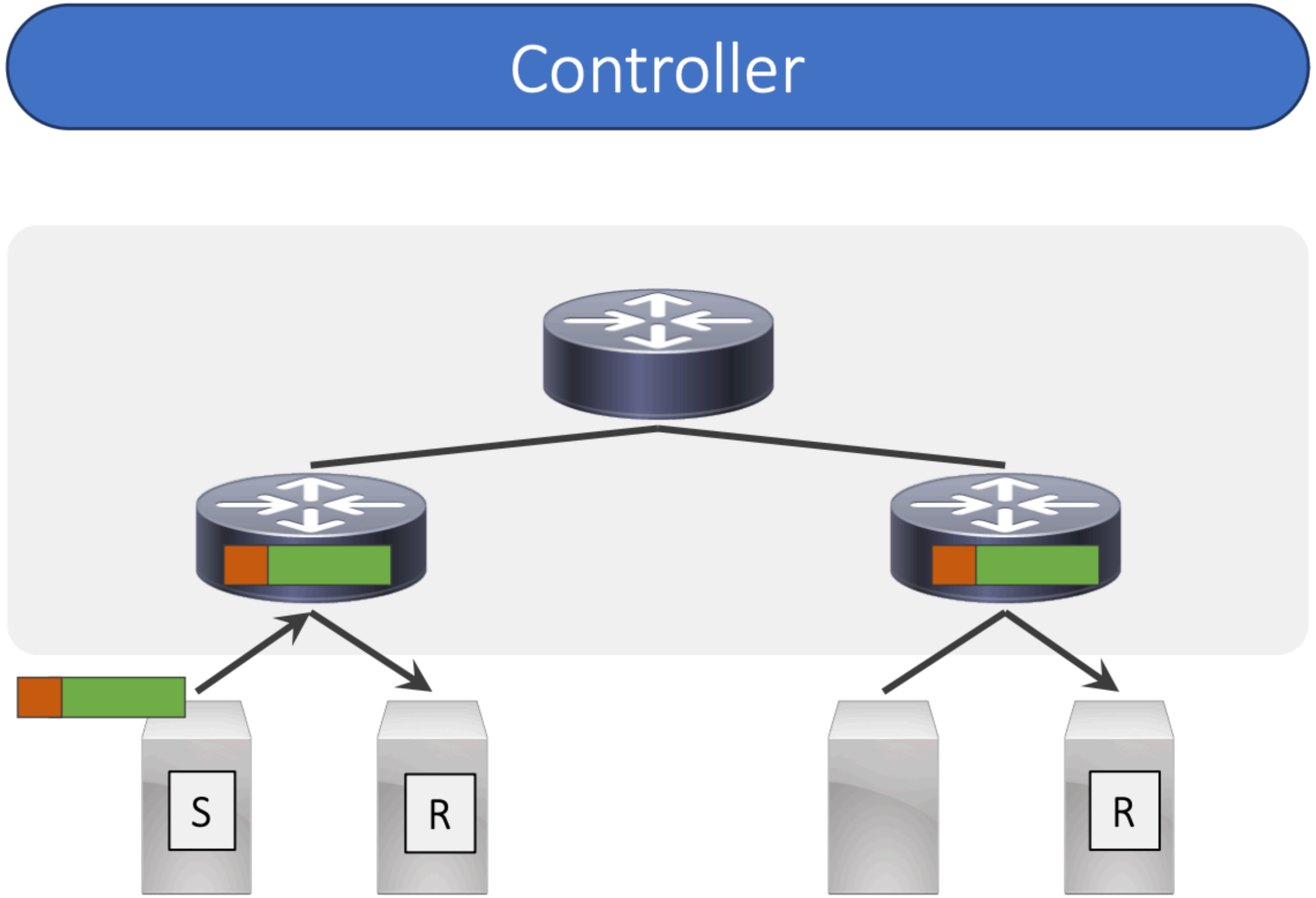


Limitations of Unicast-based Alternatives

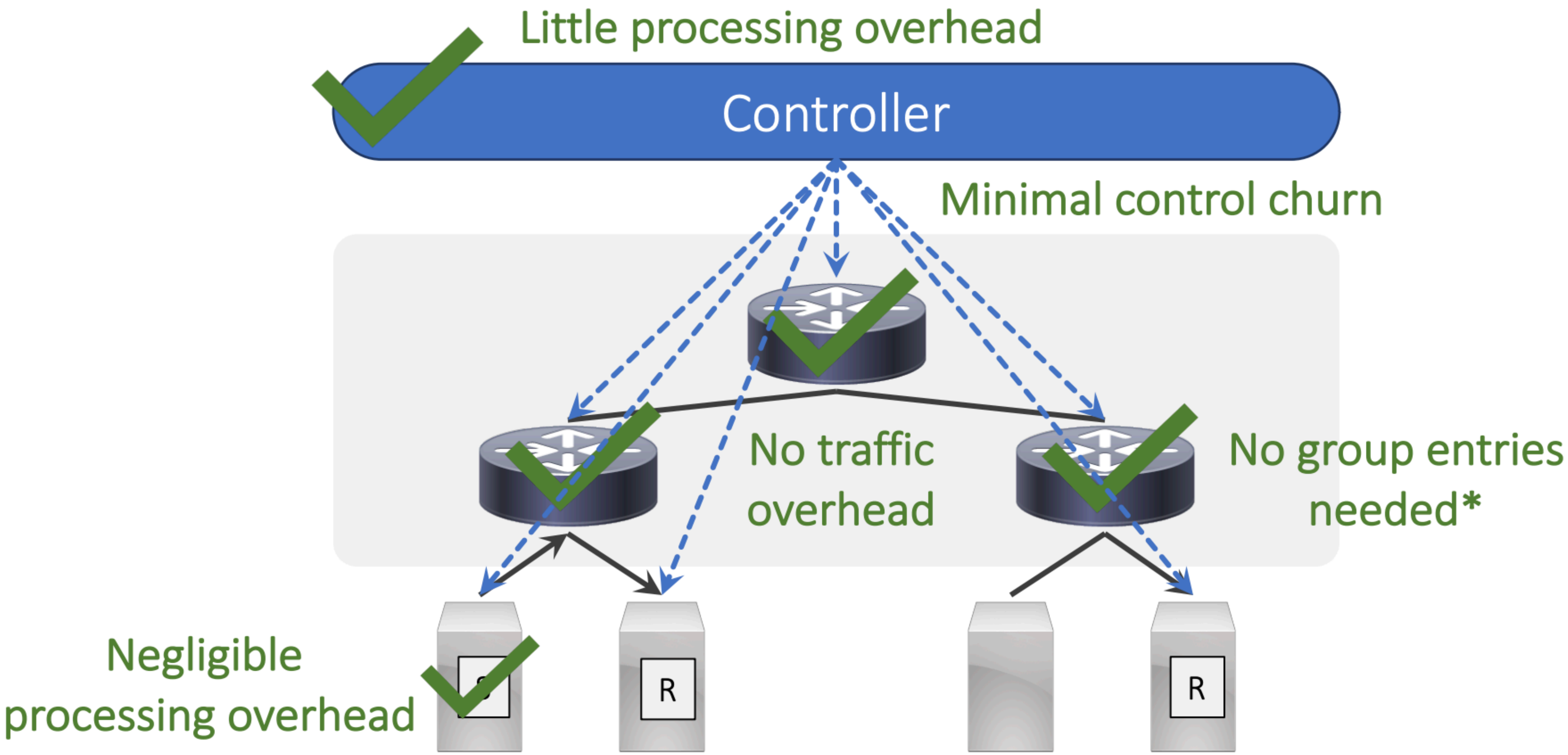


- Key challenges:
    - How to efficiently encode multicast forwarding policy inside packets?
    - How to process this encoding at line rate?

Proposal: Source Routed Multicast



Proposal: Source Routed Multicast



A Naive Source Routed Multicast

A **multicast group** encoded as a list of **(Switch, Ports)** pairs

```
Switch 1: [Ports ]
Switch 2: [.. .. ..]
Switch 3: [.. .. ..]
Switch 4: [.. .. .x ..]
Switch 5: [.x .. .. ..]
```

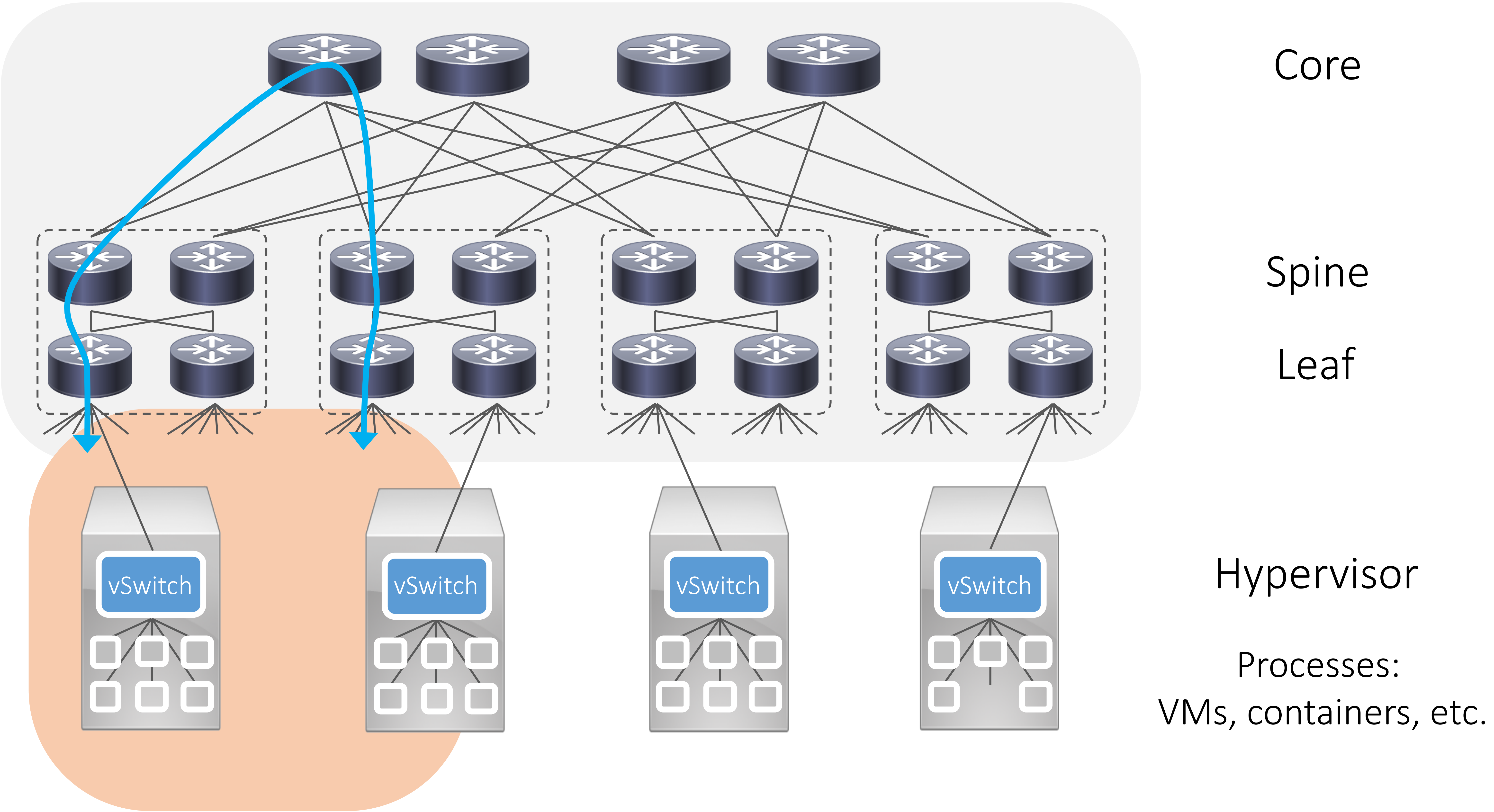
For a data center with:

- **1000** switches
- **48 ports** per switch

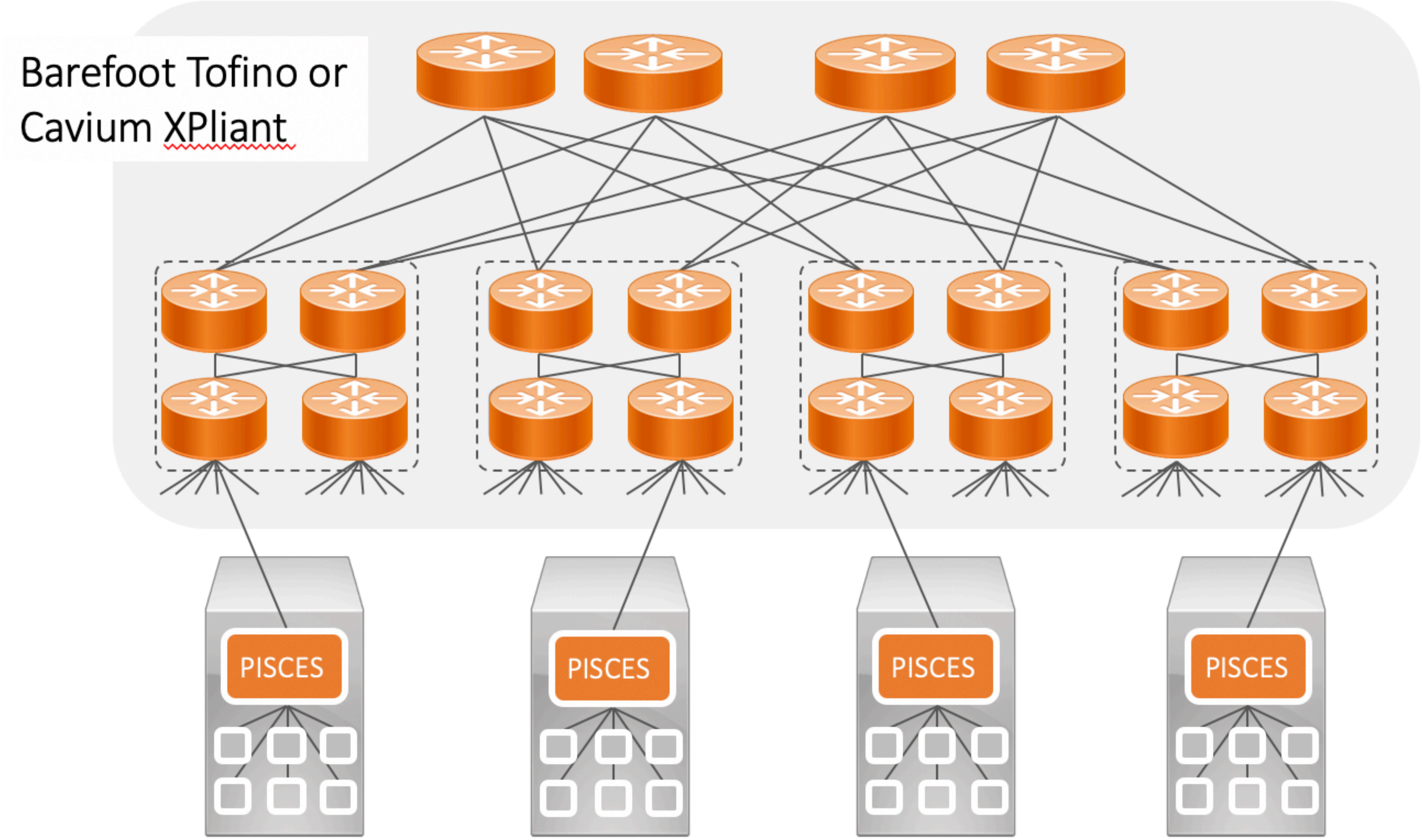
Not Scalable!

Exploiting DC Characteristics for efficient encoding

- Symmetric
- Short Paths
- Co-located Placement



Programmable Switches for Line Rate Processing



Encoding a Multicast Policy in Elmo

A multicast group encoded as a list of **(Switch, Ports)** pairs

```
Switch 1: [Bitmap]
Switch 2: [.. .. ..]
Switch 3: [.. .. ..]
Switch 4: [.. .. .x ..]
Switch 5: [.x .. .. ..]
```

1. Encode switch ports as a **bitmap**

Bitmap is the internal **data structure** that switches use for **replicating packets**

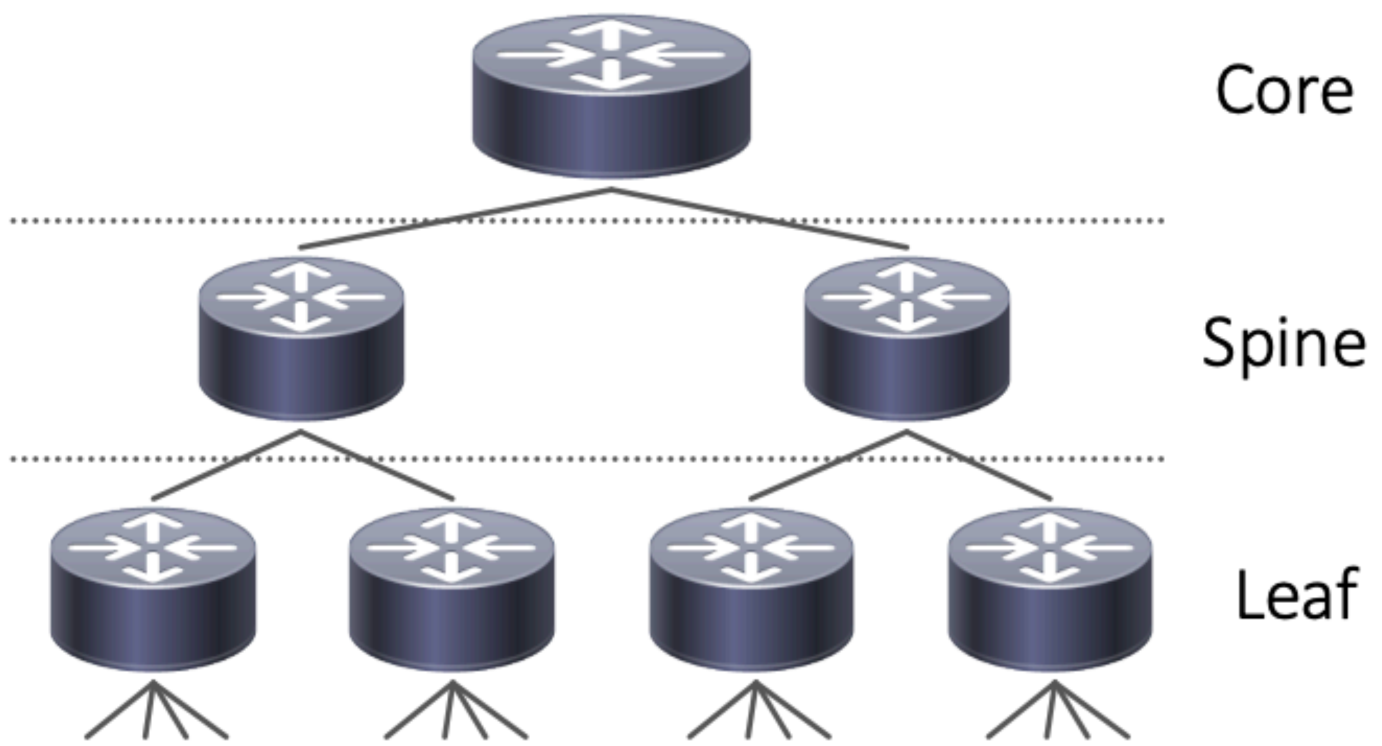
Encoding a Multicast Policy in Elmo

A multicast group encoded as
a list of **(Switch, Ports)** pairs

```
Switch 1: [Bitmap]
Switch 2: [.. .. ..]
Switch 3: [.. .. ..]
Switch 4: [.. .. .x ..]
Switch 5: [.x .. .. ..]
```

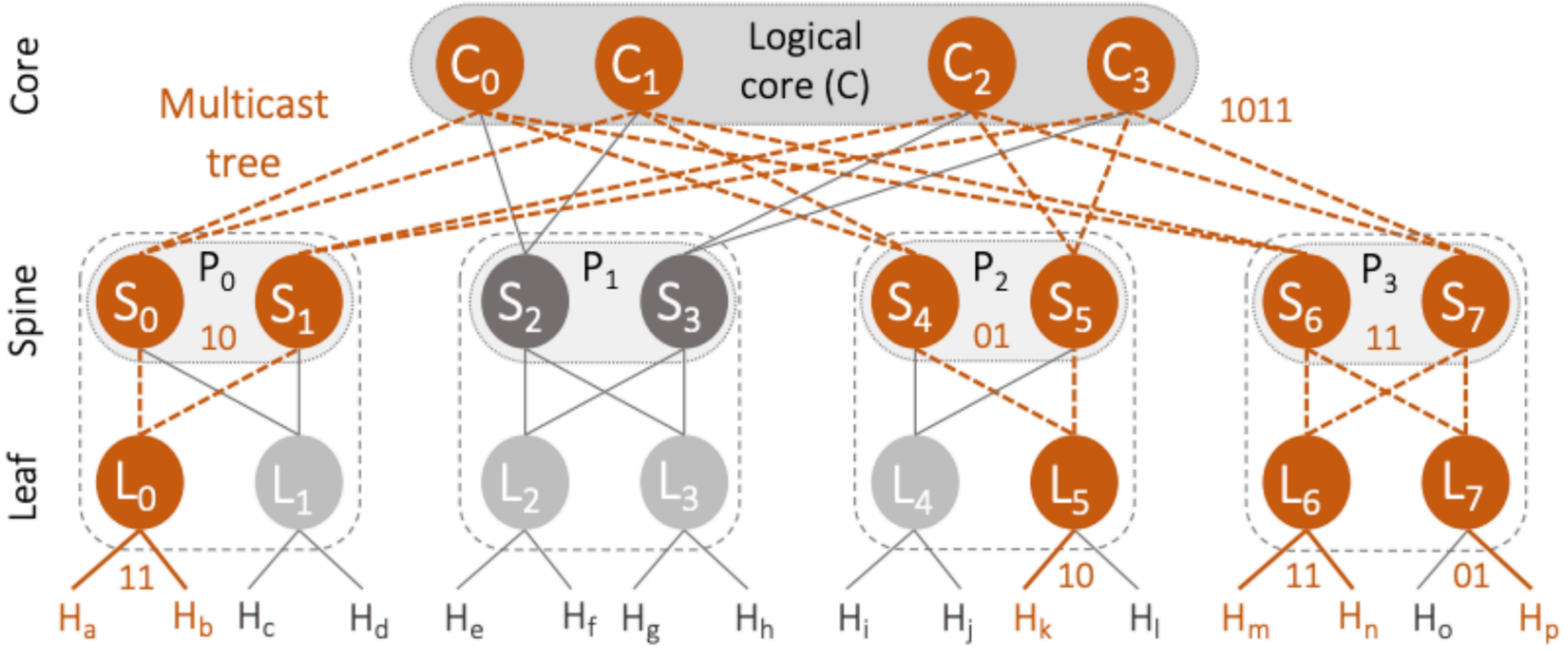
Core
Spine
Leaf

2 Group switches into **layers**



More precisely: *upstream leaf, upstream spine, core, downstream spine, downstream leaf*

Encoding a Multicast Policy in Elmo



Encoding a Multicast Policy in Elmo

A multicast group encoded as
a list of **(Switch, Ports)** pairs

```
Switch 1:   [Bitmap]
Switch 2,3: [.. .. ..]
Switch 4:   [.. .. .x ..]
Switch 5:   [.x .. .. ..]
```

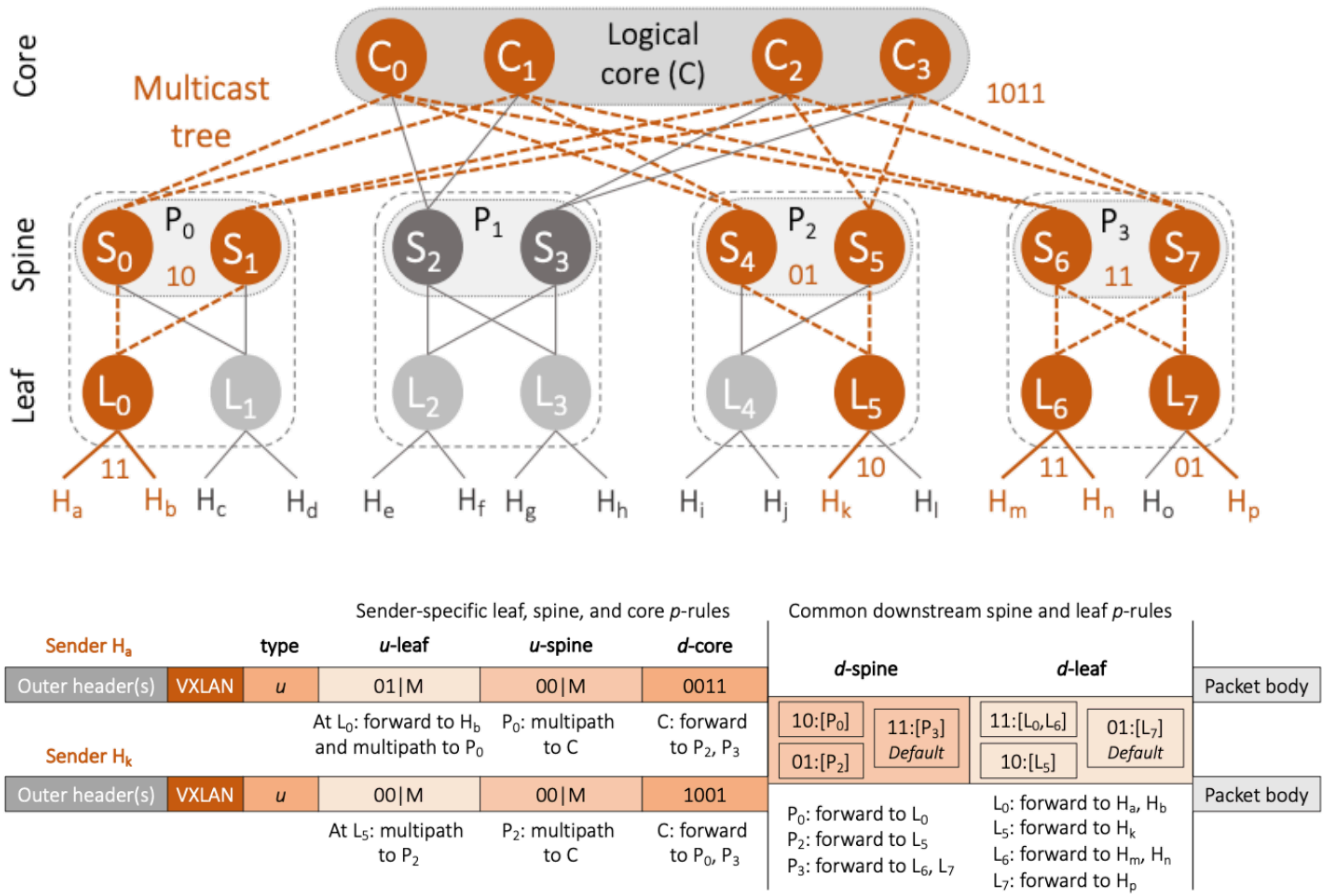
Core

Spine

Leaf

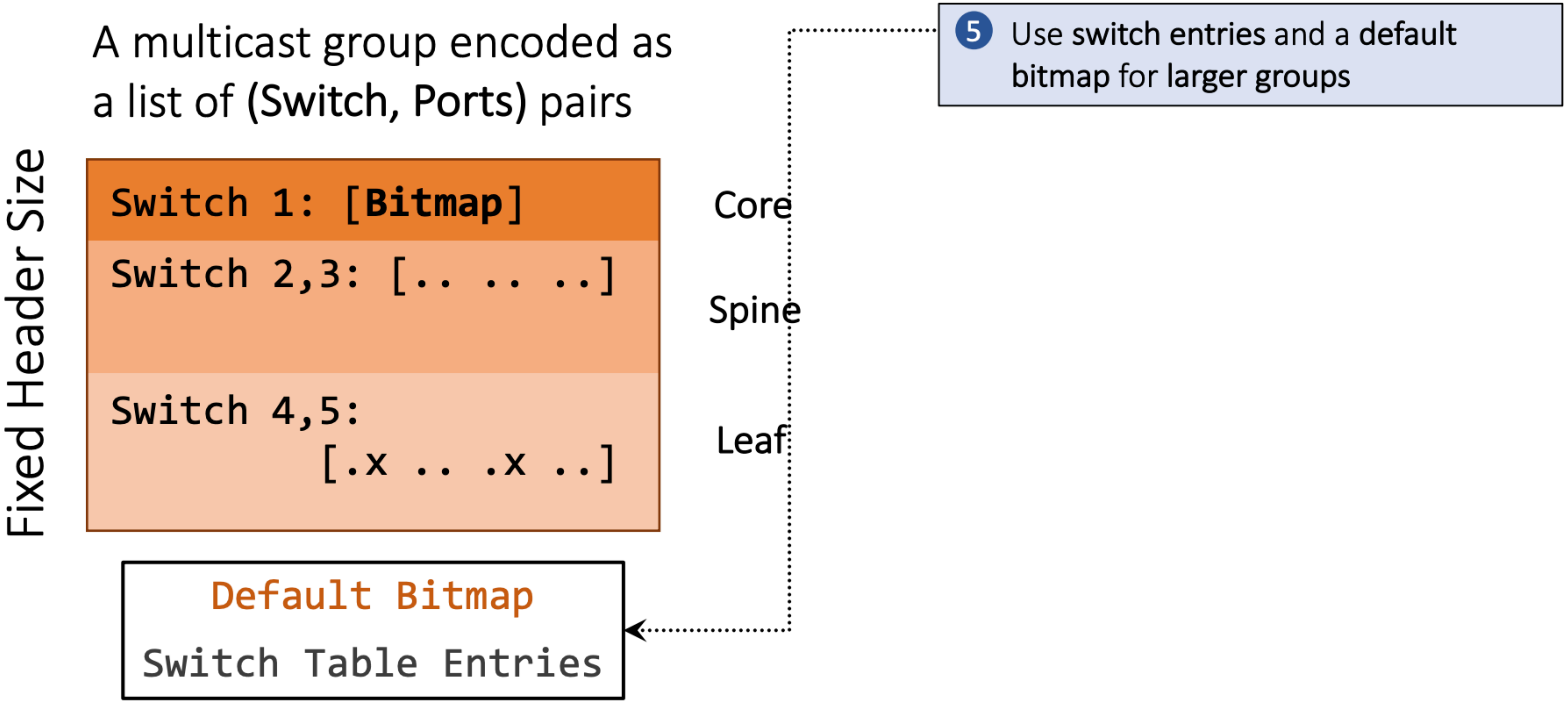
❸ Switches within a layer with **same ports share a bitmap**

Encoding a Multicast Policy in Elmo



Encoding a Multicast Policy in Elmo

A multicast group encoded as a list of **(Switch, Ports)** pairs

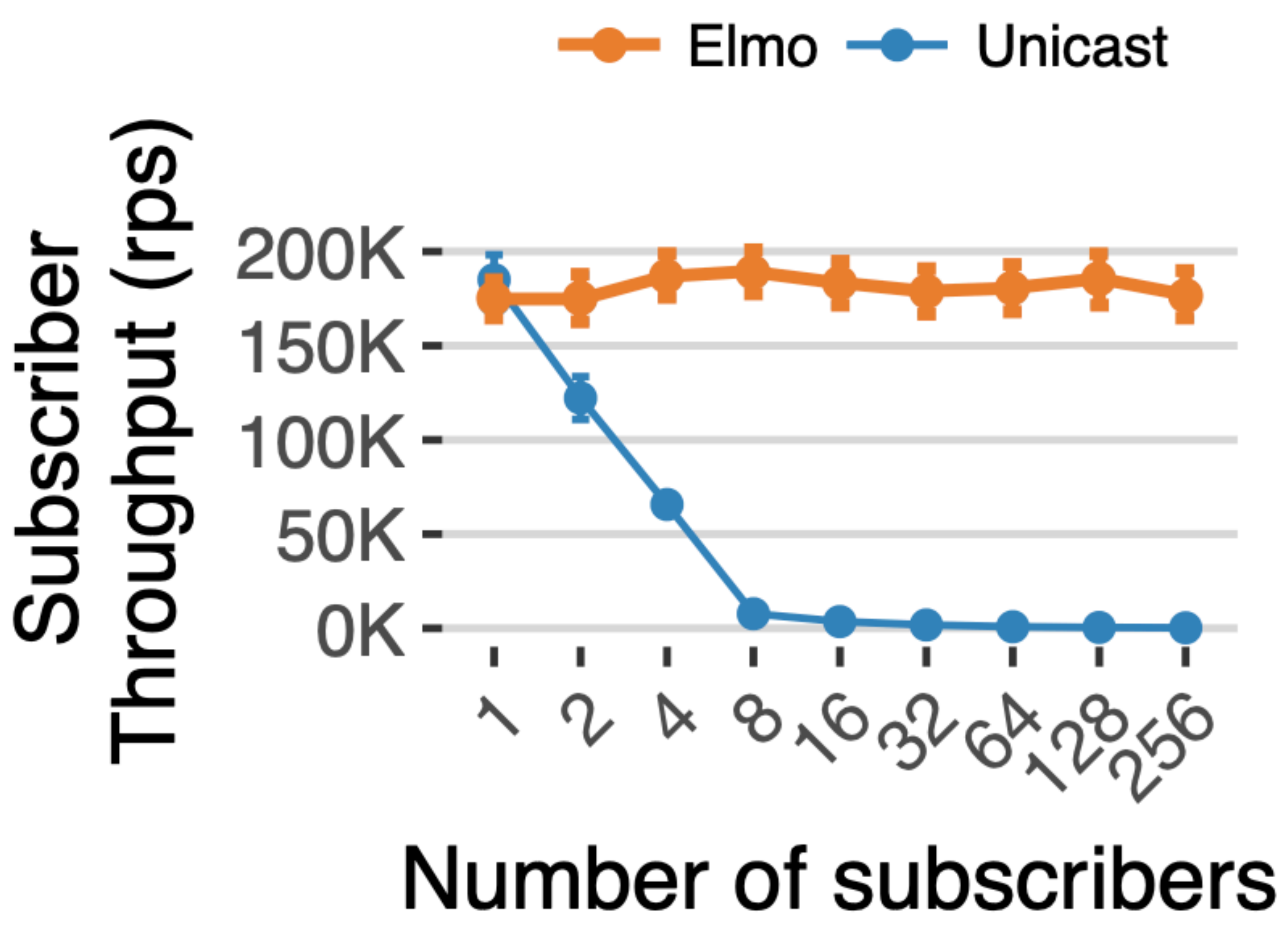


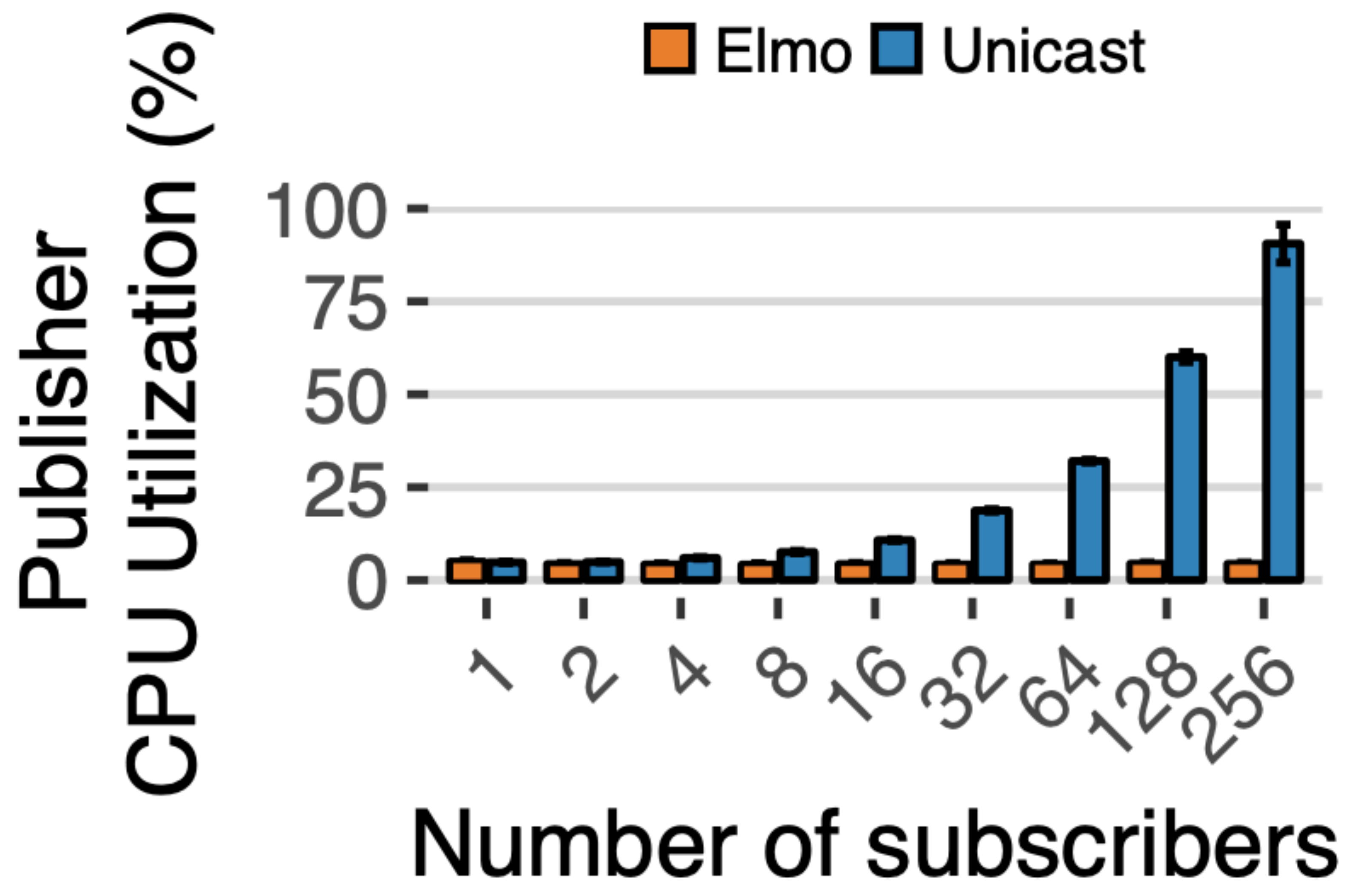
Encoding a Multicast Tree in Elmo

- Key design decisions:
  - Encoding switch output ports in a bitmap
  - Encoding on the logical topology
  - Sharing bitmap across switches
  - Dealing with limited header space using default p-rules
  - Reducing traffic overhead using s-rules

Supports a Million groups!

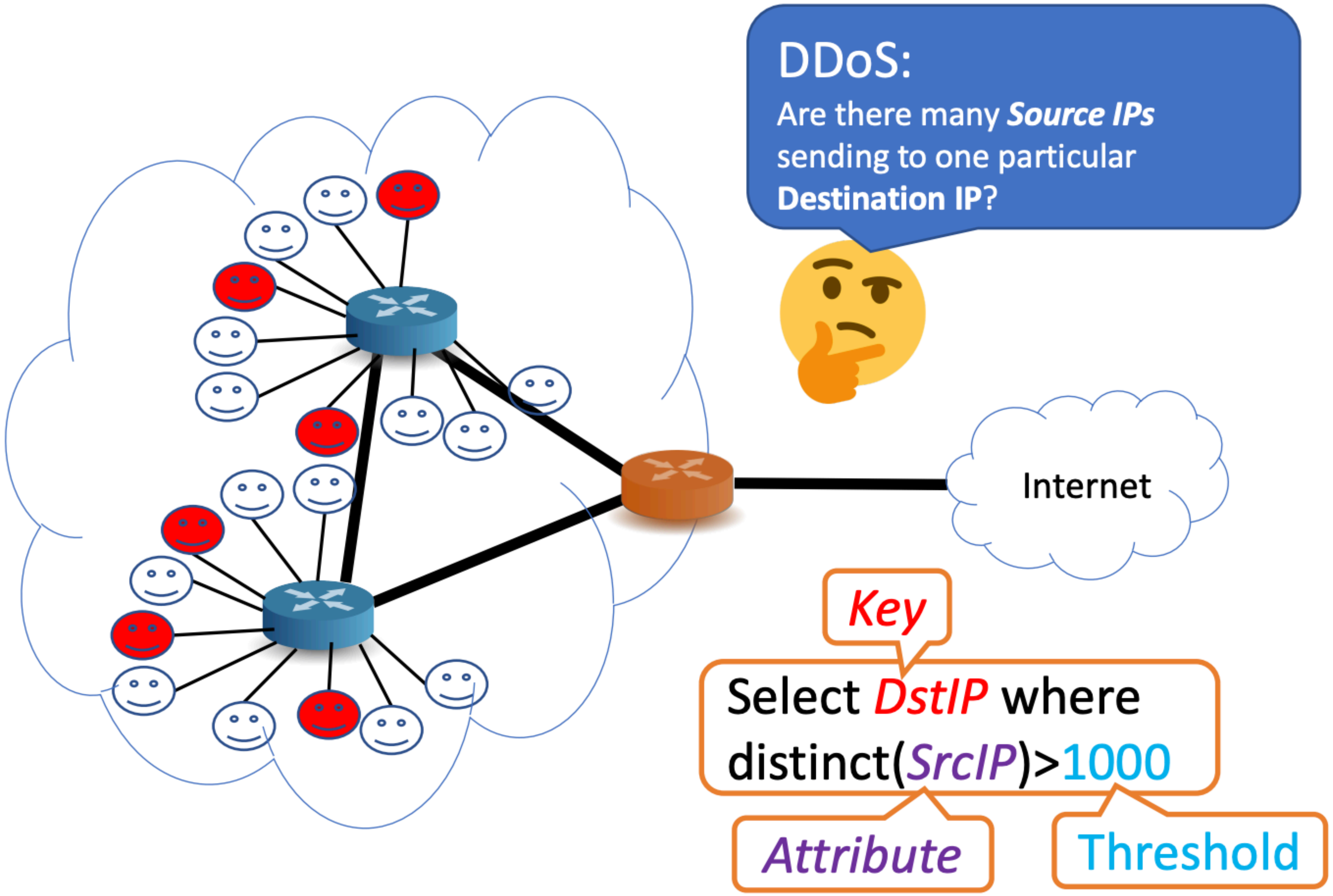
Applications Run Without Performance Overhead



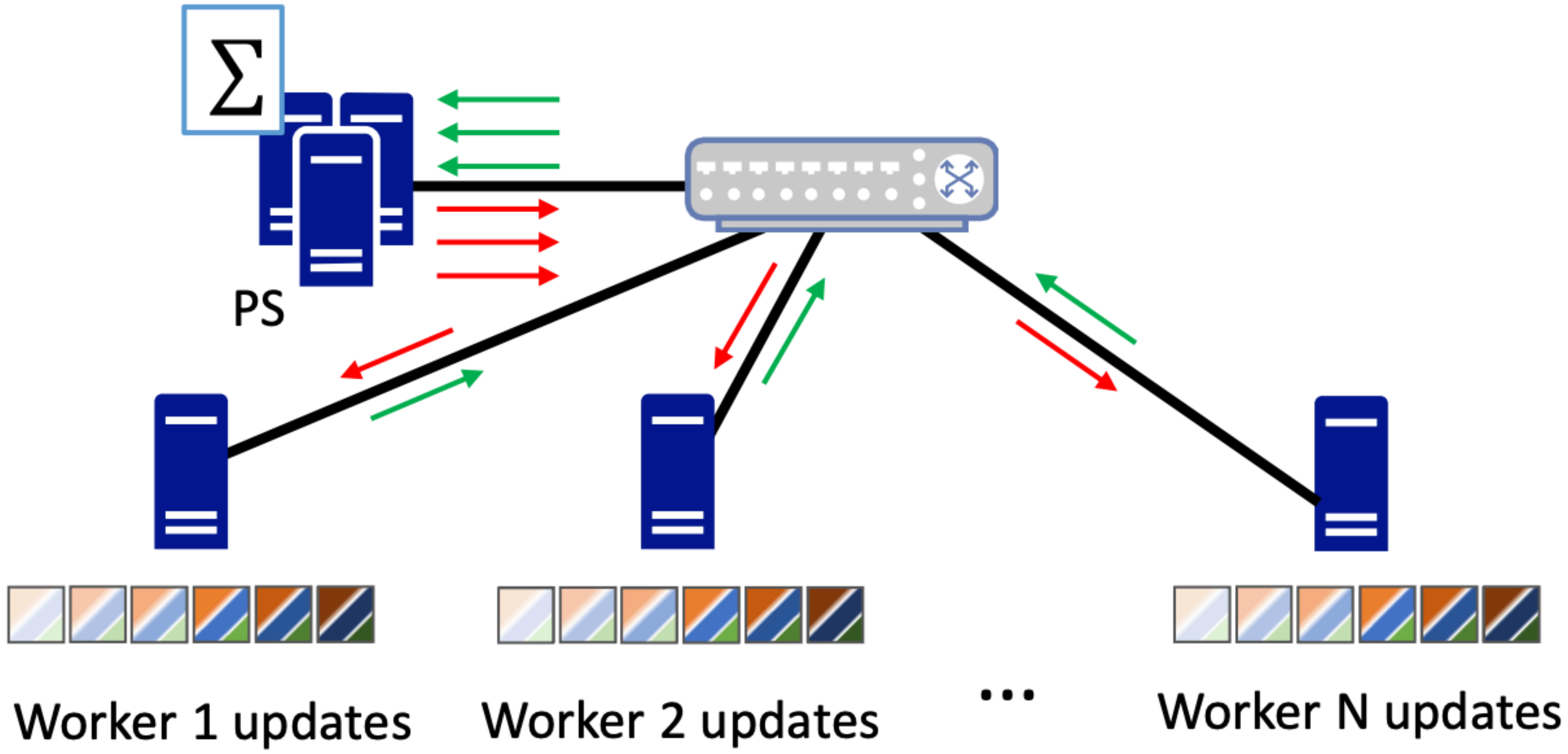


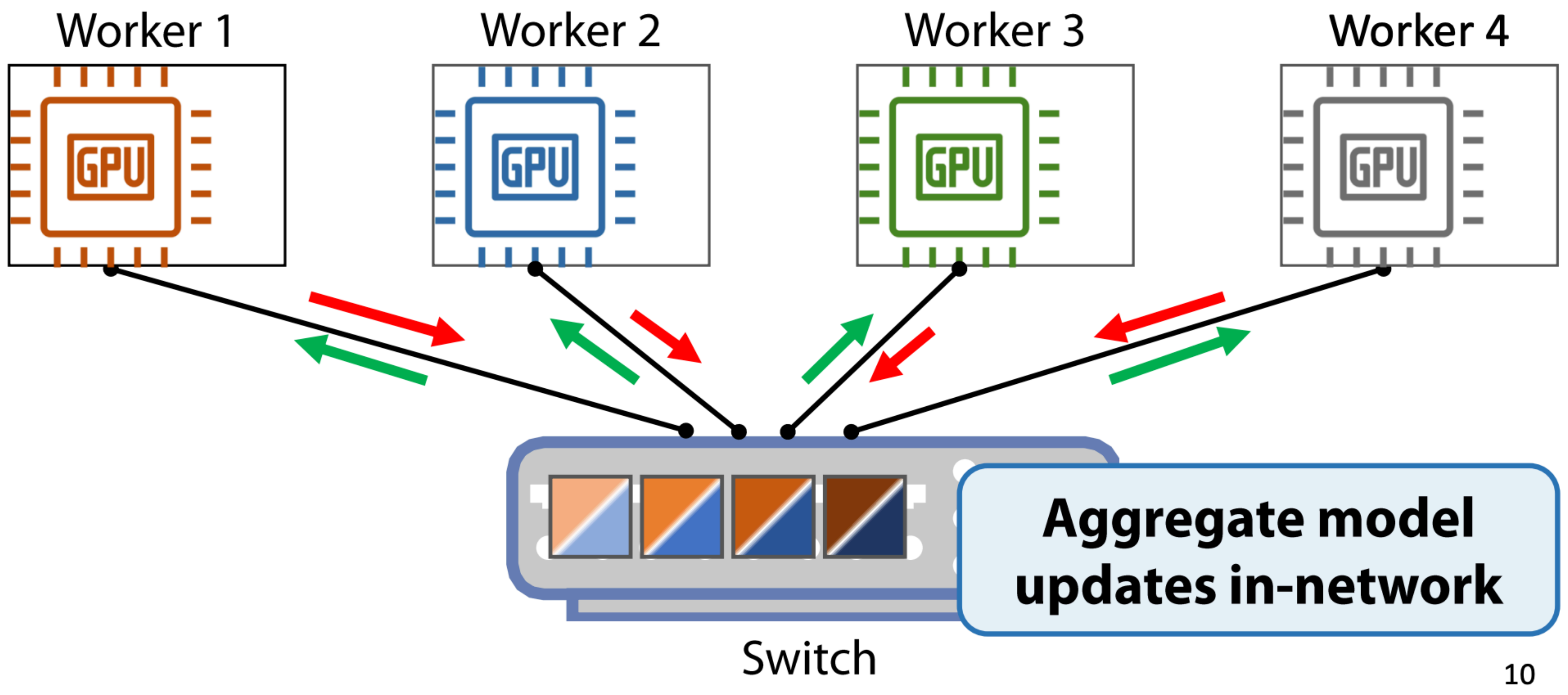
More In-Network Computing-Based Solutions

BeauCoup: Answering many network traffic queries, one memory update at a time! [SIGCOMM'20]



SwitchML [NSDI'21]





SwitchML Challenges

Challenges

Limited computation

Limited storage

No floating points

Packet loss

6.5 Tbps
programmable
data plane

Other Networking Usecases

- • Load balancing:
    - HULA: Scalable Load Balancing Using Programmable Data Planes, SOSR'16
- Congestion control:
    - Evaluating the Power of Flexible Packet Processing for Network Resource Allocation, NSDI'17
    - HPCC: High Precision Congestion Control, SIGCOMM'19
- A new protocols for more efficient L2 switching
    - The Deforestation of L2, SIGCOMM'16

Other app-level use cases

- NetChain [SOSP'17]: in-network key-value store
- NetLock [SIGCOMM'20]: Switching support to manage locks
- NetPaxos [SOSR'15]: implement Paxos on programmable switches
- NoPaxos [OSDI'16]: in-network primitives for distributed protocols

Thanks!