Elmo: Source-Routed Multicast for Public Clouds

Muhammad Shahbaz, Lalith Suresh, Jen Rexford, Nick Feamster, Ori Rottenstreich, and Mukesh Hira

1. Motivation

Modern cloud workloads (e.g., publish-subscribe, analytics, telemetry, replication, messaging, finance, and more) frequently exhibit

- one-to-many, **multicast** communication patterns
- and require sub-millisecond latencies and high throughput

Yet, **none** of the cloud providers today (e.g., Azure, GCP, aws) support native multicast

- because of the inherent data- and control-plane scalability limitations of current approaches, see →

We believe **Elmo**, a source-routed multicast can address these limitations as

- emerging **programmable data planes** and **unique characteristics of data center topologies** lead to efficient implementations of source-routed multicast
- and alleviates both the pressure on switching hardware resources and control-plane overheads during churn

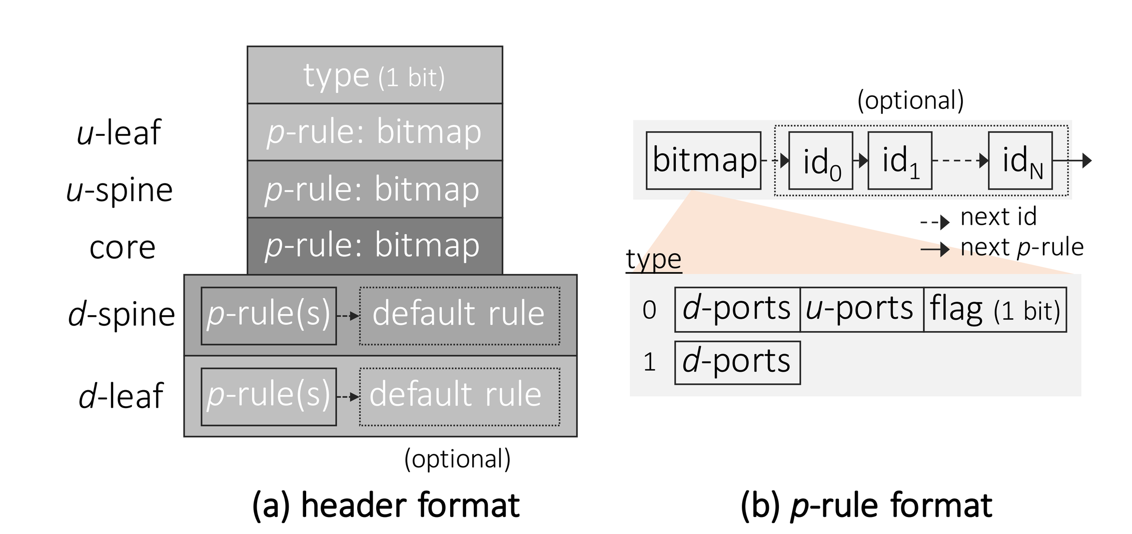
Feature	IP Multicast	Li et al.	Rule aggr.	App. Layer	BIER	SGM	Elmo
#Groups	5K	150K	500K	1M+	1M+	1M+	1M+
Group-table usage	high	high	mod	none	low	none	low
Flow-table usage	none	mod	high	none	none	none	none
Group-size limits	none	none	none	none	2.6K	<100	none
Network-size limits: #hosts	none	none	none	none	2.6K	none	none
Unorthodox switch capabilities	no	no	no	no	yes	yes	no
Line-rate processing	yes	yes	yes	no	yes	no	yes
Address-space isolation	no	no	no	yes	yes	yes	yes
Multipath forwarding	no	lim	lim	yes	yes	yes	yes
Control overhead	high	low	mod	none	low	low	low
Traffic overhead	none	none	low	high	low	none	low
End-host replication	no	no	no	yes	no	no	no

Comparison between Elmo and related multicast approaches for public clouds

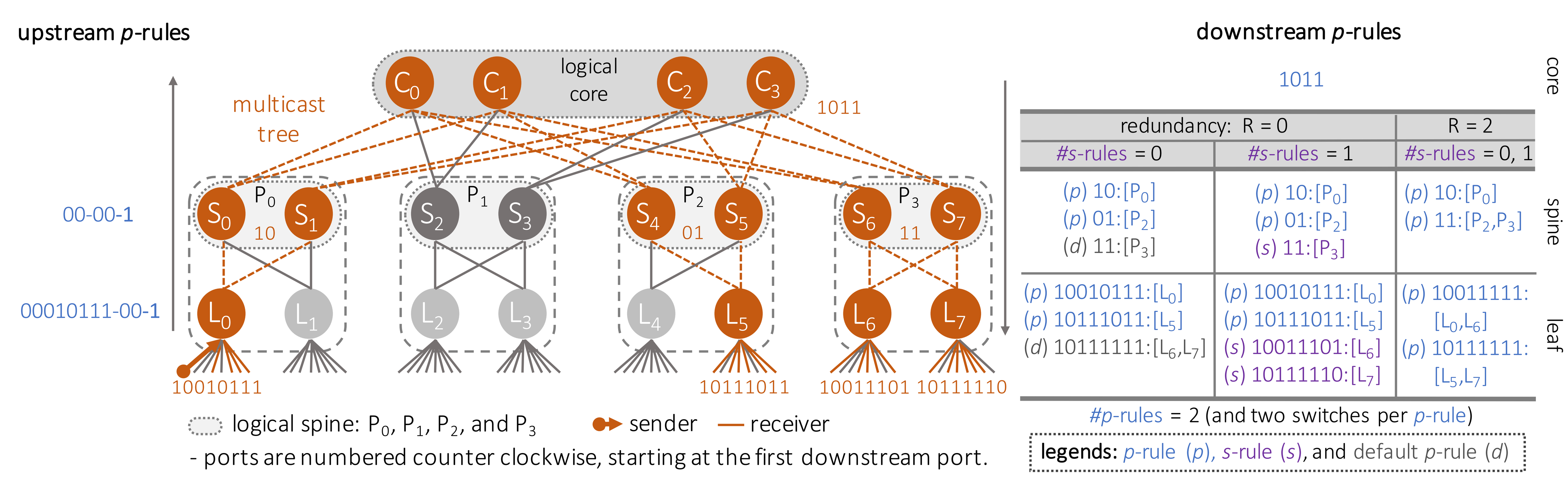
2. Approach: Encode Multicast Trees Inside Packets using Prog. Switches

Design decisions for encoding multicast trees:

1. Encoding switch ports in a **bitmap**
2. Encoding on the **logical topology**
3. **Sharing bitmap** across switches (e.g., R > 0)
4. Limiting header size using **default packet (p-) rules**
5. Reducing traffic overhead using **switch (or s-) rules**



Elmo's header and *p*-rule format. (*u*: upstream, *d*: downstream.)



Encoding multicast tree. An example multicast tree on a three-tier multi-rooted Clos topology with upstream and downstream *p*-rules (*i.e.*, rules encoded inside a packet) and *s*-rules (*i.e.*, rules installed in a switch) assignment for a group. A packet originating from the sender is forwarded up to the logical core using the upstream *p*-rules, and down to the receivers using the downstream *p*-rules (and *s*-rules). For example, when $R = 0$ and $\#s\text{-}rules = 1$, a packet arriving at P_2 (S_4 or S_5) from the core is forwarded using the *p*-rule 01, whereas at P_3, it is forwarded using the *s*-rule 11.

3. Evaluation

a. Data Plane Scalability

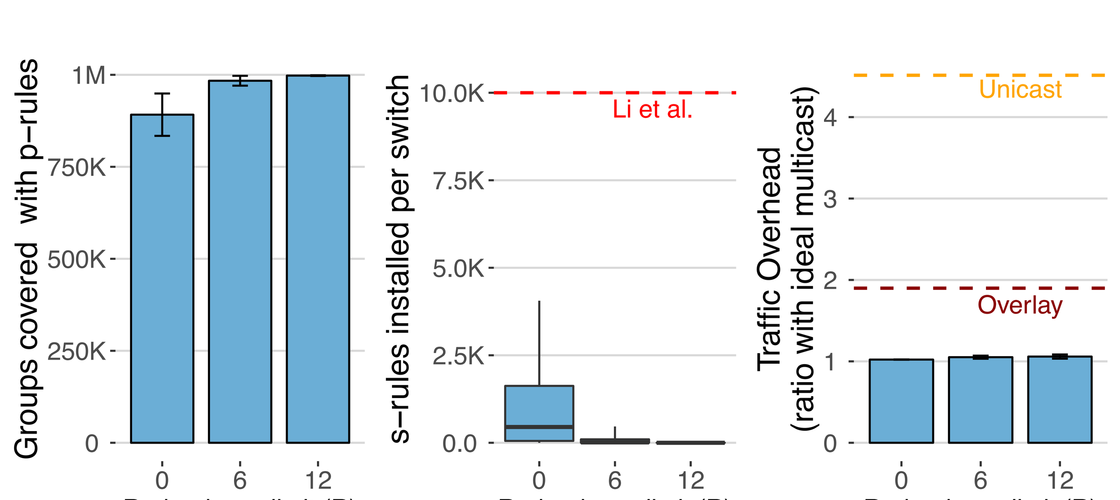


Figure 1. Placement strategy with no more than 12 VMs of a tenant per rack (*i.e.*, colocated VMs).

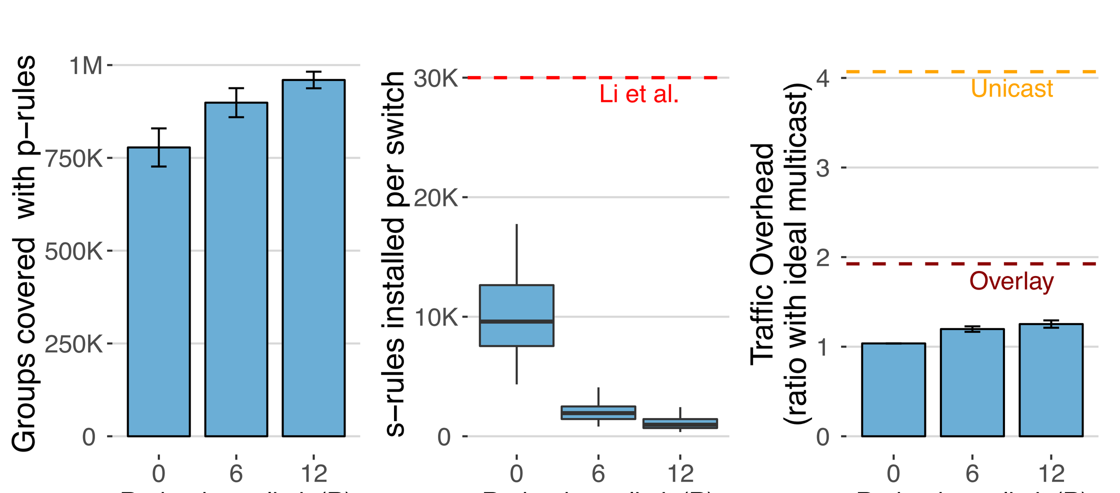


Figure 2. Placement strategy with no more than one VM of a tenant per rack (*i.e.*, dispersed VMs).

b. Control Plane Scalability

Switch	Elmo	Li et al.
hypervisor	21 (46)	NE (NE)
leaf	5 (13)	42 (42)
spine	4 (7)	78 (81)
core	0 (0)	133 (203)

Figure 3. The average (max) number of switch updates per second when no more than one VM of a tenant is placed per rack. (NE: not evaluated by Li et al.)

c. Hardware Resource Usage

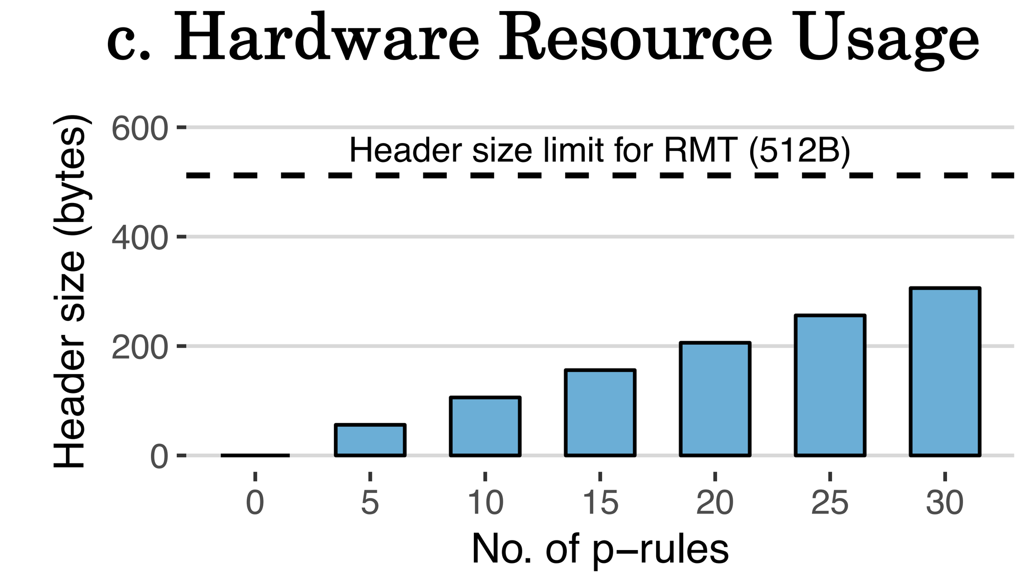


Figure 4. Header usage with varying number of *p*-rules.

d. End-to-End Application Results

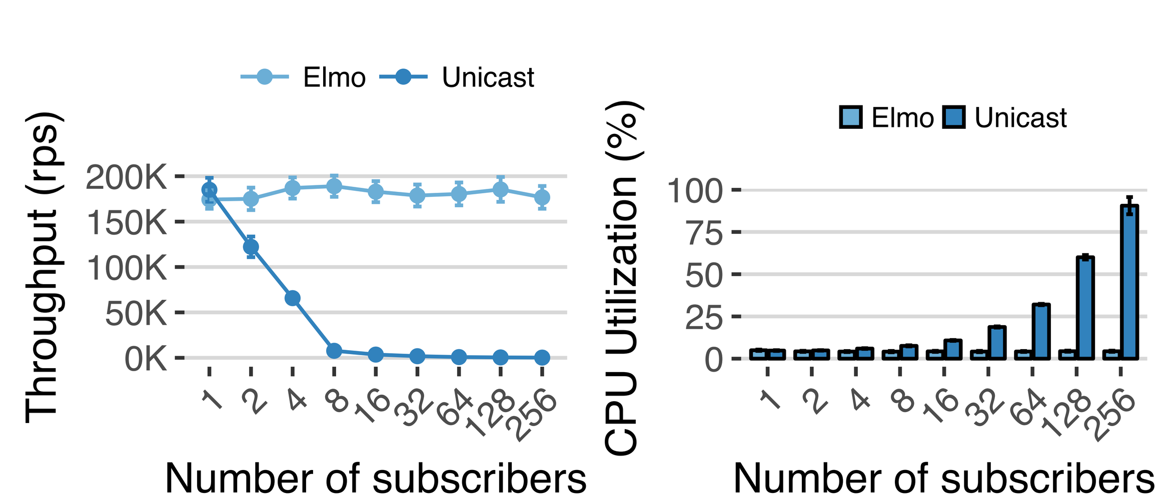


Figure 5. Comparison of a pub-sub application using ZeroMQ (over UDP) with a message size of 100 bytes.

e. Hypervisor Switch Overhead

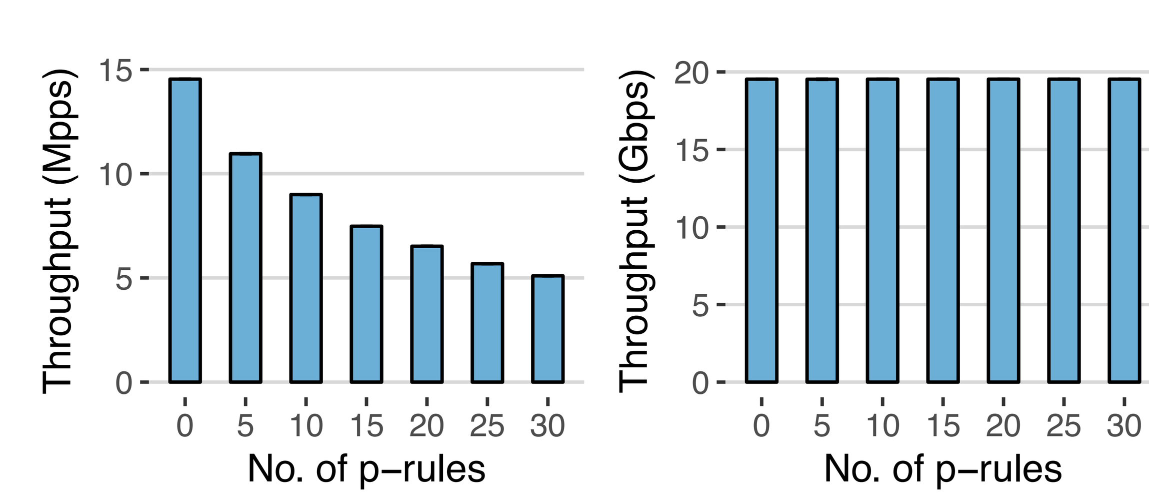


Figure 6. Hypervisor switch (*i.e.*, PISCES) throughput when adding different number of *p*-rules.