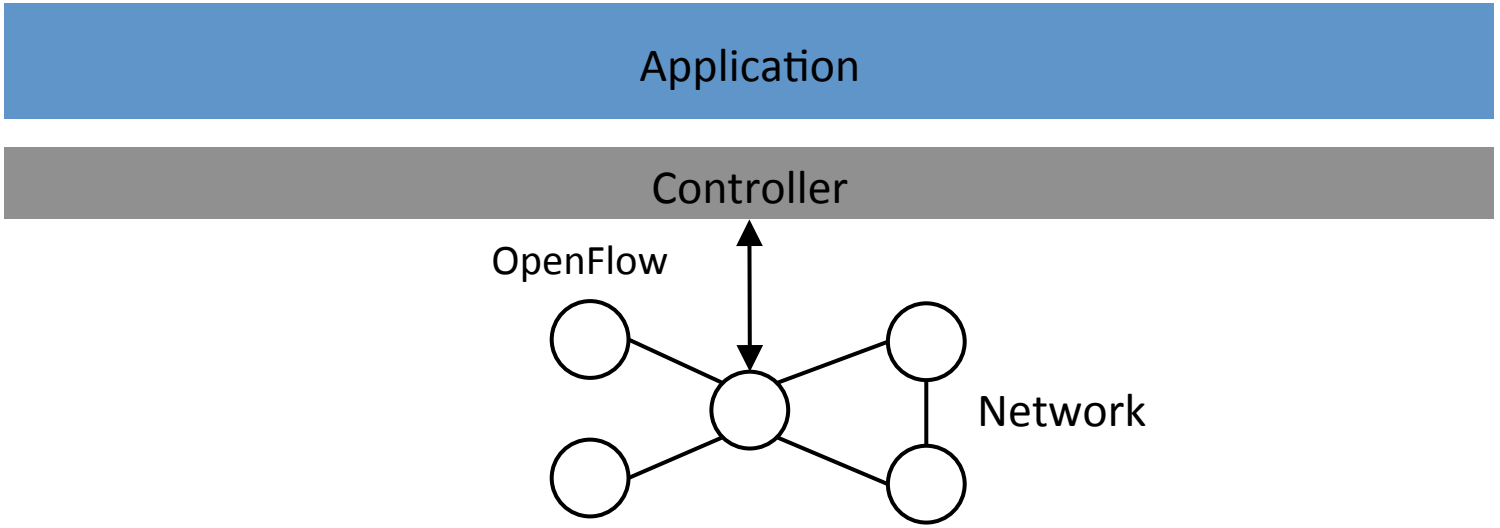
CoVisor: A Compositional Hypervisor for Software-Defined Networks

Xin Jin

Jennifer Gossels, Jennifer Rexford, David Walker

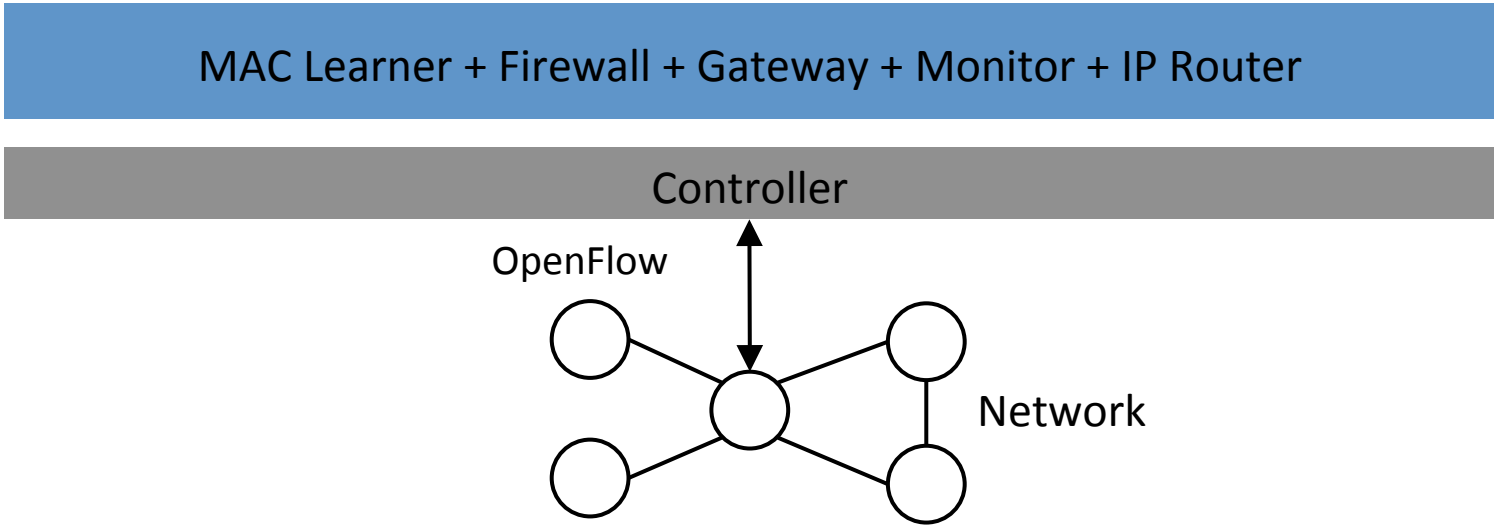
Software-Defined Networking

- Centralized control with open APIs



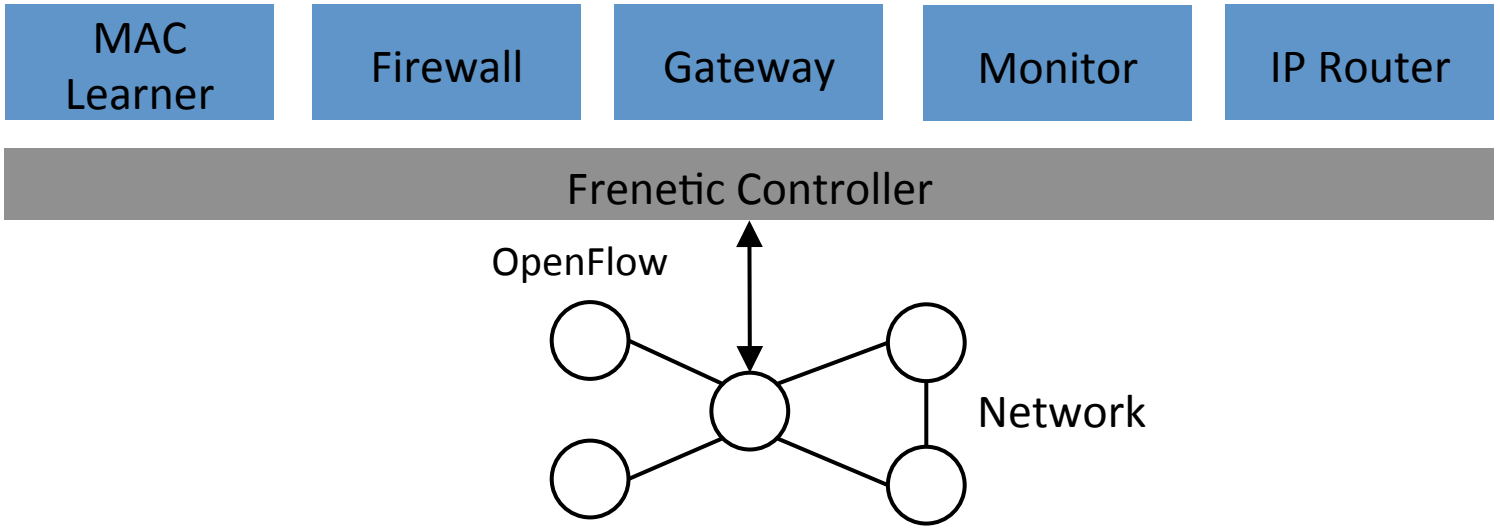
Multiple Management Tasks

- Hard to develop and maintain a monolithic application



Modular SDN Applications

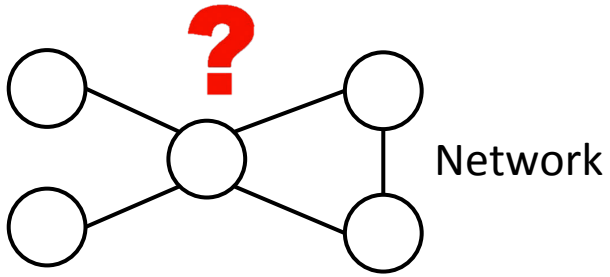
- Frenetic: composition operators to combine multiple applications
- Limitation: need to adopt Frenetic language and runtime system



Frenetic is Not Enough

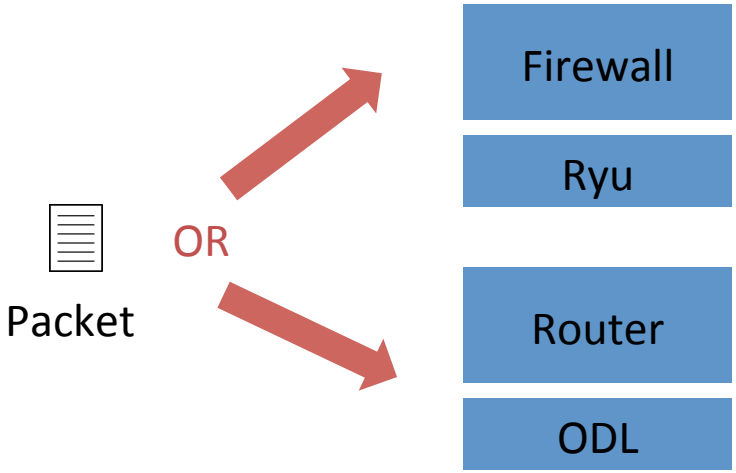
- "Best of breed" applications are developed by different parties
  - Use different programming languages
  - Run on different controllers
- Want to mix-and-match third-party controllers

MAC Learner	Firewall	Gateway	Monitor	IP Router
POX	Ryu	Floodlight	ONOS	ODL

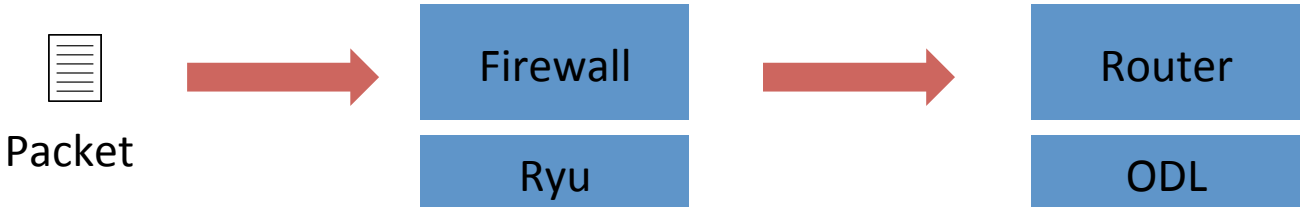


Slicing is Not Enough

- FlowVisor/Open VirteX: each controller works on a disjoint slice of traffic



- But, we want multiple controllers to collaboratively work on the same traffic



CoVisor: A Compositional Hypervisor for SDN

- Provide a clean interface to compose multiple controllers on the same network
- Composition of multiple controllers
  - Composition operators to compose multiple controllers
- Constraints on individual controllers
  - Visibility: virtual topology to each controller
  - Capability: fine-grained access control to each controller

Composition of Multiple Controllers

- Parallel operator (+): two controllers process packets in parallel

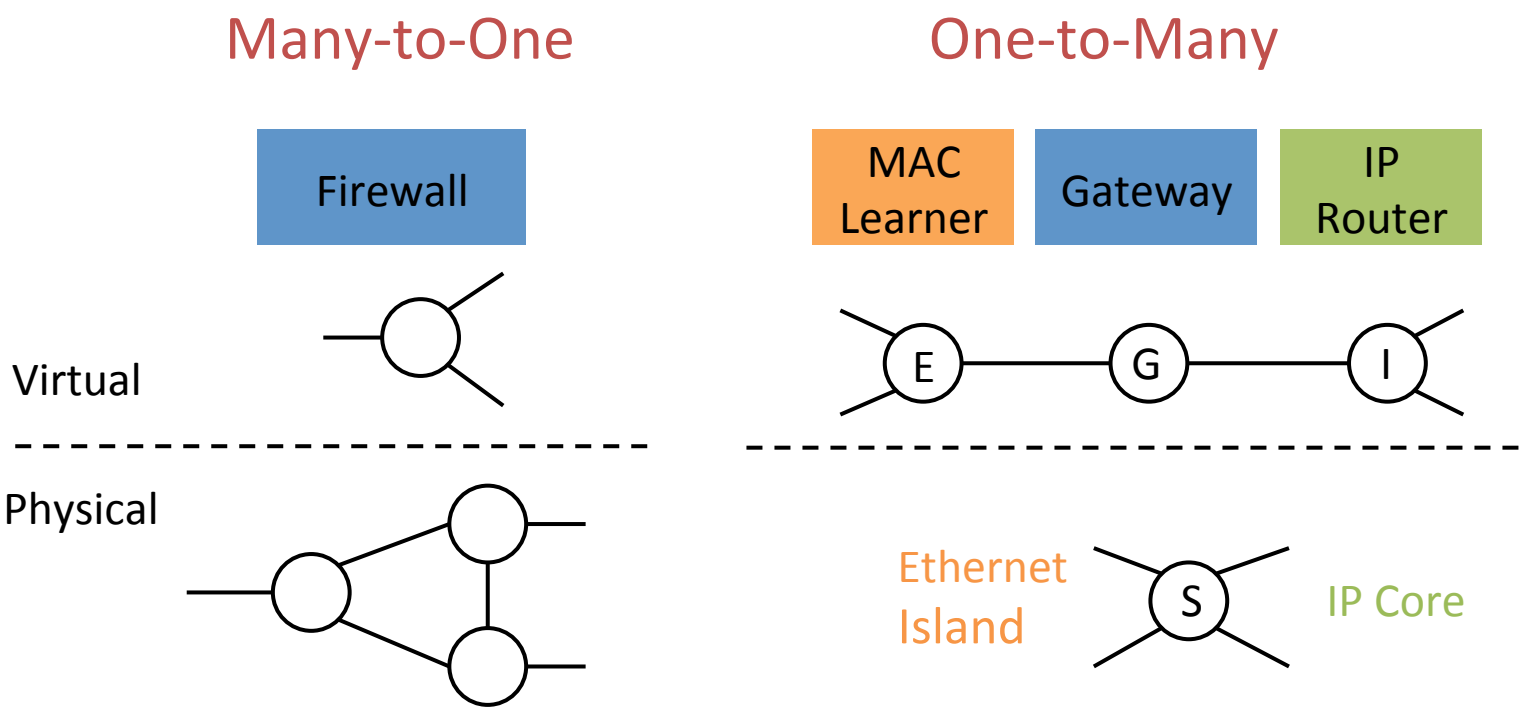
- Sequential operator (>>): two controllers process packets one after another

- Override operator (▷): one controller chooses to act or defer the process to another controller

- Use multiple operators

Constraints on Topology Visibility

- Create virtual topology with two primitives
- Benefits: information hiding, controller reuse, composition



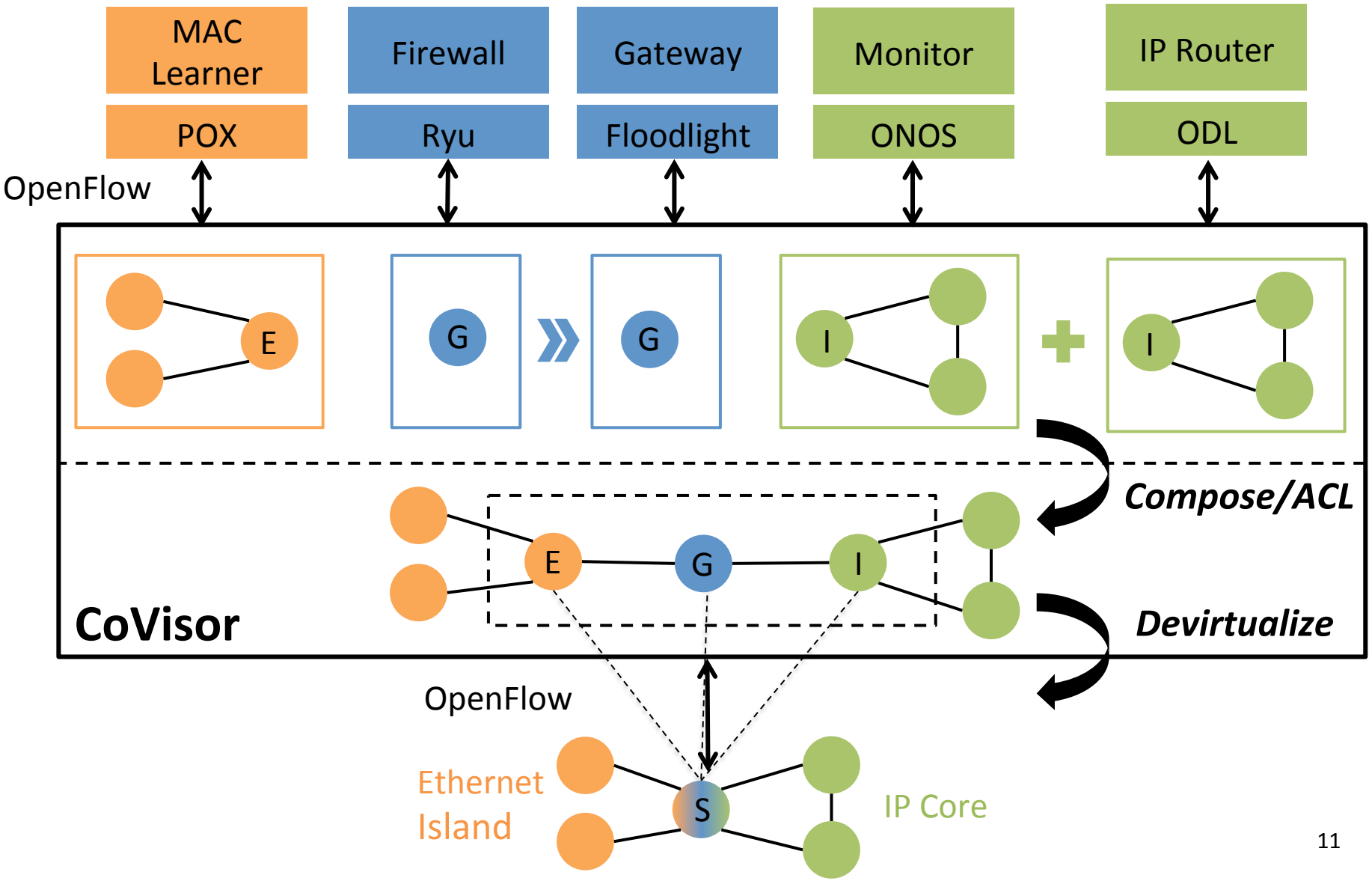
Constraints on Packet Handling Capability

- Protect against buggy or malicious third-party controllers

- Constrains on pattern: header field, match type
  - E.g., MAC learner: srcMAC(Exact), dstMAC(Exact), inport(Exact)

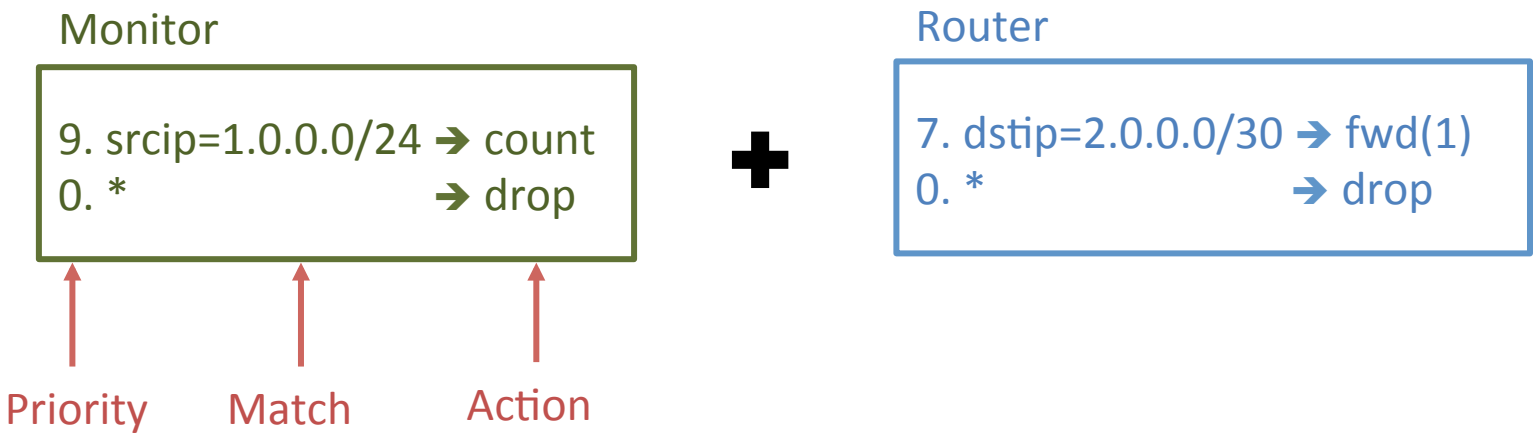
- Constraints on action: actions on matched packets
  - E.g., MAC learner: fwd, drop

CoVisor: A Compositional Hypervisor for SDN



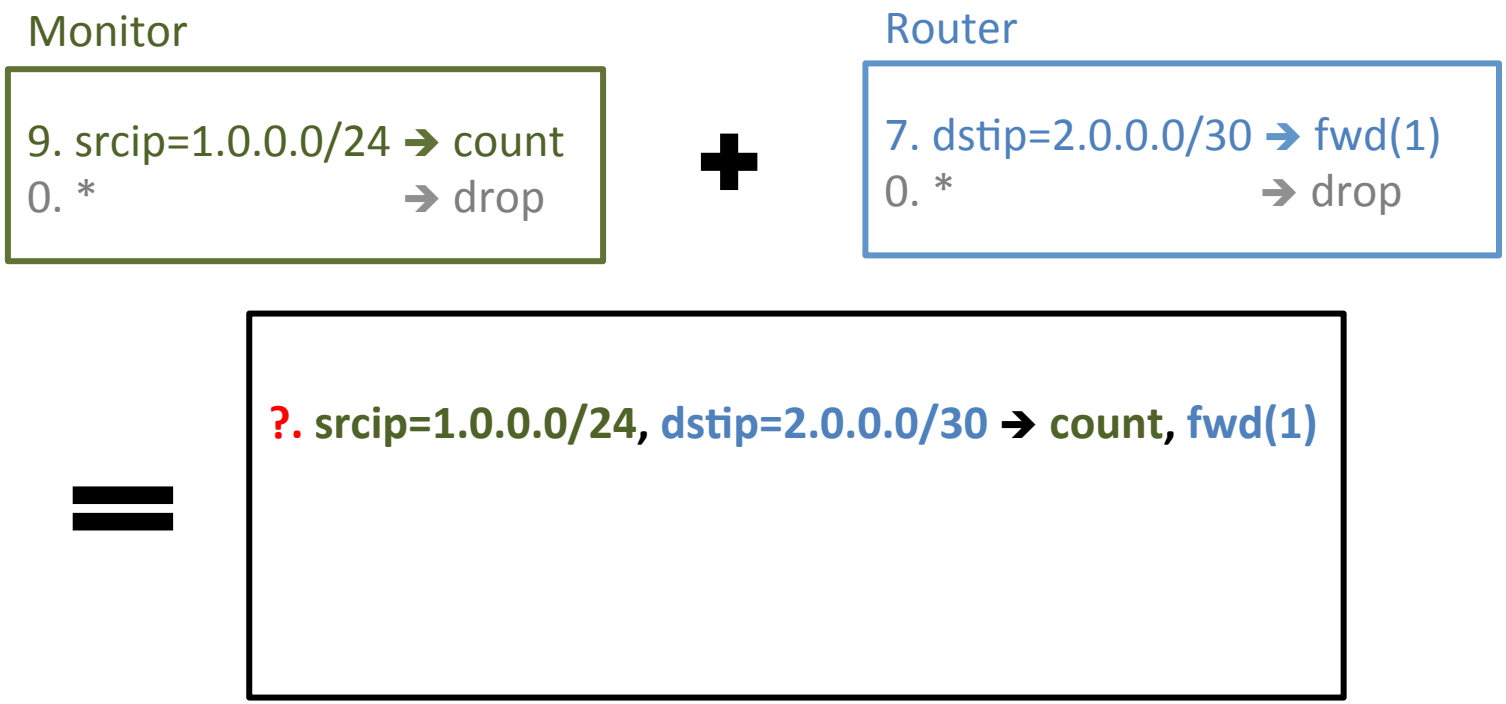
Compiling Policy Composition

- Policy: a list of rules
- Compile policies from controllers to a single policy



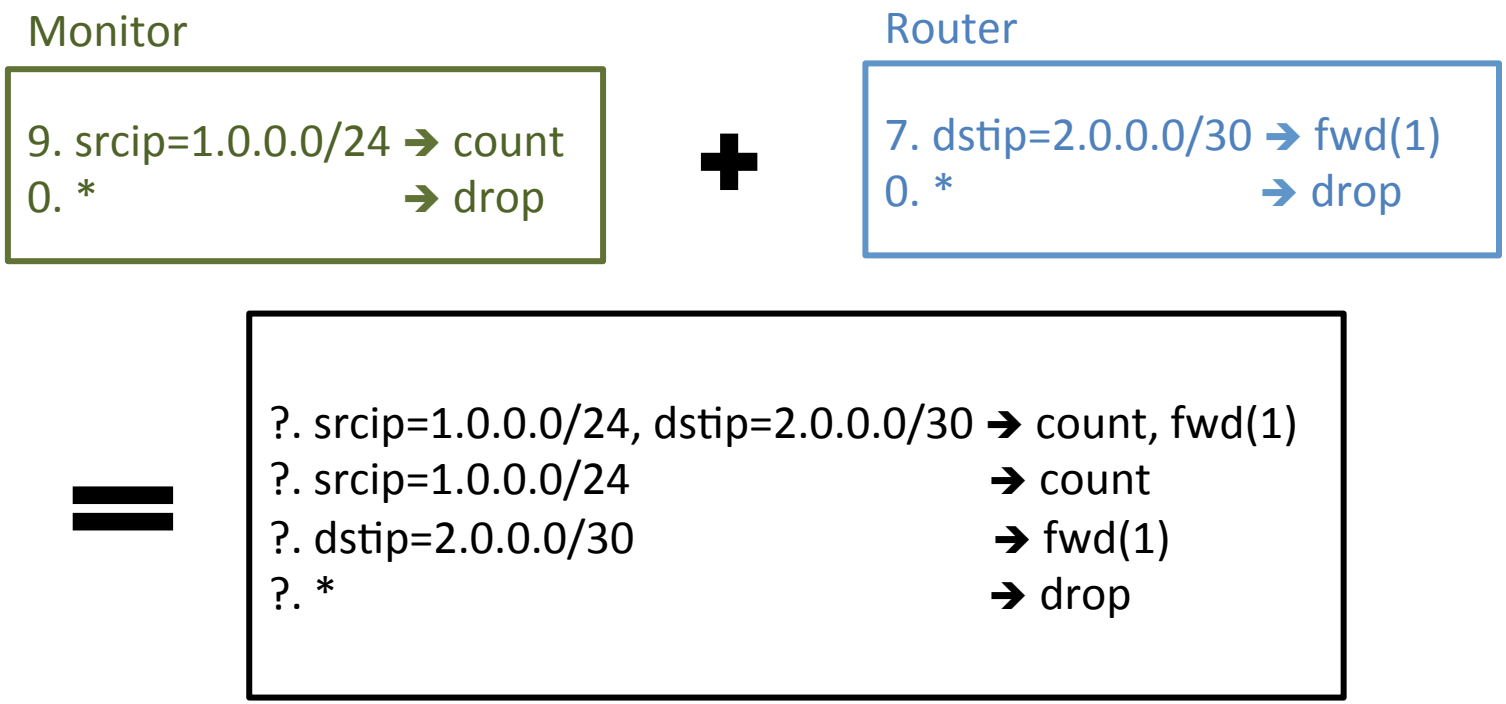
Compiling Policy Composition

- Policy: a list of rules
- Compile policies from controllers to a single policy



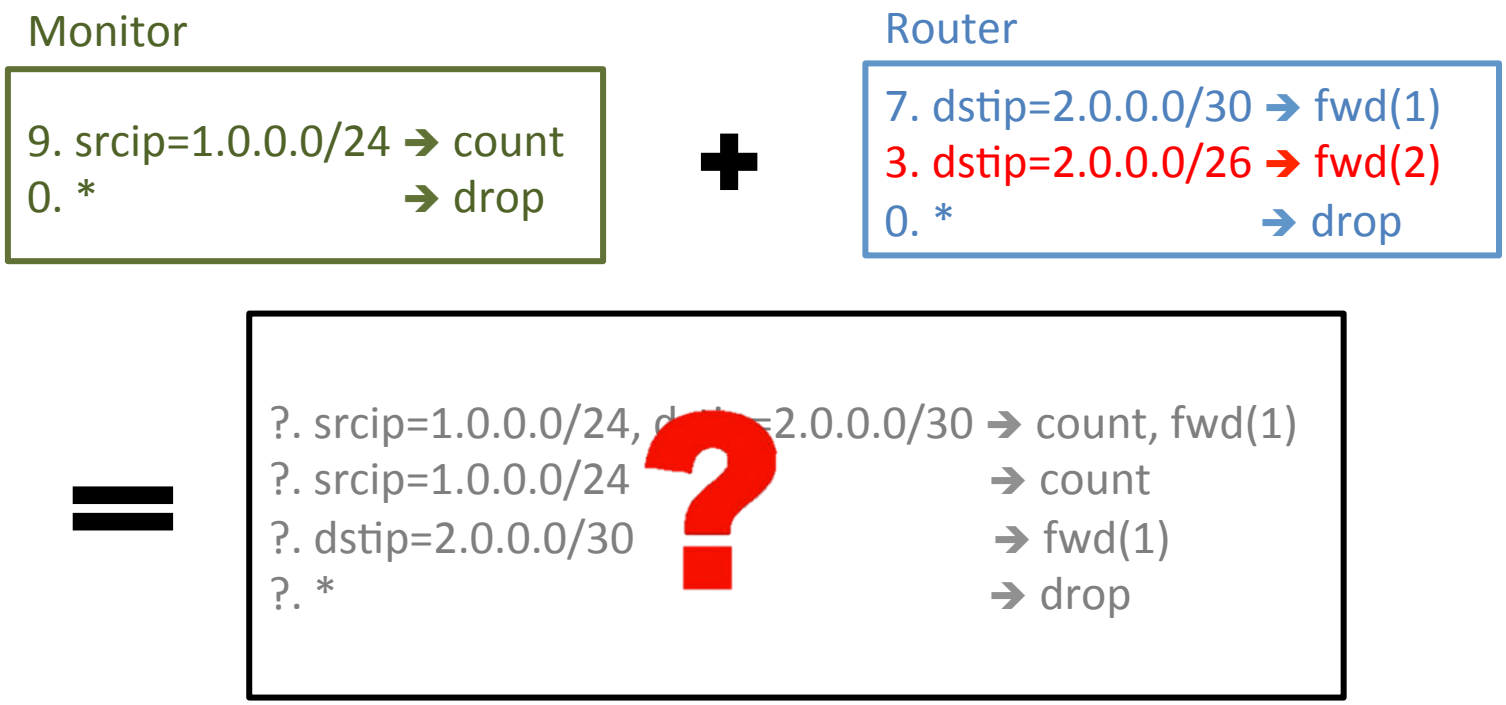
Compiling Policy Composition

- Policy: a list of rules
- Compile policies from controllers to a single policy



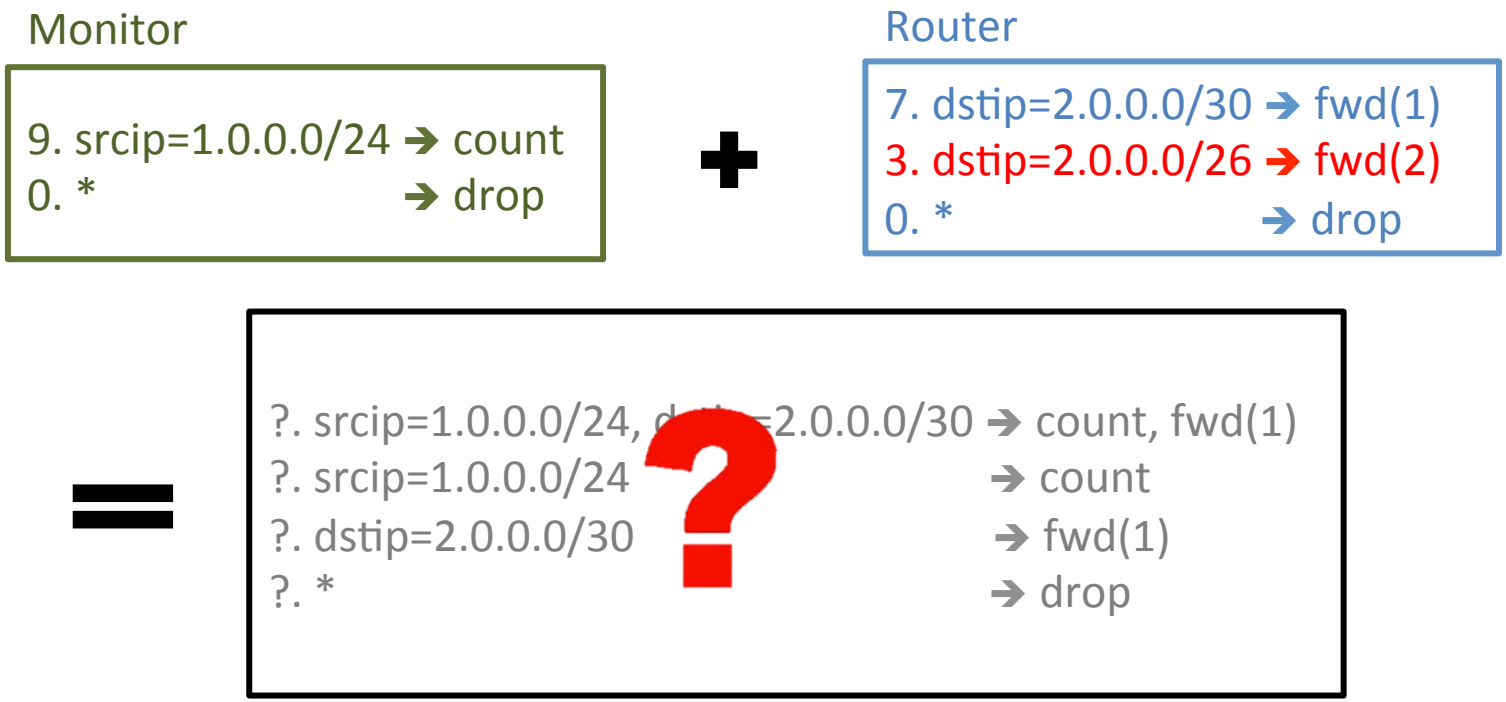
Key challenge: Efficient data plane update

- Controllers continuously update their policies
- Hypervisor recompiles them and update switches



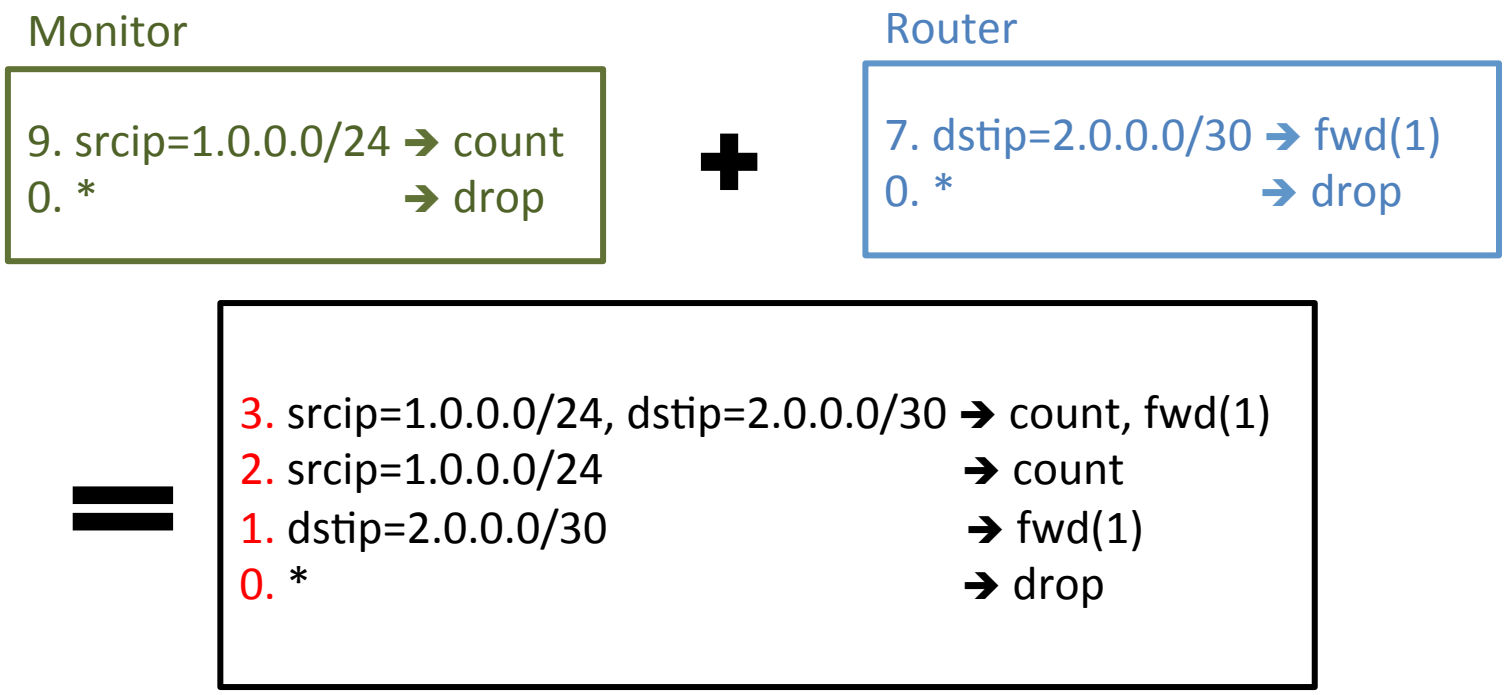
Key challenge: Efficient data plane update

- Computation overhead
  - The computation to recompile the new policy
- Rule-update overhead
  - The rule-updates to update switches to the new policy



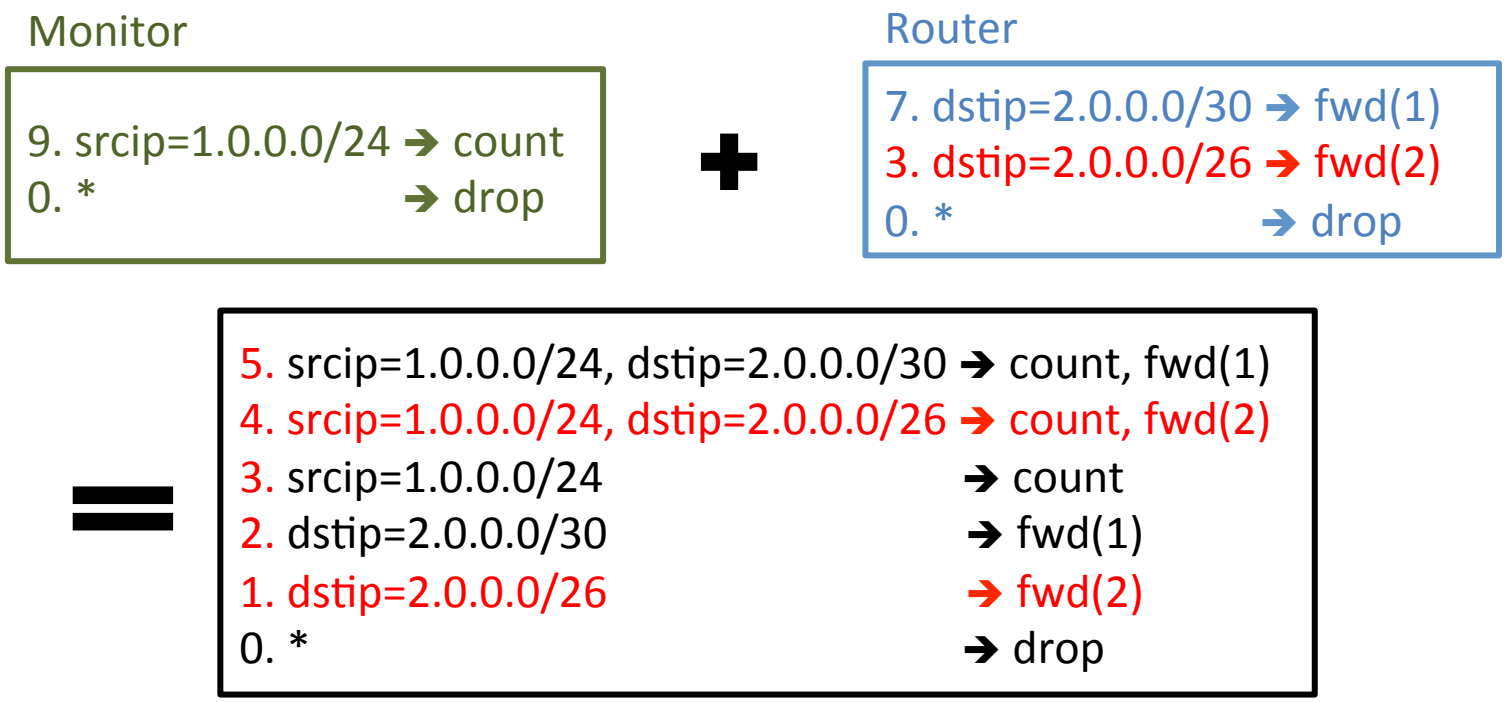
Naïve Solution

- Assign priorities from top to bottom by decrement of 1



Naïve Solution

- Assign priorities from top to bottom by decrement of 1



Naïve Solution

- Assign priorities from top to bottom by decrement of 1

3. srcip=1.0.0.0/24, dstip=2.0.0.0/30 ➔ count, fwd(1)
2. srcip=1.0.0.0/24 ➔ count
1. dstip=2.0.0.0/30 ➔ fwd(1)
0. * ➔ drop

5. srcip=1.0.0.0/24, dstip=2.0.0.0/30 ➔ count, fwd(1)
4. srcip=1.0.0.0/24, dstip=2.0.0.0/26 ➔ count, fwd(2)
3. srcip=1.0.0.0/24 ➔ count
2. dstip=2.0.0.0/30 ➔ fwd(1)
1. dstip=2.0.0.0/26 ➔ fwd(2)
0. * ➔ drop

Computation overhead

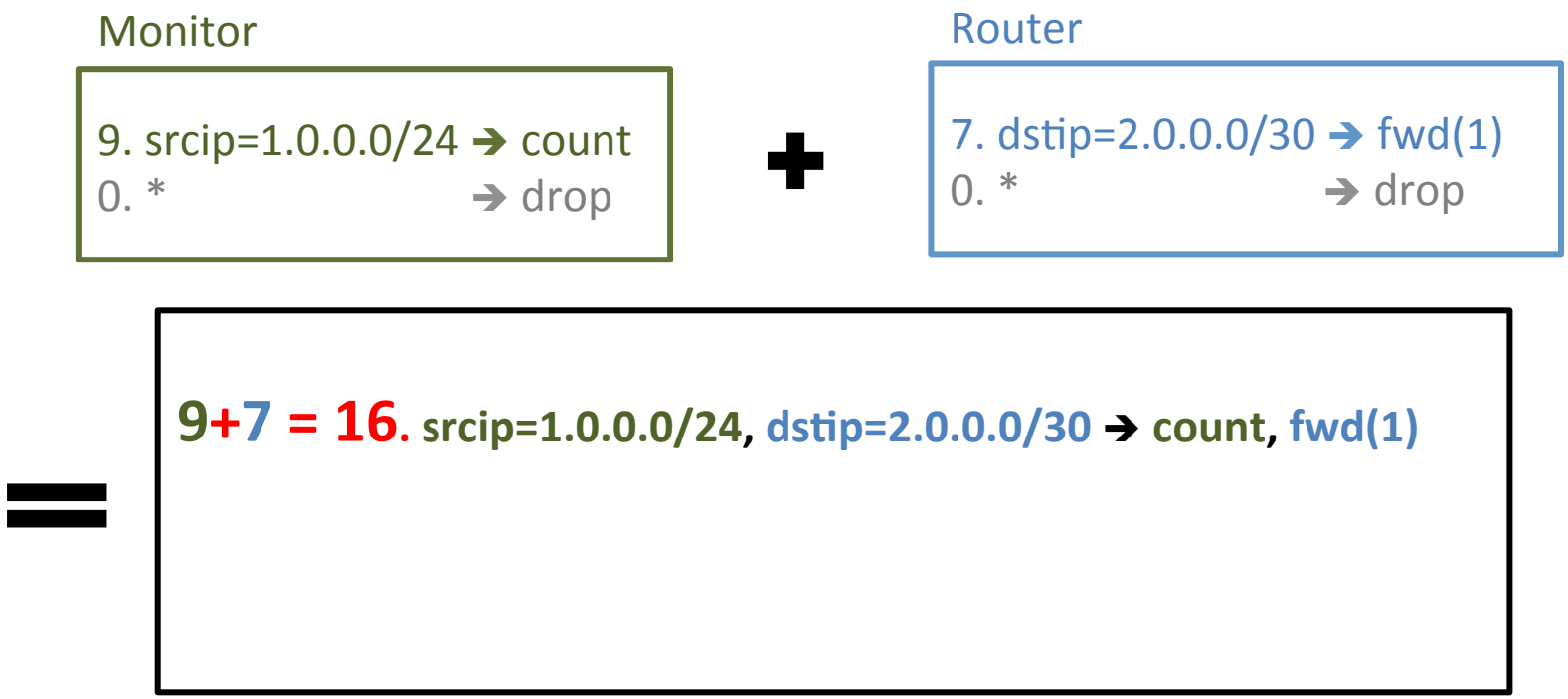
- Recompute the entire switch table and assign priorities

Rule-update overhead

- Only 2 new rules, but 3 more rules change priority

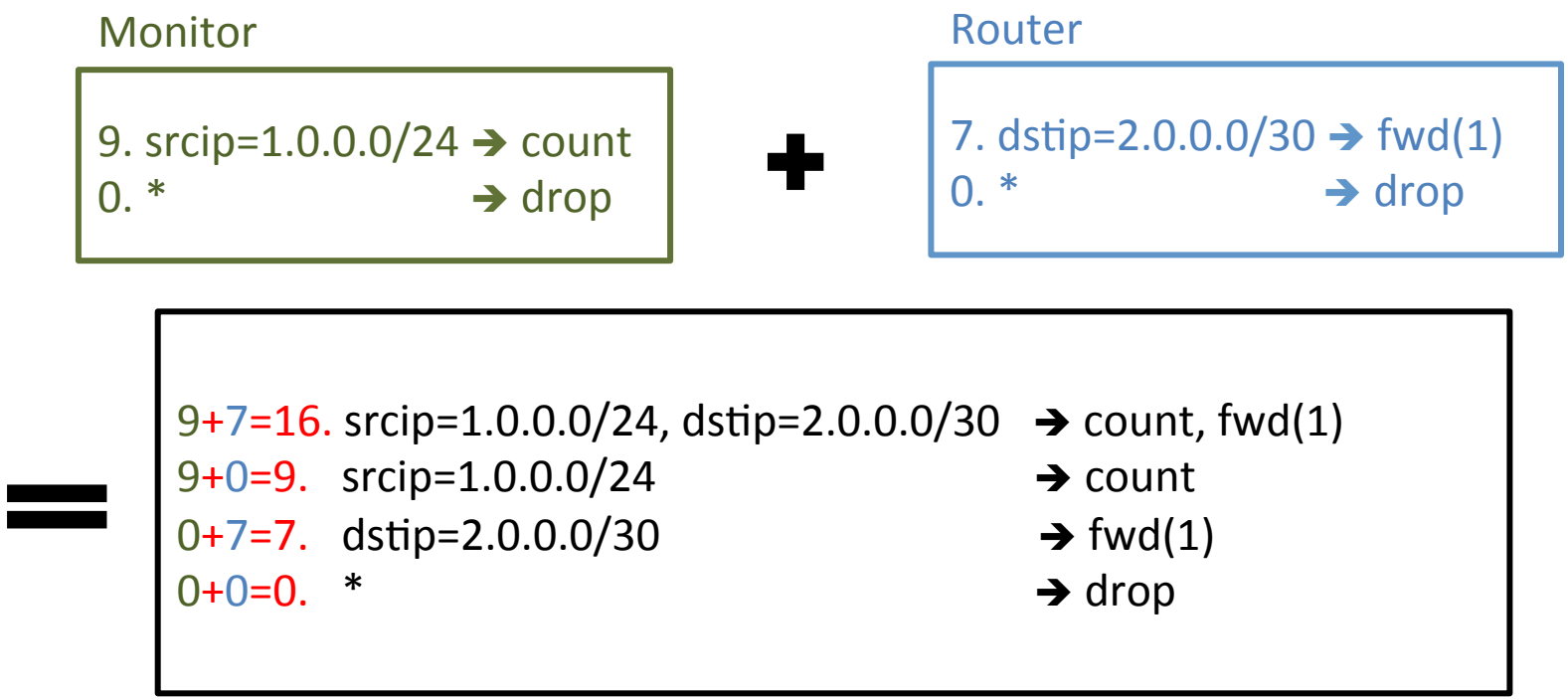
Incremental Update

- Add priorities for parallel composition



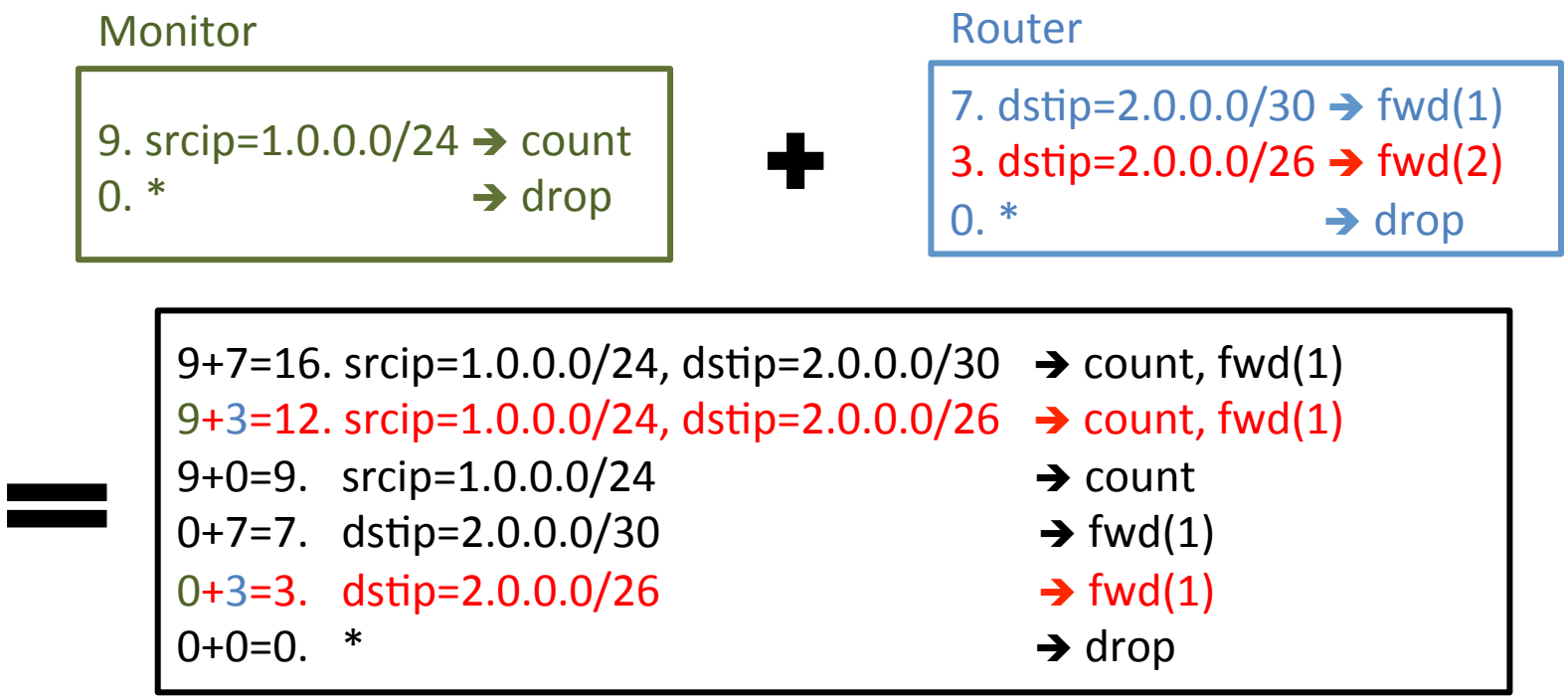
Incremental Update

- Add priorities for parallel composition



Incremental Update

- Add priorities for parallel composition



Incremental Update

- Add priorities for parallel composition

16. srcip=1.0.0.0/24, dstip=2.0.0.0/30 ➔ count, fwd(1)
9. srcip=1.0.0.0/24 ➔ count
7. dstip=2.0.0.0/30 ➔ fwd(1)
0. * ➔ drop

16. srcip=1.0.0.0/24, dstip=2.0.0.0/30 ➔ count, fwd(1)
12. srcip=1.0.0.0/24, dstip=2.0.0.0/26 ➔ count, fwd(2)
9. srcip=1.0.0.0/24 ➔ count
7. dstip=2.0.0.0/30 ➔ fwd(1)
3. dstip=2.0.0.0/26 ➔ fwd(2)
0. * ➔ drop

Computation overhead

- Only compose the new rule with rules in monitor

Rule-update overhead

- Add 2 new rules

Incremental Update

- Add priorities for parallel composition
- Concatenate priorities for sequential composition

Load Balancer

3. srcip=0.0.0.0/2, dstip=3.0.0.0 ➔ dstip=2.0.0.1
1. dstip=3.0.0.0 ➔ dstip=2.0.0.2
0. * ➔ drop

»

Router

1. dstip=2.0.0.1 ➔ fwd(1)
1. dstip=2.0.0.2 ➔ fwd(2)
0. * ➔ drop

=

.

3 >> 1 = 25, srcip=0.0.0.0/2, dstip=3.0.0.0 ➔ dstip=2.0.0.1, **fwd(1)**

011	001
High Bits	Low Bits

.

.

Incremental Update

- Add priorities for parallel composition
- Concatenate priorities for sequential composition

Load Balancer

```
3. srcip=0.0.0.0/2, dstip=3.0.0.0 ➔ dstip=2.0.0.1
1. dstip=3.0.0.0                  ➔ dstip=2.0.0.2
0. *                              ➔ drop
```

»

Router

```
1. dstip=2.0.0.1 ➔ fwd(1)
1. dstip=2.0.0.2 ➔ fwd(2)
0. *             ➔ drop
```

=

```
25. srcip=0.0.0.0/2, dstip=3.0.0.0 ➔ dstip=2.0.0.1, fwd(1)
9. dstip=3.0.0.0                   ➔ dstip=2.0.0.2, fwd(2)
0. *                               ➔ drop
```

Incremental Update

- Add priorities for parallel composition
- Concatenate priorities for sequential composition
- Stack priorities for override composition

Elephant Flow Router

1. srcip=1.0.0.0, dstip=3.0.0.0 ➔ fwd(3)

▷

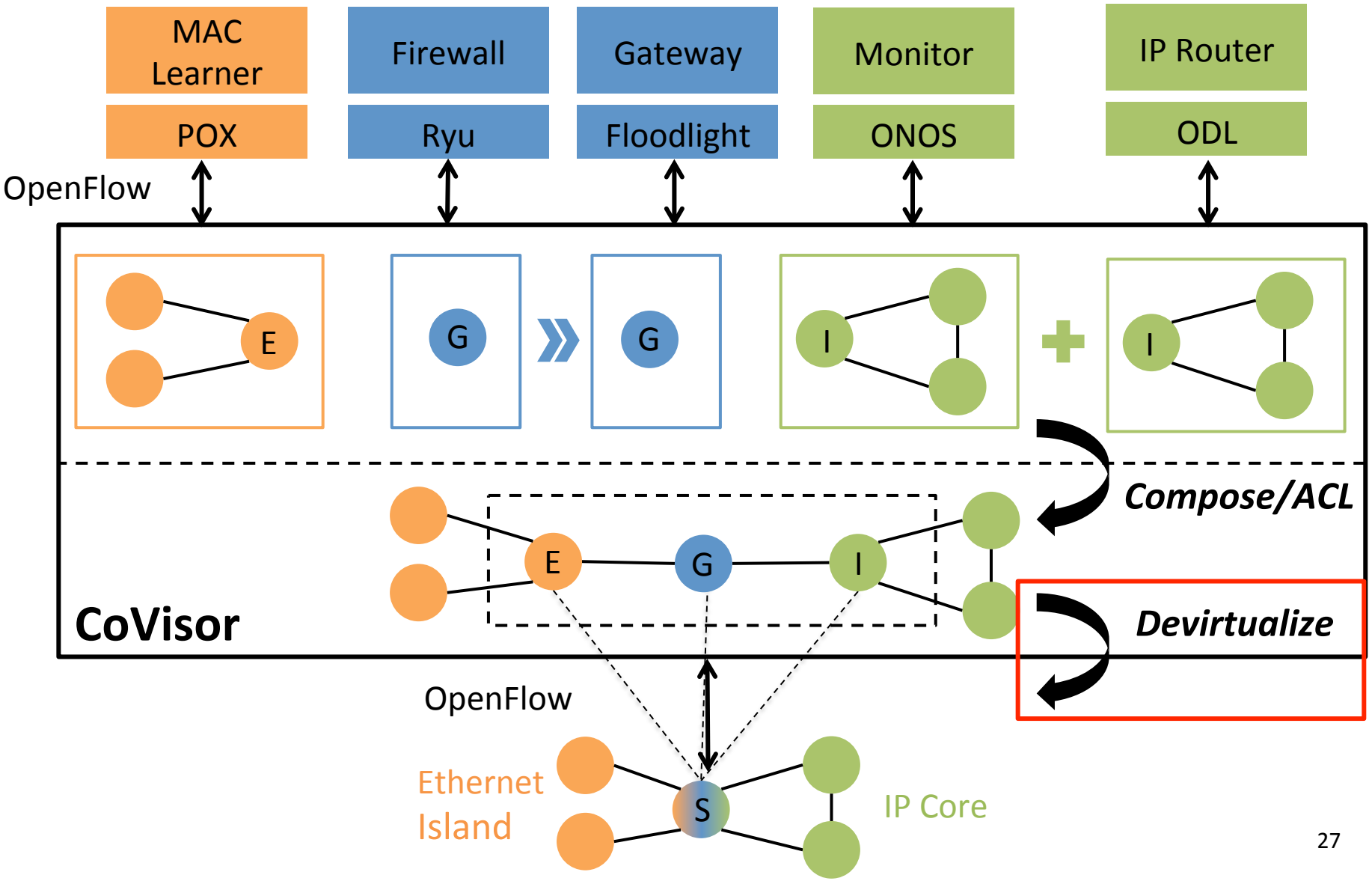
Default Router (Max priority = 8)

1. dstip=2.0.0.1 ➔ fwd(1)
1. dstip=2.0.0.2 ➔ fwd(2)
0. * ➔ drop

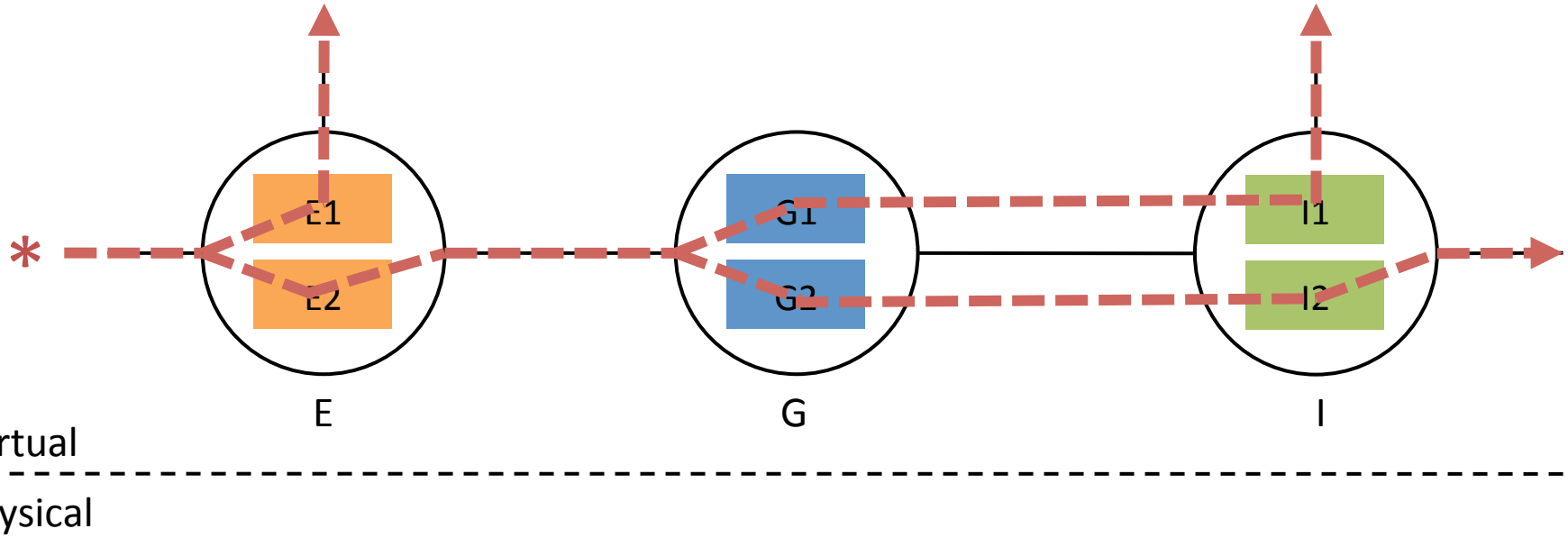
=

1 + 8 = 9. srcip=1.0.0.0, dstip=3.0.0.0 ➔ fwd(3)
1. dstip=2.0.0.1 ➔ fwd(1)
1. dstip=2.0.0.2 ➔ fwd(2)
0. * ➔ drop

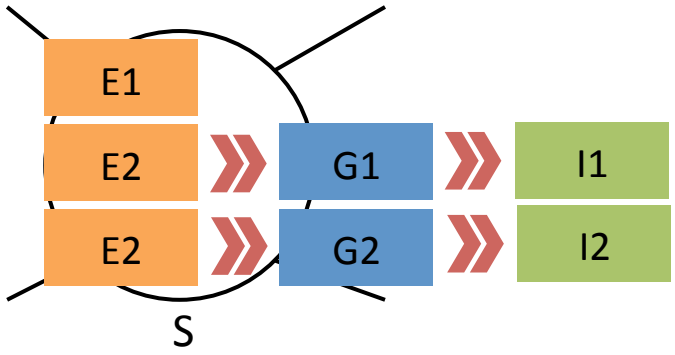
CoVisor: A Compositional Hypervisor for SDN



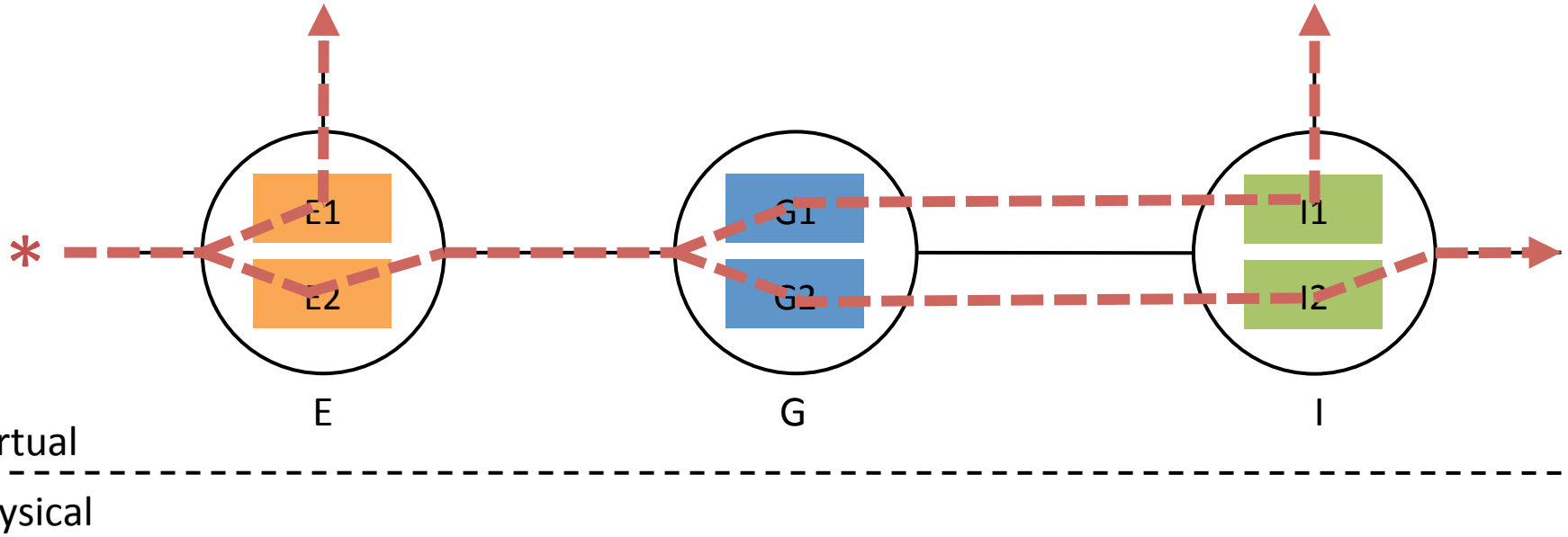
Compiling One-to-Many Virtualization



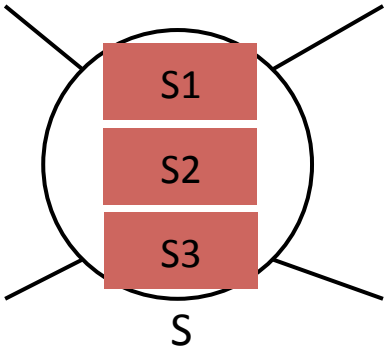
- Symbolic path generation
- Sequential composition



Compiling One-to-Many Virtualization



- Symbolic path generation
- Sequential composition
- Priority augmentation



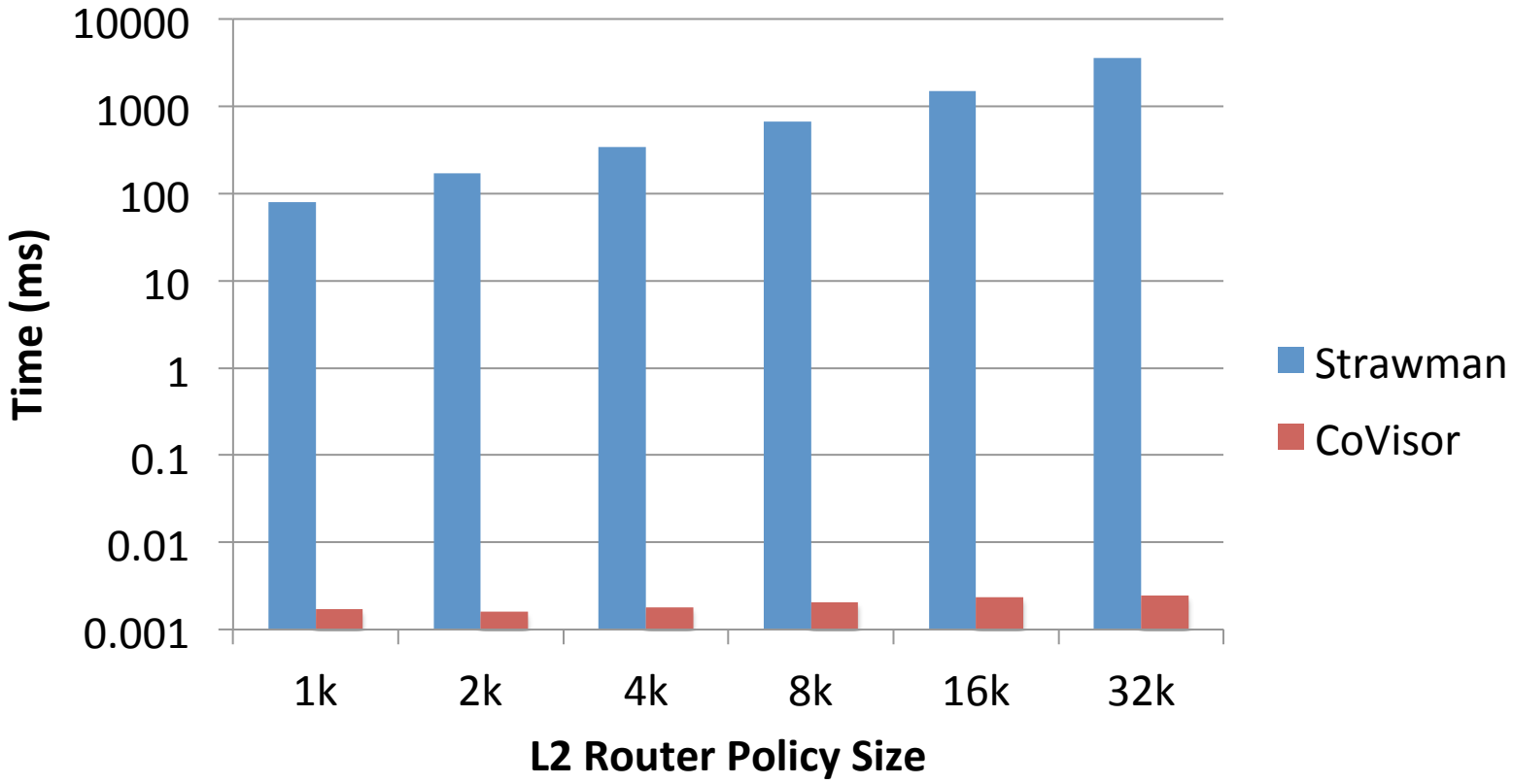
Implementation and Evaluation

- Project website: http://covisor.cs.princeton.edu
  - Code, tutorial, etc.

- Evaluation
  - Parallel composition: L2 Monitor + L2 Router
  - Sequential composition: L3-L4 Firewall >> L3 Router
  - Topology virtualization: gateway between an Ethernet island and an IP core

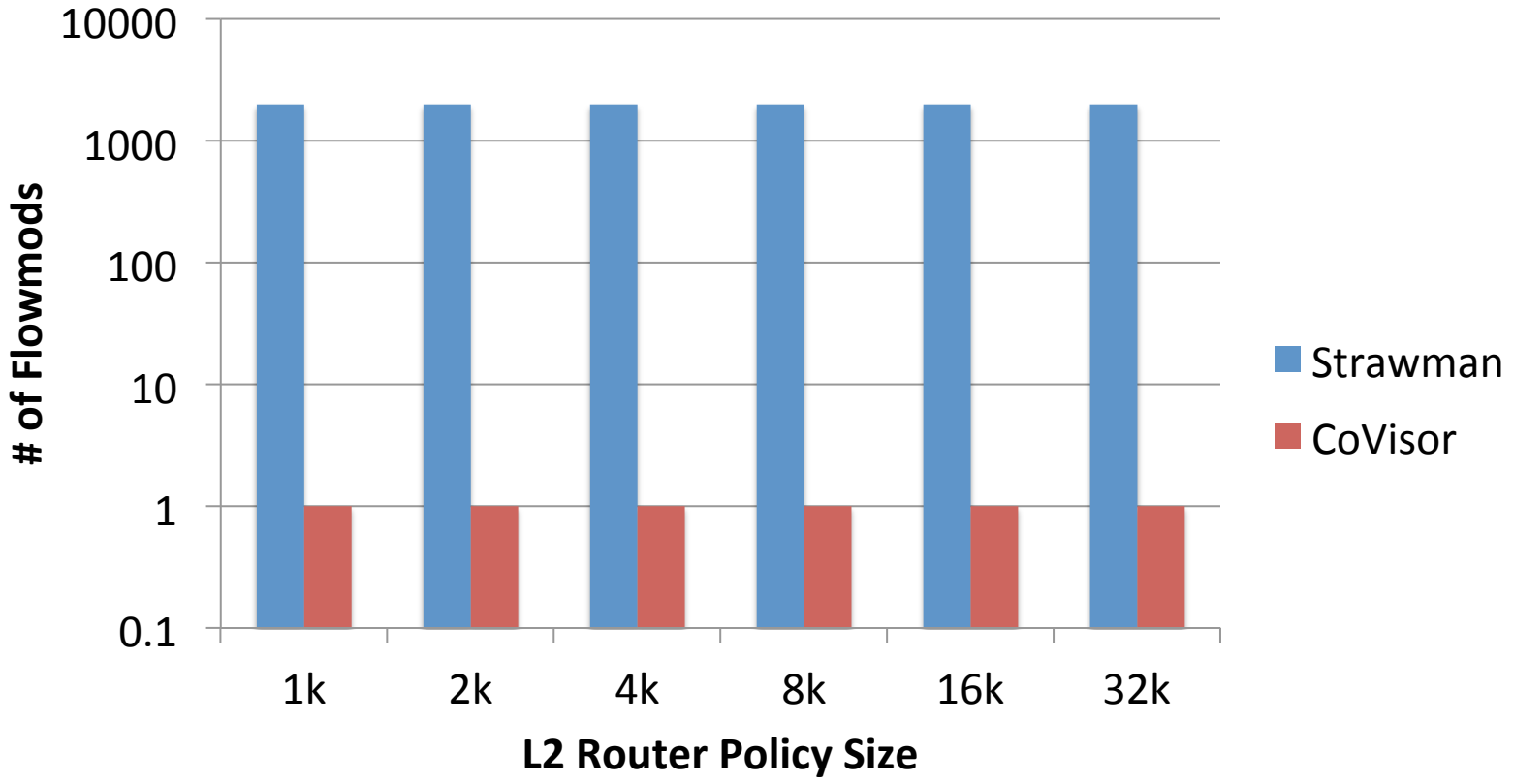
Parallel Composition: L2Monitor + L2 Router

Compilation time of inserting one rule to L2 Monitor Policy



Parallel Composition: L2Monitor + L2 Router

Rule-update overhead of inserting one rule to L2 Monitor Policy



Conclusion

- CoVisor is a compositional hypervisor for software-defined networks
- Provide a clean interface to compose multiple controllers on the same network
- For more, visit http://covisor.cs.princeton.edu
- Ongoing work: integrate into ONOS with ON.LAB

Thanks!