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

MAC Learner + Firewall + Gateway + Monitor + IP Router

Controller

OpenFlow

Network
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

<table>
<thead>
<tr>
<th>MAC Learner</th>
<th>Firewall</th>
<th>Gateway</th>
<th>Monitor</th>
<th>IP Router</th>
</tr>
</thead>
<tbody>
<tr>
<td>POX</td>
<td>Ryu</td>
<td>Floodlight</td>
<td>ONOS</td>
<td>ODL</td>
</tr>
</tbody>
</table>

Network
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
  
  ![Parallel operator example](image)

- **Sequential operator (>>):** two controllers process packets one after another
  
  ![Sequential operator example](image)

- **Override operator (▷):** one controller chooses to act or defer the process to another controller
  
  ![Override operator example](image)

- **Use multiple operators**
  
  ![Multiple operators example](image)
Constraints on Topology Visibility

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

Many-to-One

Virtual
- Firewall

Physical

One-to-Many

MAC Learner
Gateway
IP Router

Ethernet Island
IP Core
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

**CoVisor**

- MAC Learner
- Firewall
- Gateway
- Monitor
- IP Router

**OpenFlow**

- POX
- Ryu
- Floodlight
- ONOS
- ODL

**Compose/ACL**

**Devirtualize**

**OpenFlow**

**Ethernet Island**

**IP Core**
Compiling Policy Composition

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

Monitor

- 9. srcip=1.0.0.0/24 ➔ count
- 0. * ➔ drop

Router

- 7. dstip=2.0.0.0/30 ➔ fwd(1)
- 0. * ➔ drop
Compiling Policy Composition

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

Monitor

9. srcip=1.0.0.0/24 ➔ count
0. * ➔ drop

Router

7. dstip=2.0.0.0/30 ➔ fwd(1)
0. * ➔ drop

?. srcip=1.0.0.0/24, dstip=2.0.0.0/30 ➔ count, fwd(1)
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

**Monitor**
- 9. srcip=1.0.0.0/24 \(\rightarrow\) count
- 0. * \(\rightarrow\) drop

**Router**
- 7. dstip=2.0.0.0/30 \(\rightarrow\) fwd(1)
- 3. dstip=2.0.0.0/26 \(\rightarrow\) fwd(2)
- 0. * \(\rightarrow\) drop

\[ \text{? \quad srcip=1.0.0.0/24, dstip=2.0.0.0/30 \rightarrow count, fwd(1)} \]
\[ \text{? \quad srcip=1.0.0.0/24 \rightarrow count} \]
\[ \text{? \quad dstip=2.0.0.0/30 \rightarrow fwd(1)} \]
\[ \text{? \quad * \rightarrow drop} \]
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

---

**Monitor**

9. srcip=1.0.0.0/24 ➔ count
0. * ➔ drop

**Router**

7. dstip=2.0.0.0/30 ➔ fwd(1)
3. dstip=2.0.0.0/26 ➔ fwd(2)
0. * ➔ drop

|=|
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

Update

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

Computation overhead
• Recompute the entire switch table and assign priorities

Rule-update overhead
• Only 2 new rules, but 3 more rules change priority

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
Incremental Update

• Add priorities for parallel composition

Monitor

9. srcip=1.0.0.0/24 ➔ count
0. * ➔ drop

Router

7. dstip=2.0.0.0/30 ➔ fwd(1)
0. * ➔ drop

9+7 = 16. srcip=1.0.0.0/24, dstip=2.0.0.0/30 ➔ count, fwd(1)
Incremental Update

- Add priorities for parallel composition

Monitor

9. srcip=1.0.0.0/24 → count
0. * → drop

Router

7. dstip=2.0.0.0/30 → fwd(1)
0. * → drop

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

• Add priorities for parallel composition

Monitor

9. srcip=1.0.0.0/24 ➔ count
0. * ➔ drop

Router

7. dstip=2.0.0.0/30 ➔ fwd(1)
3. dstip=2.0.0.0/26 ➔ fwd(2)
0. * ➔ drop

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

0+0=0. * ➔ drop
**Incremental Update**

- Add priorities for parallel composition

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>7</td>
<td>dstip=2.0.0.0/30</td>
</tr>
<tr>
<td>9</td>
<td>srcip=1.0.0.0/24</td>
</tr>
<tr>
<td>12</td>
<td>srcip=1.0.0.0/24, dstip=2.0.0.0/26</td>
</tr>
<tr>
<td>16</td>
<td>srcip=1.0.0.0/24, dstip=2.0.0.0/30</td>
</tr>
</tbody>
</table>

**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. \text{srcip}=0.0.0.0/2, \text{dstip}=3.0.0.0 \rightarrow \text{dstip}=2.0.0.1
1. \text{dstip}=3.0.0.0 \rightarrow \text{dstip}=2.0.0.2
0. * \rightarrow \text{drop}

Router

1. \text{dstip}=2.0.0.1 \rightarrow \text{fwd}(1)
1. \text{dstip}=2.0.0.2 \rightarrow \text{fwd}(2)
0. * \rightarrow \text{drop}

\begin{array}{c|c}
\text{High Bits} & \text{Low Bits} \\
\hline
011 & 001 \\
\end{array}

3 \gg 1 = 25, \text{srcip}=0.0.0.0/2, \text{dstip}=3.0.0.0 \rightarrow \text{dstip}=2.0.0.1, \text{fwd}(1)
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 $\rightarrow$ dstip=2.0.0.1
- 1. dstip=3.0.0.0 $\rightarrow$ dstip=2.0.0.2
- 0. * $\rightarrow$ drop

Router
- 1. dstip=2.0.0.1 $\rightarrow$ fwd(1)
- 1. dstip=2.0.0.2 $\rightarrow$ fwd(2)
- 0. * $\rightarrow$ drop

25. srcip=0.0.0.0/2, dstip=3.0.0.0 $\rightarrow$ dstip=2.0.0.1, fwd(1)
9. dstip=3.0.0.0 $\rightarrow$ dstip=2.0.0.2, fwd(2)
0. * $\rightarrow$ 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

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

OpenFlow

CoVisor

Compose/ACL
Devirtualize

OpenFlow
Ethernet Island
IP Core
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

  - 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

![Bar chart showing compilation time for different L2 Router policy sizes]

- Y-axis: Time (ms)
- X-axis: L2 Router Policy Size (1k, 2k, 4k, 8k, 16k, 32k)

- Strawman
- CoVisor
Parallel Composition: L2Monitor + L2 Router

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

![Bar graph showing rule-update overhead over different policy sizes for Strawman and CoVisor. The x-axis represents L2 Router Policy Size with values 1k, 2k, 4k, 8k, 16k, and 32k. The y-axis represents the number of Flowmods with values ranging from 0.1 to 10000. The blue bars represent Strawman, and the red bars represent CoVisor.](image)

- Strawman
- CoVisor
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!