Incremental Update for a Compositional SDN Hypervisor

Xin Jin
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
Slicing is Not Enough

• Controllers work on disjoint parts of traffic
  – Useful for multi-tenancy
• But we want them to collaboratively work on the same traffic

![Network Monitoring Floodlight POX Load Balancing Ryu](image)
Our Solution: Compositional Hypervisor

- A transparent layer between switches and controllers
- Combine controllers with Frenetic-like composition operators
  - Parallel operator (+)
  - Sequential operator (>>)

![Diagram showing the integration of controllers and components using OpenFlow]

- POX (Load Balancing)
- Ryu (Routing)
- Floodlight (Monitoring)

Compositional Hypervisor
Policy Compilation

- Policy: a list of rules
- Each controller outputs a policy
- Hypervisor compiles them to a single policy

---

### Monitoring

<table>
<thead>
<tr>
<th>Priority</th>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>9. srcip=1.0.0.0/24</td>
<td>count</td>
<td>drop</td>
</tr>
<tr>
<td>0. *</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Routing

<table>
<thead>
<tr>
<th>Priority</th>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>7. dstip=2.0.0.0/30</td>
<td>fwd(1)</td>
<td></td>
</tr>
<tr>
<td>0. *</td>
<td></td>
<td>drop</td>
</tr>
</tbody>
</table>
Policy Compilation

- **Policy**: a list of rules
- Each controller outputs a policy
- Hypervisor compiles them to a single policy

**Monitoring**

9. srcip=1.0.0.0/24 \(\Rightarrow\) count
0. * \(\Rightarrow\) drop

**Routing**

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

\[ \text{srcip}=1.0.0.0/24, \text{dstip}=2.0.0.0/30 \Rightarrow \text{count, fwd}(1) \]
Policy Compilation

- Policy: a list of rules
- Each controller outputs a policy
- Hypervisor compiles them to a single policy

Monitoring

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

Routing

| 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) | 
| ?. srcip=1.0.0.0/24 ➔ count | 
| ?. dstip=2.0.0.0/30 ➔ fwd(1) | 
| ?. * ➔ drop |
Key challenge: Efficient data plane update

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

Monitoring

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

Routing

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

= 8. srcip=1.0.0.0/24, dstip=2.0.0.0/30 ➔ count, fwd(1)
8. srcip=1.0.0.0/24 ➔ count
8. dstip=2.0.0.0/30 ➔ fwd(1)
8. * ➔ 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

---

**Monitoring**

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

**Routing**

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

= ?

?. srcip=1.0.0.0/24, dstip=2.0.0.0/30 → count, fwd(1)
?. srcip=1.0.0.0/24
?. dstip=2.0.0.0/30
?. * → count
→ fwd(1)
→ drop
Naïve Solution

- Assign priorities from top to bottom by decrement of 1

Monitoring

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

Routing

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

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
Naïve Solution

- Assign priorities from top to bottom by decrement of 1

Monitoring

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

Routing

+ 7. dstip=2.0.0.0/30 ➔ fwd(1)
3. dstip=2.0.0.0/26 ➔ fwd(2)
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
# Naïve Solution

- Assign priorities from top to bottom by decrement of 1

<table>
<thead>
<tr>
<th>Rule</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Action</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>srcip=1.0.0.0/24</td>
<td>dstip=2.0.0.0/30</td>
<td>count</td>
<td>fwd(1)</td>
</tr>
<tr>
<td>2</td>
<td>srcip=1.0.0.0/24</td>
<td></td>
<td>count</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>dstip=2.0.0.0/30</td>
<td></td>
<td>fwd(1)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>*</td>
<td>dstip=2.0.0.0/30</td>
<td>drop</td>
<td></td>
</tr>
</tbody>
</table>

**Computation overhead**
- Recompute the whole 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

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

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

Monitoring

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

Routing

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

**Monitoring**

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

**Routing**

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
Incremental Update

- Add priorities for parallel composition

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.</td>
<td>srcip=1.0.0.0/24, dstip=2.0.0.0/30 ➔ count, fwd(1)</td>
</tr>
<tr>
<td>9.</td>
<td>srcip=1.0.0.0/24 ➔ count</td>
</tr>
<tr>
<td>7.</td>
<td>dstip=2.0.0.0/30 ➔ fwd(1)</td>
</tr>
<tr>
<td>0.</td>
<td>* ➔ drop</td>
</tr>
</tbody>
</table>

Computation overhead
- Only compose the new rule with rules in monitoring

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.</td>
<td>srcip=1.0.0.0/24, dstip=2.0.0.0/30 ➔ count, fwd(1)</td>
</tr>
<tr>
<td>12.</td>
<td>srcip=1.0.0.0/24, dstip=2.0.0.0/26 ➔ count, fwd(2)</td>
</tr>
<tr>
<td>9.</td>
<td>srcip=1.0.0.0/24 ➔ count</td>
</tr>
<tr>
<td>7.</td>
<td>dstip=2.0.0.0/30 ➔ fwd(1)</td>
</tr>
<tr>
<td>3.</td>
<td>dstip=2.0.0.0/26 ➔ fwd(2)</td>
</tr>
<tr>
<td>0.</td>
<td>* ➔ drop</td>
</tr>
</tbody>
</table>

Rule-update overhead
- Add 2 new rules
Incremental Update

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

Load Balancing

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

Routing

1. dstip=2.0.0.1 $\Rightarrow$ fwd(1)
1. dstip=2.0.0.2 $\Rightarrow$ fwd(2)
0. * $\Rightarrow$ drop

3 $\gg 1 = 25$, srcip=0.0.0.0/2, dstip=3.0.0.0 $\Rightarrow$ dstip=2.0.0.1, fwd(1)
Incremental Update

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

Load Balancing

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

Routing

1. dstip=2.0.0.1 ➞ fwd(1)
1. dstip=2.0.0.2 ➞ fwd(2)
0. * ➞ drop
Incremental Update

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

### Load Balancing

<table>
<thead>
<tr>
<th>Step</th>
<th>srcip</th>
<th>dstip</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>dstip=3.0.0.0</td>
<td>dstip=2.0.0.2</td>
</tr>
<tr>
<td>2.</td>
<td>dstip=3.0.0.0</td>
<td>dstip=2.0.0.3</td>
</tr>
<tr>
<td>3.</td>
<td>dstip=3.0.0.0</td>
<td>dstip=2.0.0.1</td>
</tr>
<tr>
<td>0.</td>
<td>*</td>
<td>drop</td>
</tr>
</tbody>
</table>

### Routing

<table>
<thead>
<tr>
<th>Step</th>
<th>dstip</th>
<th>fwd</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>dstip=2.0.0.1</td>
<td>(1)</td>
</tr>
<tr>
<td>1.</td>
<td>dstip=2.0.0.2</td>
<td>(2)</td>
</tr>
<tr>
<td>1.</td>
<td>dstip=2.0.0.3</td>
<td>(3)</td>
</tr>
<tr>
<td>0.</td>
<td>*</td>
<td>drop</td>
</tr>
</tbody>
</table>

25. srcip=0.0.0.0/2, dstip=3.0.0.0 \(\rightarrow\) dstip=2.0.0.1, fwd(1)
17. srcip=0.0.0.0/1, dstip=3.0.0.0 \(\rightarrow\) dstip=2.0.0.3, fwd(3)
9. dstip=3.0.0.0 \(\rightarrow\) dstip=2.0.0.2, fwd(2)
0. * \(\rightarrow\) drop
5. CONCLUSION
A compositional network hypervisor can simplify network management by allowing different applications written in different languages, or on different platforms, to work together to process the same traffic. A major challenge that arises when implementing such a hypervisor involves developing algorithms for correct, efficient, and real-time processing of rule updates from different applications. This paper presents a novel algorithm for such updates. Moreover, we analyze the correctness of our algorithm and show experimentally that it can significantly reduce update overhead as compared to the strawman solution. There are many interesting topics for future work, such as adding support for rule timeouts, adding support for OpenFlow 1.3 [14], and integrating the incremental update mechanism with consistent update mechanisms to enable efficient, network-wide updates that preserve consistency properties [15, 16, 17].

6. ACKNOWLEDGMENTS
We would like to thank Mojgan Ghasemi, Nanxi Kang, Naga Katta, Srinivas Narayana, Cole Schlesinger, and the anonymous HotSDN reviewers for their feedback on earlier versions of this paper. This work was supported by NSF grant TC-1111520 and DARPA grant MRC-007692-001.

7. REFERENCES

Monitoring + Routing

Load balancing >> Routing
Evaluation

Monitoring + Routing

Reduce computation overhead by 4x, rule updates by 5x

Load balancing >> Routing
Conclusion

• Compositional network hypervisor

• Novel algorithm to efficiently update the data plane

• Ongoing work: prototype in OpenVirteX
Thanks!