SDNRacer: Detecting Concurrency Violations in Software-Defined Networks

Jeremie Miserez, Pavol Bielik, Ahmed El-Hassany, Laurent Vanbever, Martin Vechev

ETH Zürich

OPEN NETWORKING SUMMIT 2015, JUNE 14-18
SDN Concurrency

- Internal Host
- Switch
- External Host
if ip_src == H1:
    flow_mod(10, ip_src, ip_dst, fwd)
    flow_mod(10, ip_dst, ip_src, fwd)
    packet_out(pkt, sw)
else:
    flow_mod(15, ip_src, ip_dst, drop)
if ip_src == H1:
    flow_mod(10, ip_src, ip_dst, fwd)
    flow_mod(10, ip_dst, ip_src, fwd)
    packet_out(pkt, sw)
else:
    flow_mod(15, ip_src, ip_dst, drop)
if ip_src == H1:
    flow_mod(10, ip_src, ip_dst, fwd)
    flow_mod(10, ip_dst, ip_src, fwd)
    packet_out(pkt, sw)
else:
    flow_mod(15, ip_src, ip_dst, drop)
if ip_src == H1:
    flow_mod(10, ip_src, ip_dst, fwd)
    flow_mod(10, ip_dst, ip_src, fwd)
    packet_out(pkt, sw)
else:
    flow_mod(15, ip_src, ip_dst, drop)
if ip_src == H1:
    flow_mod(10, ip_src, ip_dst, fwd)
    flow_mod(10, ip_dst, ip_src, fwd)
    packet_out(pkt, sw)
else:
    flow_mod(15, ip_src, ip_dst, drop)
if ip_src == H1:
    flow_mod(10, ip_src, ip_dst, fwd)
    flow_mod(10, ip_dst, ip_src, fwd)
    packet_out(pkt, sw)
else:
    flow_mod(15, ip_src, ip_dst, drop)

Flow table
R0:  1, * → * , ctrl
R1: 10, H1 → H2, fwd
R2: 10, H2 → H1, fwd
if ip_src == H1:
    flow_mod(10, ip_src, ip_dst, fwd)
    flow_mod(10, ip_dst, ip_src, fwd)
    packet_out(pkt, sw)
else:
    flow_mod(15, ip_src, ip_dst, drop)

Flow table
R0:  1, * → *, ctrl
R1: 10, H1 → H2, fwd
R2: 10, H2 → H1, fwd
if ip_src == H1:
    flow_mod(10, ip_src, ip_dst, fwd)
    flow_mod(10, ip_dst, ip_src, fwd)
    packet_out(pkt, sw)
else:
    flow_mod(15, ip_src, ip_dst, drop)

Flow table
R0:  1, * → * , ctrl
R1: 10, H1 → H2, fwd
R2: 10, H2 → H1, fwd
if ip_src == H1:
    flow_mod(10, ip_src, ip_dst, fwd)
    flow_mod(10, ip_dst, ip_src, fwd)
    packet_out(pkt, sw)
else:
    flow_mod(15, ip_src, ip_dst, drop)
if ip_src == H1:
    flow_mod(10, ip_src, ip_dst, fwd)
    flow_mod(10, ip_dst, ip_src, fwd)
    packet_out(pkt, sw)
else:
    flow_mod(15, ip_src, ip_dst, drop)
if ip_src == H1:
    flow_mod(10, ip_src, ip_dst, fwd)
    flow_mod(10, ip_dst, ip_src, fwd)
    packet_out(pkt, sw)
else:
    flow_mod(15, ip_src, ip_dst, drop)
if ip_src == H1:
    flow_mod(10, ip_src, ip_dst, fwd)
    flow_mod(10, ip_dst, ip_src, fwd)
    packet_out(pkt, sw)
else:
    flow_mod(15, ip_src, ip_dst, drop)
Stateful firewall allows packets to be dropped.
Detecting SDN Asynchrony

Controller

flow_mod(...)
flow_mod(...)
flow_mod(...)
writes

Switch

R0: 1, * → *, ctrl
R1: 10, H1 → H2, fwd
R2: 10, H2 → H1, fwd

rcv(...)
rcv(...)
rcv(...)
Detecting SDN Asynchrony

Detect Data Races on Switch Flow Table

Data Race: two unordered events accessing the same flow table, where at least one of the events is a write.

Controller

Switch

Host

Detect Data Races on Switch Flow Table

\[\text{Detect Data Races on Switch Flow Table}\]

Data Race: two unordered events accessing the same flow table, where at least one of the events is a write.
How to capture asynchrony in SDN?

Define high level events:

- Flow table write:
  - FLOW_MOD (ADD)
  - FLOW_MOD (DELETE)
  - FLOW_MOD (MODIFY)
- Flow table read:
  - Lookup rule for packet
- Send packet/message
- Receive packet/message

- Switch → Controller:
  - PACKET_IN
  - FLOW_REMOVED
  - PORT_STATUS
- Controller → Switch:
  - FLOW_MOD
  - PORT_MOD
  - PACKET_OUT
  - BARRIER_REQUEST
How to capture asynchrony in SDN?
Define happens-before relation between the events.
How to capture asynchrony in SDN?

Define happens-before relation between the events.
How to capture asynchrony in SDN?
Define happens-before relation between the events.
How to capture asynchrony in SDN?
Define happens-before relation between the events.

Instrumentation to track packet and message identities
Which high-level events are conflicting?
SDNRacer: Detecting Concurrency Violations

Which high-level events are conflicting?

Phrased as a graph connectivity query

SDN Network

Capturing Asynchrony

Conflict Detection

Commutativity
Which high-level events are conflicting?

Phrased as a graph connectivity query
Which high-level events are conflicting?

Phrased as a graph connectivity query
POX l2_multi controller with random network traffic

<table>
<thead>
<tr>
<th># Events</th>
<th># Races</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>564</td>
</tr>
<tr>
<td>1000</td>
<td>2822</td>
</tr>
<tr>
<td>4000</td>
<td>4026</td>
</tr>
</tbody>
</table>
Which high-level events commute?

\[
\begin{align*}
\& FLOW\_MOD(10, \text{H1} \rightarrow \text{H2}, \text{fwd}) \\
FLOW\_MOD(10, \text{H2} \rightarrow \text{H1}, \text{fwd})
\end{align*}
\]

Flow table:

- R0: 1, * → *, ctrl
- R1: 10, H1 → H2, fwd
- R2: 10, H2 → H1, fwd
SDNRacer: Detecting Concurrency Violations

Which high-level events commute?

\[\text{FLOW\_MOD}(10, \text{H}1 \to \text{H}2, \text{fwd})\]
\[\text{FLOW\_MOD}(10, \text{H}2 \to \text{H}1, \text{fwd})\]

- Commutativity
- SDN Network
- Capturing Asynchrony
- Conflict Detection

Flow table:

- R0: 1, * → *, ctrl
- R1: 10, H1 → H2, fwd
- R2: 10, H2 → H1, fwd

Flow table:

- R0: 1, * → *, ctrl
- R0: 1, * → *, ctrl
- R2: 10, H2 → H1, fwd

\[\text{RCV}(\text{H}2 \to \text{H}1)\]
\[\text{FLOW\_MOD}(10, \text{H}2 \to \text{H}1, \text{fwd})\]
# Evaluation

POX l2_multi controller with random network traffic

<table>
<thead>
<tr>
<th># Events</th>
<th># Races</th>
<th># Commutative Races</th>
<th># Analysis Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>564</td>
<td>321 (57%)</td>
<td>0.7s</td>
</tr>
<tr>
<td>1000</td>
<td>2822</td>
<td>1900 (67%)</td>
<td>3.5s</td>
</tr>
<tr>
<td>4000</td>
<td>4026</td>
<td>2702 (67%)</td>
<td>16.1s</td>
</tr>
</tbody>
</table>
SDNRacer: Detecting Concurrency Violations

**Flow table**
- R0: 1, * → *, ctrl
- R1: 10, H1 → H2, fwd
- R2: 10, H2 → H1, fwd

**Concurrency violations:**

```
if ip_src == H1:
    flow_mod(10, ip_src, ip_dst, fwd)
    flow_mod(10, ip_dst, ip_src, fwd)
    packet_out(pkt, sw)
else:
    flow_mod(15, ip_src, ip_dst, drop)
```

```
FLOW_MOD(10, H2 → H1, fwd) 4 10  RCV(H2 → H1)
```