Monitoring as a Design Target for Programmable Switches

NetPL

workshop(Networking & Programming Languages)
{ Brazil, 2016 }

Rodrigo Fonseca
Brown University
Joint work with Nicholas DeMarinis, Shriram Krishnamurthi, Tim Nelson
Scope

Dynamic monitoring correctness of stateful network behavior

*Satisfied by current programmable switch designs?*
A bit of history

• **Flowlog**\(^1\)
  – Stateful pro-active compilation + static verification
  – No stateful primitives on switches
  – No dynamic monitoring for correct behavior

• **Simon**\(^2\)
  – Dynamic monitoring,
  – Reactive language for scriptable debugging
  – Centralized, “see all packets”, works on Mininet

---


Dynamic Stateful Monitoring

• As much as possible, push monitors to switches

• Monitor as a state machine
  – For each property, find falsifying traces of network events

• Each sequence of events called an instance
  – Packets, timeouts, conf. changes, elements up/down
  – More general than a flow

“After seeing traffic from A (int) to B (ext), packets B to A are not dropped”
Dynamic Stateful Monitoring

- As much as possible, push monitors to switches
- Monitor as a state machine
  - For each property, find falsifying traces of network events
- Each sequence of events called an instance
  - Packets, timeouts, conf. changes, elements up/down
  - More general than a flow

“After seeing traffic from A (int) to B (ext), packets B to A are not dropped”
What features do we need?
Stateful Firewall

"After seeing traffic from A (int) to B (ext), packets B to A are not dropped”

- Access to relevant fields
- Access to persistent state
  - List of (A,B) pairs
- Detect dropped packets
**Stateful Firewall**

"After seeing traffic from A (int) to B (ext), packets B to A are not dropped for T seconds after seeing traffic from A to B”

- **Rule Timeouts**
  - Separate timers for each A,B pair
  - Reset when new A->B packets are seen
NAT

Return packets are translated according to their corresponding initial outgoing translation.

• Packet Identity
  – Violation requires 4 observations
  – (1) and (2); (3) and (4) must refer to the same packet
Negative match

- Step (4) detects departures with values not equal to defined values, should be possible to have negative matches
ARP Cache

“After a request for a known MAC address, send a reply within T seconds.”

• Timeout Actions
  – Here timeouts trigger an action (“flag violation”) rather than expire a rule
  – Refresh behavior is also different
Additional Features

• **Instance Identification**
  – Exact match: always same fields
  – Symmetric match: traditional notion of “flow”
  – Extended Flows
    • Match on prior values that may no longer be in the packet (e.g., NAT)
    • Multi-protocol instances, e.g., DHCP + ARP Proxy
Additional Features

• Multiple Match
  – Some events might advance several instances
  – E.g., “After topology reset, properly clear MAC cache”
  – Violation requires seeing A→B, topology reset, and then seeing X→A be unicast.
  – Problem: effect of topology reset event is dynamic and unbounded
Additional Features

• **Provenance**
  – Sequence of events that lead to violation

• **Side-effect Control**
  – Tradeoff between completeness and non-blocking state changes
How are we doing?

- OpenState (Bianchi et al., CCR’14)
- FAST (Moshref et al., HotSDN’14)
- P4 (Bosshard et al., CCR’14)
- POF (Song, HotSDN’13)
- SNAP (Arashloo et al., SIGCOMM’16)
- Varanus (dynamic and static) (ongoing)
<table>
<thead>
<tr>
<th>Feature</th>
<th>OpenFlow 1.3</th>
<th>OpenState</th>
<th>FAST</th>
<th>P4 and POF</th>
<th>SNAP</th>
<th>Varanus</th>
<th>Static Varanus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access to fields</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Persistent State</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rule Timeouts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timeout actions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Packet Identity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Negative match</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symmetric match</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“Extended” flows</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiple match</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provenance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Processing mode</td>
<td>Inline</td>
<td>Inline</td>
<td>Inline</td>
<td>Inline</td>
<td>Asyn</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td>OpenFlow 1.3</td>
<td>OpenState</td>
<td>FAST</td>
<td>P4 and POF</td>
<td>SNAP</td>
<td>Varanus</td>
<td>Static Varanus</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>--------------</td>
<td>-----------</td>
<td>------</td>
<td>------------</td>
<td>------</td>
<td>---------</td>
<td>---------------</td>
</tr>
<tr>
<td>Access to fields</td>
<td>Fixed</td>
<td>Fixed</td>
<td>Fixed</td>
<td>Dynamic</td>
<td>Dynamic</td>
<td>Dynamic</td>
<td>Fixed</td>
</tr>
<tr>
<td>Persistent State</td>
<td>Ctrl</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rule Timeouts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timeout actions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Packet Identity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Negative match</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symmetric match</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“Extended” flows</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiple match</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provenance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Processing mode</td>
<td>Inline</td>
<td>Inline</td>
<td>Inline</td>
<td>Inline</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Asyn</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Asyn</td>
</tr>
</tbody>
</table>
Varanus

• Targeted at Stateful Monitoring on switches
• Metric Temporal First-Order Logic queries -> DFA
• Extended OVS
  – recursive LEARN rules
  – “action on timeout”

Match: ARP request arrival
Action: learn into Table 2:
  {Match: ARP reply departure, dst=current(src), tpa=current(tpa),
   Action: delete self,
   Timeout: 5 seconds,
   Timeout action: notify monitor}
Varanus

• **Two versions**
  – Dynamic: one new table per instance
    • Allows multiple matches, but terrible scalability
  – Static: one table per query per state
  – See [1] for details

• *(Seems)* Fundamental limitation: updating match tables slow, seeking alternative approaches

Stateful Switch Primitives

- Previous work focused on stateful forwarding
- Monitoring presents different requirements
- Still good for sequences of
  - Positive observations
  - Symmetric or exact instance identification
Stateful Switch Primitives

- **Timeout actions**
  - Enables powerful negative observations (*within*)
- **“Extended” flow matching**
  - Shifting protocols, state not in packet
- **State updates**
  - Inline vs asynchronous
- **Recording Provenance**
  - Open problem: may be too costly
- **Multiple match**
  - Useful for external events, may be too costly
Stateful monitoring similar, but different, from stateful forwarding, worth considering for upcoming switch designs

Thank you!