Monitoring as a Design Target for Programmable Switches

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?

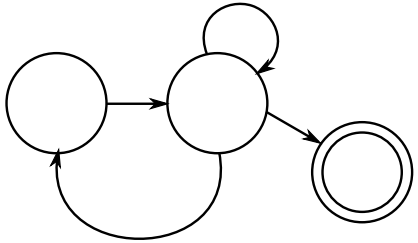
A bit of history

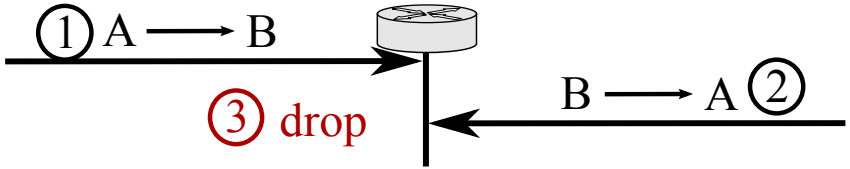
- **Flowlog[1]**
  - Stateful pro-active compilation + static verification
  - No stateful primitives on switches
  - No dynamic monitoring for correct behavior
- **Simon[2]**
  - Dynamic monitoring,
  - Reactive language for scriptable debugging
  - Centralized, “see all packets”, works on Mininet

1. T. Nelson, M. Scheer, A. Ferguson, and S. Krishnamurthi. “Tierless programming and reasoning for Software-Defined Networks”. NSDI 2014.

2. T. Nelson, D. Yu, Y. Li, R. Fonseca, and Shriram Krishnamurthi. “Simon: Scriptable Interactive Monitoring for SDNs. SOSR 2015

Dynamic Stateful Monitoring

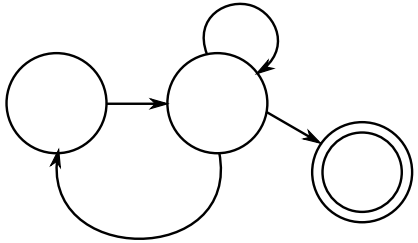


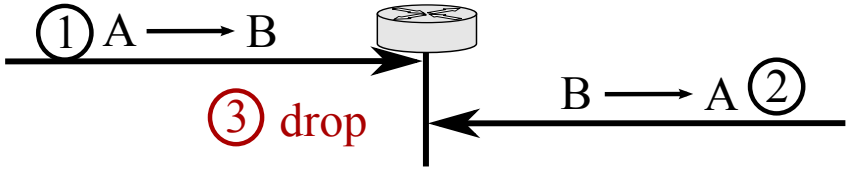


"After seeing traffic from A (int) to B (ext), packets B to A are not dropped"

- **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

Dynamic Stateful Monitoring





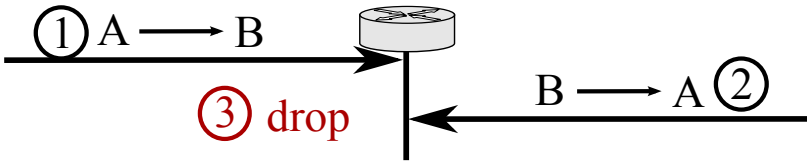
"After seeing traffic from A (int) to B (ext), packets B to A are not dropped"

- **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

Require more than just state!

What features do we need?

Stateful Firewall

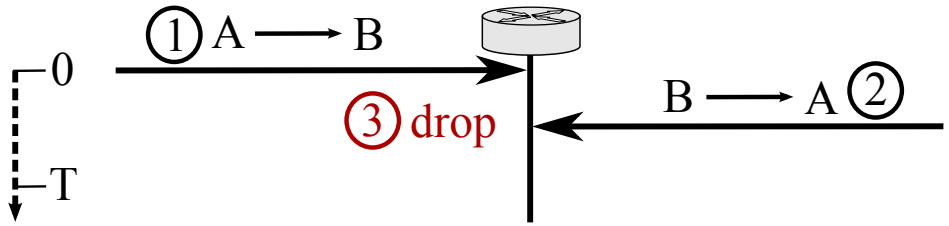


"After seeing traffic from A (int) to B (ext), packets B to A are not dropped"

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

Access to fields
Persistent State
Rule Timeouts
Timeout actions
Packet Identity
Negative match
Symmetric match
Extended flows
Multiple match
Provenance
Side-effect Control

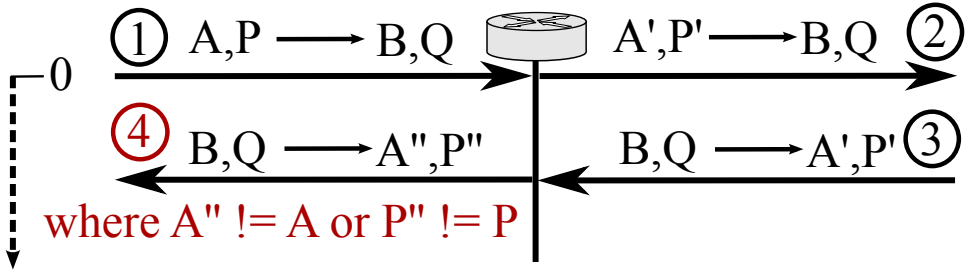
Stateful Firewall



"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

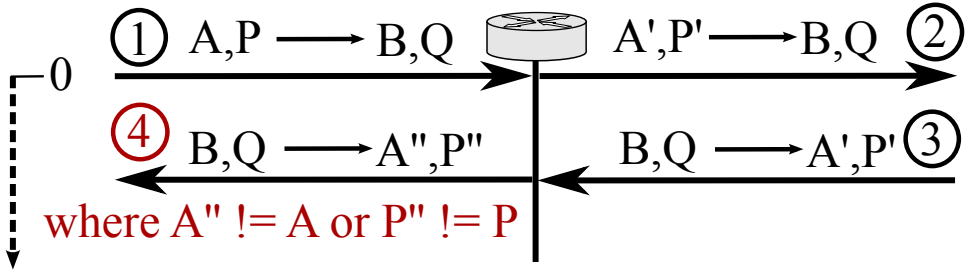


"Return packets are translated according to their corresponding initial outgoing translation."

- **Packet Identity**
  - Violation requires 4 observations
  - (1) and (2); (3) and (4) must refer to the same packet

Access to fields
Persistent State
Rule Timeouts
Timeout actions
Packet Identity
Negative match
Symmetric match
Extended flows
Multiple match
Provenance
Side-effect Control

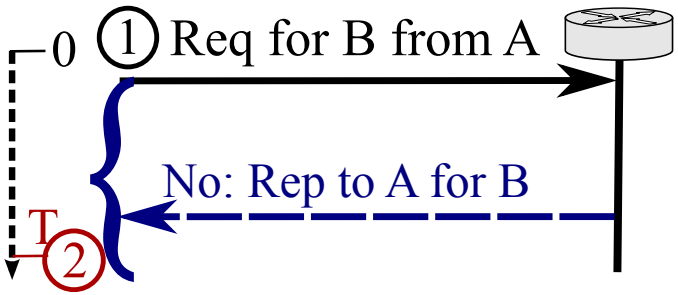
NAT



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

- **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

Access to fields
Persistent State
Rule Timeouts
Timeout actions
Packet Identity
Negative match
Symmetric match
Extended flows
Multiple match
Provenance
Side-effect Control

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

Access to fields
Persistent State
Rule Timeouts
Timeout actions
Packet Identity
Negative match
Symmetric match
Extended flows
Multiple match
Provenance
Side-effect Control

Additional Features

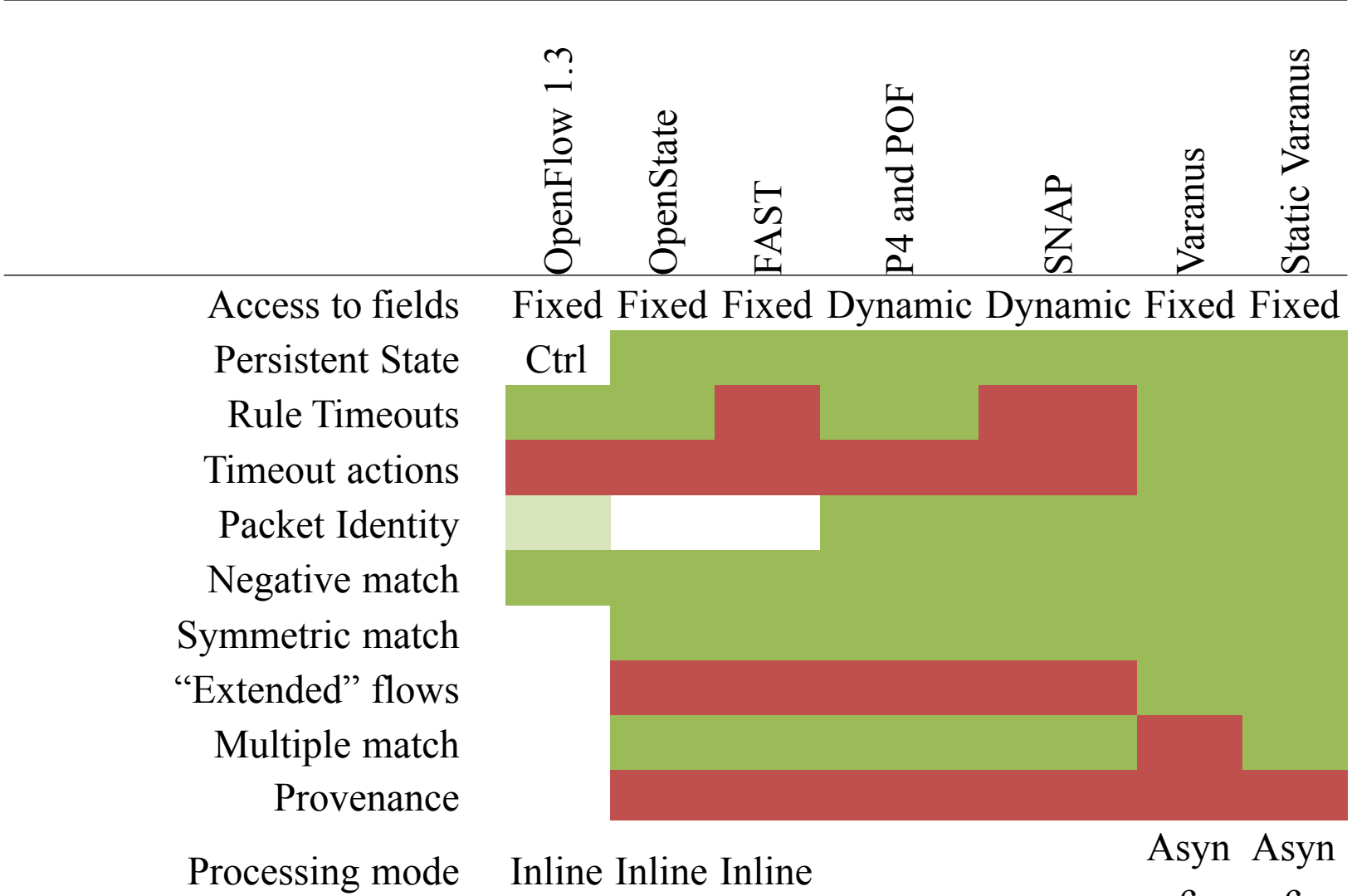
- **Provenance**
  - Sequence of events that lead to violation
- **Side-effect Control**
  - Tradeoff between completeness and non-blocking state changes

Access to fields
Persistent State
Rule Timeouts
Timeout actions
Packet Identity
Negative match
Symmetric match
Extended flows
Multiple match
Provenance
Side-effect Control

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)

	OpenFlow 1.3	OpenState	FAST	P4 and POF	SNAP	Varanus	Static Varanus
Access to fields							
Persistent State							
Rule Timeouts							
Timeout actions							
Packet Identity							
Negative match							
Symmetric match							
“Extended” flows							
Multiple match							
Provenance							
Processing mode	[illegible]	[illegible]	[illegible]				


OpenFlow 1.3
OpenState
FAST
P4 and POF
SNAP
Varanus
Static Varanus
Access to fields
Fixed Fixed Fixed Dynamic Dynamic Fixed Fixed
Persistent State
Ctrl
Rule Timeouts
Timeout actions
Packet Identity
Negative match
Symmetric match
“Extended” flows
Multiple match
Provenance
Processing mode
Inline Inline Inline
Asyn Asyn

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}

~

(lambda (src, tpa) …)

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**

[1] T. Nelson, N. DeMarinis, T. Hoff, R. Fonseca, S. Krishnamurthi. “Compiling Stateful Network Properties for Runtime Verification”, arXiv:1607.03385, July 2016

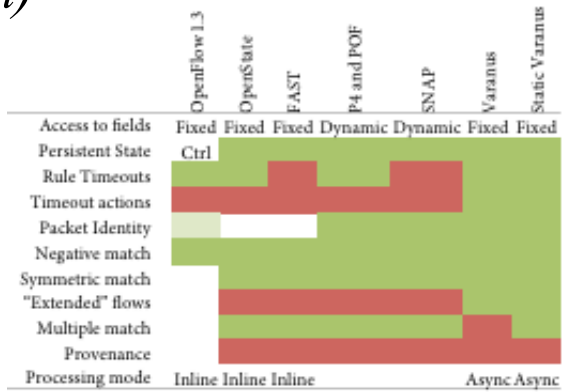
Stateful Switch Primitives

- **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

For monitoring

- **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!