



Troubleshooting SDN Control Software with Minimal Causal Sequences

Colin Scott, Andreas Wundsam, Barath Raghavan, Andrew Or,
Jefferson Lai, Eugene Huang, Zhi Liu, Ahmed El-Hassany, Sam
Whitlock, Hrishikesh B. Acharya, Kyriakos Zarifis, Scott Shenker



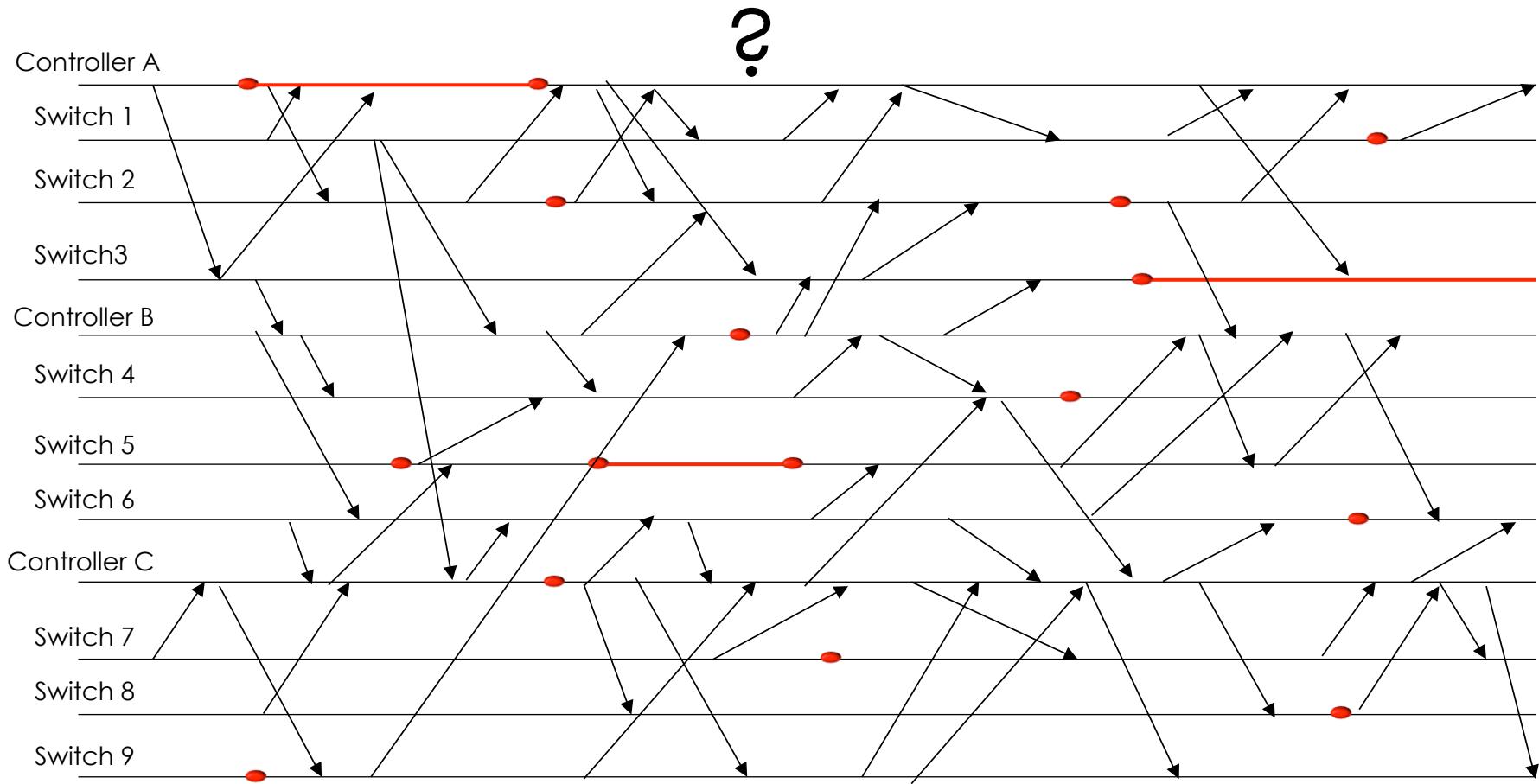
Something goes wrong!

- Loops, Blackholes, etc. found in:
 - Unit tests
 - Integration tests
 - Software QA testbeds
 - Hardware QA testbeds
 - Production

Best practice: Logs

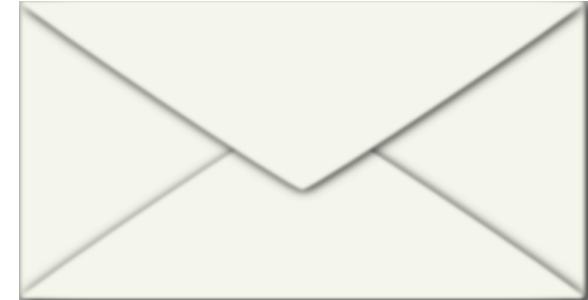
Human
analysis of
log files

Best practice: Logs



How many events?

- $20,000 \text{ servers} * 4 \text{ VMs / server} = 80,000 \text{ VMs}$
- $(6 \text{ migrations / day / VM}) + (2 \text{ power up | down / day / VM}) * 80,000 \text{ VMs} = 640,000 \text{ VM events / day}$
 $\sim= 450 \text{ VM changes / minute}$ [1]
- **$8.5 \text{ network error events / minute}$ [2]**



How many events?

= ~500 inputs /
minute

Our Goal

Identify a minimal sequence of inputs that triggers the bug

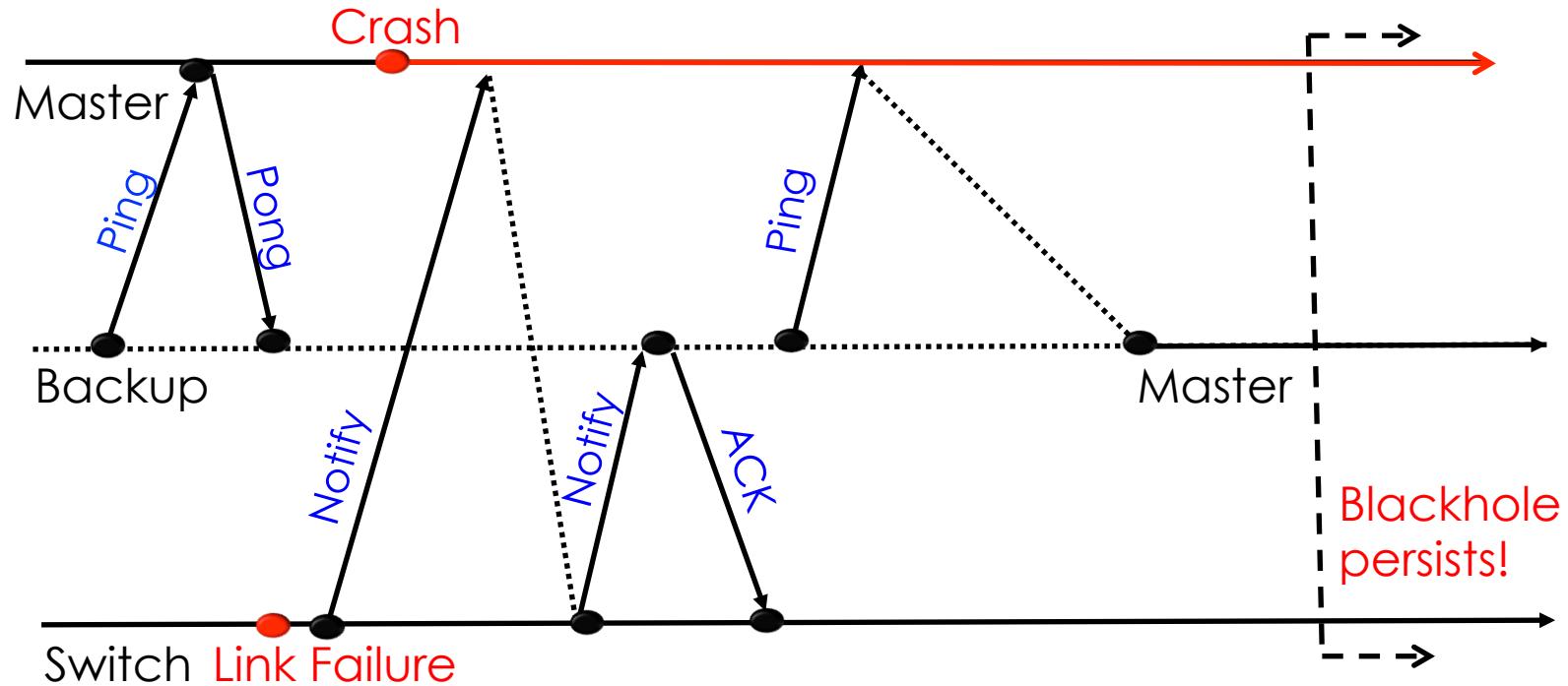
Minimal Causal Sequence

$MCS \subset Trace$ s.t.

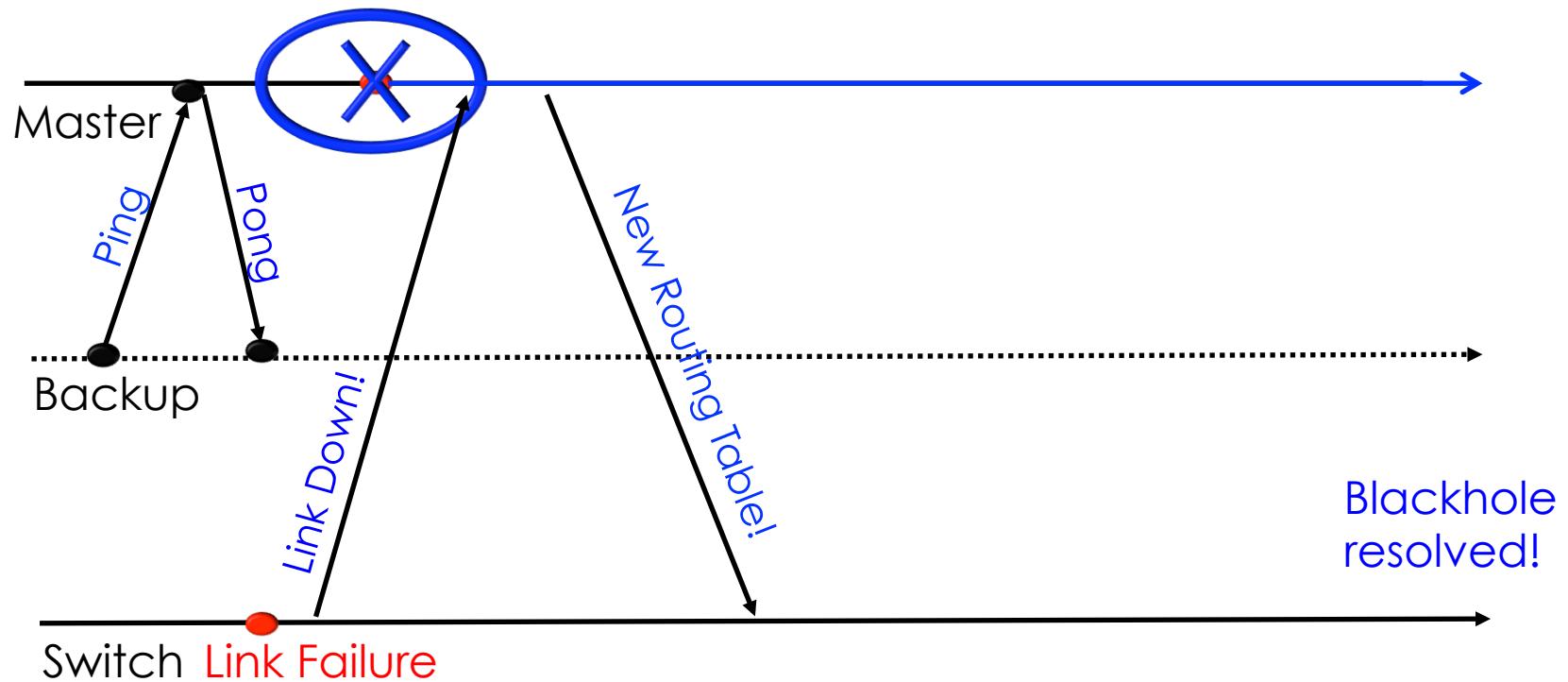
i. $replay(MCS) = \chi$

ii. $\forall_{e \in MCS} replay(MCS - \{e\}) \neq \chi$

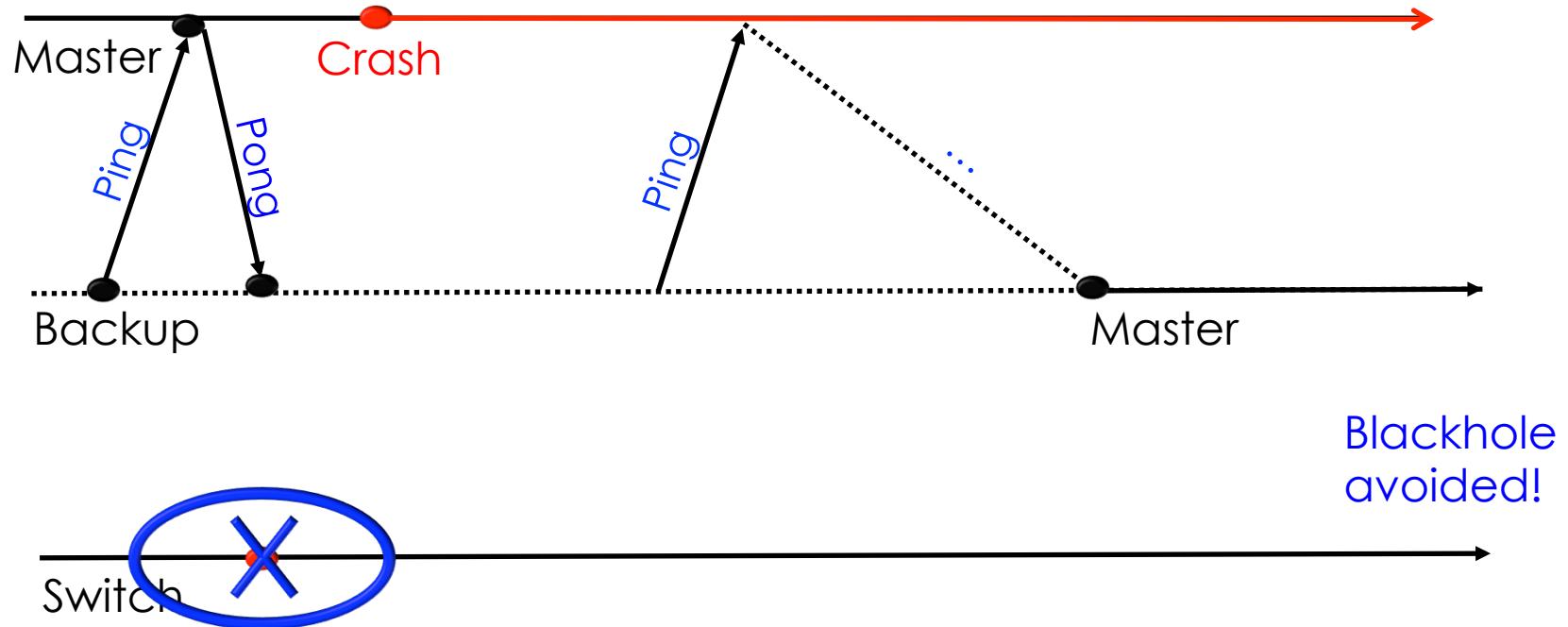
Minimal Causal Sequence



Minimal Causal Sequence

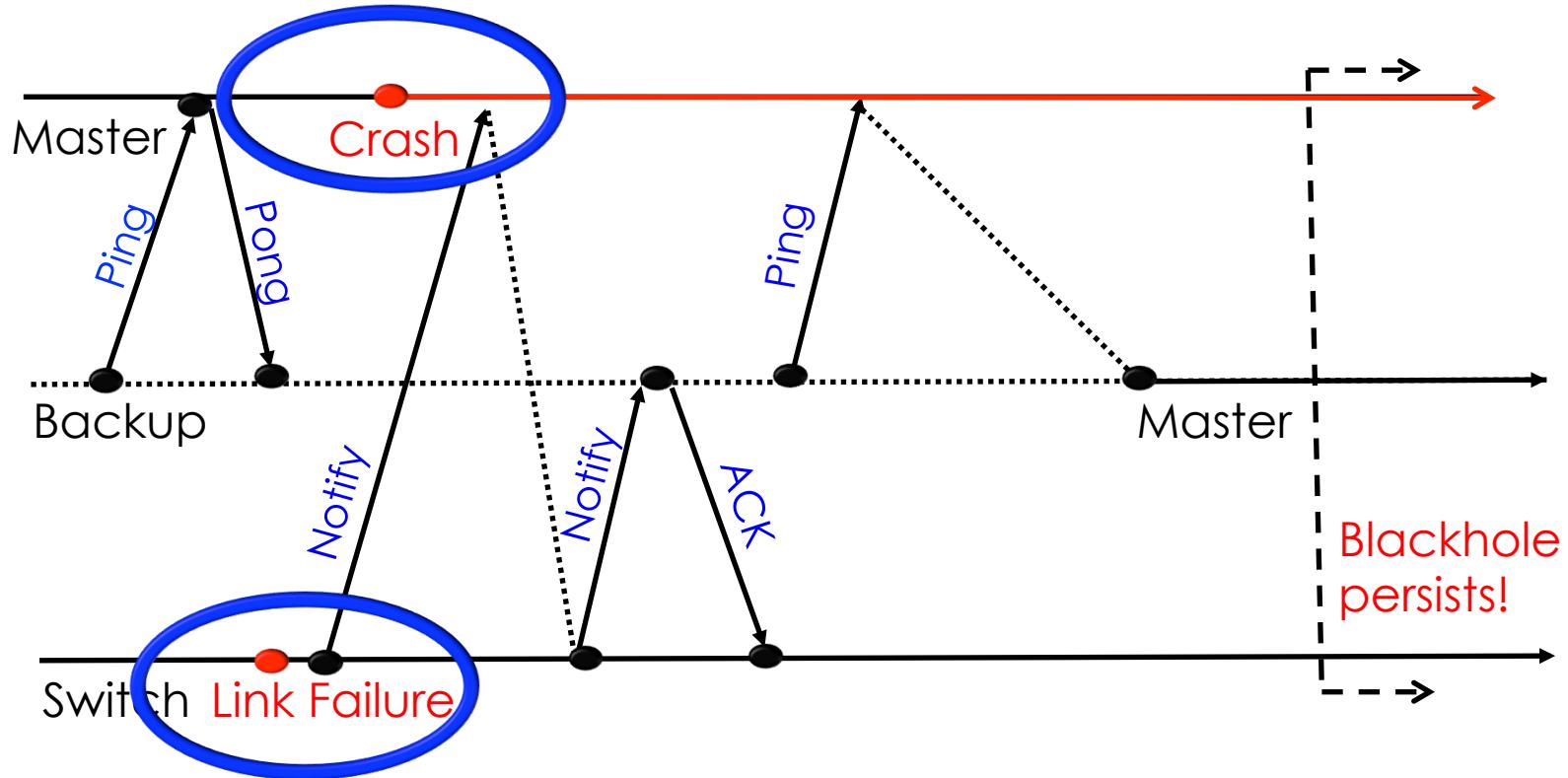


Minimal Causal Sequence

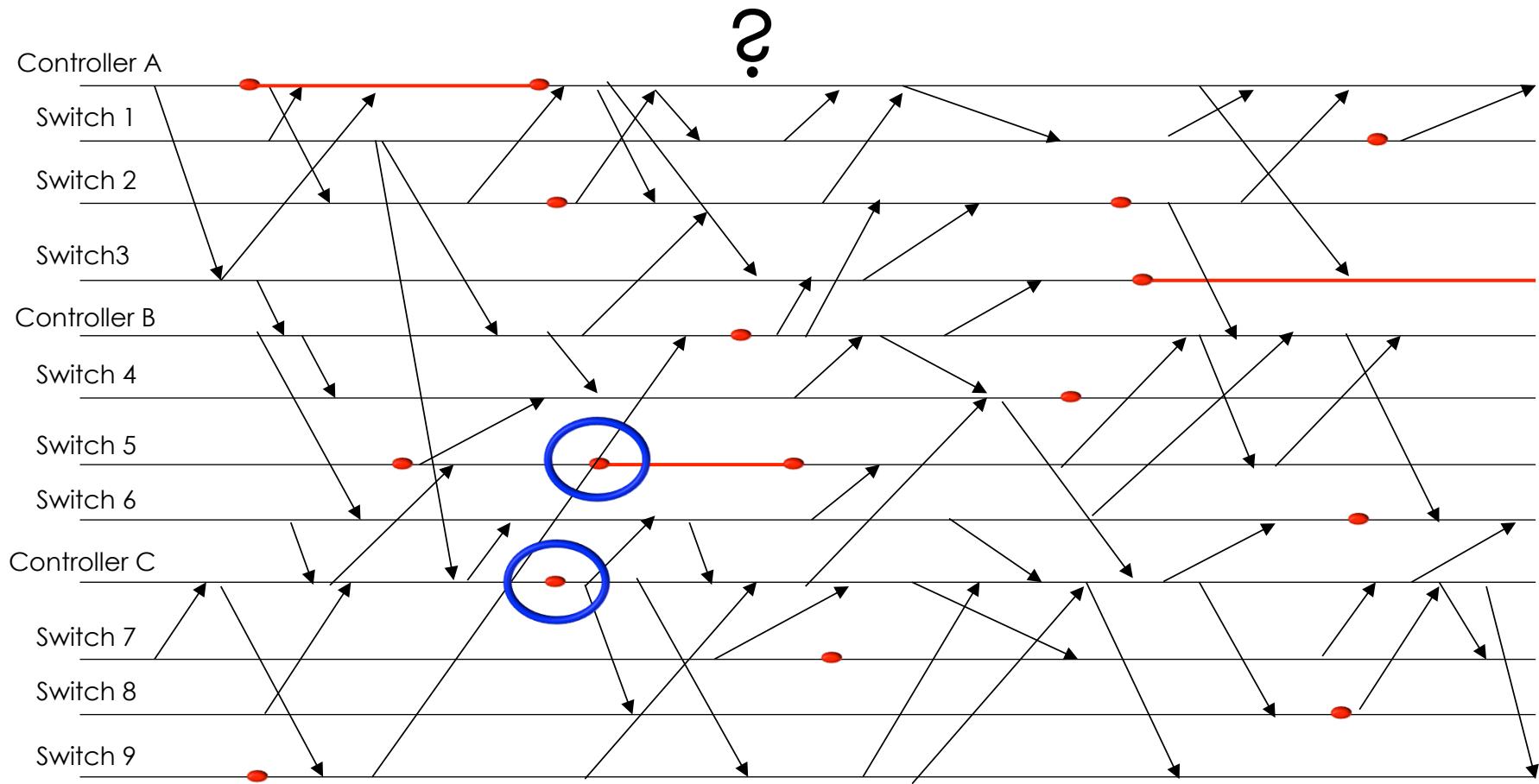


Minimal Causal Sequence

Both are necessary conditions!



Minimal Causal Sequence



Approach: Delta Debugging

Modify history!



Possible failure causes



Set up first hypothesis



Test first hypothesis



Second hypothesis



Third hypothesis

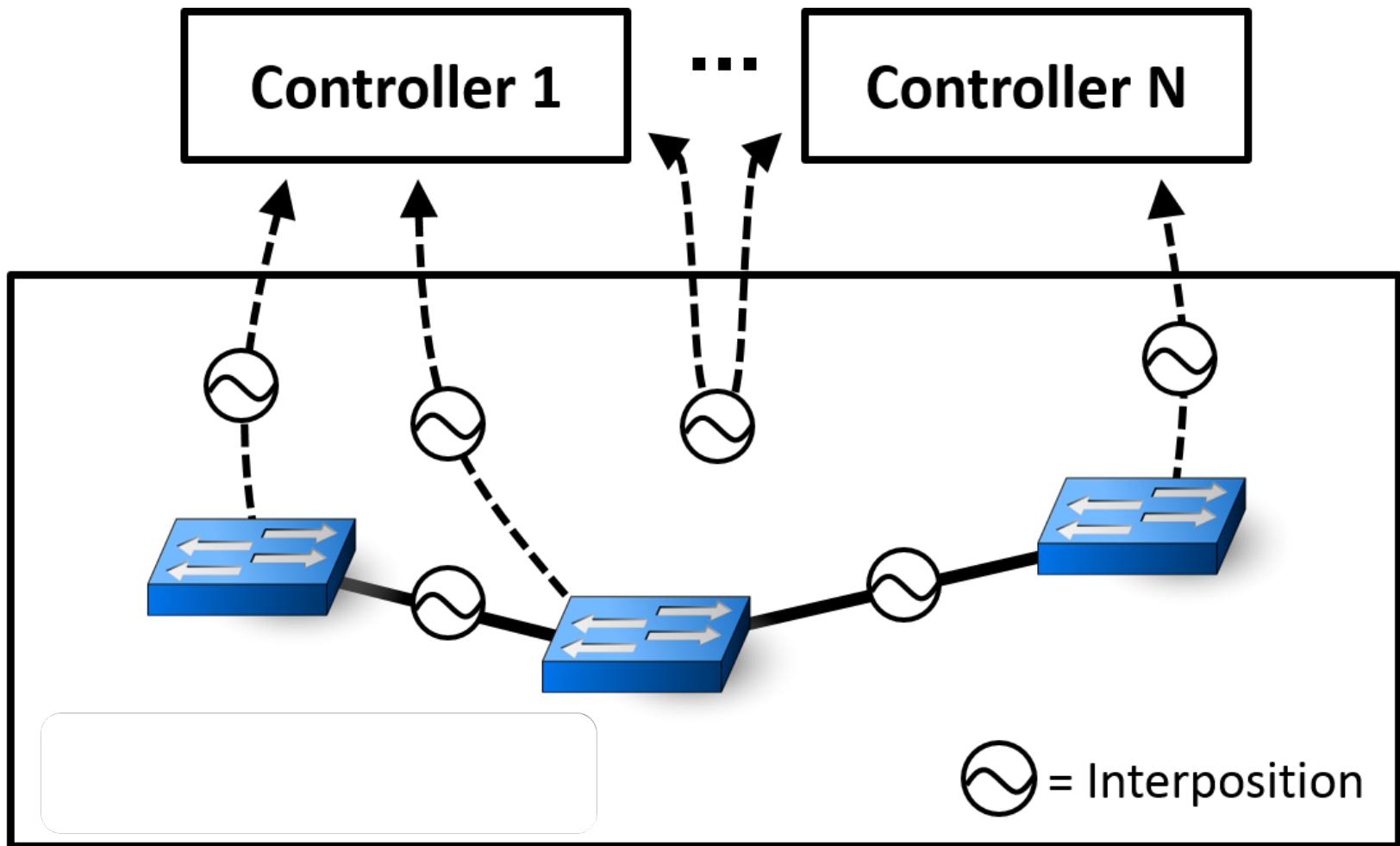


Fourth hypothesis...

Challenges

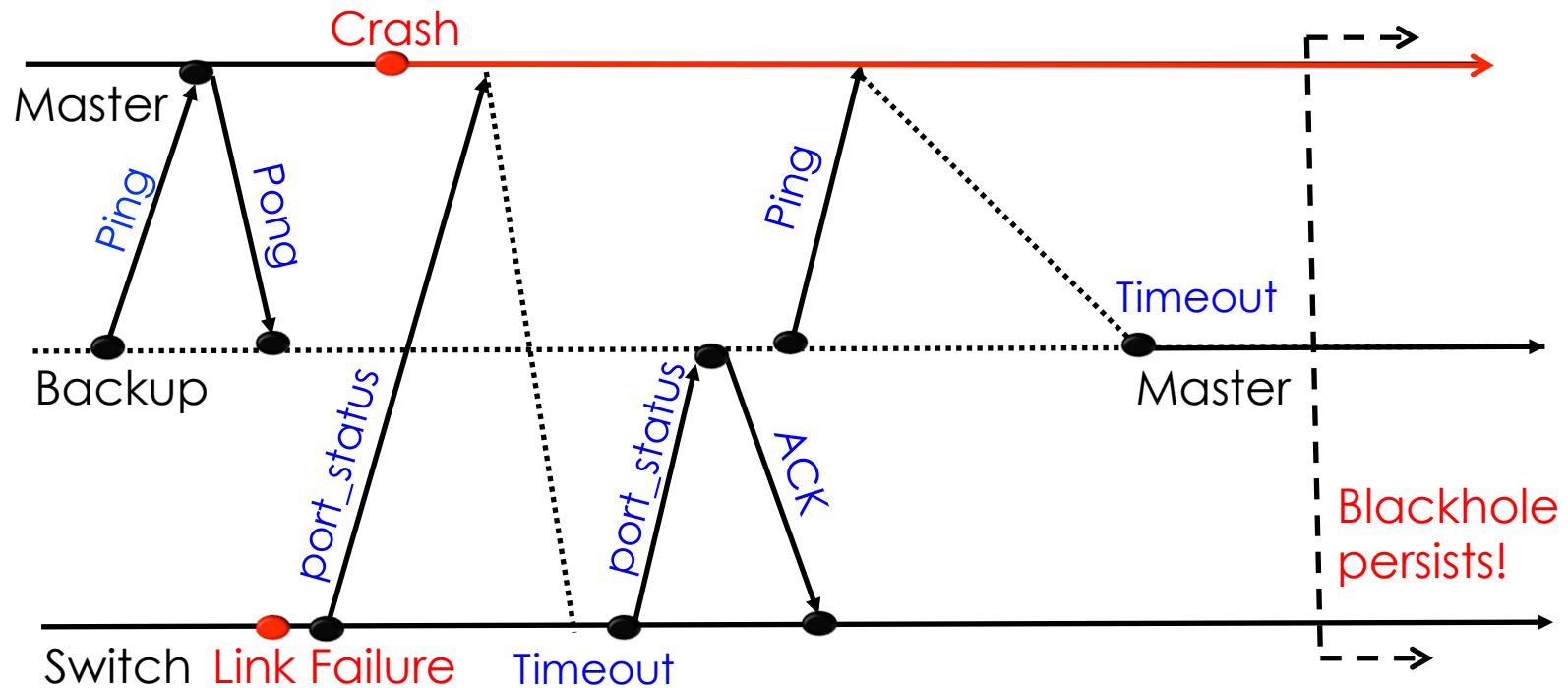
- Must carefully control timing of input events, despite
 - Modified history
 - Asynchrony
 - Non-determinism within controllers
- Without assuming language or instrumentation of control software

Replay Infrastructure



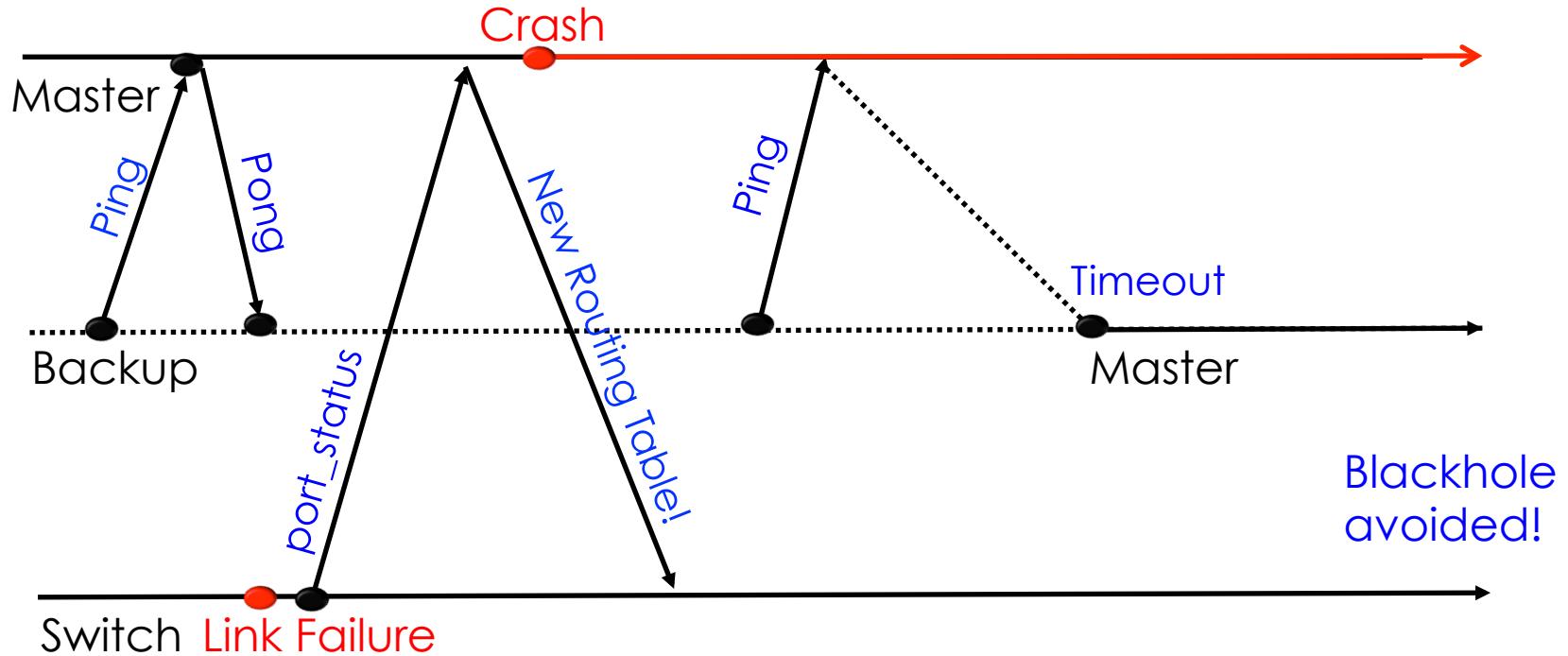
Replaying Pruned Inputs is Hard

Must consider **internal** events



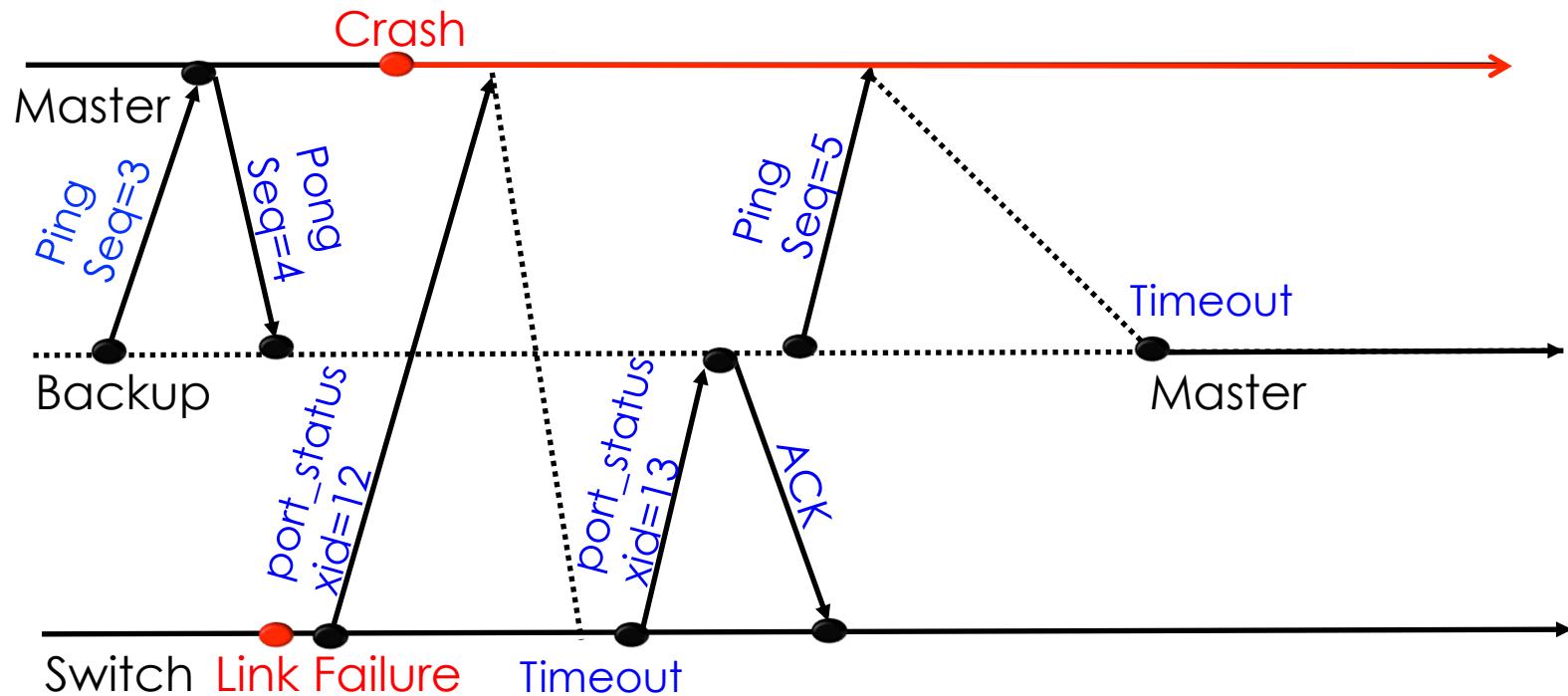
Replaying Pruned Inputs is Hard

Must consider **internal** events



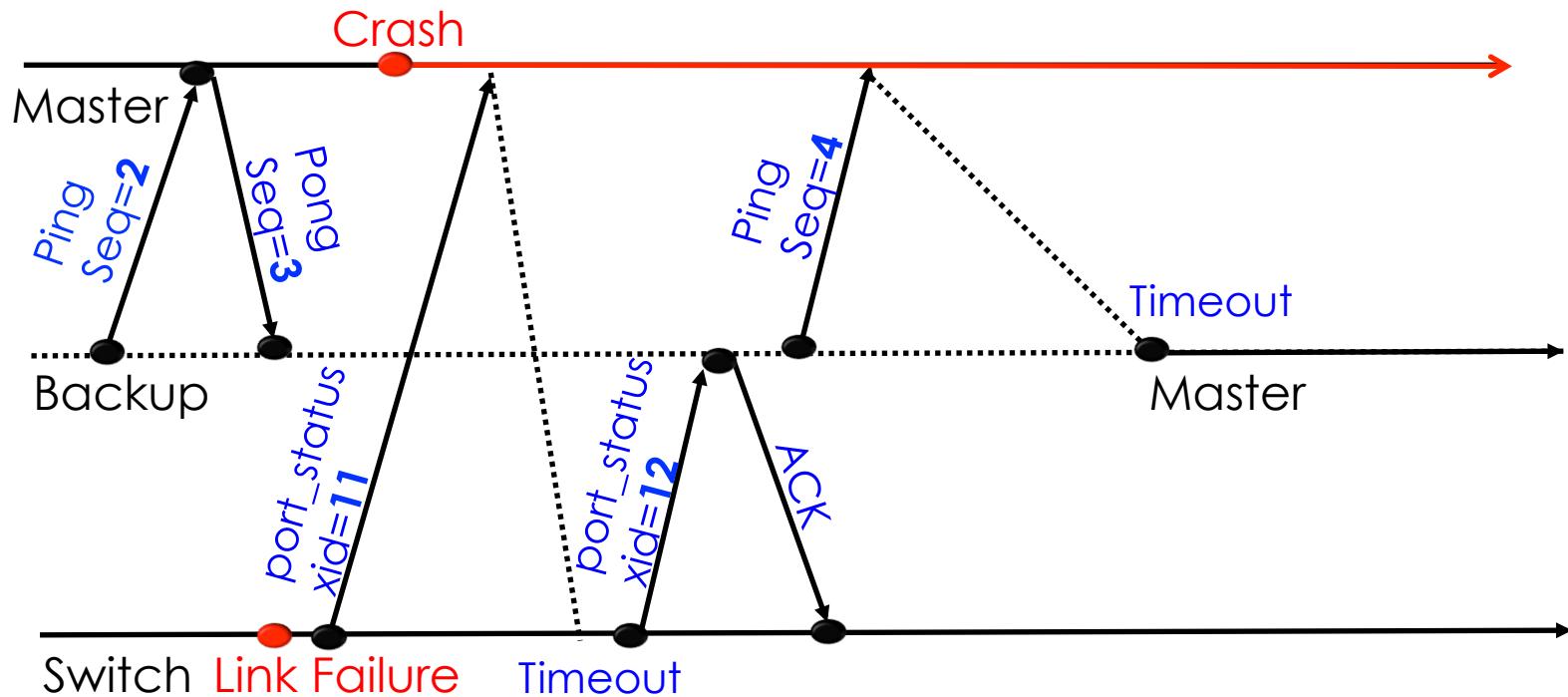
Pruning Alters Internal Events!

Prune Earlier Input..



Pruning Alters Internal Events!

Sequence Numbers Differ!



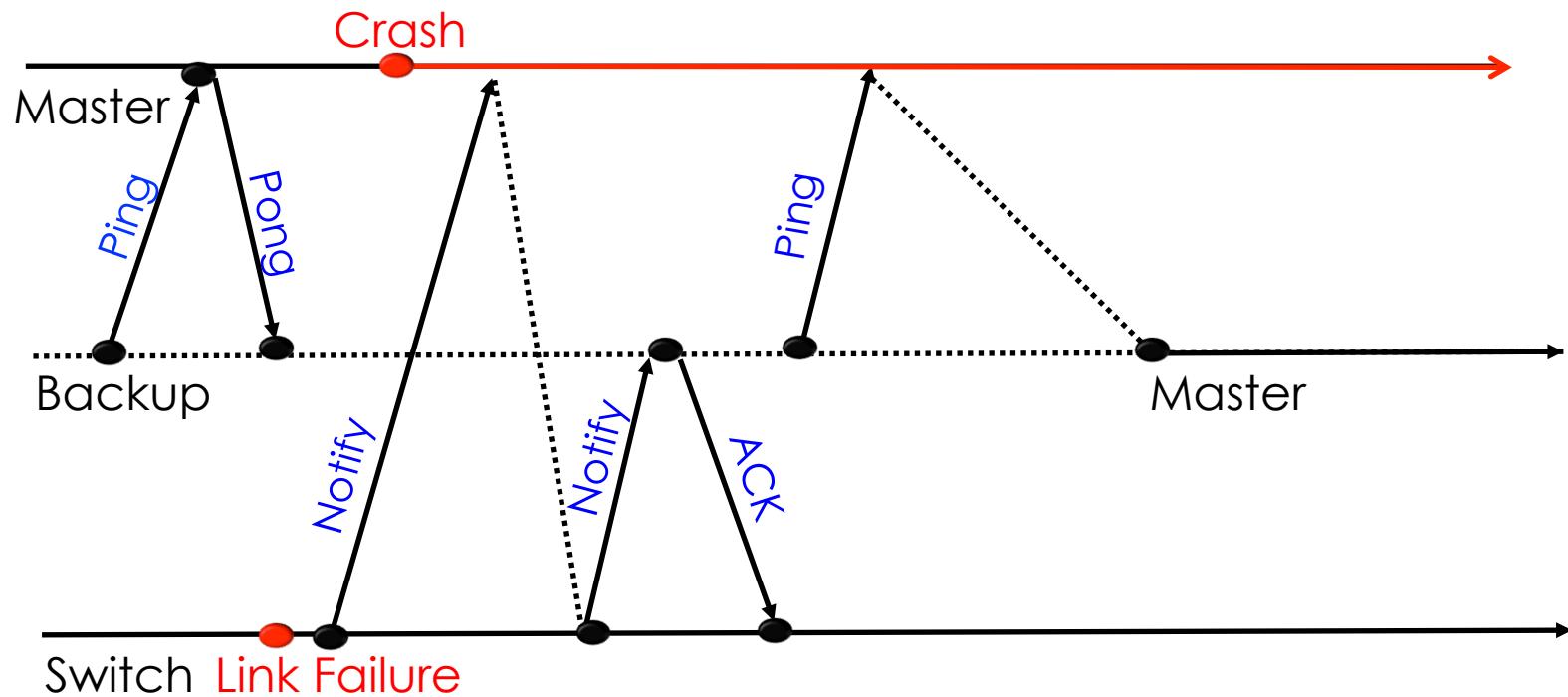
Solution: Equivalence Classes

Mask Over Extraneous Fields

Internal message	Masked values
OpenFlow messages	xac id, cookie, buffer id, stats
packet_out/in payload	all values except src, dst, data
Log statements	varargs parameters to printf

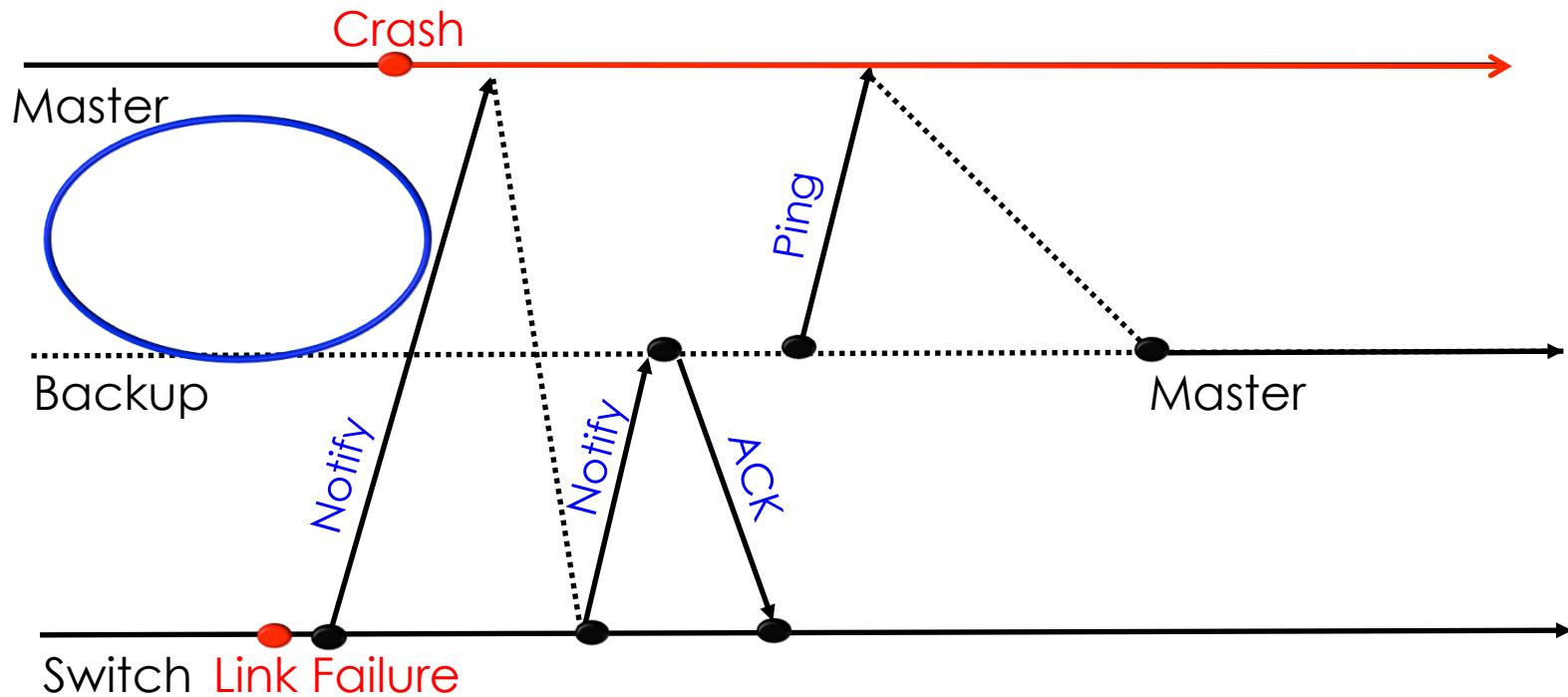
Pruning Alters Internal Events!

Prune Earlier Input..



Pruning Alters Internal Events!

Some Events No Longer Appear

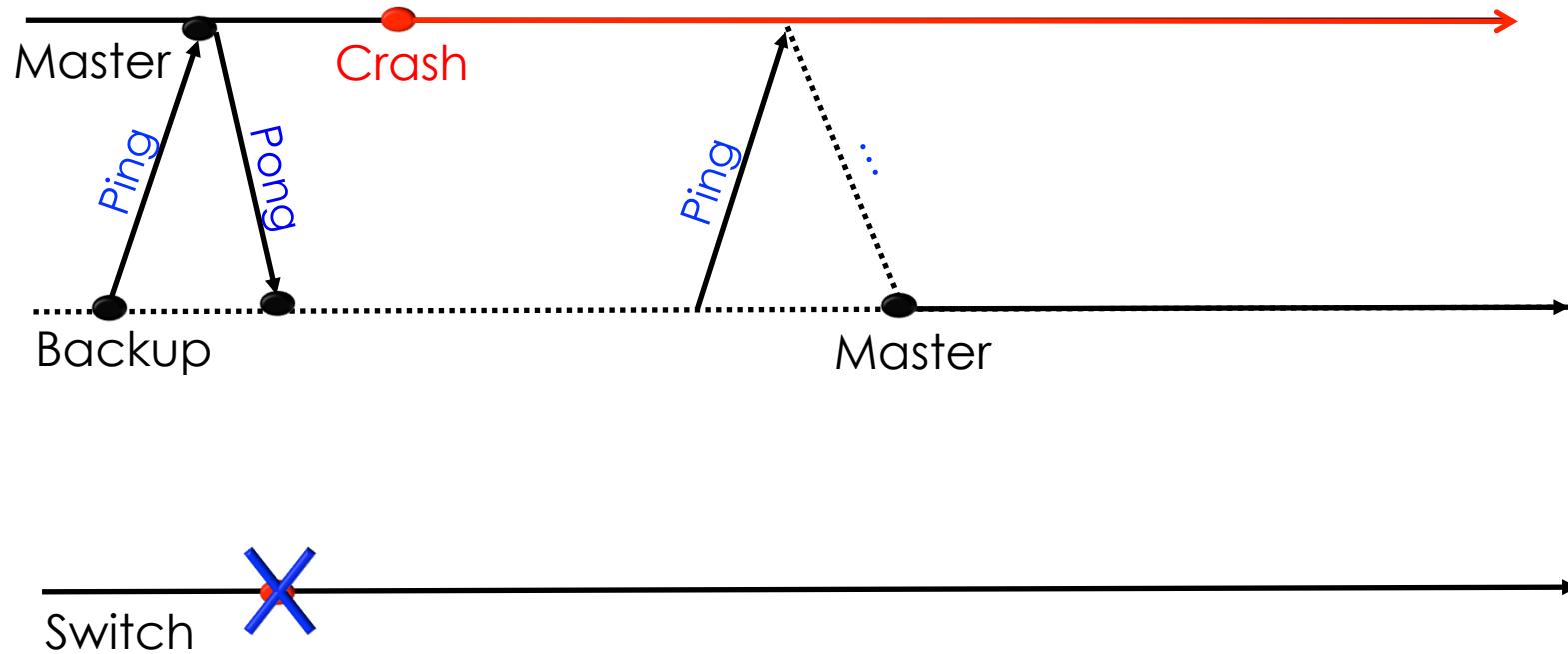


Solution: Peek ahead

```
procedure PEEK(input subsequence)
    inferred ← []
    for  $e_i$  in subsequence
        { checkpoint system
            inject  $e_i$ 
             $\Delta \leftarrow |e_{i+1}.time - e_i.time| + \epsilon$ 
            record events for  $\Delta$  seconds
            matched ← original events & recorded events
            inferred ← inferred + [ $e_i$ ] + matched
            restore checkpoint
        }
    return inferred
```

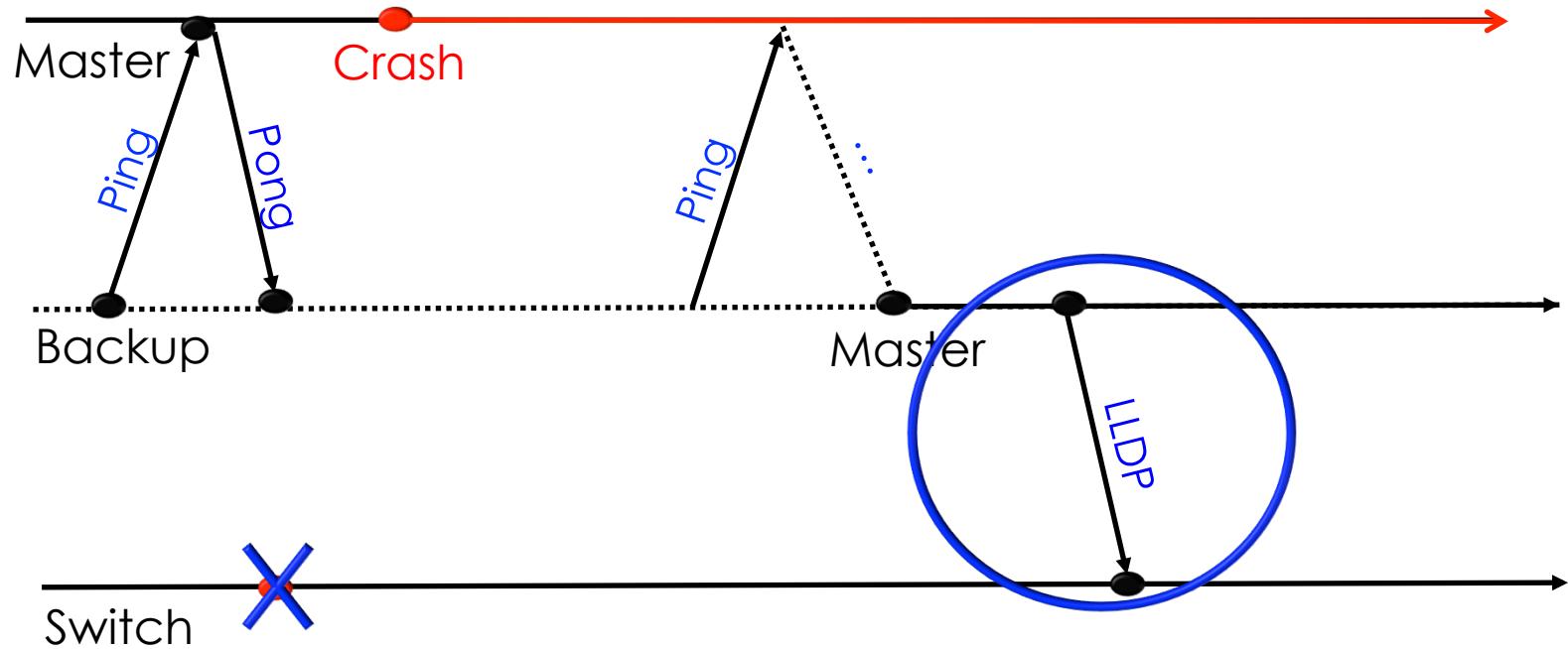
Pruning Alters Internal Events!

Prune Input..



Pruning Alters Internal Events!

Unexpected Events Appear



Dealing with unexpected events

Heuristic:

Allow unexpected
events through

Coping With Non-Determinism

- Replay multiple times per subsequence
- Probability of not finding bug modeled by:

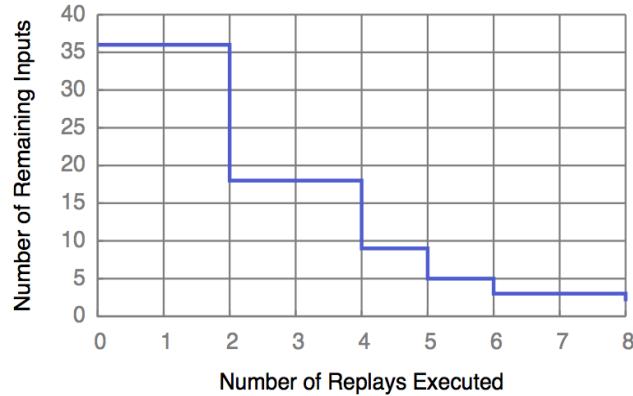
$$f(p, n) = (1 - p)^n$$

- Optionally override `gettimeofday()`,
multiplex sockets, interpose on logging
statements

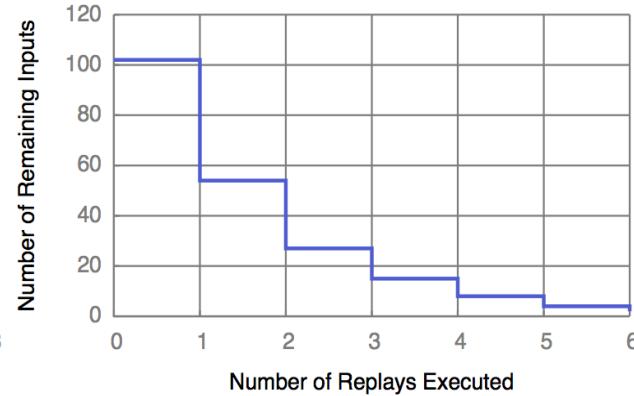
It Works!

	Bug Name	Topology	Runtime (s)	Input Size	MCS Size	MCS helpful?
Newly Found	Pyretic loop POX premature PacketIn POX in-flight blackhole POX migration blackhole NOX discovery loop Floodlight loop ONOS distributed database locking	3 switch mesh 4 switch mesh 2 switch mesh 4 switch mesh 4 switch mesh 3 switch mesh 2 switch mesh	266.2 249.1 641.1 1796.0 4990.9 27930.6 N/A	36 102 46 29 150 117 1	2 2 7 3 18 13 1	Yes Yes Yes Yes Indirectly Yes N/A
Known	Floodlight failover bug ONOS master election POX load balancer error checking	2 switch mesh 2 switch mesh 3 switch mesh	- 6325.2 2396.7	202 30 106	2 3 24 (N+1)	Yes Yes Yes
Synthetic	Null pointer on rarely used codepath Overlapping flow entries Delicate timer interleaving Algorithm misimplementation Multithreaded race condition Memory leak Memory corruption	20 switch FatTree 2 switch mesh 3 switch mesh 3 switch mesh 10 switch mesh 2 switch mesh 4 switch mesh	2396.7 115.4 N/A 525.2 36967.5 15022.6 145.7	365 27 39 40 1596 719 341	2 2 39 7 2 30 (M) 2	Yes Yes No Indirectly Indirectly Indirectly Yes

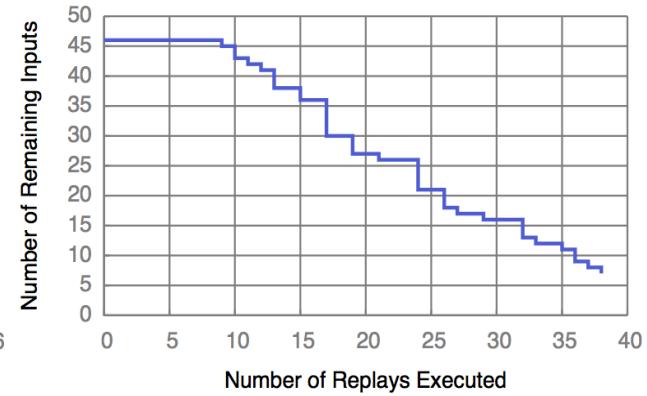
It Works!



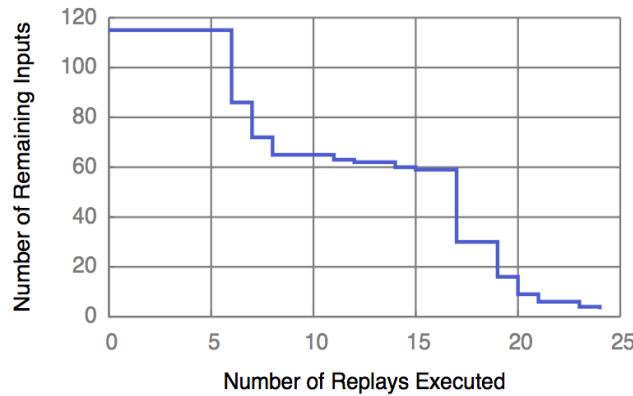
(a) Pyretic loop.



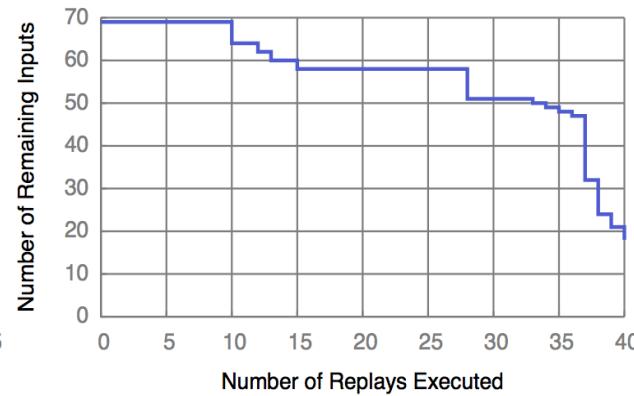
(b) POX Premature Packet-In Failure.



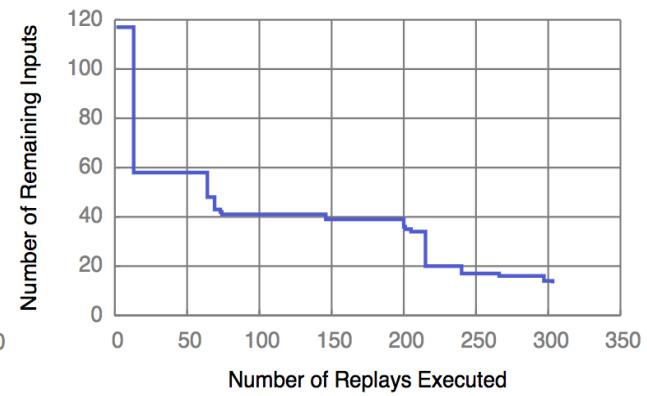
(c) POX in-flight blackhole.



(d) POX migration blackhole.



(e) NOX discovery loop.



(f) Floodlight loop.

Summary

- Goal: isolate minimal causal sequences
- We built a real system (21K+ lines of Python) that works on real controllers (Floodlight, NOX, POX, Frenetic, ONOS) including one proprietary controller
- Check us out:

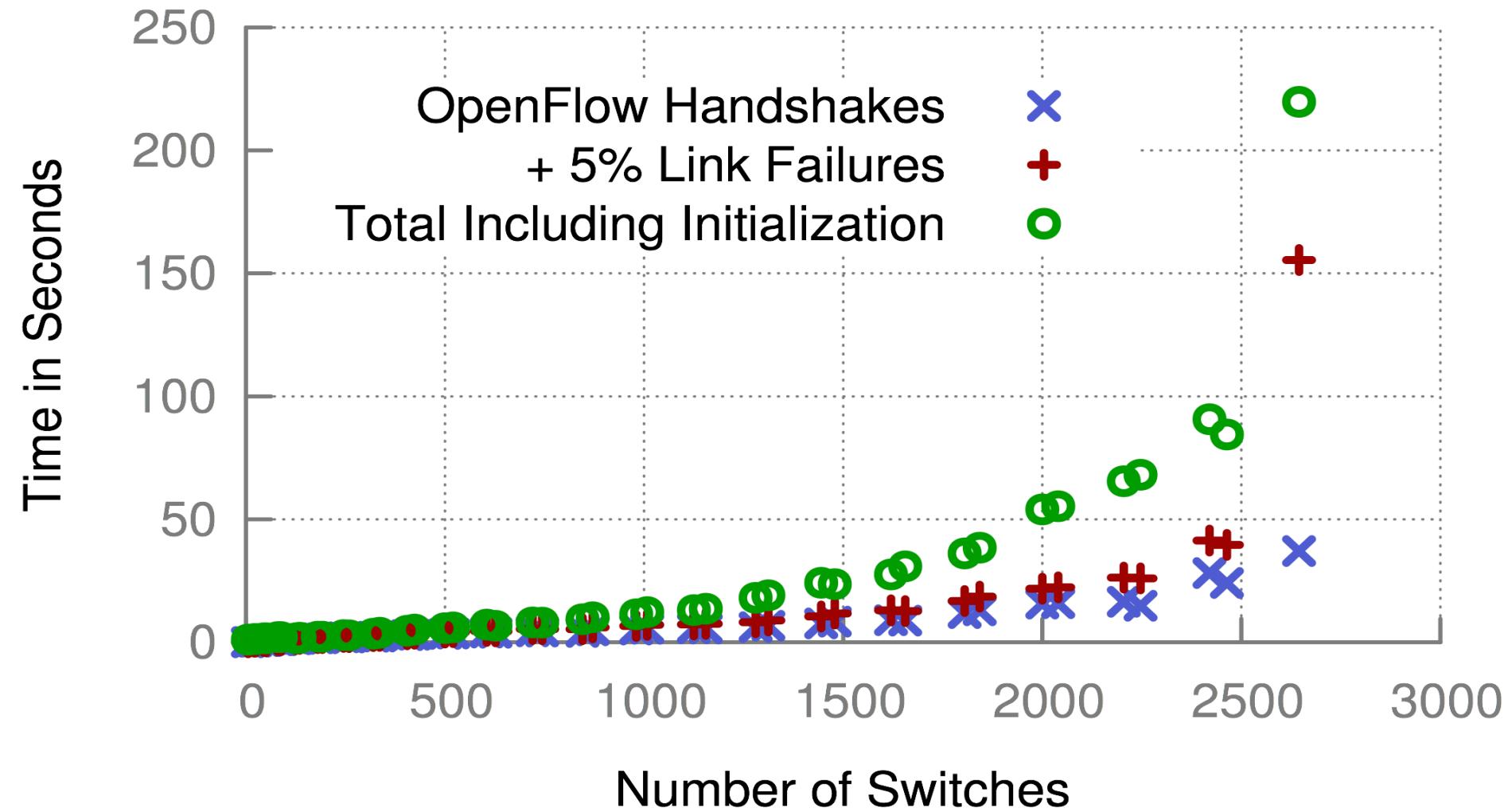
ucb-sts.github.com/sts/

References

- ❑ [1] V. Soundararajan and K. Govil. Challenges in building scalable virtualized datacenter management. SIGOPS Operating Systems Review '10.
- ❑ [2] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: a scalable and flexible data center network, Sec. 3.4. SIGCOMM '09.

Backup

Scalability



Complexity

Best Case	Worst Case
<ul style="list-style-type: none">- Delta Debugging: $\Omega(\log n)$ replays- Each replay: $O(n)$ events- Total: $\Omega(n \log n)$	<ul style="list-style-type: none">- Delta Debugging: $O(n)$ replays- Each replay: $O(n)$ events- Total: $O(n^2)$

Assumptions of Delta Debugging

- *Monotonic:*

$$P \oplus C = \chi \Rightarrow P \oplus (C \cup C') \neq \checkmark$$

- *Unambiguous:*

$$P \oplus C = \chi \wedge P \oplus C' = \chi \Rightarrow P \oplus (C \cap C') \neq \checkmark$$

- *Consistent*

$$P \oplus C \neq ?$$

Coping With Non-determinism

Results for synthetic multithreading bug in Floodlight

Max replays per subsequence	Size of final MCS	Total hours
1	65	6.10
2	20	6.37
3	15	7.78
4	12	9.59
5	9	6.38
6	9	11.20
7	9	11.83
8	6	12.35
9	6	11.13
10	6	12.86

Local vs. Global Minimality

Definition 8 (Global minimum). A set $c \subseteq c_\chi$ is called the global minimum of c_χ if: $\forall c' \subseteq c_\chi \cdot (|c'| < |c| \Rightarrow \text{test}(c') \neq \chi)$ holds.

Definition 10 (n -minimal test case). A test case $c \subseteq c_\chi$ is n -minimal if: $\forall c' \subset c \cdot |c| - |c'| \leq n \Rightarrow (\text{test}(c') \neq \chi)$ holds. Consequently, c is 1-minimal if $\forall \delta_i \in c \cdot \text{test}(c - \{\delta_i\}) \neq \chi$ holds.

Forensic Analysis of Production Logs

- ❑ Logs need to capture causality: Lamport Clocks or accurate NTP
- ❑ Need clear mapping between input/internal events and simulated events
- ❑ Must remove redundantly logged events
- ❑ Might employ causally consistent snapshots to cope with length of logs

Related Work

- Delta Debugging
- Program Flow Analysis
- Instrumentation (Tracing)
- Bug Detection (Invariant Checking, Model Checking)
- Replay
- Root Cause Analysis

Instrumentation Complexity

- ❑ Code to override gettimeofday(),
interpose on logging statements, and
multiplex sockets:
- ❑ 415 LOC for POX (Python)
- ❑ 722 LOC for Floodlight (Java)

Future Work

- ❑ Many improvements:
 - ❑ Parallelize delta debugging
 - ❑ Smarter delta debugging time splits
 - ❑ Apply program flow analysis to further prune
 - ❑ Compress time (override gettimeofday)
- ❑ Apply to other distributed systems