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

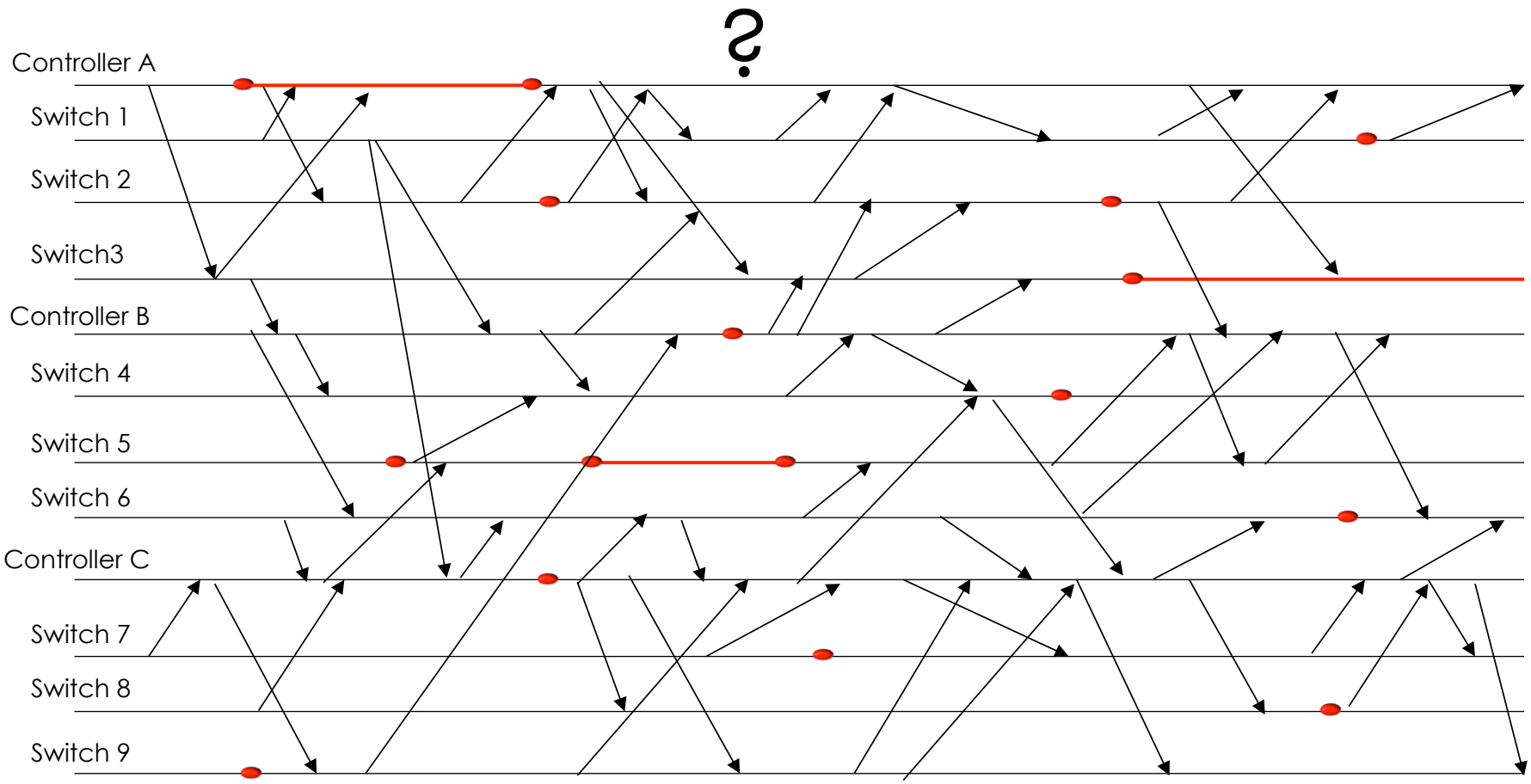
Something goes wrong!

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

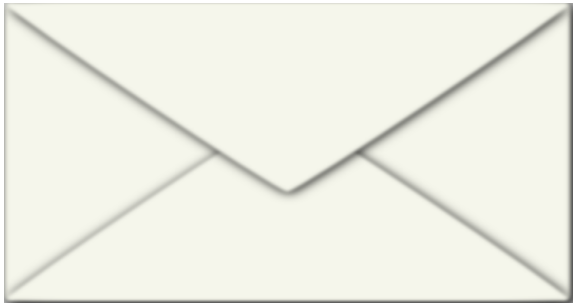
Best practice: Logs

Human analysis of log files

Best practice: Logs



How many events?



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

How many events?

= ~500 inputs / minute

Our Goal

Identify a **minimal** sequence of inputs that triggers the bug

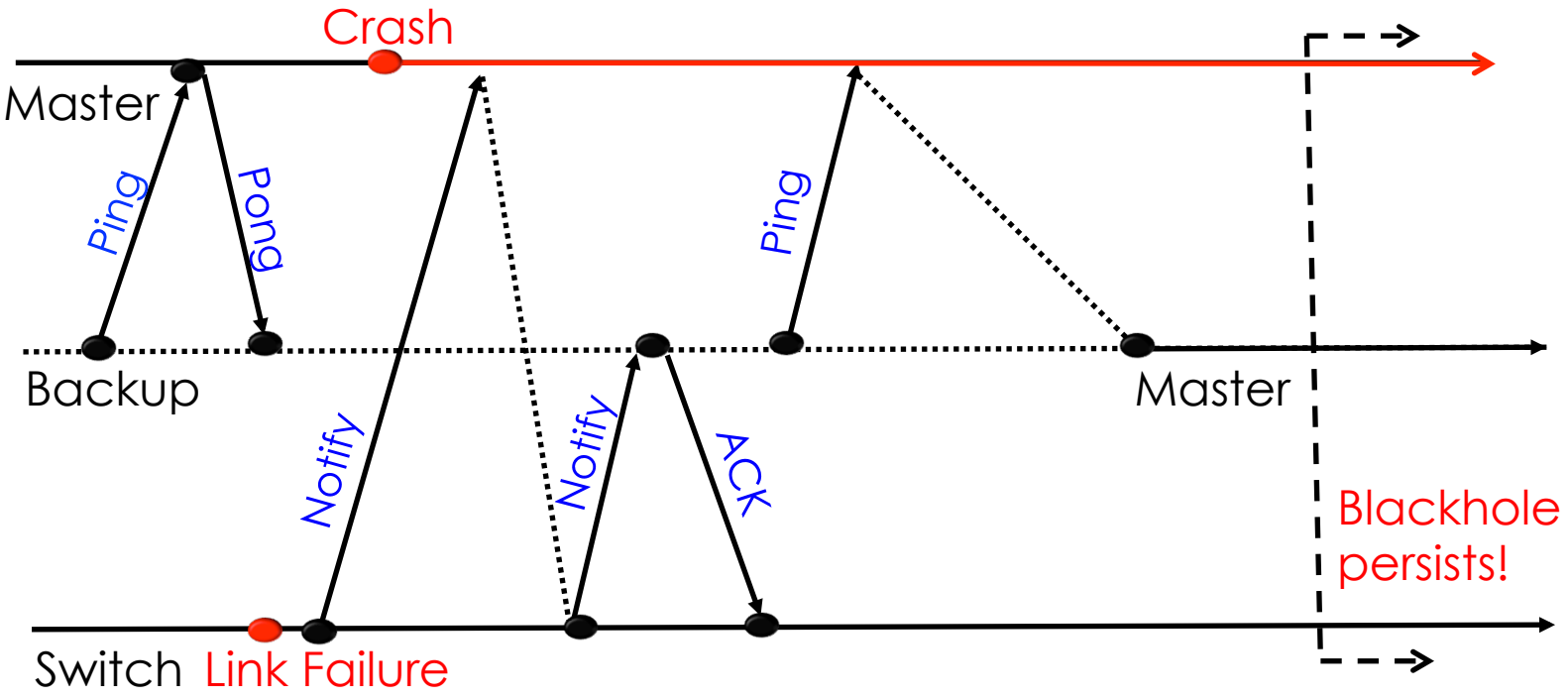
Minimal Causal Sequence

$$MCS \subset Trace \; s.t.$$

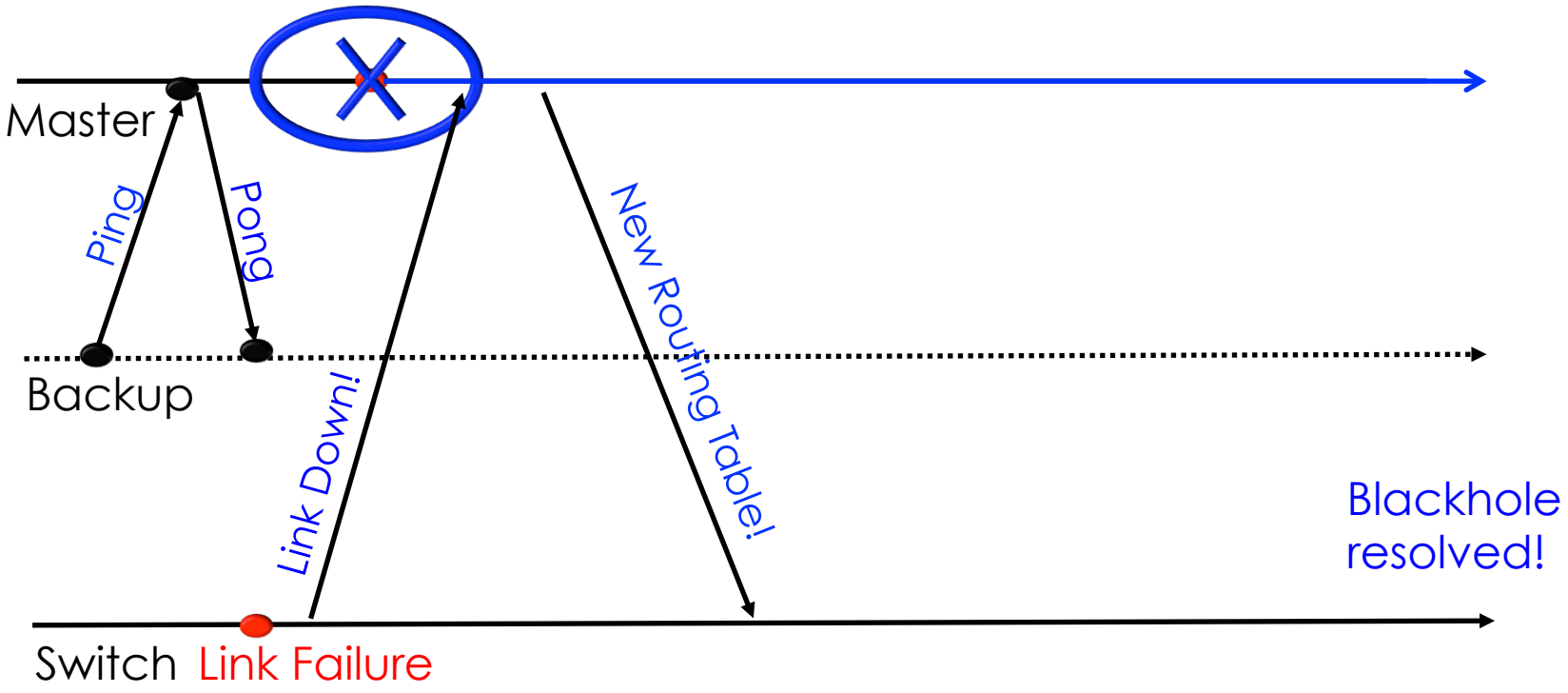
$$i.\; replay(MCS) = \chi$$

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

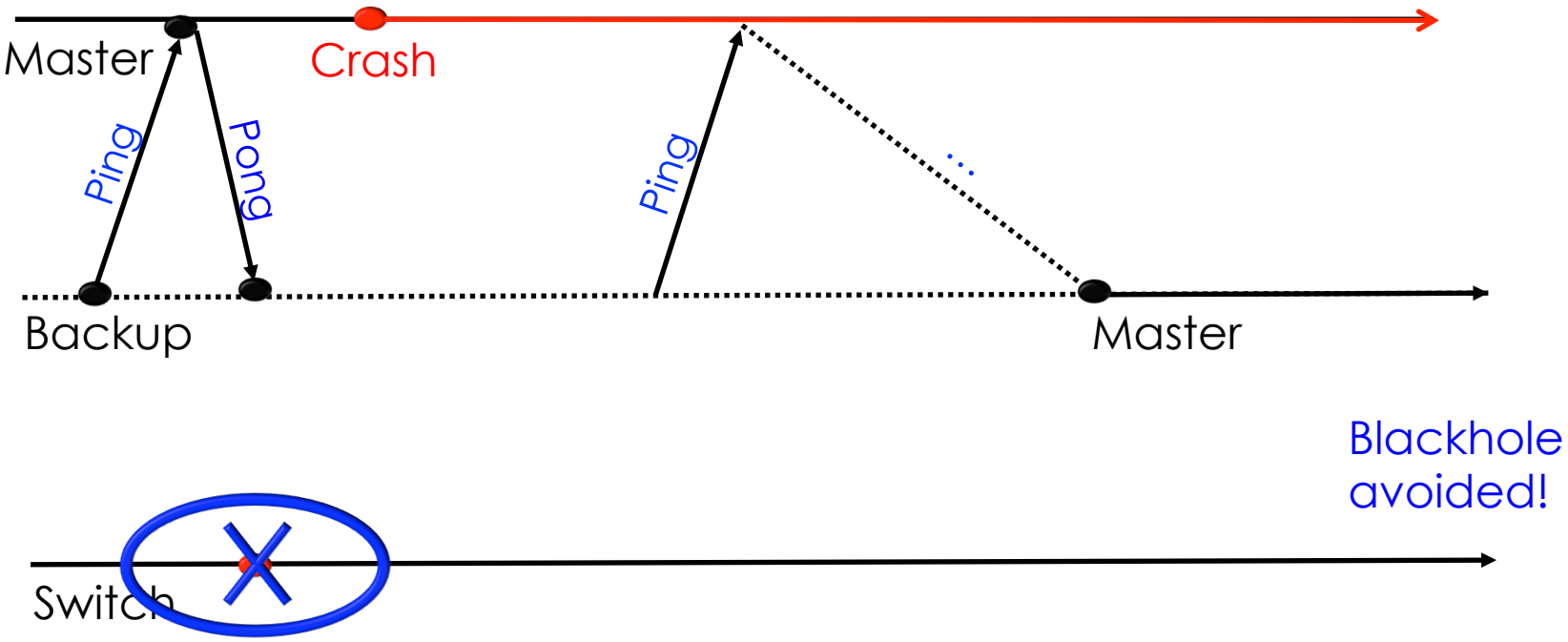
Minimal Causal Sequence



Minimal Causal Sequence

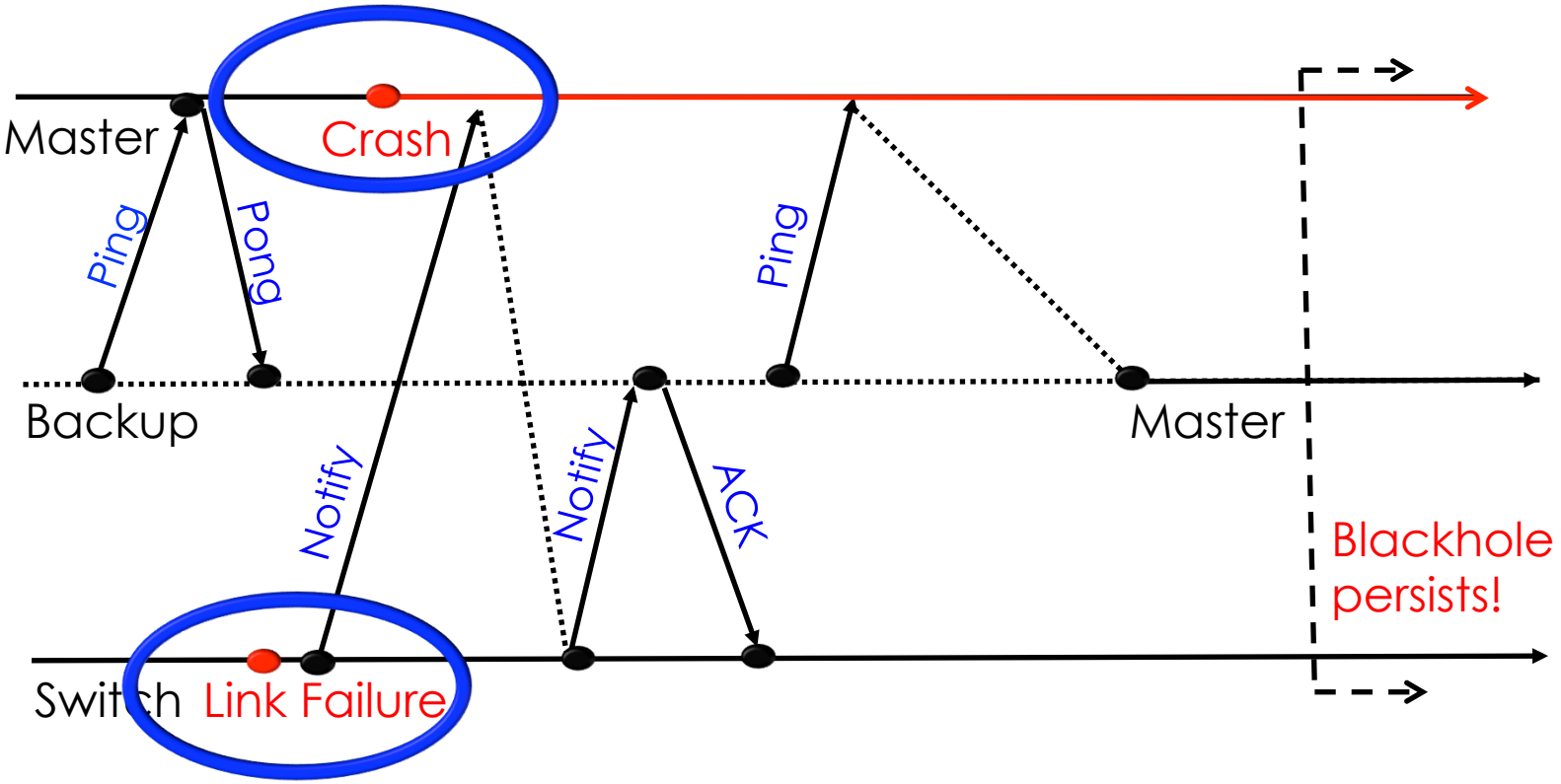


Minimal Causal Sequence

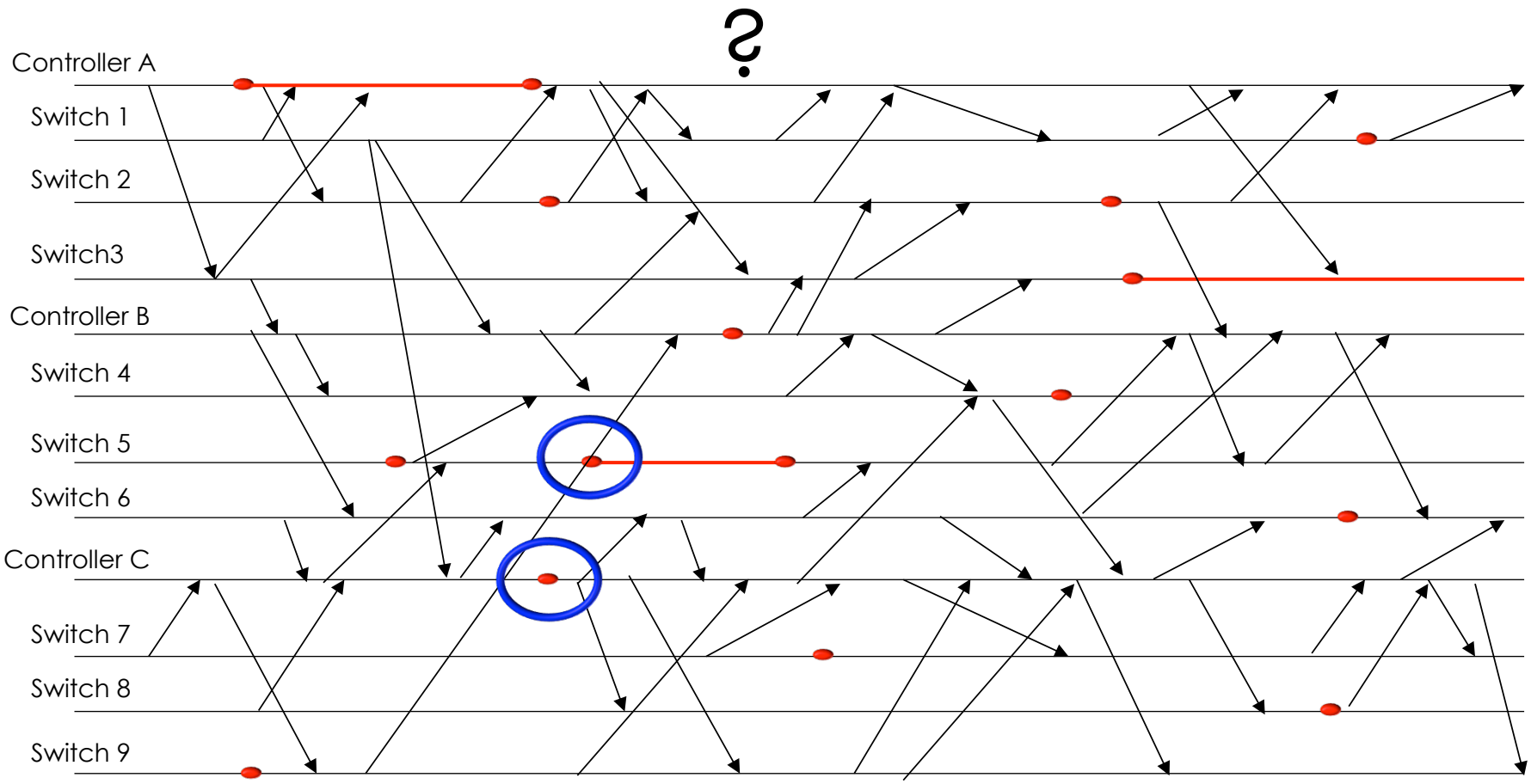


Minimal Causal Sequence

Both are necessary conditions!



Minimal Causal Sequence



Approach: Delta Debugging

Modify history!

Possible failure causes

Set up first hypothesis

Test first hypothesis

Second hypothesis

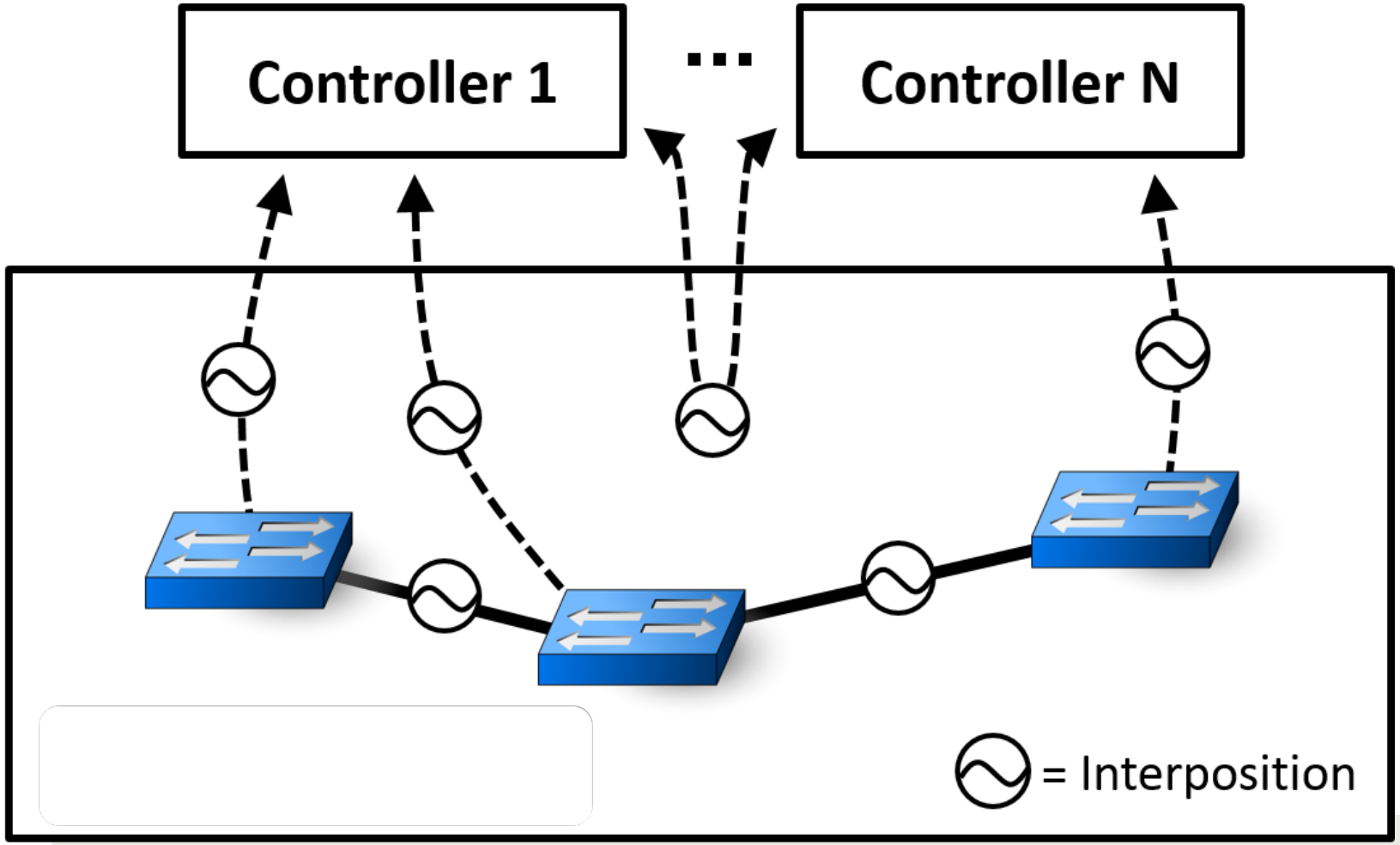
Third hypothesis

Fourth hypothesis...

Challenges

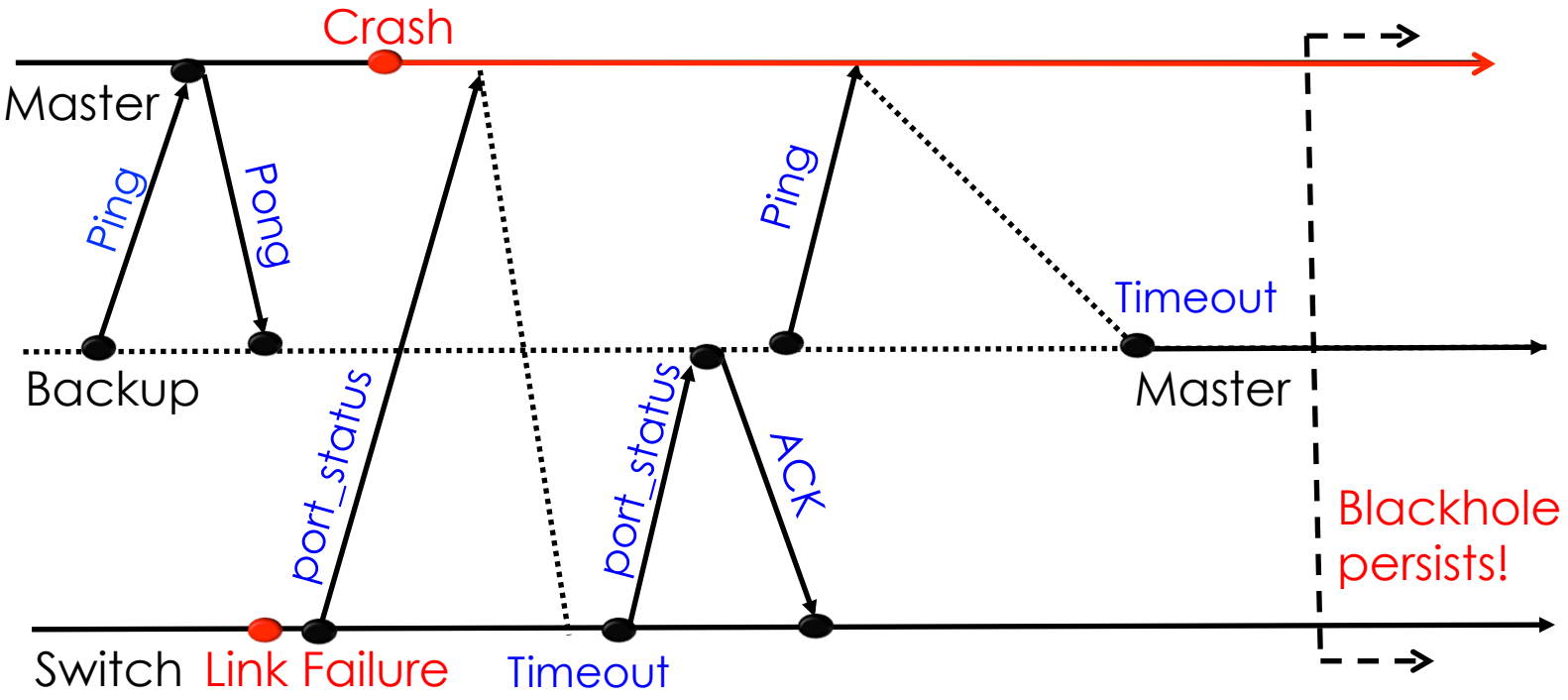
- Must carefully control timing of input events, despite
  - Modified history
  - Asynchrony
  - Non-determinism
- Without assuming language or instrumentation of control software

Replay Infrastructure



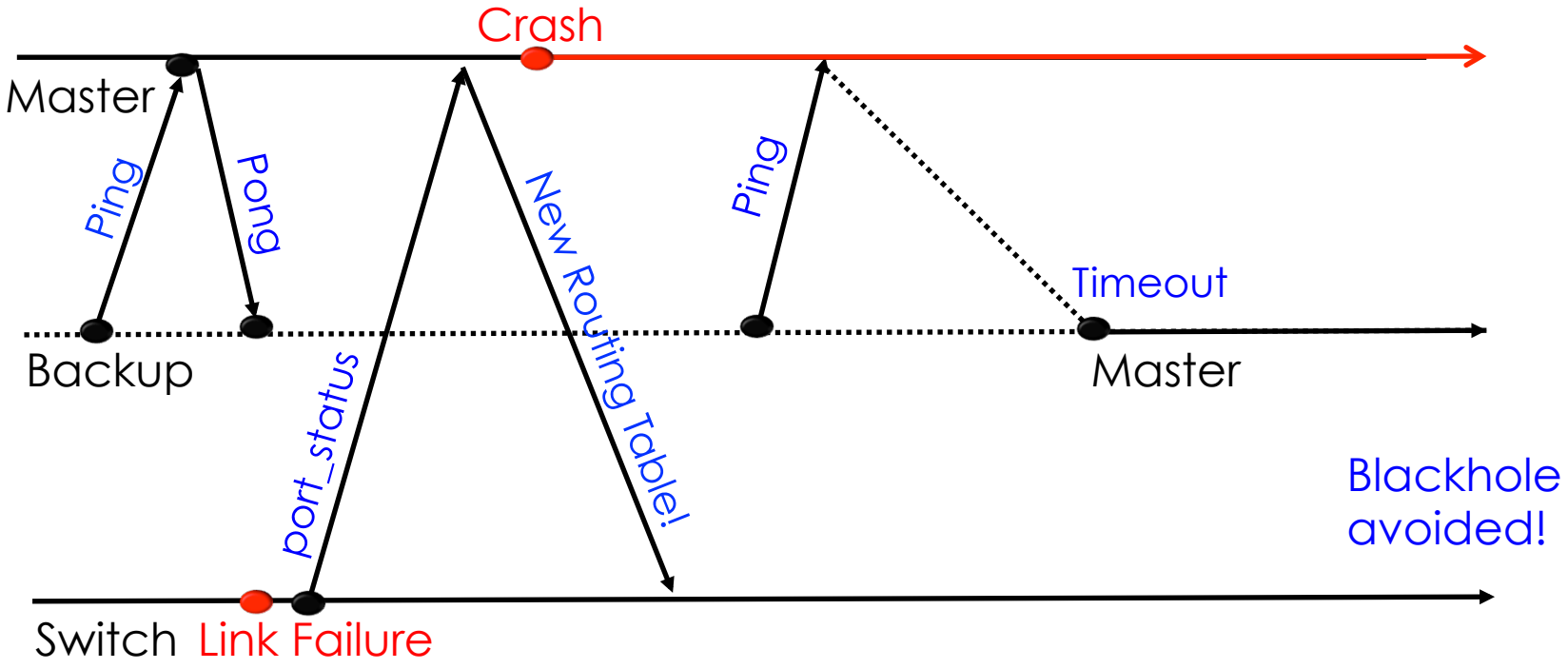
Replaying Pruned Inputs is Hard

Must consider internal events



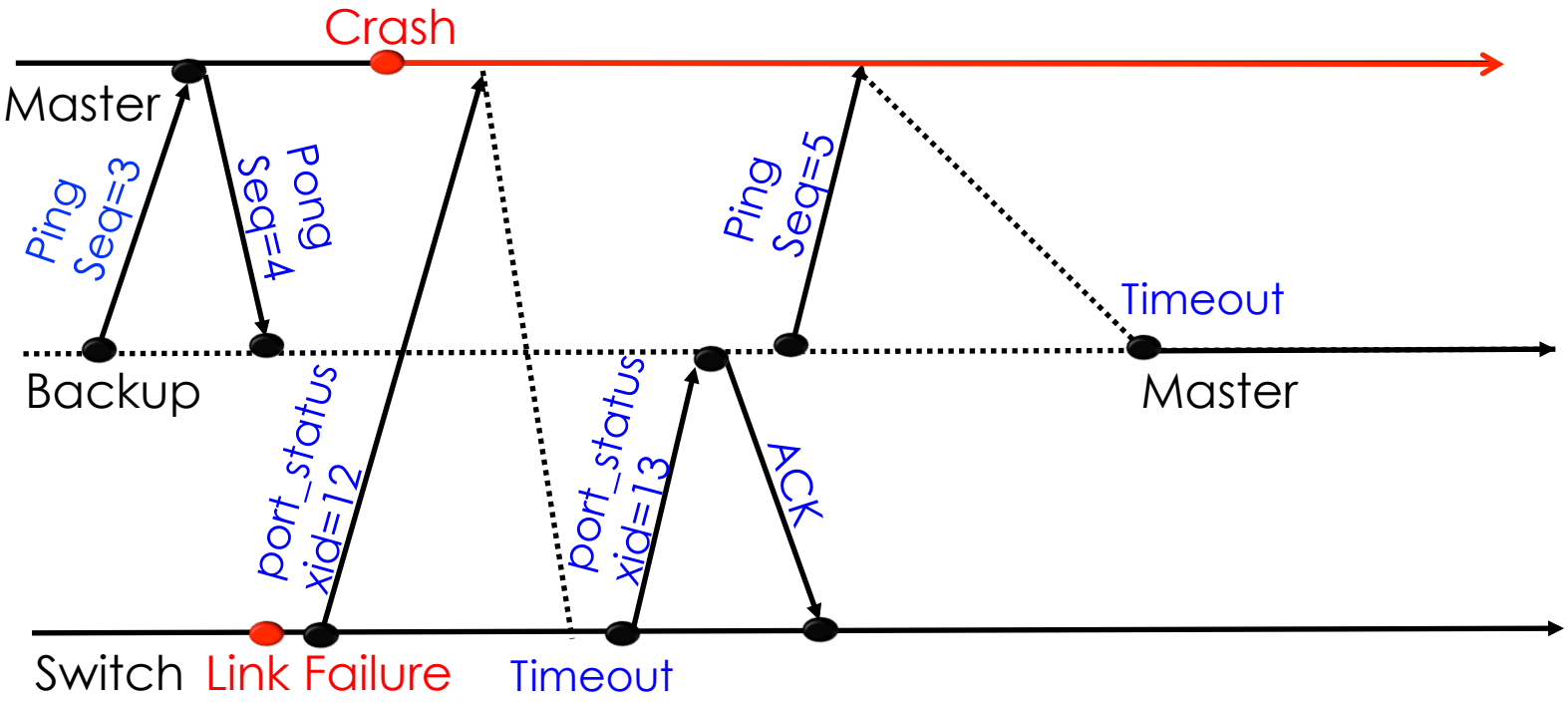
Replaying Pruned Inputs is Hard

Must consider internal events



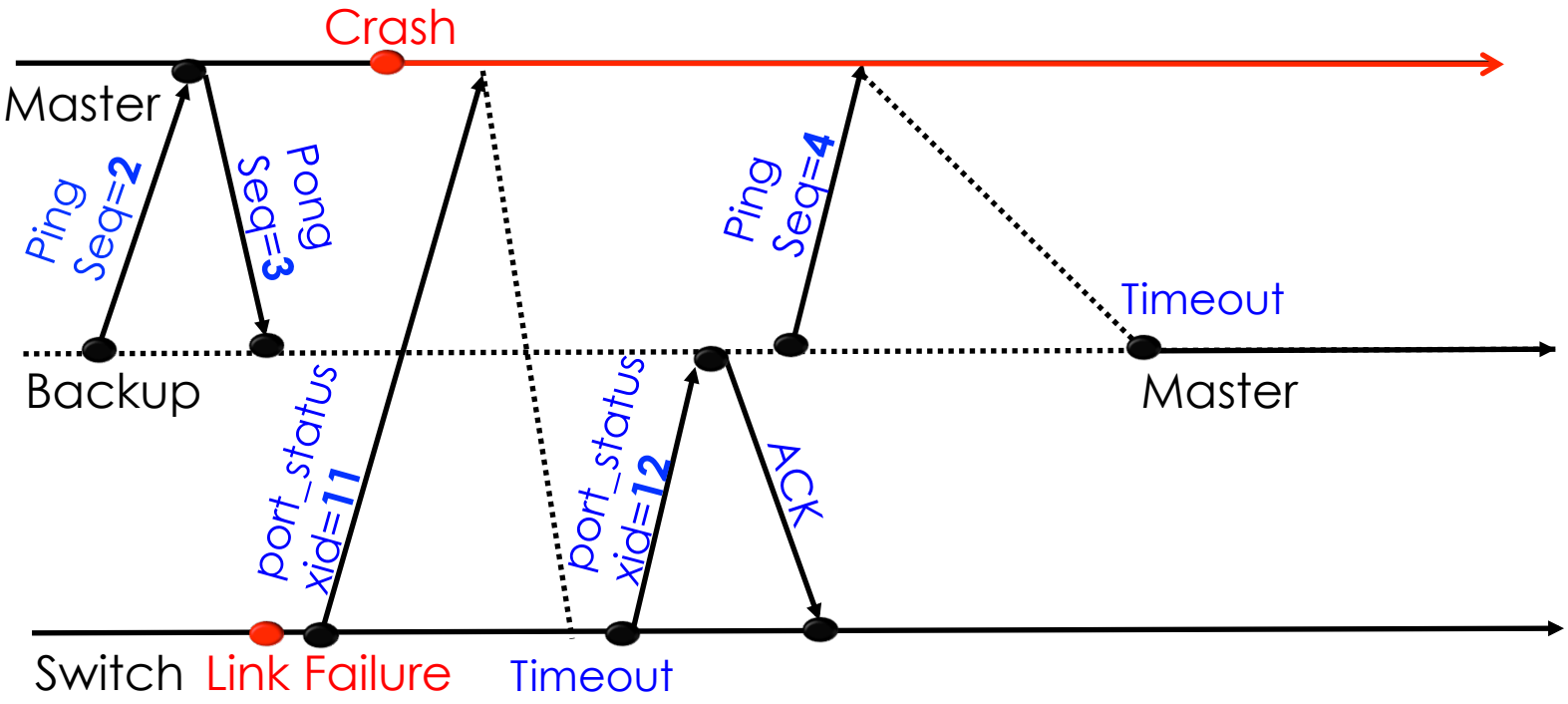
Pruning Alters Internal Events!

Prune Earlier Input..



Pruning Alters Internal Events!

Sequence Numbers Differ!



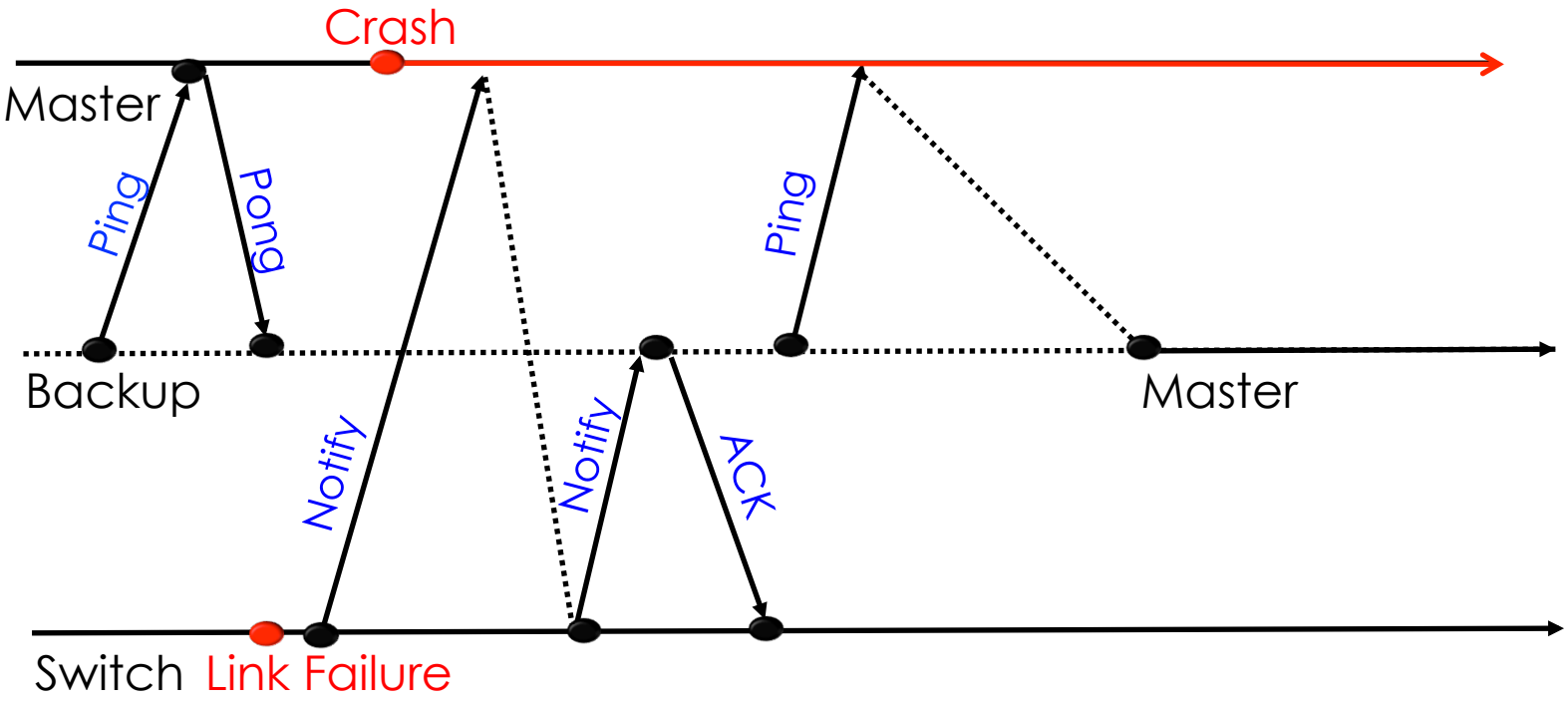
Solution: Equivalence Classes

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

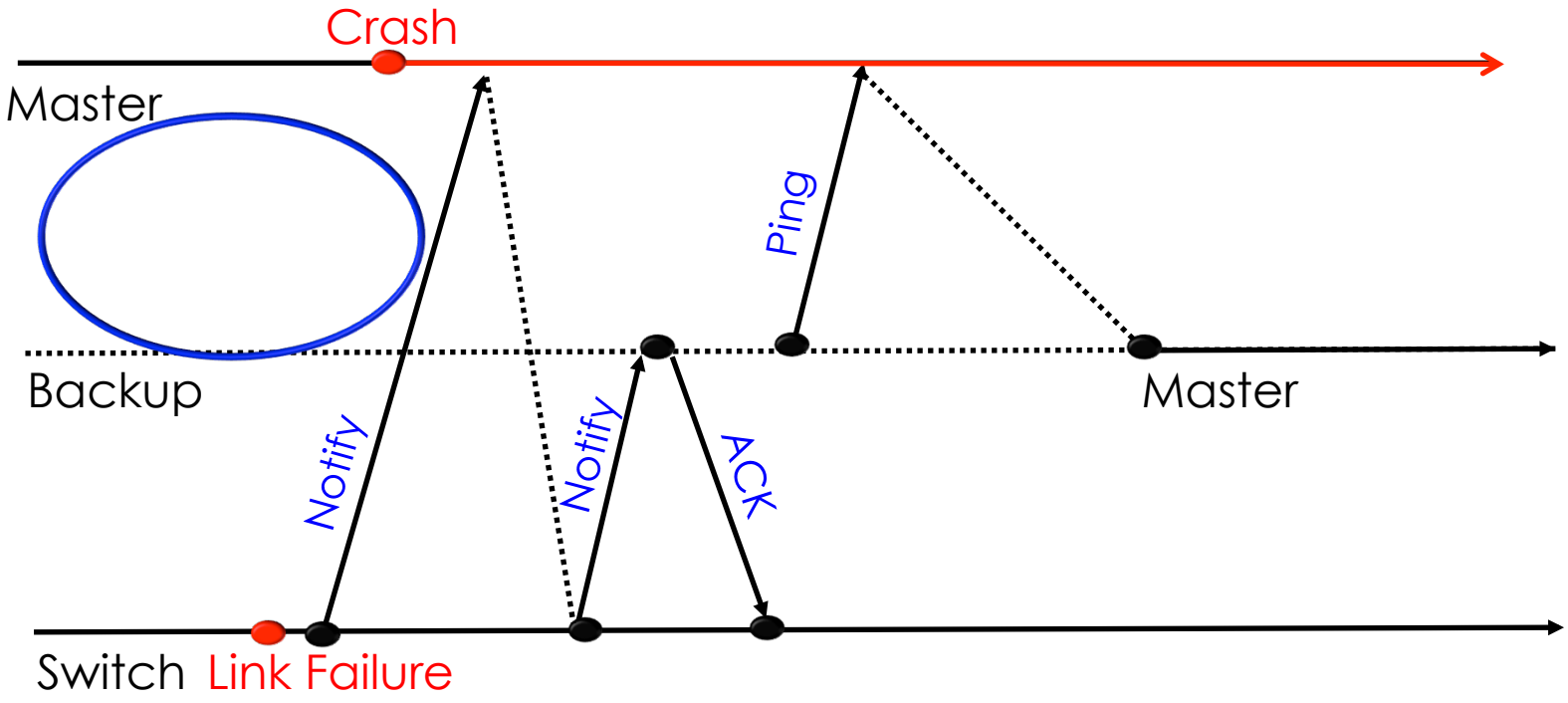
Pruning Alters Internal Events!

Prune Earlier Input..



Pruning Alters Internal Events!

Some Events No Longer Appear



Solution: Peek ahead

procedure PEEK(*input subsequence*)

$inferred \leftarrow [\,]$

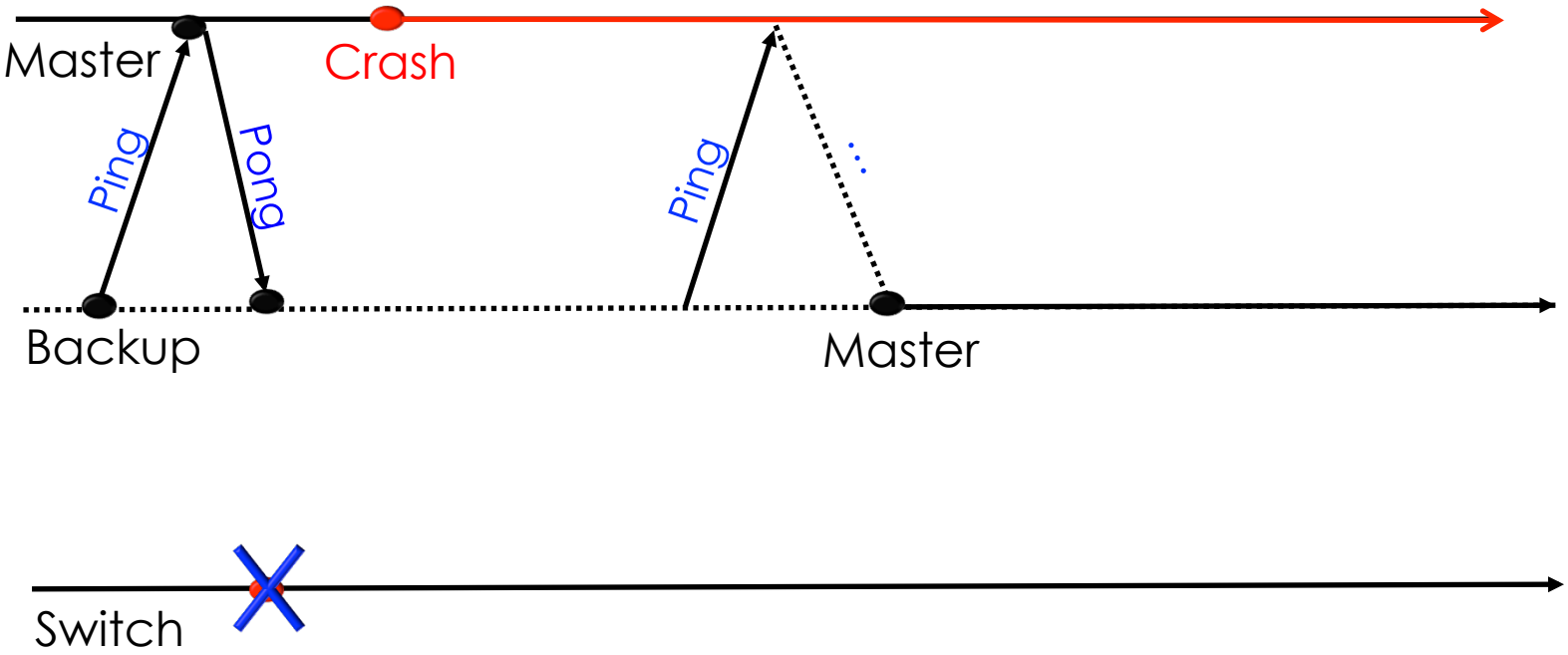
for e_i *in subsequence*

- $checkpoint\ system$
- $inject\ e_i$
- $\Delta \leftarrow |e_{i+1}.time - e_i.time| + \epsilon$
- $record\ events\ for\ \Delta\ seconds$
- $matched \leftarrow original\ events\ \&\ recorded\ events$
- $inferred \leftarrow inferred + [e_i] + matched$
- $restore\ checkpoint$

return $inferred$

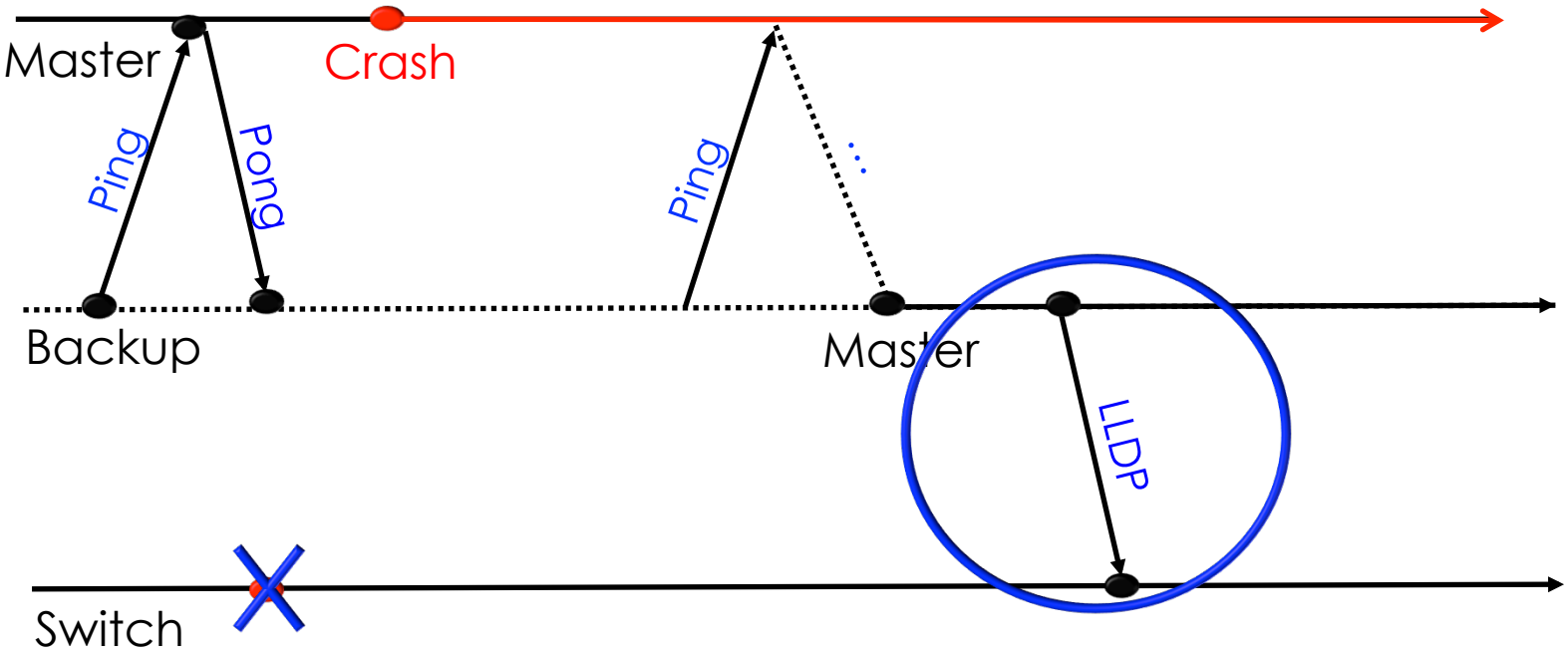
Pruning Alters Internal Events!

Prune Input..



Pruning Alters Internal Events!

Unexpected Events Appear



Dealing with unexpected events

Theory:

- Divergent paths → Exponential possibilities

Practice:

- Allow unexpected events through

Coping With Non-Determinism

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

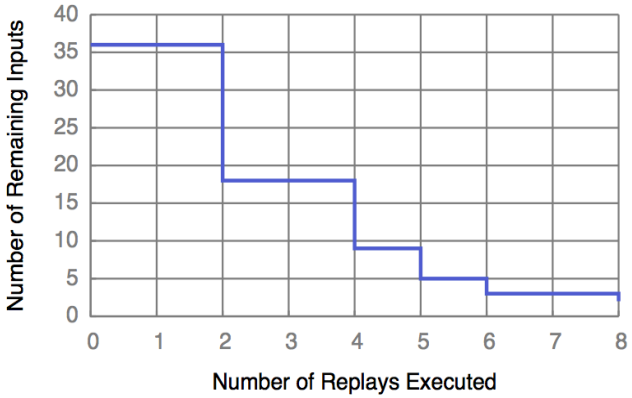
$$f(p,n) = (1-p)^n$$

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

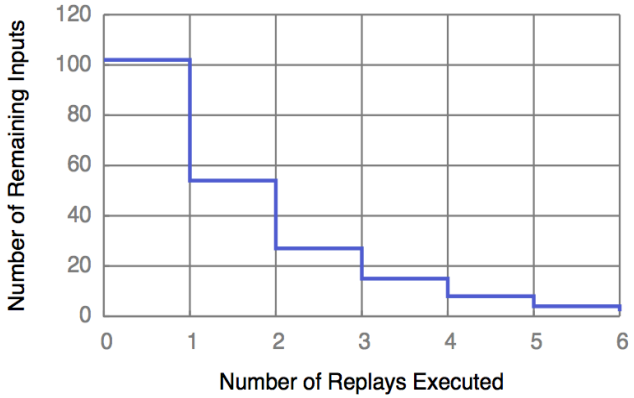
It Works!

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

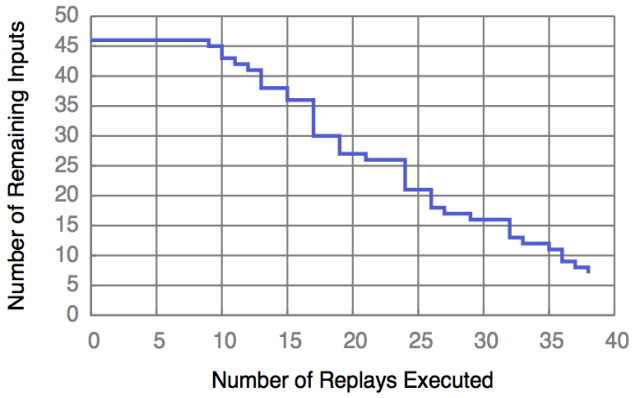
It Works!



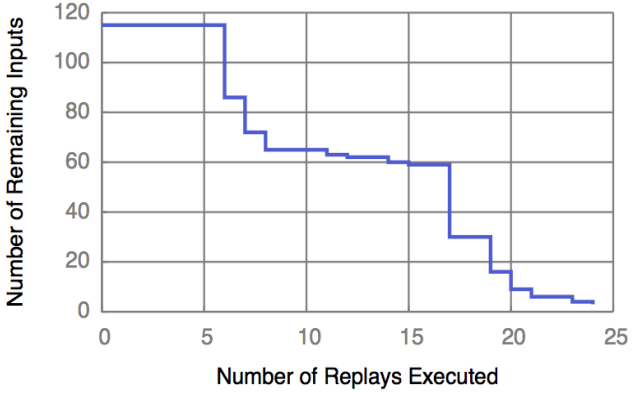
(a) Pyretic loop.



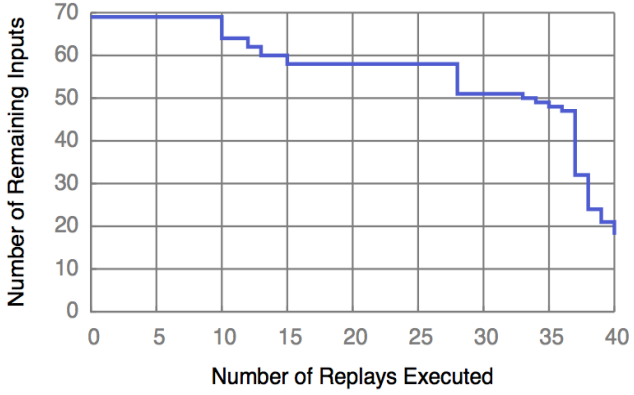
(b) POX Premature Packet-In Failure.



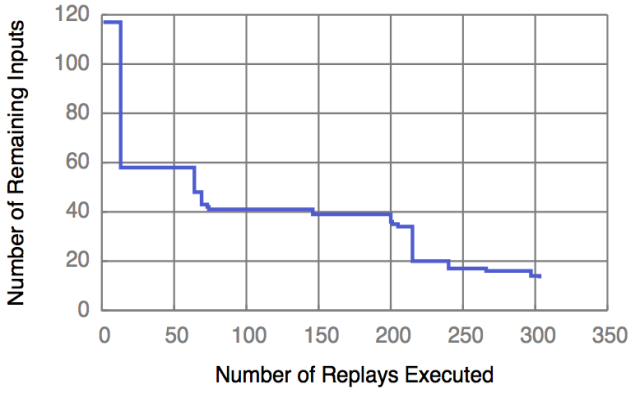
(c) POX in-flight blackhole.



(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/

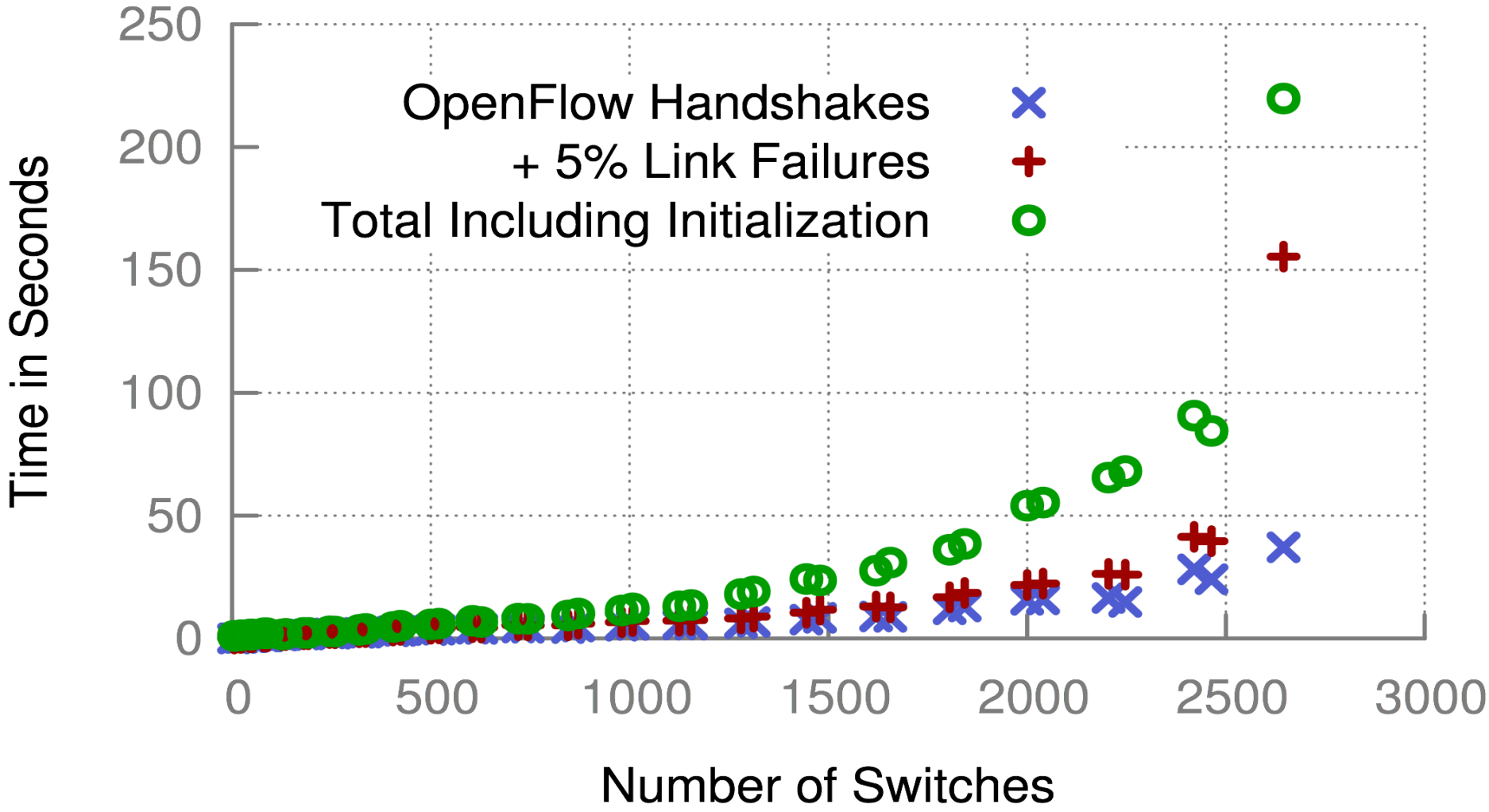
References

- [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**
- Delta Debugging: Ω(log n) replays - Each replay: O(n) events - Total: Ω(nlog n)	- 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:* $\forall c' \subseteq c_\chi \cdot \left(|c'| < |c| \Rightarrow test(c') \neq \chi\right)$ *holds.*

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