Troubleshooting SDN Control Software with Minimal Causal Sequences

Colin Scott, Andreas Wundsam, Barath Raghavan, Aurojit Panda, Andrew Or, Jefferson Lai, Eugene Huang, Zhi Liu, Ahmed El-Hassany, Sam Whitlock, Hrishikesh B. Acharya, Kyriakos Zarifis, Arvind Krishnamurthy, Scott Shenker
SDN is a Distributed System
Distributed Systems are Bug-Prone

Distributed correctness faults:

- Race conditions
- Atomicity violations
- Deadlock
- Livelock
- ...

+ Normal software bugs
Example Bug (Floodlight, 2012)

Diagram showing a sequence of events including:
- Link Failure
- Notify
- Ping
- Pong
- Crash
- ACK
- Notify

With a note: Blackhole persists!
Human analysis of log files
Best Practice: Logs

- **Ping**
- **Pong**
- **Notify**
- **ACK**
- **Crash**
- **Link Failure**
- **Notify**
- **ACK**
- **Master**
- **Backup**

Blackhole persists!
Best Practice: Logs
Our Goal

Allow developers to focus on fixing the underlying bug
Problem Statement

Identify a **minimal** sequence of inputs that triggers the bug in a blackbox fashion.
Why minimization?

Smaller event traces are easier to understand

G. A. Miller. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. Psychological Review '56.
Minimal Causal Sequence

Output:

\[ \text{MCS} \subset \text{Trace s.t.} \]

i. \( \text{replay(MCS)} \ni \exists V \quad (\text{i.e. violation occurs}) \)

ii. \( \forall e \in \text{MCS} \ni \text{replay(MCS} - \{e\}) \not\ni V \)
Minimal Causal Sequence

Controller A
Switch 1
Switch 2
Switch 3

Controller B
Switch 4
Switch 5
Switch 6

Controller C
Switch 7
Switch 8
Switch 9

?
Minimal Causal Sequence

Master

Backup

Switch

Crash

Link Failure

Notify

Ping

Pong

Notify

ACK

Ping

Blackhole persists!
Outline

• What are we trying to do?
• How do we do it?
• Does it work?
Where Bugs are Found

• Symptoms found:
  • On developer’s local machine (unit and integration tests)
Where Bugs are Found

- Symptoms found:
  - On developer’s local machine (unit and integration tests)
  - In production environment
Where Bugs are Found

• Symptoms found:
  • On developer’s local machine (unit and integration tests)
  • In production environment
  • On quality assurance testbed
Approach: Delta Debugging\(^1\) Replay

1. A. Zeller et al. Simplifying and Isolating Failure-Inducing Input. IEEE TSE '02
Approach: Modify Testbed

Controller 1

Controller N

Test Coordinator

Control Software
QA Testbed
Testbed Observables

- Invariant violation detected by testbed

- Event Sequence:
  \[ \tau_L = e_1 \rightarrow i_1 \rightarrow i_2 \rightarrow e_2 \rightarrow \cdots e_m \rightarrow \cdots i_p \]

  - External events (link failures, host migrations,..) injected by testbed

  \[ E_L = e_1, e_2 \ldots e_m \]

  - Internal events (message deliveries) observed by testbed (incomplete)

  \[ I_L = i_1, i_2 \ldots i_p \]
Approach: Delta Debugging\(^1\) Replay

Events (link failures, crashes, host migrations) injected by test orchestrator

1. A. Zeller et al. Simplifying and Isolating Failure-Inducing Input. IEEE TSE '02
Key Point

Must Carefully Schedule Replay Events To Achieve Minimization!
Challenges

• Asynchrony

• Divergent execution

• Non-determinism
Challenge: Asynchrony

Asynchrony definition:
- No fixed upper bound on relative speed of processors
- No fixed upper bound on time for messages to be delivered

Dwork & Lynch. Consensus in the Presence of Partial Synchrony. JACM ’88
Challenge: Asynchrony

Need to maintain original event order

- Crash
- Ping
- Pong
- port_status
- ACK
- Timeout
- Switch
- Link Failure
- Master
- Backup
- Blackhole persists!
Challenge: Asynchrony

Need to maintain original event order
Coping with Asynchrony

Use interposition to maintain causal dependencies
Challenge: Divergence

- Asynchrony
- Divergent execution
  - Syntactic Changes
  - Absent Events
  - Unexpected Events
- Non-determinism
Divergence: Absent Internal Events

Prune Earlier Input..

- Master
- Backup
- Switch
- Crash
- Ping
- Pong
- Notify
- ACK
- Link Failure
- Notify
- Host Migration
- Policy change
- Switch
- Link Failure
- Notify
- ACK
- Host Migration
Divergence: Absent Internal Events

Some Events No Longer Appear

- Crash
- Policy change
- Notify
- Link Failure
- Host Migration

Event Diagram:
- Master to Backup: Ping, Pong
- Switch: Link Failure
- Switch to Master: Notify
- Master to Switch: Policy change

Events that do not appear:
- Link Failure
- Host Migration
Solution: Peek Ahead

Infer which internal events will occur

- Master
- Backup
- Switch
- Crash
- Ping
- Pong
- Notify
- Link Failure
- Host Migration
- Policy change
Challenge: Non-determinism

- Asynchrony
- Divergent execution
- Non-determinism
Coping With Non-Determinism

- Replay multiple times per subsequence
- Assuming i.i.d., probability of not finding bug modeled by:

\[ f(p, n) = (1 - p)^n \]

- If not i.i.d., override gettimeofday(), multiplex sockets, interpose on logging statements
Approach Recap

- Replay events in QA testbed
- Apply delta debugging to inputs
- Asynchrony: interpose on messages
- Divergence: infer absent events
- Non-determinism: replay multiple times
Outline

• What are we trying to do?

• How do we do it?

• Does it work?
Evaluation Methodology

• Evaluate on 5 open source SDN controllers (Floodlight, NOX, POX, Frenetic, ONOS)

• Quantify minimization for:
  • Synthetic bugs
  • Bugs found in the wild

• Qualitatively relay experience troubleshooting with MCSes
Case Studies

17 case studies total
Substantial minimization except for 1 case
Conservative input sizes

- Discovered Bugs
- Known Bugs
- Synthetic Bugs

Number of Input Events

- Pyretic Loop
- POX Premature Packetin
- POX In-Flight Blackhole
- NOX Discovery Loop
- Floodlight Loop
- Floodlight Failover
- ONOS Database Locking
- ONOS Master Election
- Delicate Timer Interleaving
- Reactive Routing Trigger
- Overlapping Flow Entries
- Null Pointer
- Multithreaded Race Condition
- Memory Leak
- Memory Corruption

Input size and MCS size

Not replayable

1596, 719
Comparison to Naïve Replay

- Naïve replay: ignore internal events
- Naïve replay often not able to replay at all
  - 5 / 7 discovered bugs not replayable
  - 1 / 7 synthetic bugs not replayable
- Naïve replay did better in one case
  - 2 event MCS vs. 7 event MCS with our techniques
Qualitative Results

- 15 / 17 MCSes useful for debugging
  - 1 non-replayable case (not surprising)
  - 1 misleading MCS (expected)
Related Work


[38] G. A. Miller. The Magical Number Seven, Plus or Minus Two: Some Limits On Our Capacity for Processing Information. Psychological Review ’56.


Conclusion

- Possible to automatically minimize execution traces for SDN control software
- System (23K+ lines of Python) evaluated on 5 open source SDN controllers (Floodlight, NOX, POX, Frenetic, ONOS) and one proprietary controller

ucb-sts.github.com/sts/
- Currently generalizing, formalizing approach
Backup
Related work

• **Thread Schedule Minimization**
  - Isolating Failure-Inducing Thread Schedules. SIGSOFT ’02.
  - A Trace Simplification Technique for Effective Debugging of Concurrent Programs. FSE ’10.

• **Program Flow Analysis**
  - Enabling Tracing of Long-Running Multithreaded Programs via Dynamic Execution Reduction. ISSTA ’07.
  - Toward Generating Reducible Replay Logs. PLDI ’11.

• **Best-Effort Replay of Field Failures**
  - A Technique for Enabling and Supporting Debugging of Field Failures. ICSE ’07.
  - Triage: Diagnosing Production Run Failures at the User’s Site. SOSP ’07.
Bugs are costly and time consuming

• Software bugs cost US economy $59.5 Billion in 2002 [1]
• Developers spend ~50% of their time debugging [2]
• Best developers devoted to debugging

1. National Institute of Standards and Technology 2002 Annual Report
2. P. Godefroid et al., Concurrency at Microsoft- An Exploratory Study. CAV ‘08
Ongoing work

- Formal analysis of approach
- Apply to other distributed systems (databases, consensus protocols)
- Investigate effectiveness of various interposition points
- Integrate STS into ONOS (ON.Lab) development workflow
Scalability

Time in Seconds

Number of Switches

OpenFlow Handshakes + 5% Link Failures
Total Including Initialization
Case Studies

Techniques provide notable benefit vs. naïve replay

15 / 17 MCSes useful for debugging
## Case Studies

<table>
<thead>
<tr>
<th>Bug Name</th>
<th>Topology</th>
<th>Runtime (s)</th>
<th>Input Size</th>
<th>MCS Size</th>
<th>MCS WI</th>
<th>MCS Helpful?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Newly Found</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pyretic Loop</td>
<td>3 switch mesh</td>
<td>266.2</td>
<td>36</td>
<td>1</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>POX Premature PacketIn</td>
<td>4 switch mesh</td>
<td>249.1</td>
<td>102</td>
<td>2</td>
<td>NR</td>
<td>Yes</td>
</tr>
<tr>
<td>POX In-Flight Blackhole</td>
<td>2 switch mesh</td>
<td>1478.9</td>
<td>27</td>
<td>11</td>
<td>NR</td>
<td>Yes</td>
</tr>
<tr>
<td>POX Migration Blackhole</td>
<td>4 switch mesh</td>
<td>1796.0</td>
<td>29</td>
<td>3</td>
<td>NR</td>
<td>Yes</td>
</tr>
<tr>
<td>NOX Discovery Loop</td>
<td>4 switch mesh</td>
<td>4990.9</td>
<td>150</td>
<td>18</td>
<td>NR</td>
<td>Indirectly</td>
</tr>
<tr>
<td>Floodlight Loop</td>
<td>3 switch mesh</td>
<td>27930.6</td>
<td>117</td>
<td>13</td>
<td>NR</td>
<td>Yes</td>
</tr>
<tr>
<td>ONOS Database Locking</td>
<td>2 switch mesh</td>
<td>N/A</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Known</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Floodlight Failover</td>
<td>2 switch mesh</td>
<td>-</td>
<td>202</td>
<td>2</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>ONOS Master Election</td>
<td>2 switch mesh</td>
<td>2746.0</td>
<td>20</td>
<td>2</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>POX Load Balancer</td>
<td>3 switch mesh</td>
<td>2396.7</td>
<td>106</td>
<td>24 (N+1)</td>
<td>26</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Synthetic</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delicate Timer Interleaving</td>
<td>3 switch mesh</td>
<td>N/A</td>
<td>39</td>
<td>NR</td>
<td>NR</td>
<td>No</td>
</tr>
<tr>
<td>Reactive Routing Trigger</td>
<td>3 switch mesh</td>
<td>525.2</td>
<td>40</td>
<td>7</td>
<td>2</td>
<td>Indirectly</td>
</tr>
<tr>
<td>Overlapping Flow Entries</td>
<td>2 switch mesh</td>
<td>115.4</td>
<td>27</td>
<td>2</td>
<td>3</td>
<td>Yes</td>
</tr>
<tr>
<td>Null Pointer</td>
<td>20 switch FatTree</td>
<td>157.4</td>
<td>62</td>
<td>2</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>Multithreaded Race Condition</td>
<td>10 switch mesh</td>
<td>36967.5</td>
<td>1596</td>
<td>2</td>
<td>2</td>
<td>Indirectly</td>
</tr>
<tr>
<td>Memory Leak</td>
<td>2 switch mesh</td>
<td>15022.6</td>
<td>719</td>
<td>32 (M+2)</td>
<td>33</td>
<td>Indirectly</td>
</tr>
<tr>
<td>Memory Corruption</td>
<td>4 switch mesh</td>
<td>145.7</td>
<td>341</td>
<td>2</td>
<td>2</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Runtime

(a) Pyretic Loop.

(b) POX Premature PacketIn.

(c) POX In-Flight Blackhole.

(d) POX Migration Blackhole.

(e) NOX Discovery Loop.

(f) Floodlight Loop.
Coping with Non-Determinism

Size of Final MCS

Maximum Number of Replays per Subsequence
Replay Requirements

- Need to maintain original happens-before relation
- Includes *internal* events
- Message Deliveries
- State Transitions
Naïve Replay Approach

Schedule events according to wall-clock time
## Complexity

<table>
<thead>
<tr>
<th>Best Case</th>
<th>Worst Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Delta Debugging: $\Omega(\log n)$ replays</td>
<td>- Delta Debugging: $O(n)$ replays</td>
</tr>
<tr>
<td>- Each replay: $O(n)$ events</td>
<td>- Each replay: $O(n)$ events</td>
</tr>
<tr>
<td>- Total: $\Omega(n\log n)$</td>
<td>- Total: $O(n^2)$</td>
</tr>
</tbody>
</table>
Assumptions of Delta Debugging

- **Monotonic:**
  \[ P \oplus C = \chi \Rightarrow P \oplus (C \cup C') \neq \checkmark \]

- **Unambiguous:**
  \[ P \oplus C = \chi \land P \oplus C' = \chi \Rightarrow P \oplus (C \cap C') \neq \checkmark \]

- **Consistent**
  \[ P \oplus C \neq ? \]
**Definition 8 (Global minimum).** A set \( c \subseteq c_\chi \) is called the global minimum of \( c_\chi \) 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
Instrumentation Complexity

- Code to override gettimeofday(), interpose on logging statements, and multiplex sockets:

  - 415 LOC for POX (Python)
  - 722 LOC for Floodlight (Java)
Improvements

- Many improvements:
  - Parallelize delta debugging
  - Smarter delta debugging time splits
  - Apply program flow analysis to further prune
  - Compress time (override gettimeofday)
Divergence: Syntactic Changes

Prune Earlier Input..

Diagram showing the flow of packets and messages between Master, Backup, and Switch. Key events include:
- **Ping Seq=3** from Backup to Master
- **Ping Seq=4** from Backup to Master
- **Ping Seq=5** from Backup to Master
- **Port_status xid=12** from Backup to Master
- **Port_status xid=13** from Backup to Master
- **ACK** from Master to Backup
- **Crash** event
- **Timeout** event
- **Link Failure** event

The diagram illustrates the divergence and the process of pruning earlier input.
Divergence: Syntactic Changes

Sequence Numbers Differ!

Master

Backup

Switch

Link Failure

Crash

Timeout

Port Status

xid=11

xid=12

Ping

Seq=2

Seq=3

Ping

Seq=4

Ping

Seq=4

Timeout

ACK
**Solution: Equivalence Classes**

**Mask Over Extraneous Fields**

<table>
<thead>
<tr>
<th>Internal message</th>
<th>Masked values</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenFlow messages</td>
<td>xac id, cookie, buffer id, stats</td>
</tr>
<tr>
<td>packet_out/in payload</td>
<td>all values except src, dst, data</td>
</tr>
<tr>
<td>Log statements</td>
<td>varargs parameters to printf</td>
</tr>
</tbody>
</table>
Solution: Peek ahead

**procedure** `PEEK(input subsequence)`

`inferred ← []`

**for** `e_i` **in** `subsequence`

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

`restore checkpoint`

`return inferred`
Divergence: Unexpected Events

Prune Input..
Divergence: Unexpected Events Appear

Unexpected Events Appear

Master
Backup
Crash
Pong
Ping
Switch

LLDP
Solution: Empirical Heuristic

Theory:
• Divergent paths $\rightarrow$ Exponential possibilities

Practice:
• Allow unexpected events through