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)
Human analysis of log files
Best Practice: Logs

Diagram showing the sequence of events during a link failure.

- **Switch** sends a **Link Failure** notification to the **Backup**.
- **Backup** sends a **Notify** message to the **Master**.
- **Master** sends a **Ping** message to the **Backup**.
- **Backup** sends an **ACK** message to the **Master**.
- **Master** checks the connection and finds it failed.
- **Master** sends a **Ping** message to the **Backup**.
- **Backup** sends a **Notify** message to the **Master**.
- **Master** sends a **Ping** message to the **Backup**.
- **Backup** sends an **ACK** message to the **Master**.
- **Master** checks the connection and finds it failed.

**Blackhole persists!**
Best Practice: Logs

Controller A
Switch 1
Switch 2
Switch 3
Controller B
Switch 4
Switch 5
Switch 6
Controller C
Switch 7
Switch 8
Switch 9

...
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.
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:

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

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

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

Link Failure → Notify → Pong → Crash

Notify → ACK

Ping

Master 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\textsuperscript{1} Replay

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

Controller 1

Controller N

Control Software
QA Testbed

Test Coordinator
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

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

---

Switch Link Failure Timeout

Backup

Master

Ping Pong port_status

Timeout

ACK

Ping port_status

Timeout

Blackhole persists!
Challenge: Asynchrony

Need to maintain original event order

Blackhole avoided!
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..

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

Some Events No Longer Appear

Master

Backup

Switch

Link Failure

Notify

Crash

Ping

Pong

Ping

Policy change

Host Migration
Solution: Peek Ahead

Infer which internal events will occur

Master
Ping
Notify
Pong
Crash
Backup
Switch
Link Failure
Host Migration
Policy change

Master
Ping
Notify
Pong
Crash
Backup
Switch
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

Not replayable
1596 719
Not replayable
(n) (m)

Number of Input Events

Input size
MCS size

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


[16] Floodlight Controller.
  https://www.tinyurl.com/ntjx61

  http://tinyurl.com/afshn3j


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


[51] G. Tel. Introduction to Distributed Algorithms. Thm. 2.21.


[59] A. Zeller and R. Liddlebrandt. Simplifying and Isolating Faults in Distributed Networks. TOOLS ’02.


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

OpenFlow Handshakes + 5% Link Failures
Total Including Initialization

Time in Seconds

Number of Switches
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>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>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>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
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**
  \[ P \oplus C \neq ? \]
Local vs. Global Minimality

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

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

Master

Backup

Switch

Link Failure

Ping Seq=3

Pong Seq=4

port_status xid=12

port_status xid=13

ACK

Ping Seq=5

Timeout

Master

Crash

Timeout
Divergence: Syntactic Changes

Sequence Numbers Differ!

- Master
  - Ping Seq=2
  - Pong Seq=3
  - port_status xid=11
  - Timeout

- Backup
  - Crash
  - port_status xid=12
  - ACK
  - Timeout

- Switch
  - Link Failure

- Master
  - Ping Seq=4

- Timeout
## 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)

```plaintext
inferred ← []
for eᵢ in subsequence
  checkpoint system
  inject eᵢ
  Δ ← |eᵢ₊₁.time − eᵢ.time| + ε
  record events for Δ seconds
  matched ← original events & recorded events
  inferred ← inferred + [eᵢ] + matched
restore checkpoint
return inferred
```
Divergence: Unexpected Events

Prune Input..
Divergence: Unexpected Events

Unexpected Events Appear

Master
Backup
Crash
Ping
Pong
Switch
Master
LLDP
Solution: Empirical Heuristic

Theory:
• Divergent paths → Exponential possibilities

Practice:
• Allow unexpected events through