Using Time Travel to Avoid Bugs and Failures in SDN Applications
Zhenyu Zhou¹, Theophilus Benson¹, Marco Canini², Balakrishnan Chandrasekaran³
¹Duke University, ²KAUST, ³TU Berlin

Problem
How to devise an online technique to safely and systematically circumvent a bug that manifests even in a well-designed and well-tested real applications running in a production environment?

- Bugs are endemic in real applications.
- Network outages cost $1000s per minute.
- SDN controllers do not tolerate application failures.
- Simple failover strategies suffer from deterministic faults.

Design
Delorean provides a quick, safe, online recovery of the real applications, even in case of deterministic faults.

Key insight: Discover a set of possible code paths (i.e., path through the source code indicating the control flow) and drive the execution away from the crash path.

- App Manager: Record messages sent from/to the applications.
- Transaction Manager: Enable rollback of both control and data plane.
- Transformation Generator: Generate transformations for a given event.
- Recovery Optimizer: Rank the transformations for efficient tests.
- Crash Cause Analyzer: Find the crash cause that changes the code path to the crash path, i.e. it changes the state and its previous events do not fall into the crash path.
- Symbolic Execution Analyzer: Build the execution tree, which comprises all code paths, to help circumvent the crash path.

Workflow
When a crash happens:
1. Determine the crash path and the crash cause.
2. Produce transformations of the crash cause and rank them.
3. Restart the application and restore its states.
4. Inject a set of transformed events.
5. If current transformation fails, resume from (3) with the next one. If all transformations are exhausted, revert to (1) to find another crash cause from an earlier instance of time.

Challenges
Delorean addresses two fundamental challenges in crash recovery:
- Determine the precise input event to rollback.
- Determine the transformations to apply while minimizing recovery time and the likelihood of another crash.

Takeaway: Delorean aims to recover from both deterministic and non-deterministic bugs of real applications based on the insights provided by symbolic execution.

Evaluation
Experiment Setup:
Data plane: Mininet and simple traffic generators.
Control plane: Realistic Floodlight applications.

Key takeaways:
- Naive rollback strategies do not suffice for complex failures; a more principled approach is required.
- Delorean performs better and faster than naive strategies such as controller reboot or application reboot (both of which are similar to “last” in the figure).
- The time for symbolic execution does increase with more code paths, while not significantly: 2.33s, 2.61s and 2.78s for Hub (1 path), Learning Switch (9 paths) and Hedera (13 paths), respectively.

This work is partially supported by NSF grant CNS-1409426.