Tolerating Application Failures with LegoSDN

Balakrishnan Chandrasekaran

Theophilus Benson

Duke University

Quality of Code

"In C, I never learned to use the debugger, so I used to never make mistakes …"

"I went millions and millions of hours with no problems—probably tens of millions of hours with no problems."

— Arthur Whitney, creator of *A*, *K* and *Q*.

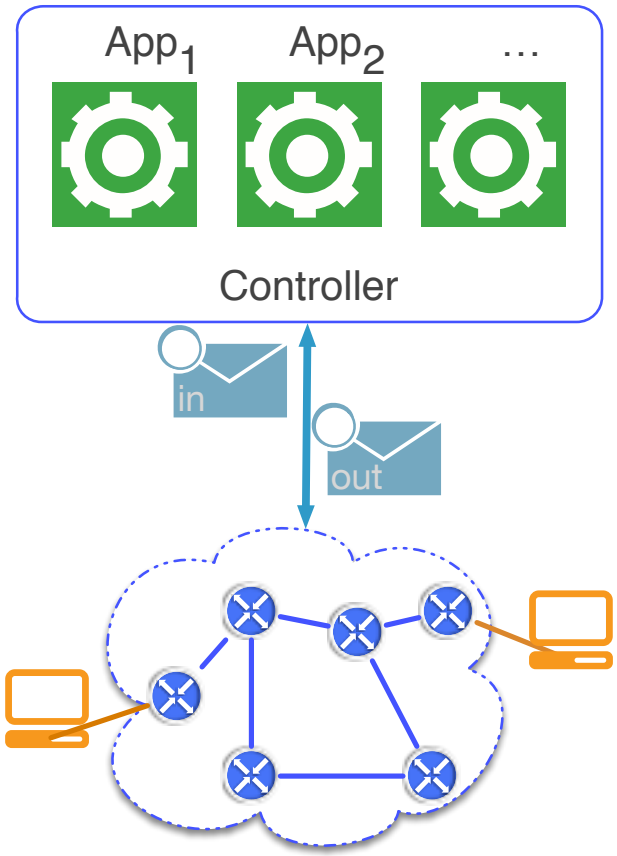
ACM Queue, Feb 2009.

Bugs are *endemic* in software!

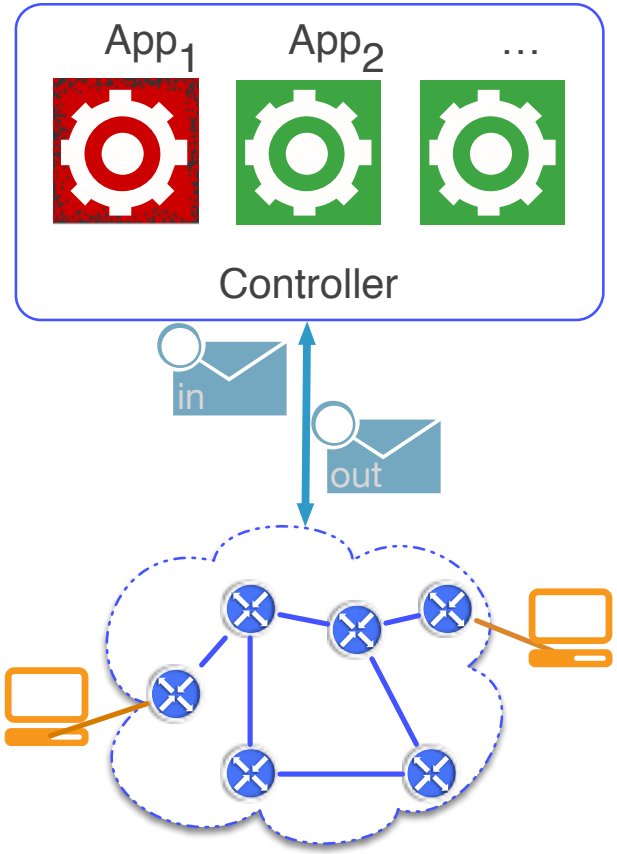
- Bugs can be *deterministic* or *non-deterministic*

- [STS] *Pox Premature PacketIn*
  - l2_multi routing module failed unexpectedly with a KeyError.

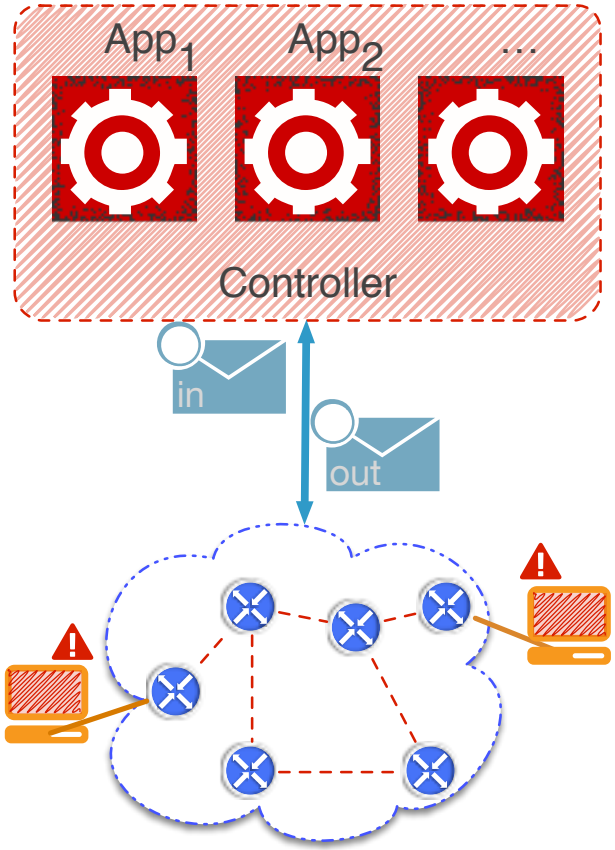
Cascading Crashes



Cascading Crashes



Cascading Crashes



LegoSDN

- *Availability* is of utmost importance
  - Second only to security

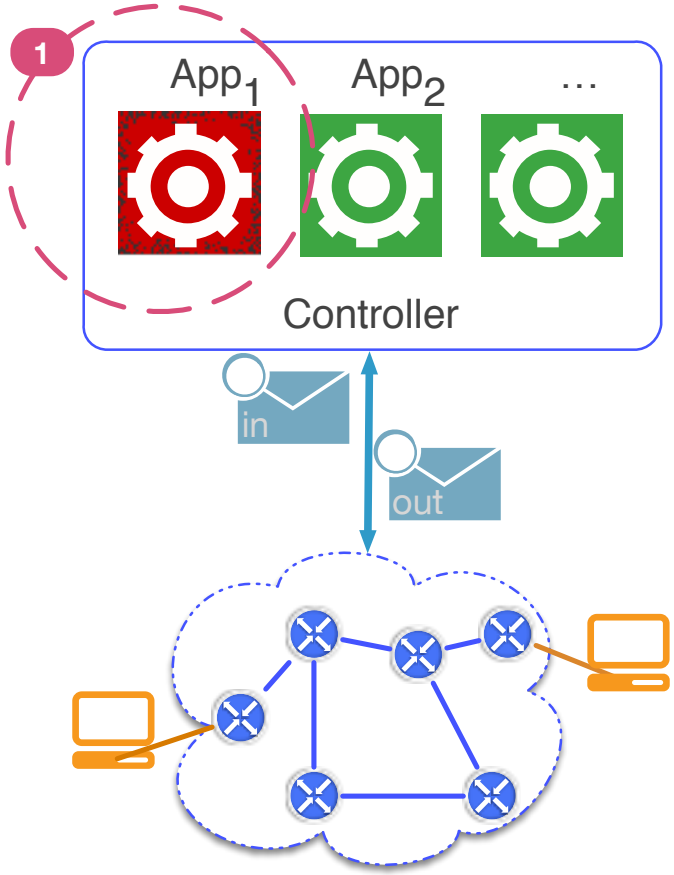
Fate-sharing

- Fate-sharing relationships between
  - the SDN controller and the SDN application(s)
    (also between SDN applications)
  - the SDN application and the network

- *Failure in any one SDN application brings down the other applications, and the SDN controller.*

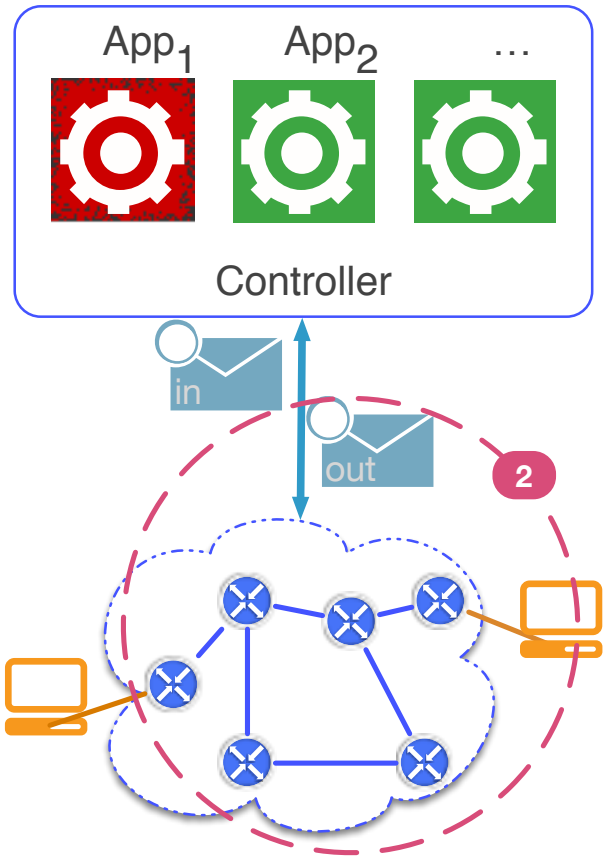
Three-pronged approach

Contain crash



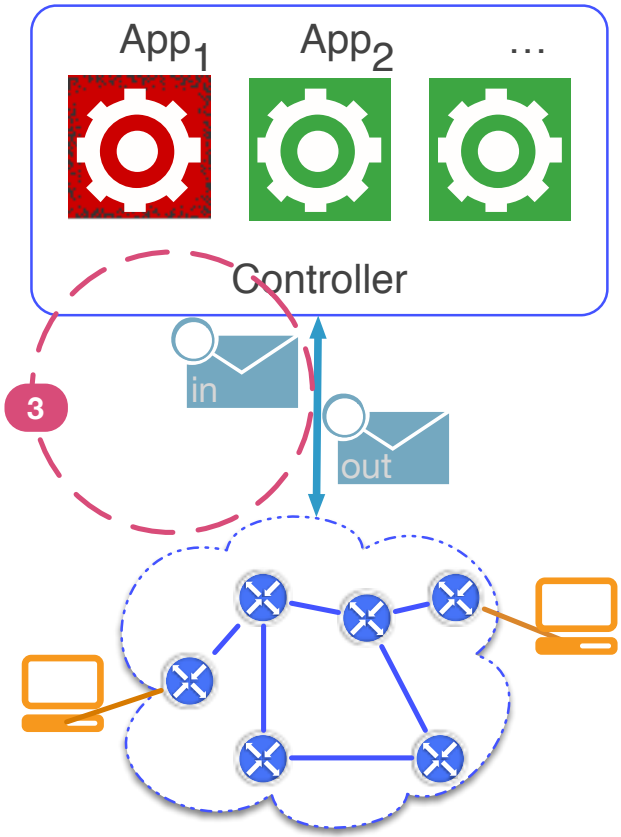
Three-pronged approach

Undo changes



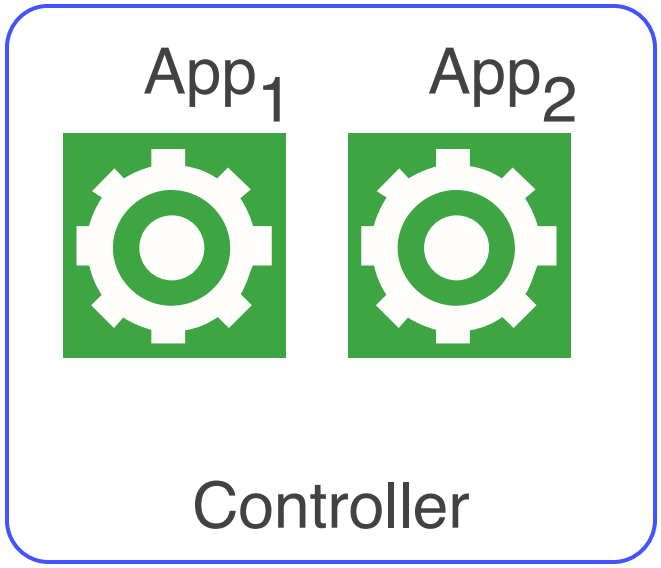
Three-pronged approach

Handle message

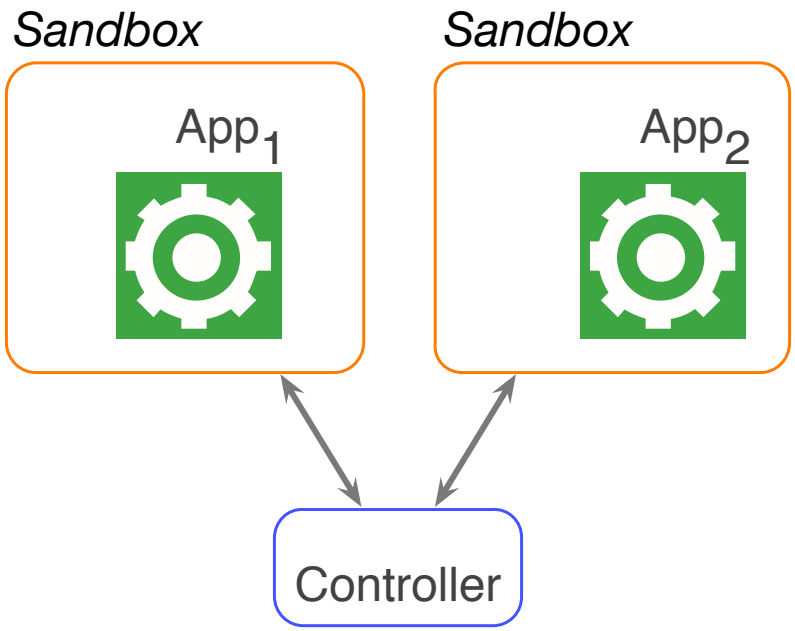


Controller architecture *must* support two new abstractions

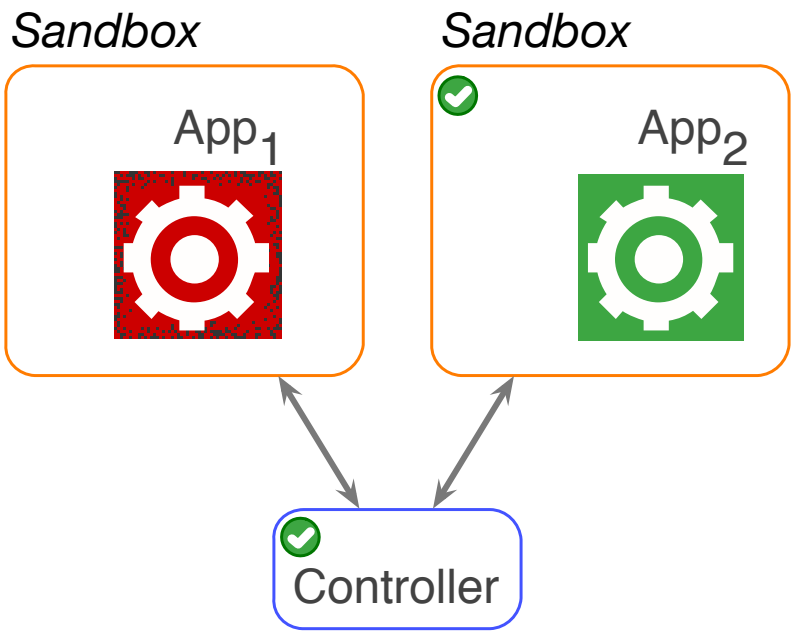
Current architecture



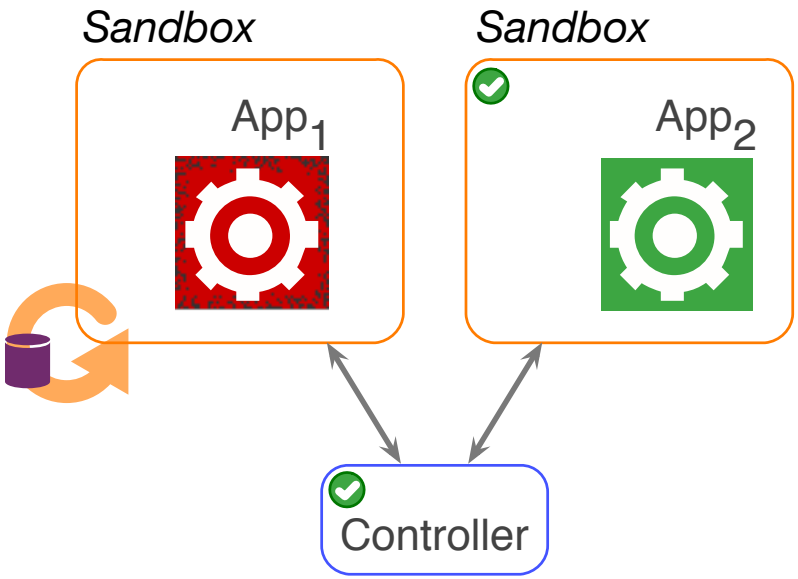
Isolate SDN-Apps from the controller



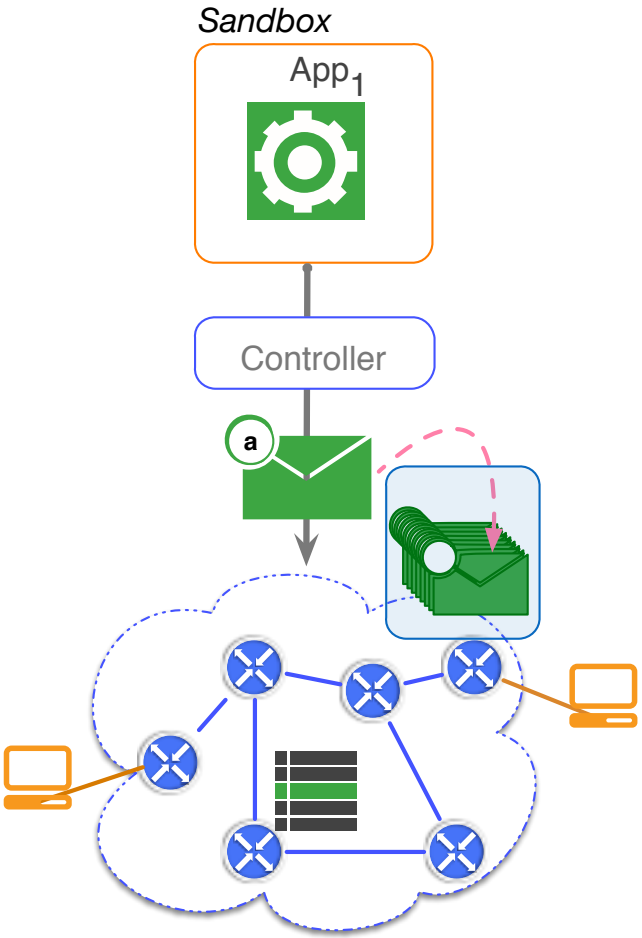
Isolate SDN-Apps from the controller



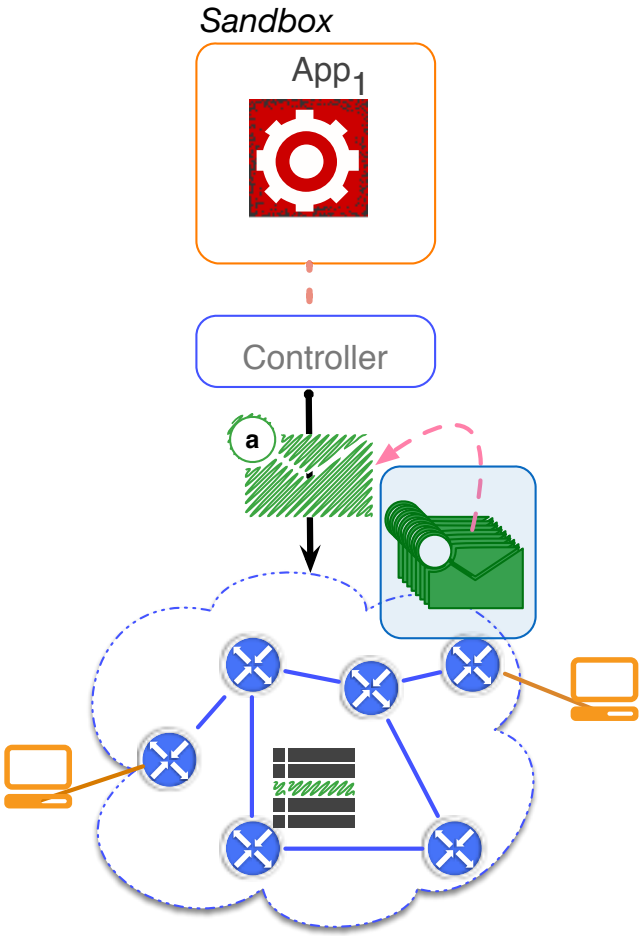
Isolate SDN-Apps from the controller



Isolate SDN-Apps from the network



Isolate SDN-Apps from the network



LegoSDN

AppVisor Stub

Lightweight wrapper

AppVisor Proxy

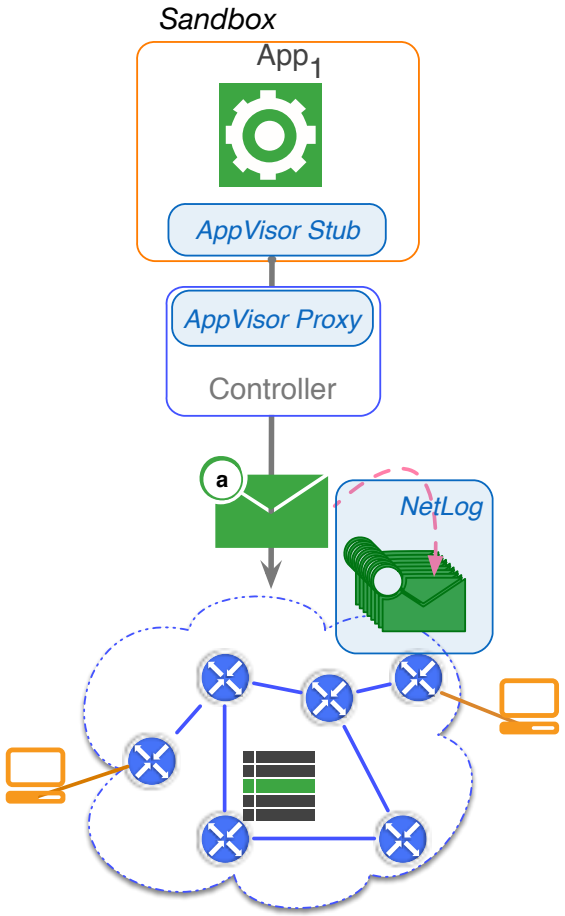
Message dispatcher

SDN-App is treated as a black-box.

Stub and proxy allow SDN-Apps to talk to controller.

NetLog

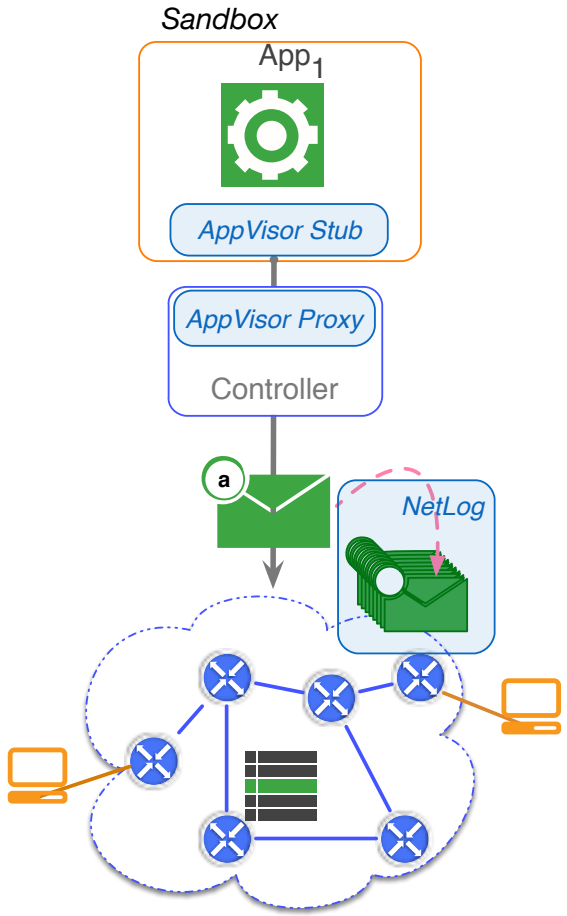
Transactional support



LegoSDN

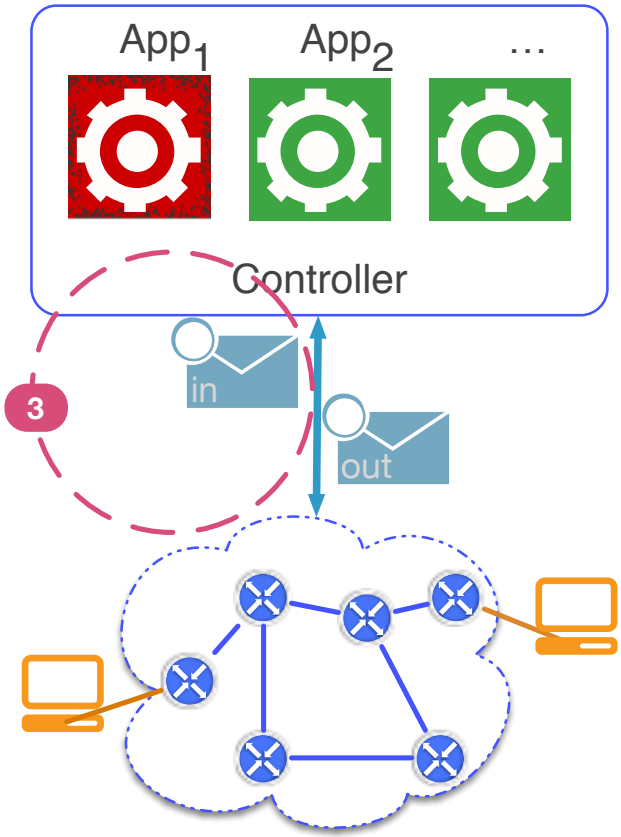
Built on top of FloodLight

Ported three applications bundled with FloodLight to LegoSDN

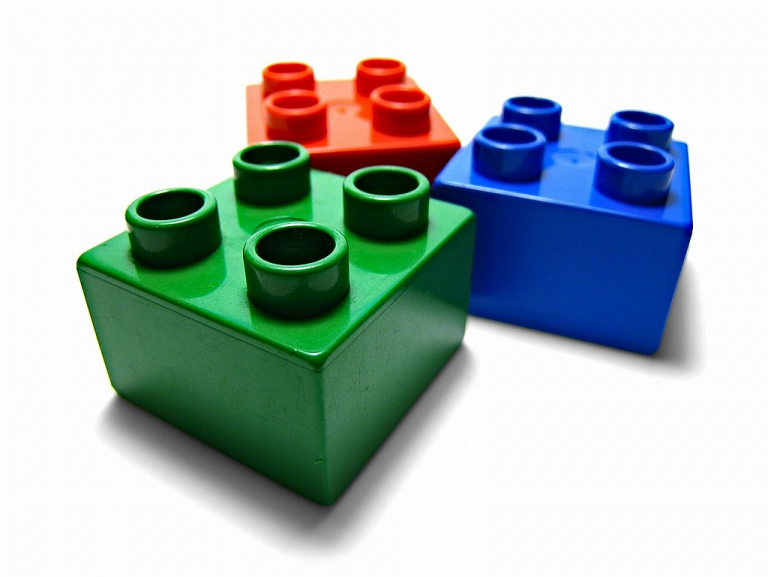


Three-pronged approach

Handle message



How do you handle the crash inducing message?



1. Crash and burn

- Halt the application
  - SDN-App cannot continue processing
  - *Other SDN-Apps can continue unaffected*

- *No Compromise*
  - *Think of security related SDN-Apps*

Correctness:

SDN-App's ability to implement its functionality without change, according to the specification.

2. Induce amnesia

- Ignore or drop the crash inducing message
  - SDN-App will not see the message again

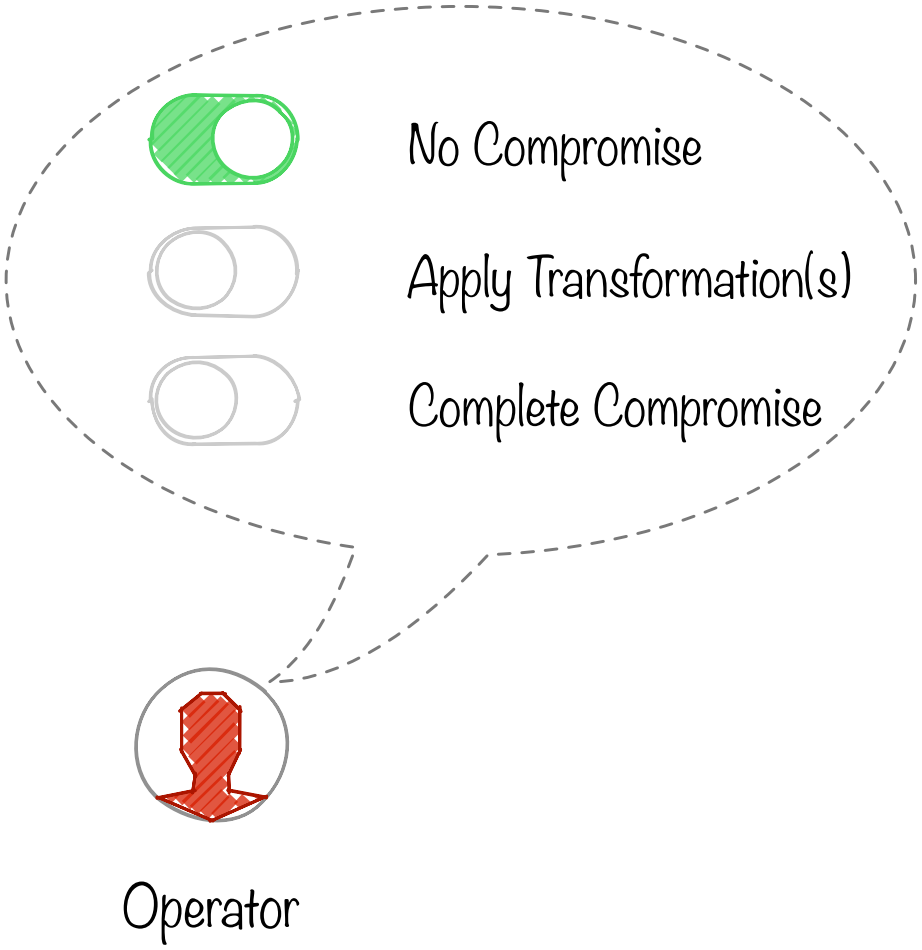
- *Complete Compromise*

3. Apply transformations

- Transform the offending message into another one that the application can handle
  - application will continue with a modified input

- *Equivalence Compromise*

Course of action?



Related work

- Fault tolerance
  - via reboots
  - applying Paxos for leader selection
- Debugging SDN-Apps or the controller

Message equivalence

- How do you determine two messages are equivalent?

Rollbacks are non-trivial

- Rollback of one or more rules installed changes controller's view of the state of network
  - Might induce crashes of other SDN applications that rely on a consistent view of network state

Error propagation

- Last message received by the SDN-App prior to the crash need not be the culprit!
  - How far along should we go back in history to find the root cause of the crash?
  - Recovery from an earlier checkpoint; How many checkpoints should we maintain?

Road ahead

- Rethink controller architecture
  - LegoSDN is only the tip of the iceberg.

- Resilient controllers can catalyze adoption
- Failures need to be a *first-class* citizen


99 little bugs in the code.
99 little bugs in the code.
Take one down, patch it around.
127 little bugs in the code...