OFRewind

Enabling Record and Replay Debugging in Networks

SPONSORED BY THE
Federal Ministry of Education and Research

Andi Wundsam, Dan Levin, Srini Seetharaman, Anja Feldmann
TU Berlin / Deutsche Telekom Labs

Debugging Networks can be very HARD

Challenges in Network Debugging:

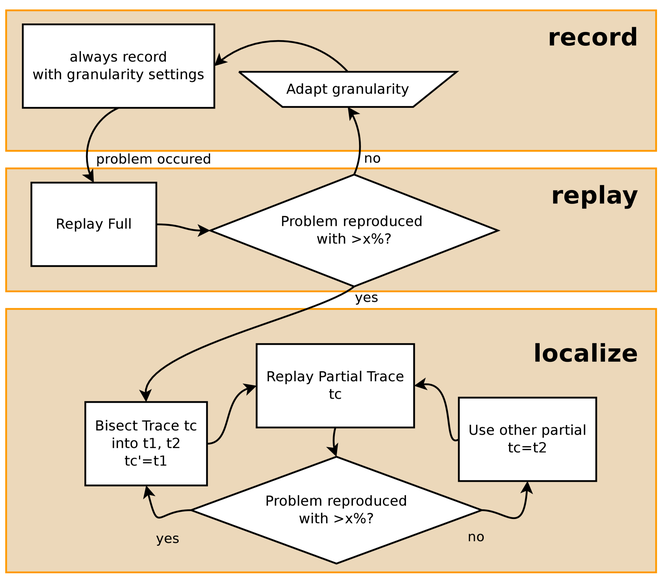
Networks are large and distributed

Many Black-Box Components
→ Poorly instrumentable

Current Debugging Tools:

Aggregated Statistics: SNMP
Sampled Data: Netflow
Local Measurements: tcpdump

What about Replay Debugging?



Our Proposal: OFRewind Network Record and Replay

Enabled by *Split Forwarding Architecture*
Implemented on OpenFlow

Select to Record High-Value, Low-Volume Flows (e.g. Routing Updates)

Always-On Recording of OpenFlow Control-Plane
Dynamic, Flexible Partial Recording of Data-Plane

After Fault Occurance, Sub-select Recorded Control- and Data-Plane Traffic for Replay

Centrally Orchestrate Both Recording and Replay from OpenFlow Controller

System Architecture

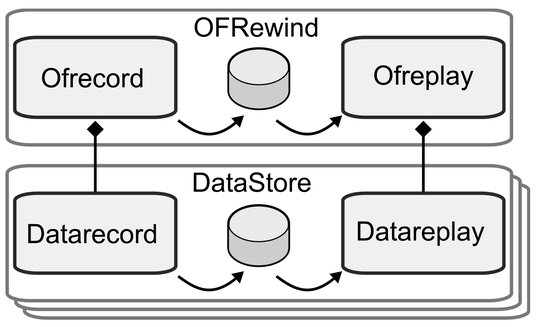
Two Primary System Components:
OFRewind + DataStore

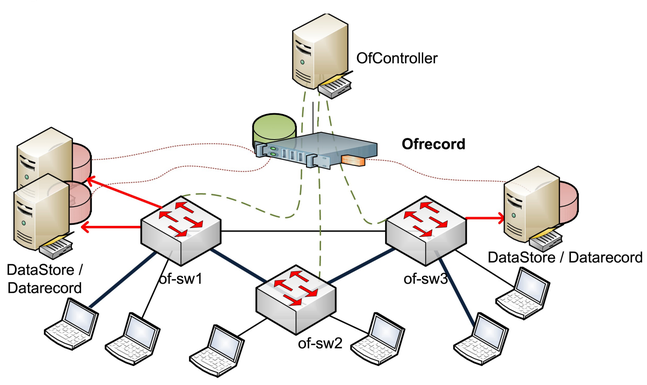
→ OFRewind acts as proxy between Controller and switches for control-plane recording/replay

→ Orchestrates multiple Datastores for data-plane recording/replay

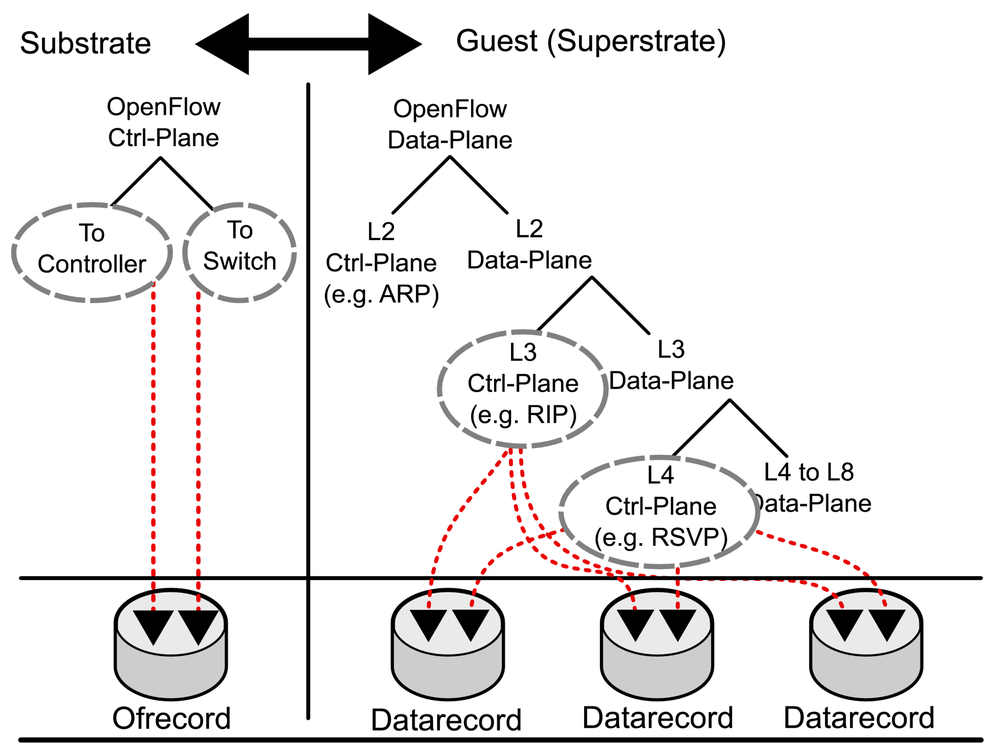
→ Maintains global ordering of all flows observed in network

→ Allows precise time-control over replay pace, ensuring flow ordering during replay is preserved



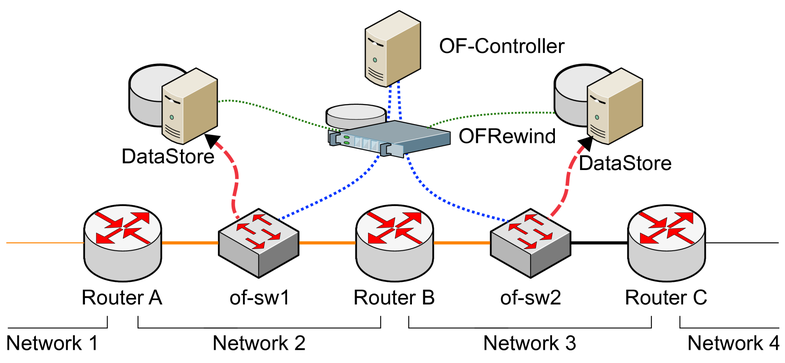


Example: Recording Selection



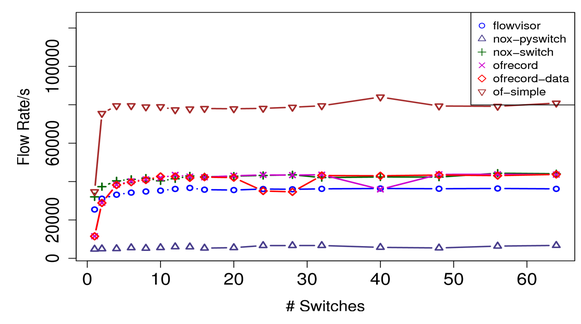
Case Studies

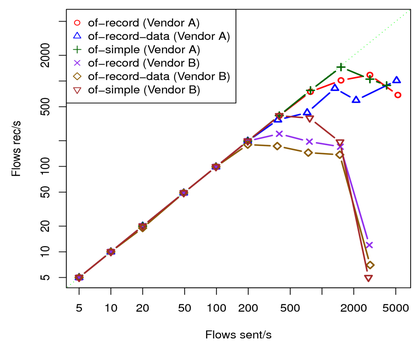
Faulty RIP Daemon (Quagga bug #235)
Ofrecord + Datarecord enable recording of specific RIP message sequence leading to pathological routing state machine error

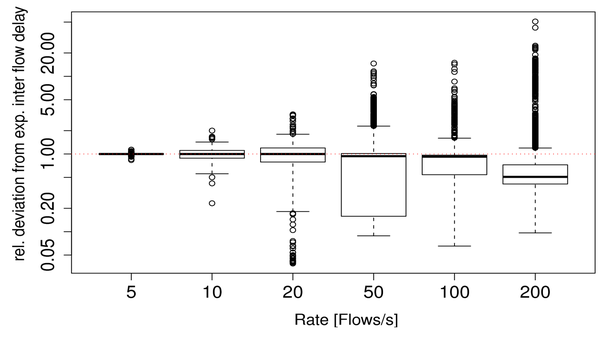


1) Observed Fault: Router C loses connectivity to Network 1. 2) Ofrecord has captured the control-plane view of RIP update flows 3) Inspection of global RIP flow ordering shows that at time of observed fault, RIP updates arriving at B do not propagate to C. 4) Playback of RIP updates onto identically configured lab environment reproduces this error 5) Continued replay of trigger event onto Router B with host-level process-debugger reveals code-level fault, responsible for failure to propagate RIP updates.

Performance Evaluation







Related work references and further information available from our full paper currently under USENIX ATC 2011 submission