DEFINED:
Deterministic Execution for
Interactive Control-Plane Debugging

Chia-Chi Lin, Virajith Jalaparti, and
Matthew Caesar

University of Illinois at Urbana-Champaign

Jacobus Van der Merwe

University of Utah
Control-Plane Software

• Participates in routing protocols to draw a network map

• Responsible for 95-99% of the observed bugs in today’s networks
  (Altekar et al. Focus Replay Debugging Effort on the Control Plane. HotDep ’10.)
Automatic Control-Plane Debugging

• Builds models of control-plane software to check for bugs and defects

• Detects anomalies but does not correct them

• Eventually, requires developers to understand and fix the bugs
Today’s Solution: Interactive Debugging with Logging

• Records nondeterministic events to enable deterministic replay
• Two varieties:
  – Comprehensive logging
    • Records everything
    • Able to reproduce everything
    • Doesn’t scale to today’s production networks
  – Partial logging
    • Records partial information
    • Scales to large-scale networks
    • Unable to precisely reproduce execution
DEFINED Goals

• Reproducibility
  – Precisely preserve execution without comprehensive logging

• Efficiency
  – Maintain fast convergence time in production networks

• Usability
  – Enable interactive control for debugging

• Scalability
  – Support enterprise and campus networks
Interactive Debugging with Deterministic Execution

• Nondeterministic events in control-plane software
  – External events
    • E.g., routers or links go down
  – Internal events
    • E.g., routers exchange messages
      (Bergan et al. Deterministic Process Groups in dOS. OSDI ’10.)
• Logs only external events
•Eliminates nondeterminism from internal events
DEFINED Overview

- A library
- Records external events
- Intercepts internal message events
- Provides deterministic timer APIs

Network Node

Control-Plane Software

DEFINeD

Operating System
DEFINED Algorithms

- DEFINED-RB for production networks
  - Designed for efficiency
  - Implements speculative execution with RollBacks
- DEFINED-LS for debugging networks
  - Designed for interactive control
  - Steps through network execution with a LockStep algorithm
Outline

• DEFINED-RB in Production Networks
• DEFINED-LS in Debugging Networks
• Evaluation
• Conclusion
• Each network node independently determines an ordering function to order internal events

• If events execute in the “wrong” order, DEFINED-RB rolls back the state of the network node and replays events in the “correct” order
Rolling Back Software States

Shared Memory

Control-Plane Software

Control-Plane Software

Control-Plane Software

DEFINED-RB
Ordering Internal Events with Logical Timestamps

• One network node periodically broadcast logical timestamps
• Each node records external events in logical time
• Each node tags and orders internal messages with logical timestamps and fires timers in logical time
Cascading Rollbacks Within a Logical Time Unit

Order □ before □
Optimized Ordering Function with Latency Information
Stepping through Debugging Networks

• DEFINED-LS divides network execution into logical steps

• Each step has two phases
  – Transmission phase
    • Each network node sends messages to neighboring nodes
  – Processing phase
    • Each network node processes its internal events
A Step in DEFINED-LS

Processing Phase

Control-Plane Software

DEFINED-LS
Centralized Coordinator for Interactive Stepping

- Coordinates phase transitions among network nodes
- Allows developers to issue a “step” command
- Steps may be chosen at various levels of granularity (per-event or per-logical-time-unit)
Evaluation Methodology

- Software: XORP OSPF 1.6
- Environment: Emulab
- Topology: Rocketfuel and BRITE  
  (*we present results from the Rocketfuel Sprintlink topology*)
- Traces: 2 weeks of Tier-1 ISP area 0 OSPF traces  
  (*324 network nodes and 651 events*)
99% of the updates are comparable to XORP.
Every step command completes in less than 1 second
Optimized Ordering reduces the convergence time by 1.35 seconds.

Optimized Ordering adds only 0.73 seconds to the convergence time.
A step command completes in less than 0.8 seconds on average.
Conclusion

• A debugger for control-plane software
• Uses deterministic execution to avoid logging internal nondeterminism
• Implements speculative execution to maintain efficiency in production networks
• Leverages a lockstep algorithm to provide interactive control in debugging networks
THANK YOU