Leveraging SDN Layering to Systematically Troubleshoot Networks

Brandon Heller★
Colin Scott◆
Nick McKeown✉
Scott Shenker ◆☯
Andreas Wundsam §
Hongyi Zeng✉
Sam Whitlock◆
Vimalkumar Jeyakumar✉
Nikhil Handigol★
James McCauley◆☯
Kyriakos Zarifis∞
Peyman Kazemian★

◆Stanford
◆Berkeley
∞USC
☺ICSI
★SDN Academy
§Big Switch Networks

HotSDN 2013
Hong Kong
This Talk

2: Troubleshoot

Admin

skills + tools

3: Fix Stuff!

1: Configure

Policy

• connect hosts A + B
• quarantine virus-infected hosts
• route guest traffic to an HTTP proxy
• prioritize SSH

Network

Protocols

Configuration

Topology

Ethane, overlays, consistency primitives, network programming languages, ...
This Talk

Policy

Admin

Network

#1 request from network admins: Automatic Troubleshooting


1: Configure

Ethane, overlays, consistency primitives, network programming languages, ...

2: Troubleshoot
How to automate troubleshooting?

Challenging in traditional networks.

Two requirements.

(1) Know the intended policy: impractical to infer
- confusing: different config format for each protocol
- distributed: configuration spread among all nodes
- hard: must understand all protocols & their interactions

(2) Check behavior against policy: difficult to check
- confusing: don’t know lowest-level forwarding behavior
- distributed: hard to get a meaningful snapshot
Control-Plane Layering in SDN

State Layers
- Policy
- Logical View
- Physical View
- Device State
- Hardware

Code Layers
- Policy
- App
- Code Layers
- Network Hypervisor
- Network OS
- Firmware
- Hardware

State Layers
- Logical View
- Physical View
- Device State
- Hardware

Code Layers
- Policy
- App
- Code Layers
- Network Hypervisor
- Network OS
- Firmware
- Hardware
Systematically Troubleshooting an SDN

**State Layers**
- Policy
- Logical View
- Physical View
- Device State
- Hardware

**Code Layers**
- Policy
- Logical View
- Physical View
- Device State
- Hardware

**Observation:**
Each state layer fully specifies network behavior.

**Insight:**
Bugs manifest as mistranslations between layers.

**Systematic Approach:**
1. Binary search to isolate to a code layer.
2. Leverage state to isolate within the code layer.
Phase 1: Localizing to a code layer
Symptom: Hosts unable to communicate

[Operator Intent]

Policy
Apps
Logical View
NetHyperV
Physical View
NetOS
Device State
Firmware
Hardware
[Actual Behavior]

Cause:
Firmware Bug
Phase 1: Localizing to a code layer

Symptom: Tenant Isolation Breach

[Operator Intent]

Policy

Apps

Logical View

NetHyperV

Physical View

NetOS

Device State

Firmware

Hardware

[Actual Behavior]

Correspondence Checking

HSA [NSDI ’12]

OFRewind [ATC ’11]

Cause:

NetHypervisor Bug
How to automate troubleshooting?

Possible in Software-Defined Networks

Policy

Two requirements.

(1) Know the intended policy: directly provided
   • confusing: different config format for each protocol
   • distributed: configuration spread among all nodes
   • hard: must understand all protocols & their interactions

(2) Check behavior against policy: directly accessible
   • confusing: don’t know lowest level forwarding behavior
   • distributed: hard to get a meaningful snapshot
Takeways

• Control plane layering enables systematic troubleshooting

• Thinking about troubleshooting in terms of layers shows us where tools fit in
  – Reveals missing tools
  – Highlights choices between tools, with tradeoffs

• Plenty of opportunities left.

Operationalize!
Leverage the layers in SDN.

Questions?

HotSDN 2013
Hong Kong
How is this different than general distributed systems debugging?

• Simple answer: it’s not! SDN is an excellent opportunity to draw upon ideas from other distributed systems

• Subtlety: networks are solving a much more constrained problem than general distributed systems
Limitations

• Correctness only, not performance
• Side effects not reflected in state
• No guarantee of finding single code layer
• No guarantee of individual layer correctness
• No guarantee of future correctness
• Layer visibility may be imperfect
Plenty of Opportunities Remain

• Automatic Troubleshooting → Actionable Bug Reports
  – Filtering the signal from the noise
  – Creating consistent views of state

• Improving Invariant Checkers
  – Scale
  – Flexible Policy Input

• Hybrid Traditional + SDN Debugging
Plenty of Opportunities Remain

- Automatic Troubleshooting → Actionable Bug Reports
  - Filtering the signal from the noise
  - Creating consistent views of state

Packet History:
Path + Headers + Forwarding State

[HotSDN 2012: Where is the Debugger for My Software-Defined Network?]
Plenty of Opportunities Remain

• Automatic Troubleshooting \( \rightarrow \) Actionable Bug Reports
  – Filtering the signal from the noise

[Berkeley Tech Report: How Did We Get Into This Mess? Isolating Fault-Inducing Inputs to SDN Control Software]
Isn’t this unnecessary with consistency primitives/languages/etc?

• No
• Catch/rule out bugs outside the framework
• Catch instances where the framework pushes config that breaks the policy
What’s novel about this work?

• Simple answer: nothing!
Control-Plane Layering in SDN

**State Layers**
- Policy
- Logical View
- Physical View
- Device State
- Hardware

**Code Layers**
- Network Hypervisor
- Network OS
- Firmware
- HW
- App

**Example Errors**
- [Unintended Config]
- Configuration Parsing Error
- Tenant isolation breach (policy mistranslation)
- Failover logic error, synchronization bug
- Register misconfiguration, Router memory corruption
- [External Connectivity Error]