Online Data Plane Checking

June 12, 2013

Summer School on Formal Methods and Networks
Cornell University
VeriFlow: Verifying Network-Wide Invariants in Real Time*

Ahmed Khurshid, Xuan Zou, Wenxuan Zhou, Matthew Caesar, P. Brighten Godfrey
University of Illinois at Urbana-Champaign (UIUC)

June 12, 2013

Summer School on Formal Methods and Networks
Cornell University

*HotSDN 2012, NSDI 2013, ONS 2013
Challenges in Network Debugging

- Complex interactions
- Misconfigurations
- Unforeseen bugs
- Difficult to test the entire network state space before deployment

http://groups.geni.net/geni/chrome/site/thumbnails/wiki/TangoGENI/OF-VLAN3715_1000.jpg
Data Plane Verification in Action

- **FlowChecker** [Al-Shaer et al., SafeConfig 2010]
  - Uses BDD-based model checker
- **Anteater** [Mai et al., SIGCOMM 2011]
  - Uses SAT-based model checking
  - Revealed 23 real bugs in the UIUC campus network
- **Header Space Analysis** [Kazemian et al., NSDI 2012]
  - Uses set-based custom algorithm
  - Found multiple loops in the Stanford backbone network

**Running time:** Several seconds to a few hours
Can we run verification in real time?

Checking network-wide invariants in real time as the network evolves

Need to verify new updates at high speeds

Block dangerous changes

Provide immediate warning
Challenges in Real-Time Verification

• **Challenge 1**: Obtaining real-time view of network
  – Solution: Utilize the *centralized* data-plane view available in an **SDN (Software-Defined Network)**

• **Challenge 2**: Verification speed
  – Solution: Off-the-shelf techniques?
    
    No, too slow!
Our Tool: VeriFlow

• VeriFlow checks network-wide invariants in **real time** using data-plane state
  – Absence of routing loops and black holes, access control violations, etc.

• VeriFlow functions by
  – Monitoring **dynamic changes** in the network
  – Constructing a **model** of the **network behavior**
  – Using **custom algorithms** to automatically derive whether the network contains errors
VeriFlow Operation

Network Controller

New rules

VeriFlow

Generate equivalence classes

Generate forwarding graphs

Run queries

Good rules

Rules violating network invariant(s)

Diagnosis report
- Type of invariant violation
- Affected set of packets
1. Limit the Search Space

**VeriFlow**

- Generate Equivalence Classes
- **Equivalence class:** Packets experiencing the same forwarding actions throughout the network.

Fwd’ing rules:

- 0.0.0.0/1
- 64.0.0.0/3
- 0.0.0.0/0

Equiv. classes:

- 1
- 2
- 3
- 4
Computing Equivalence Classes

(don’t care/wildcard)

(device, rule) pairs

Equivalence classes

Header value ranges
2. Represent Forwarding Behavior

VeriFlow

- Generate Forwarding Graphs
- Generate Equivalence Classes

Updates

Equivalence Class 1

Equivalence Class 2

All the info to answer queries!
3. Run Query to Check Invariants

VeriFlow

Generate Equivalence Classes → Generate Forwarding Graphs → Run Queries

- Updates
- Black holes, Routing loops, Access control policies
- Good rules
- Bad rules

Diagnosis report
- Type of invariant violation
- Affected set of packets
API to write custom invariants

• VeriFlow provides a set of functions to write custom query algorithms
  – Gives access to the affected set of equivalence classes and their forwarding graphs
  – Verification becomes a standard graph traversal algorithm

• Can be used to
  – Check forwarding behavior of specific packet sets
  – Verify effects of potential changes
Experiment

• Simulated an IP network using a Rocketfuel topology
  – 172 routers

• Replayed Route Views BGP traces
  – 5 million RIB entries
  – 90K BGP updates

• Checked for loops and black holes

• Microbenchmarked each phase of VeriFlow’s operation
97.8% of the updates were verified within 1 millisecond.
Effect of Equivalence Class Count

Number of ECs strongly influences verification time.

Number of ECs affected by new rule vs. Average verification time (ms)
Experiment (cont.)

- Mininet OpenFlow network
  - Rocketfuel topology with 172 switches, one host per switch
- NOX controller, learning switch application
- TCP connections between random pairs of hosts
Effect on Flow Table Update Throughput

Overhead of VeriFlow is low

Update throughput (msg/sec)

TCP connection attempts (per sec)
Effect of Multiple Header Fields

- Data link source
- Data link destination
- Network source
- Network destination
- Data link type

More fields -> More equivalence classes -> Longer verification time
Conclusion

• VeriFlow achieves real-time verification
  – A layer between SDN controller and network devices
  – Handles multiple packet header fields efficiently
  – Runs queries within hundreds of microseconds
  – Exposes an API for writing custom invariants

• Ongoing work
  – Handling packet transformations efficiently
  – Dealing with multiple controllers
Demo Network
Forwarding Graphs from the Rocketfuel-RouteViews Experiment
VeriFlow source code is available at

http://www.cs.illinois.edu/~khurshi1/projects/veriflow/
Thank you

khurshi1@illinois.edu
http://www.cs.illinois.edu/~khurshi1
Backup Slides
Related Work

• Real time network policy checking using header space analysis, NSDI 2013
• Header space analysis: Static checking for networks, NSDI 2012
• A NICE way to test OpenFlow applications, NSDI 2012
• Abstractions for network update, SIGCOMM 2012
• Can the production network be the testbed?, OSDI 2010
• FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures, SafeConfig 2010
• Network configuration in a box: Towards end-to-end verification of network reachability and security, ICNP 2009
• On static reachability analysis of IP networks, INFOCOM 2005