VeriFlow: Verifying Network-Wide Invariants in Real Time

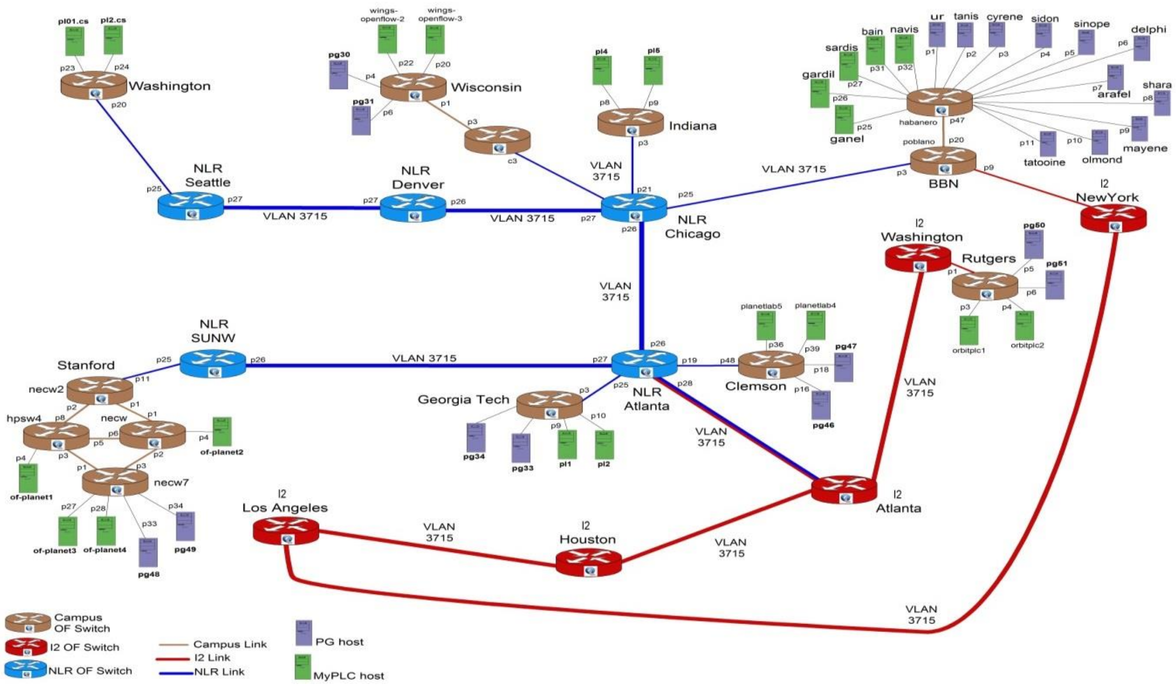
Ahmed Khurshid, Xuan Zou, Wenxuan Zhou, Matthew Caesar, P. Brighten Godfrey
University of Illinois at Urbana-Champaign (UIUC)

April 16, 2013

ONS 2013

Open Networking Summit

Challenges in Network Debugging

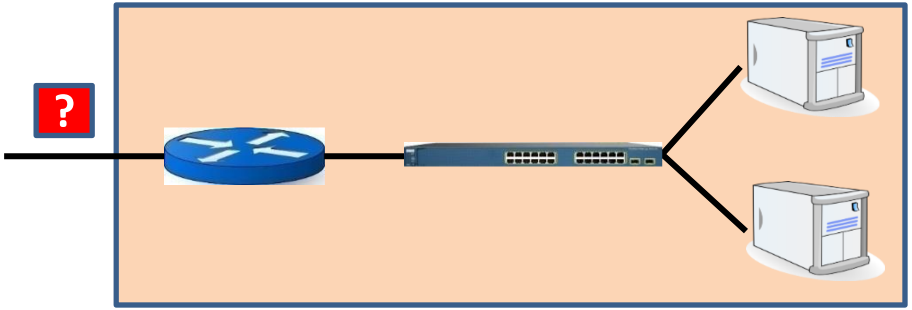


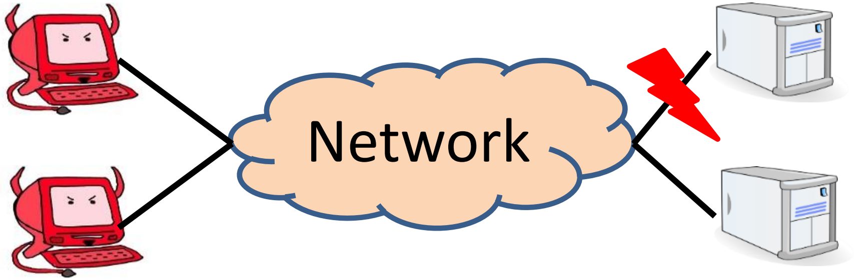
http://groups.geni.net/geni/chrome/site/thumbnails/wiki/TangoGENI/OF-VLAN3715_1000.jpg

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

Effects of Network Errors

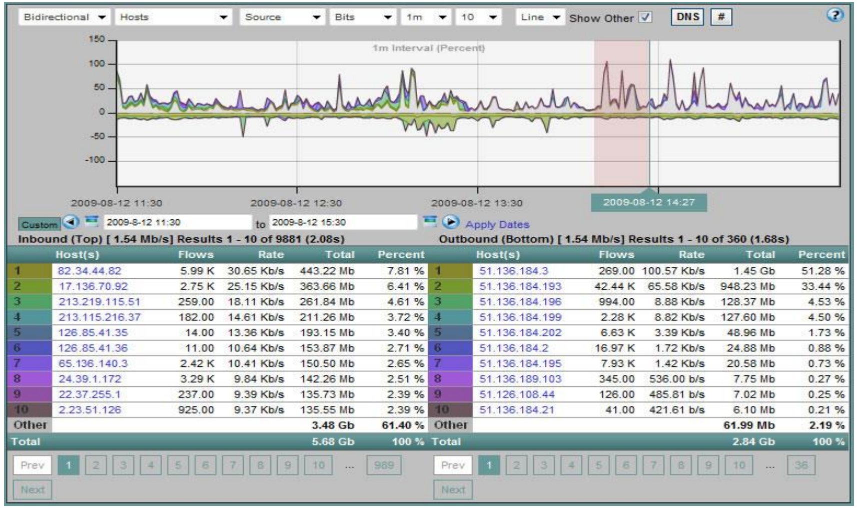
- Allow unauthorized packets to enter a secured zone in a network
- Make services and the infrastructure prone to attacks
- Make critical services unavailable
- Affect network performance





Network Debugging Techniques

Traffic/Flow Monitoring



Inbound (Top) [1.54 Mb/s] Results 1 - 10 of 9881 (2.08s)

	Host(s)	Flows	Rate	Total	Percent
1	82.34.44.82	5.99 K	30.65 Kb/s	443.22 Mb	7.81 %
2	17.136.70.92	2.75 K	25.15 Kb/s	363.66 Mb	6.41 %
3	213.219.115.51	259.00	18.11 Kb/s	261.84 Mb	4.61 %
4	213.115.216.37	182.00	14.61 Kb/s	211.26 Mb	3.72 %
5	126.85.41.35	14.00	13.36 Kb/s	193.15 Mb	3.40 %
6	126.85.41.36	11.00	10.64 Kb/s	153.87 Mb	2.71 %
7	65.136.140.3	2.42 K	10.41 Kb/s	150.50 Mb	2.65 %
8	24.39.1.172	3.29 K	9.84 Kb/s	142.26 Mb	2.51 %
9	22.37.255.1	237.00	9.39 Kb/s	135.73 Mb	2.39 %
10	2.23.51.126	925.00	9.37 Kb/s	135.55 Mb	2.39 %
Other				3.48 Gb	61.40 %
Total				5.68 Gb	100 %

Outbound (Bottom) [1.54 Mb/s] Results 1 - 10 of 360 (1.68s)

	Host(s)	Flows	Rate	Total	Percent
1	51.136.184.3	269.00	100.57 Kb/s	1.45 Gb	51.28 %
2	51.136.184.193	42.44 K	65.58 Kb/s	948.23 Mb	33.44 %
3	51.136.184.196	994.00	8.88 Kb/s	128.37 Mb	4.53 %
4	51.136.184.199	2.28 K	8.82 Kb/s	127.60 Mb	4.50 %
5	51.136.184.202	6.63 K	3.39 Kb/s	48.96 Mb	1.73 %
6	51.136.184.2	16.97 K	1.72 Kb/s	24.88 Mb	0.88 %
7	51.136.184.195	7.93 K	1.42 Kb/s	20.58 Mb	0.73 %
8	51.136.189.103	345.00	536.00 b/s	7.75 Mb	0.27 %
9	51.126.108.44	126.00	485.81 b/s	7.02 Mb	0.25 %
10	51.136.184.21	41.00	421.61 b/s	6.10 Mb	0.21 %
Other				61.99 Mb	2.19 %
Total				2.84 Gb	100 %

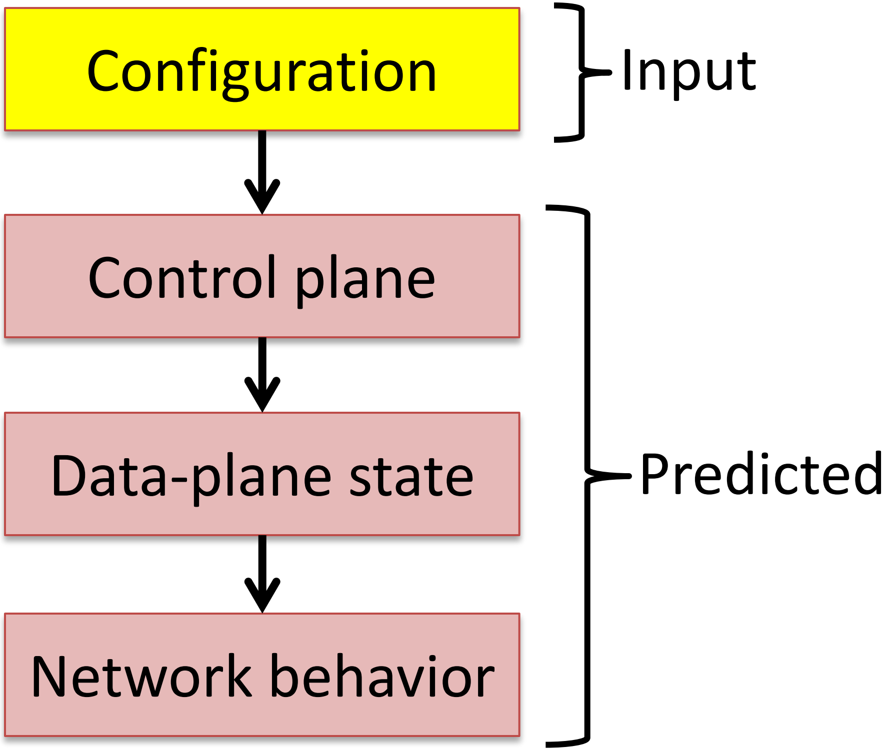
Software using Cisco NetFlow

http://snmp.co.uk/scrutinizer/

Configuration Verification

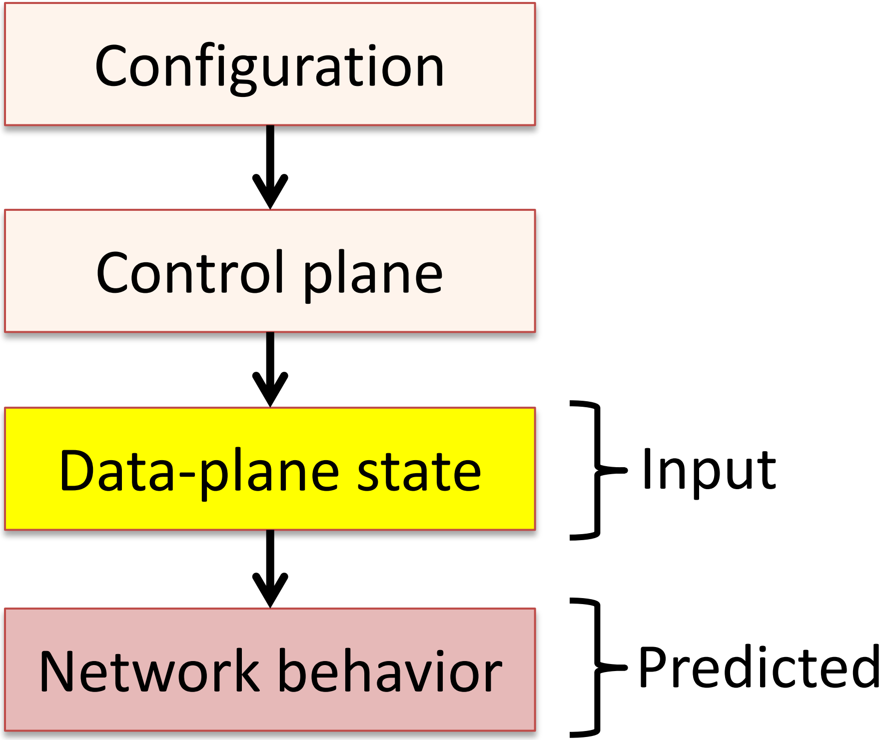
```
hostname bgpdA
password zebra
!
router bgp 8000
  bgp router-id 10.1.4.2

! for the link between A and B
  neighbor 10.1.2.3 remote-as 8000
  neighbor 10.1.2.3 update-source lo0

  network 10.0.0.0/7

! for the link between A and C
  neighbor 10.1.3.3 remote-as 7000
  neighbor 10.1.3.3 ebgp-multihop
  neighbor 10.1.3.3 next-hop-self
  neighbor 10.1.3.3 route-map PP out

! for link between A and D
  neighbor 10.1.4.3 remote-as 6000
  neighbor 10.1.4.3 ebgp-multihop
  neighbor 10.1.4.3 next-hop-self
  neighbor 10.1.4.3 route-map TagD in

! route update filtering
  ip community-list 1 permit 8000:1000
!
```

Limitations of Configuration Verification



- Prediction is difficult
  - Various configuration languages
  - Dynamic distributed protocols
- Prediction misses implementation bugs in control plane

Our Approach: Data-plane Verification



- Less prediction
- Closer to actual network behavior
- Unified analysis for multiple control-plane protocols
- Can catch control-plane implementation bugs

Data Plane Verification in Action

- Our first tool, Anteater*, uses data plane verification technique to debug network operations
- We evaluated Anteater with UIUC campus network
  - 178 routers
  - 1,627 FIB entries per router (mean)
- It revealed **23 real bugs** in 2 hours

Finds problems after they occur and (potentially) cause damage

* Haohui Mai, Ahmed Khurshid, Rachit Agarwal, Matthew Caesar, P. Brighten Godfrey, and Samuel T. King, "Debugging the Data Plane with Anteater", ACM SIGCOMM, August 2011.

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.
- Provides a set of functions to write custom query algorithms
  - Check forwarding behavior of specific packet sets
  - Verify effects of potential changes

VeriFlow Operation

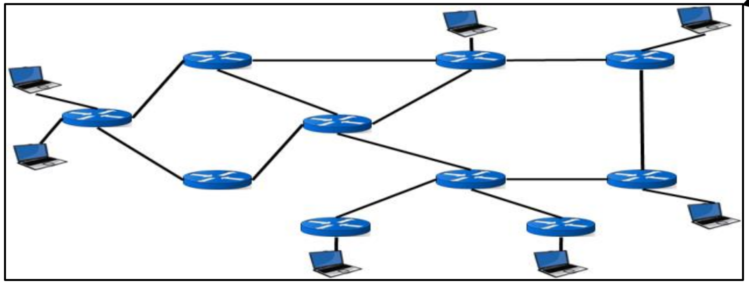
Network Controller

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

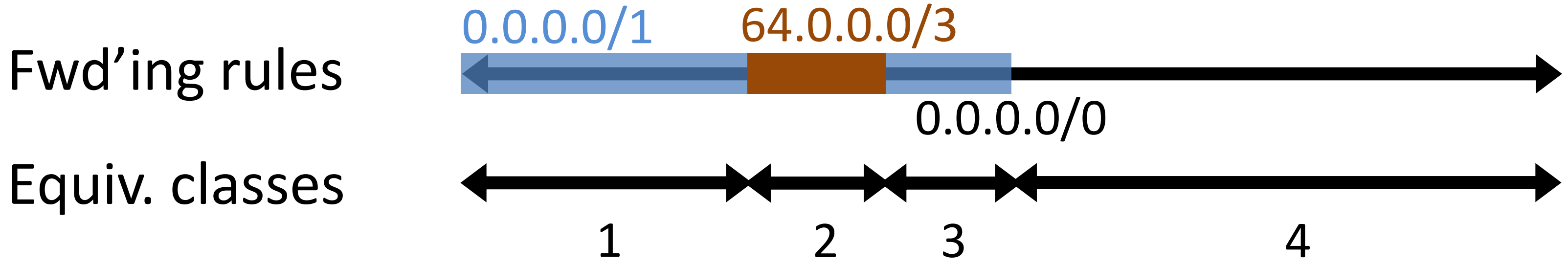
Limit the Search Space

VeriFlow

Generate Equivalence Classes

Updates

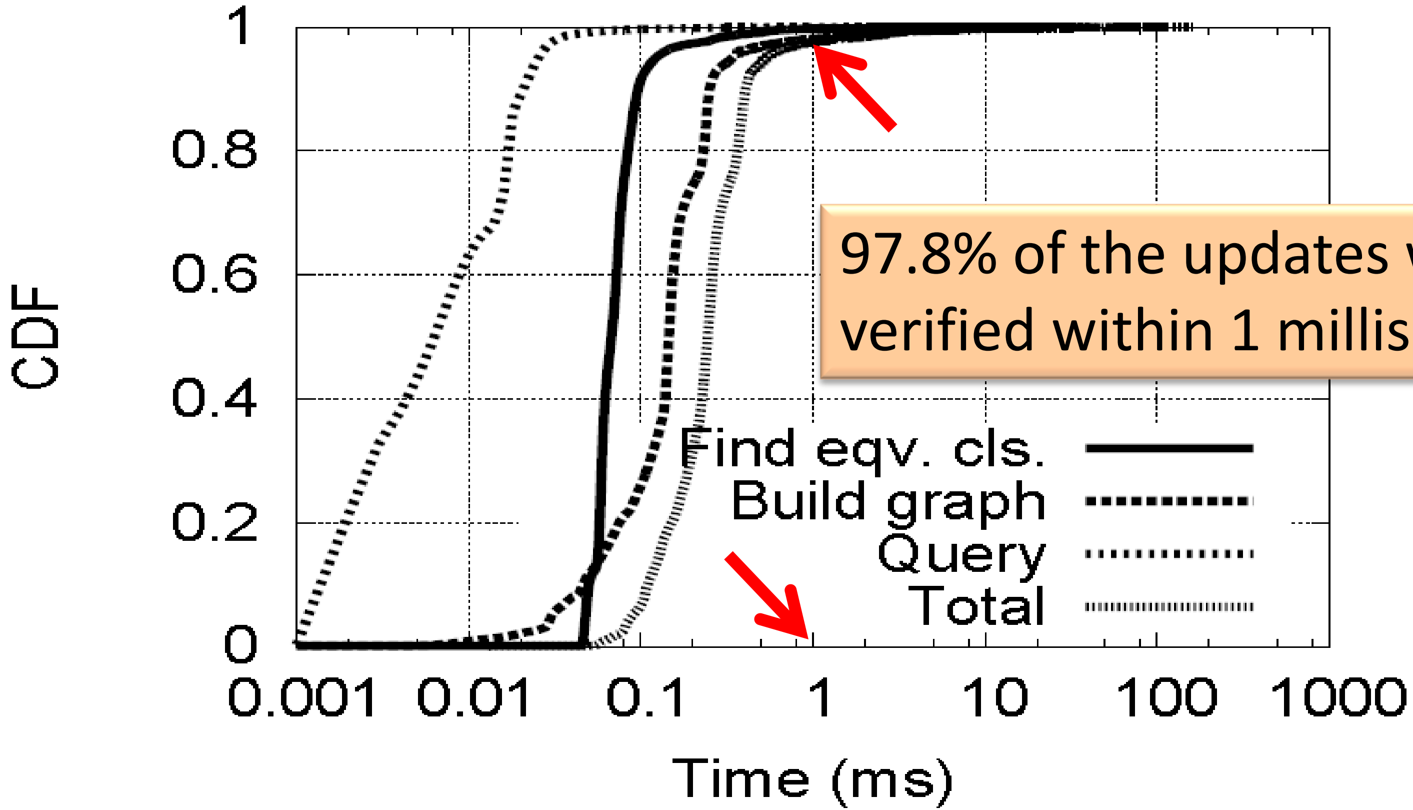
Equivalence class:
Packets experiencing the same forwarding actions throughout the network.



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

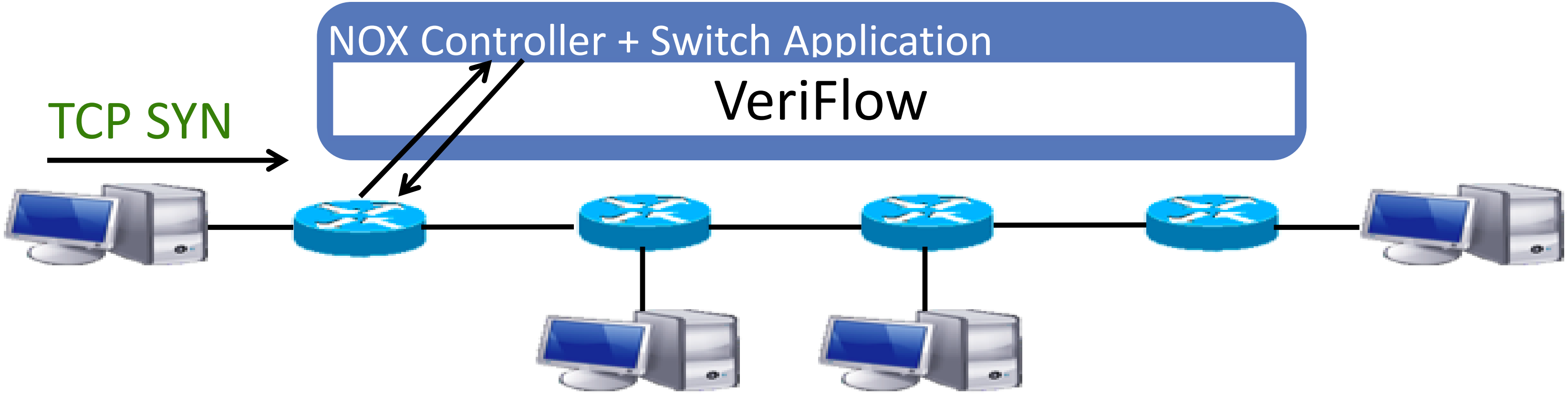
Performance Result



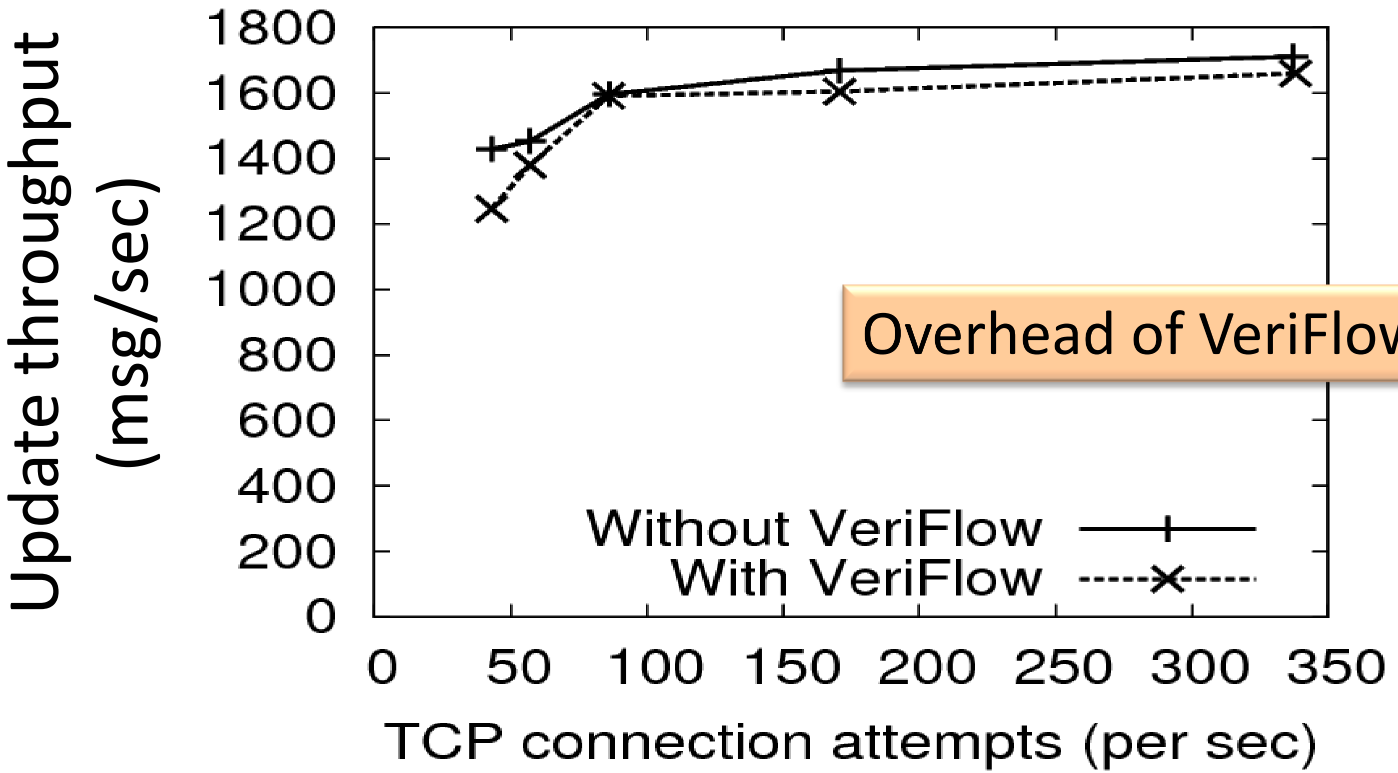
97.8% of the updates were verified within 1 millisecond

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

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
- Future work
  - Handling packet transformations efficiently
  - Dealing with multiple controllers

Thank you

khurshi1@illinois.edu

http://www.cs.illinois.edu/~khurshi1

Backup Slides

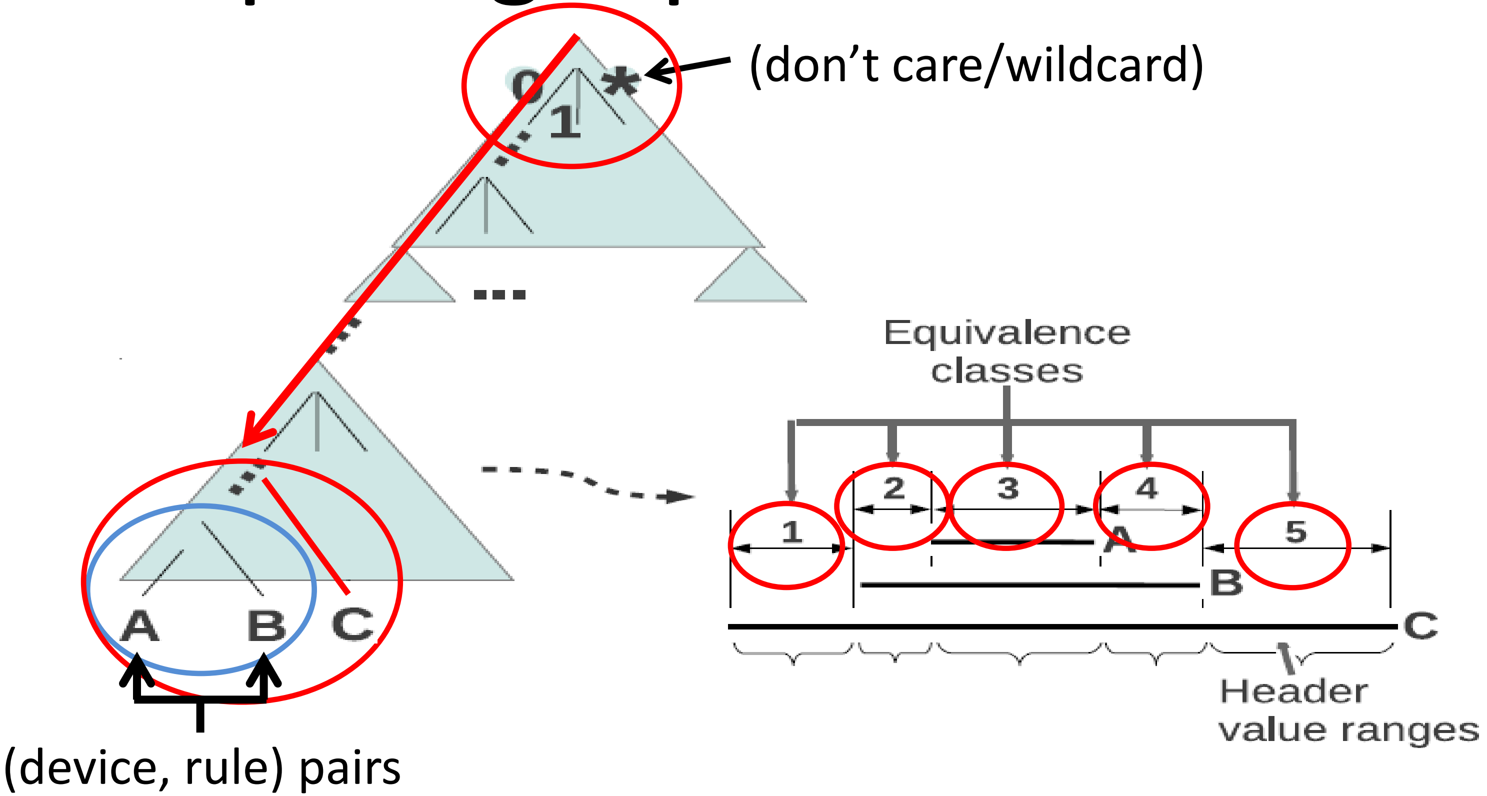
Data Plane Verification in Action

- 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

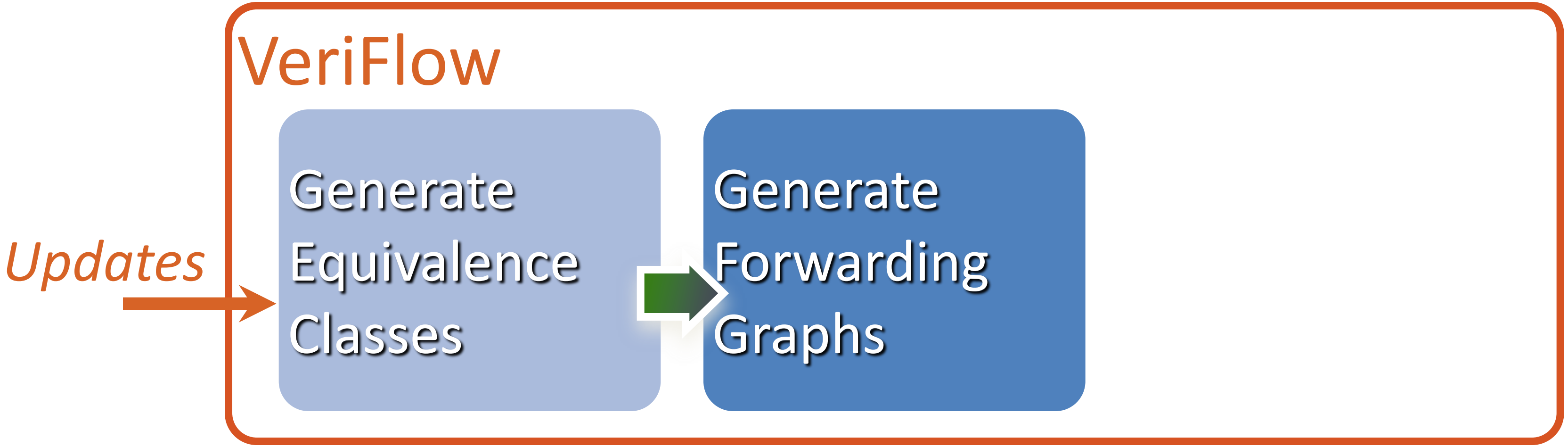
Find problems after they occur and (potentially) cause damage

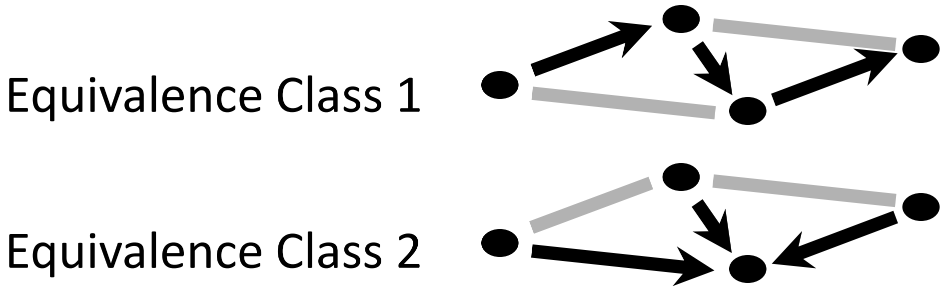
Running time: Several seconds to a few hours

Computing Equivalence Classes



2. Represent Forwarding Behavior





All the info to answer queries!

3. Run Query to Check Invariants

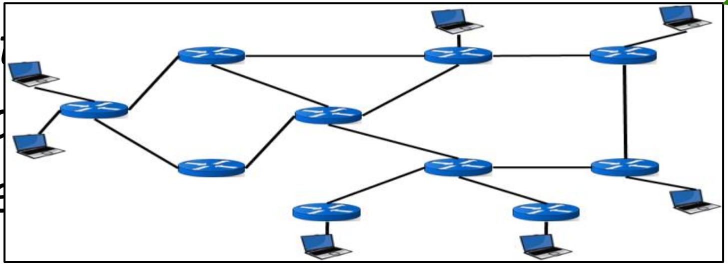
VeriFlow

Updates → Generate Equivalence Classes → Generate Forwarding Graphs → Run Queries

Good rules

Bad rules

Black holes,
Rout
Isola
Acce



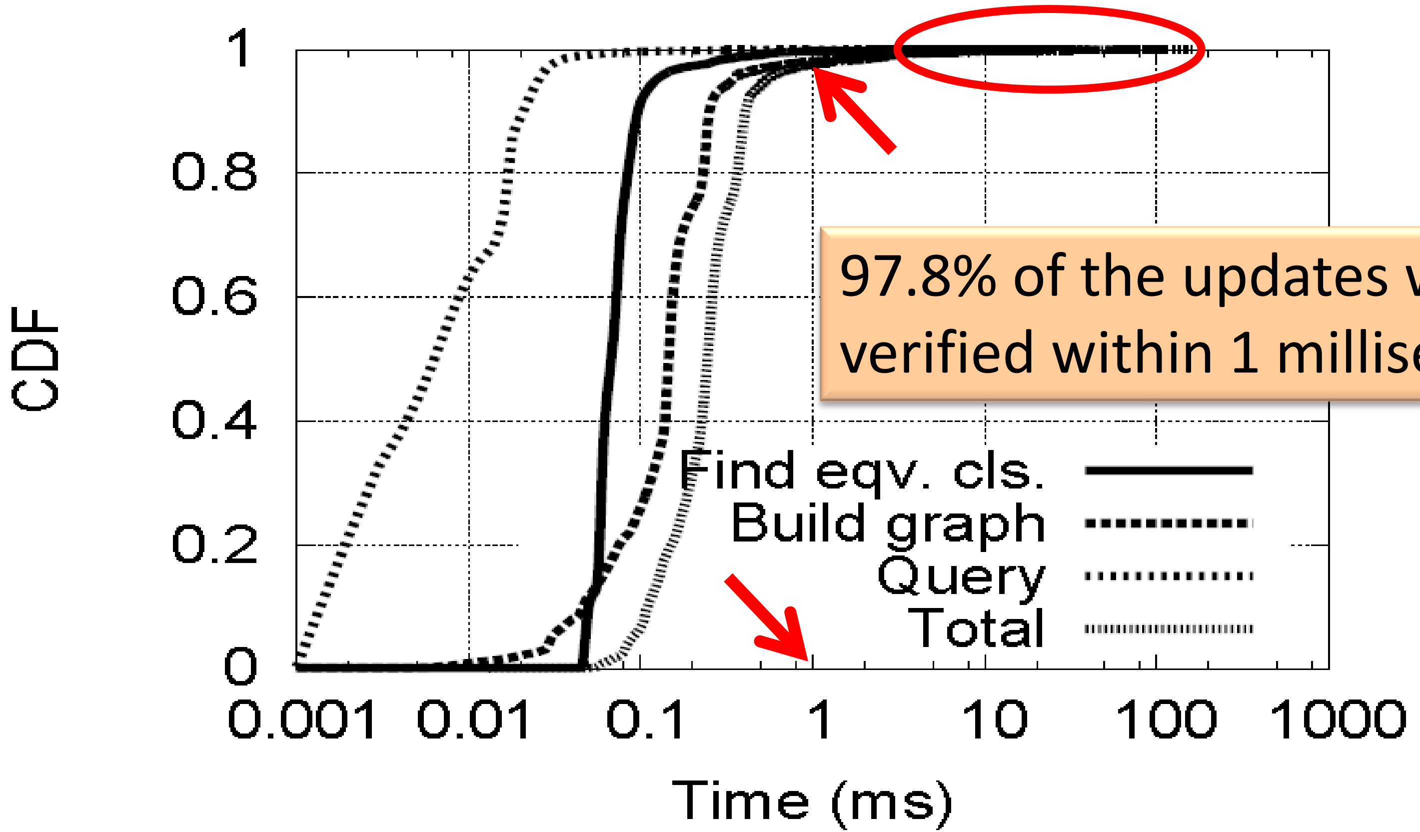
Diagnosis report

- Type of invariant violation
- Affected set of packets

API to write custom invariants

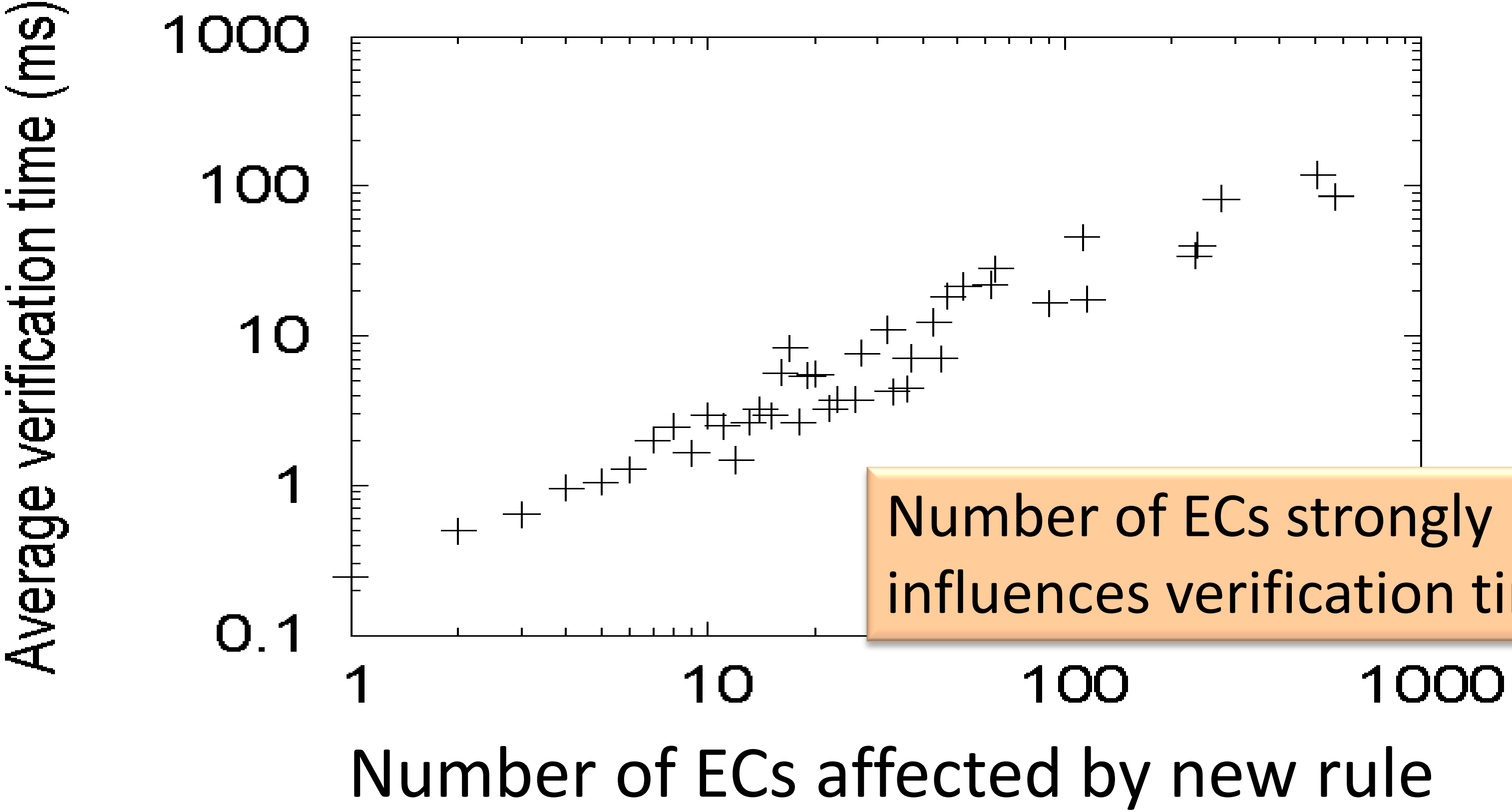
- VeriFlow provides a set of functions to write custom query algorithms
  - Gives access to the affected set of equivalence classes and their forwarding graphs
  - Verification becomes a standard graph traversal algorithm
- Can be used to
  - Check forwarding behavior of specific packet sets
  - Verify effects of potential changes

Performance Result



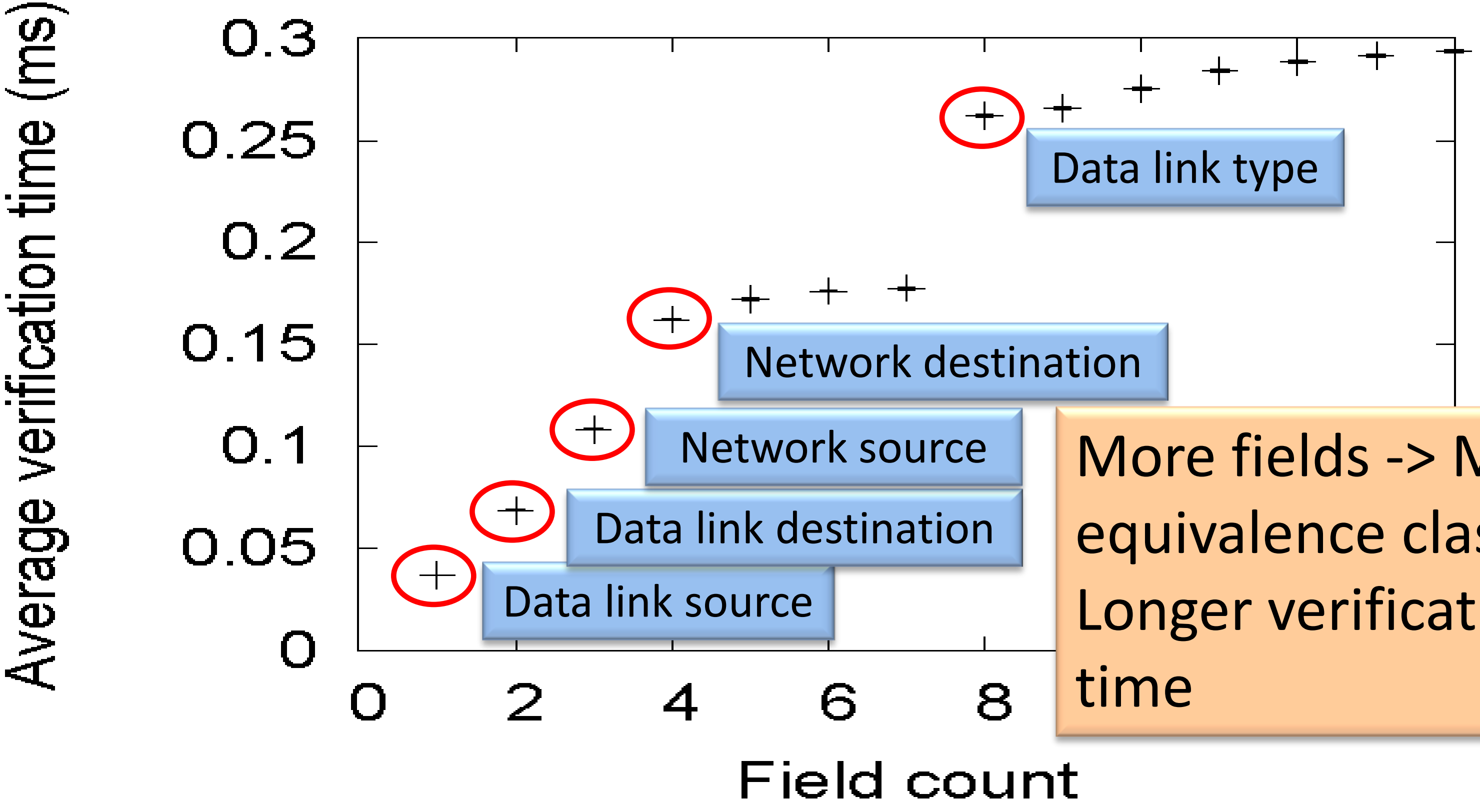
97.8% of the updates were verified within 1 millisecond

Effect of Equivalence Class Count



Number of ECs strongly influences verification time

Effect of Multiple Header Fields

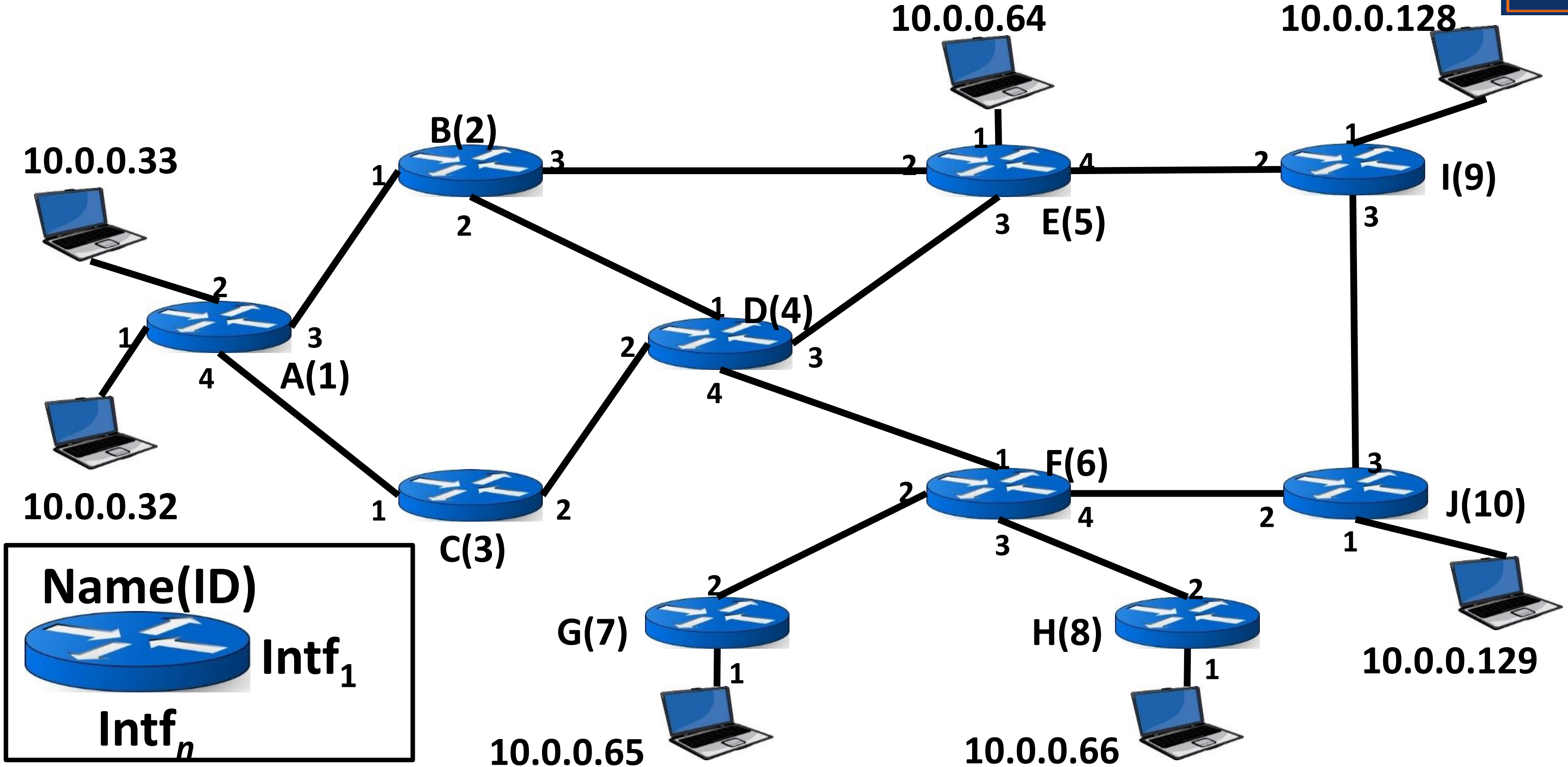


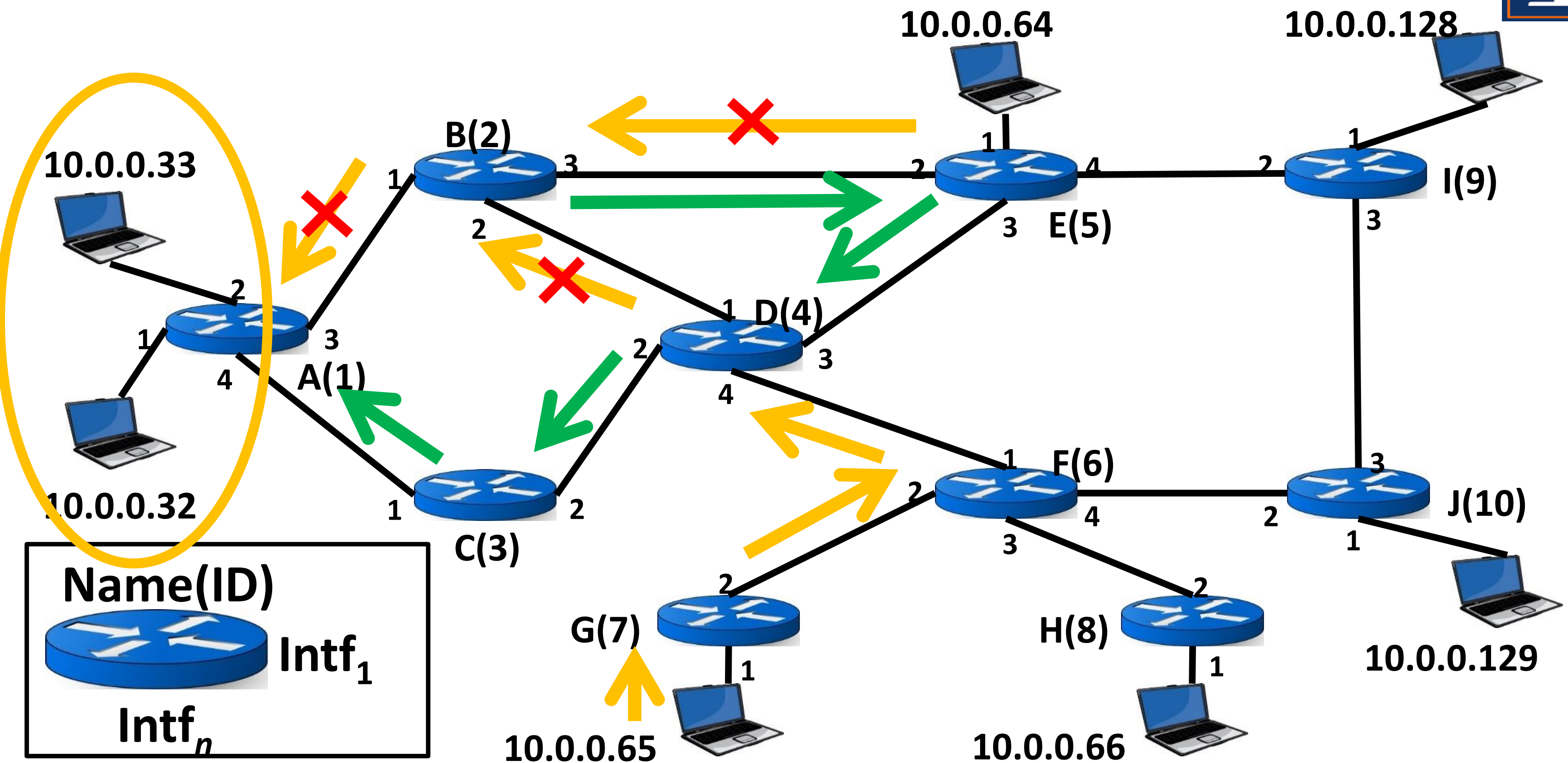
More fields -> More equivalence classes -> Longer verification time

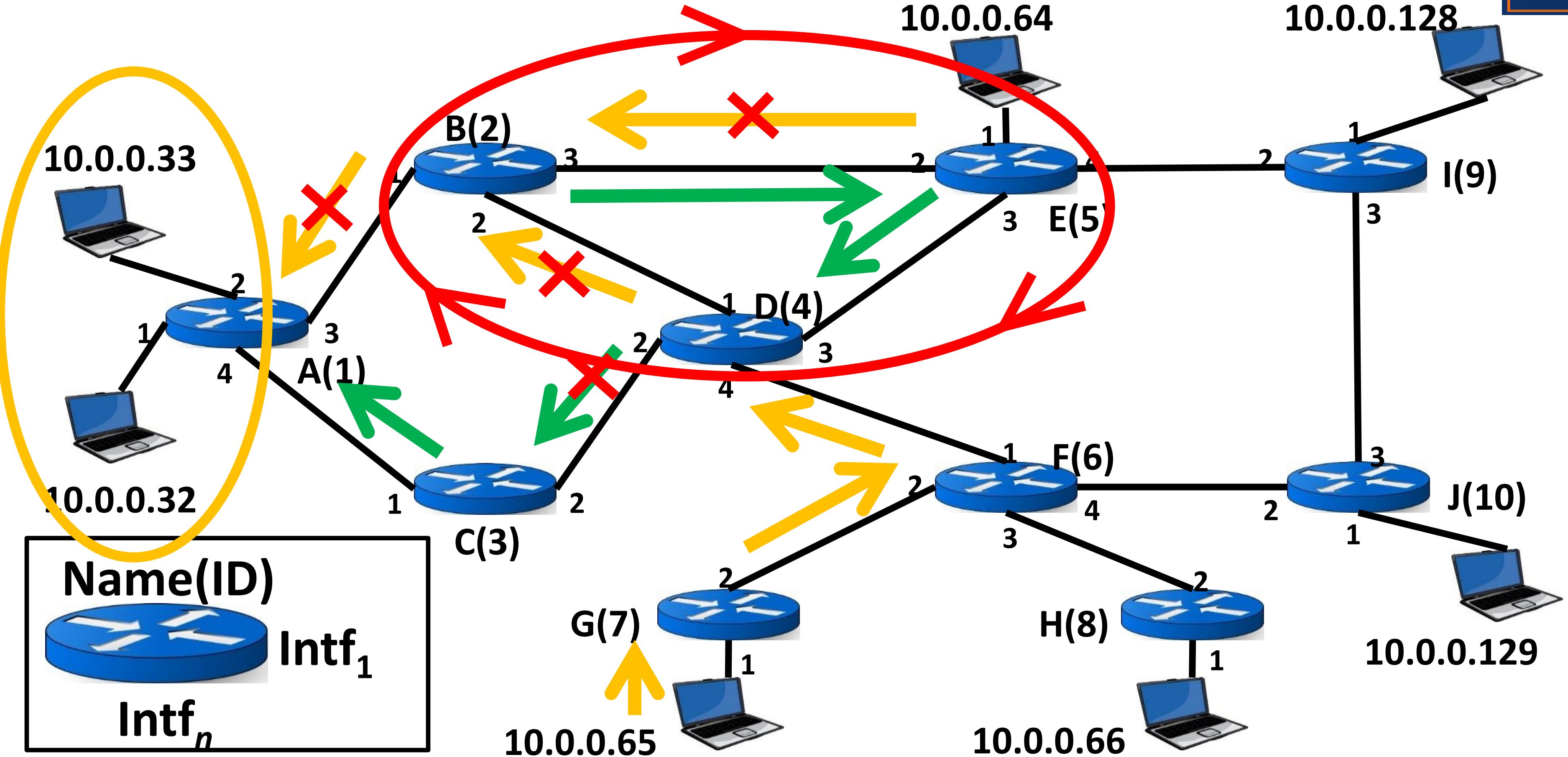
Related Work

- Header space analysis: Static checking for networks, NSDI 2012
- A NICE way to test OpenFlow applications, NSDI 2012
- Abstractions for network update, SIGCOMM 2012
- Debugging the data plane with Anteater, SIGCOMM 2011
- 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

Demo Network


10.0.0.64
10.0.0.128
B(2)
10.0.0.33
1
3
1
2
4
2
I(9)
2
3
3
E(5)
2
1
D(4)
1
3
2
A(1)
4
3
4
10.0.0.32
1
F(6)
3
2
J(10)
1
2
4
2
C(3)
3
1
2
2
G(7)
H(8)
1
1
10.0.0.129
10.0.0.65
10.0.0.66
Name(ID)
Intf1
Intfn


10.0.0.64
10.0.0.128
B(2)
10.0.0.33
1
3
2
1
4
2
3
E(5)
I(9)
3
2
1 D(4)
2
3
1
3
4
A(1)
4
1
F(6)
3
10.0.0.32
1
2
C(3)
2
4
2
J(10)
3
1
2
2
Name(ID)
Intf$_1$
Intf$_n$
G(7)
H(8)
1
1
10.0.0.129
10.0.0.65
10.0.0.66


10.0.0.64
10.0.0.128
10.0.0.33
B(2)
3
2
1
I(9)
E(5)
D(4)
A(1)
10.0.0.32
C(3)
F(6)
J(10)
G(7)
H(8)
10.0.0.129
10.0.0.65
10.0.0.66
Name(ID)
$Intf_1$
$Intf_n$