Online Data Plane Checking

June 12, 2013

Summer School on Formal Methods and Networks
Cornell University

VeriFlow: Verifying Network-Wide Invariants in Real Time*

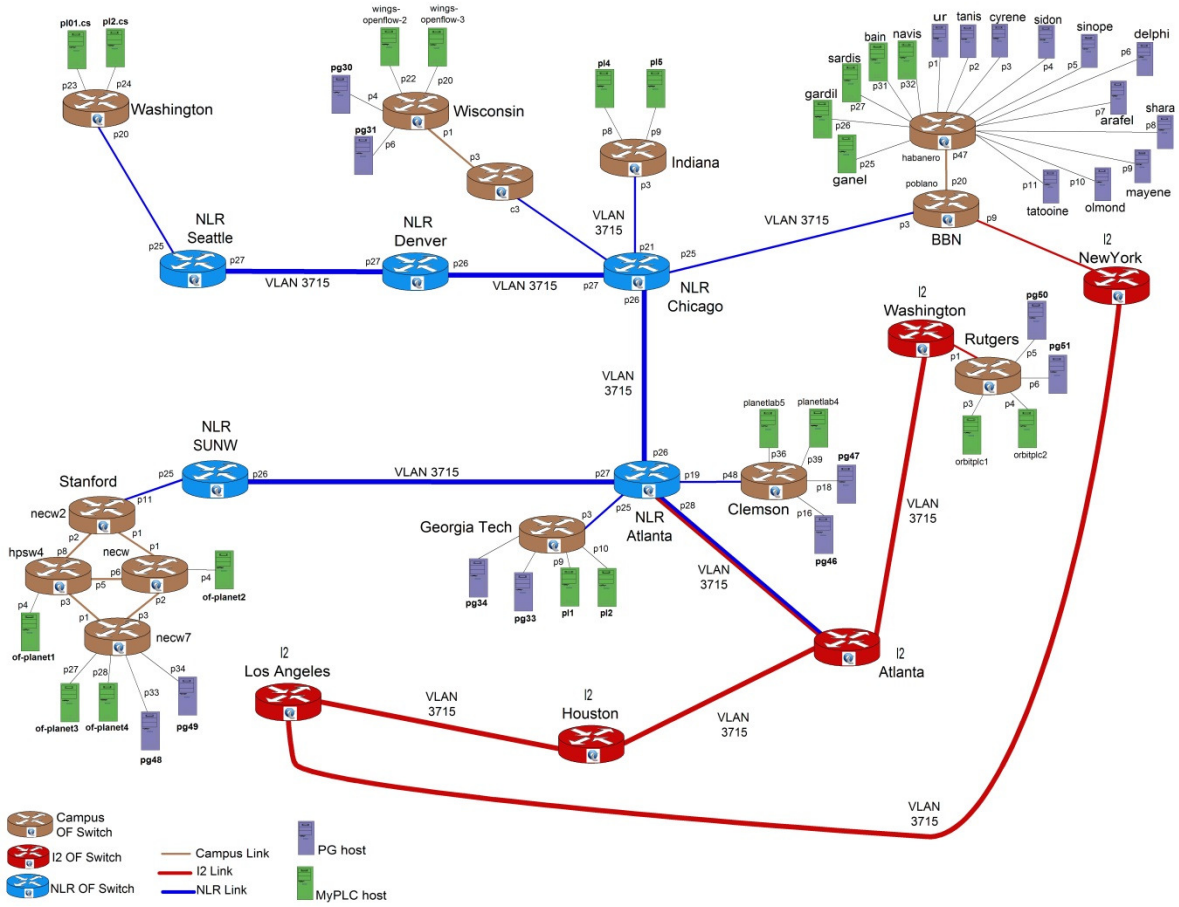
Ahmed Khurshid, Xuan Zou, Wenxuan Zhou,
Matthew Caesar, P. Brighten Godfrey
University of Illinois at Urbana-Champaign (UIUC)

June 12, 2013

Summer School on Formal Methods and Networks
Cornell University

***HotSDN 2012, NSDI 2013, ONS 2013**

Challenges in Network Debugging



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

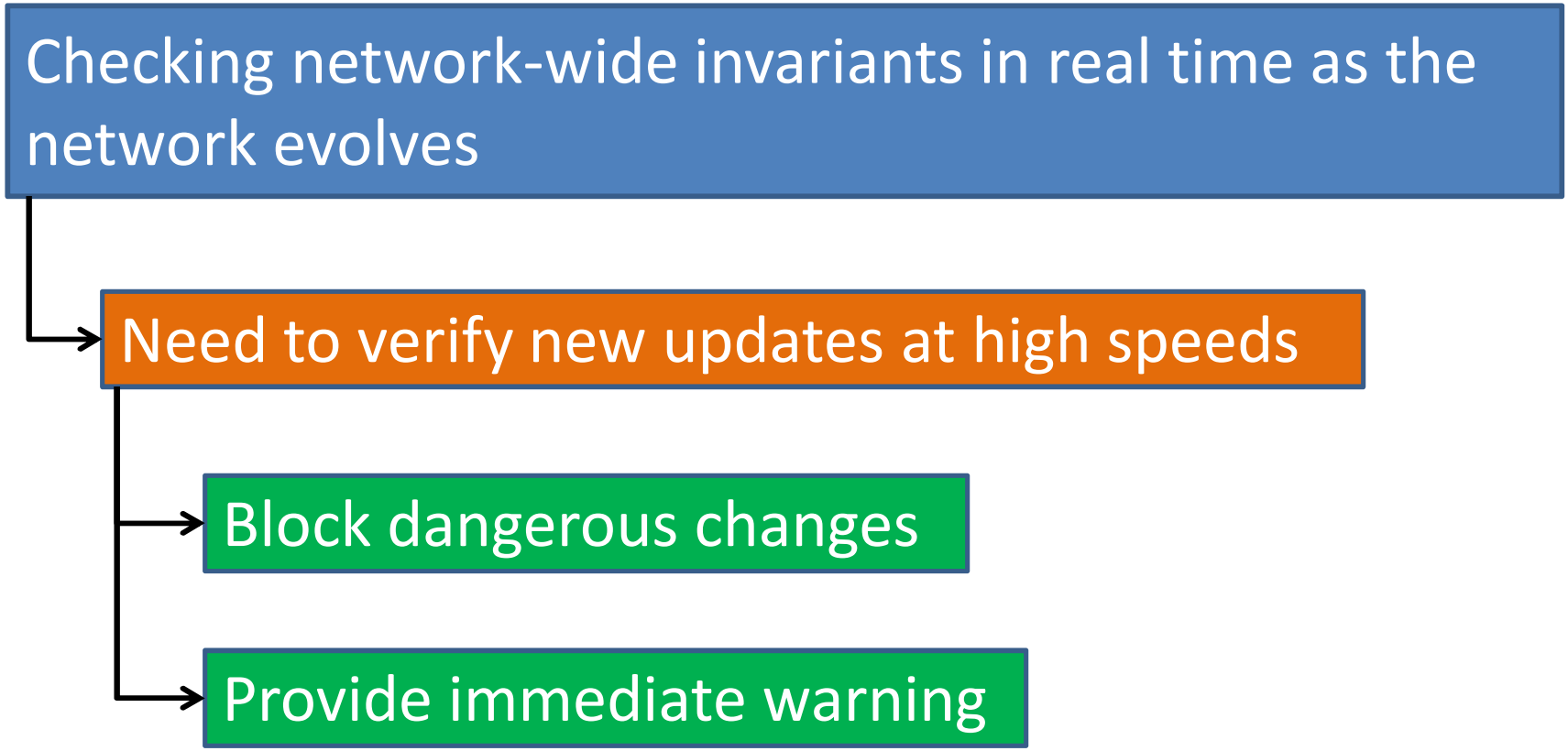
Data Plane Verification in Action

- 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

Can we run verification in real time?



Challenges in Real-Time Verification

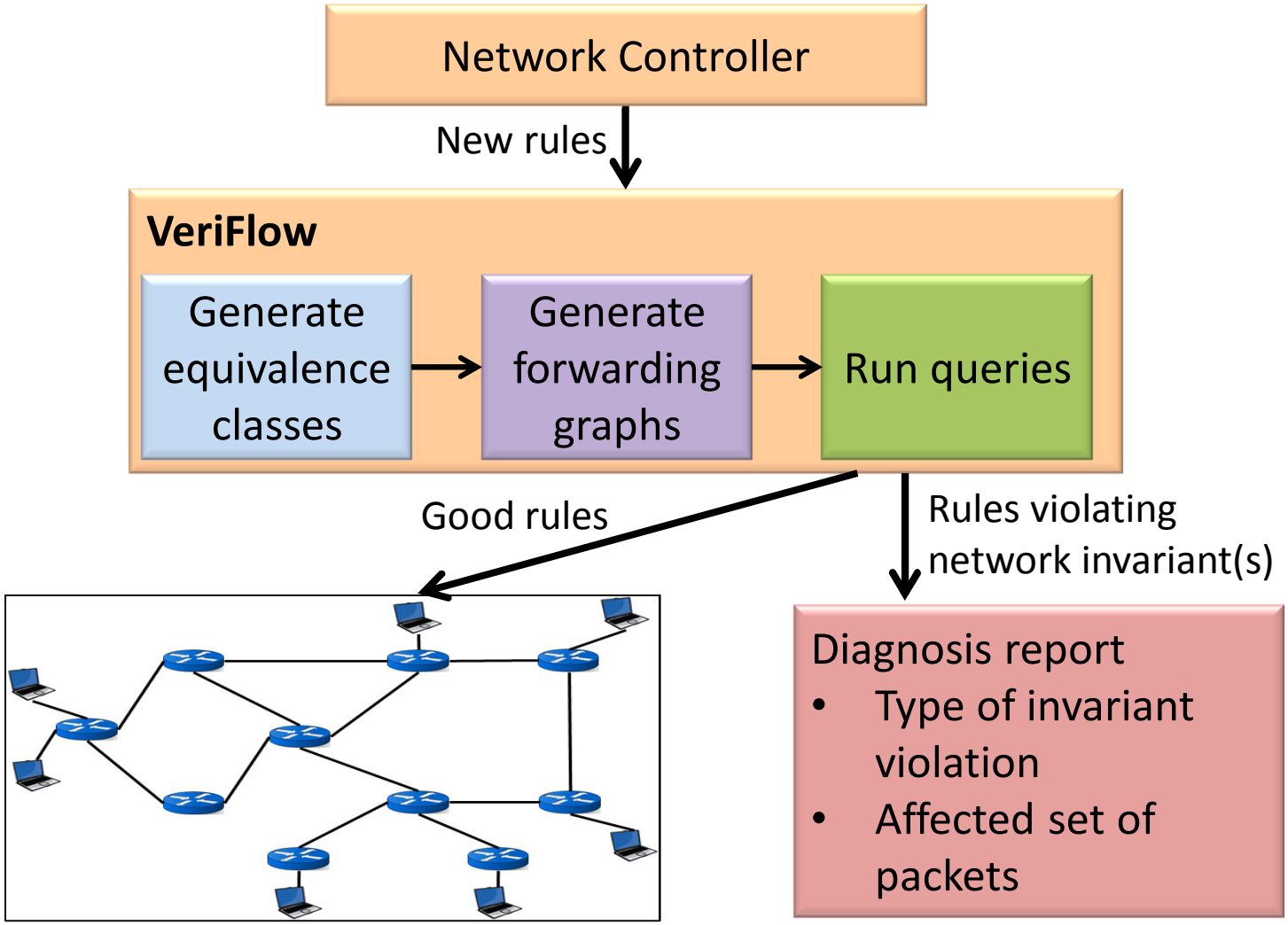
- Challenge 1: Obtaining real-time view of network
  - Solution: Utilize the **centralized** data-plane view available in an **SDN (Software-Defined Network)**
- Challenge 2: Verification speed
  - Solution: Off-the-shelf techniques?

No, too slow!

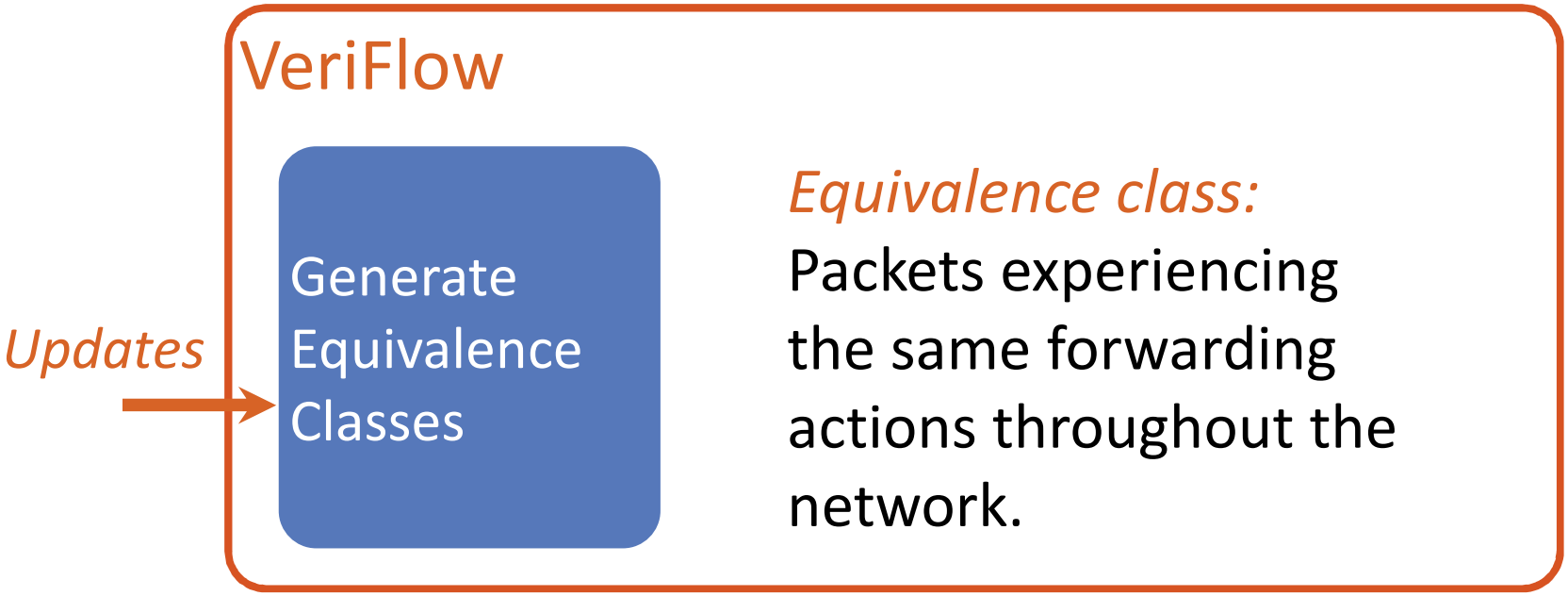
Our Tool: VeriFlow

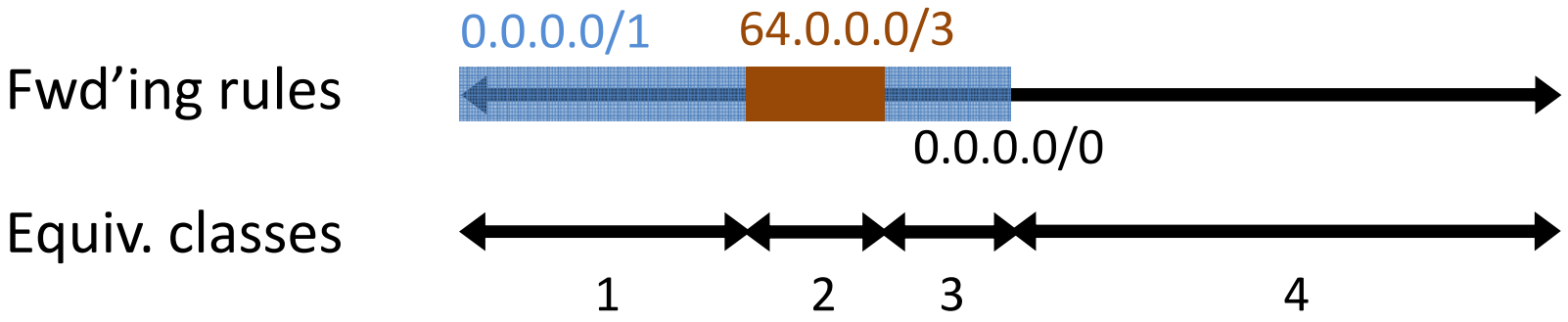
- VeriFlow checks network-wide invariants in **real time** using data-plane state
  - Absence of routing loops and black holes, access control violations, etc.
- VeriFlow functions by
  - Monitoring **dynamic changes** in the network
  - Constructing a **model** of the **network behavior**
  - Using **custom algorithms** to automatically derive whether the network contains errors

VeriFlow Operation

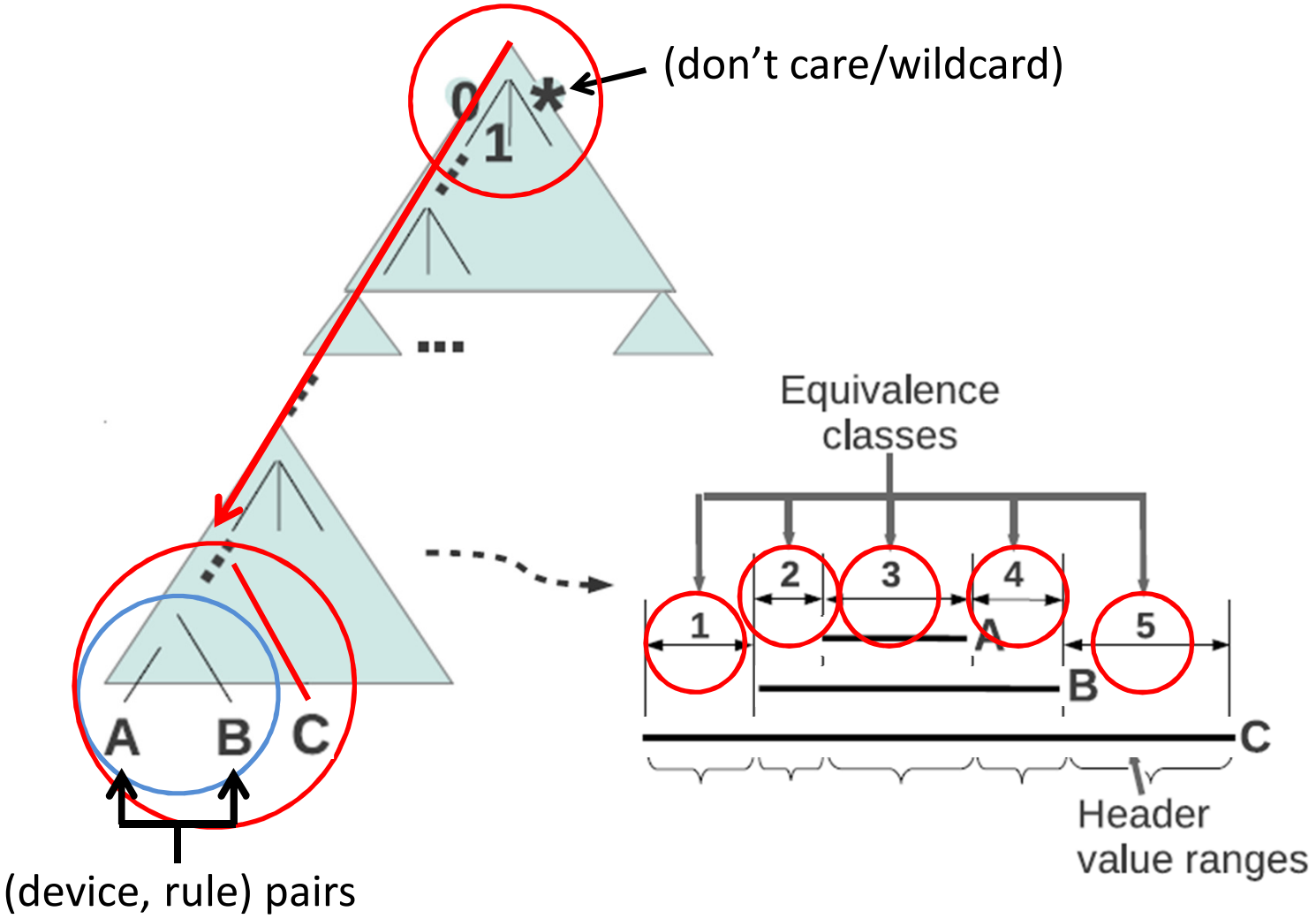


1. Limit the Search Space

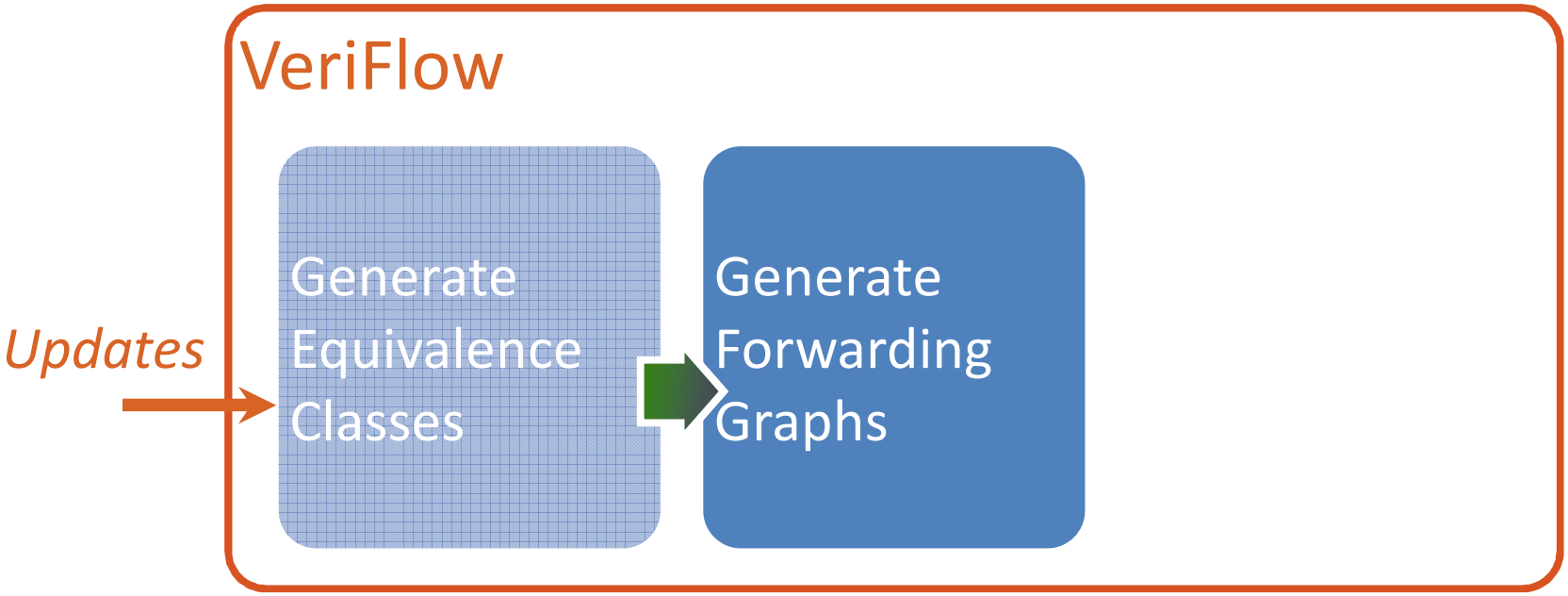


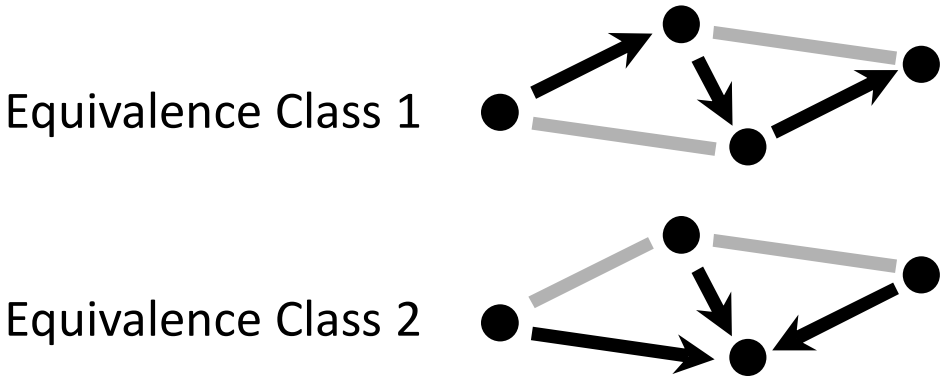


Computing Equivalence Classes



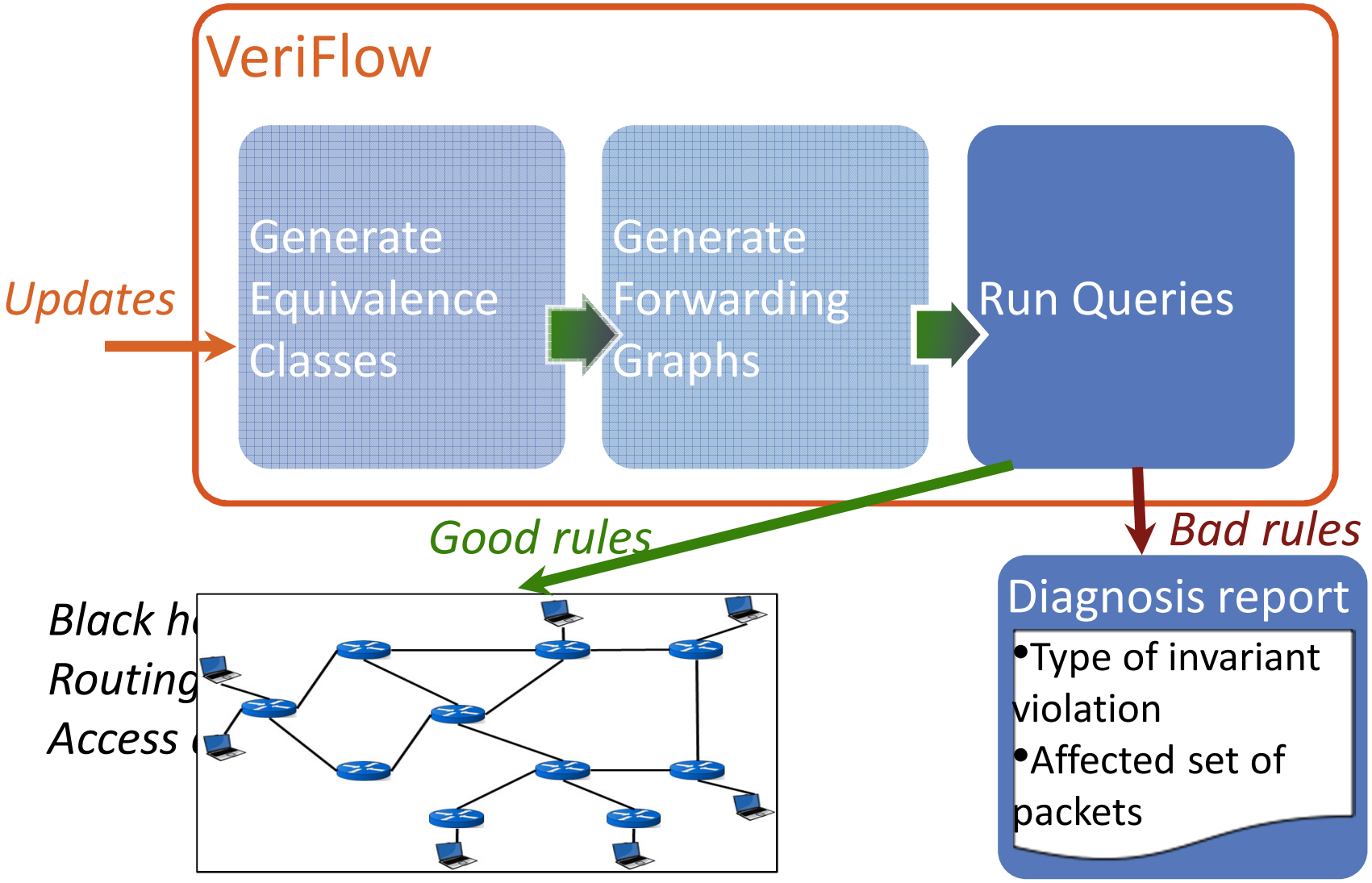
2. Represent Forwarding Behavior





All the info to answer queries!

3. Run Query to Check Invariants



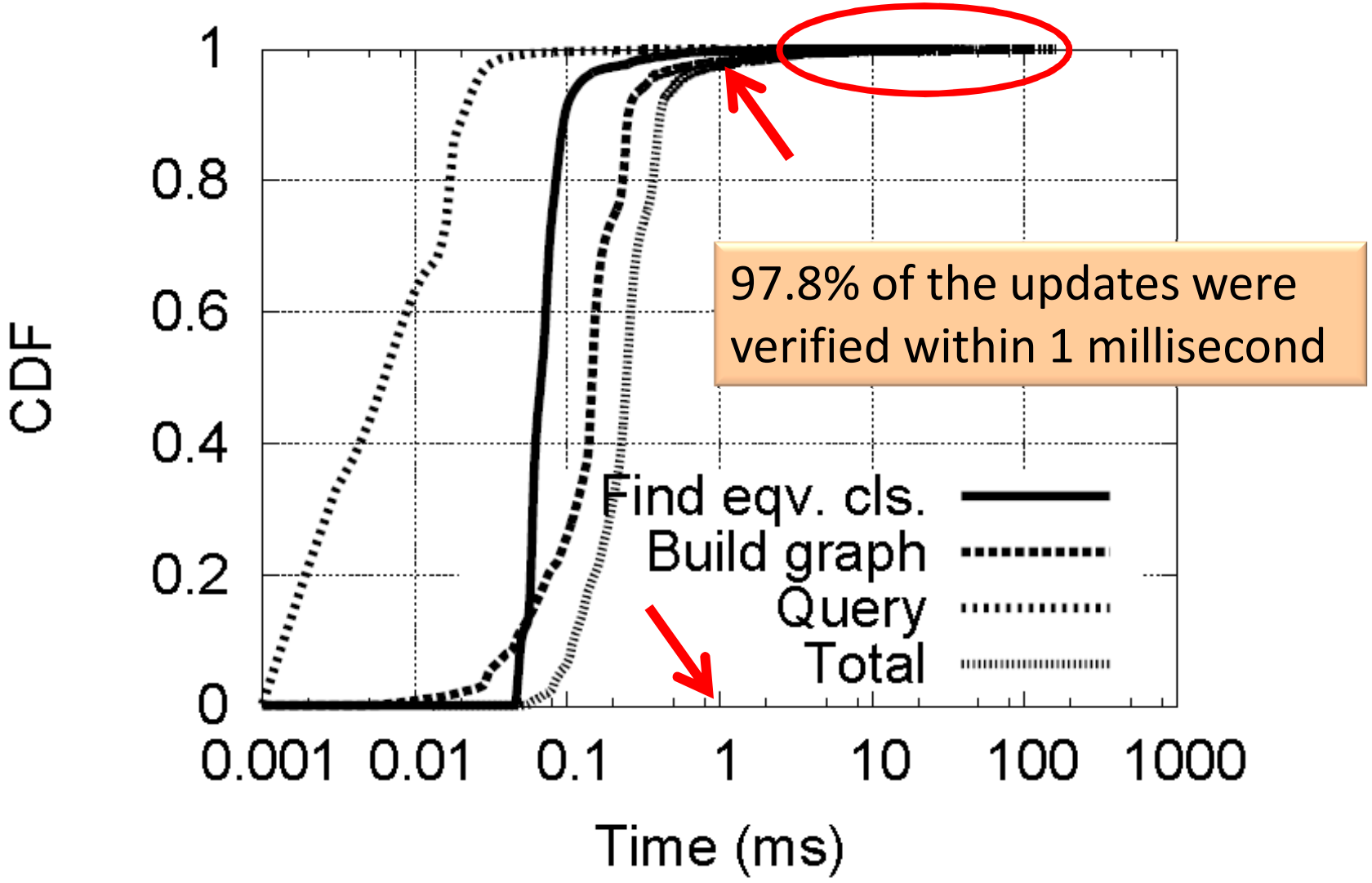
API to write custom invariants

- 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

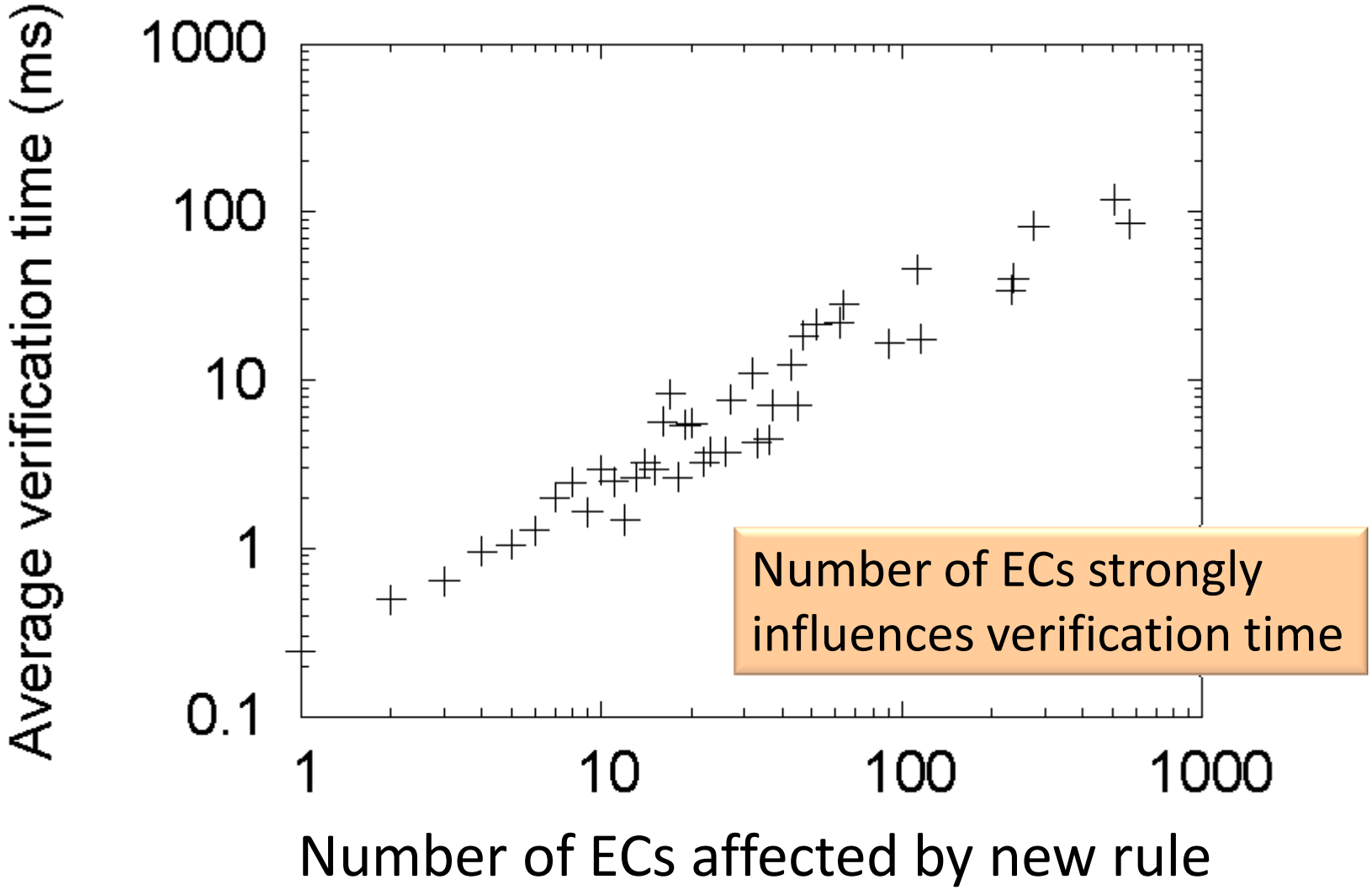
Experiment

- 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

Performance Result

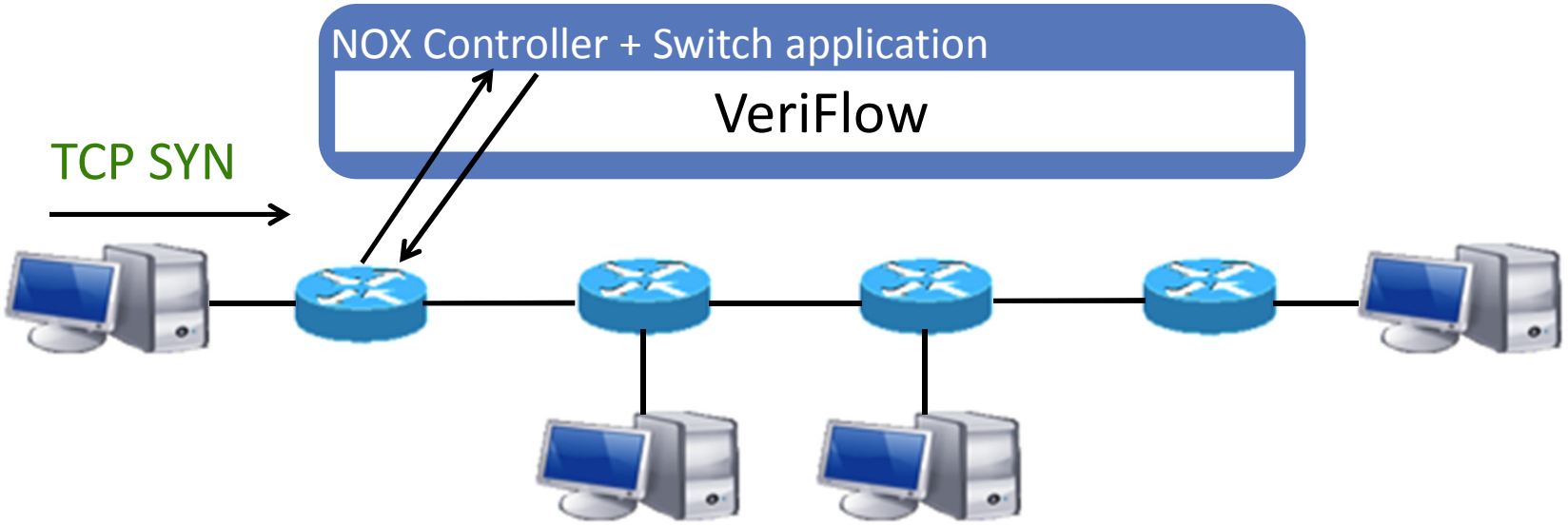


Effect of Equivalence Class Count

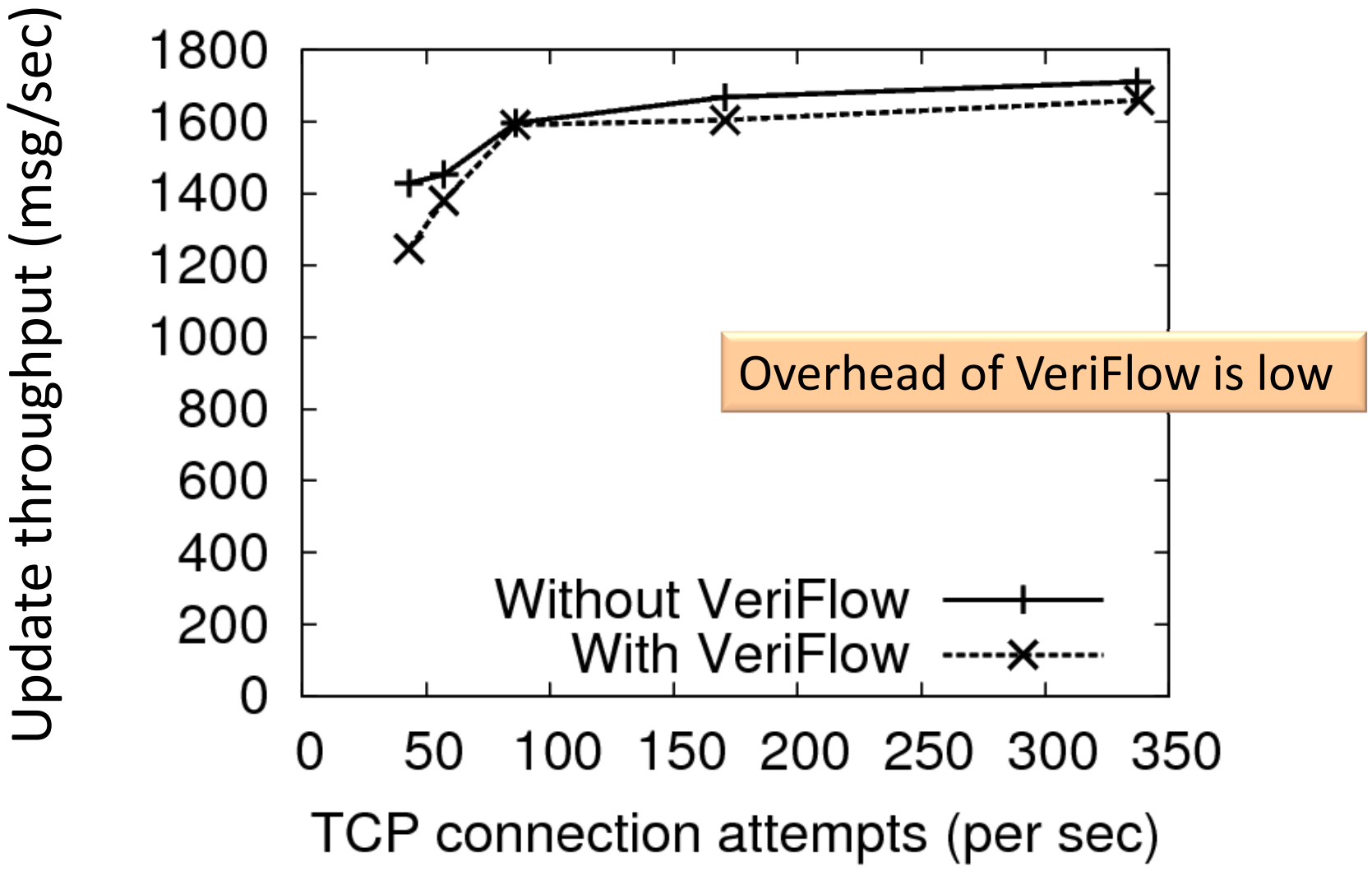


Experiment (cont.)

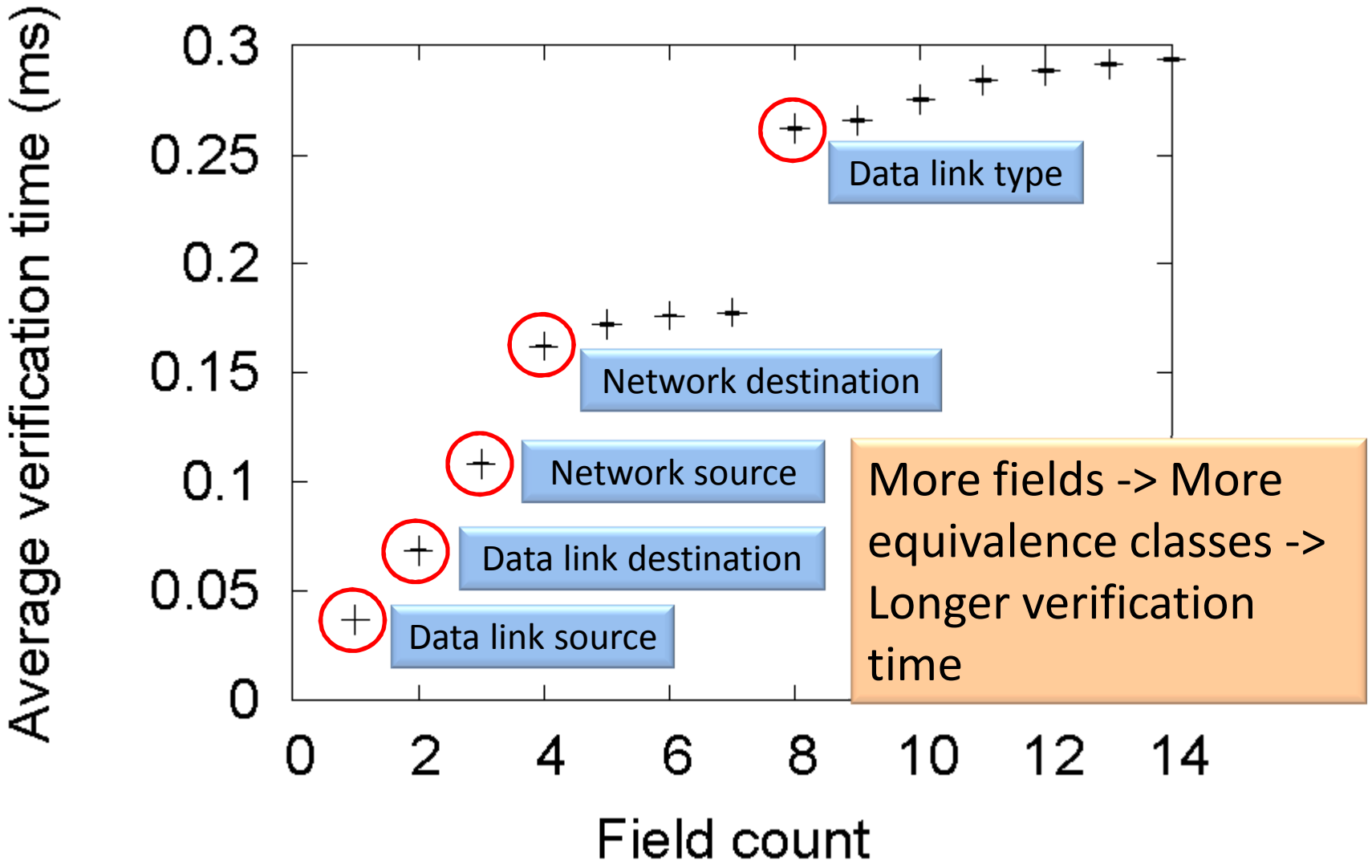
- Mininet OpenFlow network
  - Rocketfuel topology with 172 switches, one host per switch
- NOX controller, learning switch application
- TCP connections between random pairs of hosts



Effect on Flow Table Update Throughput



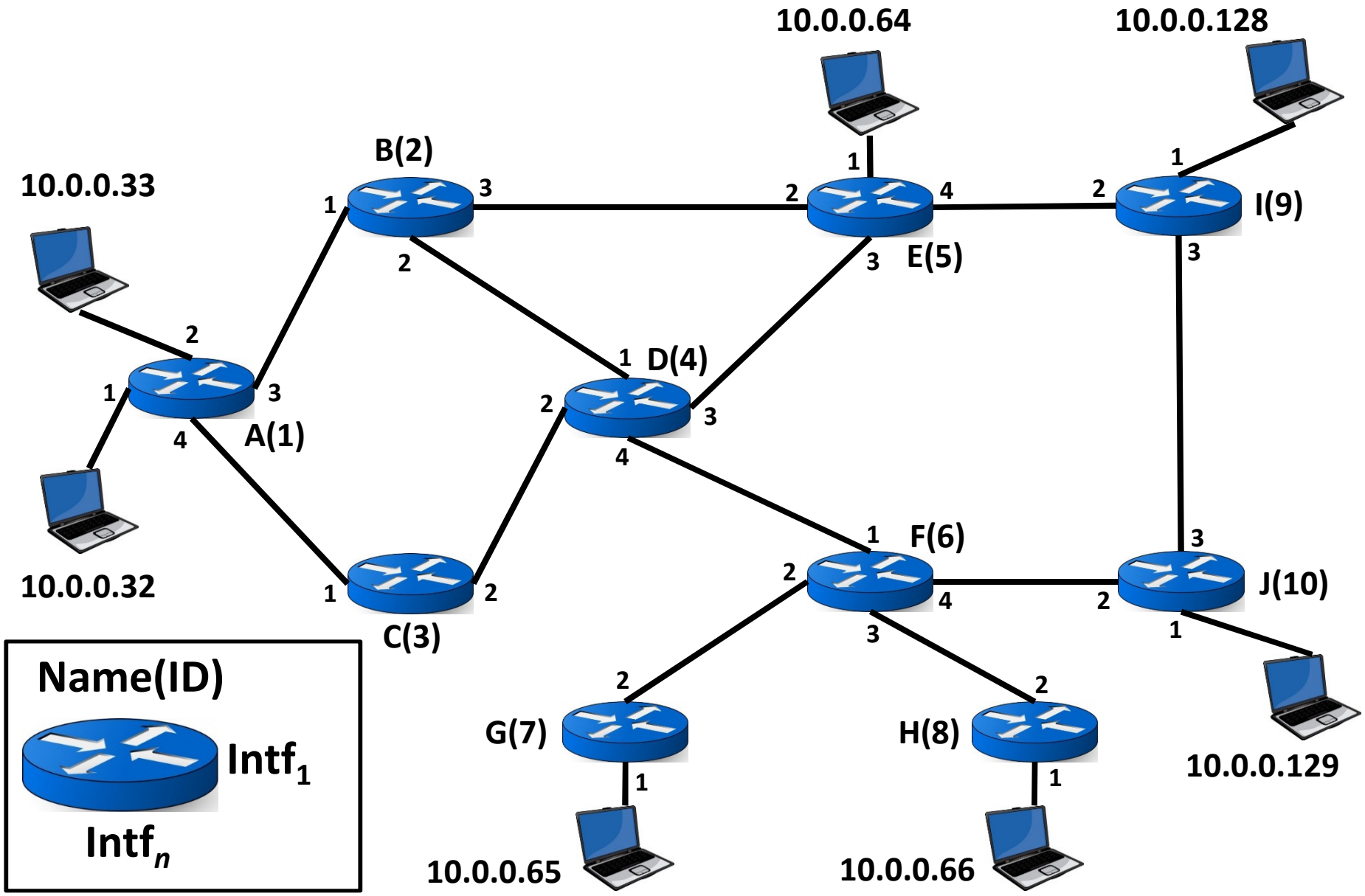
Effect of Multiple Header Fields

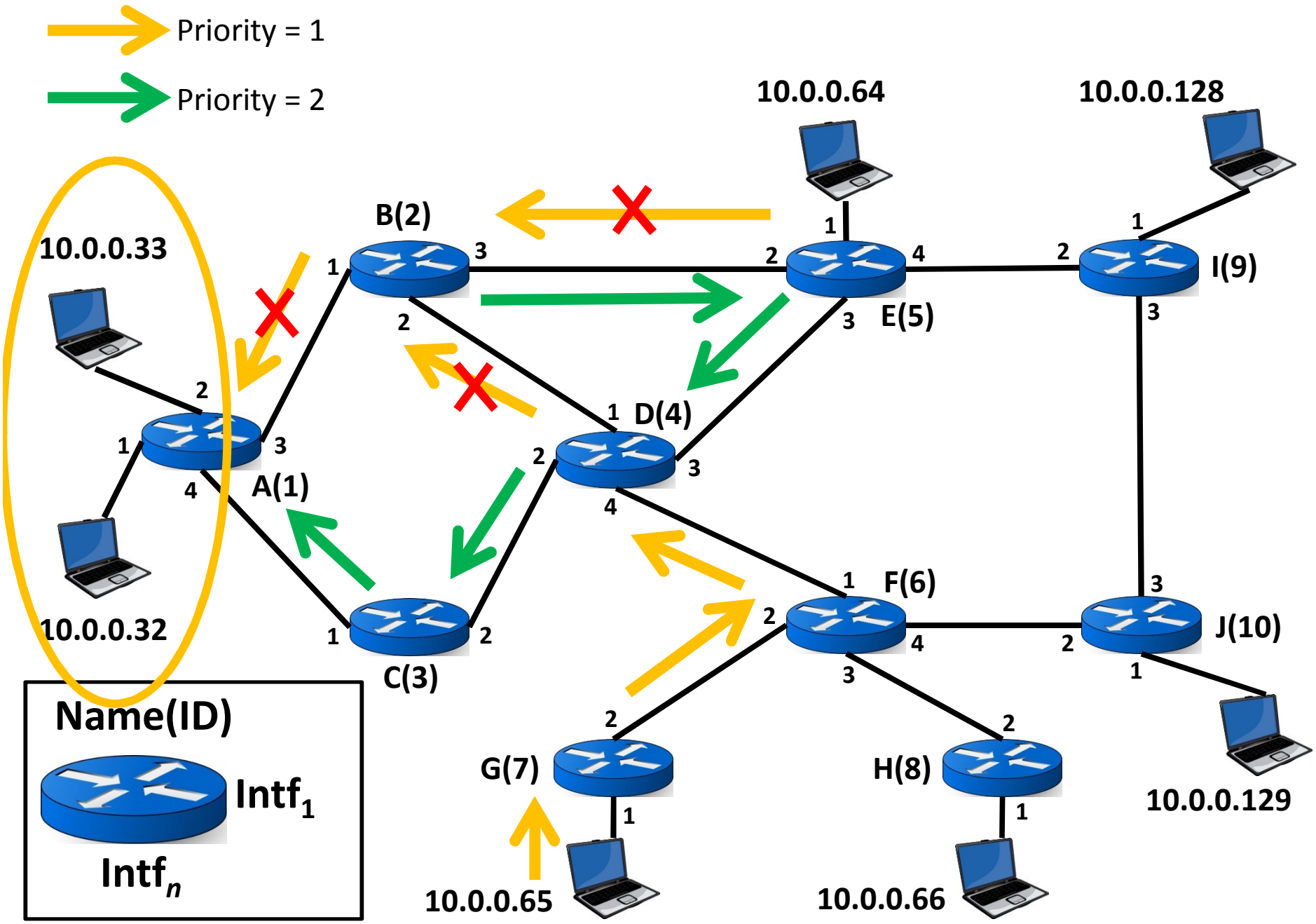


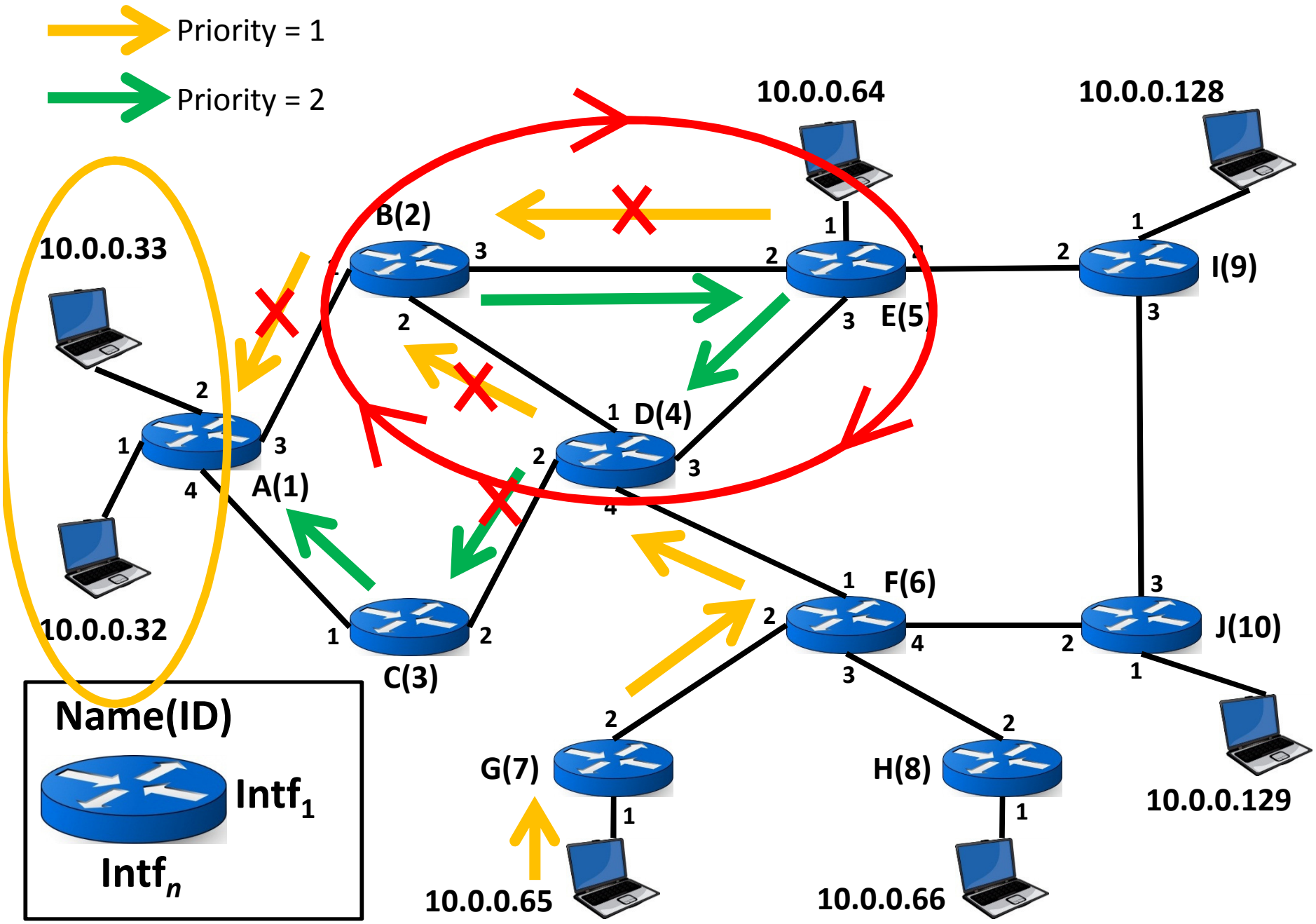
Conclusion

- 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

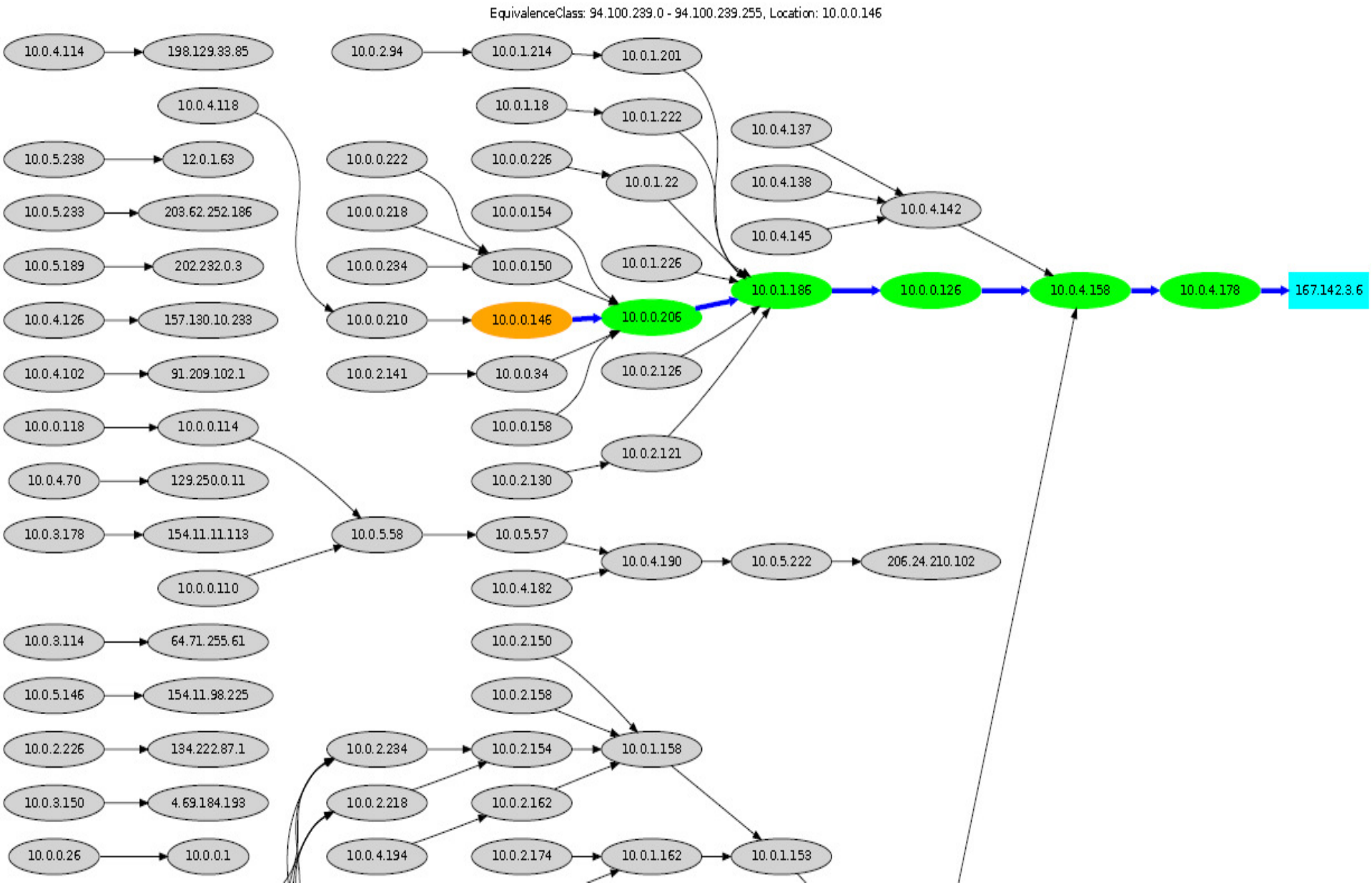
Demo Network

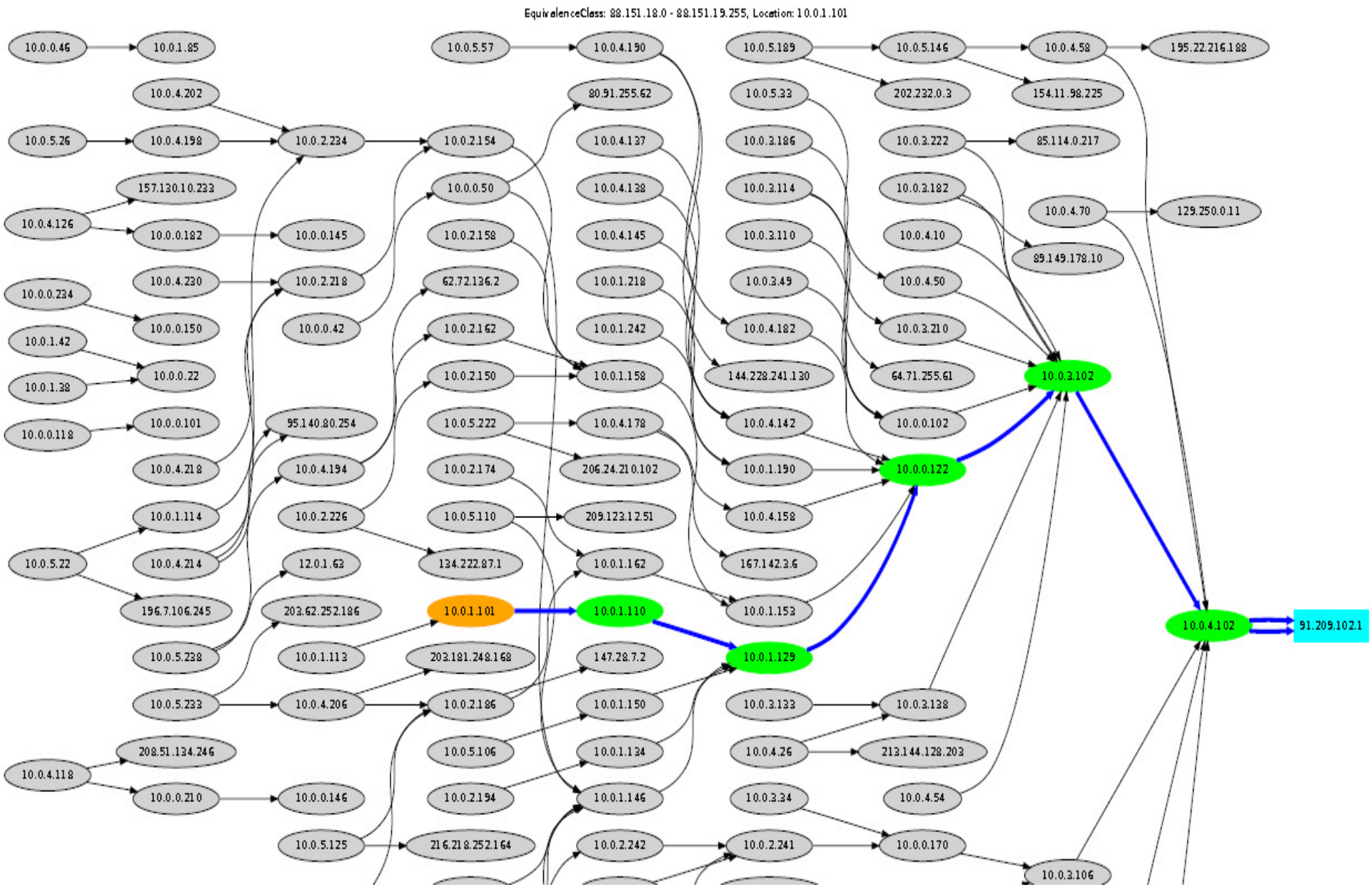

10.0.0.64
10.0.0.128
B(2)
10.0.0.33
E(5)
I(9)
D(4)
A(1)
10.0.0.32
C(3)
F(6)
J(10)
G(7)
H(8)
10.0.0.129
10.0.0.65
10.0.0.66
Name(ID)
Intf1
Intfn

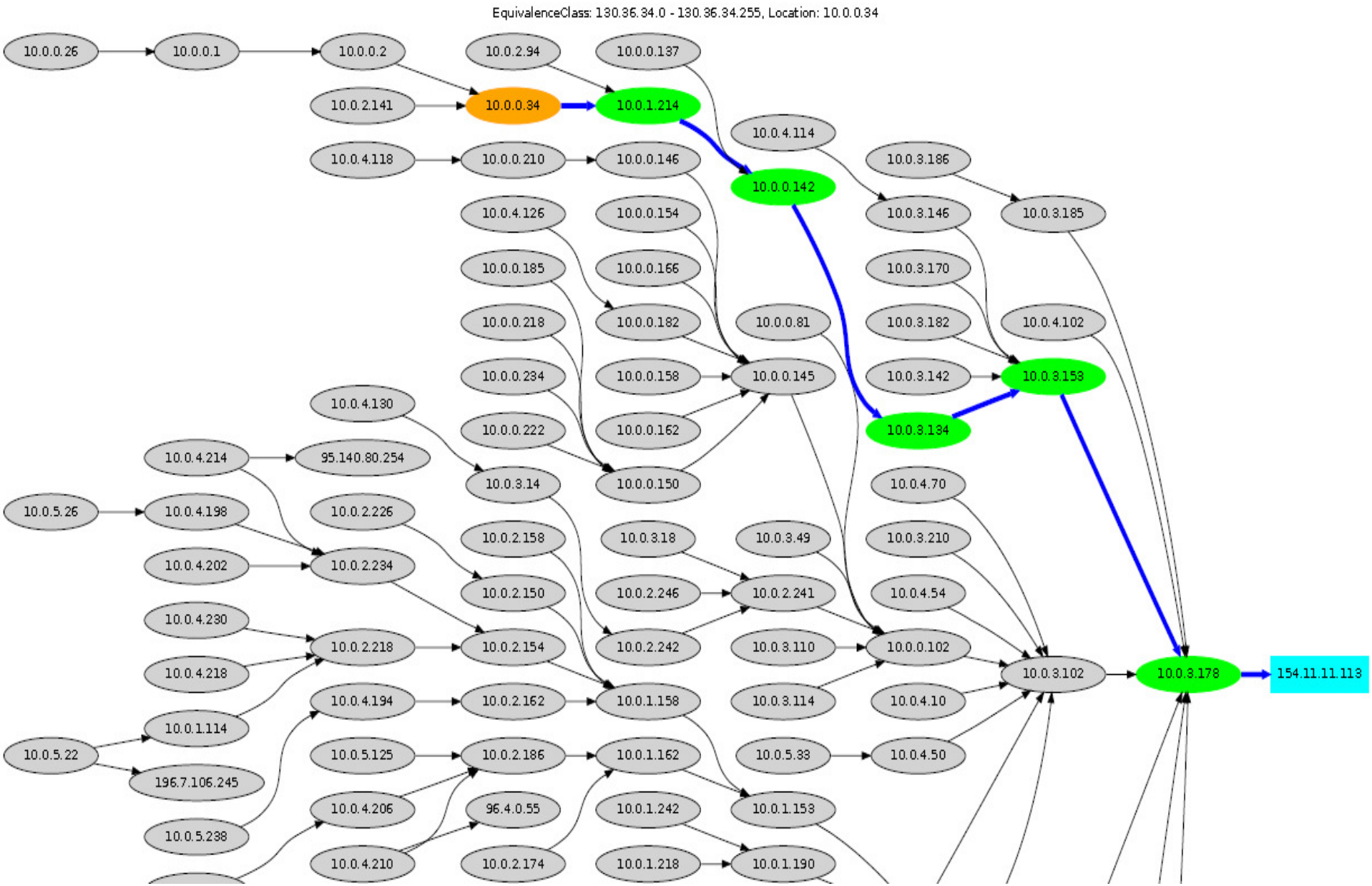

Priority = 1
Priority = 2
10.0.0.64
10.0.0.128
10.0.0.33
B(2)
I(9)
E(5)
D(4)
A(1)
10.0.0.32
C(3)
F(6)
J(10)
G(7)
H(8)
10.0.0.129
10.0.0.65
10.0.0.66
Name(ID)
Intf1
Intfn

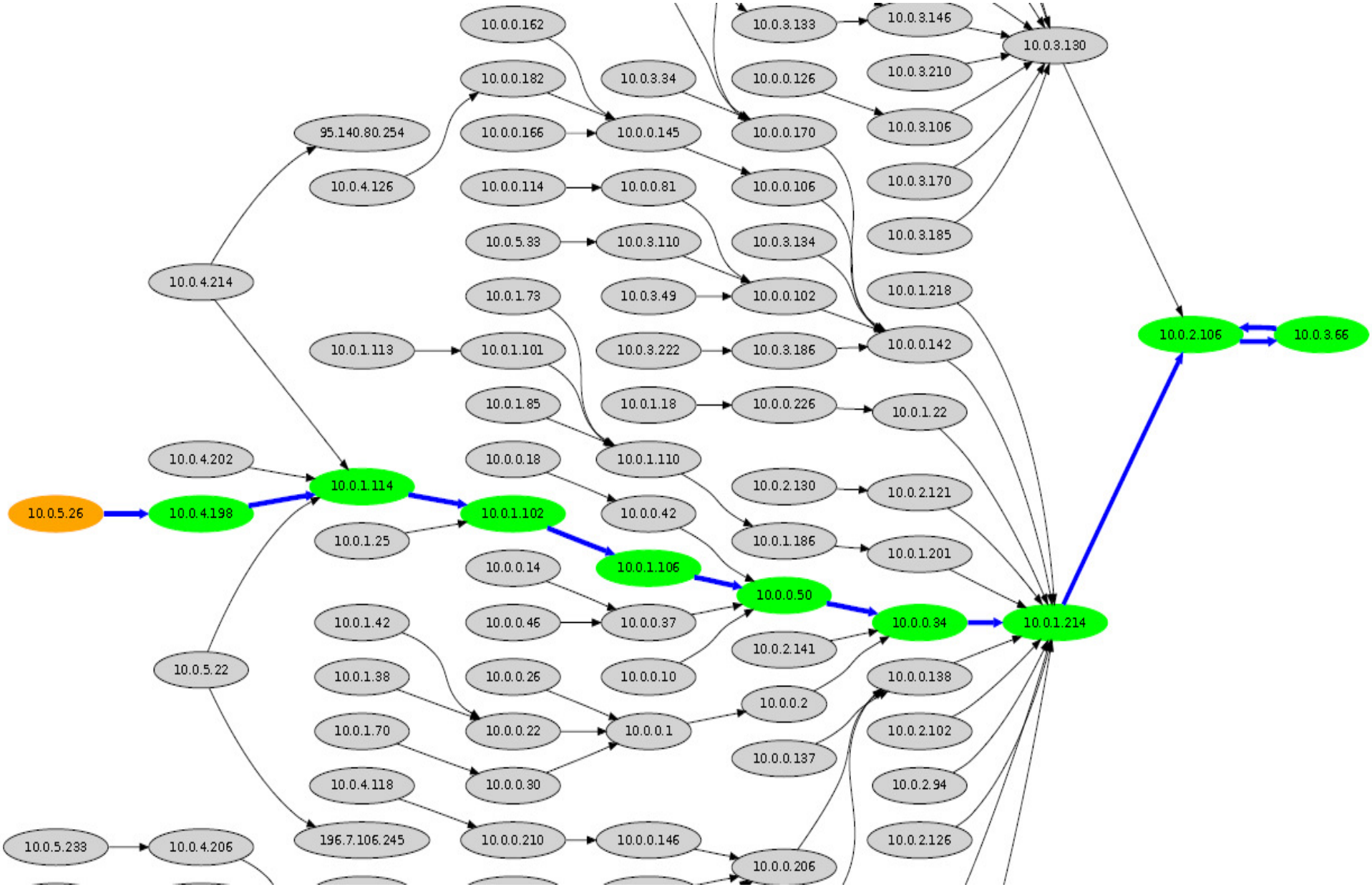

Priority = 1
Priority = 2
10.0.0.64
10.0.0.128
10.0.0.33
B(2)
I(9)
E(5)
D(4)
A(1)
10.0.0.32
C(3)
F(6)
J(10)
G(7)
H(8)
10.0.0.129
10.0.0.65
10.0.0.66
Name(ID)
Intf1
Intfn

Forwarding Graphs from the Rocketfuel-RouteViews Experiment


EquivalenceClass: 94.100.239.0 - 94.100.239.255, Location: 10.0.0.146
10.0.4.114
198.129.33.85
10.0.4.118
10.0.5.238
12.0.1.63
10.0.5.233
203.62.252.186
10.0.5.189
202.232.0.3
10.0.4.126
157.130.10.233
10.0.4.102
91.209.102.1
10.0.0.118
10.0.0.114
10.0.4.70
129.250.0.11
10.0.3.178
154.11.11.113
10.0.0.110
10.0.3.114
64.71.255.61
10.0.5.146
154.11.98.225
10.0.2.226
134.222.87.1
10.0.3.150
4.69.184.193
10.0.0.26
10.0.0.1
10.0.2.94
10.0.1.214
10.0.1.201
10.0.1.18
10.0.1.222
10.0.0.222
10.0.0.226
10.0.1.22
10.0.0.218
10.0.0.154
10.0.0.234
10.0.0.150
10.0.1.226
10.0.0.210
10.0.0.146
10.0.0.206
10.0.2.141
10.0.0.34
10.0.2.126
10.0.0.158
10.0.2.121
10.0.2.130
10.0.5.58
10.0.5.57
10.0.4.190
10.0.5.222
206.24.210.102
10.0.4.182
10.0.2.150
10.0.2.158
10.0.1.158
10.0.2.234
10.0.2.154
10.0.2.218
10.0.2.162
10.0.4.194
10.0.2.174
10.0.1.162
10.0.1.153
10.0.4.137
10.0.4.138
10.0.4.142
10.0.4.145
10.0.1.186
10.0.0.126
10.0.4.158
10.0.4.178
167.142.3.6


EquivalenceClass: 88.151.18.0 - 88.151.19.255, Location: 10.0.1.101
10.0.0.46
10.0.1.85
10.0.4.202
10.0.5.26
10.0.4.198
10.0.2.234
157.130.10.233
10.0.4.126
10.0.0.182
10.0.0.145
10.0.4.230
10.0.2.218
10.0.0.234
10.0.0.150
10.0.1.42
10.0.0.22
10.0.1.38
10.0.0.118
10.0.0.101
10.0.4.218
95.140.80.254
10.0.4.194
10.0.1.114
10.0.5.22
10.0.4.214
12.0.1.63
196.7.106.245
203.62.252.186
10.0.5.238
10.0.1.113
10.0.5.233
10.0.4.206
208.51.134.246
10.0.4.118
10.0.0.210
10.0.0.146
10.0.5.125
216.218.252.164
10.0.5.57
10.0.4.190
80.91.255.62
10.0.2.154
10.0.0.50
10.0.2.158
62.72.136.2
10.0.0.42
10.0.2.162
10.0.2.150
10.0.5.222
206.24.210.102
10.0.2.174
10.0.5.110
209.123.12.51
10.0.2.226
134.222.87.1
10.0.0.1.101
203.181.248.168
10.0.2.186
10.0.5.106
10.0.2.194
10.0.4.137
10.0.4.138
10.0.4.145
10.0.1.218
10.0.1.242
10.0.1.158
10.0.4.178
10.0.1.162
147.28.7.2
10.0.1.150
10.0.1.134
10.0.1.146
10.0.2.242
10.0.1.110
10.0.5.189
10.0.5.146
10.0.4.58
195.22.216.188
202.232.0.3
154.11.98.225
10.0.5.33
10.0.3.186
10.0.3.114
10.0.3.110
10.0.3.49
10.0.4.182
144.228.241.130
10.0.4.142
10.0.1.190
10.0.4.158
167.142.3.6
10.0.1.153
10.0.1.129
10.0.3.133
10.0.4.26
10.0.3.34
10.0.2.241
10.0.3.222
85.114.0.217
10.0.3.182
10.0.4.10
89.149.178.10
10.0.4.50
10.0.3.210
64.71.255.61
10.0.0.102
10.0.0.122
10.0.3.138
213.144.128.203
10.0.4.54
10.0.0.170
10.0.4.70
129.250.0.11
10.0.3.102
10.0.4.102
91.209.102.1
10.0.3.106


EquivalenceClass: 130.36.34.0 - 130.36.34.255, Location: 10.0.0.34
10.0.0.26
10.0.0.1
10.0.0.2
10.0.2.94
10.0.0.137
10.0.2.141
10.0.0.34
10.0.1.214
10.0.4.114
10.0.3.186
10.0.4.118
10.0.0.210
10.0.0.146
10.0.0.142
10.0.3.146
10.0.3.185
10.0.4.126
10.0.0.154
10.0.0.185
10.0.0.166
10.0.3.170
10.0.0.218
10.0.0.182
10.0.0.81
10.0.3.182
10.0.4.102
10.0.0.234
10.0.0.158
10.0.0.145
10.0.3.142
10.0.3.153
10.0.4.130
10.0.0.222
10.0.0.162
10.0.3.134
10.0.4.214
95.140.80.254
10.0.3.14
10.0.0.150
10.0.4.70
10.0.5.26
10.0.4.198
10.0.2.226
10.0.2.158
10.0.3.18
10.0.3.49
10.0.3.210
10.0.4.202
10.0.2.234
10.0.2.150
10.0.2.246
10.0.2.241
10.0.4.54
10.0.4.230
10.0.2.218
10.0.2.154
10.0.2.242
10.0.3.110
10.0.0.102
10.0.4.218
10.0.1.114
10.0.4.194
10.0.2.162
10.0.1.158
10.0.3.114
10.0.4.10
10.0.3.102
10.0.3.178
154.11.11.113
10.0.5.22
196.7.106.245
10.0.5.125
10.0.2.186
10.0.1.162
10.0.5.33
10.0.4.50
10.0.4.206
96.4.0.55
10.0.1.242
10.0.1.153
10.0.5.238
10.0.4.210
10.0.2.174
10.0.1.218
10.0.1.190


10.0.0.162
10.0.0.182
95.140.80.254
10.0.0.166
10.0.4.126
10.0.0.114
10.0.5.33
10.0.4.214
10.0.1.73
10.0.1.113
10.0.1.101
10.0.1.85
10.0.4.202
10.0.5.26
10.0.4.198
10.0.1.114
10.0.1.25
10.0.0.18
10.0.0.14
10.0.1.42
10.0.5.22
10.0.1.38
10.0.1.70
10.0.4.118
10.0.5.233
10.0.4.206
196.7.106.245
10.0.0.210
10.0.0.146
10.0.3.34
10.0.0.145
10.0.0.81
10.0.3.110
10.0.3.49
10.0.3.222
10.0.1.18
10.0.1.110
10.0.0.42
10.0.1.102
10.0.1.106
10.0.0.46
10.0.0.37
10.0.0.26
10.0.0.22
10.0.0.1
10.0.0.30
10.0.0.10
10.0.0.50
10.0.0.206
10.0.3.133
10.0.0.126
10.0.0.170
10.0.0.106
10.0.3.134
10.0.0.102
10.0.3.186
10.0.0.226
10.0.1.186
10.0.2.130
10.0.2.141
10.0.0.2
10.0.0.137
10.0.3.146
10.0.3.210
10.0.3.106
10.0.3.170
10.0.3.185
10.0.3.130
10.0.1.218
10.0.0.142
10.0.1.22
10.0.2.121
10.0.1.201
10.0.0.34
10.0.1.214
10.0.0.138
10.0.2.102
10.0.2.94
10.0.2.126
10.0.2.106
10.0.3.66

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