Delta-net: Real-time Network Verification Using Atoms

Alex Horn¹, Ali Kheradmand² and Mukul R. Prasad¹

¹ Fujitsu Labs of America
² University of Illinois at Urbana-Champaign (Internship)
**Context:** “Network Verification”

- [Canini et al., NSDI 2012], **NICE**
- [Ball et al., PLDI 2014], **VeriCon**
- [Fayaz et al., NSDI 2016], **BUZZ**
- [Stoenescu et al., SIGCOMM 2016], **SymNet**
- [Fogel et al., NSDI 2015], **Batfish**
- [Lopes et al., NSDI 2015], **NoD in Z3**
- ...

- [Khurshid et al., NSDI 2013], **Veriflow**
- [Kazemian et al., NSDI 2013], **NetPlumber**
- [Yang and Lam, ICNP 2013], **AP Verifier**
- ...

**Goal:** detect network outages before they occur
Context: “Network Verification”

Goal: detect network outages before they occur

- [Canini et al., NSDI 2012], **NICE**
- [Ball et al., PLDI 2014], **VeriCon**
- [Fayaz et al., NSDI 2016], **BUZZ**
- [Stoenescu et al., SIGCOMM 2016], **SymNet**
- [Fogel et al., NSDI 2015], **Batfish**
- [Lopes et al., NSDI 2015], **NoD in Z3**
- ...

- [Khurshid et al., NSDI 2013], **Veriflow**
- [Kazemian et al., NSDI 2013], **NetPlumber**
- [Yang and Lam, ICNP 2013], **AP Verifier**
- ...

Undecidable

Decidable
Big Picture of Real-time Network Verification

Properties + Staged, or actual data plane

e.g., forwarding loop

Data Plane Checker

Forwarding Table

<table>
<thead>
<tr>
<th>Priority</th>
<th>IP Prefix</th>
<th>Action</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>192.8.0.10/31</td>
<td>Forward</td>
<td>123.9.1.12</td>
</tr>
</tbody>
</table>

Forwarding Rule

Errors

s0 —— z —— s1

s1 —— z —— s2

s2 —— z —— s3

s3
Taxonomy

Real-time Network Verification

- Packet Equivalence Classes
  - [Khurshid et al., NSDI 2013]
  - [Kazemian et al., NSDI 2013]
  - [Yang and Lam, ICNP 2013]

- Incremental Network Verification
  - [Khurshid et al., NSDI 2013]
Taxonomy

“Local” Similarity

Real-time Network Verification

Packet Equivalence Classes

Incremental Network Verification

New

[Khurshid et al., NSDI 2013]
[Kazemian et al., NSDI 2013]
[Yang and Lam, ICNP 2013]
Incremental Network Verification

Network

Control Plane

Data plane 2

Data plane 2

Data plane 2

Data plane 2

Data plane k

\(\triangle 1\)

\(\triangle 2\)
Incremental Network Verification

Control Plane

Network

Data plane 2

Data plane 2

Data plane 2

Data plane k

Possibly a lot of overlap
Problem: disruptive changes (e.g. outages) are still challenging to analyze in real-time.
Our Contribution: \( \Delta \) Delta-net

- Exploit similarity among forwarding behavior of packets through parts of the network, rather than its entirety.

- **Experiments:** 10-100x faster on network-wide use cases

Example next ...
Forwarding Graphs Per Equivalence Class

range of IP addresses

Highest priority

Lowest priority
Forwarding Graphs Per Equivalence Class
Forwarding Graphs Per Equivalence Class
Forwarding Graphs Per Equivalence Class
Our Contribution: \( \triangle \) Delta-net

“delta of deltas”

Rather than re-computing forwarding graphs ...

incrementally maintain a single edge-labelled graph; represents *all* packet flows.
Our Contribution: \( \triangle \) Delta-net

“delta of deltas”

Rather than re-computing forwarding graphs ...

incrementally maintain a single edge-labelled graph; represents *all* packet flows.

- Single graph data structure to answer reachability queries, exposed through a simple C++ API.
Our Contribution: \(\Delta\) Delta-net

Rather than re-computing forwarding graphs ...

incrementally maintain a single edge-labelled graph; represents all packet flows.

- Single graph data structure to answer reachability queries, exposed through a simple C++ API.
Atoms

“Complex Stuff = \sum \text{Simpler Stuff}”

**Example:** factorization into prime numbers, e.g.

\[ 1479 = 3 \times 17 \times 29. \]

For IP prefix based networks, **atoms**:

[Diagram showing IP address ranges with \(\alpha_1, \alpha_2, \alpha_3\) and range of IP addresses marked.]
Compactness of Atoms

More compact than a Patricia tree, e.g. consider $\alpha_0 = [0 : 10)$:

$[0 : 8) = 00^{***}$

$[0 : 8) \quad 0 \quad 1$

$[8 : 10) = 0100^*$
Atoms and Graph Transformation
Atoms and Graph Transformation
Atoms and Graph Transformation
Atoms and Graph Transformation

\[ \alpha_1, \alpha_2, \alpha_3, \alpha_4 \]

\[ r_1 \quad r_2 \quad r_3 \quad r_4 \]

\[ s_1 \quad s_2 \quad s_3 \quad s_4 \]
Atoms and Graph Transformation

Graph Transformation
High-level Flowchart

Start

Modify forwarding table

New atoms required?

Yes

Create new atoms

Refine precision of abstraction

No

Transform edge-labelled graph

Check properties

More modifications?

Yes

No

End
All-Pairs Reachability

- Essential for Datalog-style “what-if” queries:

```
1:  for k, i, j in V do                  ▶ Triple nested loop
2:    label[i, j] ← label[i, j] ∪ (label[i, k] ∩ label[k, j])
3:  end for
```

Adaptation of Floyd–Warshall Algorithm
All-Pairs Reachability

- Essential for Datalog-style "what-if" queries:

```
1: for k, i, j in V do ▶ Triple nested loop
2:   label[i, j] ← label[i, j] ∪ (label[i, k] ∩ label[k, j])
3: end for
```

Adaptation of Floyd–Warshall Algorithm
All-Pairs Reachability

- Essential for Datalog-style “what-if” queries:

```
1: for k, i, j in V do                        ▶ Triple nested loop
2:     label[i, j] ← label[i, j] ∪ (label[i, k] ∩ label[k, j])
3: end for
```

Adaptation of Floyd–Warshall Algorithm
Experimental Setup (2 Classes of Data Sets)

- Synthetic data sets similar to [Zeng et al., NSDI 2014]
  - Rocketfuel and Berkeley topologies from [Narayana et al., NSDI 2016]
  - IP prefixes from RouteViews project

- SDN-IP [Lin et al., SIGCOMM 2013] in ONOS
  - Globally deployed, ONOS flagship application
SDN-IP Experimental Setup

Diagram showing the setup of ONOS SDN-IP connected to Delta-net through OpenFlow. SDN receives link failures and sends iBGP to AS 1 and AS 3. BGP speakers connect to AS 1, AS 2, and AS 3 through eBGP. AS 1 has 78.88.0.0/17, AS 2 has 199.204.96.0/22, and AS 3 has 46.21.16.0/24.
### Data Sets

- Hundreds of million IP prefix rule insertions + removals

<table>
<thead>
<tr>
<th>Data set</th>
<th>Nodes</th>
<th>Max Links</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Berkeley</td>
<td>23</td>
<td>252</td>
<td>$25.6 \times 10^6$</td>
</tr>
<tr>
<td>INET</td>
<td>316</td>
<td>40,770</td>
<td>$249.5 \times 10^6$</td>
</tr>
<tr>
<td>RF 1755</td>
<td>87</td>
<td>2,308</td>
<td>$67.5 \times 10^6$</td>
</tr>
<tr>
<td>RF 3257</td>
<td>161</td>
<td>9,432</td>
<td>$149.0 \times 10^6$</td>
</tr>
<tr>
<td>RF 6461</td>
<td>138</td>
<td>8,140</td>
<td>$150.0 \times 10^6$</td>
</tr>
<tr>
<td>Airtel 1</td>
<td>68</td>
<td>260</td>
<td>$14.2 \times 10^6$</td>
</tr>
<tr>
<td>Airtel 2</td>
<td>68</td>
<td>260</td>
<td>$505.2 \times 10^6$</td>
</tr>
<tr>
<td>4Switch</td>
<td>12</td>
<td>16</td>
<td>$1.12 \times 10^6$</td>
</tr>
</tbody>
</table>

**Synthetic**

**SDN-IP**
Experiments: Measuring Rule Updates

- Find all forwarding loops introduced by a rule insertion.

![Graph showing CDF vs Processing time (Microseconds) for different datasets: Berkley, INET, RF-1755, RF-3257, RF-6461, 4Switch, Airtel1, Airtel2.](image-url)
Experiments: Measuring Rule Updates

- Find all forwarding loops introduced by a rule insertion.

Vast majority of rule updates analyzed in $<< 1$ ms
Experiments: Beyond Network Updates

What parts of the network are affected by link failures?

- Query proposed by [Khurshid et al., NSDI 2013].

<table>
<thead>
<tr>
<th>Data plane</th>
<th>Rules</th>
<th>Average query time (ms) Veriflow-RI</th>
<th>Delta-net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Berkeley</td>
<td>12,817,902</td>
<td>3,073.0</td>
<td>4.7</td>
</tr>
<tr>
<td>INET</td>
<td>124,733,556</td>
<td>29,117.5†</td>
<td>0.7</td>
</tr>
<tr>
<td>RF 1755</td>
<td>33,732,869</td>
<td>8,100.6</td>
<td>1.3</td>
</tr>
<tr>
<td>RF 3257</td>
<td>74,492,920</td>
<td>17,645.3†</td>
<td>1.0</td>
</tr>
<tr>
<td>RF 6461</td>
<td>75,005,738</td>
<td>17,594.5†</td>
<td>0.4</td>
</tr>
<tr>
<td>Airtel</td>
<td>38,100</td>
<td>4.5</td>
<td>0.04</td>
</tr>
<tr>
<td>4Switch</td>
<td>1,120,000</td>
<td>433.4</td>
<td>21.1</td>
</tr>
</tbody>
</table>

Summary: Delta-net can answer queries where Veriflow-RI times out.
Concluding Remarks

- Delta-net, a new real-time data plane checker.
  - Our research considers the “delta of deltas”.
  - Opens up new Datalog-style use cases, previously out of reach.
  - Data sets publically available now:
    https://github.com/delta-net/datasets

For questions and comments, contact: ahorn@us.fujitsu.com.
We are also looking for industry/academic partners, interns etc.

Future work:
- Parallelization
- Multi-range support
- Avoid space/time trade-offs, at what cost for query expressiveness?