Capturing Intent

Brighten Godfrey
CS 538 March 5, 2018
Two key goals

- Benchmark: Demonstrate concrete progress
- Feedback & discussion with your peers

Content

- What problem are you solving?
- Why has past work not addressed the problem?
- What is your approach for solving it?
- What are your preliminary results & progress?

Logistics

- 10 minutes total: 6:40 min presentation + 4 min discuss
- PechaKucha format: 20 slides x 20 seconds, auto-advance
Grand Challenge: Capturing Intent

We need networks that are

• Flexible
  - As adaptable and programmable as a well-designed software system

• Intuitive
  - Given a high level goal, the details are automated
Network Updates

Slides courtesy
Nate Foster
Abstractions for Network Update

Nate Foster
Mark Reitblatt

Jen Rexford
Cole Schlesinger
Dave Walker
Updates Happen

Network Updates
- Maintenance
- Failures
- ACL Updates

Desired Invariants
- No black-holes
- No loops
- No security violations
Network Updates Are Hard
Network Update Abstractions

Goal
• Tools for whole network update

Our Approach
• Develop update abstractions
• Endow them with strong semantics
• Engineer efficient implementations
Example: Distributed Access Control

Security Policy

<table>
<thead>
<tr>
<th>Src</th>
<th>Traffic</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web</td>
<td>Allow</td>
<td></td>
</tr>
<tr>
<td>Non-web</td>
<td>Drop</td>
<td></td>
</tr>
<tr>
<td>Any</td>
<td>Allow</td>
<td></td>
</tr>
</tbody>
</table>

Traffic
Naive Update

Security Policy

<table>
<thead>
<tr>
<th>Src</th>
<th>Traffic</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web</td>
<td>Allow</td>
<td></td>
</tr>
<tr>
<td>Non-web</td>
<td>Drop</td>
<td></td>
</tr>
<tr>
<td>Any</td>
<td>Allow</td>
<td></td>
</tr>
</tbody>
</table>

Order
F1
F2
F3
I

Traffic
Use an Abstraction!

Security Policy

UPDATE

✓

✓

✓
Q: What’s the right order to update?

<table>
<thead>
<tr>
<th>Src</th>
<th>Traffic</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web</td>
<td>Allow</td>
<td></td>
</tr>
<tr>
<td>Non-web</td>
<td>Drop</td>
<td></td>
</tr>
<tr>
<td>Any</td>
<td>Allow</td>
<td></td>
</tr>
</tbody>
</table>

Security Policy

Traffic flow:
- I ➔ F1, F2 ➔ F1 ➔ F2, F3 ➔ F3
A: Even atomic update doesn’t work!

Security Policy

<table>
<thead>
<tr>
<th>Src</th>
<th>Traffic</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web</td>
<td>Allow</td>
<td></td>
</tr>
<tr>
<td>Non-web</td>
<td>Drop</td>
<td></td>
</tr>
<tr>
<td>Any</td>
<td>Allow</td>
<td></td>
</tr>
</tbody>
</table>
Per-Packet Consistent Updates

<table>
<thead>
<tr>
<th>Src</th>
<th>Traffic</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web</td>
<td>Allow</td>
<td></td>
</tr>
<tr>
<td>Non-web</td>
<td>Drop</td>
<td></td>
</tr>
<tr>
<td>Any</td>
<td>Allow</td>
<td></td>
</tr>
</tbody>
</table>

Obeys policy:

Security Policy:

Each packet processed with old or new configuration, but not a mixture of the two.
Universal Property Preservation

**Theorem:** Per-packet consistent updates preserve all trace properties.

**Trace Property**
Any property of a *single* packet’s path through the network.

**Examples of Trace Properties:**
- Loop freedom, access control, waypointing ...

**Trace Property Verification Tools:**
- Anteater, Header Space Analysis, ConfigChecker ...
**Corollary**: To check an invariant, verify the old and new configurations.

**Verification Tools**
- Anteater [SIGCOMM ’11]
- Header Space Analysis [NSDI ’12]
- ConfigChecker [ICNP ’09]
MECHANISMS
2-Phase Update

Overview
• Runtime instruments configurations
• Edge rules stamp packets with version
• Forwarding rules match on version

Algorithm (2-Phase Update)
1. Install new rules on internal switches, leave old configuration in place
2. Install edge rules that stamp with the new version number
2-Phase Update in Action

Traffic
Optimized Mechanisms

Optimizations
- Extension: strictly adds paths
- Retraction: strictly removes paths
- Subset: affects small # of paths
- Topological: affects small # of switches

Runtime
- Automatically optimizes
- Power of using abstraction
IMPLEMENTATION & EVALUATION
Implementation

Runtime
- NOX Library
  - OpenFlow 1.0
  - 2.5k lines of Python
  - `update(config, topology)`
  - Uses VLAN tags for versions
  - Automatically applies optimizations

Verification Tool
- Checks OpenFlow configurations
- CTL specification language
- Uses NuSMV model checker
Evaluation

**Question:** How much extra rule space is required?

**Setup**
- Mininet VM

**Applications**
- Routing and Multicast

**Scenarios**
- Adding/removing hosts
- Adding/removing links
- Both at the same time

**Topologies**
- Fattree
- Small-world
- Waxman
Results: Routing Application

- Fattree
- Small-world
- Waxman

Worst-Case Rule Overhead

<table>
<thead>
<tr>
<th></th>
<th>Host</th>
<th>Link</th>
<th>Both</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subset</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Host
- Link
- Both

- Fattree
- Small-world
- Waxman
Propane
Propane: Key Concepts

Capabilities

- Ability to express network-wide goals
- Domain-specific language to describe policy conveniently
- Compiled to distributed control plane configurations (BGP)

Internal design

Don’t Mind the Gap: Bridging Network-wide Objectives and Device-level Configurations
Beckett, Mahajan, Millstein, Padhye, Walker
SIGCOMM 2016
since routers running BGP lack a global view of the network. This becomes even more challenging in the presence of failures.

The distributed nature of BGP makes setting preferences locally to achieve a network-wide routing policy difficult. This task is complicated by the need to set preferences on a per-device basis. However, the distributed nature of BGP makes setting preferences locally to achieve a network-wide routing policy difficult. This task is complicated by the need to set preferences on a per-device basis.

5.3 Failure-safety analysis

In this section, we present a simple approximation for removing many nodes and edges in the PGIR. We repeatedly apply the minimizations above until no further simplification is possible.

We use graph dominators [21] as a relatively cheap approximation for removing many nodes and edges in the PGIR. We repeatedly apply the minimizations above until no further simplification is possible.

Policy Automata

In the PGIR, a node is a valid topological path, leading to a path that satisfies the preference 2 policy, but which contains a loop.

For example, in Figure 5, the path W, A, C, D, E is removed because node W domines node A. Similarly, a node out domines a node in if it can not reach the end node. In the PGIR, a node out domines a node in if it is not reachable from the start node.

To make matters worse, the second preference for the path is never used. Thus, a path for the best possible route available after the policy is set to have a total ordering preference over all conditions, however, if the A–B link fails, then suddenly the second preference for the path can not be implemented in BGP in a way that is policy compliant. The compiler must determine which path is available in the network but not being used. Thus, a path for the best possible route available after the policy is set to have a total ordering preference over all conditions.
Propane: Discussion

How broad is the policy coverage?

Did they solve the configuration complexity problem?

Does the Propane system help detect errors?
Towards high-level abstractions

High-level

“Make the world a better place”

Dumb devices, smart controller

SDN control languages
(Frenetic, NetKAT)

SDN controllers: centralized abstraction
(e.g.: state database, consistent updates)

Network-wide declarative policy languages
(PGA, Propane, Merlin)

Low-level

Traditional device-by-device configs
(BGP, OSPF, VRRP, ECMP, …)

Propane
Beyond today’s research

OpenConfig

- Industry effort to abstract vendor-specific details
- Analogous to Propane’s ABGP

OpenStack Congress [https://wiki.openstack.org/wiki/Congress]

- “App A is only allowed to communicate with app B.”
- “Virtual machine owned by tenant A should always have a public network connection if tenant A is part of the group B.”
- “Virtual machine A should never be provisioned in a different geographic region than storage B.”
"Intent-based networking"

- Category of industry products aiming to help control and verify networks based on network-wide business goals

High-level abstractions for flexible programmability a grand challenge for networking

- What can people use? Who is doing the programming?
- Does different hardware change the abstraction?
- Can we carve out killer apps?