Minesweeper and Propane: Two Tools for Improving Network Reliability

Ryan Beckett
with
Aarti Gupta, Ratul Mahajan, Jitu Padhye, Todd Millstein, David Walker
Why configuration is hard

**Complexity**
- BGP, OSPF, RIP
- Route redistribution
- Protocol preference
- Metric conversions

**Low-level**
- Protocol parameters
- Interface metrics
- Route maps
- Access control lists

**Distributed**
- Device configurations
- 100,000s of lines

**Failures**
- Link failures common
- Router failures possible

**Bugs**
Misconfigurations are common

**South Africa: FNB solves crippling connectivity issues**

July 25, 2016 • Johannesburg, South Africa, The Star

**BGP errors are to blame for Monday’s Twitter outage, not DDoS attacks**

No, your toaster didn’t kill Twitter; an engineer did.

**Unions want Southwest CEO removed after IT outage**

**Massive route leak causes Internet slowdown**

Posted by Andre Toonk • June 12, 2015 • BGP instability • No Comments

**Microsoft: misconfigured network device led to Azure outage**

30 July 2012 • By Yevgeny Sverdlik

**Router Crashes Trigger Major Southwest IT System Failure**

By Chris Preimesberger | July 21, 2016

**BlackBerry outage could cost RIM $100 million**

**Xbox Live outage caused by network configuration problem**

By TONI BISARA on April 15, 2013 at 5:27 am
Misconfigurations are expensive

- Lack of automation causes outages and breaches. 20% of organizations experienced a security breach, 48% had an application outage and 42% had a network outage as a result of a misconfiguration caused by a manual security-related process.

Average of 105 server racks. The study concluded that the average duration of a single data center outage was 95 minutes equating to a cost of $740,357.
Minesweeper
Find bugs in **legacy** networks

Propane
High-level design of **new** networks
Minesweeper
Current Approach: Heuristics

Configurations

String Matching

- `ospf interface int2_1 metric 1`
- `interface int2_10 10.0.0.1/24`
- `ospf redistribute connected metric 10`
- `prefix-list PL_C 10.0.0.0/24`

Potential Issues

Examples: RCC, SolarWinds, HPNA / TruControl, NetDoctor
Heuristics: Limitations

- Can miss many bugs
- Can report false positives
- Hard to test **forwarding behavior**

String Matching

```
ospf interface int2_1 metric 1
interface int2_10 10.0.0.1/24
ospf redistribute connected metric 10
prefix-list PL_C 10.0.0.0/24
```
Current Approach: Simulation

Control Plane

Data Plane

Simulate

Examples: Batfish, C-BGP

Traceroute
Inspect FIB
Data plane analysis
Simulation: Limitations

Cannot test for all routing messages

Cannot test for all possible link failures
Overview

We present a new network analysis tool called **Minesweeper**:

- Can check many properties for all external routing messages and for all link failures
- Encodes the network as a collection of Logical constraints and leverages off-the-shelf constraint solvers
- [https://batfish.github.io/minesweeper/](https://batfish.github.io/minesweeper/)
Workflow

1. Vendor-Specific Configs
   - Cisco
   - Arista
   - Juniper

2. Vendor-Independent Format
   - Interfaces:
     - Ethernet0/0: {
       - InterfaceCost: 1,
       - importPolicy: “PEER_IN”,
       - ...
     }

3. Constraint Encoding
   - 192.0.0.0 ≤ out.prefix
   - out.prefix ≤ 192.1.0.0
   - best.valid ⇒ out.lp = 120
   - best.valid ⇒ out.ad = 20

4. Output
   - Parse
   - Encode
   - Query
   - Solve
Constraint Encoding

Protocol View
Constraint Encoding

Redistribution Connected to BGP

Constraint view for R1 BGP

protocol message
Constraint Encoding

Symbolic message

- valid: 1 bit
- prefix: \([0,2^{32}]\)
- prefixLen: \([0,2^5]\)
- adminDist: \([0,2^8]\)
- localPref: \([0,2^{32}]\)
- metric: \([0,2^{32}]\)
- med: \([0,2^{32}]\)
- ospfType: \([0,2^2]\)
Constraint Encoding

R1 BGP import filter from R2

`ip prefix_list L deny 192.168.0.0/16 le 32
ip prefix_list L allow
route-map R1_Import_From_R2 10
match ip address prefix-list L
set local-preference 120`

Logical Constraints

```
if out4.valid ∧ failed_R1_R2 = 0
  then
    if ¬ (FBM(out4.prefix, 192.168.0.0, 16) ∧ 16 ≤ out4.prefixLen ≤ 32)
      then
        in4.valid = true
        in4.lp = 120
        in4.ad = out4.ad
        in4.prefix = out4.prefix
        in4.metric = out4.metric
        in4.prefixLen = out4.prefixLen
      else
        in4.valid = false
    else
      in4.valid = false
```
Demo: Topology

Host firewall (iptables)
IGP (OSPF)
EGP (eBGP and iBGP)
Router ACLs
Example Properties

"Can router X always reach router Y"
Example Properties

“Can router X always reach router Y”

“Do all routers in a pod have equal length paths to a destination port?”
Example Properties

- Can router X always reach router Y?
- Do all routers in a pod have equal length paths to a destination port?
- Can my network ever have loops?
Example Properties

- Can router X always reach router Y?
- Do all routers in a pod have equal length paths to a destination port?
- Can my network ever have loops?
- Are multiple paths treated equally?
Example Properties

- Can router X always reach router Y?
- Do all routers in a pod have equal length paths to a destination port?
- Can my network ever have loops?
- Are multiple paths treated equally?
- Do two routers serve equal roles?
# Supported Features

<table>
<thead>
<tr>
<th>Features</th>
<th>Implemented</th>
<th>Continued...</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSPF Intra-area</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>OSPF Inter-area</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>eBGP Local-pref</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>eBGP Communities</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>eBGP MEDs</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>eBGP Path Prepending</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>eBGP Aggregation</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>iBGP</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Route Reflectors</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Static Routes</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Route Redistribution</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Multipath Routing</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Access Control Lists</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>IPV6</td>
<td>✗</td>
<td></td>
</tr>
</tbody>
</table>
Evaluation: Bug Finding

Ran Minesweeper on **152 legacy data center networks**

- Management interface reachability
  - Found 67 violations of the property
  - Each required a specific environment
  - Example: BGP peer sends /32 with length 2

- Local equivalence of routers
  - Found 29 violations
  - Many caused by simple copy-paste errors
  - Example: ACL has missing entry
Evaluation: Scalability

Management interface reachability

Local equivalence of routers (For all n comparisons)
Evaluation: Scalability

The graph illustrates the verification time (in milliseconds) for different scenarios of scalability, with the number of routers (Pods) ranging from 5 to 405. The scenarios include:

- No Blackholes
- Multipath Consistency
- Local Consistency
- Single-ToR Reachability
- All-Tor Reachability
- Single-Tor Bounded Length
- All-Tor Bounded Length
- Equal Length Pod

The y-axis represents the verification time on a logarithmic scale, while the x-axis shows the number of routers (Pods) with specific counts indicated in parentheses.
Conclusion

**Minesweeper** is a general control plane verification tool

- Checks a wide variety of properties for all packets, all possible environments, and all combinations of k-failures
- Encodes the network as a hardware circuit and leverages modern off-the-shelf theorem provers
- Can find bugs in many real networks

[https://batfish.github.io/minesweeper/](https://batfish.github.io/minesweeper/)
Minesweeper
Find bugs in legacy networks

Propane
High-level design of new networks
Propane
Fundamental Tradeoff?

**Configuration**

- **Distributed**
  - OSPF
  - RIP
  - BGP
  - Scalability
  - Robustness
  - Complexity

- **Centralized**
  - Ideal

**Control Mechanism**

- **Distributed**

- **Centralized**

**SDN**

- Scalability
  - Robustness
  - Complexity
Propane Overview

Topology

Propane Policy

Compiler

BGP Configurations
Propane System

(1) Language for expressing **network-wide policy objectives** with:

- Path **constraints** and **preferences**
- Uniform abstractions for **intra-** and **inter-domain routing**
Propane System

(2) Compiler for generating BGP configurations

- Guarantees policy-compliance for all possible failures
Example: A DC network

Policy Objectives
• Local prefixes reachable only internally
• Global prefixes reachable externally
• Aggregate global prefixes as GP
• Prefer leaving through Peer₁ over Peer₂
• Prevent transit traffic between peers

Demo

!
define pfx1 = 1.0.0.0/24
define pfx2 = 1.0.1.0/24
define pfx3 = 2.0.0.0/24
define pfx4 = 2.0.1.0/24
define local_prefixes = (pfx3 or pfx4)
define peer = {IDAT, CORE}

define basic_routing = {
    pfx1 => end(A),
    pfx2 => end(B),
    pfx3 => end(E),
    pfx4 => end(F),
    true => end(out)
}
define main = basic_routing
How Compilation Works

Network Policy → Propane FE → RIR

Topology → PGIR → ABGP

Rewriting Rules + Well-Formedness
BGP Control Graph
Safety Analysis
Config Generation + Minimization

Quagga → Cisco → ...

Propane Compiler

- Generates Cisco and Quagga configs
- Includes a number of other analyses
  - Unused backup paths
  - Possible reachability issues
  - Aggregation-induced black holes
  - Unused prefixes / aggregates
- Can enable / disable MEDs, prepending, …
Evaluation

- **Language expressiveness**
  - Translated configurations from a large cloud provider
  - Policy described in English documents
  - Both data center and backbone networks

- **Compiler performance**
  - Used cloud provider’s routing policy
  - Scaled the size of backbone and data center topologies
Language Expressiveness

Not counting prefix / peer definitions

- Data center policy: ~30 lines of Propane
- Backbone policy: ~50 lines of Propane
- Actual networks: ~1000s lines of Configuration
Compiler Performance

Data center (< 9 min)  

Backbone (< 3 min)
Conclusion

High-level language
- **Centralized** network programmability
- Constraints specify preferred paths and backup paths
- Core policy in 30-50 lines of Propane vs 1000s of config

Compiler
- **Distributed** implementation via BGP
- Static analysis guarantees policy compliance for **all failures**
- **Scales** to many large network topologies

http://www.propane-lang.org
Minesweeper
Find bugs in legacy networks

Propane
High-level design of new networks

rbeckett@princeton.edu