Diagnosing Distributed Routing Configurations Using Sequential Program Analysis

Rulan Yang¹, Xing Fang¹, Lizhao You¹, Qiao Xiang¹, Hanyang Shao¹, Gao Han¹, Ziyi Wang¹, Jiwu Shu¹, ², Linghe Kong³

¹ Xiamen University, ² Minjiang University, ³ Shanghai Jiao Tong University

2023/6/29
Router configuration errors are common

Microsoft: misconfigured network device led to Azure outage

Level 3 blames huge network outage on human error

Facebook is back after router misconfiguration caused largest-ever outage

Due to a router misconfiguration, Cloudflare suffers short outage on Friday

"configuration error" as the root cause of its nationwide network outage on Tuesday.
What does network verification do?

• Detecting if configurations are erroneous:

  Configurations

  Results

  • reachability?
  • waypoint?
  • ...

3
What does network verification do?

- Detecting if configurations are erroneous: **answer the YES/NO question**

<table>
<thead>
<tr>
<th>Verification tools</th>
<th>Network model they generate</th>
<th>The way they verify properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation-based tools (e.g., Batfish)</td>
<td>RIB/FIBs (i.e., The data plane)</td>
<td>Data plane verification → Compute and verify the data plane</td>
</tr>
<tr>
<td>Graph-based tools (e.g., ARC)</td>
<td>Abstraction graph</td>
<td>Compute graph invariants</td>
</tr>
<tr>
<td>SMT-based tools (e.g., Minesweeper)</td>
<td>Logic formula</td>
<td>Check the satisfiability of the formula</td>
</tr>
</tbody>
</table>

- reachability ✓
- waypoint ✗
- ...
Why diagnosis is hard?

Network is a (complex) distributed system

Configuration errors on one router will lead to the property violation of other routers

Property: S-B-D

Reality: S reaches D through “S-A-D”
Verification is not enough

- Verification can tell us:

  - How do the configuration errors contribute to the results?

The error flow: s->a->d

The error result: false

However, they cannot diagnose:

Batfish

Minesweeper

The counter-example:
Key insight: understanding the distributed routing as a sequential program

- Distributed routing processes

Distributed routing is a program consisting of the three parts:

1. Receive routes
2. Route selection
3. Send routes

- Distributed network execution

receive ([D])
select ([D])
send ([A, D])

receive ([A, D])
receive ([B, A, D])

receive ([A, D])
receive ([D])
select ([A, D])
send ([B, A, D])

receive ([A, D])
receive ([D])

send ([D])
Key insight: understanding the distributed routing as a sequential program

- Distributed network execution → The causal relationship graph (CRG)

Using (sender, receiver) and timestamp to capture the causal relationship
Key insight: understanding the distributed routing as a sequential program

- Transforming the distributed program into a linear trace
Analyzing the trace

• Configurations are input to the program

Verification executes the configurations

Diagnosis reasons about the execution

- Network properties are assertions that the program need to hold
Analyzing the trace

**Input: Configurations**

- send ([D])@D
- receive ([D])@A
- select ([D])@A
- send ([A, D])@A
- receive ([D])@B
- receive ([A, D])@B
- select ([A, D])@B
- send ([B, A, D])@B
- receive ([A, D])@S
- receive ([B, A, D])@S

... 

**assert** Property == true

**executing**

- How erroneous paths were computed?

**reasoning**

- For **observable errors** (ones with traces that generate erroneous paths):

  We diagnose errors by analyzing the **data dependency**

```
[S, A, D]  [B, A, D]
[A, D]
[D]  
[S]  [A, D]  [B, A, D]  
```

12
Data-flow analysis using MUC&EI

• Get the data dependency with explanation:

Minimal Unsatisfiable Core (MUC):
A minimal set of statements leading to the error

Error Invariants:
Key program states of statements

- D sent [D] to A and B
- A received [D], selected it and sent it to S and B
- B received [D] and [A, D], selected [A, D] since that B prefers A
- S received [A, D] and [B, A, D], selected [A, D] since it has the shorter as-path length
Data-flow analysis has limitations

- **Unobservable errors**  (have exponential searching space)
- **Latent errors**  (need repeated DF diagnosis)

**Property**: $S \text{-} * \text{-} D$

**Reality**: There is no path from $S$ to $D$

**The trace**: $D(\text{Static})$

S drops routes from both A and B

D doesn’t redistribute the route to BGP process
Control-flow analysis using symbolic execution

Collect more traces with their path conditions to analyze:

A path condition is a predicate that a symbolic execution trace needs to hold

- **How to diagnose?**
  
  Differentiate these traces

Program  

Correct execution path  Incorrect execution path  Errors
Control-flow analysis using symbolic execution

Collect more traces with their path conditions to analyze:

➤ Pruning: Selective symbolic execution

Selectively applying symbolic execution on these branches:
- route selection
- export/import for target routes
- peer connection establishment

Program

... ... ...

Program
Control-flow analysis using symbolic execution

Collect more traces with their path conditions to analyze:

Program

Data plane simulation

Conditions are attached as symbols with route announcements during the data plane simulation

Errors are configurations that deviate from those path conditions!
Diagnosis results

**Property:** S-*-D

**Reality:** There is no path from S to D

**Error path conditions:**
- Export(Static→BGP, T1)) ∧
- Import(S←A, (T1, A, [D]))

**Error 1:** The route-map in S (line 19, 22-24) filtered routes from A

**Error configuration lines:**
- line22: route-map in_s deny 10
- line23: match ip address prefix-list in_s
- line24: route-map in_s permit 20
- line19: ip prefix-list in_s seq 10 permit 10.0.1.0/24 le 32

**Error 2:** There was no redistribution command in node D
Evaluation

Evaluate Scalpel on synthetic configurations

• Routing protocols: BGP, static route
• Protocol features: redistribution, route filters
• Networks size: \( O(10) \sim O(100) \) routers
• \( O(100) \) lines of configuration per router
• Errors injection: wrong-local-pref error, propagation-deny error, origin-deny error
Evaluation: observable error diagnosis

For observable errors, all error traces can be generated < 4s

For observable errors, we diagnose all errors within several minutes

![CDF of CRG generation time](image1)

![CDF of MUC and EI computation time](image2)
Evaluation: unobservable error diagnosis

For unobservable errors, we diagnose 74% of them within 50ms

<table>
<thead>
<tr>
<th>Network</th>
<th>#Errors</th>
<th>#RE</th>
<th>#FP</th>
<th>#FN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airte</td>
<td>(1,2)</td>
<td>(1,2)</td>
<td>(0,0)</td>
<td>(0,0)</td>
</tr>
<tr>
<td>Arnes</td>
<td>(1,2)</td>
<td>(1,4)</td>
<td>(0,2)</td>
<td>(0,0)</td>
</tr>
<tr>
<td>Bics</td>
<td>(2,2)</td>
<td>(3,3)</td>
<td>(1,1)</td>
<td>(0,0)</td>
</tr>
<tr>
<td>Cogentco</td>
<td>(2,2)</td>
<td>(10,10)</td>
<td>(8,8)</td>
<td>(0,0)</td>
</tr>
<tr>
<td>Columbus</td>
<td>(2,2)</td>
<td>(1,2)</td>
<td>(0,0)</td>
<td>(1,0)</td>
</tr>
<tr>
<td>Latnet</td>
<td>(2,2)</td>
<td>(2,2)</td>
<td>(0,0)</td>
<td>(0,0)</td>
</tr>
<tr>
<td>UsCarrier</td>
<td>(1,2)</td>
<td>(1,2)</td>
<td>(0,0)</td>
<td>(0,0)</td>
</tr>
</tbody>
</table>

Table: Diagnosis results on unobservable errors
Summary

• We take the first step to apply program analysis to distributed router configuration diagnosis.

• We design Scalpel, an automated router configuration diagnosis tool.

• We implement Scalpel and demonstrate the capability of Scalpel.
Open questions

• **Non-deterministic** execution (e.g., leveraging Hoyan’s design)
• Supporting **more protocols** (e.g., link state protocols)
• Scaling control-flow analysis for **large networks** (e.g., slicing)
• Automatic **repair** based on diagnosis (e.g., constraint-based repair)

*SNGroup site: http://sngroup.tech/*
*Contact us: sngroup.xmu@outlook.com*