Symbolic Router Execution

Peng Zhang, Dan Wang

Aaron Gember-Jacobson
Space Exploration
Space Exploration

network header & failure

Network Verifier
Which hosts are reachable from here?
Which hosts are reachable from here?

<table>
<thead>
<tr>
<th>Dst</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.0.0/8</td>
<td>-</td>
</tr>
<tr>
<td>2.0.0.0/8</td>
<td>B</td>
</tr>
<tr>
<td>3.0.0.0/8</td>
<td>B</td>
</tr>
<tr>
<td>4.0.0.0/8</td>
<td>D</td>
</tr>
</tbody>
</table>

Header space
Is 3.0.0.3 reachable 99.9% of the time?
Is 3.0.0.3 reachable 99.9% of the time?
Is 3.0.0.3 reachable 99.9% of the time?

Probabilistic failures

<table>
<thead>
<tr>
<th>Dst</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.0.0/8</td>
<td>-</td>
</tr>
<tr>
<td>2.0.0.0/8</td>
<td>B</td>
</tr>
<tr>
<td>3.0.0.0/8</td>
<td>D</td>
</tr>
<tr>
<td>4.0.0.0/8</td>
<td>D</td>
</tr>
</tbody>
</table>
How many failures can occur and still reach 3.0.0.3?

Deterministic failures
The first task required exploring the entire header space for a single failure scenario.
The second task required exploring a single header for the entire failure space.
The second task required exploring a single header for the entire failure space

...and either probabilistic or deterministic failures
Some management tasks require exploring the entire header space for the entire failure space.
Some management tasks require exploring the entire header space for the entire failure space

...and either probabilistic or deterministic failures
Some management tasks require exploring the product space

1. Verify a proposed configuration change has no side-effects
Some management tasks require exploring the product space

1. Verify a proposed configuration change has no side-effects

access-list inbound
+ deny 6.0.0.0/8
permit any
Some management tasks require exploring the product space

1. Verify a proposed configuration change has no side-effects

access-list inbound
+ deny 6.0.0.0/8
permit any
Some management tasks require exploring the product space

1️⃣ Verify a proposed configuration change has no side-effects

Access-list inbound
+ deny 6.0.0.0/8
permit any
Some management tasks require exploring the product space

① Verify a proposed configuration change has no side-effects

② Mine network requirements (e.g., Config2Spec)
Some management tasks require exploring the product space

① Verify a proposed configuration change has no side-effects

② Mine network requirements (e.g., Config2Spec)
Some management tasks require exploring the product space

① Verify a proposed configuration change has no side-effects

② Mine network requirements (e.g., Config2Spec)
Some management tasks require exploring the product space

① Verify a proposed configuration change has no side-effects

② Mine network requirements (e.g., Config2Spec)
Can we use the verifiers we already have?
Packet Equivalence Classes

Deny route 192.0.0.0/2 out

192.0.0.0/2
128.0.0.0/1
Packet Equivalence Classes

Deny route 192.0.0.0/2 out

192.0.0.0/2 → B
128.0.0.0/1 → C

192.0.0.0/2 → C
128.0.0.0/1 → C
Packet Equivalence Classes

192.0.0.0/2 → B
128.0.0.0/1 → C

Deny route 192.0.0.0/2 out

Dst: 192.0.0.1

192.0.0.0/2 → C
128.0.0.0/1 → C
Packet Equivalence Classes

Deny route 192.0.0.0/2 out

192.0.0.0/2 → B
128.0.0.0/1 → C

Dst: 128.0.0.1

192.0.0.0/2 → C
128.0.0.0/1 → C

192.0.0.0/2

128.0.0.0/2
Packet Equivalence Classes

192.0.0.0/1 $\rightarrow$ B
128.0.0.0/1 $\rightarrow$ B

192.0.0.0/2 $\rightarrow$ C
128.0.0.0/1 $\rightarrow$ C

Deny route 192.0.0.0/2 out
Divide headers into PECs

Efficient header space exploration

Inefficient failure space exploration
Divide failures into FECs

- Config2Spec
- Tiramisu
- DNA
- Minesweeper
- Bagpipe
- Origami
- Hoyan
- ARC
- NV
- ProbNV
- NetDice

Efficient failure space exploration

Inefficient failure space exploration

Divide headers into PECs

- Bagpipe
- Plankton
- ERA
- DNA
- ShapeShifter
Failure Equivalence Classes

A

Deny route 192.0.0.0/2 out

B

C

192.0.0.0/2
Failure Equivalence Classes

Deny route 192.0.0.0/2 out

192.0.0.0/2 → B

192.0.0.0/2 → C
Failure Equivalence Classes

Deny route 192.0.0.0/2 out

192.0.0.0/2 → B
192.0.0.0/2 → C

192.0.0.0/2
Failure Equivalence Classes

Deny route 192.0.0.0/2 out

192.0.0.0/2 → C
Failure Equivalence Classes

Deny route 192.0.0.0/2 out

192.0.0.0/2
Deny route 192.0.0.0/2 out
Failure Equivalence Classes

Deny route 192.0.0.0/2 out

192.0.0.0/2
Failure Equivalence Classes

Deny route 192.0.0.0/2 out

128.0.0.0/1

128.0.0.0/2

A
B
C

A
B
C

A
B
C

A
B
C

A
B
C

A
B
C

A
B
C

A
B
C

A
B
C

A
B
C

A
B
C

A
B
C

A
B
C
Divide failures into FECs

Divide headers into PECs

Efficient failure space exploration

Inefficient header space exploration

Config2Spec
Tiramisu
Minesweeper
Bagpipe
NetDice
ProbNV

ARC
Hoyan
Origami
DNA

Bagpipe
Plankton
ERA
ShapeShifter
Scales to the product space of headers and failures by computing PFECs
Scales to the product space of headers and failures by computing PFECs.

Generalizes to deterministic and probabilistic failures by delaying binding to a failure model.
Basic idea: Symbolic execution

```python
x = input("Integer:")
if (x > 0):
    print("Positive")
else:
    if (x < 0):
        print("Negative")
    else:
        print("Zero")
```
Basic idea: Symbolic execution

```python
x = ?
if (x > 0):
    print("Positive")
else:
    if (x < 0):
        print("Negative")
    else:
        print("Zero")
```
Basic idea: Symbolic execution

? > 0
x = ?
if (x > 0):
    print("Positive")

? ≤ 0
else:
    if (x < 0):
        print("Negative")
    else:
        print("Zero")
Basic idea: Symbolic execution

```python
x = ?
if (x > 0):
    print("Positive")
else:
    if (x < 0):
        print("Negative")
    else:
        print("Zero")
```
Symbolic execution of control and data planes
Symbolic execution of control and data planes

Available links

Configurations
Symbolic execution of control and data planes

Configurations

Available links

Symbolic routing

Data Plane
Symbolic execution of control and data planes

Configurations

Available links

Symbolic routing

Symbolic RIBs

Data Plane
Symbolic execution of control and data planes

Configurations

Available links

Symbolic routing

Symbolic RIBs

Data Plane

Symbolic packets
Symbolic execution of control and data planes

Symbolic links

Symbolic routing

Symbolic RIBs

Symbolic packets

Symbolic packet forwarding

PFECs

Configurations

A

B

C

$p_1$

$p_2$

$l_{AB}$

$l_{AB}$

$l_{AC}$

$l_{AC}$

$l_{BC}$

0

1
Symbolic execution of control and data planes

Configurations

Symbolic links

Symbolic routing

Symbolic RIBs

Symbolic packets

Symbolic packet forwarding

PFECs

Symbolic execution of control and data planes
How many failures can occur and A still reach 128.0.0.0/2?
How many failures can occur and A still reach 128.0.0.0/2?
How many failures can occur and A still reach 128.0.0.0/2?
How many failures can occur and A still reach 128.0.0.0/2?

Failure tolerance = (Len of shortest path to terminal 0) - 1
How many failures can occur and A still reach 128.0.0.0/2?

Failure tolerance = $2 - 1$

Deny route 192.0.0.0/2 out
What is the probability A can reach 128.0.0.0/2?

Probability = Weighted sum of all paths to terminal 1
What is the probability A can reach 128.0.0.0/2?

Probability =

0.99 * (0.01 * 0.99 + 0.99) + 0.01 * 0.99
Experiment: compute all-pairs reachability

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Bics</th>
<th>Columbus</th>
<th>US Carrier</th>
<th>Fattree (20 nodes)</th>
<th>Fattree (80 nodes)</th>
<th>SRE</th>
</tr>
</thead>
</table>
Experiment: compute all-pairs reachability
Experiment: compute all-pairs reachability

- Batfish
- Minesweeper
- Tiramisu
- SRE

Time (s)

- Bics
- Columbus
- US Carrier
- Fattree (20 nodes)
- Fattree (80 nodes)
Experiment: compute all-pairs reachability
Experiment: compute all-pairs reachability
Scales to the product space of headers and failures by computing PFECs.

Generalizes to deterministic and probabilistic failures by delaying binding to a failure model.