

# Guided Exploration of Control-Plane Routing States

Tibor Schneider, *Jean Mégret*, Laurent Vanbever

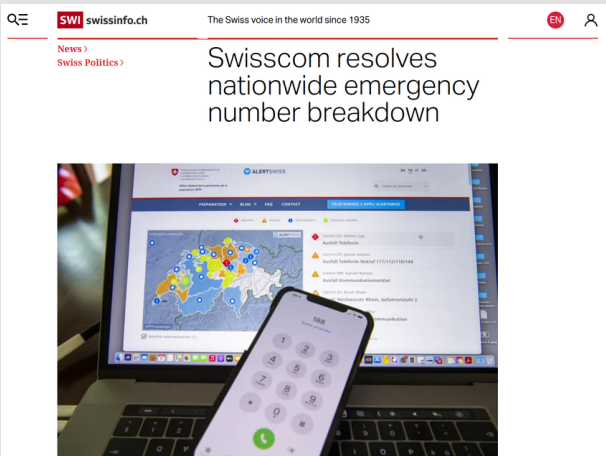
*ETH Zürich - Networked Systems Group*

24th of September, 2025

What do you call a team of network engineers?

... an outage.

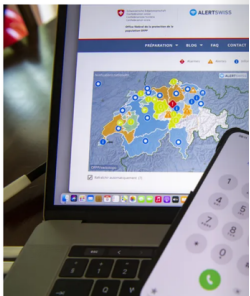
... an outage



▲ This is the second such national breakdown in 18 months. Keystone / Martial Trezzini

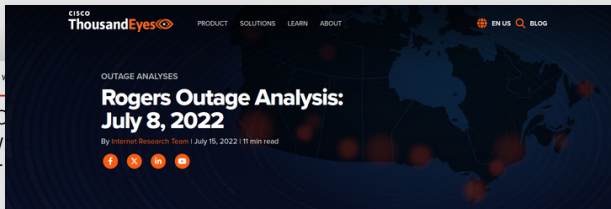
Emergency telephone numbers are working again in Switzerland after a technical failure caused a

... many outages



▲ This is the second such national breakdown in 18 months. Key

Emergency teleph  
Switzerland after a



## SUMMARY

Beginning at 08:44 UTC on July 8th, 2022, Rogers Communications, one of the largest Internet service providers in Canada, experienced a significant disruption that rendered its network unavailable for long periods of time over the course of approximately 24 hours. The following brief analysis is designed to provide insight into the incident for Infrastructure and Operations professionals, as well as the interested general public.

## Summary of Findings

Based on ThousandEyes' broad visibility into Internet networks, we observed the following:

- The incident appeared to be triggered by the withdrawal of a large number of Rogers' prefixes, rendering their network unreachable across the global Internet. However, behavior observed in their network around this time suggests that the withdrawal of external BGP routes may have been precipitated by internal routing issues.
- Because Rogers continued to announce at least some of its customers prefixes, traffic continued to flow into its network but could not route properly, leading to the traffic terminating in their network. This behavior suggests an internal network control plane issue may have been an underlying factor in the incident rather than strictly due to external BGP route withdrawals, which has been widely reported.
- Shortly after the initial withdrawal of routes, Rogers was able to reinstate some prefix advertisements, which restored availability for parts of its service; however, the same withdrawal (and eventual readvertisement) behavior was repeated several more times causing varying levels of impact until the incident was fully resolved.
- The scale and overall duration of the incident suggests an internal control plane issue that took time to be discovered given that the initial route withdrawal was repeated multiple times throughout the incident, possibly due to the underlying cause of the incident getting inadvertently retrIGGERED.

... many outages

CISCO  
ThousandEyes

PRODUCTSOLUTIONSLEARNABOUT

EN US

BLOG

NEWS

Just InFor YouPoliticsWorldBusinessMore

Q


SW

New  
Swis

What caused Optus's nationwide outage, and how long was it down for? Here's what we know

By Tom WilliamsMobile Phones

Wed 8 Nov 2023

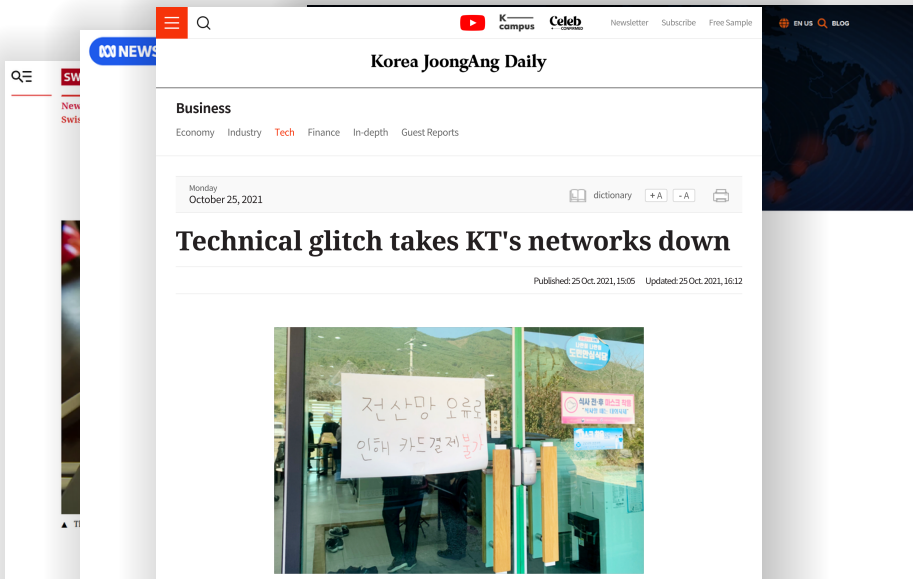


The Optus network is recovering after a major nationwide outage on Wednesday morning. (AAP: Dan Peled)

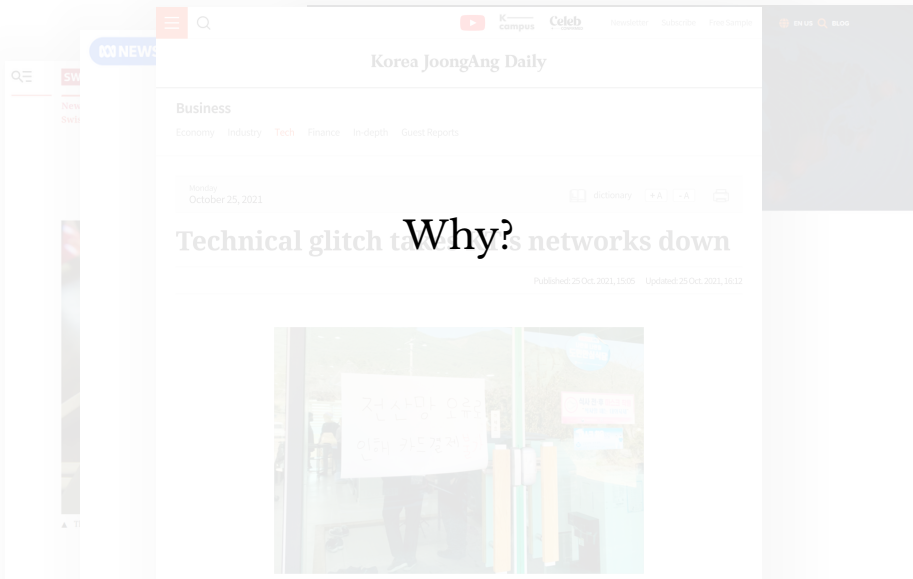
Millions of customers of Australia's second-largest telecommunications company, Optus, were affected by an outage of mobile phone and internet services on Wednesday morning.

The outage also impacted hundreds of thousands of

Large-scale Internet outages are *too* common.

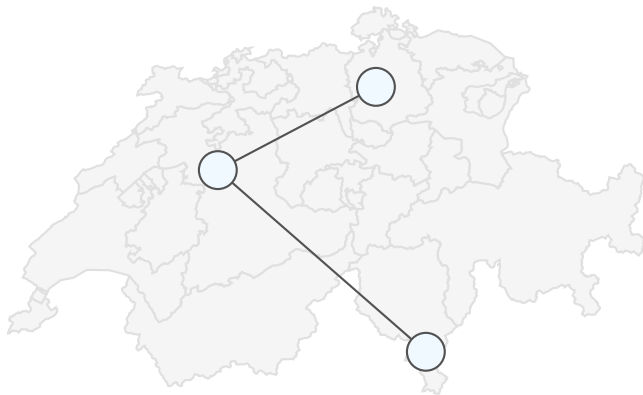


Large-scale Internet outages are *too* common.

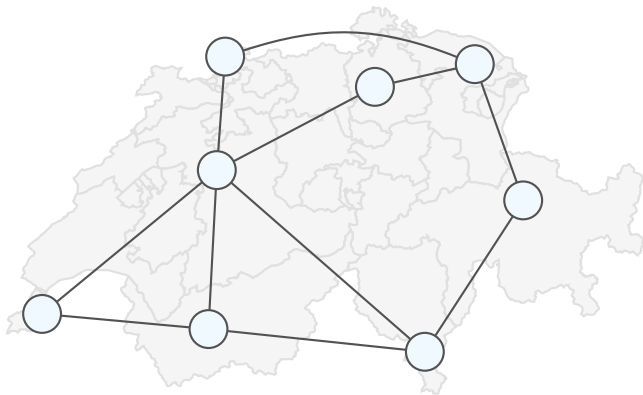




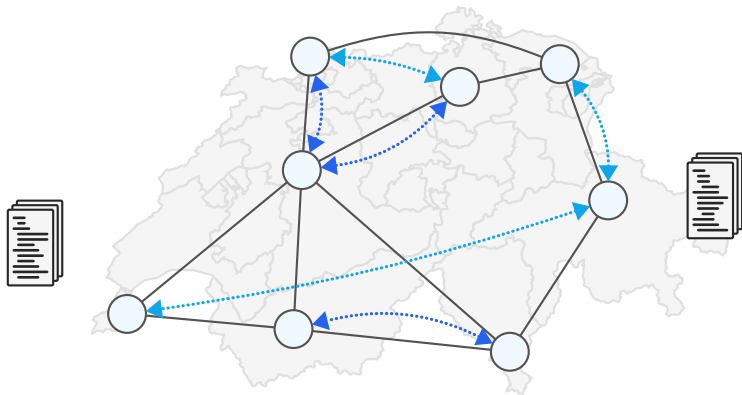
Networks are complex systems, they ...



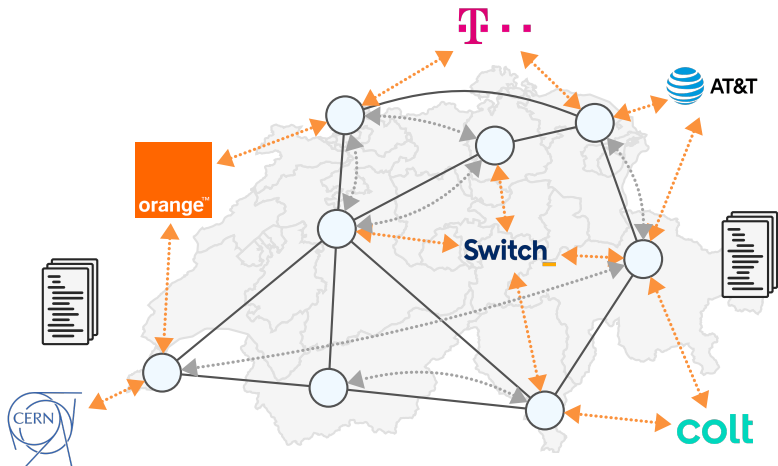
...span across thousands of routers and edges ...



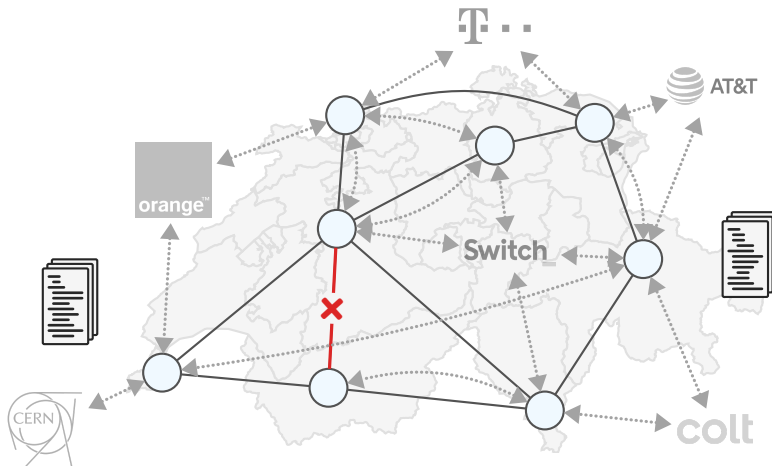
...run complex configurations...

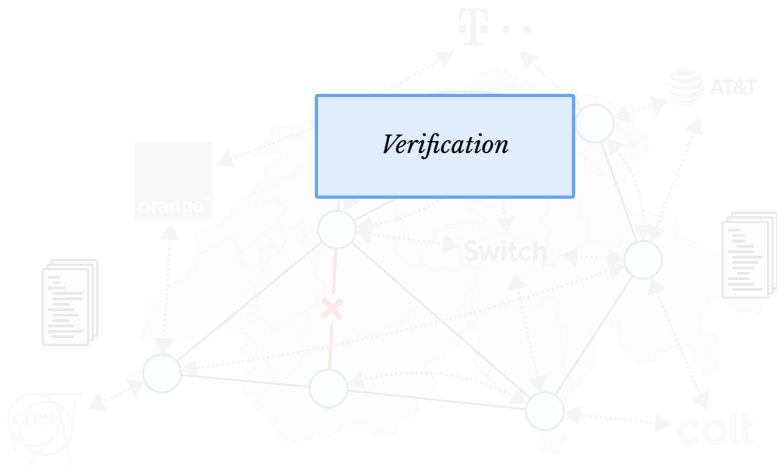


...depend on external announcements...



...and are exposed to device failures.

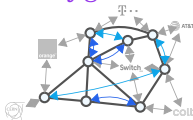




## *Verification*

Does the configuration satisfy the operators' specification?

### *Network & Configuration*



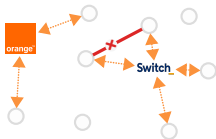
### *Specification*

reachability

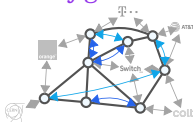
## *Verification*

Does the configuration satisfy the operators' specification?

### Environment



### *Network & Configuration*



### *Specification*

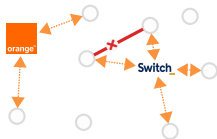
reachability



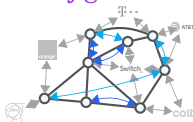
## *Verification*

Does the configuration satisfy the operators' specification?

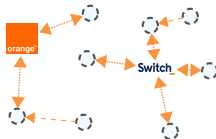
Environment



*Network &  
Configuration*



Routing State



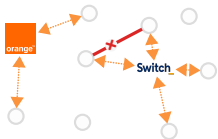
*Specification*

reachability

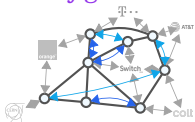
## *Verification*

Does the configuration satisfy the operators' specification?

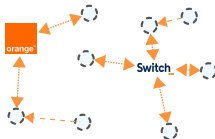
Environment



*Network &  
Configuration*



Routing State



*Specification*

reachability

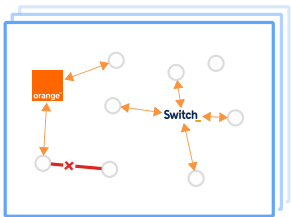
Result



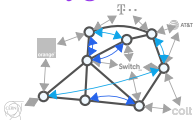
## Verification

Does the configuration satisfy the operators' specification  
*in all environments*?

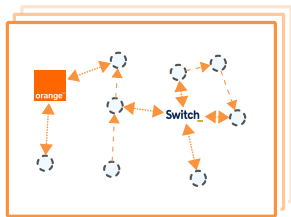
### Environments



### Network & Configuration



### Routing States



### Specification

reachability



*Verification*

Does the configuration satisfy the operators' specification  
*in all environments*?

## *Verification*

Does the configuration satisfy the operators' specification  
*in all environments?*

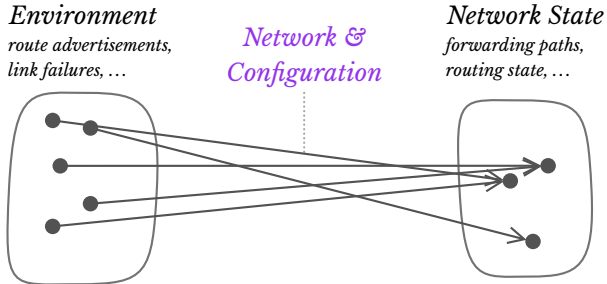
### *Environment*

*route advertisements,  
link failures, ...*



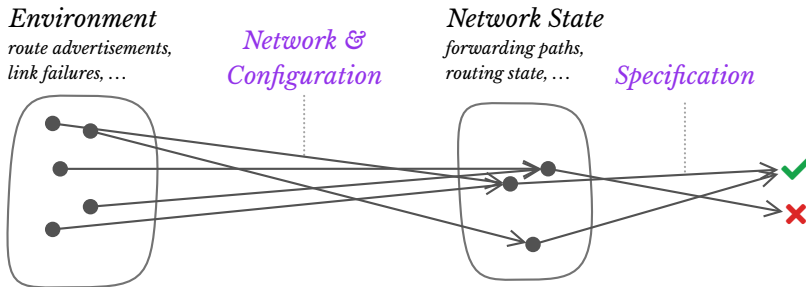
## Verification

Does the configuration satisfy the operators' specification  
*in all environments?*



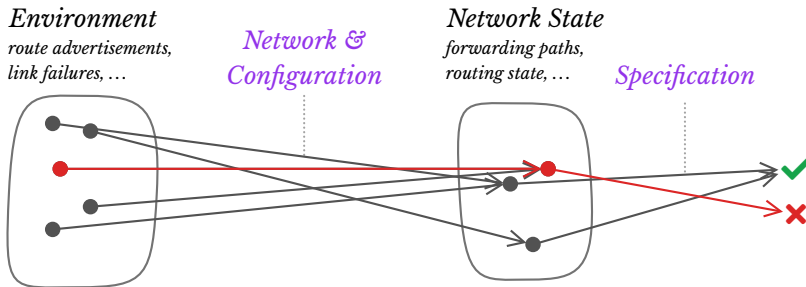
## Verification

Does the configuration satisfy the operators' specification  
*in all environments?*



## Verification

Does the configuration satisfy the operators' specification  
*in all environments?*

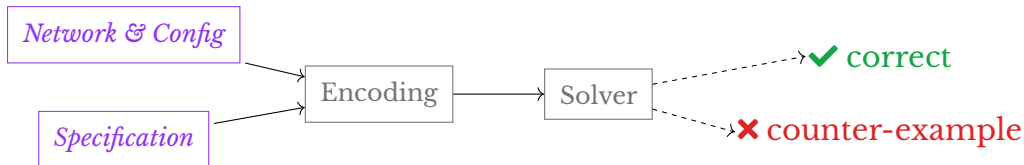


*Counter example*: an environment that violates the specification



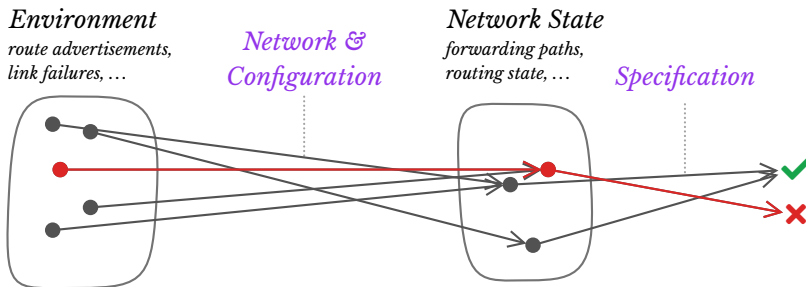
## *Limitations*

Typically, verifiers encode the network and specification and use generic solvers to find a counter-example.



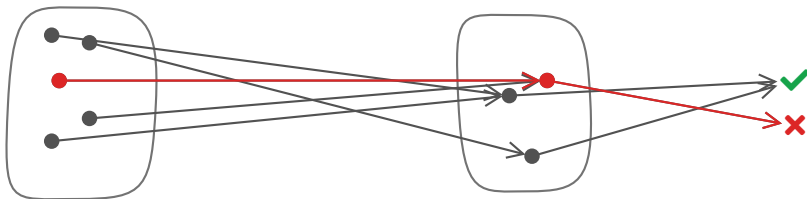
## Limitations

Traditionally, verifiers explore the space of environments.



## *Limitations*

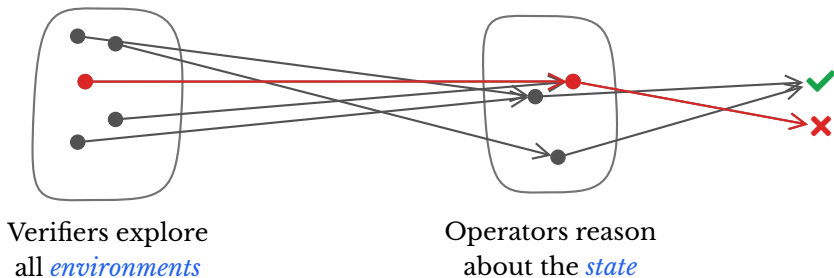
Traditionally, verifiers explore the space of environments.



Verifiers explore  
all *environments*

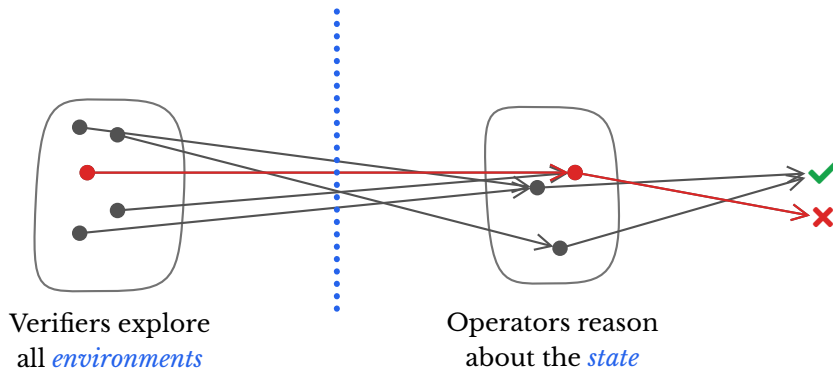
## *Limitations*

Traditionally, verifiers explore the space of environments.  
While operators reason about networks states.



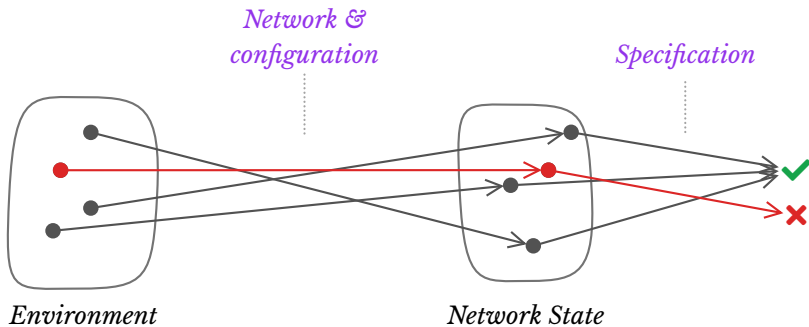
## Limitations

Loss of networking context for the verification



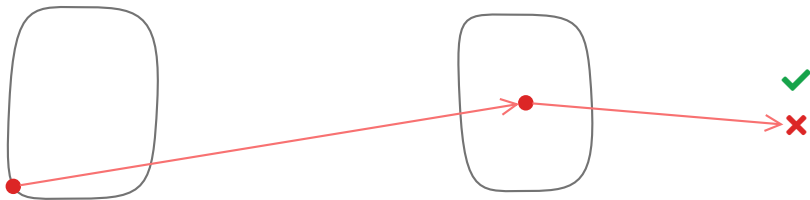
*Improvement 1*

A single counter-example is not very informative.



### *Improvement 1*

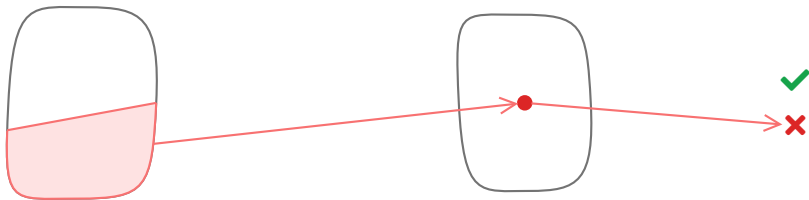
Instead of single edge case



E.g. Zürich is disconnected if  
AS-3 announces a route with  
AS-path length 4  
and community 3:1000  
and link zh-bsl is down  
and AS-2 does not announce a route

*Improvement 1*

Fully characterise the **routing states**  
with the corresponding **subspace of environments**.



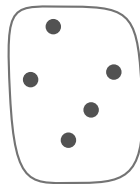
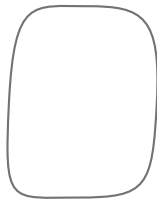
E.g. Zürich is disconnected if  
AS-3 announces a route with  
AS-path length greater than 2.



## *Improvement 2*

Exploring routing states lets you guide your exploration

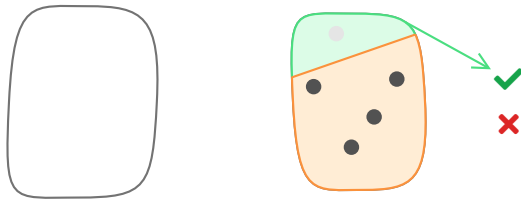
Look only where you expect  
mistakes



## *Improvement 2*

Exploring routing states lets you guide your exploration

Look **only** where you expect mistakes

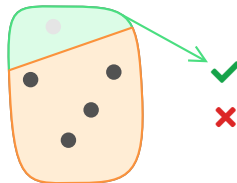
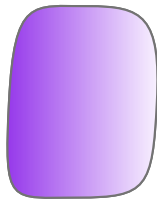


E.g. only explore the states where at least one router is disconnected.

## *Improvement 2*

Exploring routing states lets you guide your exploration

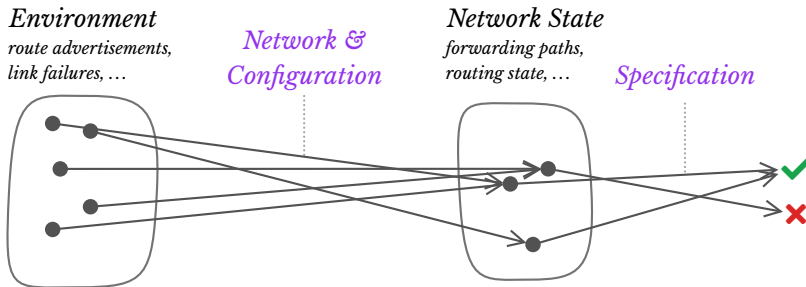
Look only where you expect  
mistakes and **where it matters  
most first**



←  
e.g. by increasing number  
of link failures

## Verification

Should be read right to left!



The challenge is therefore to explore the space of routing states by

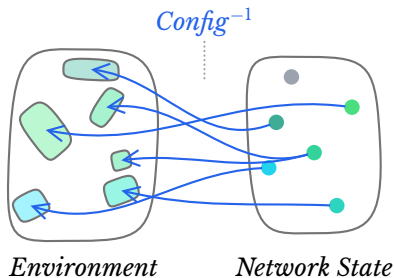


*Environment*

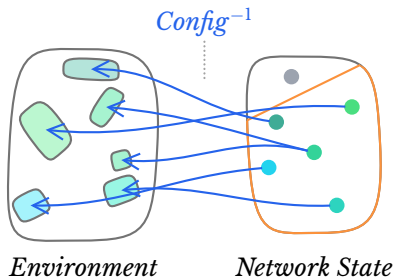


*Network State*

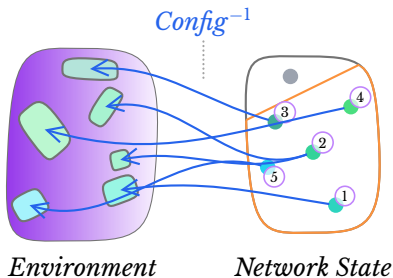
The challenge is therefore to explore the space of routing states by  
inverting the configuration function



The challenge is therefore to explore the space of routing states by  
inverting the configuration function  
partially (based on the specification)



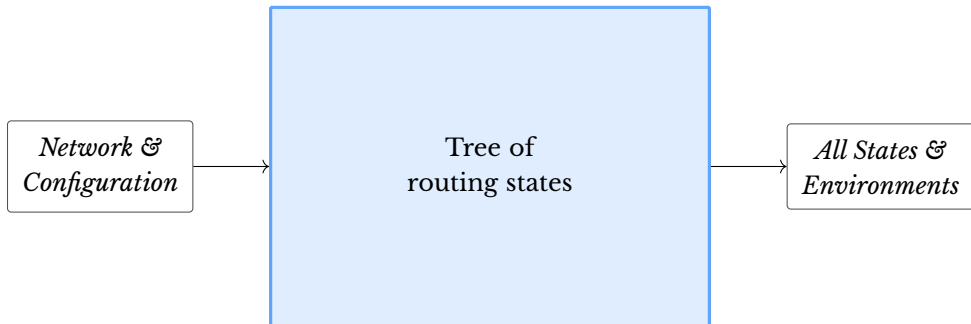
The challenge is therefore to explore the space of routing states by  
inverting the configuration function  
partially (based on the specification) and  
in order (based on domain knowledge).



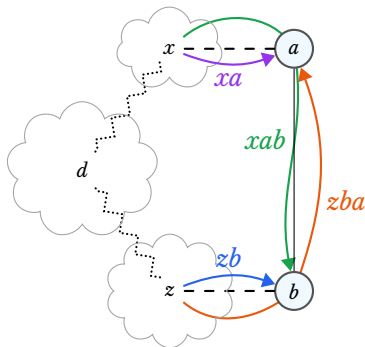


How do we construct this inverse mapping?

By building a *tree of routing states*

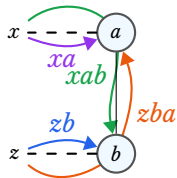


## Building the tree from a bgp network



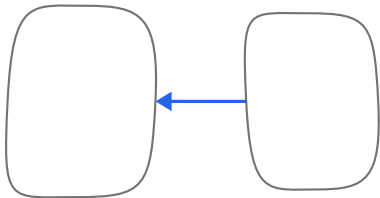
solid arrows are routes towards destination  $d$

## Building the tree to invert the configuration

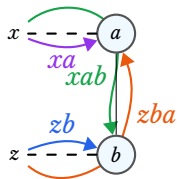


*Environment*

*Network State*

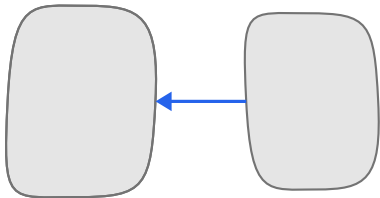


## Building the tree of (partial) routing states

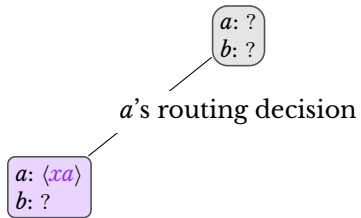
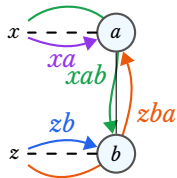


*Environment*

*Network State*

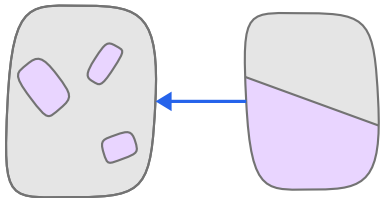


# Building the tree

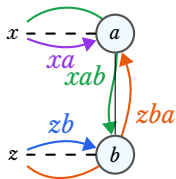


*Environment*

*Network State*

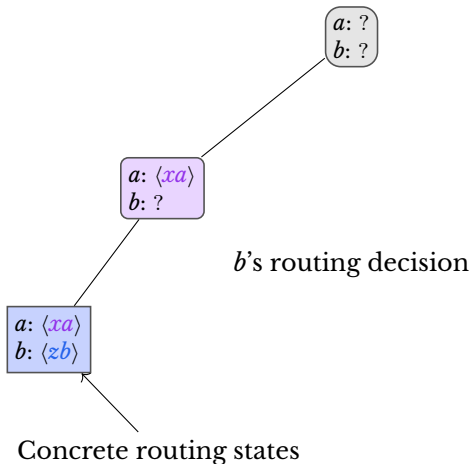
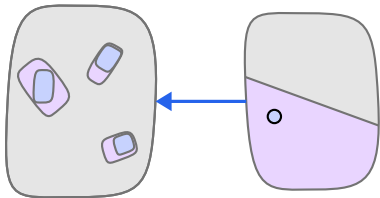


## Building the tree

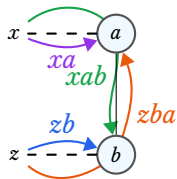


*Environment*

*Network State*

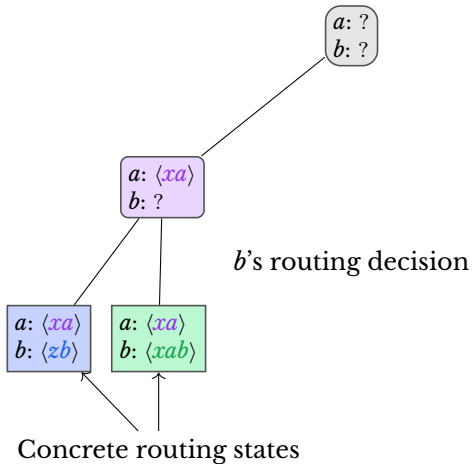
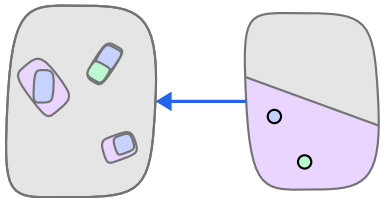


# Building the tree



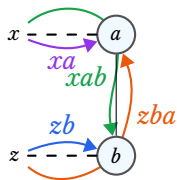
*Environment*

*Network State*



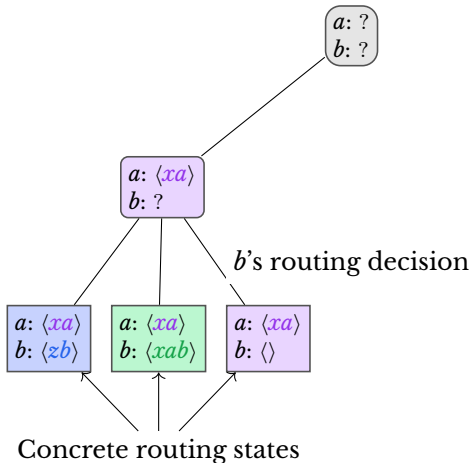
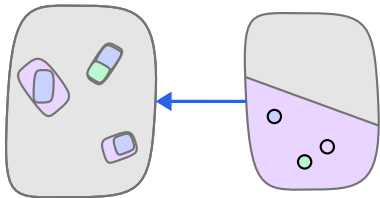


# Building the tree

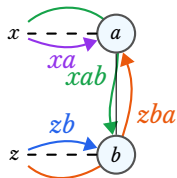


Environment

Network State

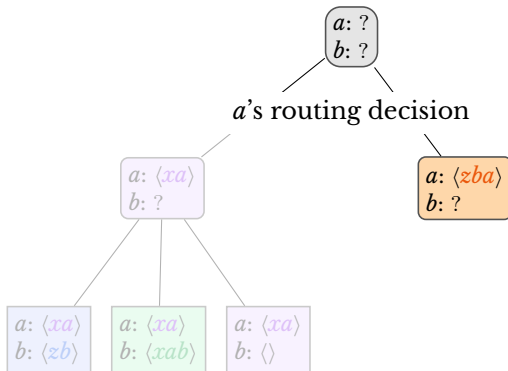
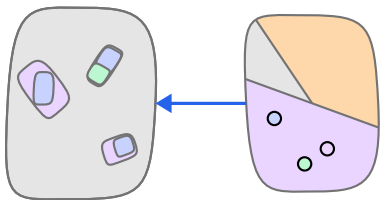


# Building the tree

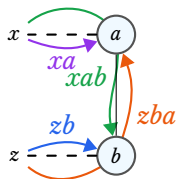


*Environment*

*Network State*

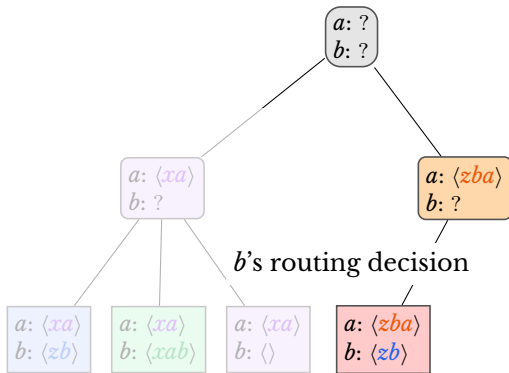
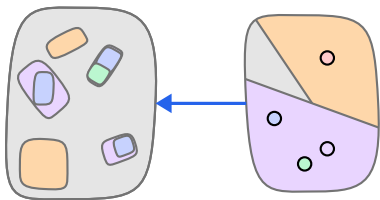


# Building the tree

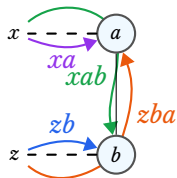


Environment

Network State

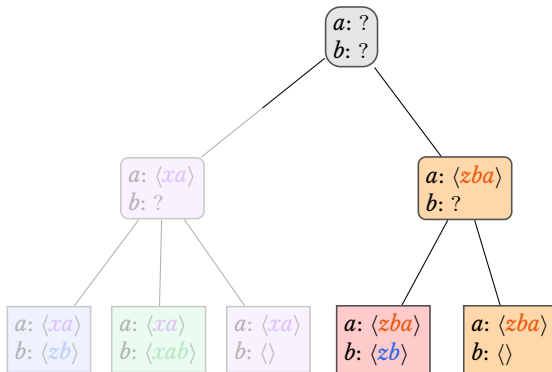
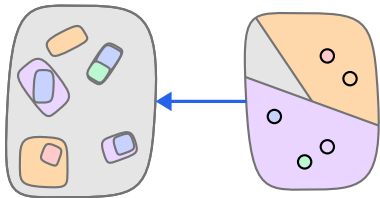


# Building the tree

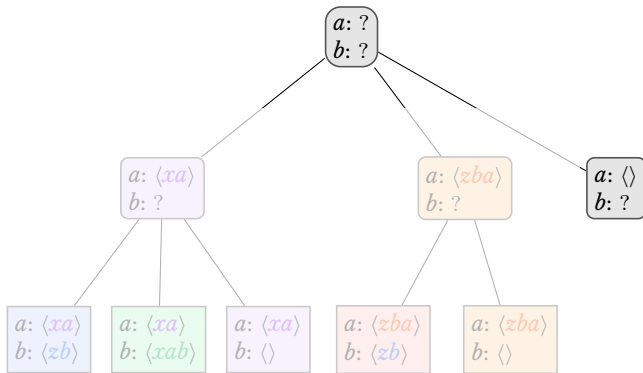
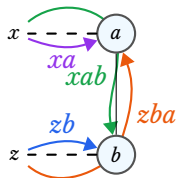


*Environment*

*Network State*

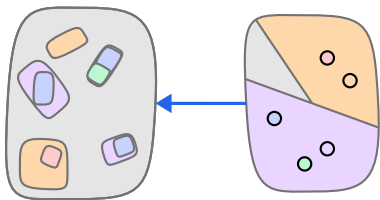


# Building the tree

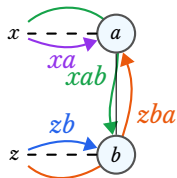


Environment

Network State

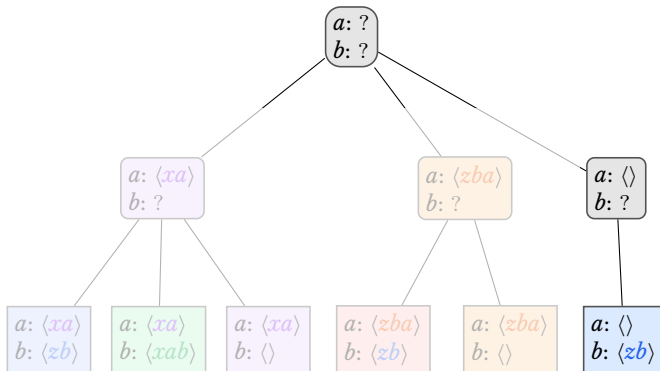
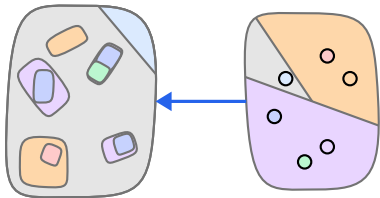


# Building the tree

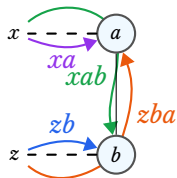


Environment

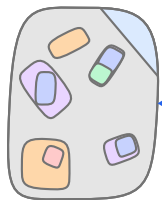
Network State



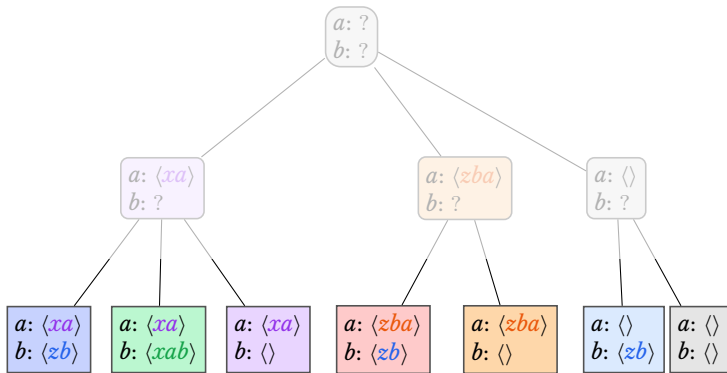
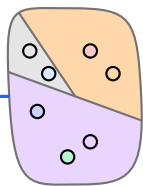
# Building the tree



Environment

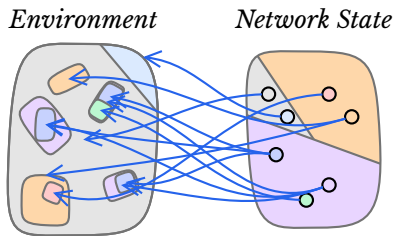


Network State



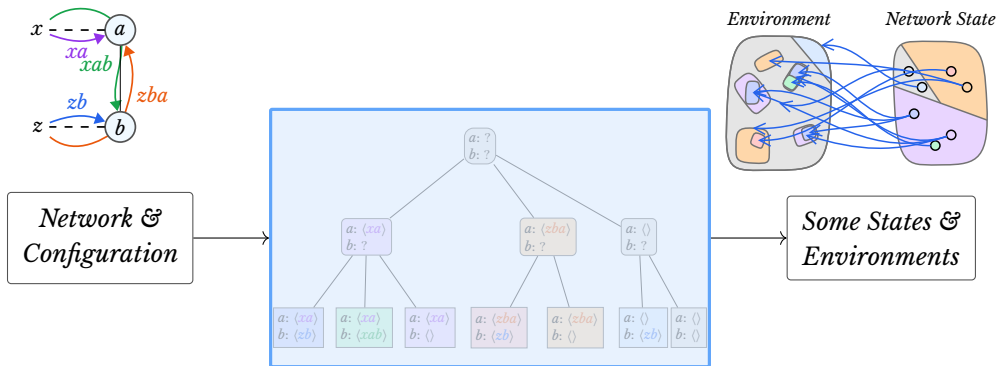
All possible routing states

The routing states tree builds an inverse mapping of the configuration!

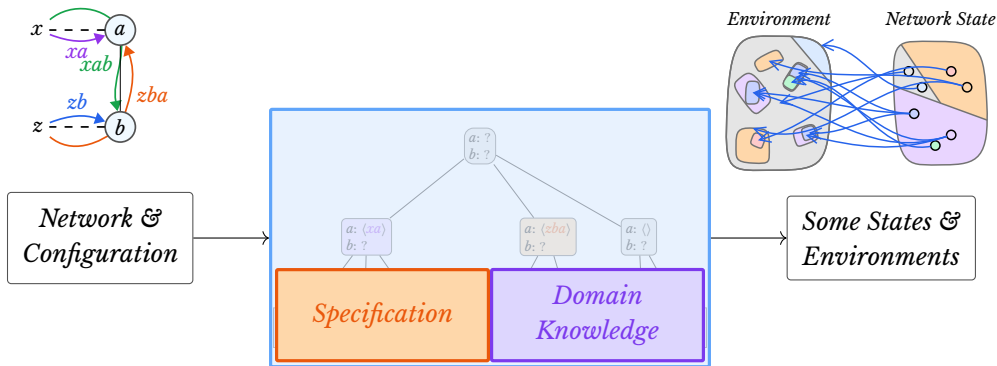




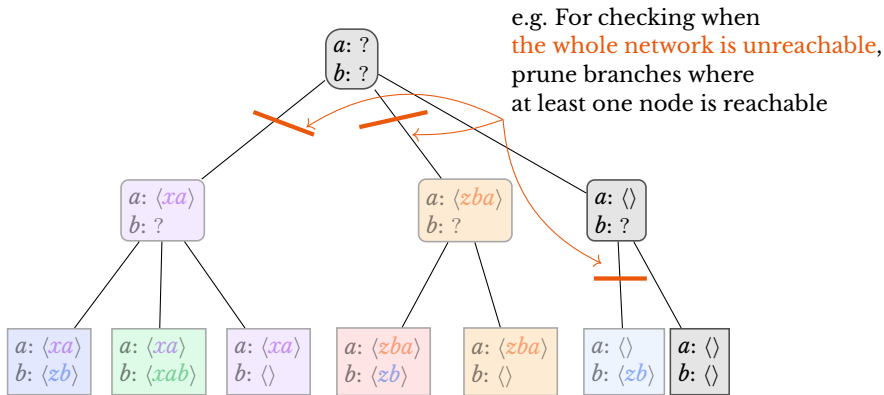
The routing states tree builds an inverse mapping of the configuration!



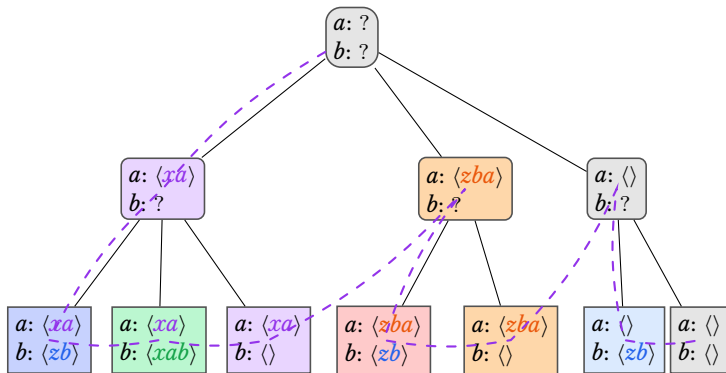
The routing states tree builds an inverse mapping of the configuration!  
 But how may we do it **partially** and **in order**?



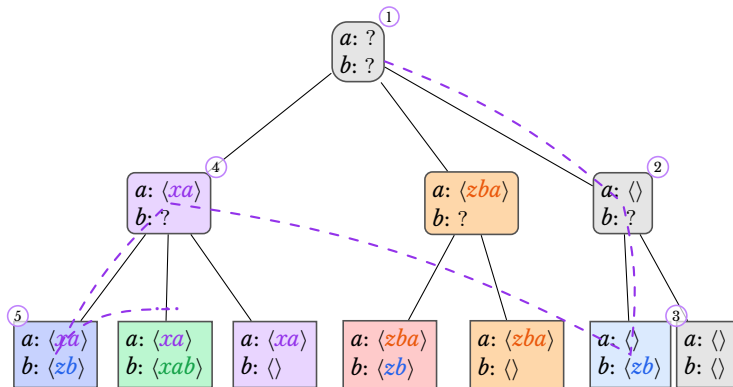
Using the specifications, we can **prune branches of the tree**



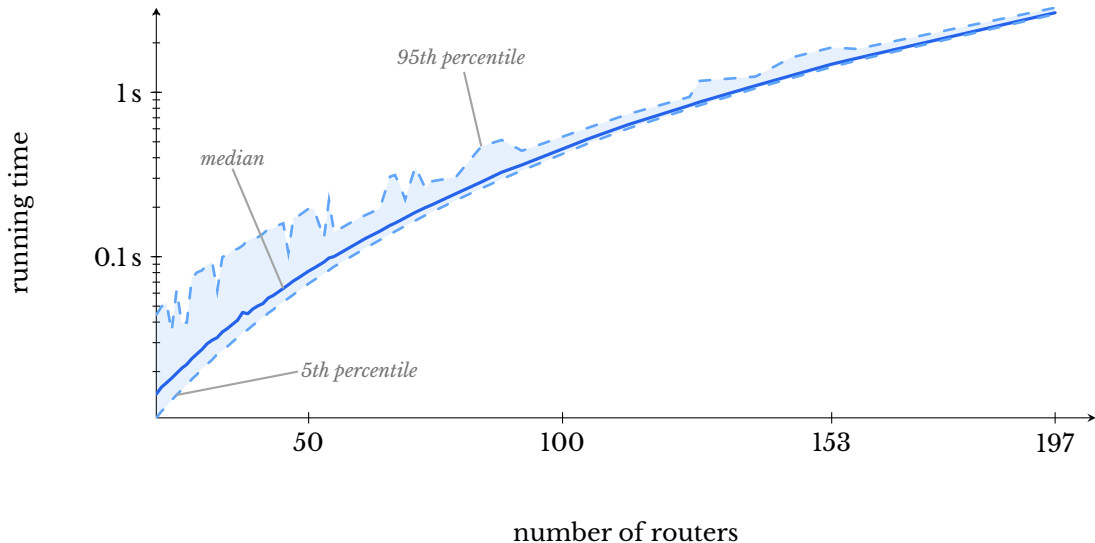
Using domain knowledge we may influence the traversal



Using domain knowledge we may influence the traversal



We can construct the state tree *within seconds*



# Complete evaluation available in the paper

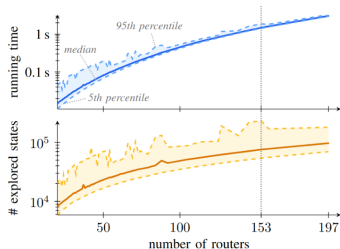


Fig. 2: Larger networks make the *BGP State Iterator* explore more states, and thus, increase its running time. The blue line in the top plot shows the running time, while the orange line in the bottom plot counts the number of states explored. The plot shows the median and the 5th and 95th percentiles.

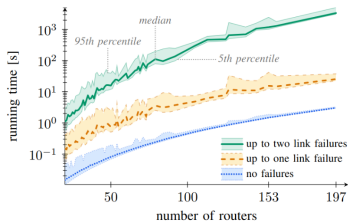
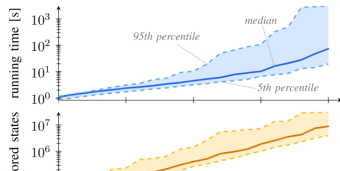


Fig. 4: The *BGP State Iterator* can find all states with up to two link failures within one hour. The green solid line shows the running time when considering up to two simultaneous link failures. The orange dashed line shows the same for a single failure, and the blue dotted line does so for no failures.

increases the number of possible routes and egress combinations in stable states. Despite that, we usually find *all* stable states within a couple of minutes. In the worst case, it takes the iterator up to 10 hours to explore the entire space.

The large variance can be attributed to the random assignment of the roles of the external networks (customer, peer, or provider). Routes from networks of the same role have the same local-preference, and hence, a stable state can select routes from any combination of networks of the same role. An even distribution of the roles results in fewer stable states, while a more skewed distribution yields many more.

# Guided Exploration of Control Plane Routing State

Tibor Schneider, Jean M  gret, Laurent Vanbever

