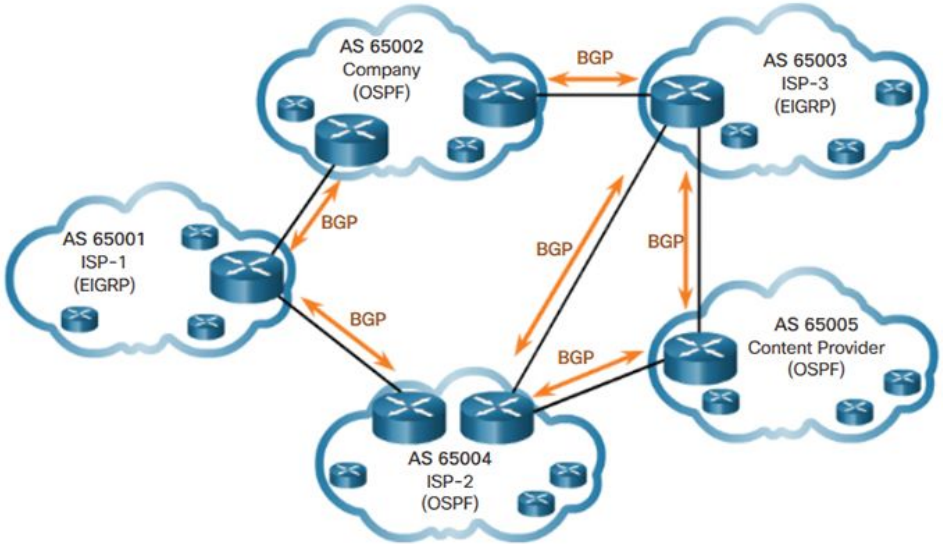
MESSI : Behavioral Testing of BGP Implementations

Rathin Singha, Rajdeep Mondal,
Ryan Beckett, Siva Kesava Reddy Kakarla,
Todd Millstein, George Varghese


UCLA


Microsoft

BGP : The backbone of modern internet routing



Connects autonomous systems

Ubiquitous (used by ISPs, enterprises, DCs etc.)

Large blast radius!

Outages caused by BGP bugs

Previous works on BGP testing

Significant research done on identifying BGP config bugs

RCC [NSDI'05]	HSA [NSDI'12]	Batfish [NSDI'15]	Bagpipe [NetPL'16]
ARC [SIGCOMM'16]	Minesweeper [SIGCOMM'17]	RCDC [SIGCOMM'19]	Plankton [NSDI'20]
Tiramisu [NSDI'20]	Hoyan [SIGCOMM'20]	Timepiece [PLDI'23]	Lightyear [SIGCOMM'23]

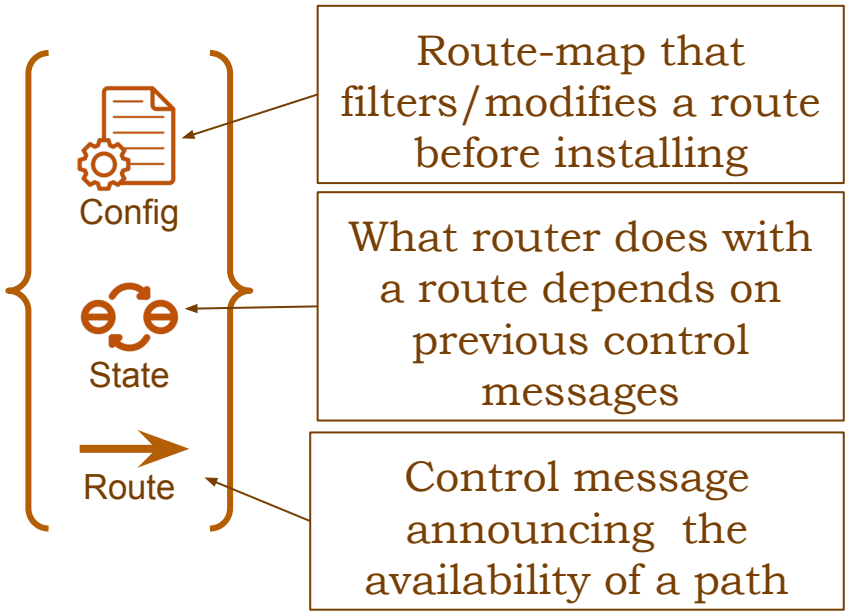
According to a study*, 36% of the significant and customer-impacting incidents in Microsoft's network are caused by **implementation bugs**.
But, less work on automatically finding BGP protocol implementation errors!

* Crystalnet: Faithfully emulating large production networks. [SOSP '17]

Our goal

Automatically **generating tests for BGP implementations to find behavioral bugs**

Test case =



Key Challenge is to jointly generate these triplets

Existing approach limitations

Fuzz testing → randomly generated triplets are unlikely to find bugs that need a particular combination

Symbolic execution → implementation has lot of paths due to low level optimizations. i.e. limited coverage

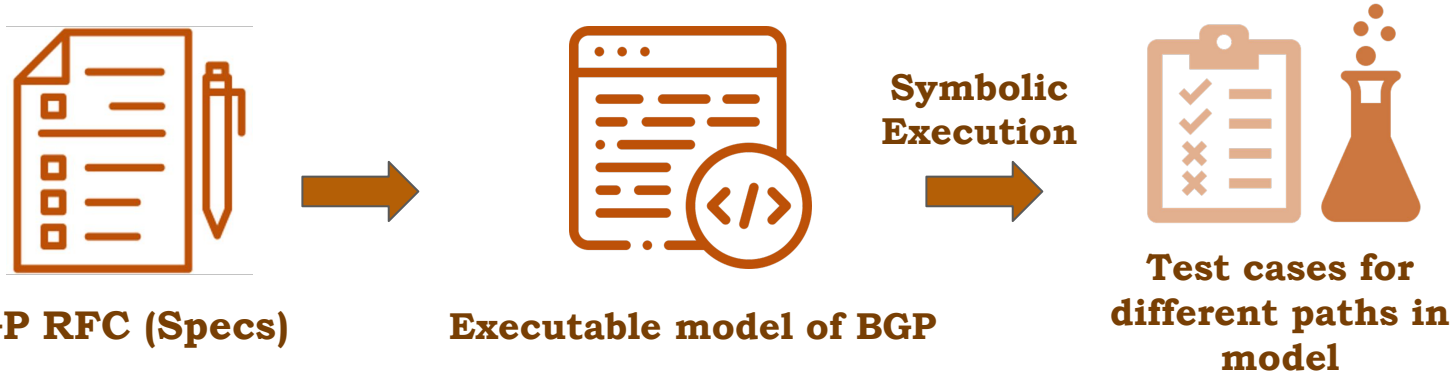
MESSI: **M**odular **E**xploration of **S**tate and **S**tructure **I**nclusively

The **first automated approach** and tool MESSI to identify RFC violations in black-box BGP implementations.

Key Results

- Generated 150K+ test cases
- 22 bugs found across 6 popular BGP implementations - FRR, Quagga, BIRD, GoBGP, Batfish, Fastplane
- Found bugs in: Prefix lists, Regexes, Communities, AS path, MED, incremental updates etc.
- 8 bugs already fixed.

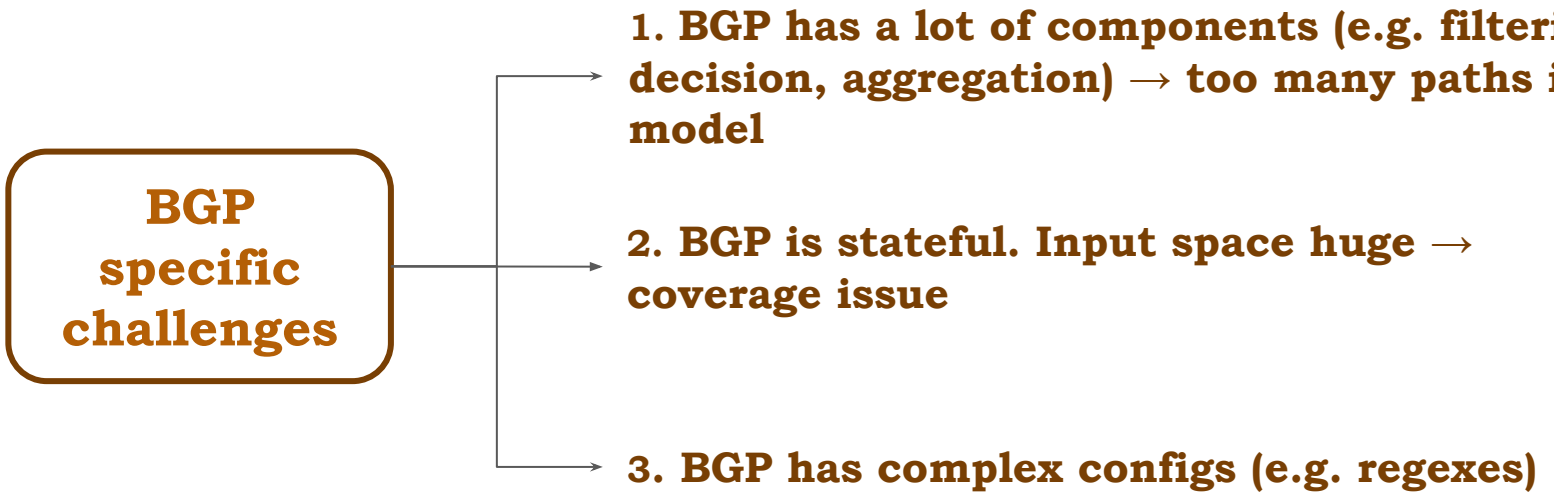
Our approach : Model-based testing



- **Generates tests that capture complex semantic behaviors (joint generation of triplet)**
- **Simpler model → Symbolic execution possible**
- **Coverage guarantees with bounded size of symbolic inputs.**

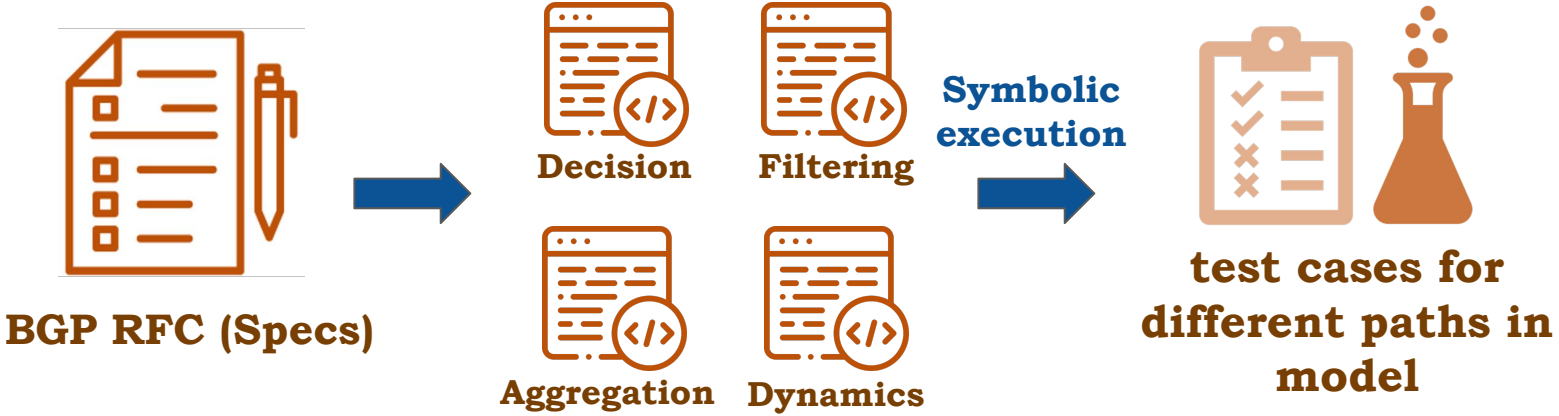
Key challenges

*Previous work **SCALE*** used Model-based testing approach to find bugs in **DNS** nameservers.*



* *Siva Kesava Reddy Kakarla, Ryan Beckett, Todd Millstein, and George Varghese. {SCALE}: Automatically finding {RFC} compliance bugs in {DNS} nameservers.(NSDI 22)*

1. Dealing with protocol complexity via modularity

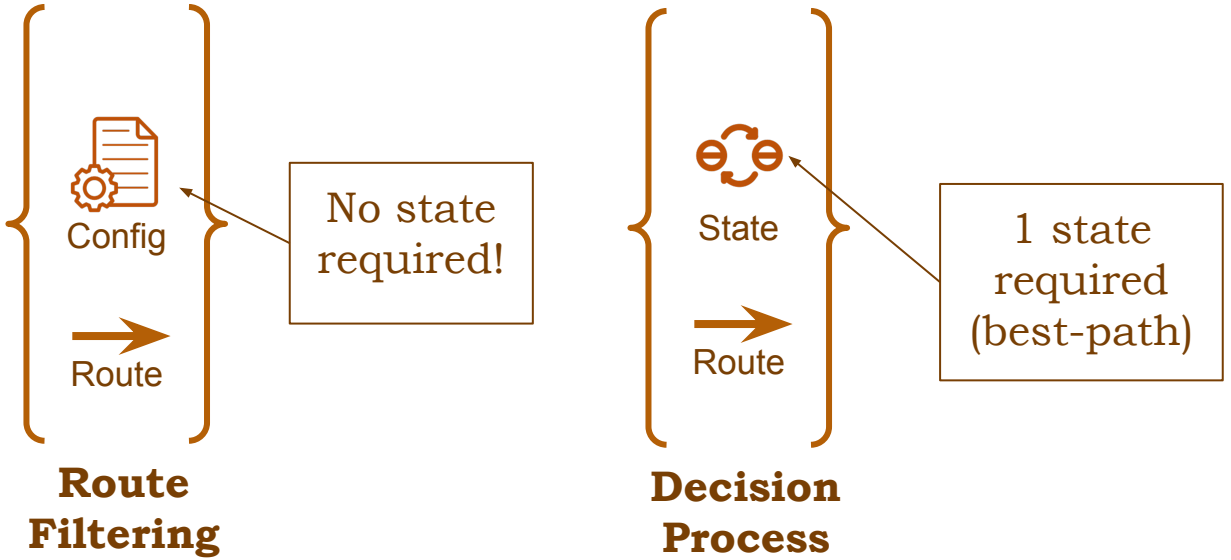


Filtering (m paths) & Decision (n paths)

$$m \times n \Rightarrow m + n$$

Alleviates complexity problem!

2. Modularity also helps to deal with Statefulness

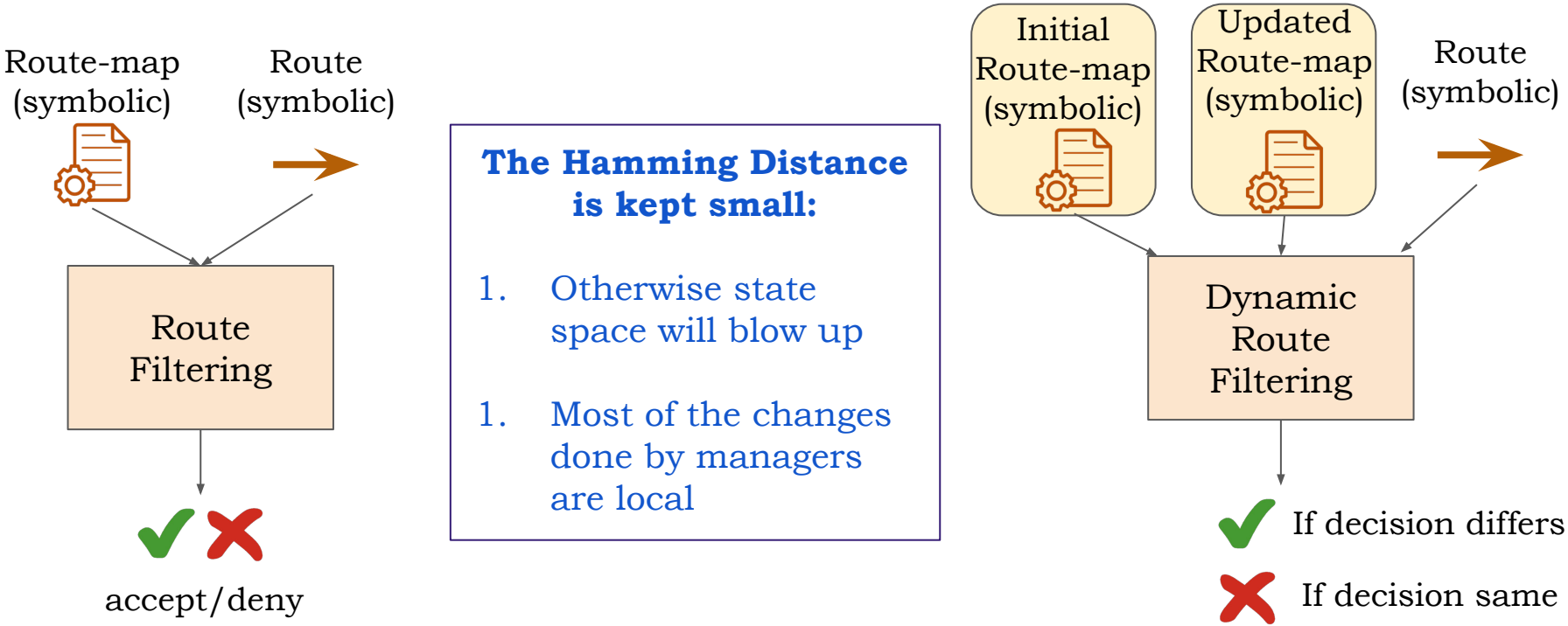


Each model only needs minimum symbolic states

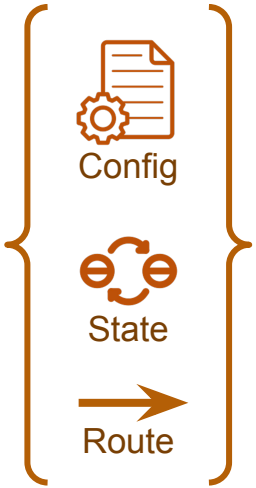
Alleviates state problem!

Testing BGP dynamics (another kind of statefulness)

Many implementations have bugs because they use optimizations for **incremental updates** of route maps



3. Dealing with complex structures: Regexes



Test case

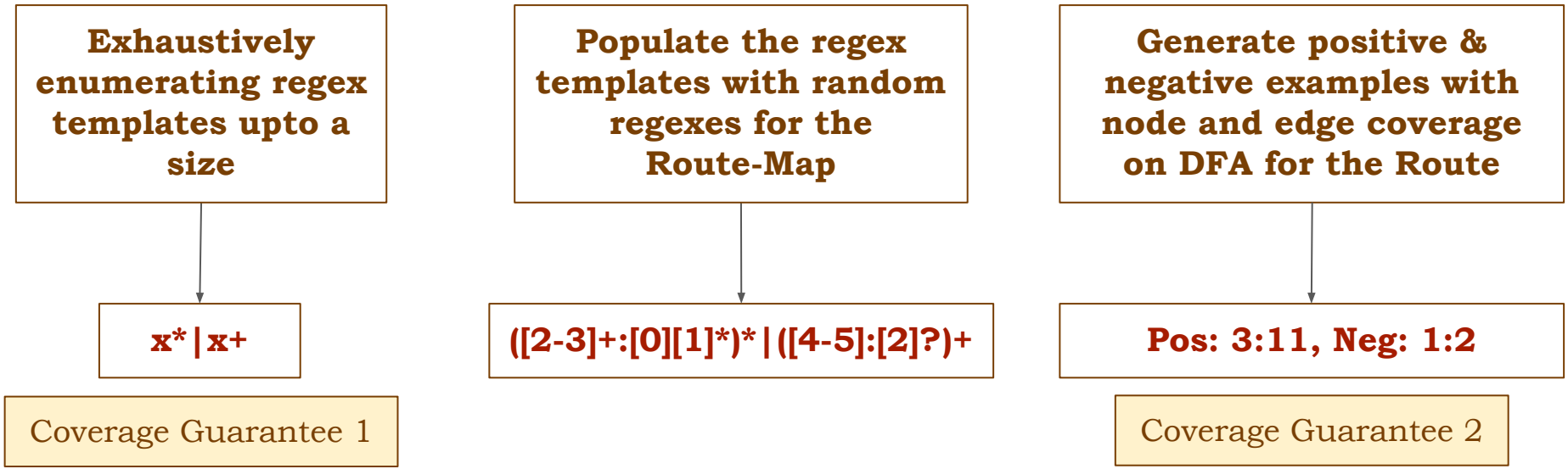
- **Route-maps can have regexes (community, as-path)**
- **We need to deal with symbolic regex. i.e. beyond current solver capabilities**
- **So the model needs to use a fixed set of regexes**

How to generate fixed set of regexes with coverage guarantees?

Regex enumeration and testing framework

Hybrid approach

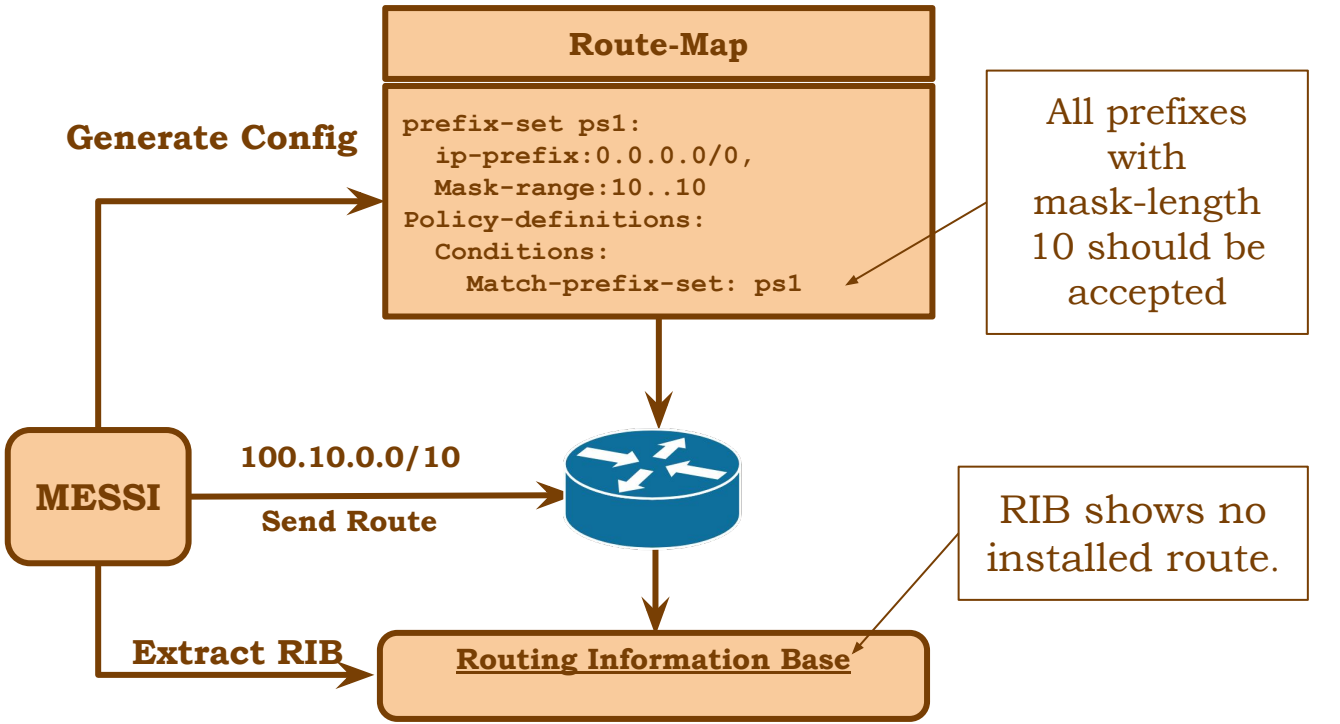
ENUMERATION + **FUZZING** + **GRAPH TRAVERSAL**



Results

- Generated 150K+ test cases
- 22 bugs found across 6 popular BGP implementations - FRR, Quagga, BIRD, GoBGP, Batfish, Fastplane
- Found bugs in: Prefix lists, Regexes, Communities, AS path, MED, incremental updates etc.
- 8 bugs already fixed.

Example Bug #1 (GoBGP)



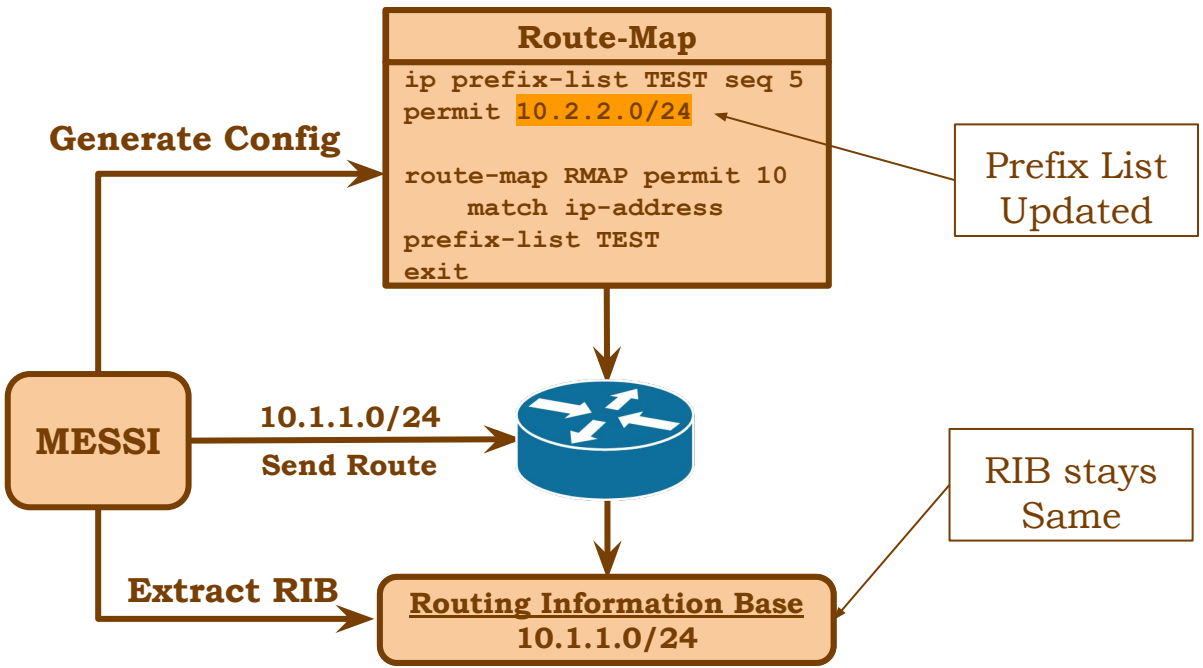
Reason:

Buggy code for dealing 0.0.0.0/0 prefix and a single mask length

Challenge:

Bug caused by some very specific combination of route and config

Example Bug #2 (FRR)



Reason:

The underlying “any” flag stays enabled even after the change in route map

Challenge:

The bug is only reproduced if the config changed from “permit-any”, i.e. depends on **state**

Contributions

The **first automated approach** and tool MESSI to identify RFC violations in black-box BGP implementations.

Modular exploration to deal with protocol complexity.

Efficient enumerative testing of regular expression which cannot otherwise be handled by symbolic testing.

A testing framework to catch bugs due to **BGP dynamics** caused by incorrect implementation attempts to do incremental computation

Thank You!

Future Work

1. Support BGP features like Redistribution, Reflection etc.

1. Testing the integration of multiple BGP features

1. Automate the process of Model Building using LLMs

1. Applying these ideas to other stateful protocols with complex structures

Limitations

1. We focus on RFC compliance bugs but **not performance or coding bugs** e.g. overflow
2. We don't test **route reflection, confederations, redistribution**.
3. We don't model **some regex features** like constraining a route's community set size
4. Modular exploration possibly doesn't test **complicated interaction between BGP features** but it allows to test each feature extensively.