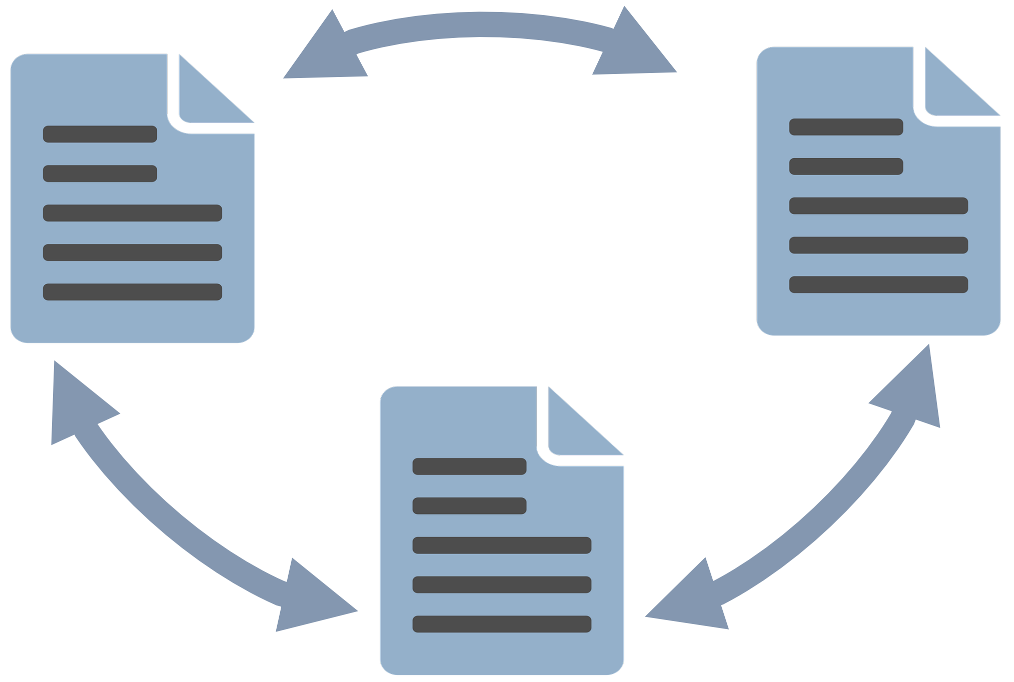
Formal Verification of Distributed Network Control Planes

Aarti Gupta
Princeton University

Joint work with:
Ryan Beckett (Microsoft Research), Ratul Mahajan (Univ. of Washington),
Divya Raghunathan (Princeton), Tim Alberdingk Thijm (Princeton),
and David Walker (Princeton)

Network Design

Control Plane



Protocol + Configuration

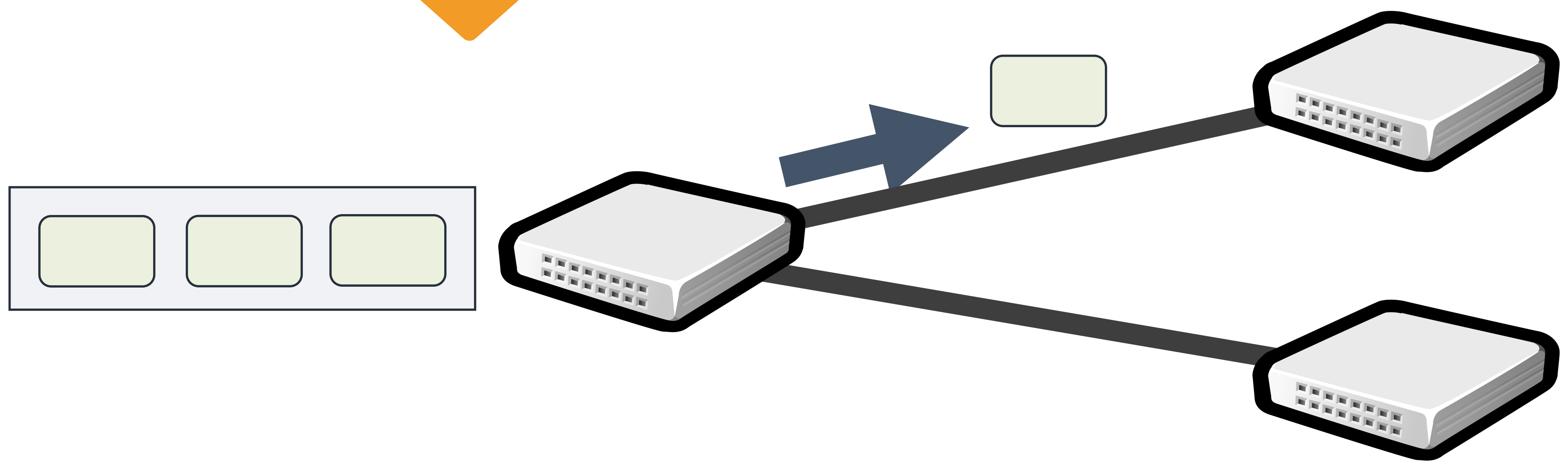
Control Plane Protocols
RIP, OSPF, BGP, …

Data Plane

7.2.0.*	port 1
9.2.*.*	port 2

7.2.0.*	port 5
9.2.*.*	port 3

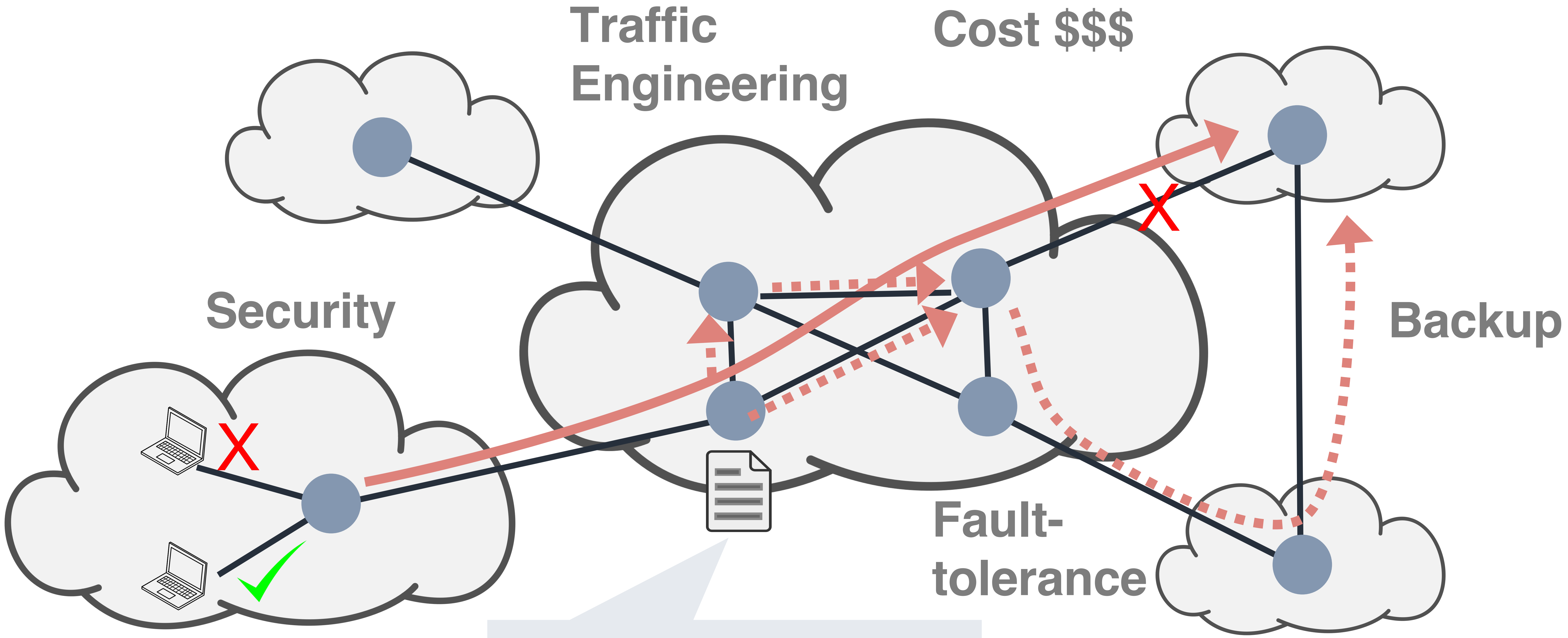
Forwarding Tables



Runtime Behavior

Network Control Plane

Primary goal is to get traffic from point A to point B but …



Configuration

Distributed program

Vendor-specific

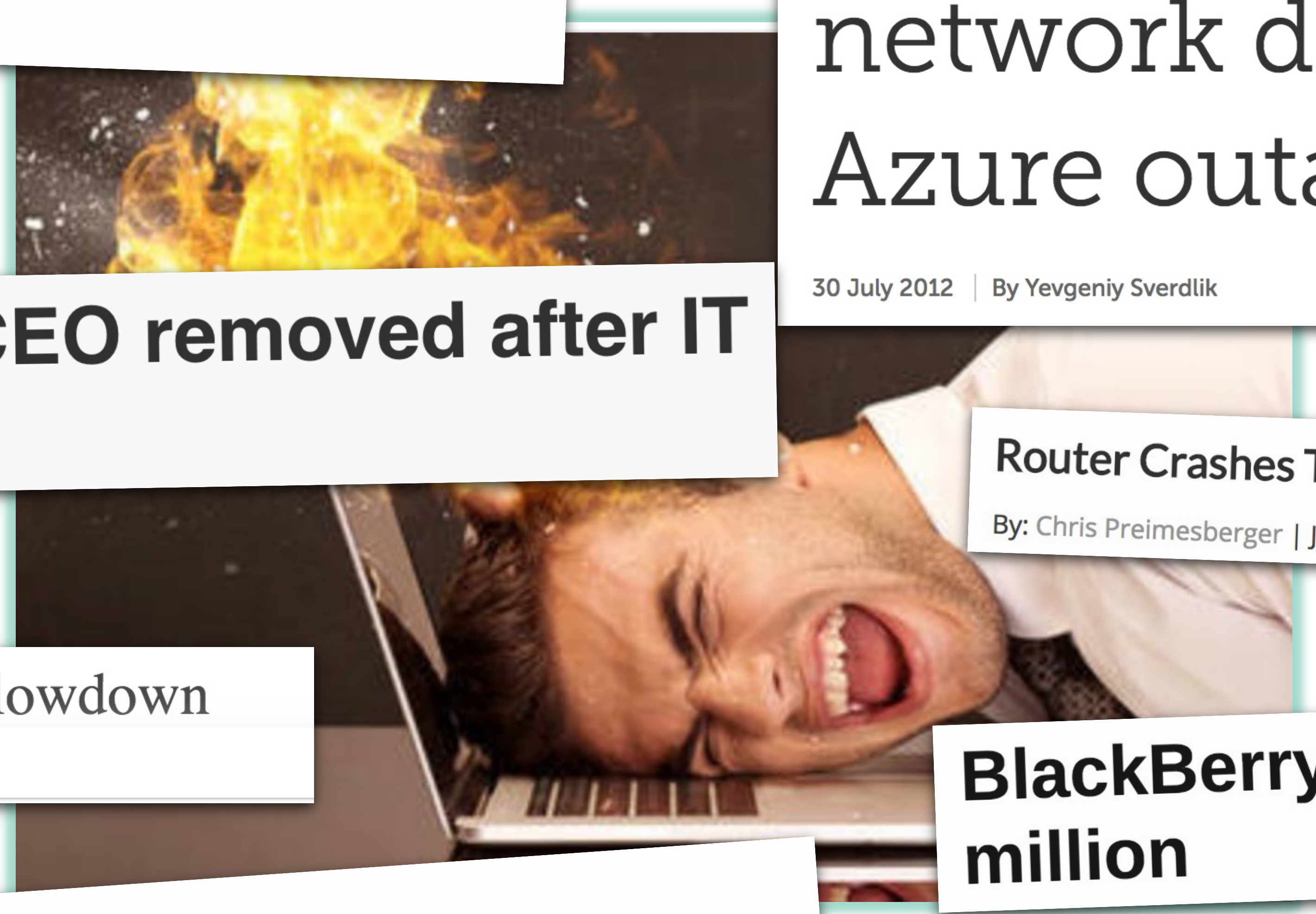
Assembly-language

Thousands of lines

Misconfiguration is a BIG problem

BGP errors are to blame for Monday's Twitter outage, not DDoS attacks
No, your toaster didn't kill Twitter, an engineer did

Microsoft: misconfigured network device led to Azure outage
30 July 2012 | By Yevgeniy Sverdlik



Unions want Southwest CEO removed after IT outage

Router Crashes Trigger Major Southwest IT System Failure
By: Chris Preimesberger | July 21, 2016

Massive route leak causes Internet slowdown
Posted by Andree Toonk – June 12, 2015 – BGP instability – No Comments

BlackBerry outage could cost RIM $100 million

Home / Cisco Security / Security Advisories and Alerts
Security Activity Bulletin
Misconfigured Router Causes Increased BGP Traffic and Isolated Outages for Internet Services

Xbox Live outage caused by network configuration problem
BY TODD BISHOP on April 15, 2013 at 9:27 am

Motivated many formal verification efforts

- Data plane verification checks a *snapshot* of the network
  - Analyzes the given forwarding rules at routers to check various properties
  - Based on model checking, symbolic simulation, SAT/SMT/BDD techniques
  - Routinely applied in large data centers (~10k routers)

Data Plane Verifiers

Anteater	[Mai 2011]
HSA	[Kazemian 2012]
Veriflow	[Kurshid 2013]
NoD	[Lopes 2015]
Symmetries	[Plotkin 2016]

…

Motivated many formal verification efforts

- Control plane verification checks *router configurations* that determine the forwarding rules
  - Verifies *all possible data planes* that result from the configurations
  - Based on model checking, graph-based techniques, SAT/SMT/BDD techniques
  - So far, they have been demonstrated on 2-3 k routers (max)

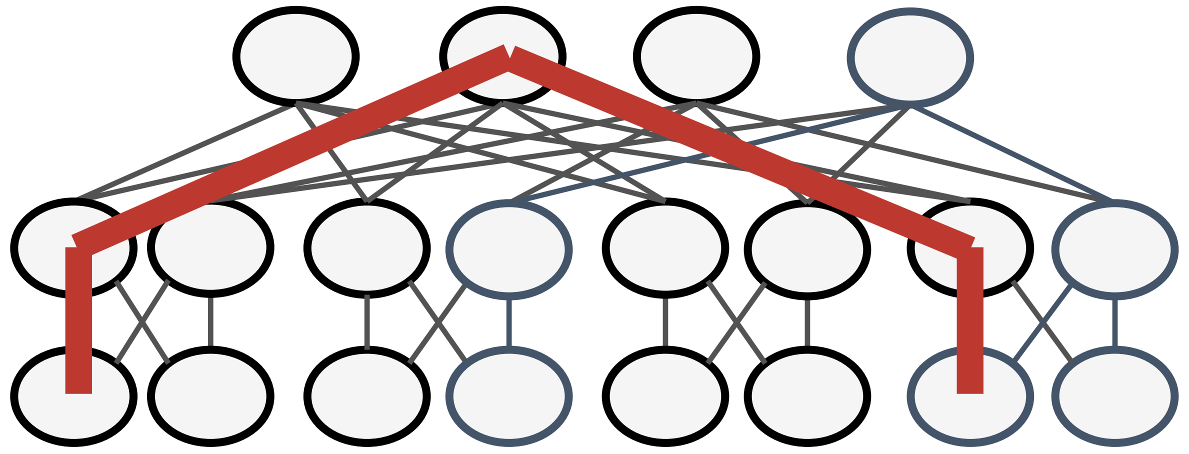
Control Plane *Simulators*

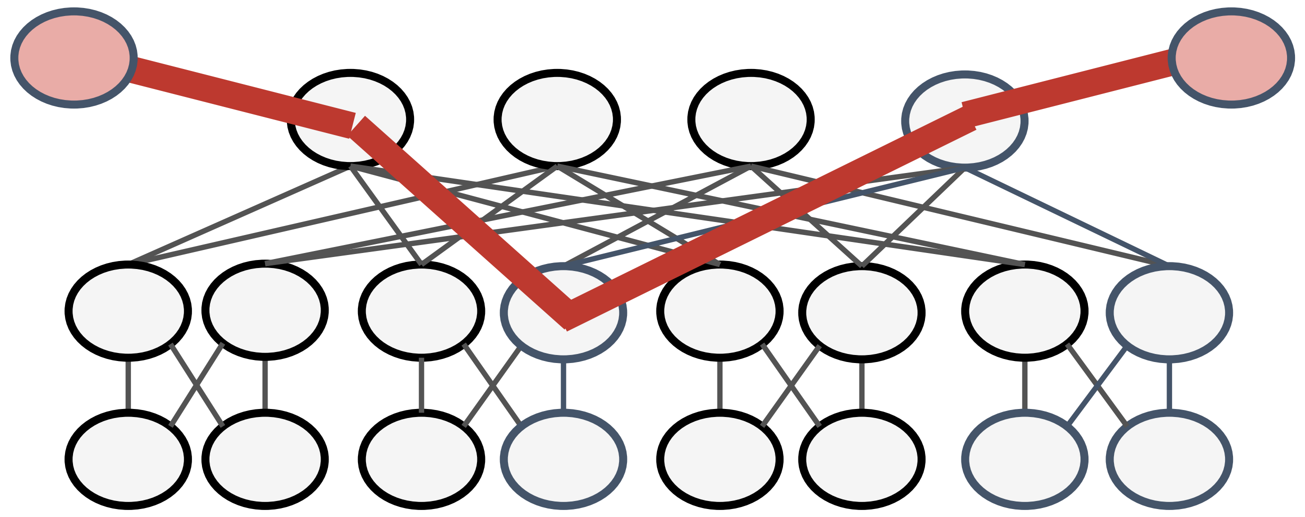
C-BGP	[Quotin 2005]
Batfish	[Fogel 2015]

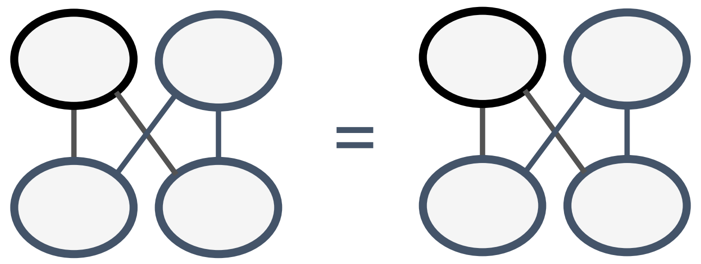
Control Plane *Verifiers*

Bagpipe	[Weitz 2016]	*symbolic execution*
ARC	[Gember-Jacobsen 2016]	*graph-based*
ERA	[Fayaz 2017]	*semi-symbolic*
FastPlane	[Lopes 2019]	*semi-symbolic*
Plankton	[Prabhu 2020]	*semi-symbolic*
Hoyan	[Ye 2020]	*semi-symbolic*
Our work	[Becket 2017, 2018, …]	*fully-symbolic*

Network Properties


reachability


no transit


Router or subnet equivalence


no loops

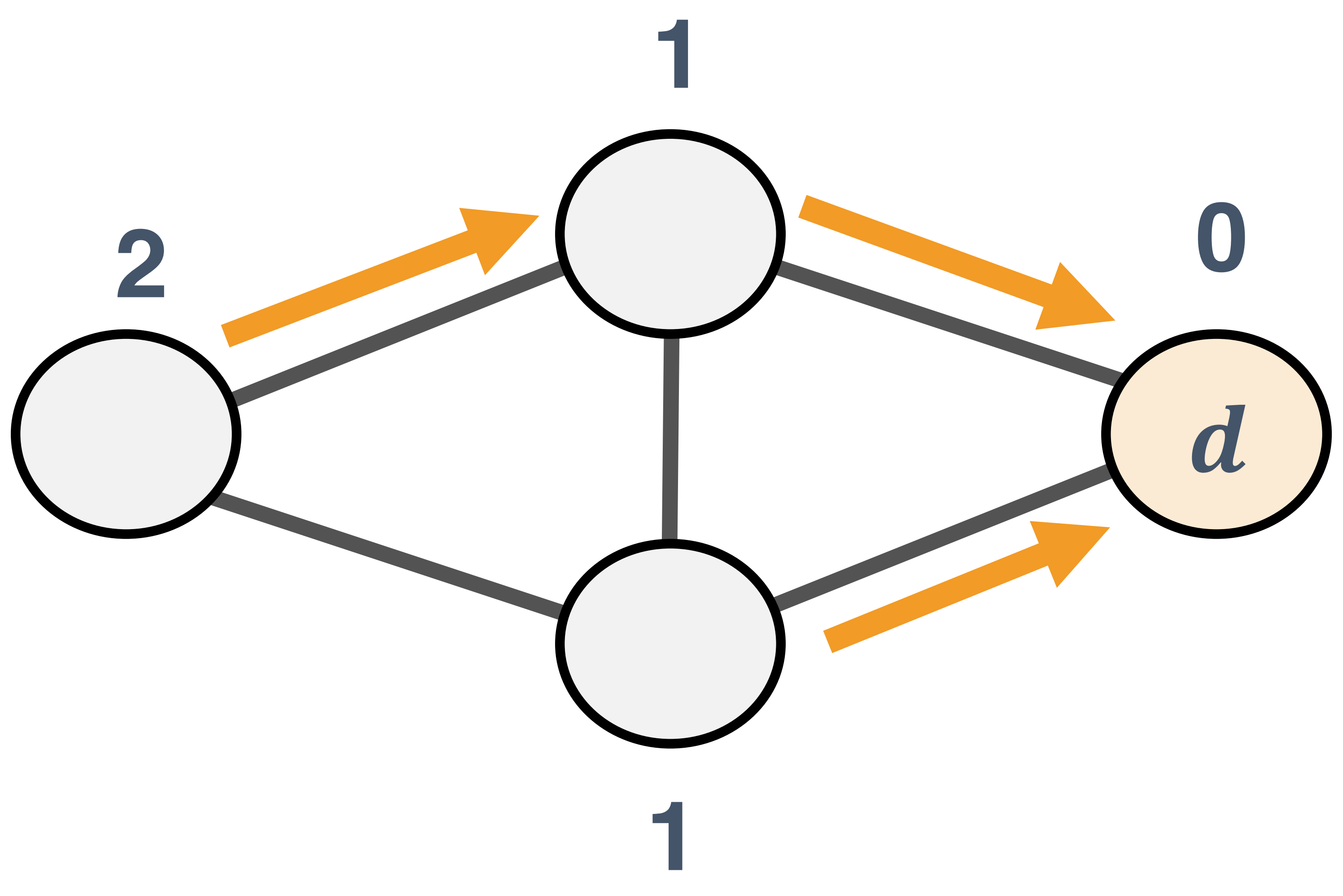

no black holes

Modeling Network Control Planes in MineSweeper

Ryan Beckett, Aarti Gupta, Ratul Mahajan, David Walker:
A General Approach to Network Configuration Verification. SIGCOMM 2017: 155-168

A Generic Routing Protocol

[Griffin and Sobrinho: Metarouting]



Idealized RIP:
A simple routing protocol

- The origin creates an ***initial announcement*** stating it has a path to destination *d*
- Other nodes that receive the announcement pass it on to their neighbors, ***possibly modifying it***
- When nodes receive multiple announcements, they ***choose a best one***
- Eventually (hopefully), the system converges to a ***stable solution***: all nodes are happy

Stable Paths Problem (SPP)

[Griffin et al. 2002]

Imperative, Stateful Program

```
process spvp(u)
begin
    receive P from w  ⟶
        begin
            rib-in(u ⇐ w) := P
            if rib(u) ≠ best(u) then
            begin
                rib(u) := best(u)
                for each v ∈ peers(u) do
                begin
                    send rib(u) to v
                end
            end
        end
end
```

Logic Model

Choices (P, u) = …
Best (u) = …

Each node is locally stable

- **Prior work on reasoning about protocol/network convergence**
- **We apply it for verifying *network configurations of protocols***

Minesweeper: Insights

Network protocols are designed to generate stable paths
i.e., routers exchange messages to make best choice, which stays stable

- **Our Idea**

Capture network control plane behavior in terms of logical constraints, such that *satisfying solutions are stable paths* in the network

- **Analogy to Program Verification**

Program: Satisfying solution represents a path in the program graph
Network: Satisfying solution represents *stable* paths in the network graph

But: arbitrary (not well-structured) graphs in network topology, a single solution corresponds to many paths; we target a *stable routing tree*

Minesweeper: Key Choices

Choice 1: Model routing *graphs*, not paths at a time

- Too many paths, but all paths share the same graph
- In the data plane, reasoning is done per-packet because there is no interference between packets along different paths
- But, in the control plane, routing messages along different paths *interact* with each other – modeling this interaction can be expensive!

Choice 2: Don't compute states due to exchange of routing messages, but perform *search* on final stable states

- Familiar lesson from symbolic model checking vs. bounded model checking
- Solve a *search* problem over state space, use modern SAT/SMT solvers
- Often scales better than *computing* sets of states iteratively

Minesweeper Approach

- **SRP** (Stable Routing Paths) – a general, logical control plane model
- Framed in terms of **local** route processing constraints at a node

- Applies translation to **SMT-based logic** for **verification**
- Heavily **optimized** to make the resulting tool practical

Minesweeper: SRP Model

Topology:	$G = (V, E, d)$ where $d: dest \in V$	
Attributes:	$A_\perp = A \cup \{\perp\}$	protocol
Preference relation:	$\prec: \ A \times A$	
Transfer function:	$\text{transfer}: \ E \times A \to \ A_\perp$	configuration
Initial value:	$a_d \in A_\perp$	

- An SRP solution is a labeling (based on neighbors): $L: \ V \to A_\perp$
- **An SRP solution is locally stable, i.e., each node is happy**
- Each SRP solution corresponds to a forwarding relation

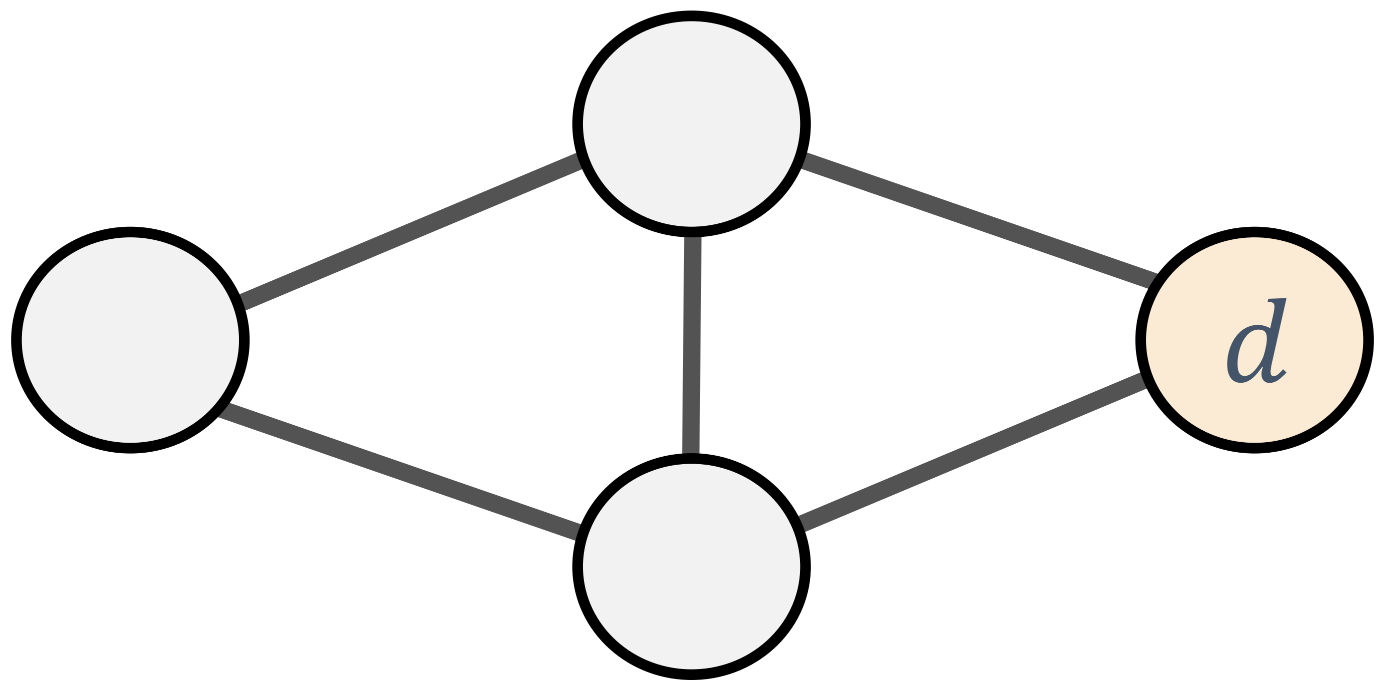
SRP Models for Popular Protocols

- Many network protocols are used in practice
  - RIP
  - OSPF
  - BGP (eBGP, iBGP)
  - Static Routing

- Minesweeper handles them all as SRPs
  - uniform model allows handling fancy features like route redistribution etc.

Example SRP

Routing Information Protocol (RIP)



Attributes:

$A = \{0..15\}$

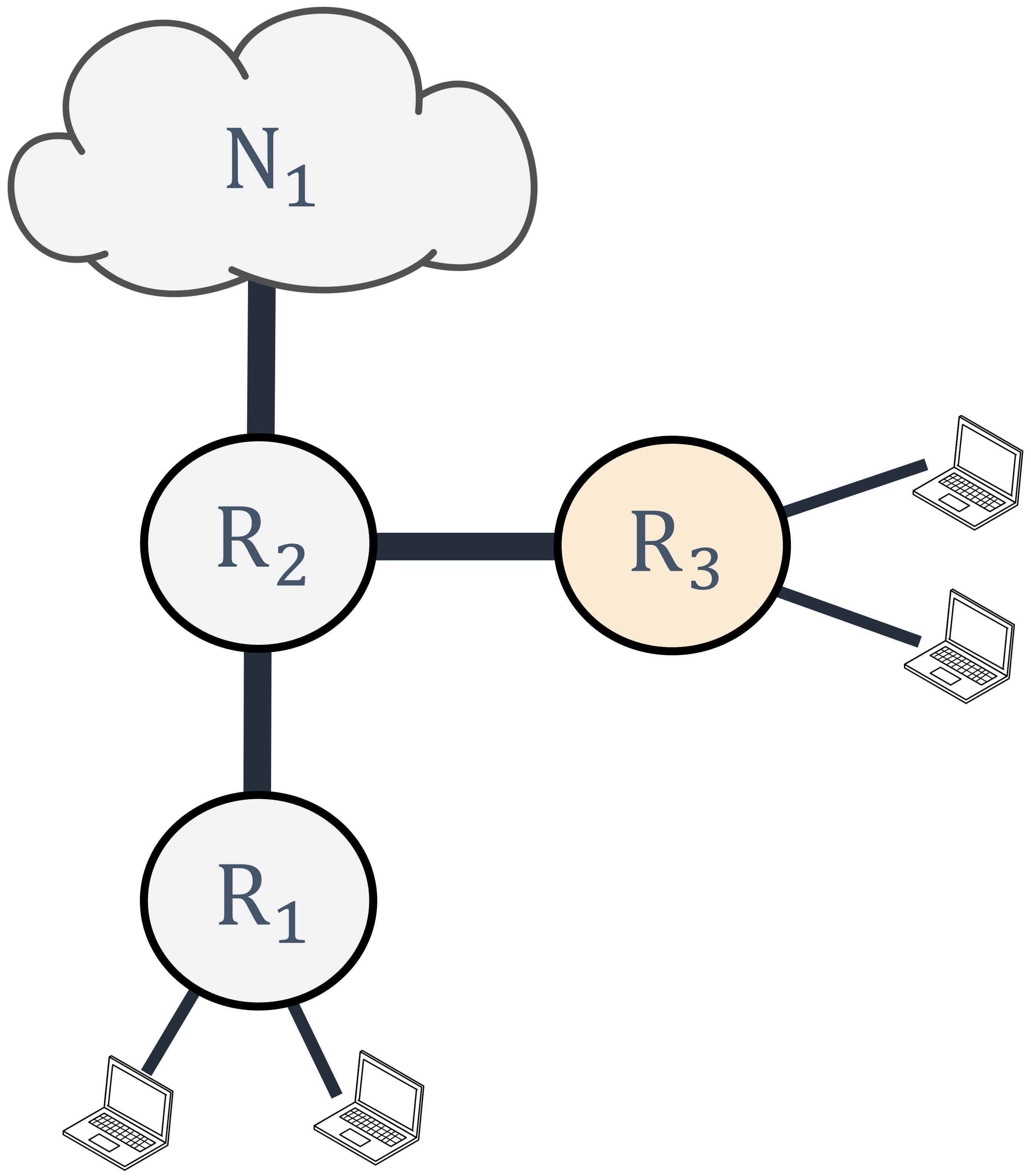
Preference relation:

$a \prec b \Leftrightarrow a < b$

Transfer function:

$$\text{transfer}(e, a) = \begin{cases} \bot & \text{If } a=15 \\ a + 1 & \text{otherwise} \end{cases}$$

SMT Encoding for Verification



“ Does P hold in the network? ”

Network Encoding (SRP): N

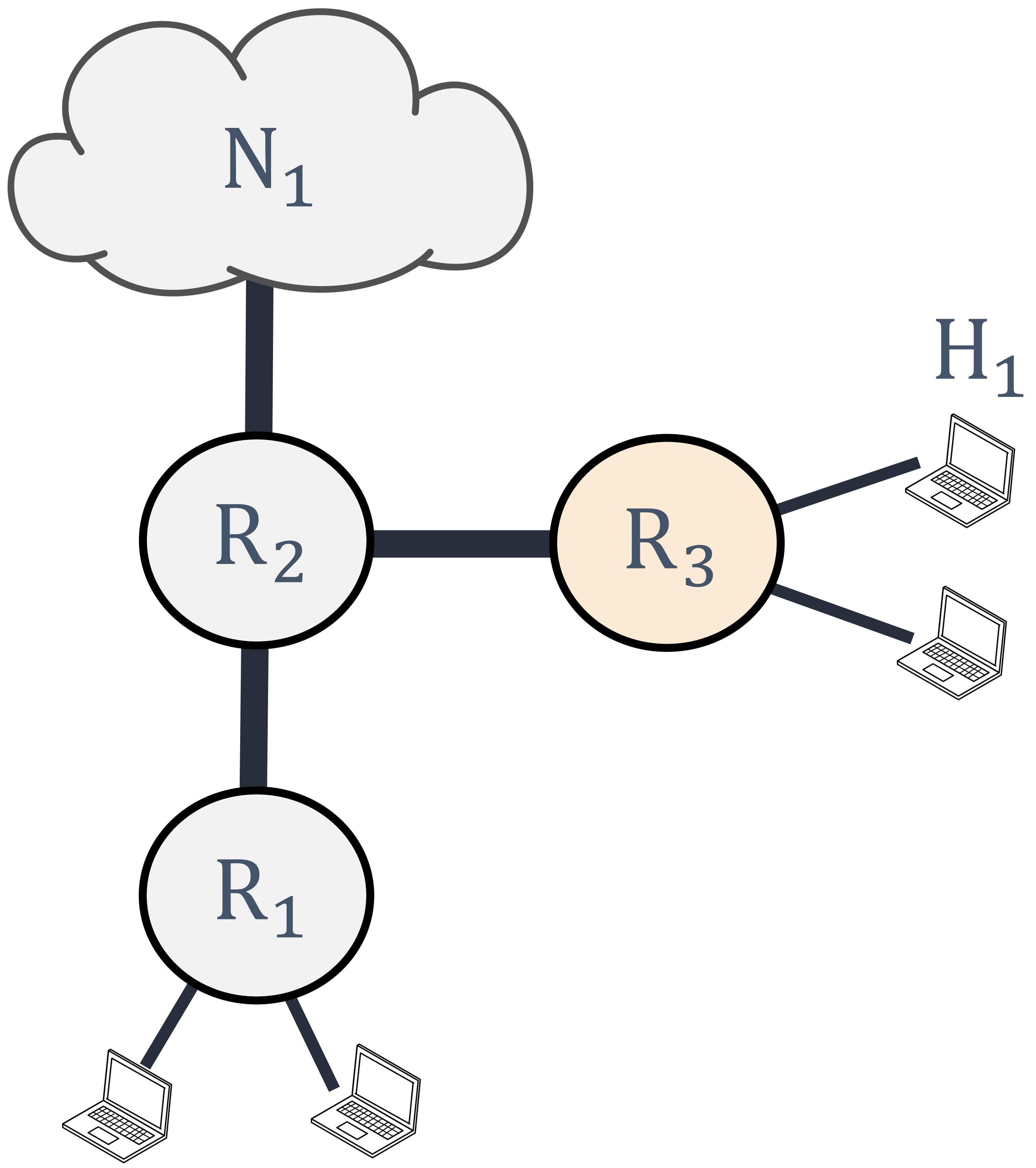
$\wedge$

Network Property (Negated): ¬P

Satisfiable: Property violation

Unsatisfiable: Property holds *for all data planes*

Example: Reachability Property



“ Can router R1 reach host H1? ”

$canReach_{R3} \leftrightarrow forwards_{R3,H1}$

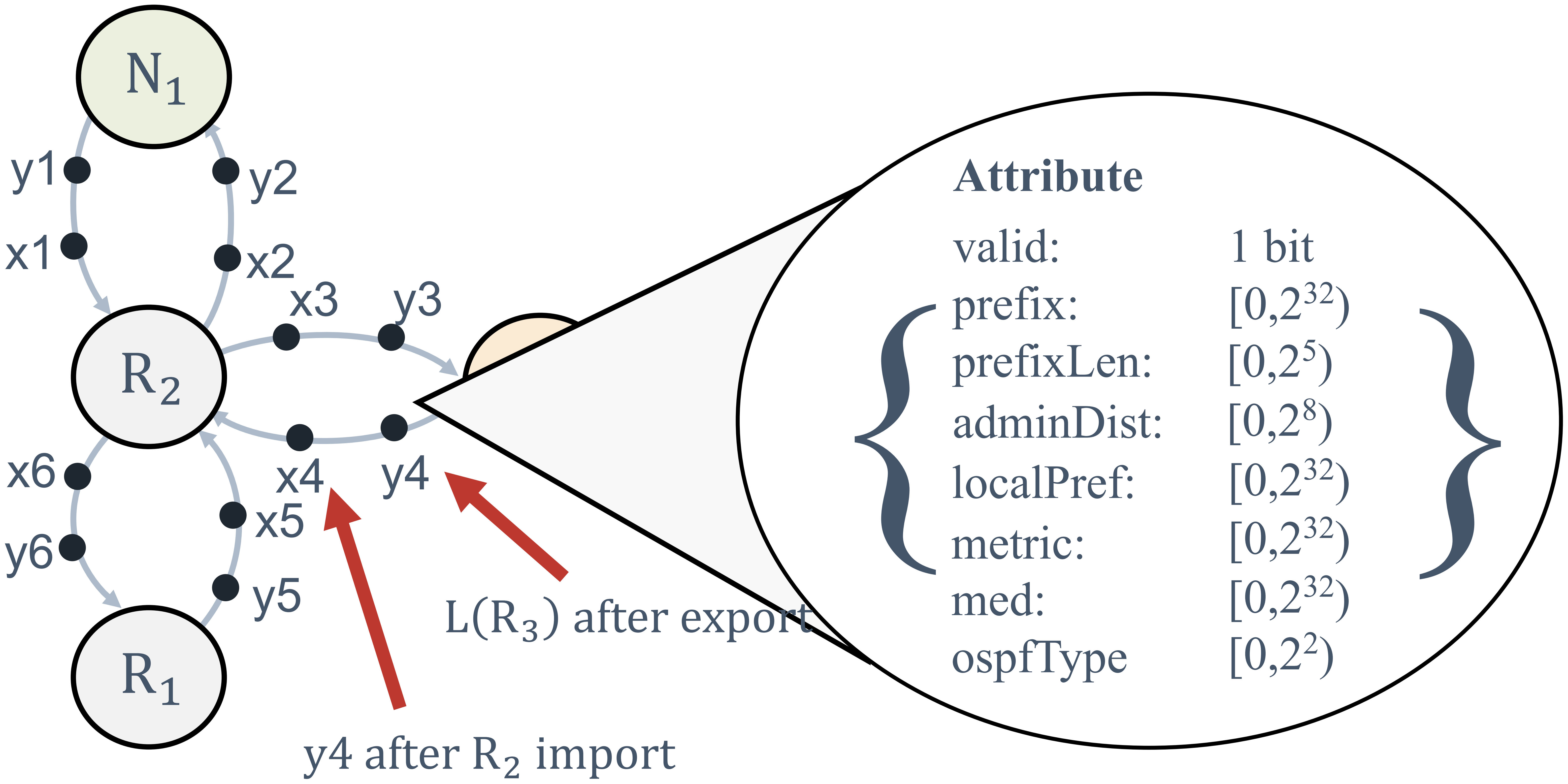
$canReach_{R2} \leftrightarrow$
$(forwards_{R2,R3} \land canReach_{R3}) \lor$
$(forwards_{R2,R1} \land canReach_{R1}) \lor$
$(forwards_{R2,N1} \land canReach_{N1})$

$canReach_{R1} \leftrightarrow forwards_{R1,R2} \land canReach_{R2}$

Property: $canReach_{R1}$

Encoding Transfer Function

Attributes are like states
Transfer function encodes how state is updated along a link



SMT theories: bit vectors, LIA

Common Network Design Features

Features	Implemented
OSPF Intra-area	✓
OSPF Inter-area	✓
eBGP Local-pref	✓
eBGP Communities	✓
eBGP MEDs	✓
eBGP Path Prepending	✓
eBGP Aggregation	✓

Continued…

iBGP	✓
Route Reflectors	✓
Static Routes	✓
Route Redistribution	✓
Multipath Routing	✓
Access Control Lists	✓
IPV6	✗

Properties Supported

Properties	Implemented
Reachability	✓
Bounded Path Length	✓
Equal Path Lengths	✓
Disjoint Paths	✓
Multipath Consistency	✓
Routing Loops	✓
Black Holes	✓
ECMP Load Balancing	✓
Router Equivalence	✓

- Minesweeper Limitations:
  Does not support **convergence, quantitative/probabilistic properties**
  Checking **multiple destinations** is expensive

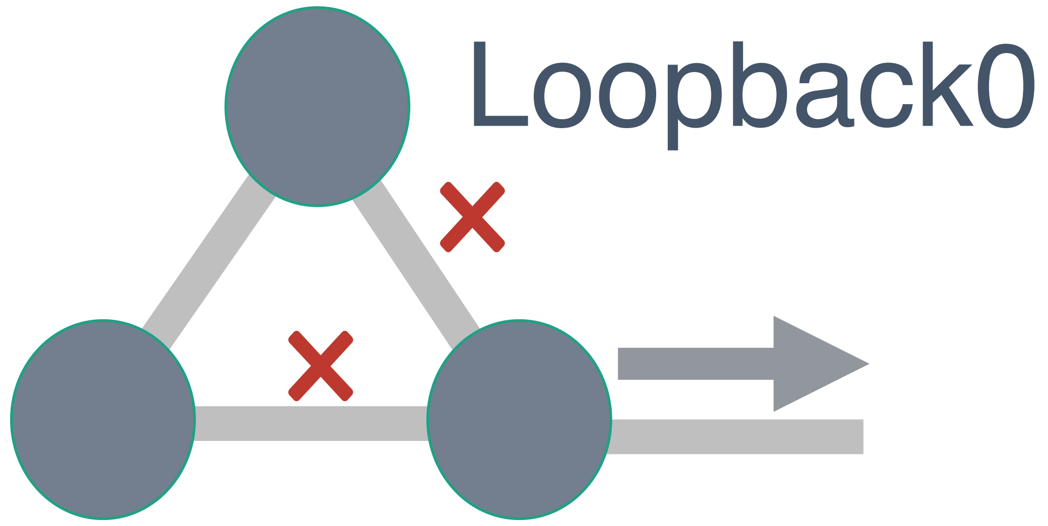
Minesweeper Evaluation

- Can Minesweeper find real bugs?
  - Ran on a collection of 152 legacy networks
  - 1—23K lines of configuration each

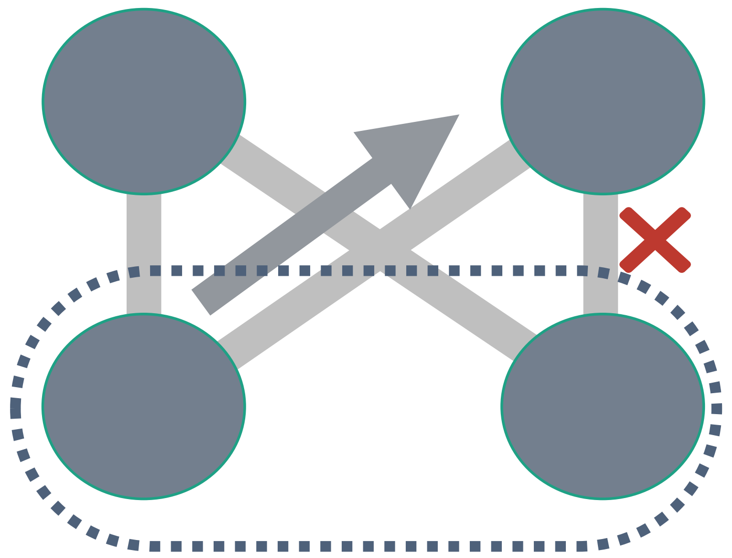
- How well does Minesweeper scale?
  - Tested on a collection of synthetic data center benchmarks
  - Compared verification time across a wide variety of properties

Evaluation: Bug Finding

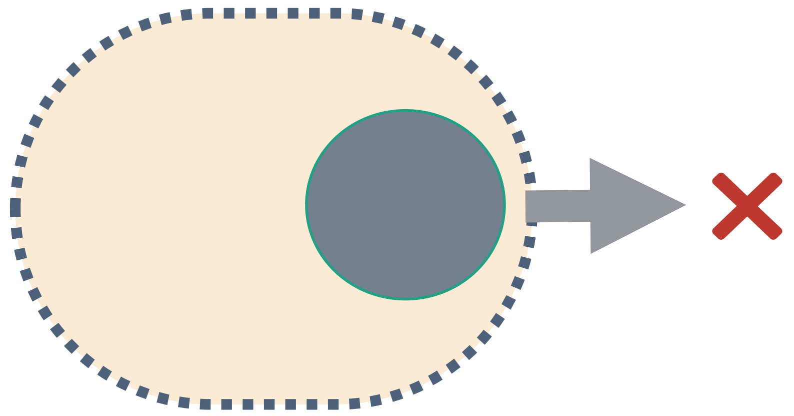
- **Management interface reachability**
  - Found 67 violations of the property



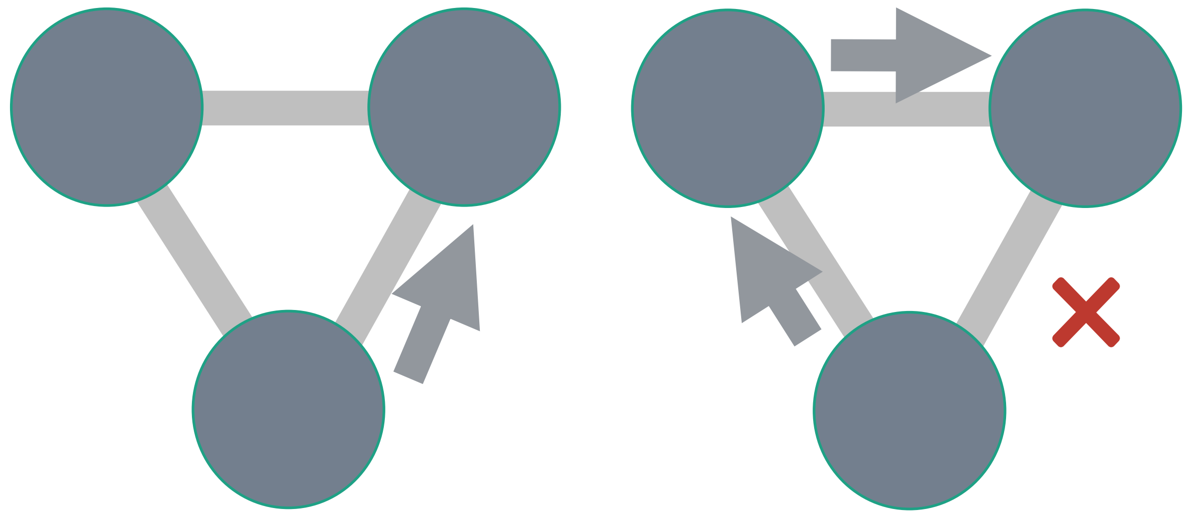
- **Local equivalence of routers**
  - Found 29 violations
  - Example: ACL has missing entry



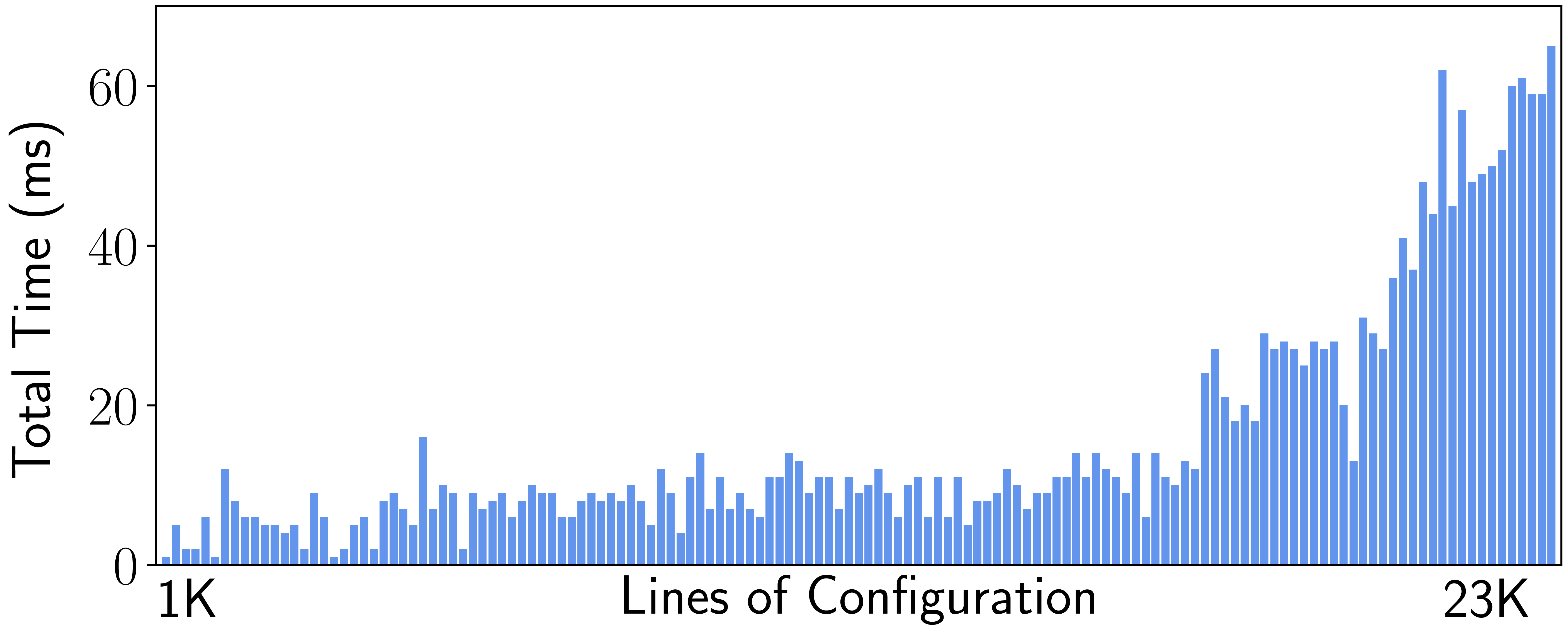
- **Blackholes occur only at the network edge**
  - Found 24 violations of the property



- **Reachability is the same after any 1 failure**
  - Found no violations of the property

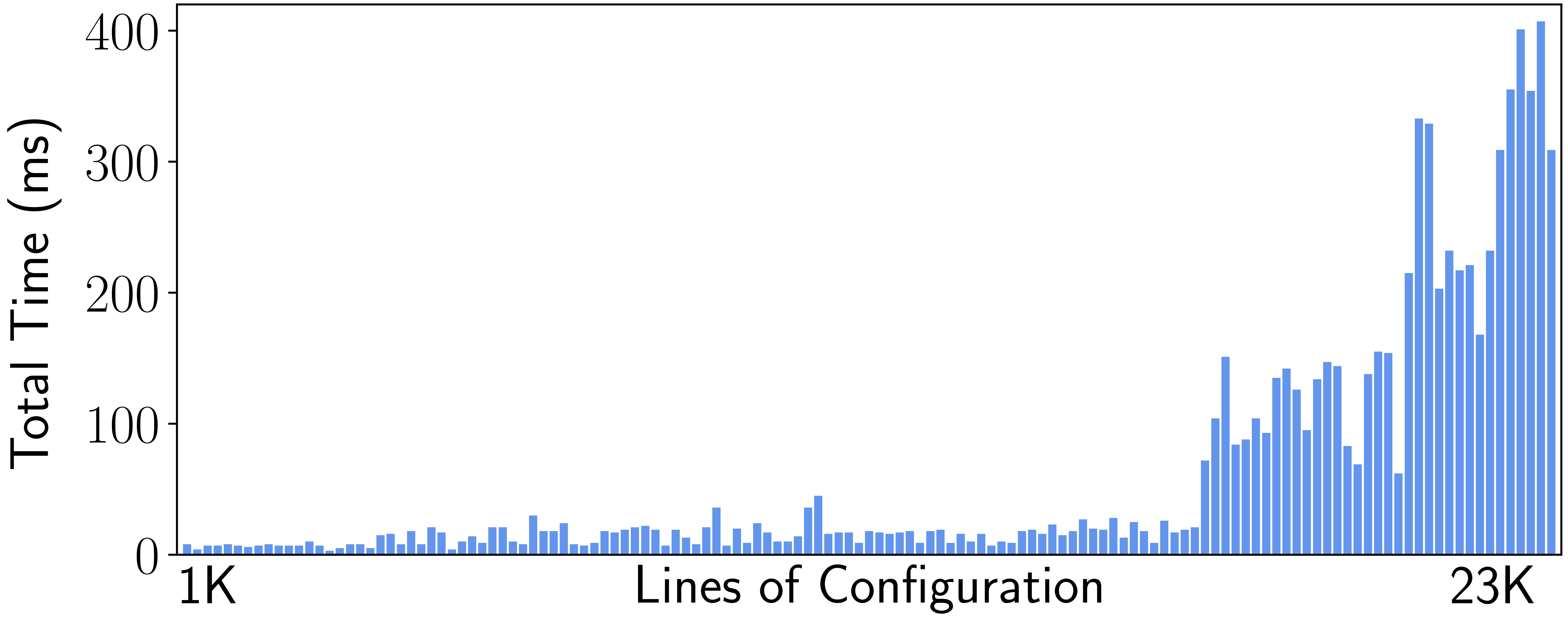


Evaluation Results



Management interface reachability

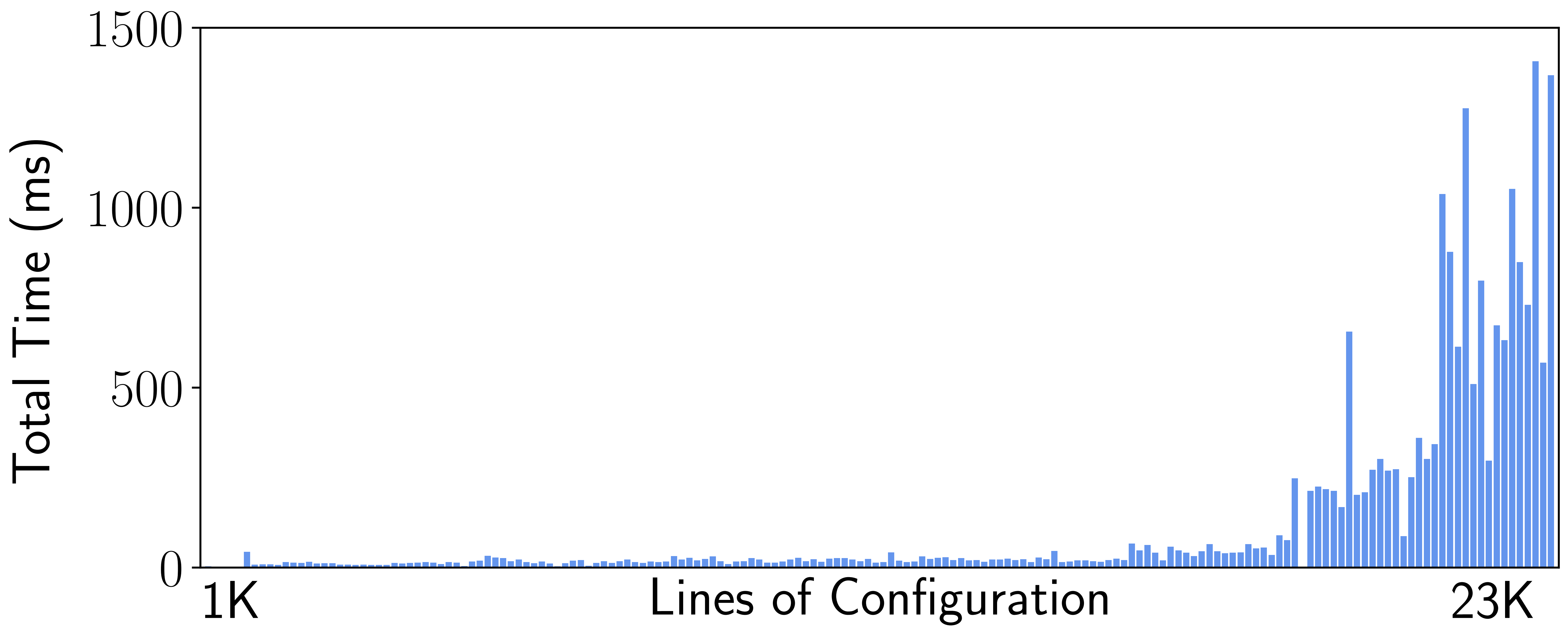
< 60 ms



Local equivalence of routers
(For all n comparisons)

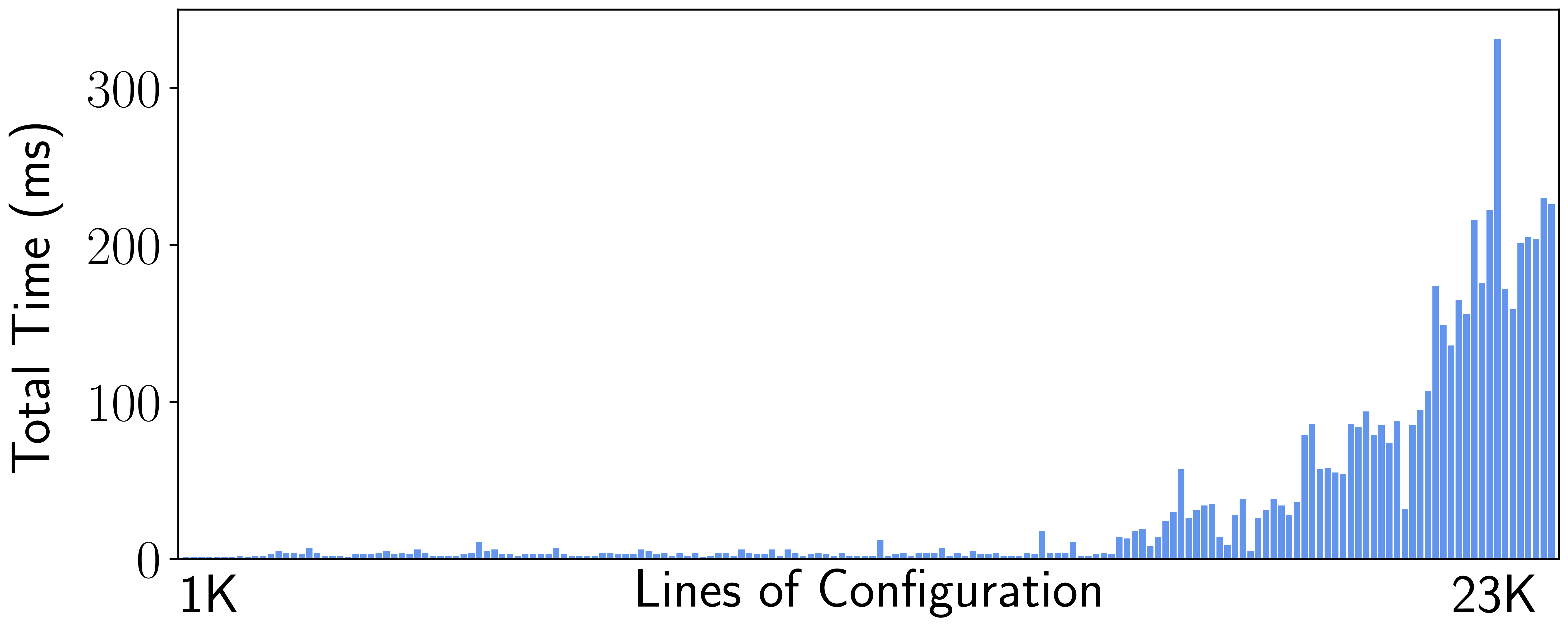
< 400 ms

Evaluation: Scalability



Black holes only occur at the network edge

< 1.5 sec

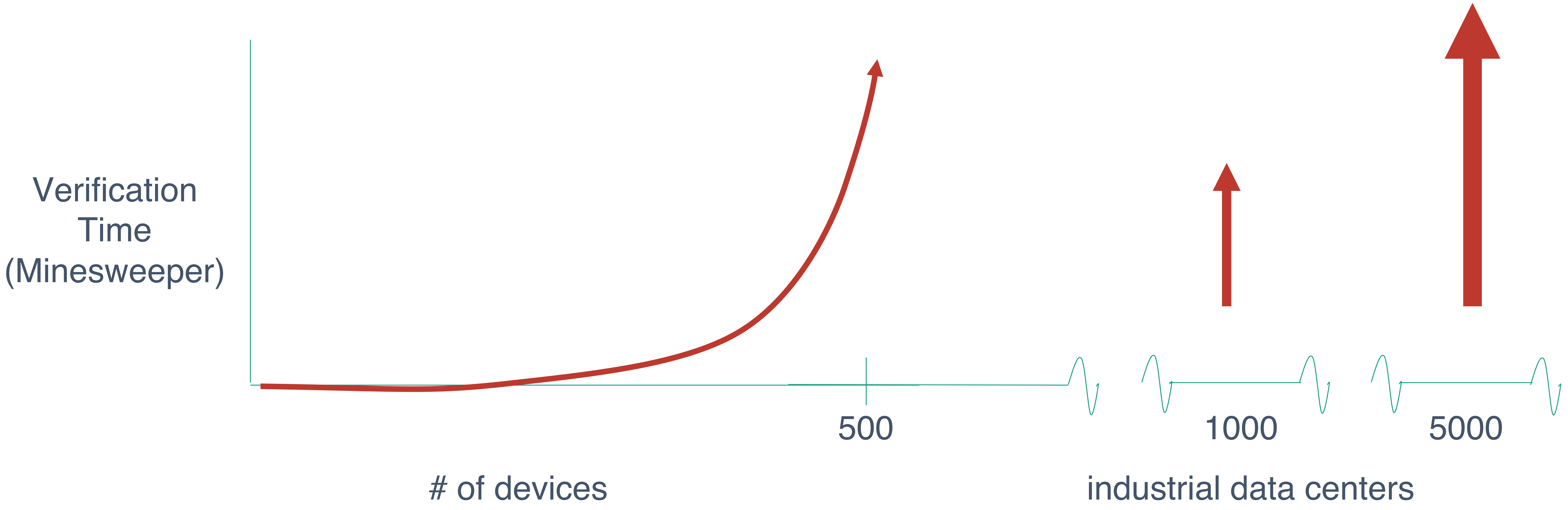


Reachability is the same after any single link failure

< 350 ms

BUT …

A Problem of Scale



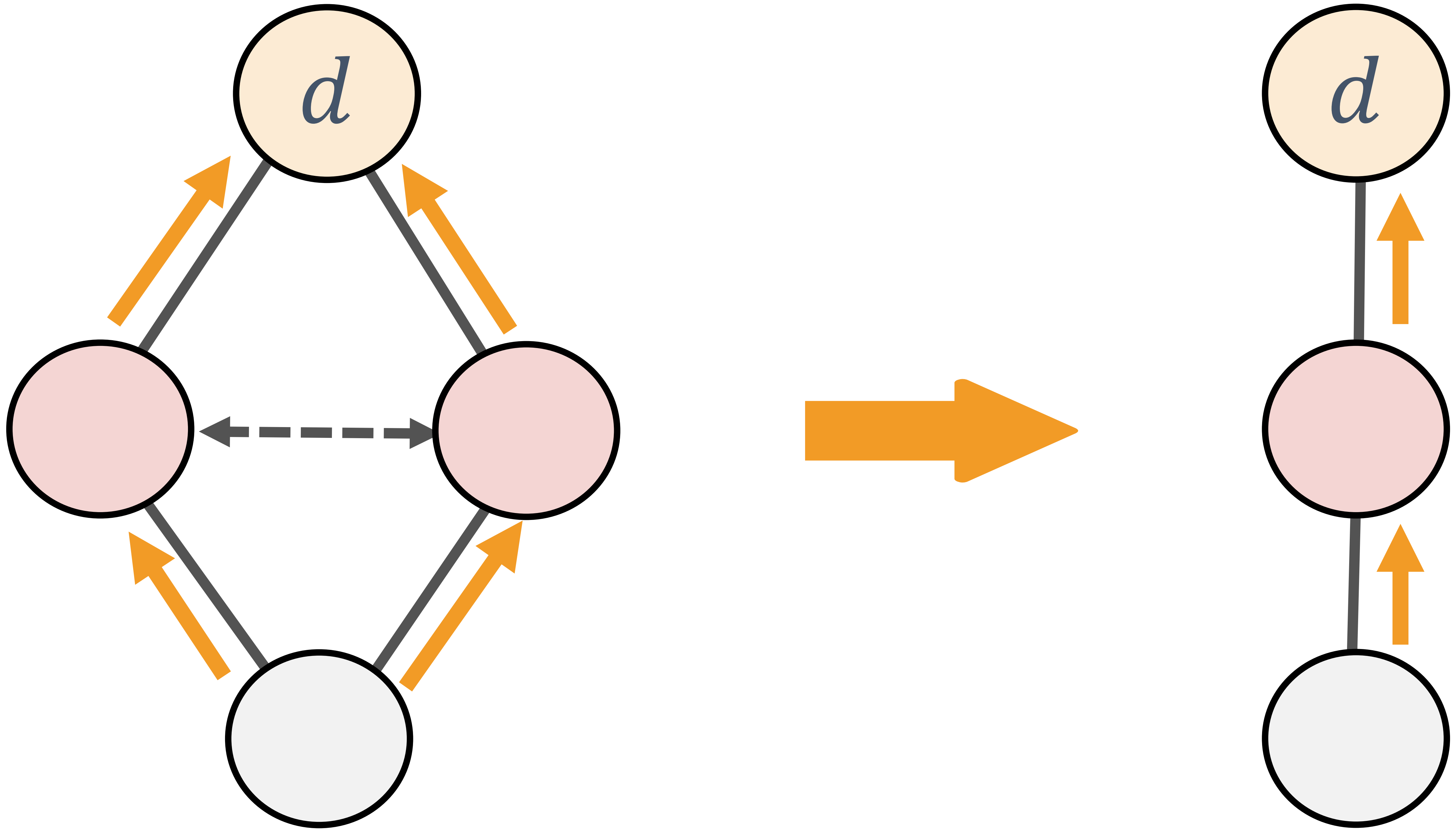
Other technologies, such as simulation, suffer similar, although less severe trends.

Abstraction

Part 1

Ryan Beckett, Aarti Gupta, Ratul Mahajan, David Walker: **Control plane compression**. SIGCOMM 2018: 476-489

Network Abstraction based on Symmetry



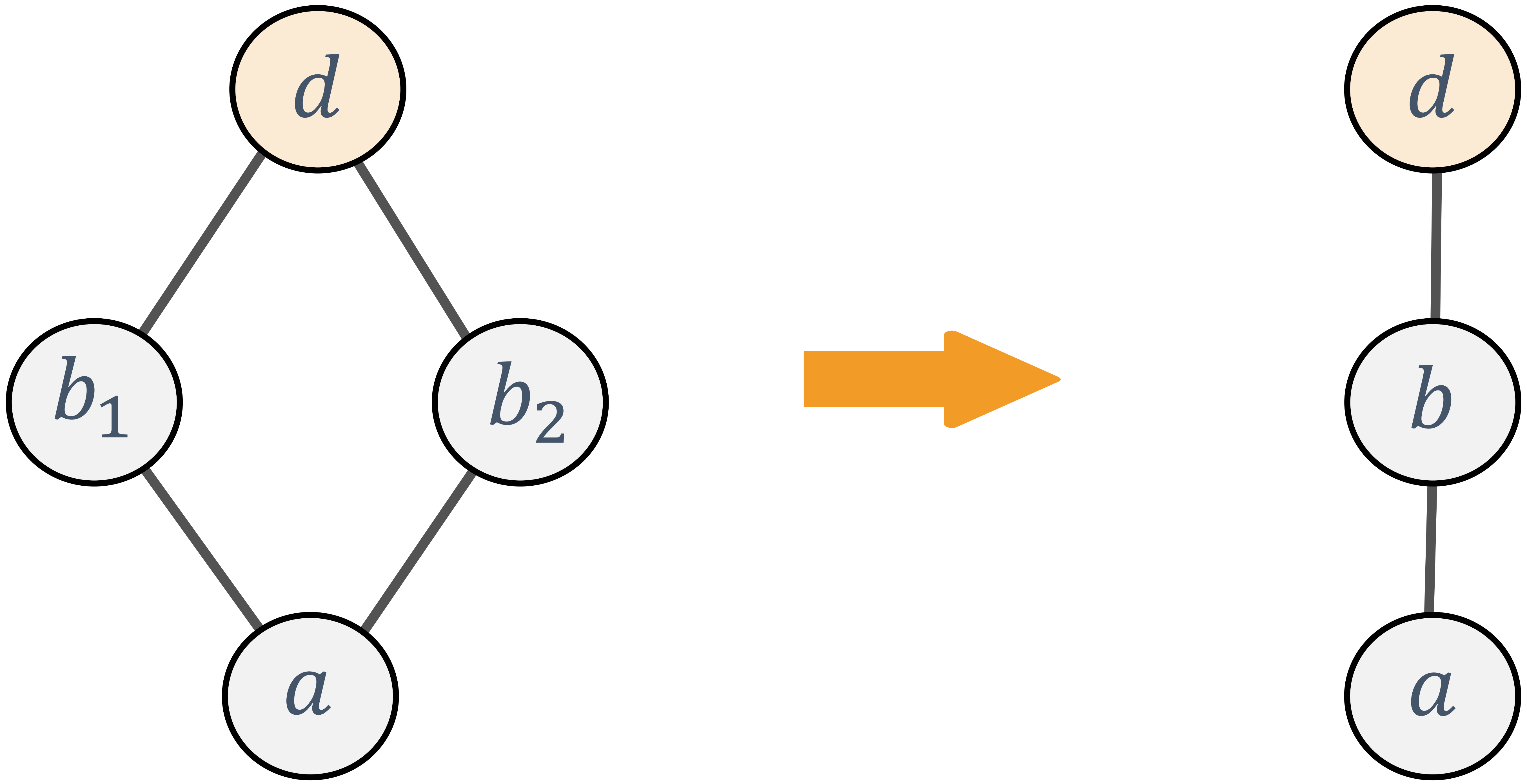
Goal: Compute a small *compressed* network with a “similar” solution to the big one

Formalizing a Compressed SRP

A pair of abstraction functions: (f, h)

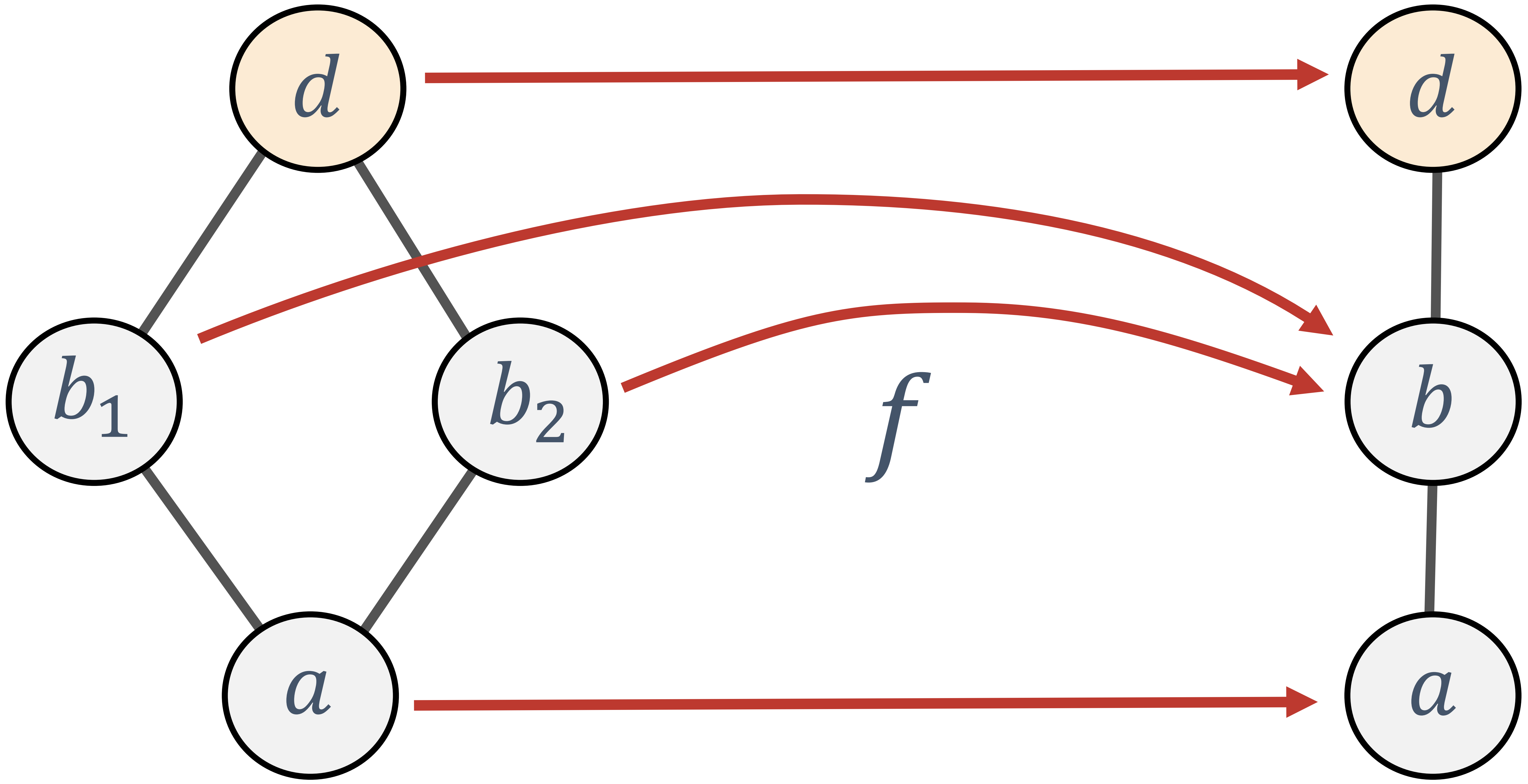
abstracts network topology

abstracts route announcements



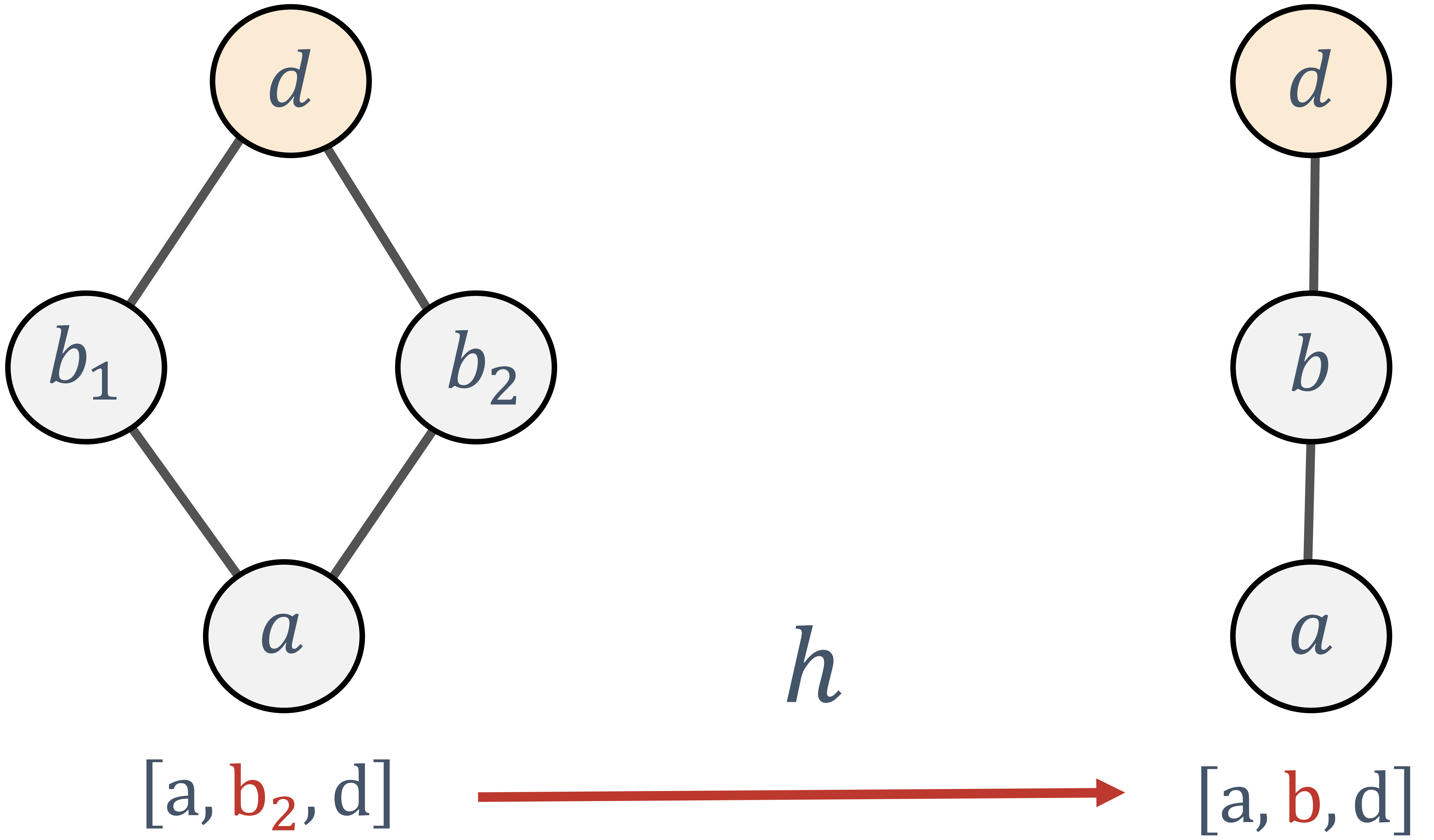
SRP Abstraction

A pair of functions: (f, h)



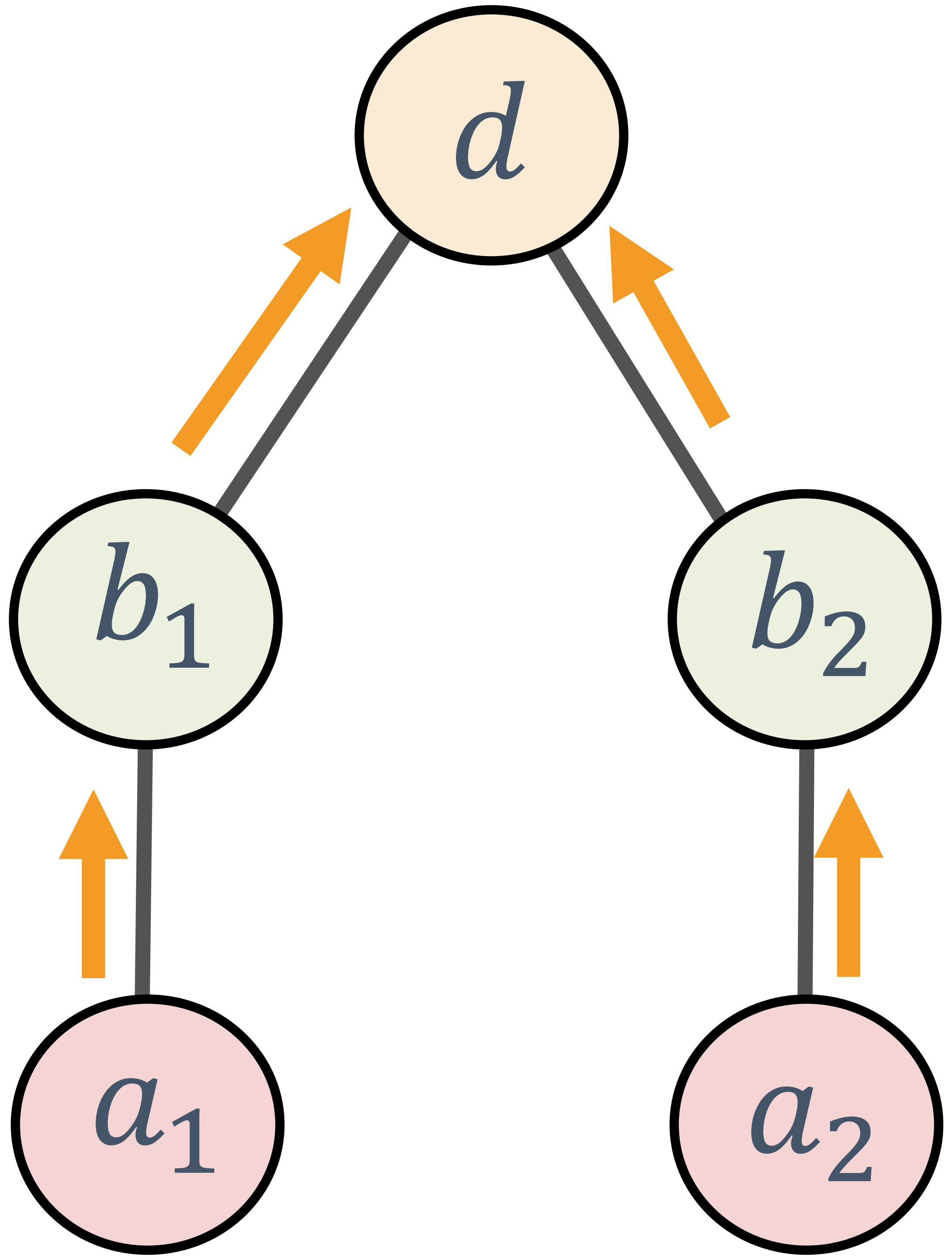
SRP Abstraction

A pair of functions: (f, h)



Soundness of SRP Compression Abstraction

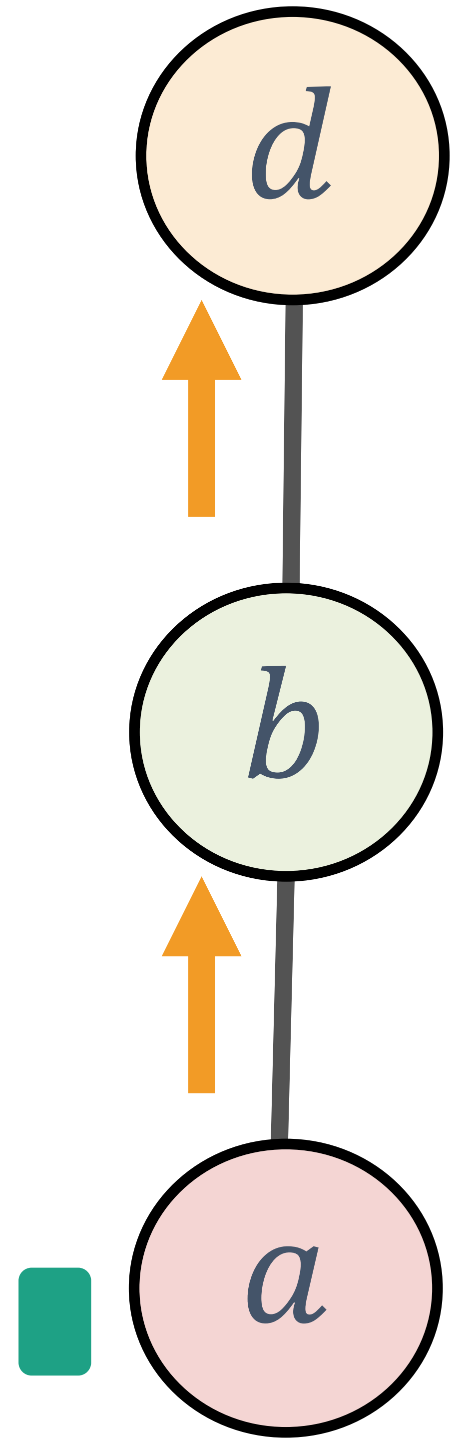
Theorem: If an **abstraction** satisfies certain requirements (forall-exists requirement, transfer equivalence requirement), then it will compute **similar global solutions** as its related concrete network.



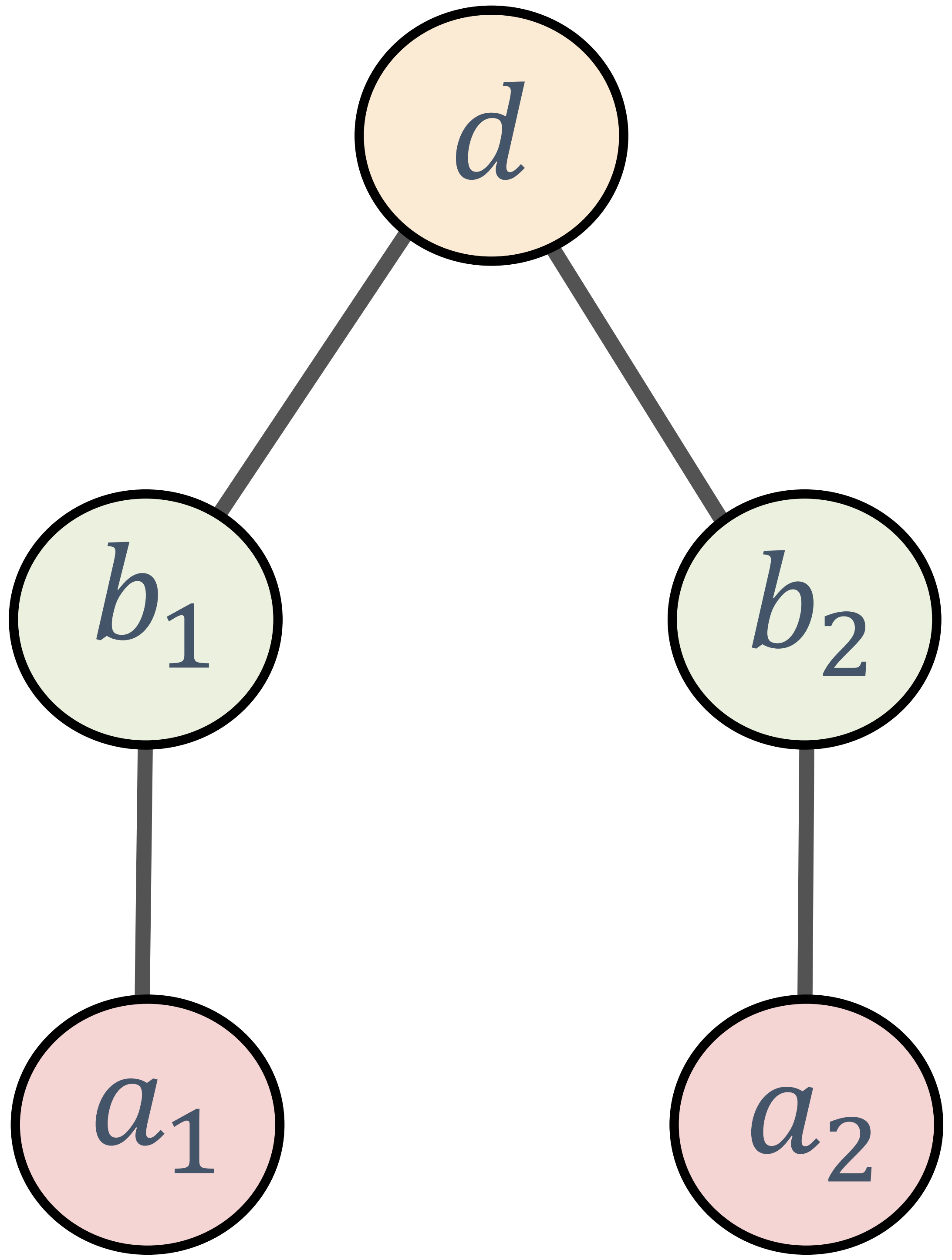
$$\forall v,\ h(L(v)) = L'(f(v))$$

where:
L is a solution in **concrete** SRP
L’ is a solution in **abstract** SRP

similar (modulo h) best routes



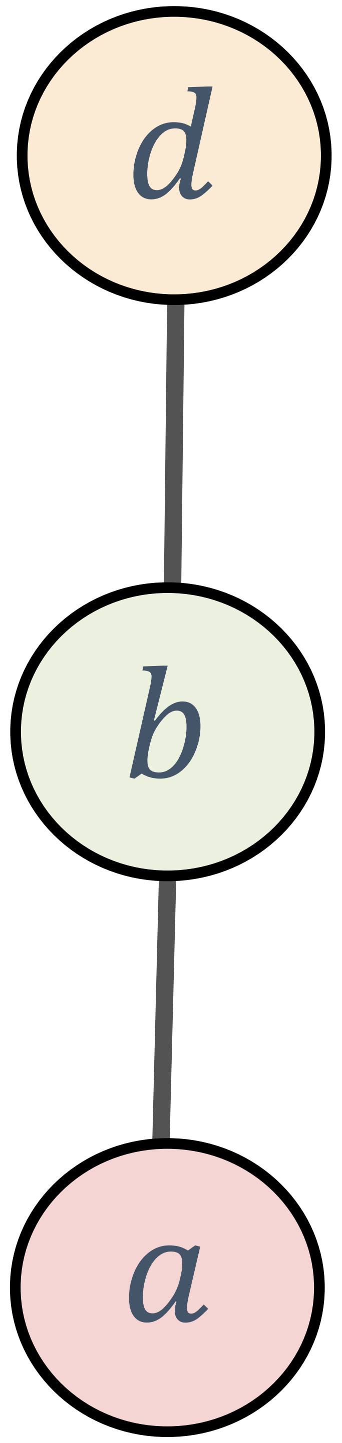
Corollary



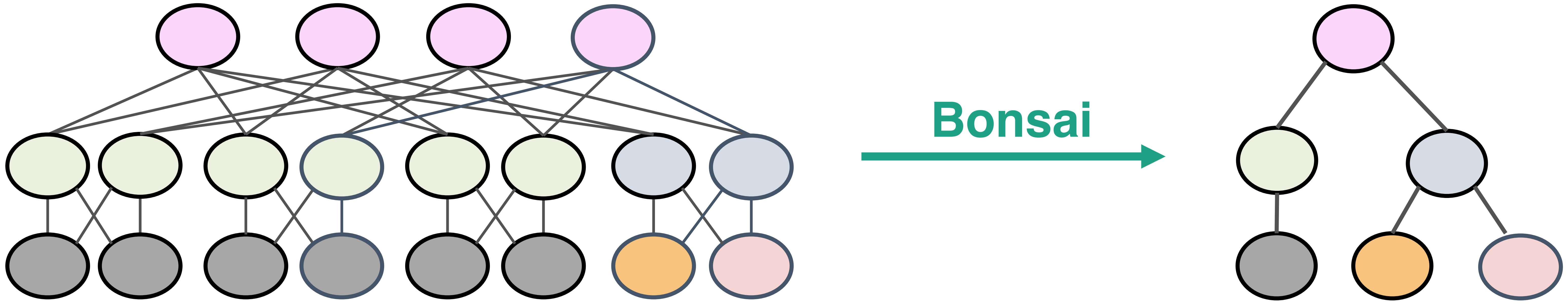
Valid abstractions preserve:

(1) Reachability
(2) Routing Loops
(3) Hop Count
(4) Multipath Consistency
(5) Waypointing

But not fault-tolerance



Bonsai: Control Plane Compression

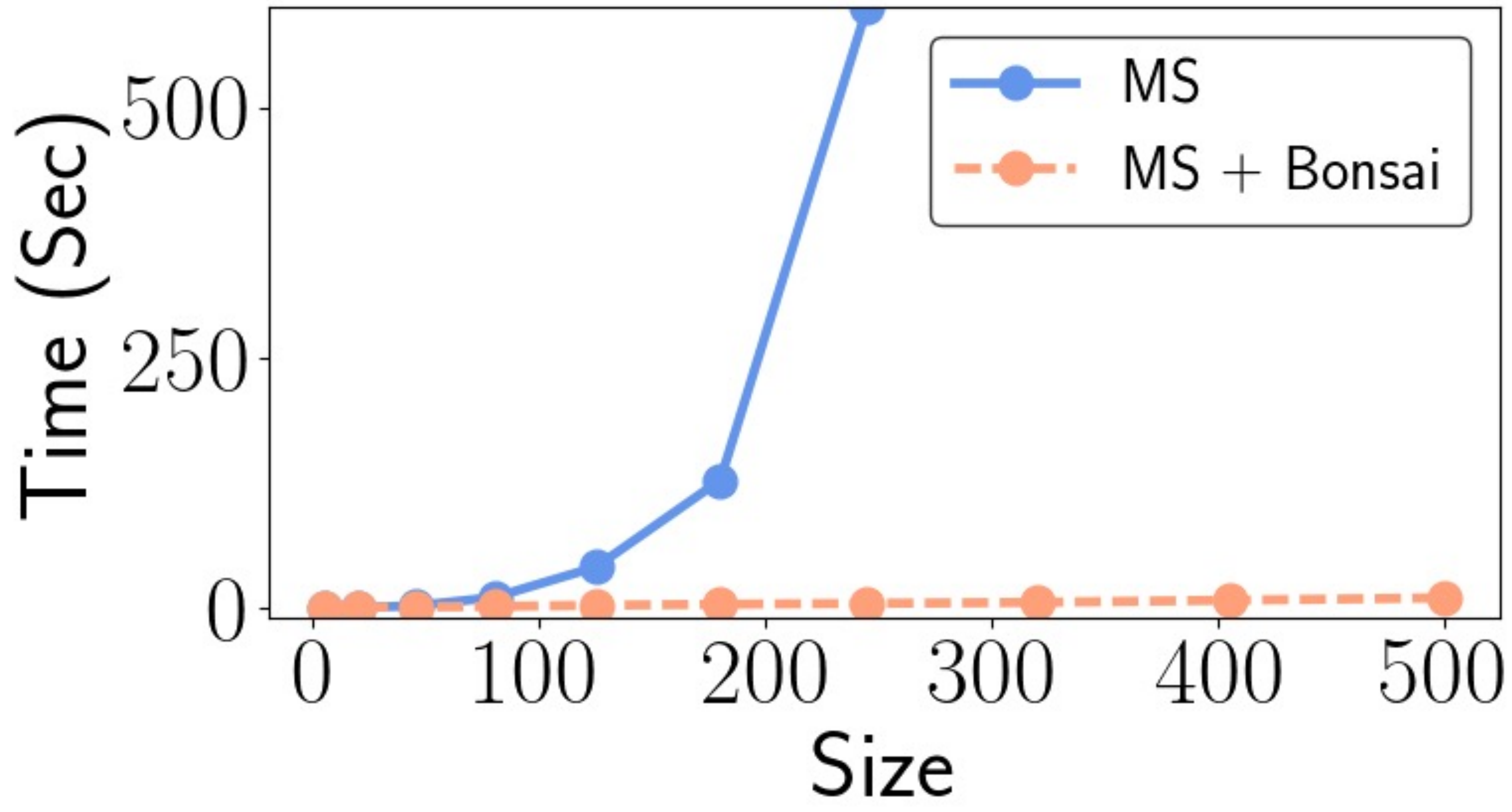


- The Bonsai algorithm compresses real networks by a factor of 5-7 in the number of nodes and 5-100 in the number of edges.
- It preserves many path properties, such as reachability, but not fault tolerance.
- We have proven it correct for a wide range of routing protocols.

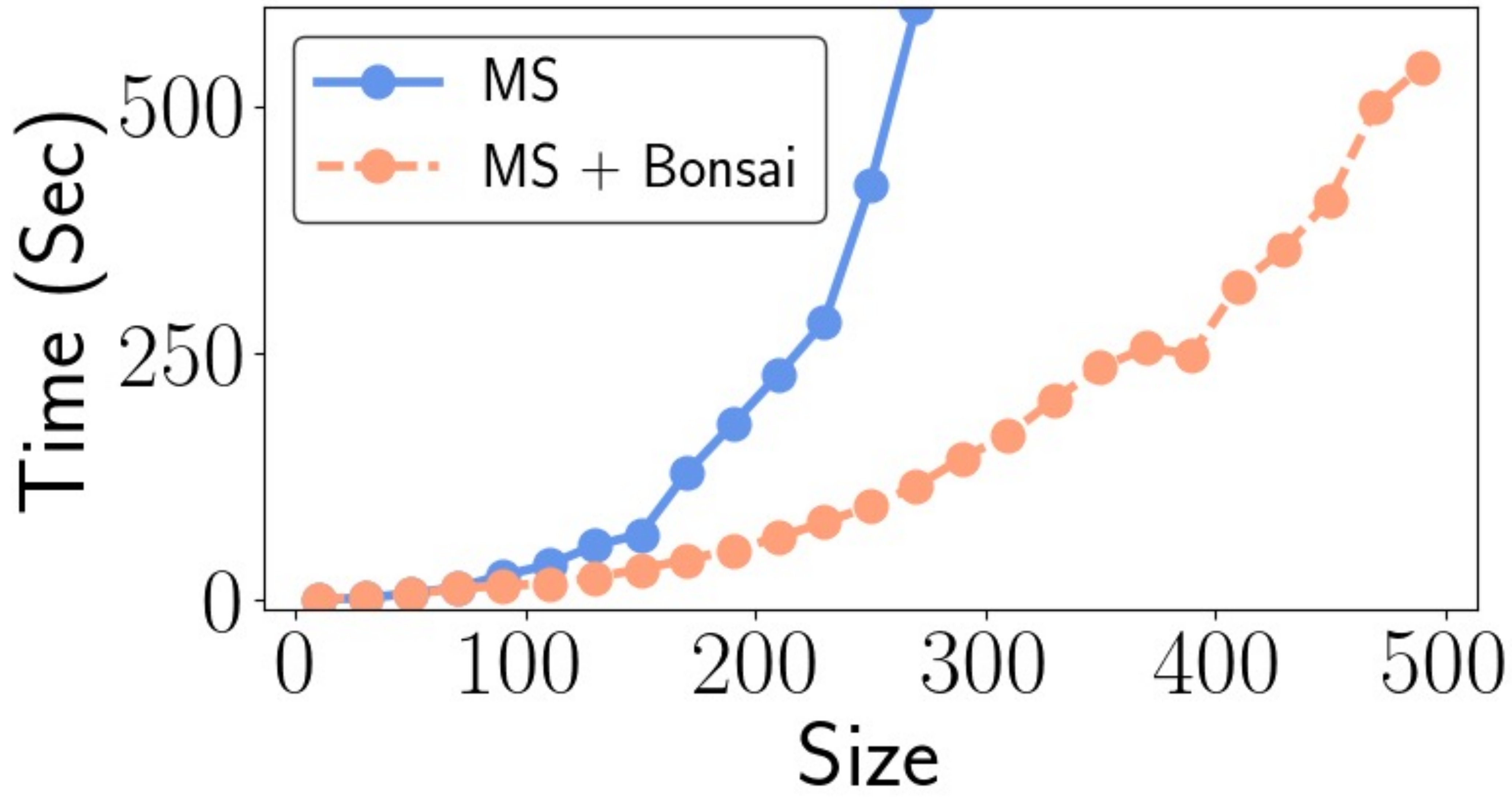
Synthetic Benchmarks

[MineSweeper verifying all-pairs reachability]

Fattree



Ring



BUT …

what if there is no topological symmetry?

Abstraction

Part 2

Ryan Beckett, Aarti Gupta, Ratul Mahajan, David Walker:
Abstract interpretation of distributed network control planes. POPL 2020

Leveraging Abstract Interpretation

Use abstraction on route announcements to combat complexity

- Formalize theory of abstraction for routing protocols
  - Combine abstract interpretation … [Cousot and Cousot 1977]
  - … with the theory of routing algebras. [Sobrinho 2005]

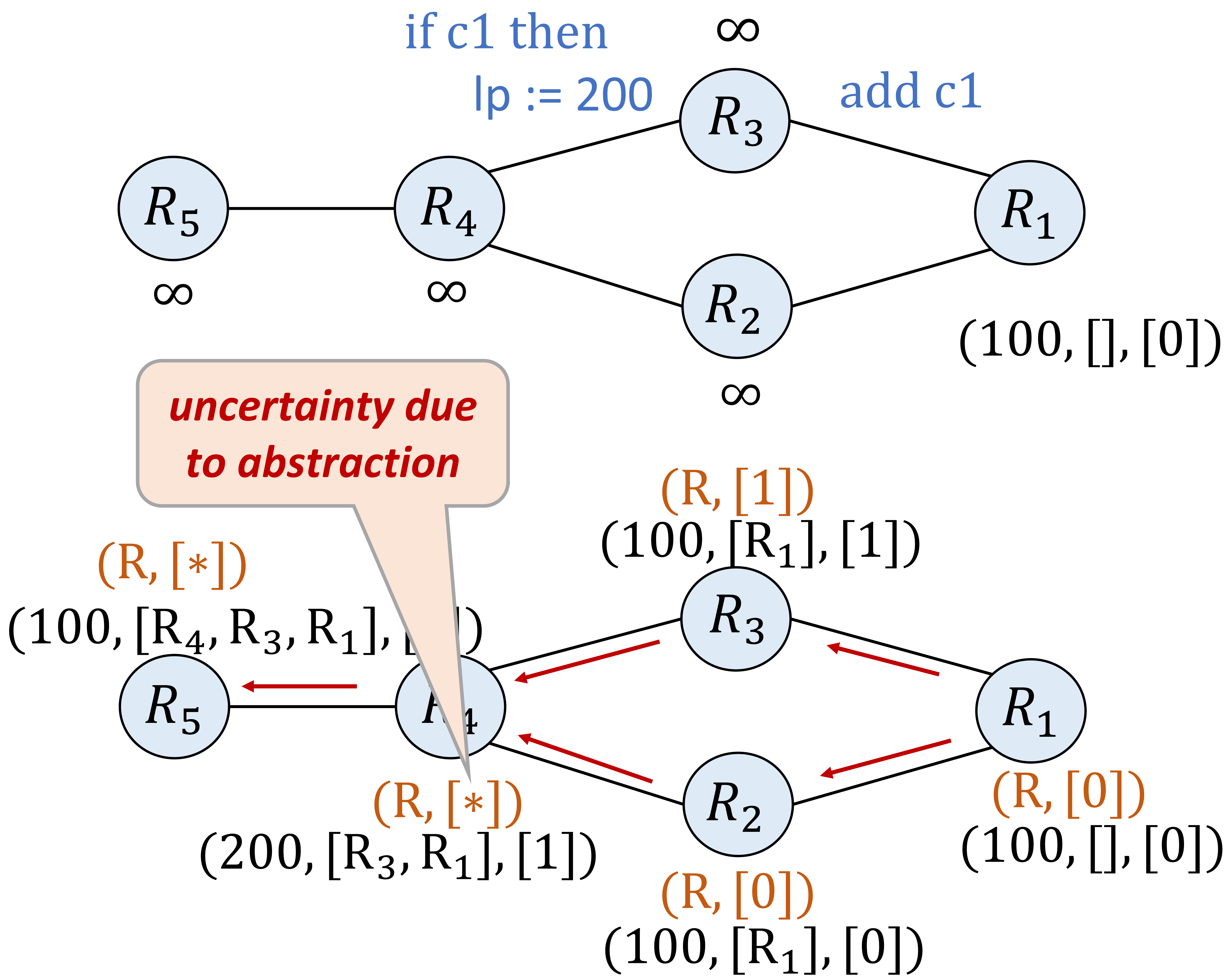
Routing algebra basics

Topology: (V, E)

Algebra: $(S, \oplus, F, 0, \infty)$

- set of routes: S
- merge: $S \to S \to S$
- transfer: F, $E \to S \to S$
- initial route: 0
- invalid route: ∞

Example: Abstracting (simplified) BGP



- Routing algebra:
  (R, ⊕, F , 0, ∞)
- Routing messages R:
  (local_pref, path, comm)
- Transfer F on edge (i, j):
  add j to path, local_pref update

- ***Abstract*** routing messages:
  (R, comm*)
  R: reachability marker
  comm*: (0, 1, *} for each c

Abstractions as ***abstract*** routing algebras

Concrete Algebra $A = (S, \oplus, F, 0, \infty)$

Abstract Algebra $A^{\#} = (S^{\#}, \oplus^{\#}, F^{\#}, 0^{\#}, \infty^{\#})$

	Property	Definition
(1)	$\oplus$, $\oplus^\sharp$ commutative	$c \oplus d = d \oplus c$
(2)	$\oplus$, $\oplus^\sharp$ associative	$c \oplus (d \oplus e) = (c \oplus d) \oplus e$
(3)	$\oplus^\sharp$ monotone	$a \sqsubseteq^\sharp c \wedge b \sqsubseteq^\sharp d \implies a \oplus^\sharp b \sqsubseteq^\sharp c \oplus^\sharp d$
(4)	$\oplus$,$\oplus^\sharp$ sound for α	$\alpha(c \oplus d) \sqsubseteq^\sharp \alpha(c) \oplus^\sharp \alpha(d)$
(5)	F, $F^\sharp$ sound for α	$\alpha(f(c)) \sqsubseteq^\sharp f^\sharp(\alpha(c))$
(6)	$F^\sharp$ monotone	$a \sqsubseteq^\sharp b \implies f^\sharp(a) \sqsubseteq^\sharp f^\sharp(b)$

Soundness theorems

Theorem 1: For a fixed asynchronous schedule, an execution of abstract algebra $\mathrm{A}^{\#}$ is sound with respect to an execution of concrete algebra A.

Theorem 2: If $\mathrm{A}^{\#}$ is uniquely converging, then for **any** asynchronous schedule, an execution of abstract algebra $\mathrm{A}^{\#}$ is sound with respect to A.

Abstraction: key benefits

- Abstraction can (often) give precise answers
  - In our experiments, we got precise answers on 95% of real networks
- Abstraction improves performance
  - Tracking less information -> less memory
  - Smaller state space -> fewer iterations to converge -> less time
  - Creates opportunities for sharing (e.g., for multiple destinations)
- Abstraction can enable new network analyses
  - Example: potential hijacking analysis

ShapeShifter: Fast Reachability Analysis

- Goal: allow fast analysis, while minimizing loss in precision on real networks

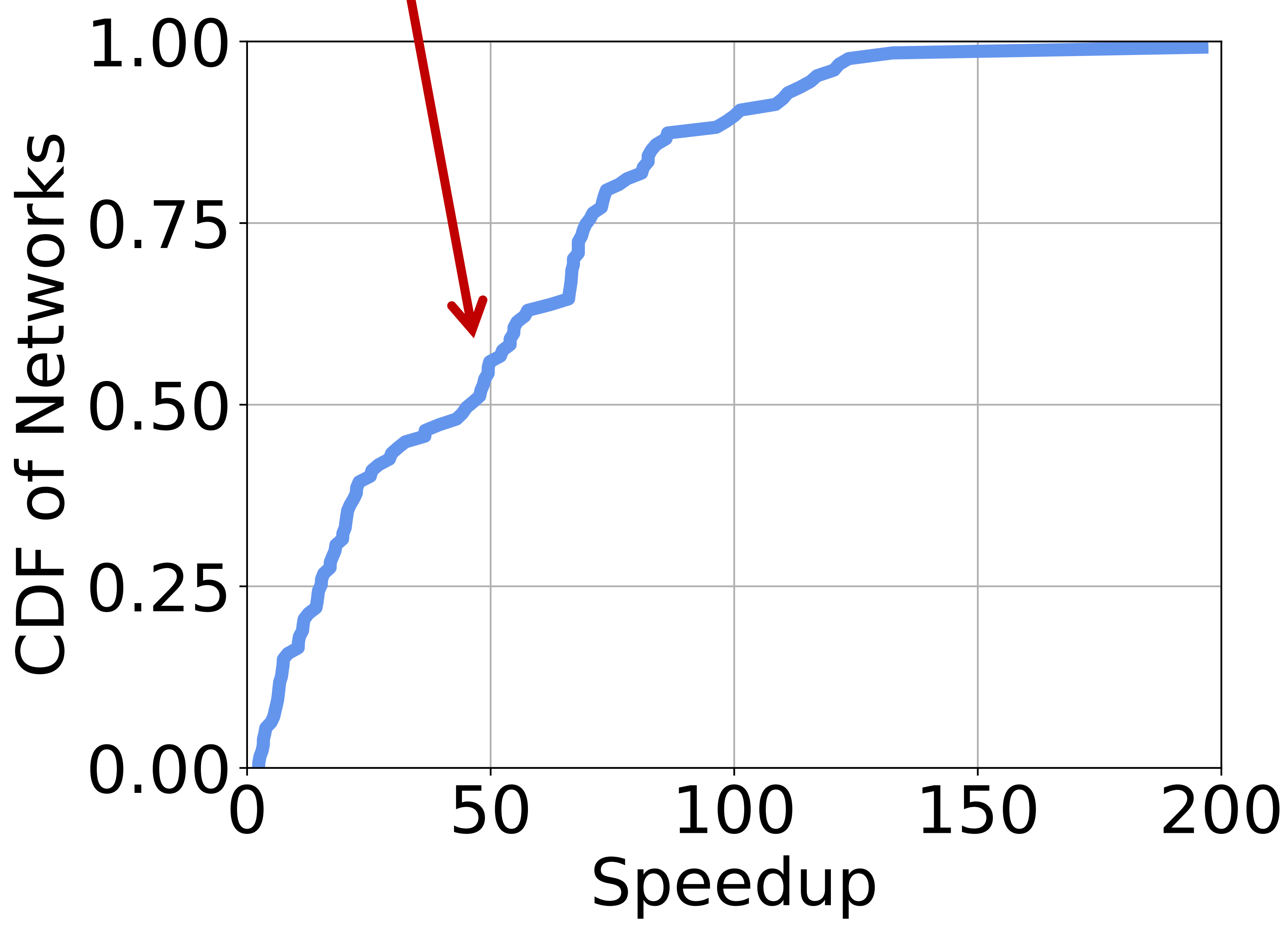
- What to abstract?
  - Throw away: AS-path, protocol decision process variables
  - Keep: BGP communities, BGP origin, protocol used
  - Example map [dest ↦ abstract route] :
    - [168/8 ↦ ([0,*,0], {R1}, {BGP})]
    - Means: “second community may be attached or not, on a route originating from R1 in the BGP protocol”

- Other features
  - BDD-based representations for prefix sets for improved sharing
  - Message scheduling heuristics for improving performance
  - Modeling multiple protocols, iBGP, …

ShapeShifter: Evaluation on real networks

- Abstract simulation finished in less than 40 sec. on each of the 127 networks
  - *1-2 orders of magnitude speedup over Batfish (concrete simulator)*

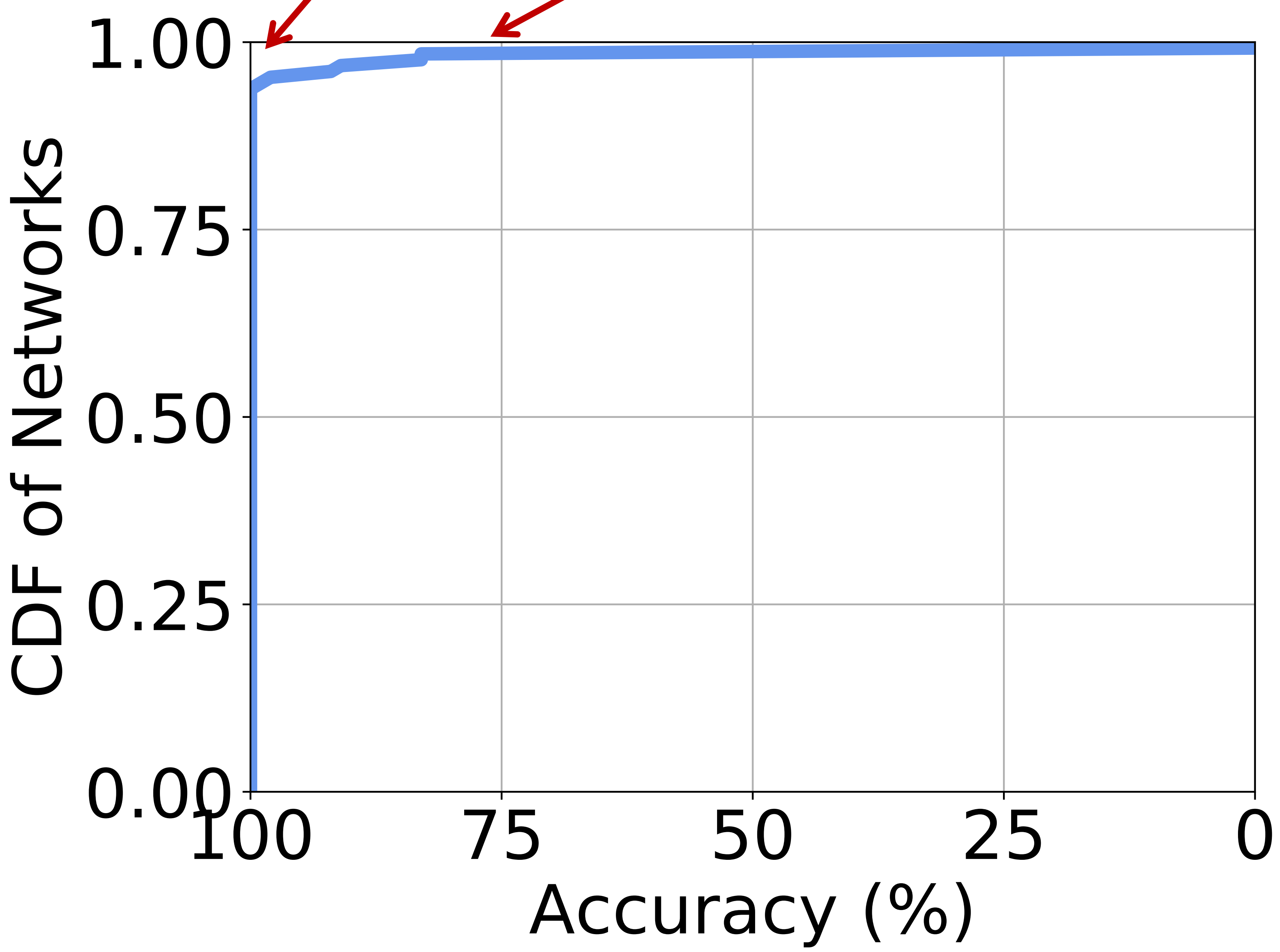
About half of networks have more than 50x speedup



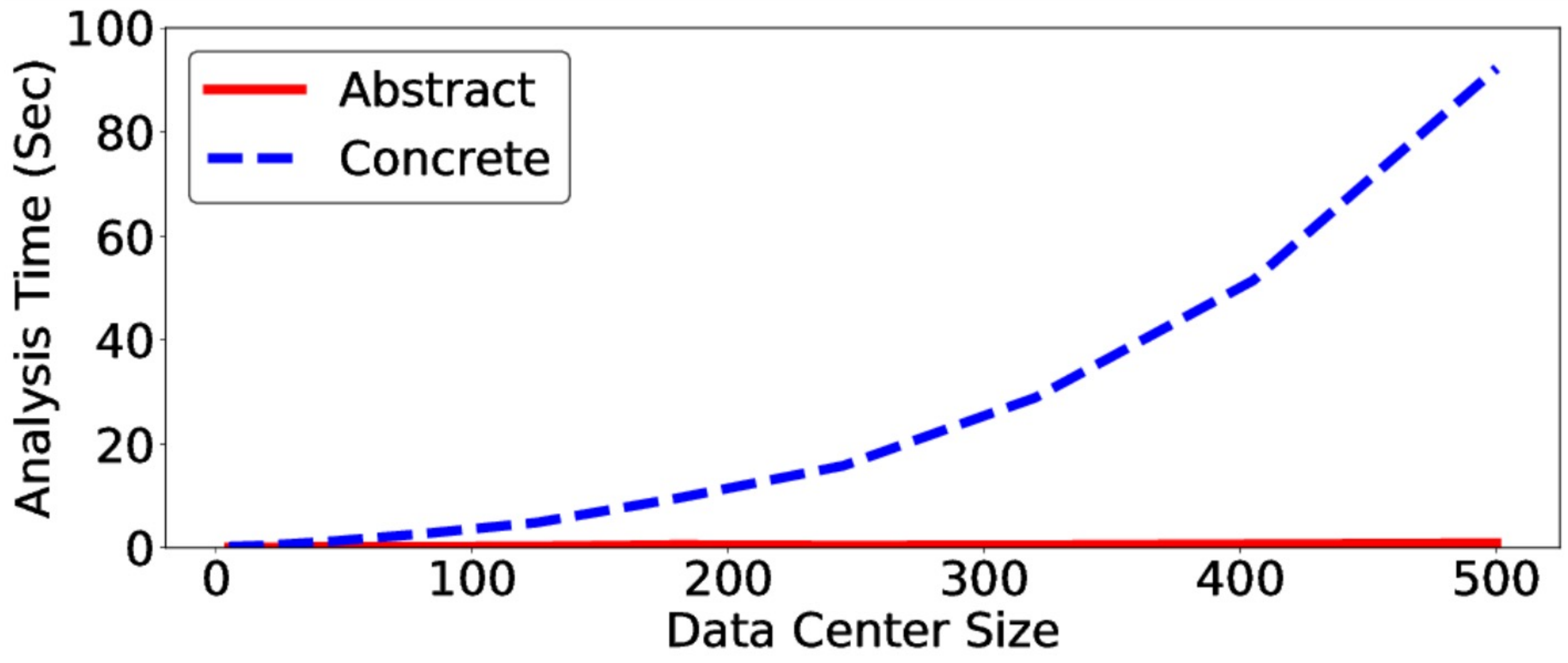
over Batfish

Can prove reachability for all destinations for 95% of networks

For the remaining 5% of networks, can prove reachability for the majority of destinations



ShapeShifter: Synthetic datacenter results



Simulation time vs. data center size
for verifying all-pairs connectivity

BUT …

Can we scale the
general SMT-based approach?

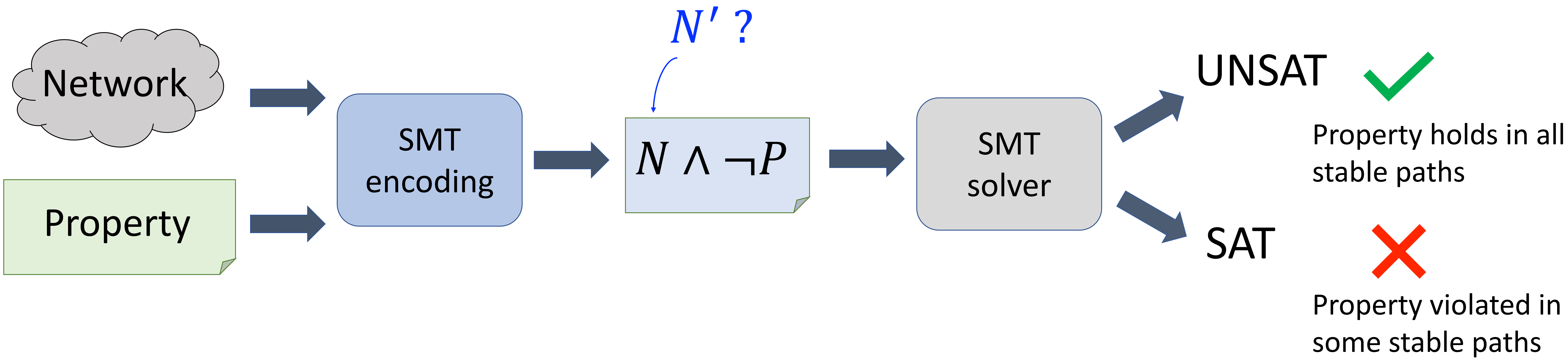
(Analogy: static program analysis vs.
SMT-based program verification)

Abstraction

Part 3

Divya Raghunathan, Ryan Beckett, Aarti Gupta, David Walker:
ACORN: Network Control Plane Abstraction using Route Nondeterminism.
(Under submission)

Inspiration: Minesweeper's SMT-based approach



- N : SMT formula representing network behavior
  - Satisfying assignments of N represent stable paths
- P : SMT formula representing property to be checked

Goal: improve scalability

Two-part Approach

1. Hierarchy of Nondeterministic Routing Choice (NRC) Abstractions

Abstract away route selection while preserving soundness

2. SMT encoding and solvers

Symbolic graph-based encoding [Bayless et al. AAAI 2015]

Use SMT solvers with specialized graph-theories, e.g., MonoSAT

Strategy: Abstract away route selection

- Insight: Verifying certain properties (e.g., reachability) may not need to identify the best route available

- Modeling route selection is expensive
  - Especially for complex protocols like BGP

- One difference between data plane and control plane: route selection
  - Data plane verifiers routinely scale to *several thousands* of routers
  - Question: can we get closer to their performance?

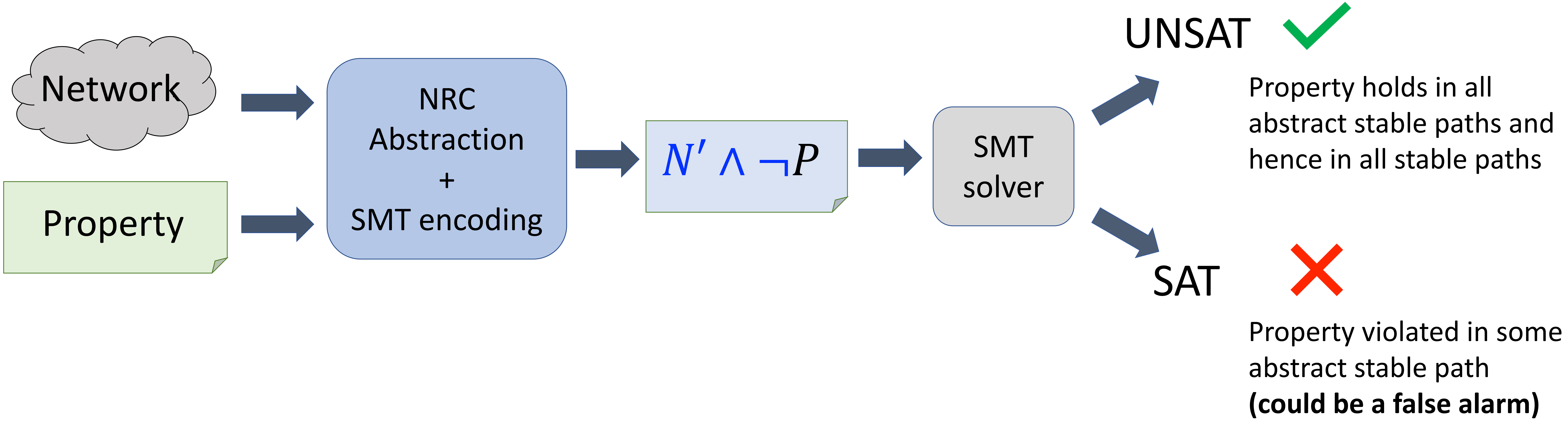
… Yes!

Key Idea: *Nondeterministic* Routing Choice (NRC) Abstractions

- Each router nondeterministically chooses one of the routes received
  - **Any available route, not necessarily the best, may be chosen**
    - Route announcement fields involved only in route selection can be abstracted away
  - **Any chosen route must be compliant with policy**
    - For example, routes filtered based on community tags will not be chosen

- Abstract network model N’ has more routes than real network N
  - N’ includes best routes *and other policy-compliant routes*
  - **N’ overapproximates N**
  - A hierarchy of NRC Abstractions based on routing fields

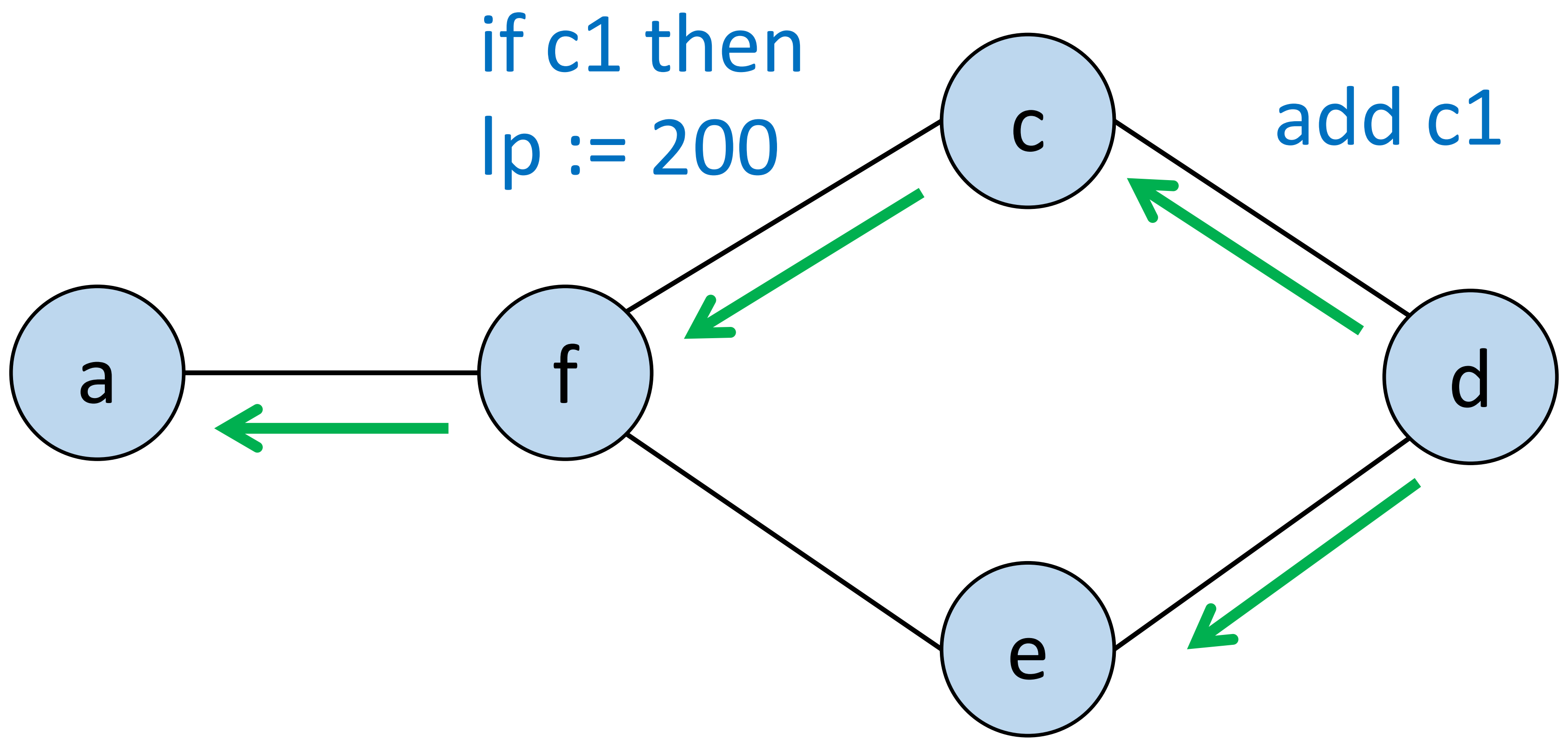
SMT-based verification with NRC abstraction



- N' : SMT formula **overapproximates** network behavior
  - Satisfying assignments of N' represent **abstract** stable paths
- P : SMT formula representing property to be checked

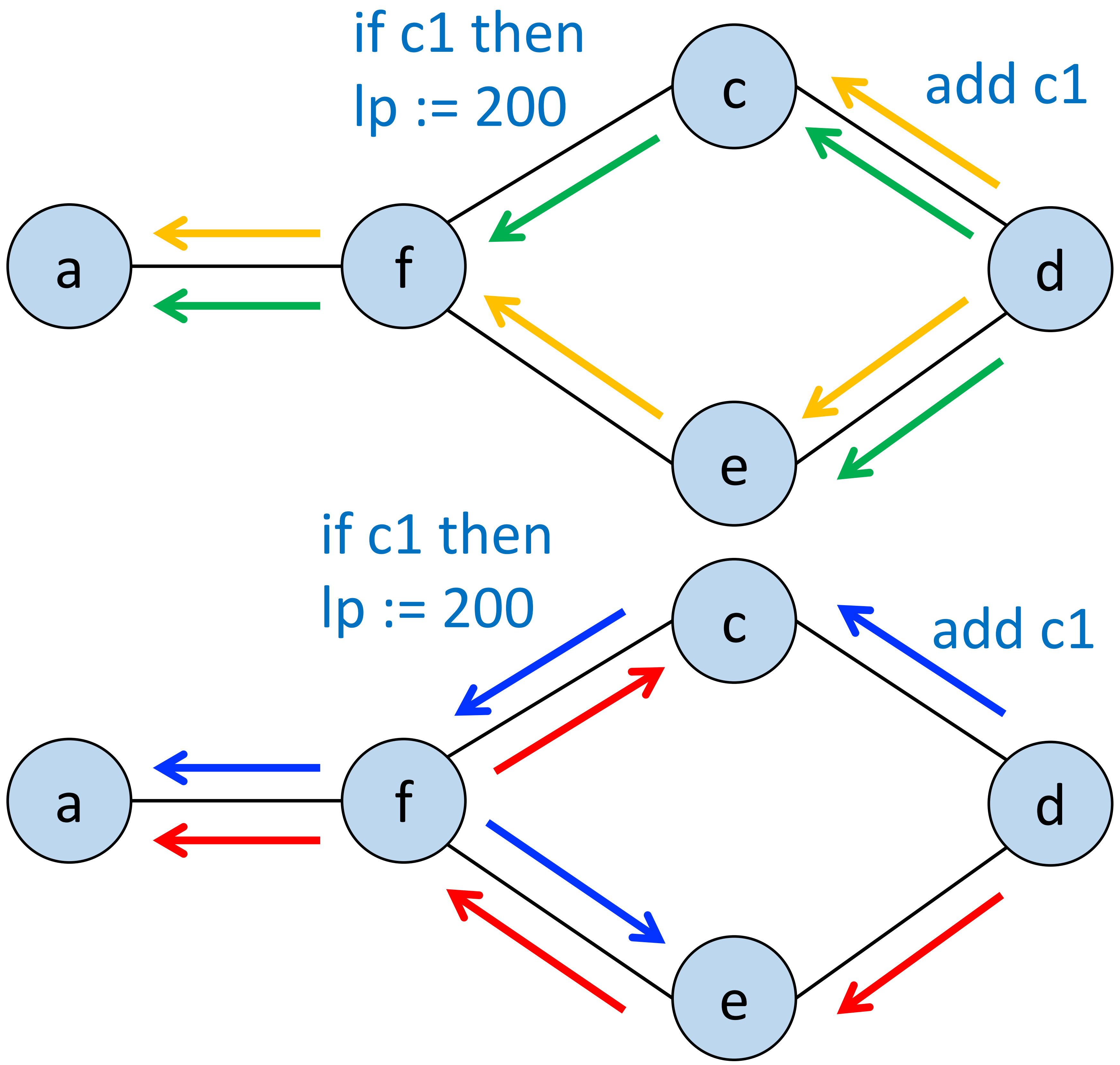
Motivating Example with BGP

Real network N



Route announcement from d reaches a
⇒ **a** can reach **d** in real network

Abstract network N'



4 solutions: **a** can reach **d** in *all* abstract stable routing trees ✓

Soundness of NRC abstractions

- We use the SRP model of the network control plane
- NRC is formulated in terms of an *abstract* SRP
  - where the preference relation is a partial order, rather than a total order
  - Partial order must be consistent with the concrete total order

- Soundness: Abstract SRP S' overapproximates corresponding SRP S
  - Stable solutions of SRP S are guaranteed to be contained in stable solutions of SRP S’

Goal: improve scalability

Two-part Approach

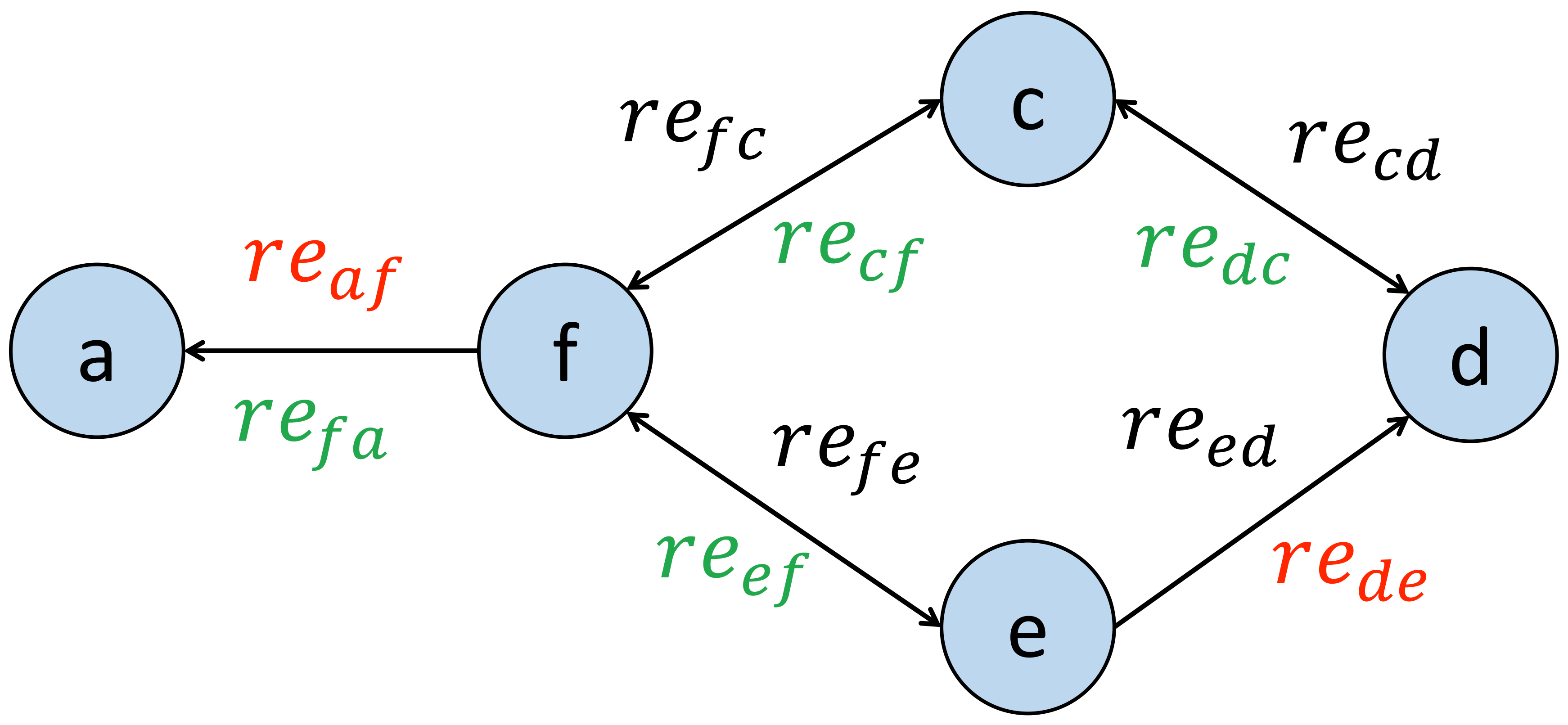
1. Hierarchy of Nondeterministic Routing Choice (NRC) Abstractions
Abstract away route selection while preserving soundness

2. SMT encoding and solvers
Symbolic graph-based encoding [Bayless et al. AAAI 2015]
Use SMT solvers with specialized graph-theories, e.g., MonoSAT

MonoSAT solver

[Bayless et al. AAAI 2015]

- SMT with graph theory solver
- Uses symbolic graphs, graphs with a Boolean variable per edge



Symbolic graph $G_{RE} = (G, RE)$
$G = (V, E)$
$RE = \{re_{uv} \mid (u, v) \in E\}$

Reachability predicate: $G_{RE}.reaches(d, v)$

Stable paths: symbolic graph-based encoding

- Encode abstract SRP S' with SMT formula N'
- Key idea: abstract stable routing trees of S' are captured by symbolic graph solutions

Formula F over RE and other variables, can also use graph-based predicates

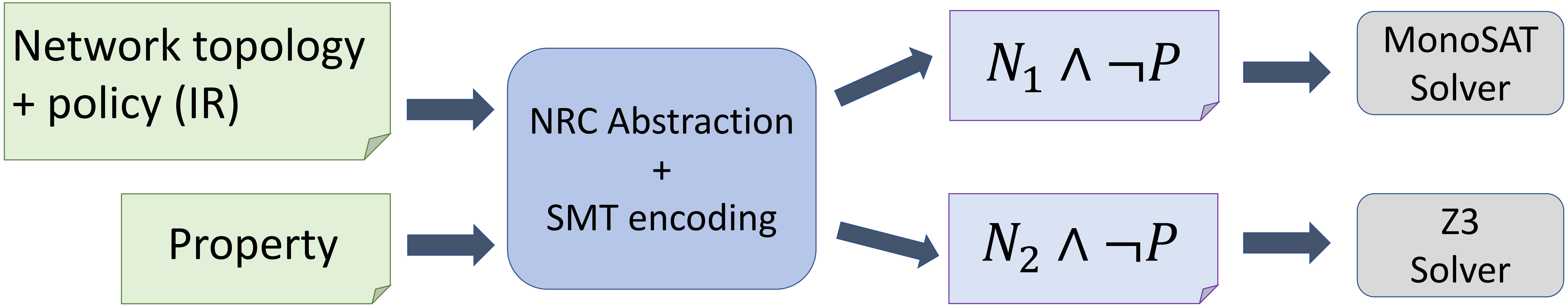
- Satisfying assignment of F → subgraph of G s.t. re_{uv} is true

Benefits of NRC and graph-based encoding

- Fewer variables
  - Route announcement fields used only in route selection are discarded
  - Community attribute sufficient for most policies evaluated

- Expensive transfers can become irrelevant during solver search
  - Concrete formulation (i.e., no NRC abstraction) considers transfers from *all* neighbors to compute best
  - In the abstract formulation once a symbolic edge variable is assigned true, other neighbors become irrelevant

ACORN prototype verifier



- Input format is an intermediate representation (IR)
- Two backend SMT solvers: MonoSAT and Z3

ACORN Evaluation

1. Relative performance of NRC abstraction (with / without)
2. Relative performance of graph theory capable SMT solver (MonoSAT / Z3)

Four experiment settings:

- **abs_mono**: NRC abstraction using MonoSAT
- **abs_z3**: NRC abstraction using Z3
- **mono**: no abstraction using MonoSAT
- **z3**: no abstraction using Z3

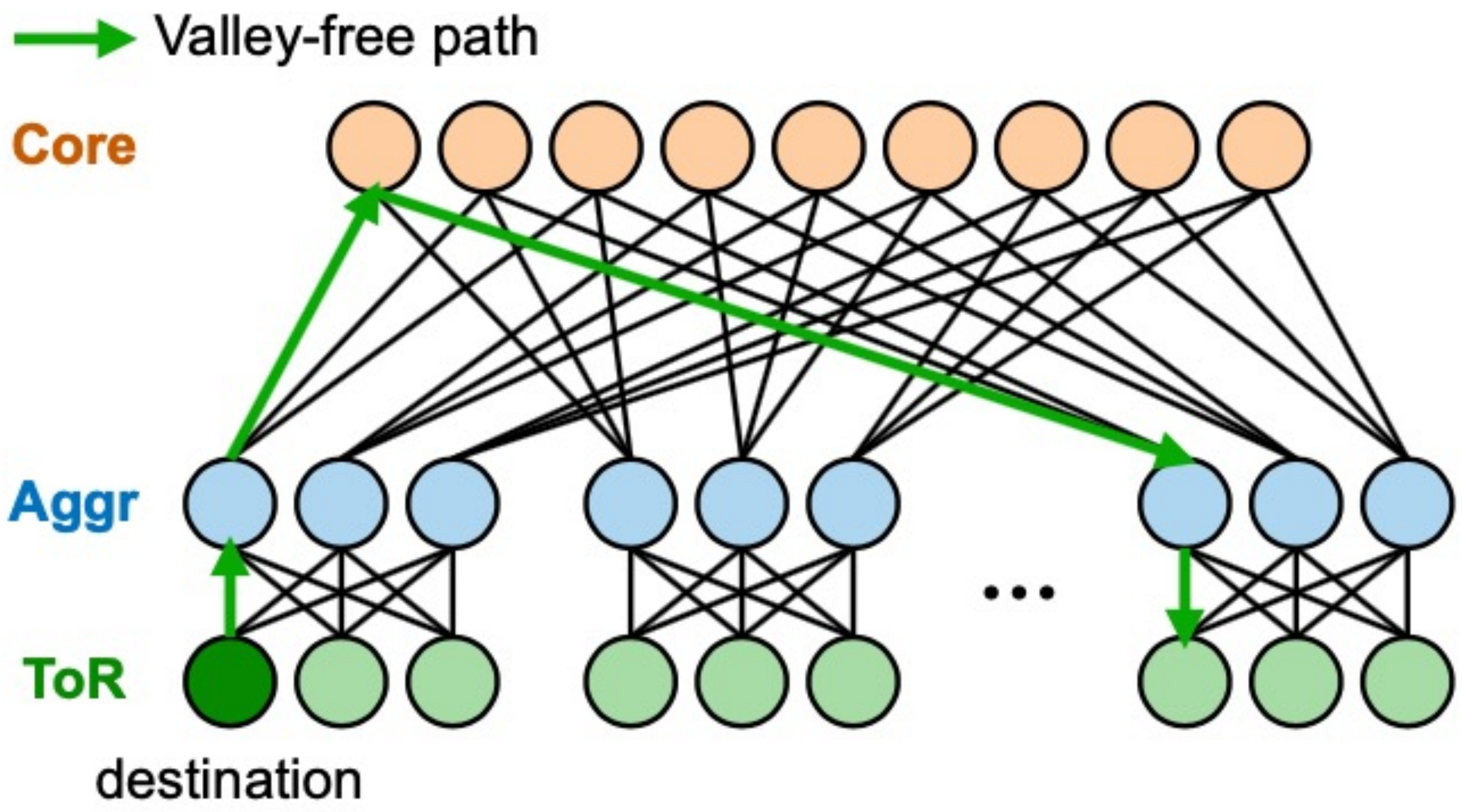
Benchmark examples

1. Data center examples with FatTree topology (to evaluate scalability)
   - Valley-free policy
   - Properties: reachability (single-src), valley-free property

2. Wide Area Network (WAN) examples
   - Topology zoo examples that we annotated with business relationships
   - BGPStream examples, annotated using the CAIDA AS relationships dataset
   - Policy implements the Gao-Rexford conditions [Gao and Rexford. SIGMETRICS 2000]
   - Properties: reachability (all-src), no-transit property

Machine details: 2.3 GHz Intel i7 processor, 16 GB RAM

FatTree network with valley-free policy



c: community attribute
(bit vector of width 2)

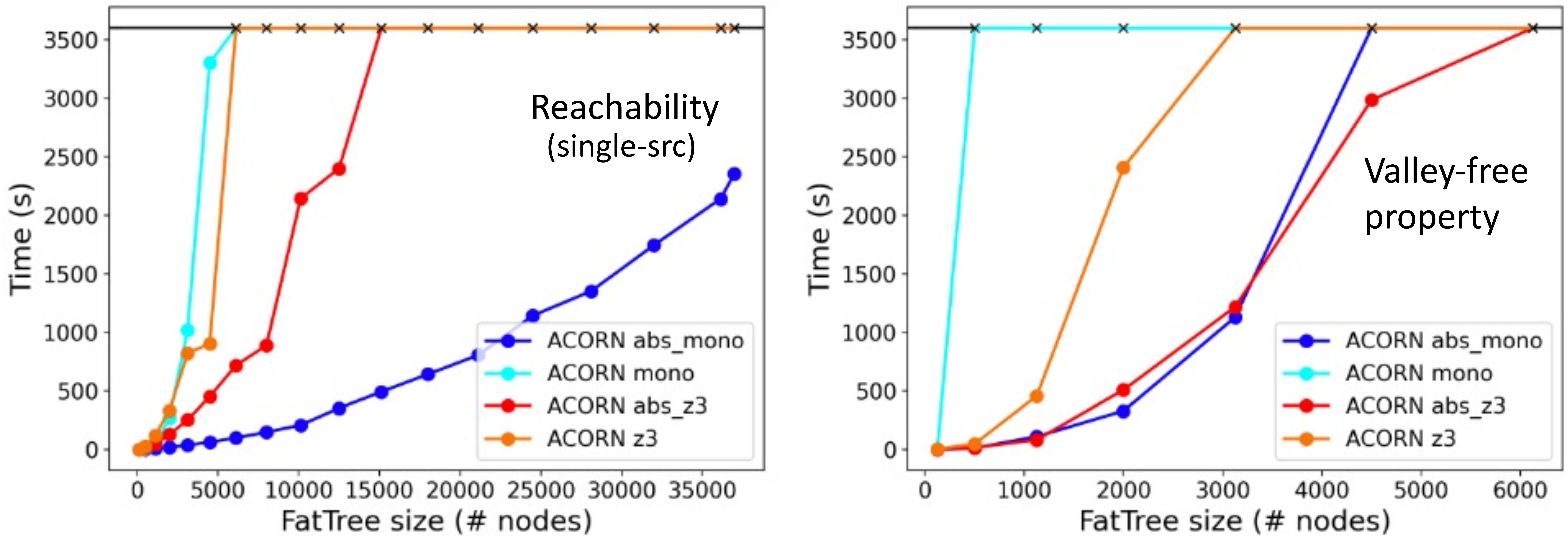
Aggr → Core: if c == 0 then c = 1
else drop route

Aggr → ToR: if c == 0 then c = 1
else if c == 1 then c = 2
else c = 3

ToR → Aggr: if c != 0 then drop route

c = 0: route has 0 Aggr nodes
c = 1: route has 1 Aggr nodes
c = 2: route has 2 Aggr nodes
c = 3: route has $\geq$ 3 Aggr nodes

Results for FatTrees with valley-free policy

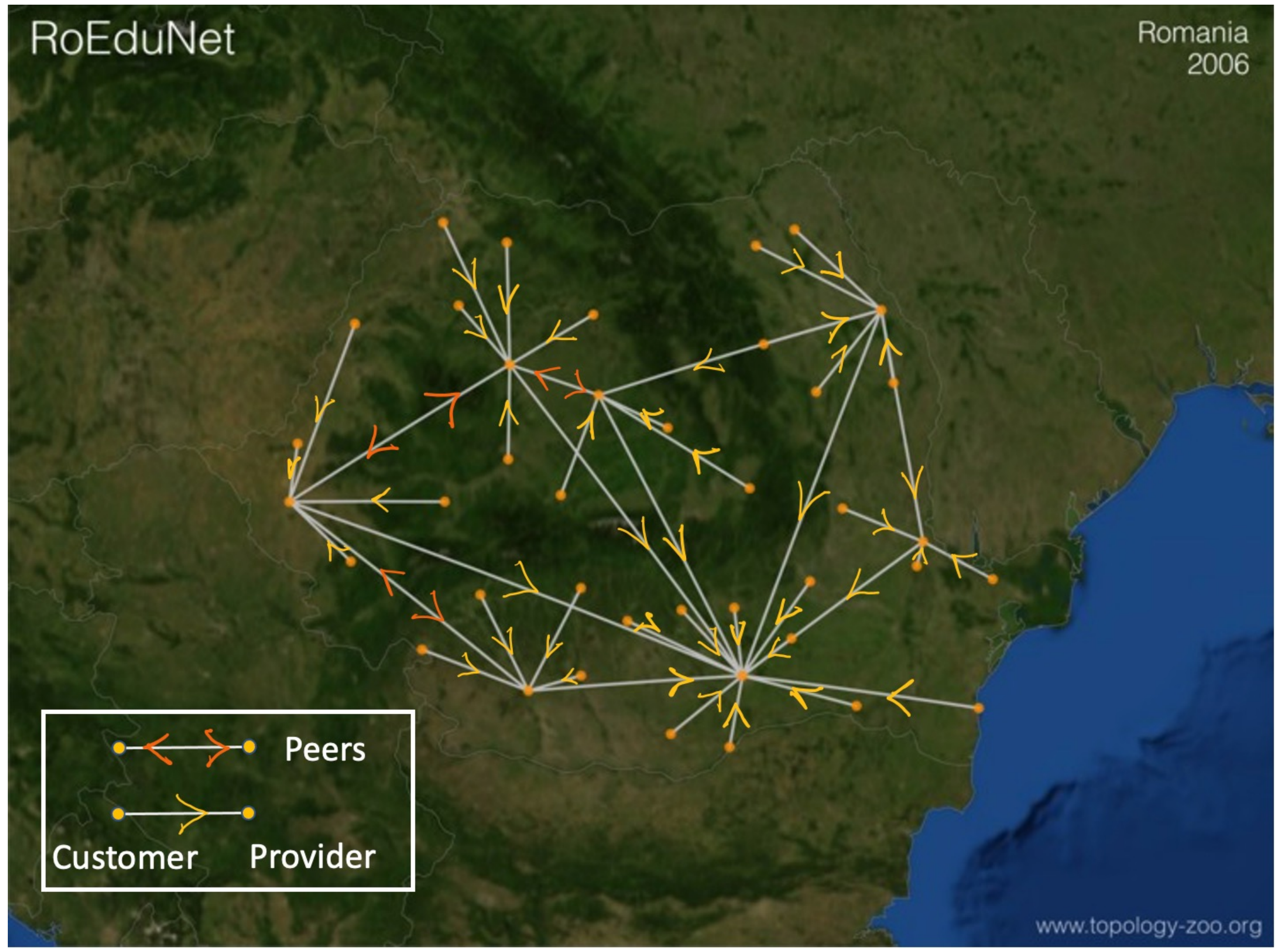


- Abstract settings verify both properties without false alarms
- abs_mono verifies reachability for a FatTree with 36,980 routers in 40 mins

- Abstract settings are *uniformly* better than concrete settings (upto 52x speedup for MonoSAT)
- MonoSAT better for reachability but Z3 better for valley-free property

Topology Zoo network: Example

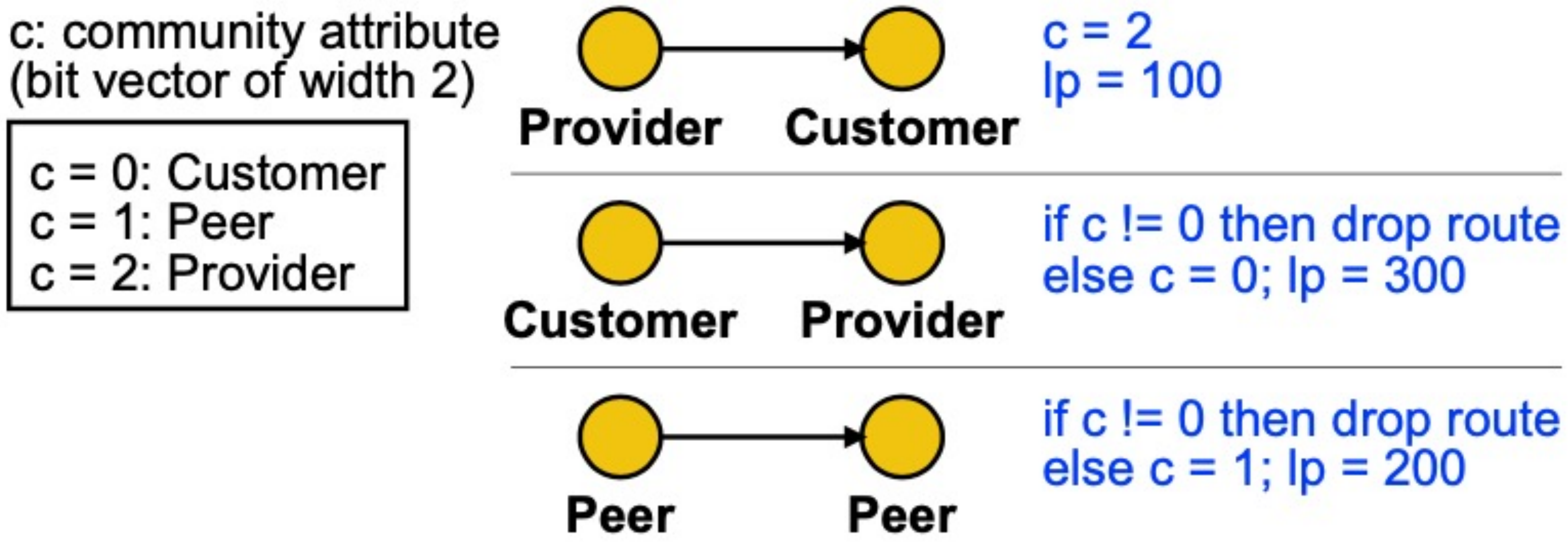
[Gao and Rexford. SIGMETRICS 2000]



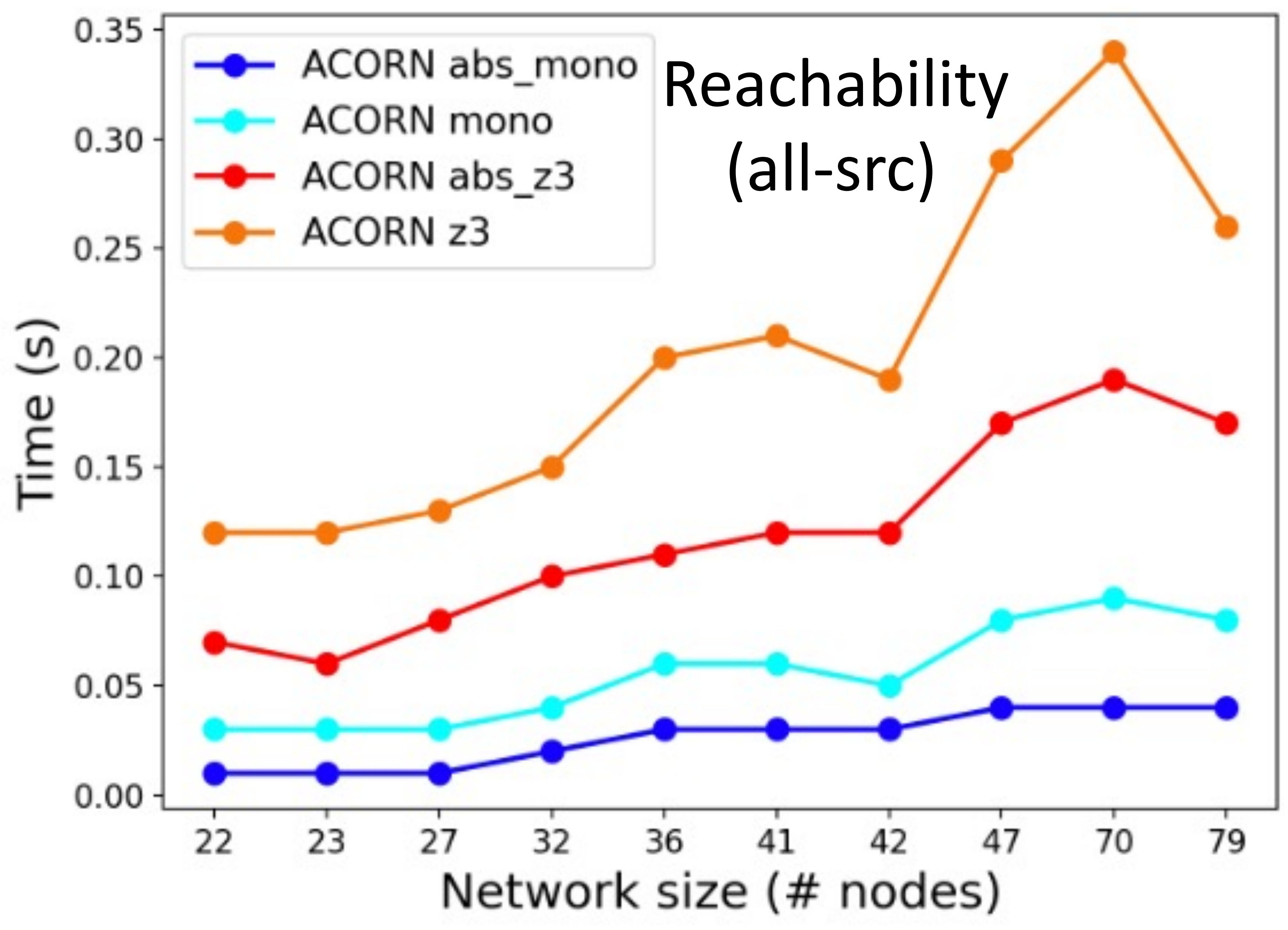
42 nodes, 50 edges. Each node is an AS.
Edges annotated with business relationships (customer/peer/provider)

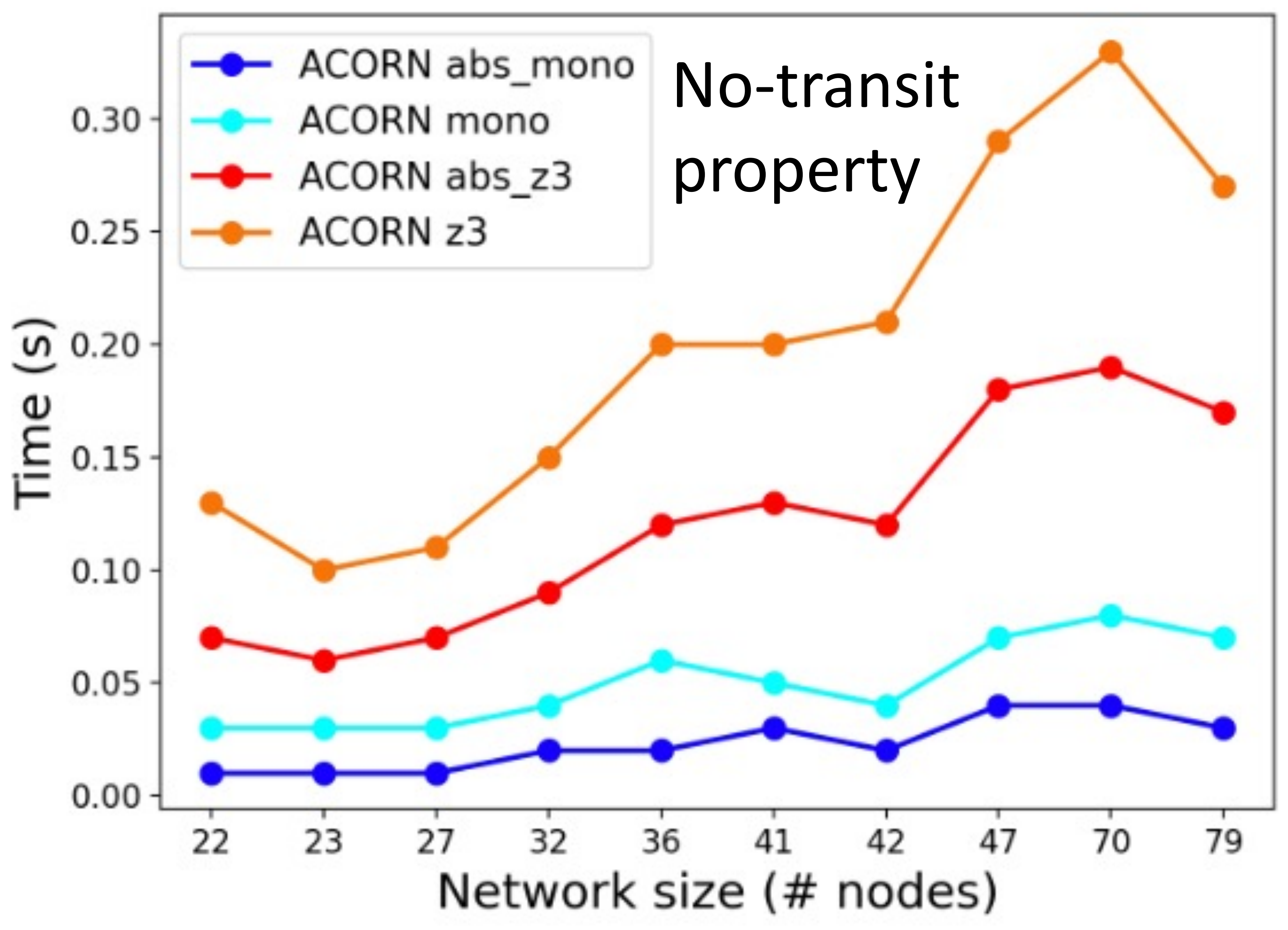
Policy implements Gao-Rexford conditions:

- Prefer Cust < Peer < Prov (Cust most preferred)
- Don't export routes from Peer/Prov to another Peer/Prov (no-transit property)



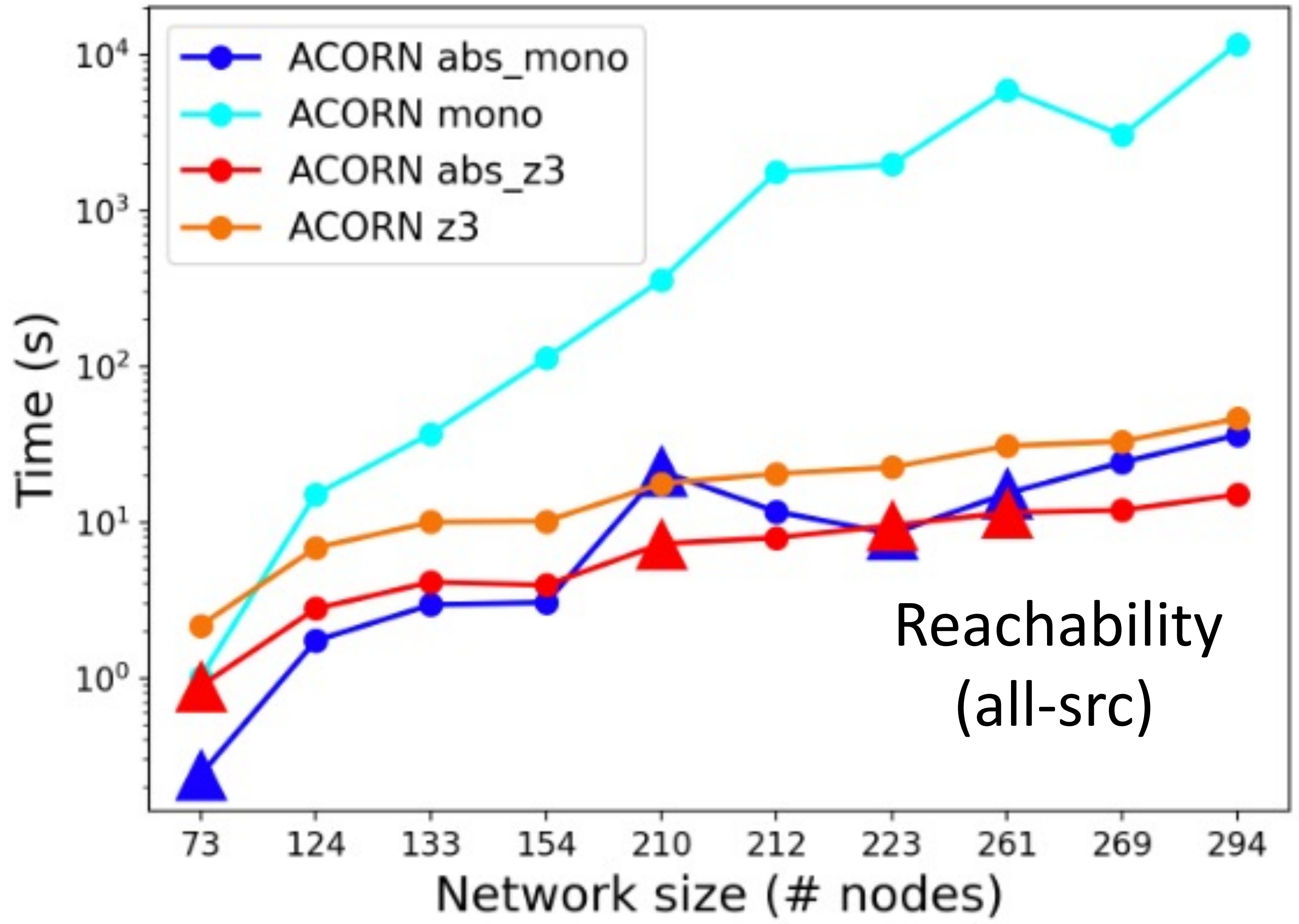
Results for Topology Zoo networks

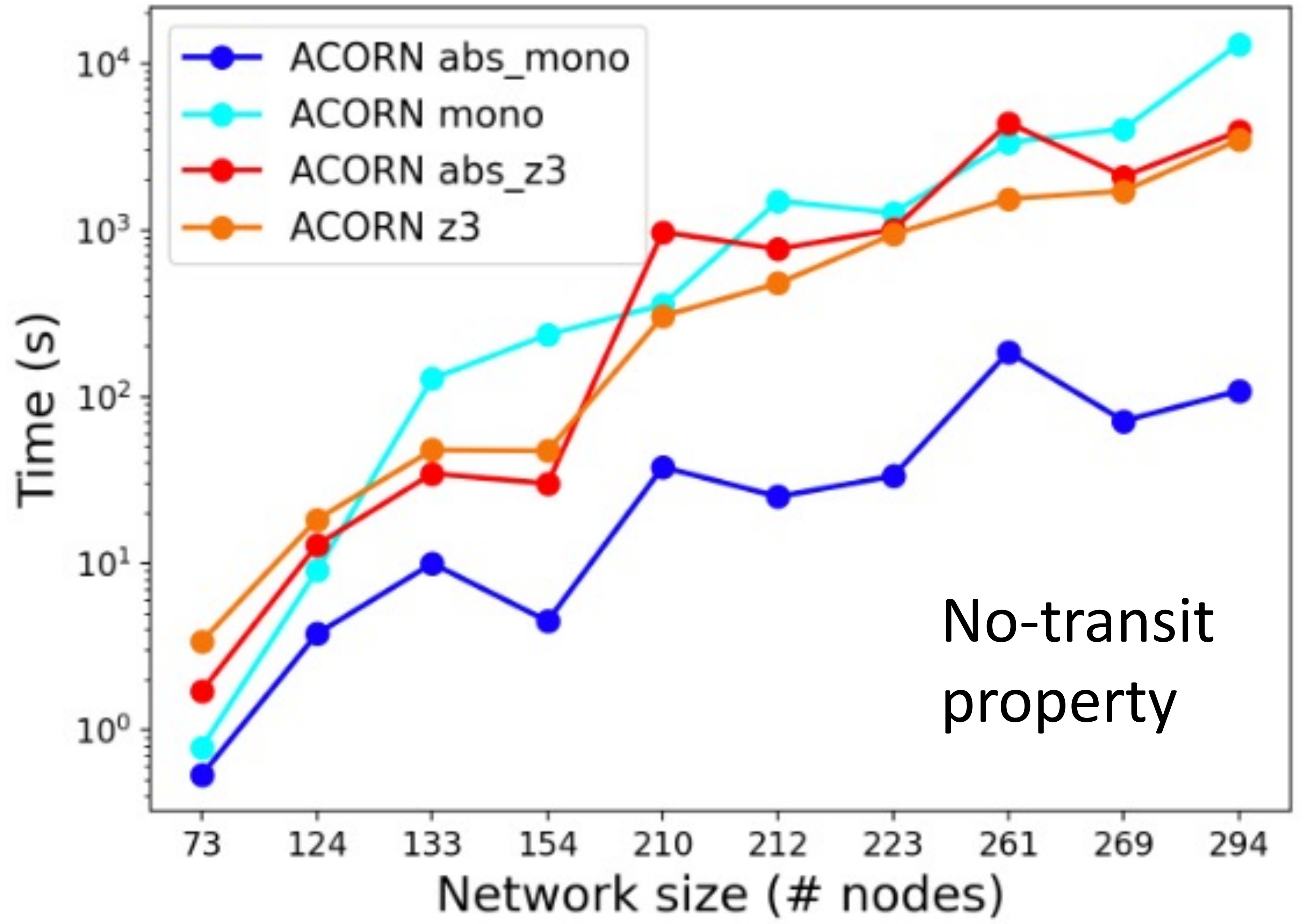




- Abstract settings verify both properties without false alarms
- Abstract settings are *uniformly* better than concrete settings (relative speedup of 3x)
- MonoSAT better for both properties

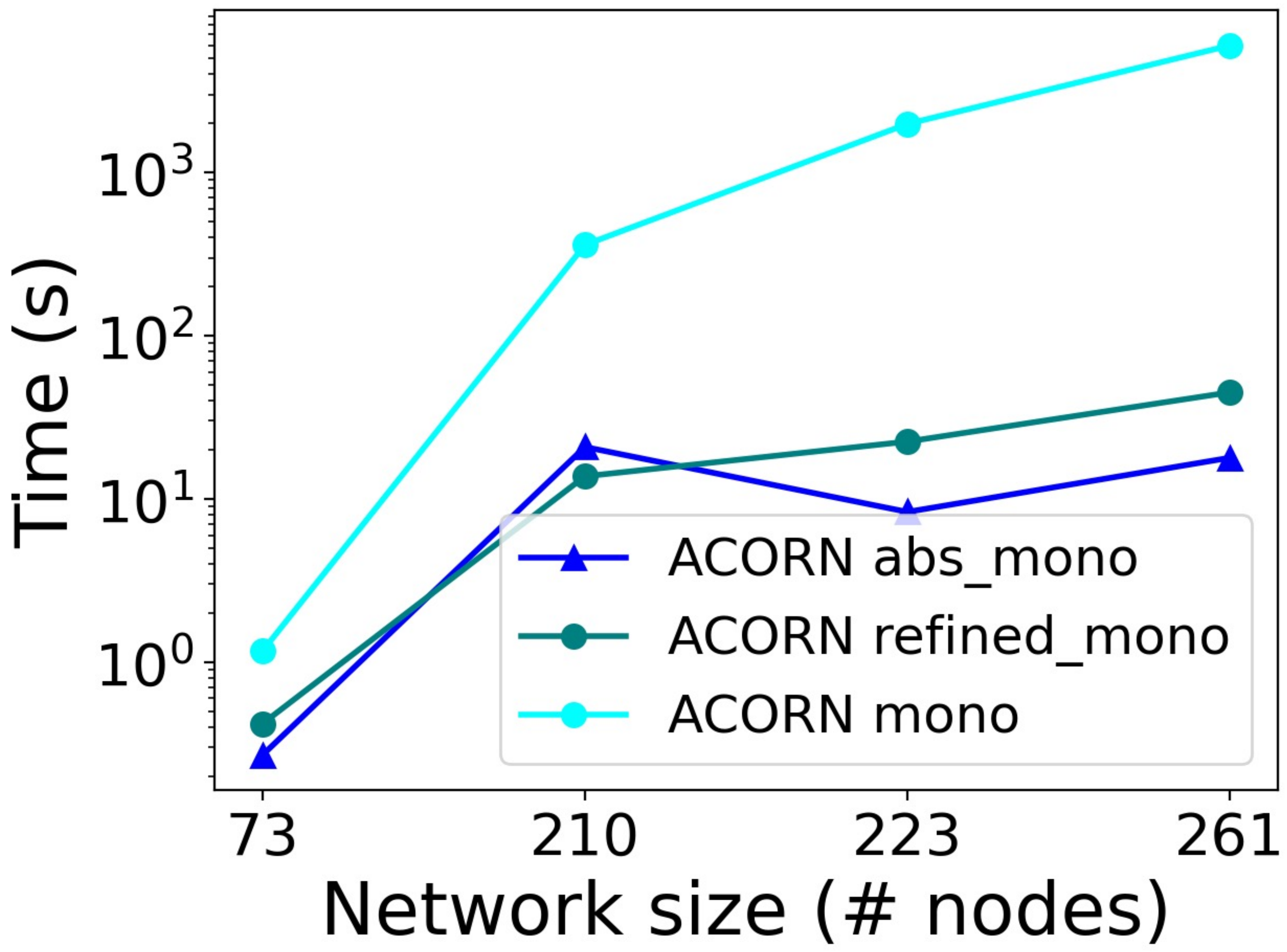
Results for BGPStream networks

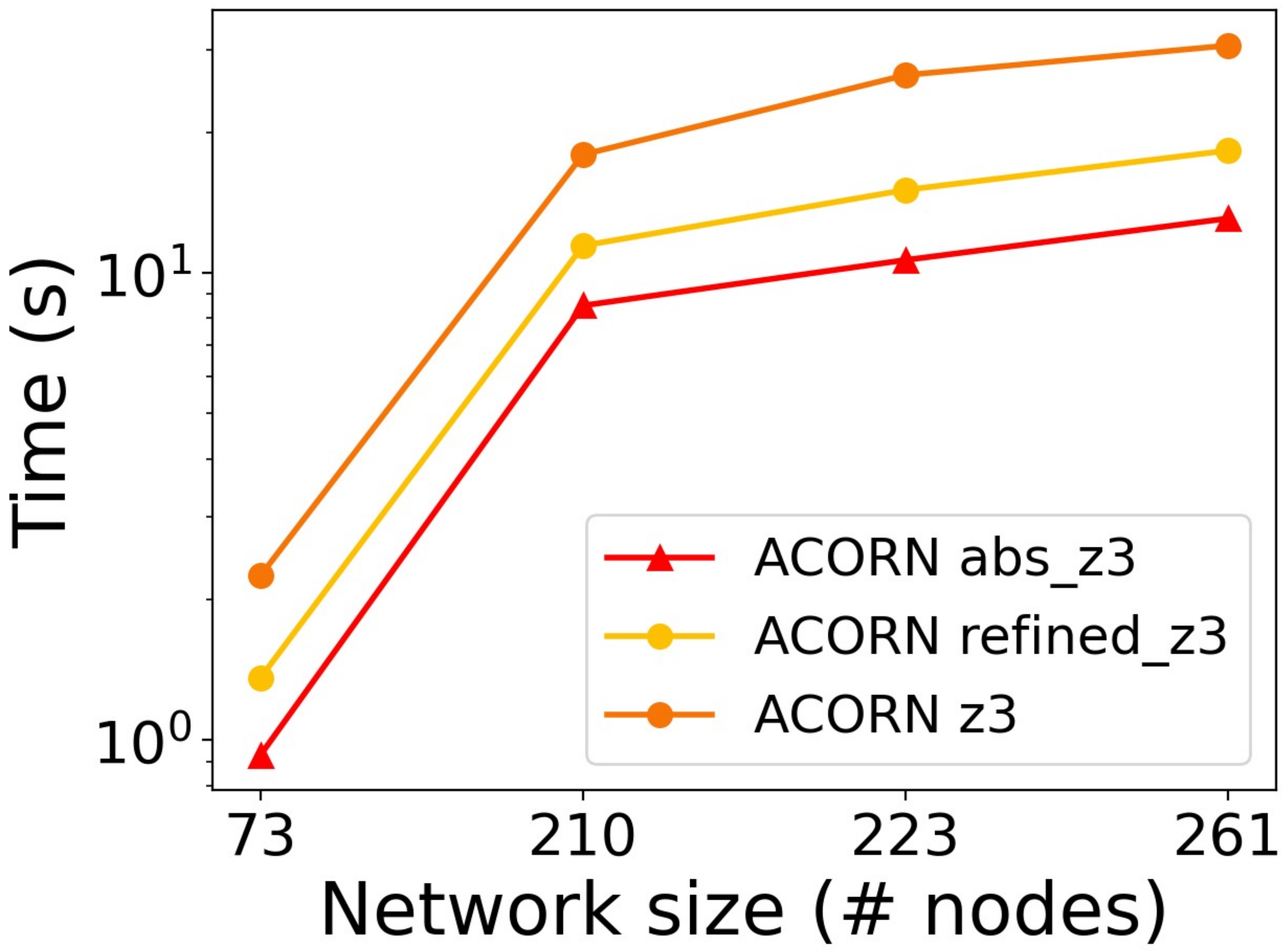




- Abstract settings verify no-transit property in all networks, and reachability in 6/10 networks
- Abstract settings are better than concrete settings
  - MonoSAT: speedup of 323x for reachability, and 120x for no-transit
  - Z3: benefit of NRC not as pronounced; no-abstraction setting sometimes better for no-transit

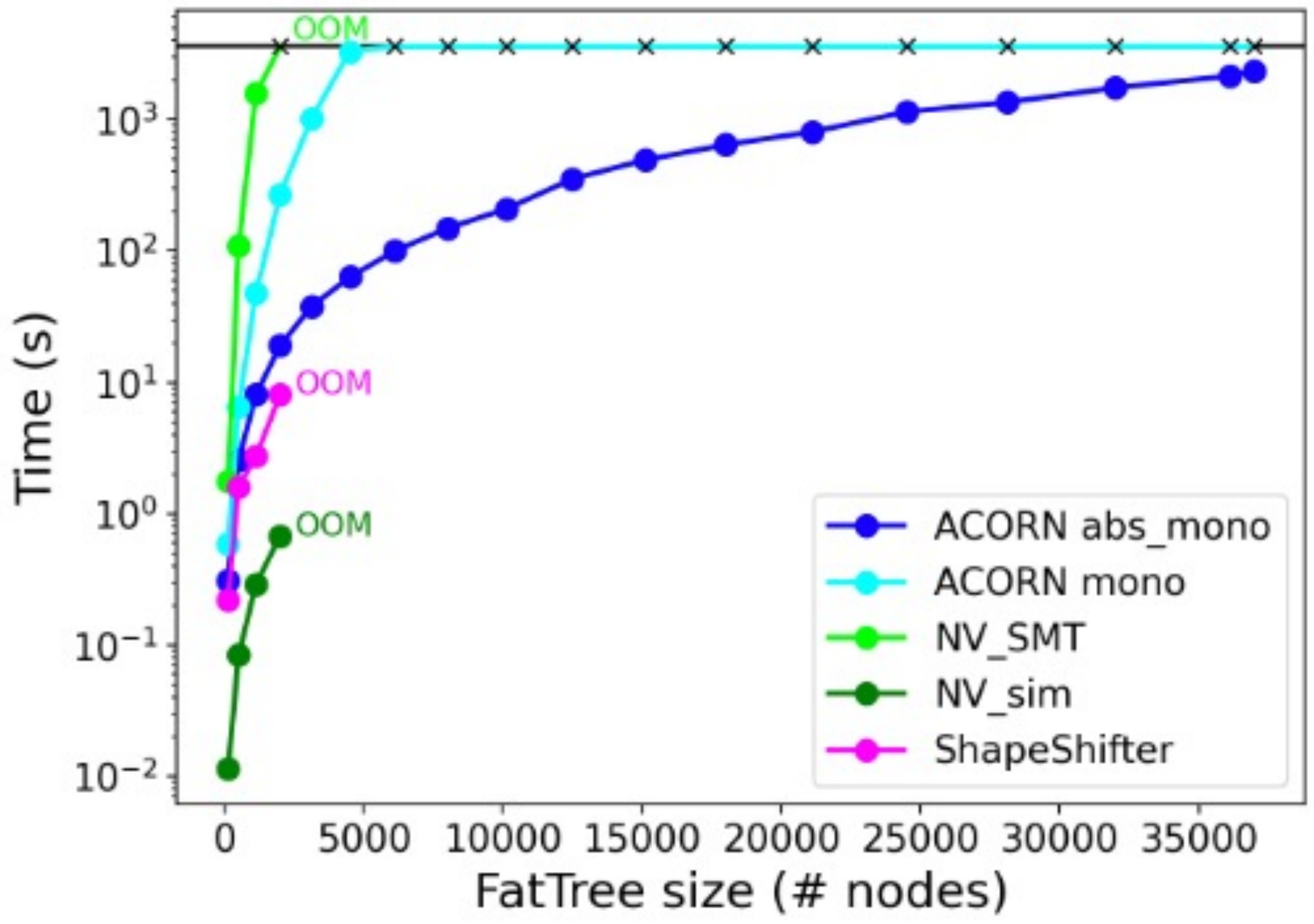
NRC Refinement





- 4/10 false alarms are handled using a more precise abstraction that models local preference
- NRC abstraction hierarchy provides a tradeoff between performance and precision

Comparison of ACORN with other tools



Reachability (single-src) on FatTree benchmarks with valley-free policy

- NV uses MTBDD-based simulation and SMT
- ShapeShifter uses BDD-based simulation with abstract interpretation

- Both NV and ShapeShifter *run out of memory* for networks with > 3000 nodes

- ACORN scales to ≈37,000 nodes

- SAT/SMT techniques seem more scalable than BDD-based methods *(again!)*

ACORN: Summary of experimental results

- NRC can verify reachability for ≈37,000 routers within an hour
  - ***Far exceeds performance of existing control plane verifiers***
- NRC improves scalability for both solvers, all benchmarks
  - Abstract settings uniformly better than concrete settings
- NRC could verify realistic policies
  - Common policies on data center networks, no false alarms with least precision
  - Some false alarms in WANs, refinements verified successfully
  - Future work: CEGAR-based refinement
- MonoSAT's graph theory solver useful for reachability
  - Z3 sometimes better for policy-based properties
  - This needs further investigation

Lessons (re-)Learned

- Build the logic-based model first, leverage domain insights
  - modeling the network control plane *stable behavior* enabled direct use of SMT technology for verification
- Don't abstract too early
  - our SMT-based model is rich in detail, captures many features (e.g., local preferences, route redistribution) considered important by network practitioners
- Build logic-based abstractions, compositional methods, CEGAR, <*your-favorite-method*> on top of the logic-based model
  - Bonsai: Symmetry-based abstractions [Beckett *et al.* SIGCOMM 2018]
  - Shapeshifter: Abstract Interpretation [Beckett *et al.* POPL 2020]
  - Origami: Failure analysis [Giannarakis *et al.* CAV 2019]
  - NV: Programmable Platform [Giannarakis *et al.* PLDI 2020]
  - ACORN: Nondeterministic route selection [under submission]
  - Timepiece: Modular verification [under submission]

Future Opportunities

- Many challenges still remain: scalability, failure analysis, …

- Quantitative/probabilistic properties
  - Some existing efforts (e.g., Probabilistic NetKat, ApproxFlow)
  - ➢ Model counting techniques
  - ➢ Probabilistic verification

- Automated synthesis with verification
  - ➢ Leverage machine learning + deductive techniques

Collaborators

Ryan Beckett
Microsoft Research

Ratul Mahajan
Univ of Washington

Divya Raghunathan
Princeton

Tim Thijm
Princeton

Dave Walker
Princeton

Thank you!