Minesweeper and Propane: Two Tools for Improving Network Reliability

Ryan Beckett
with
Aarti Gupta, Ratul Mahajan, Jitu Padhye, Todd Millstein, David Walker

PRINCETON UNIVERSITY
Microsoft
INTENTIONET

Why configuration is hard

Complexity

-BGP, OSPF, RIP
-Route redistribution
-Protocol preference
-Metric conversions

Low-level

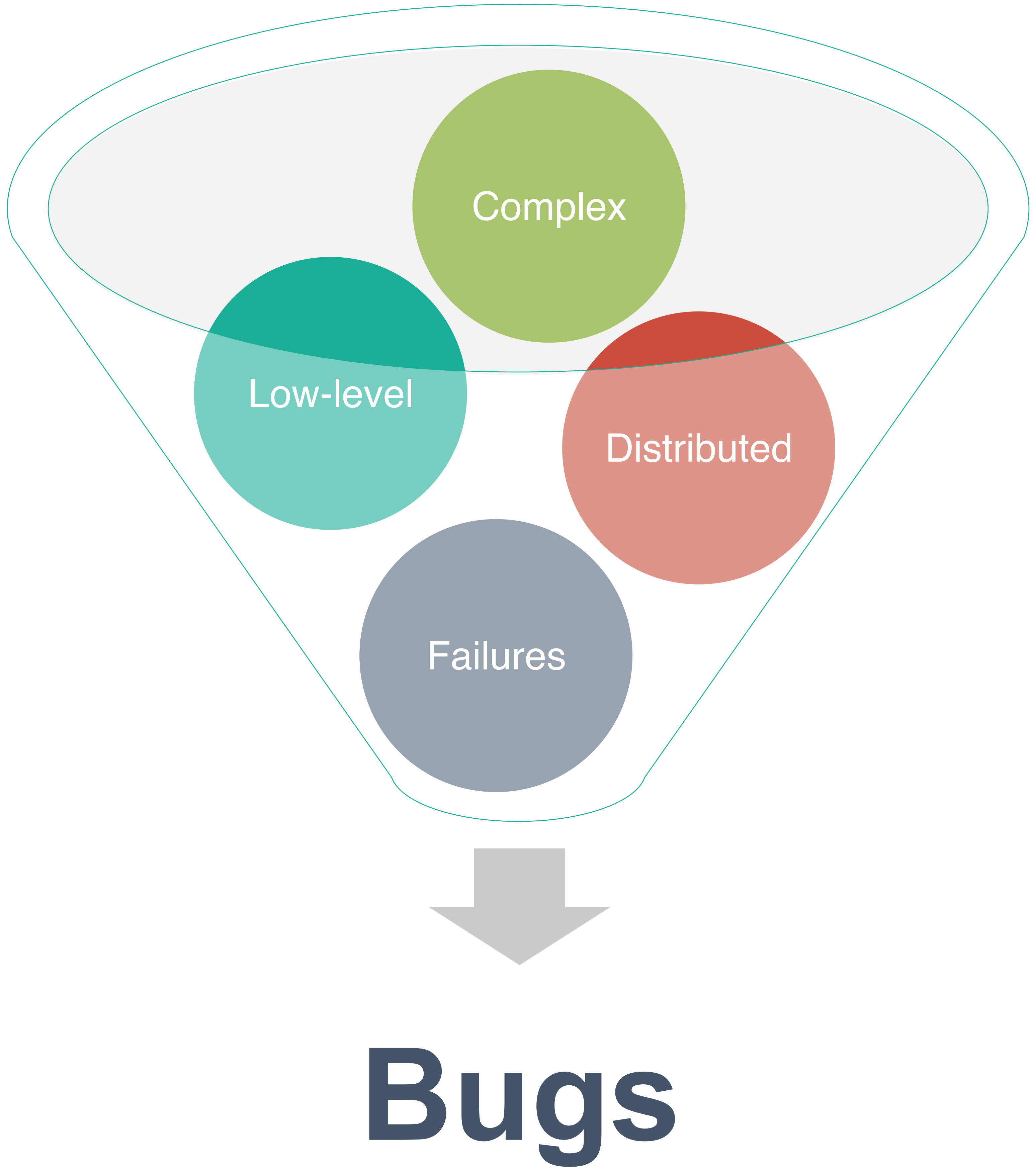
-Protocol parameters
-Interface metrics
-Route maps
-Access control lists

Distributed

-Device configurations
-100,000s of lines

Failures

-Link failures common
-Router failures possible



Misconfigurations are common

South Africa: FNB solves crippling connectivity issues

July 25, 2016 • Finance, Southern Africa, Top Stories

BGP errors are to blame for Monday's Twitter outage, not DDoS attacks

No, your toaster didn't kill Twitter, an engineer did

Unions want Southwest CEO removed after IT outage

Massive route leak causes Internet slowdown

Posted by Andree Toonk – June 12, 2015 – BGP instability – No Comments

Home / Cisco Security / Security Advisories and Alerts

Security Activity Bulletin

Misconfigured Router Causes Increased BGP Traffic and Isolated Outages for Internet Services

Microsoft: misconfigured network device led to Azure outage

30 July 2012 | By Yevgeniy Sverdlik

Router Crashes Trigger Major Southwest IT System Failure

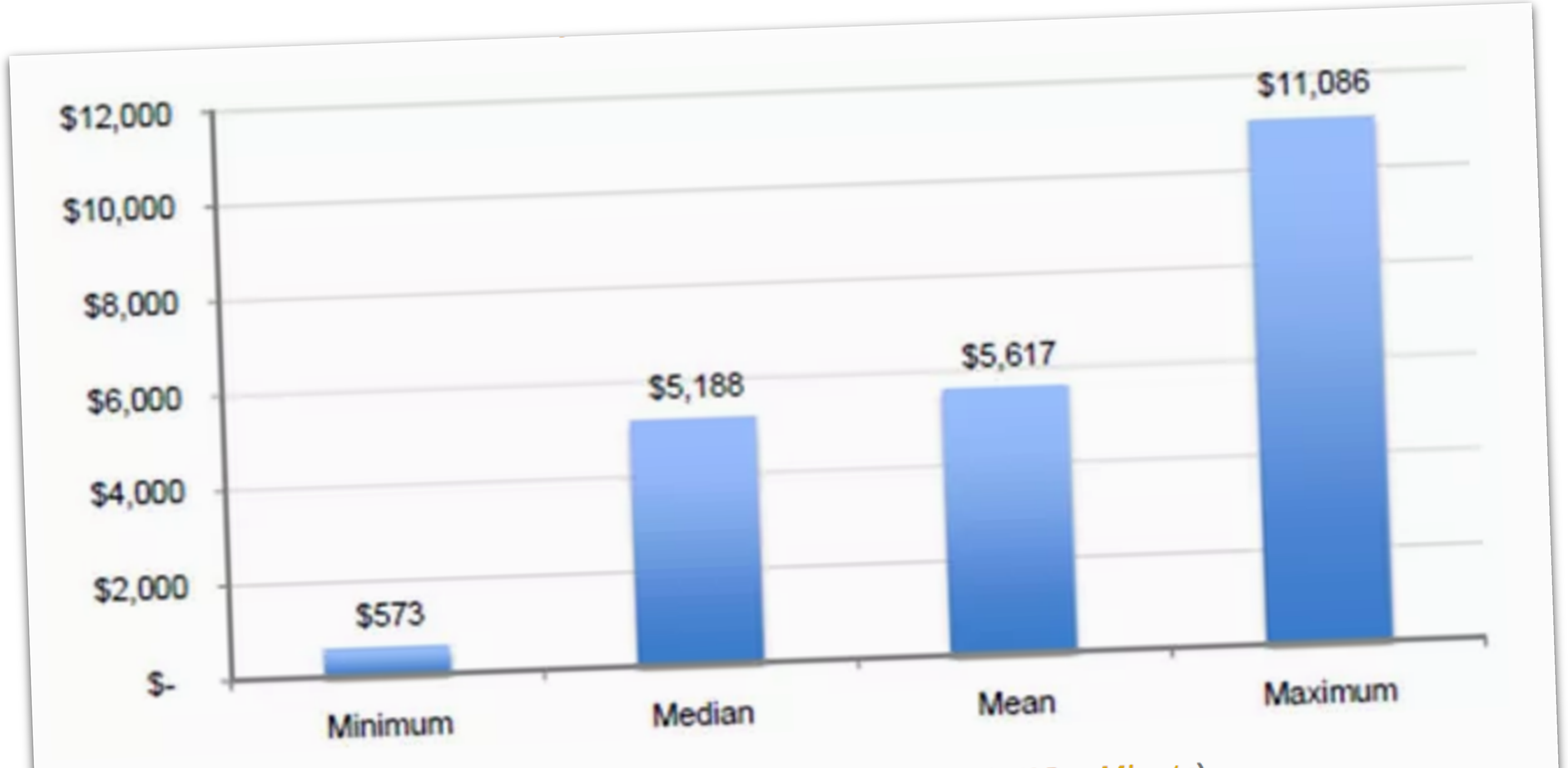
By: Chris Preimesberger | July 21, 2016

BlackBerry outage could cost RIM $100 million

Xbox Live outage caused by network configuration problem

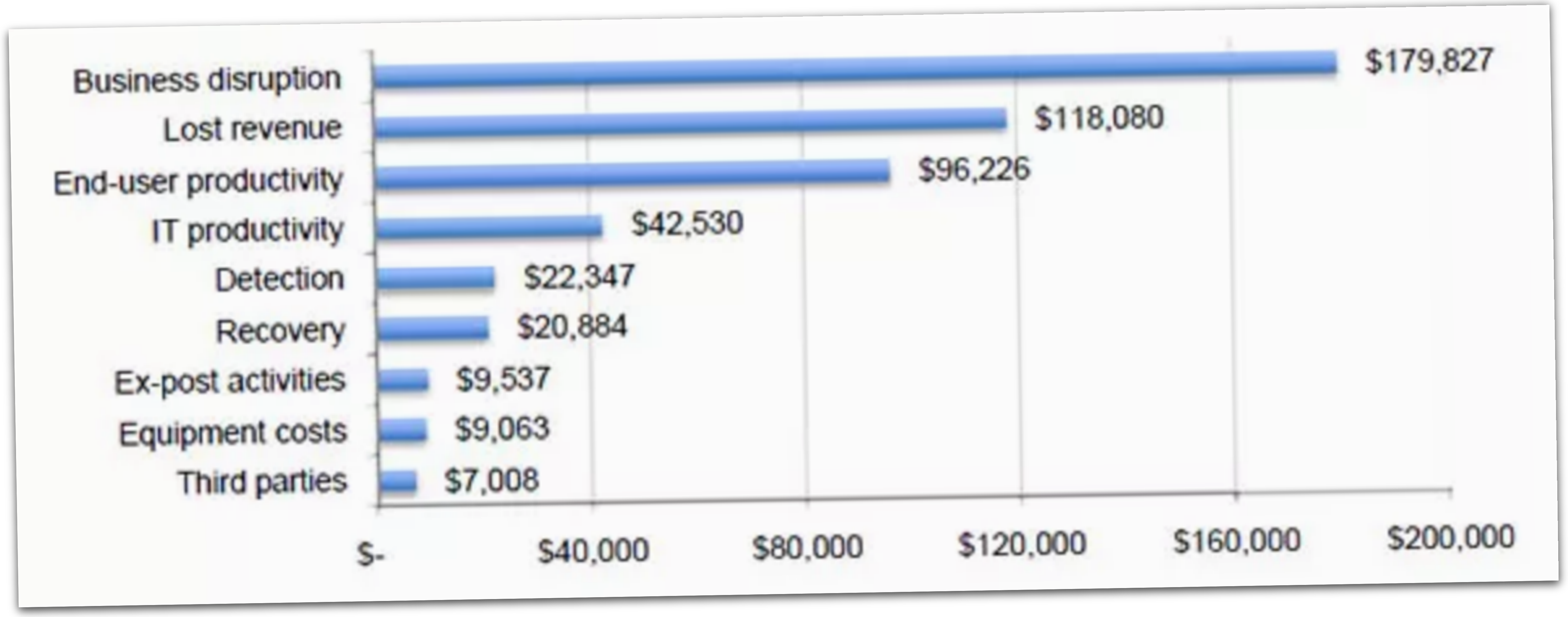
BY TODD BISHOP on April 15, 2013 at 9:27 am

Misconfigurations are expensive



(Source: *Unplanned IT Downtime Can Cost $5K Per Minute*)

- **Lack of automation causes outages and breaches.** 20% of organizations experienced a security breach, 48% had an application outage and 42% had a network outage as a result of a misconfiguration caused by a manual security-related process.



average of 105 server racks. The study concluded that the average duration of a single data center outage was 95 minutes equating to a cost of $740,357

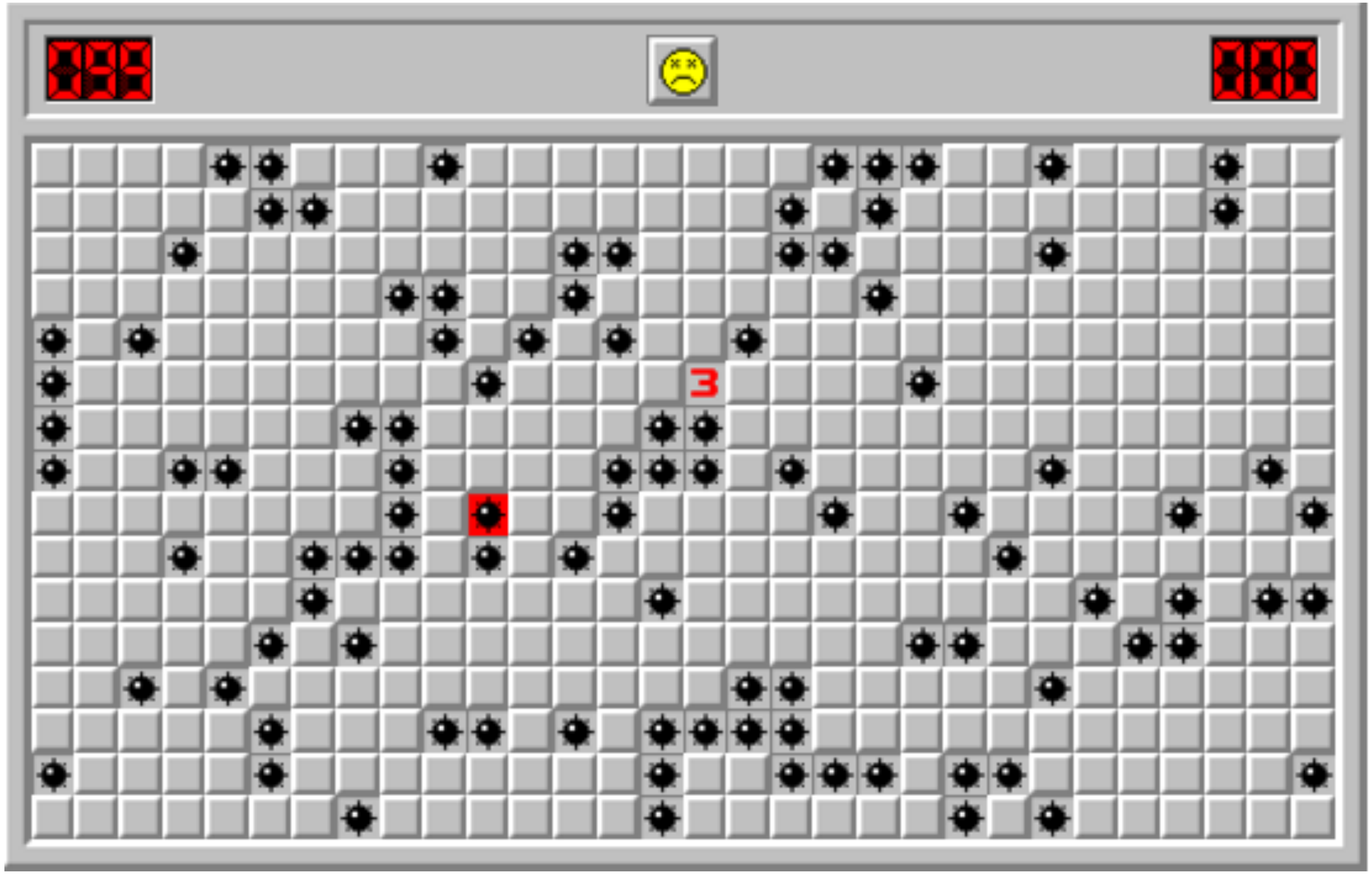
Minesweeper

Find bugs in **legacy** networks

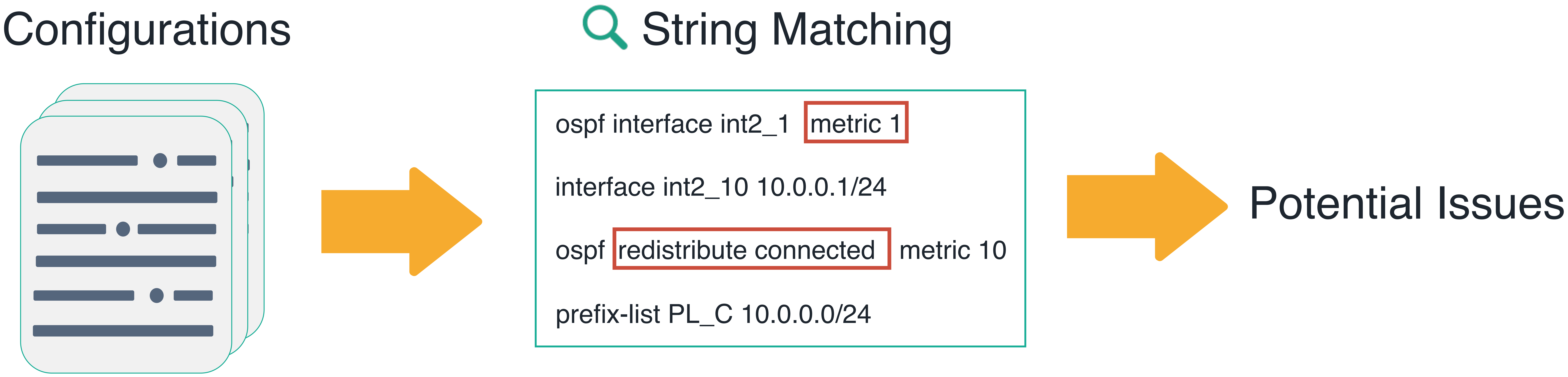
Propane

High-level design of **new** networks

Minesweeper



Current Approach: Heuristics



Examples: RCC, SolarWinds, HPNA / TruControl, NetDoctor

Heuristics: Limitations

- Can miss many bugs
- Can report false positives
- Hard to test **forwarding behavior**

String Matching

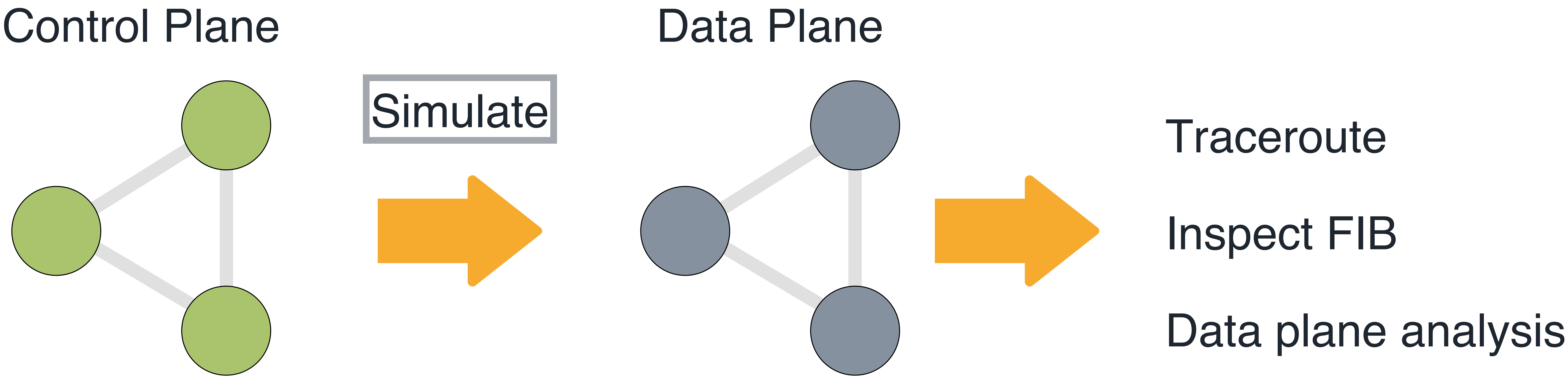
ospf interface int2_1 metric 1

interface int2_10 10.0.0.1/24

ospf redistribute connected metric 10

prefix-list PL_C 10.0.0.0/24

Current Approach: Simulation



Examples: Batfish, C-BGP

Simulation: Limitations

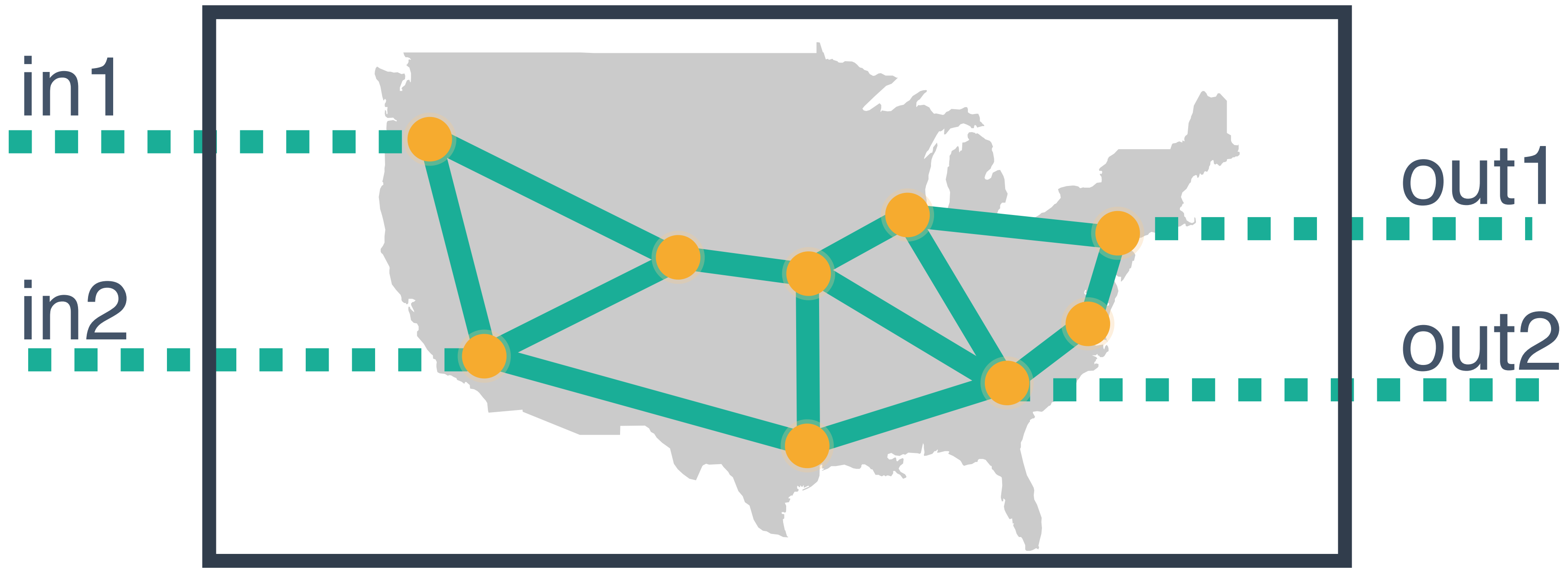
Cannot test for all routing messages

Cannot test for all possible link failures

Overview

We present a new network analysis tool called **Minesweeper**:

- Can check many properties for all external routing messages and for all link failures
- Encodes the network as a collection of Logical constraints and leverages off-the-shelf constraint solvers
- https://batfish.github.io/minesweeper/



Workflow

1. Vendor-Specific Configs

Parse

2. Vendor-Independent Format

```
Interfaces: {
  Ethernet0/0: {
     InterfaceCost: 1,
     importPolicy: "PEER_IN",
     …
  }
}
```

Encode

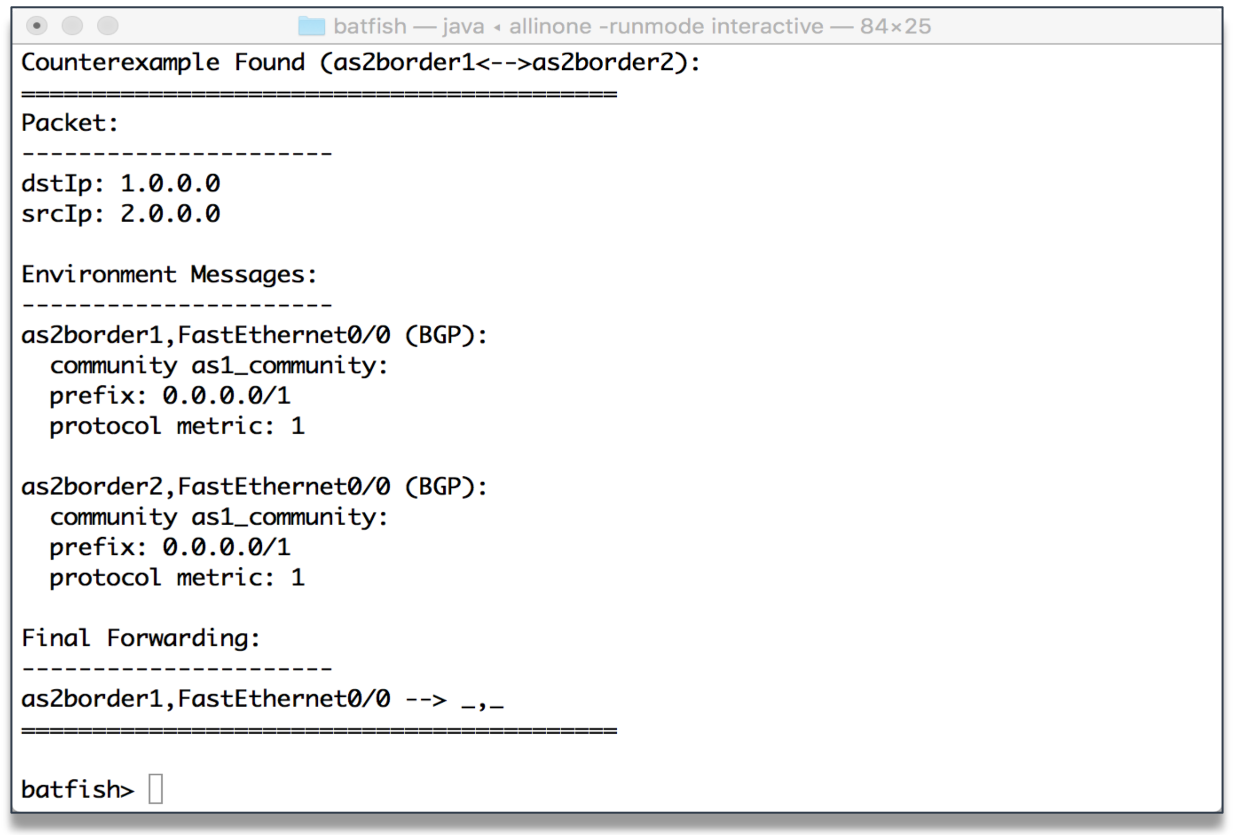
3. Constraint Encoding

$192.0.0.0 \leq out.prefix$
$out.prefix \leq 192.1.0.0$
$best.valid \Rightarrow out.lp = 120$
$best.valid \Rightarrow out.ad = 20$

+ Query

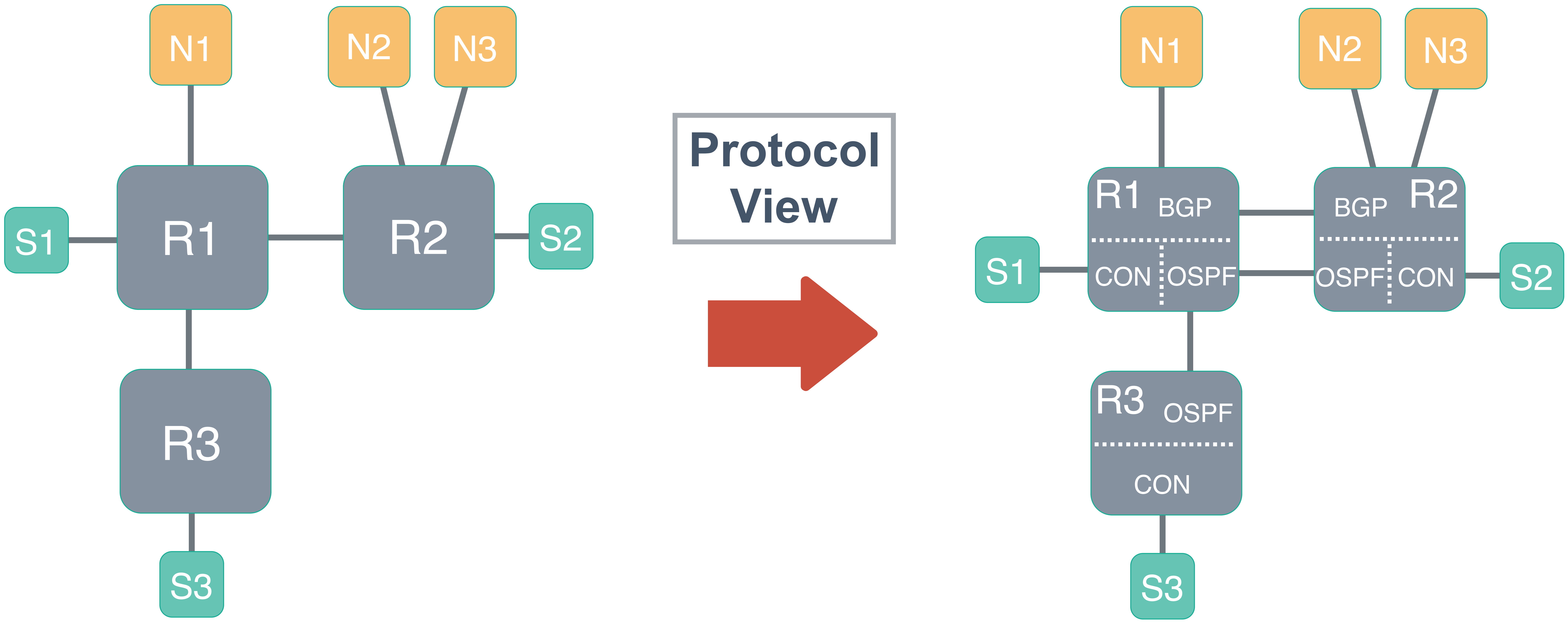
Solve

4. Output

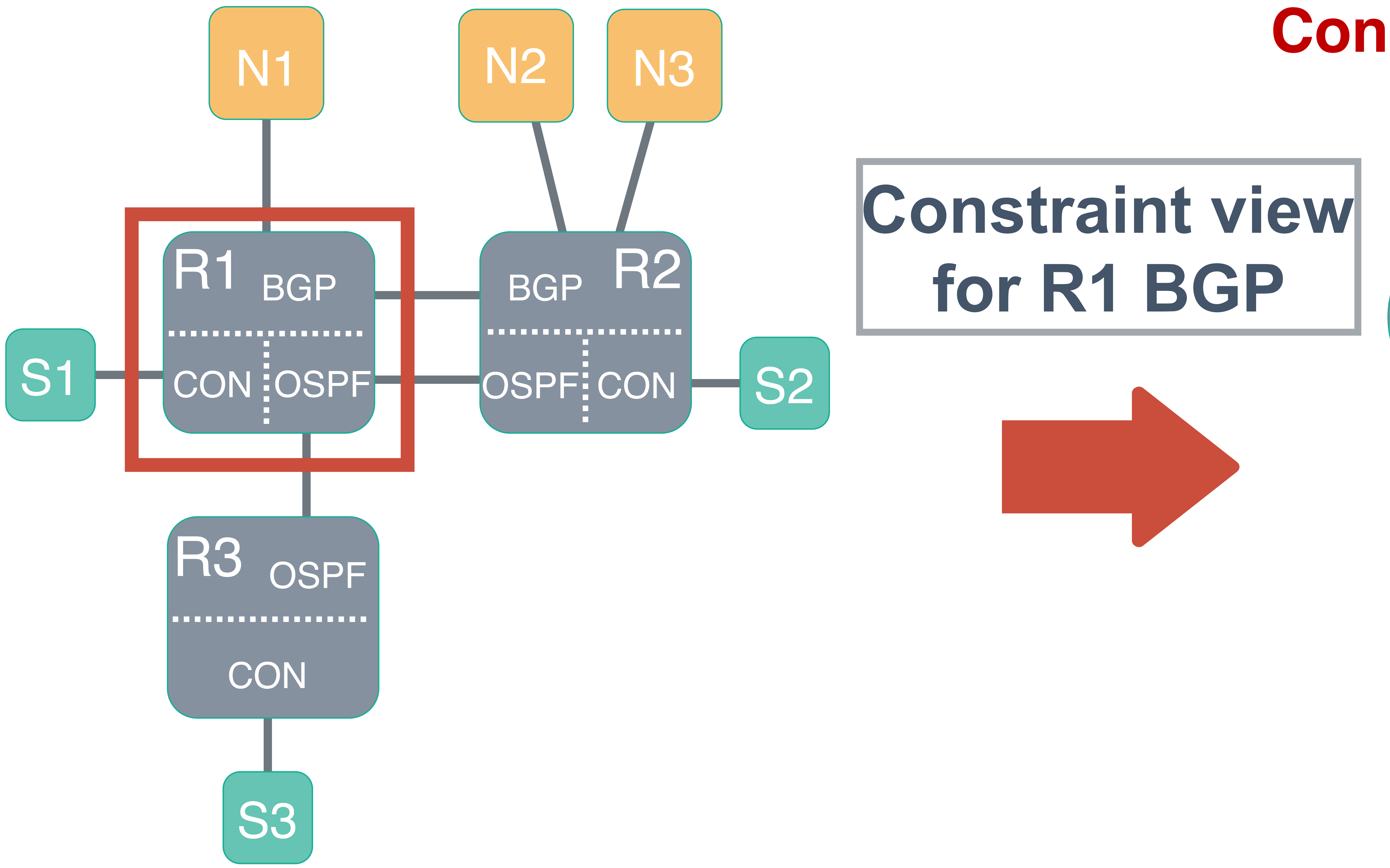


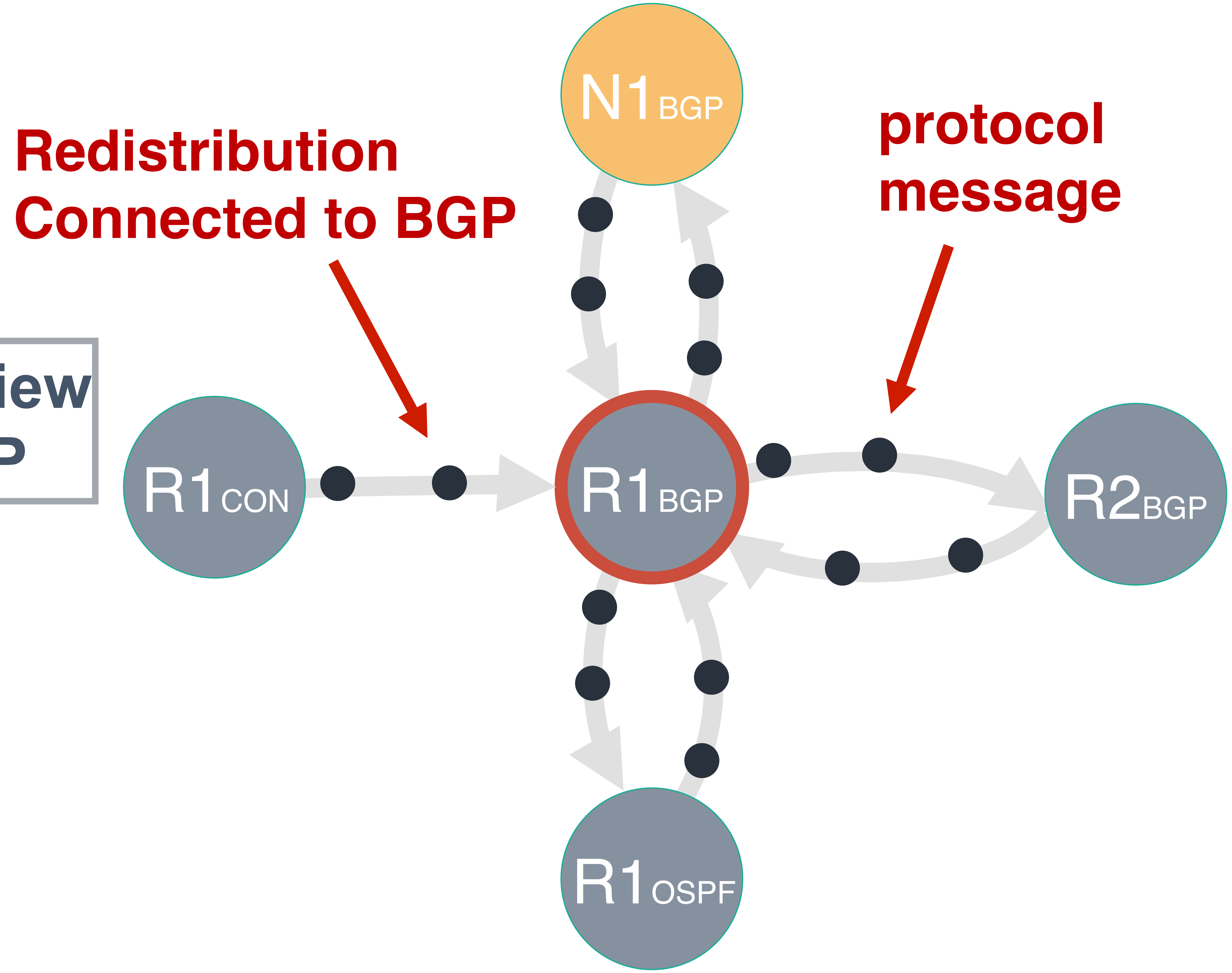
```
batfish — java ◂ allinone -runmode interactive — 84×25
Counterexample Found (as2border1<-->as2border2):
==========================================
Packet:
----------------------
dstIp: 1.0.0.0
srcIp: 2.0.0.0

Environment Messages:
----------------------
as2border1,FastEthernet0/0 (BGP):
  community as1_community:
  prefix: 0.0.0.0/1
  protocol metric: 1

as2border2,FastEthernet0/0 (BGP):
  community as1_community:
  prefix: 0.0.0.0/1
  protocol metric: 1

Final Forwarding:
----------------------
as2border1,FastEthernet0/0 --> _,_
==========================================

batfish>
```

Constraint Encoding

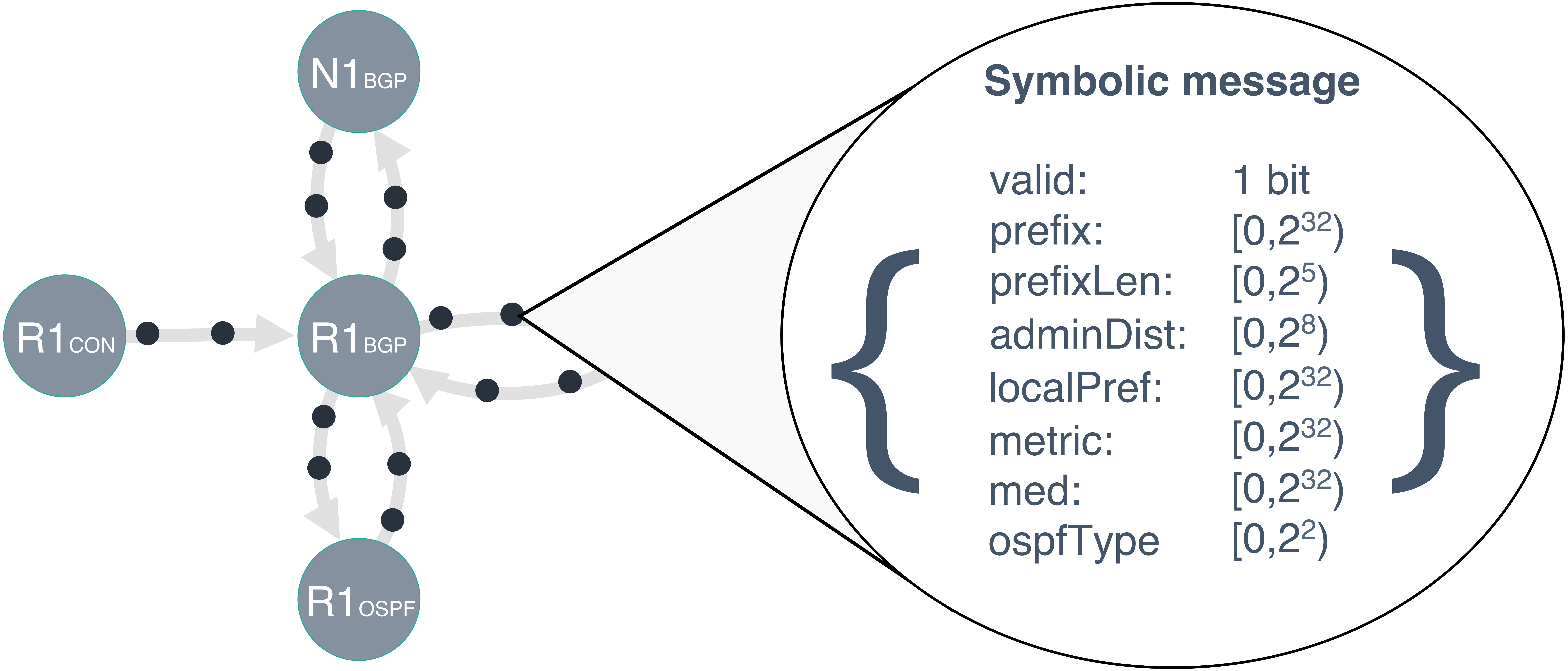


Constraint Encoding

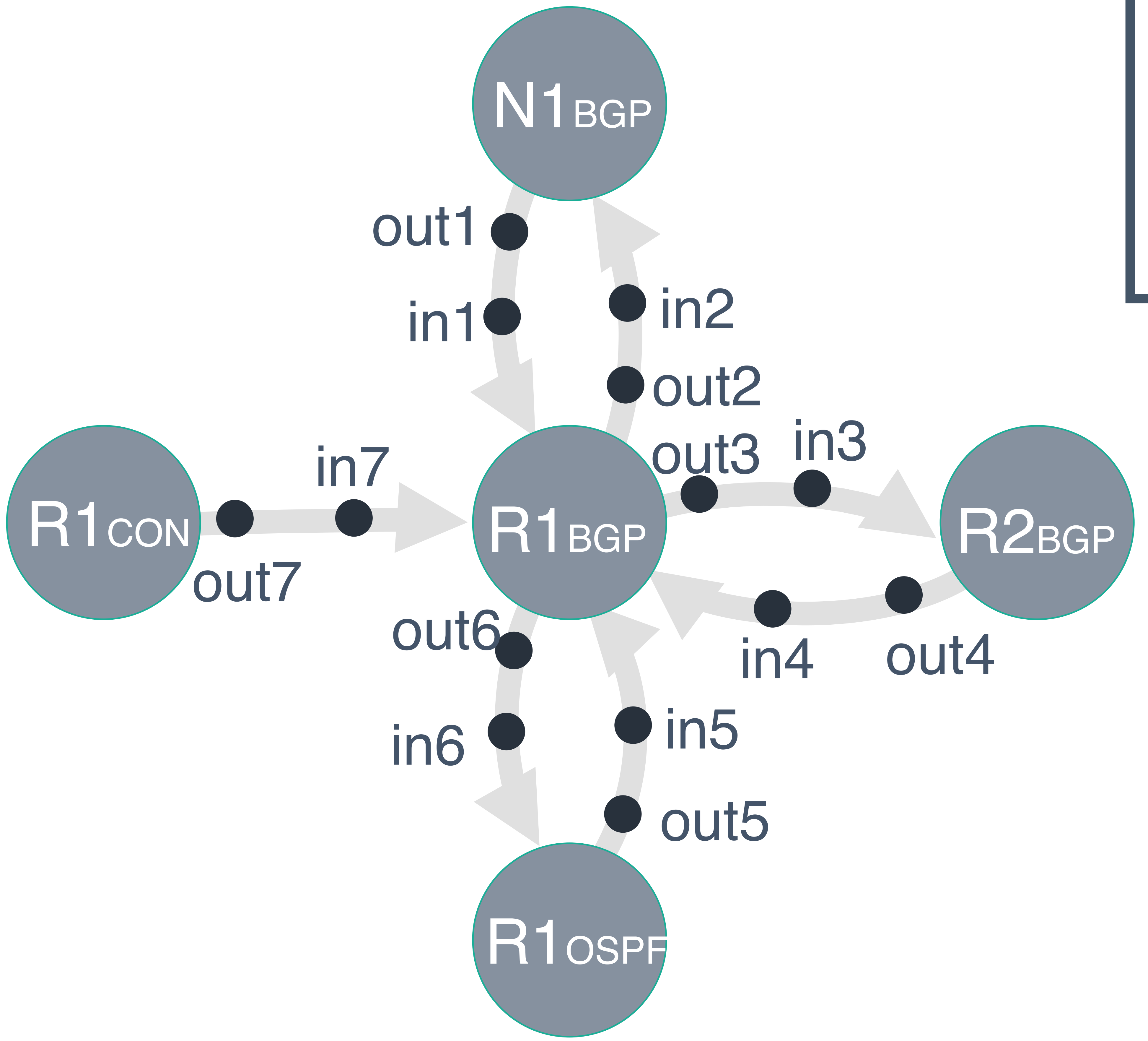




Constraint Encoding



Constraint Encoding



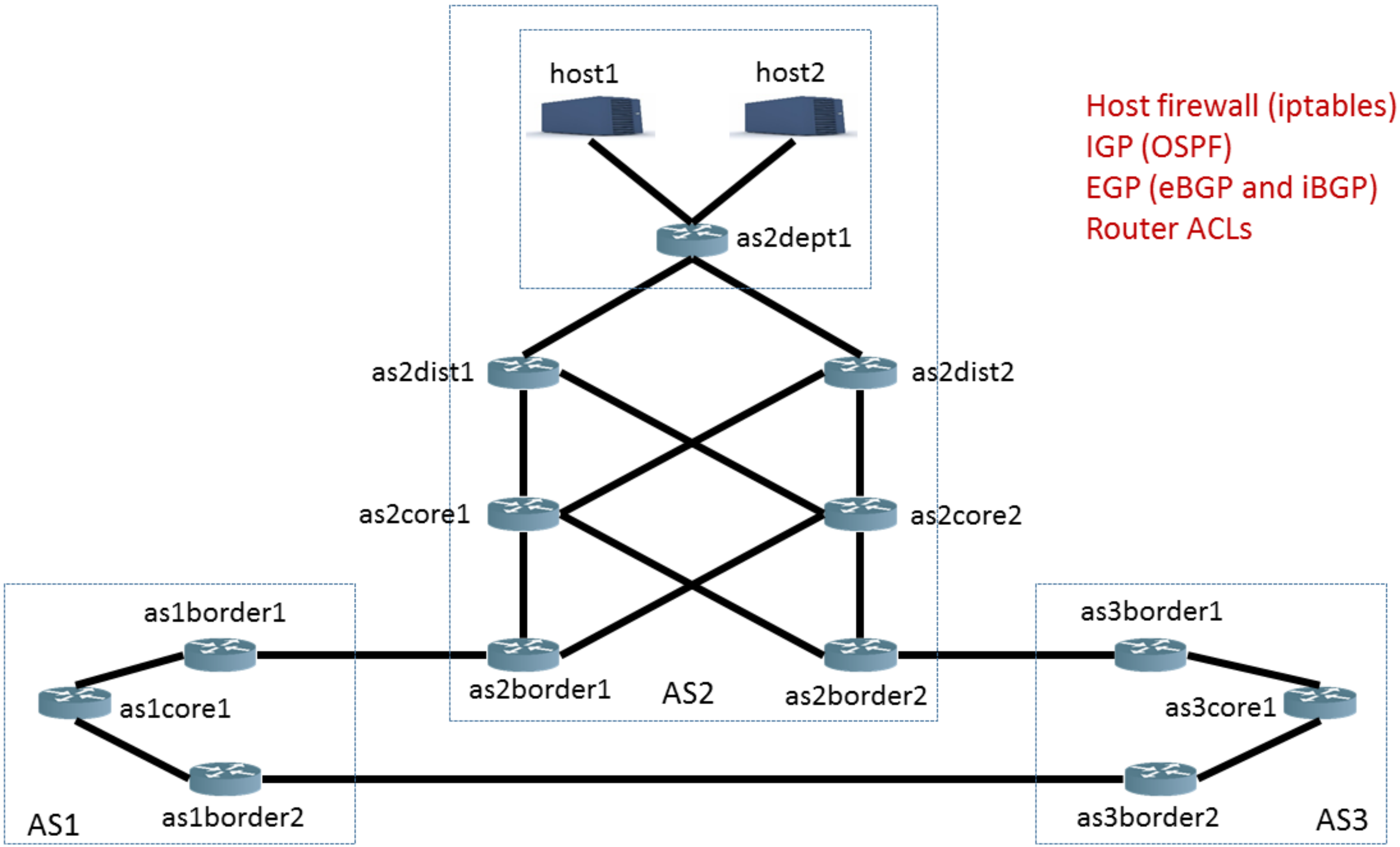
R1 BGP import filter from R2

```
ip prefix_list L deny 192.168.0.0/16 le 32
ip prefix_list L allow
route-map R1_Import_From_R2 10
  match ip address prefix-list L
  set local-preference 120
```

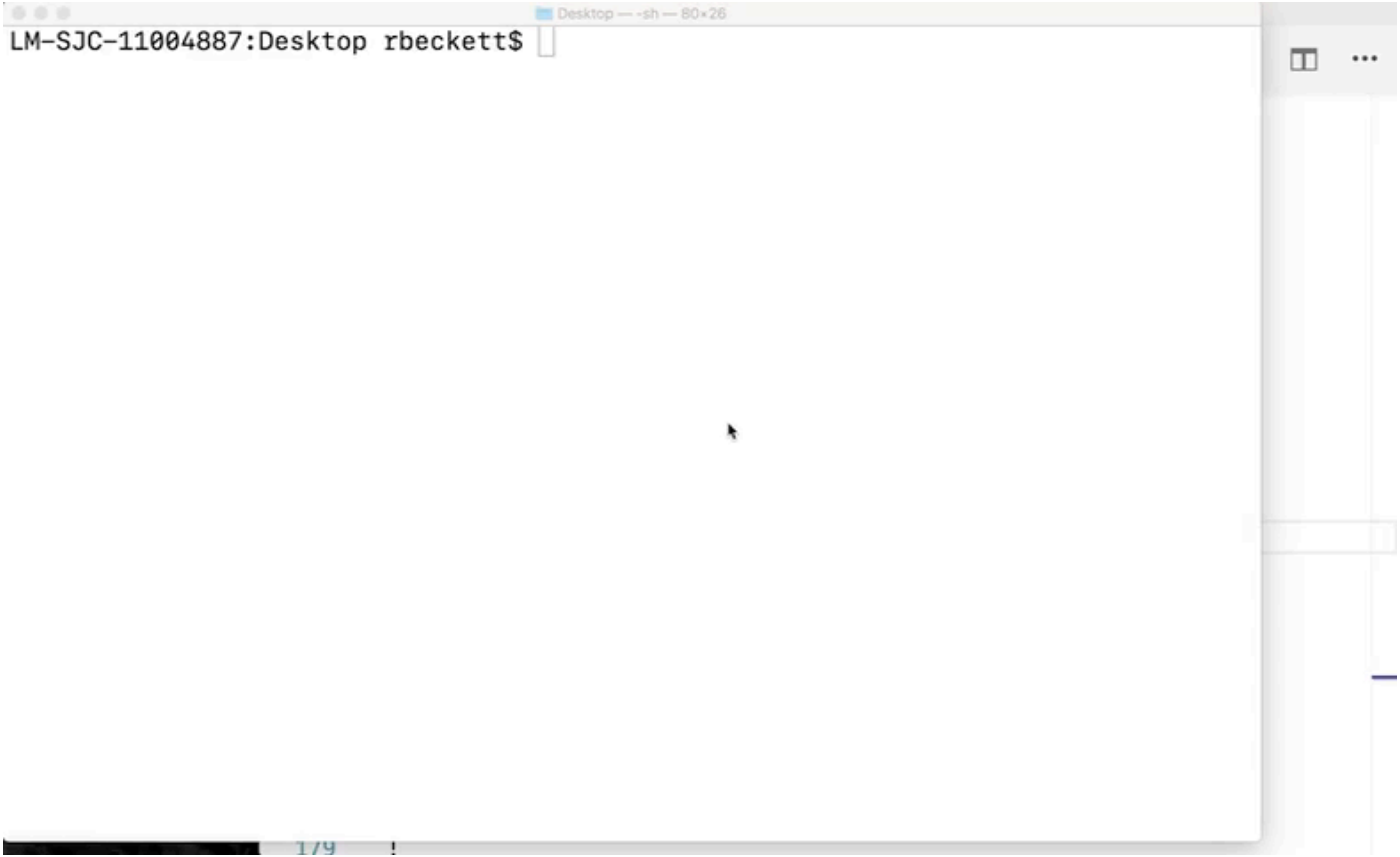
Logical Constraints

if $out_4.valid \wedge failed_{R1\_R2} = 0$ **then**
  if $\neg\,(FBM(out_4.prefix, 192.168.0.0, 16) \wedge 16 \leq out_4.prefixLen \leq 32)$ **then**
    $in_4.valid = true$
    $in_4.lp = 120$
    $in_4.ad = out_4.ad$
    $in_4.prefix = out_4.prefix$
    $in4.metric = out_4.metric$
    $in_4.prefixLen = out_4.prefixLen$
  else $in_4.valid = false$
else $in_4.valid = false$

Demo: Topology

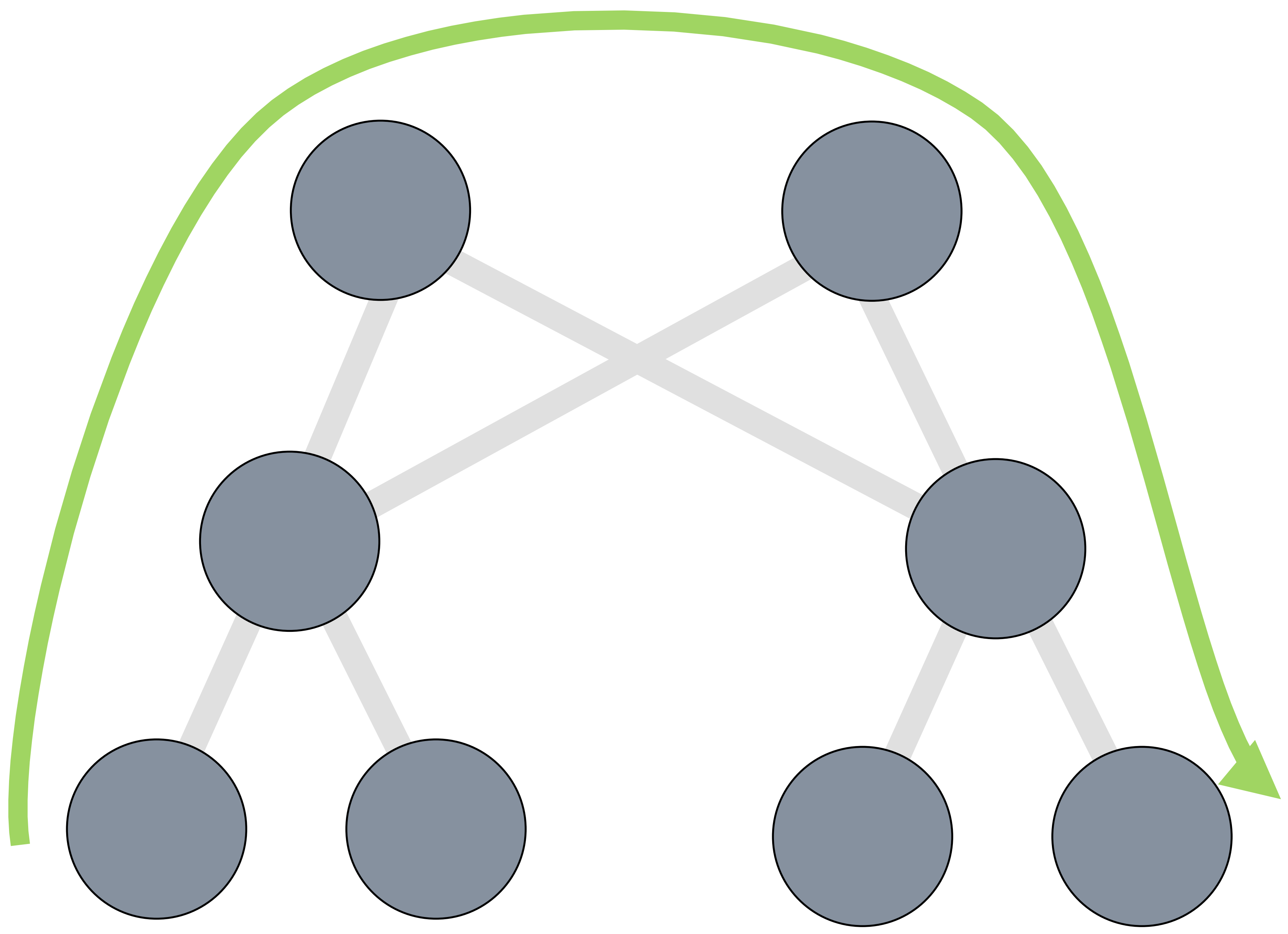


Host firewall (iptables)
IGP (OSPF)
EGP (eBGP and iBGP)
Router ACLs


Desktop — -sh — 80×26
LM-SJC-11004887:Desktop rbeckett$

Example Properties

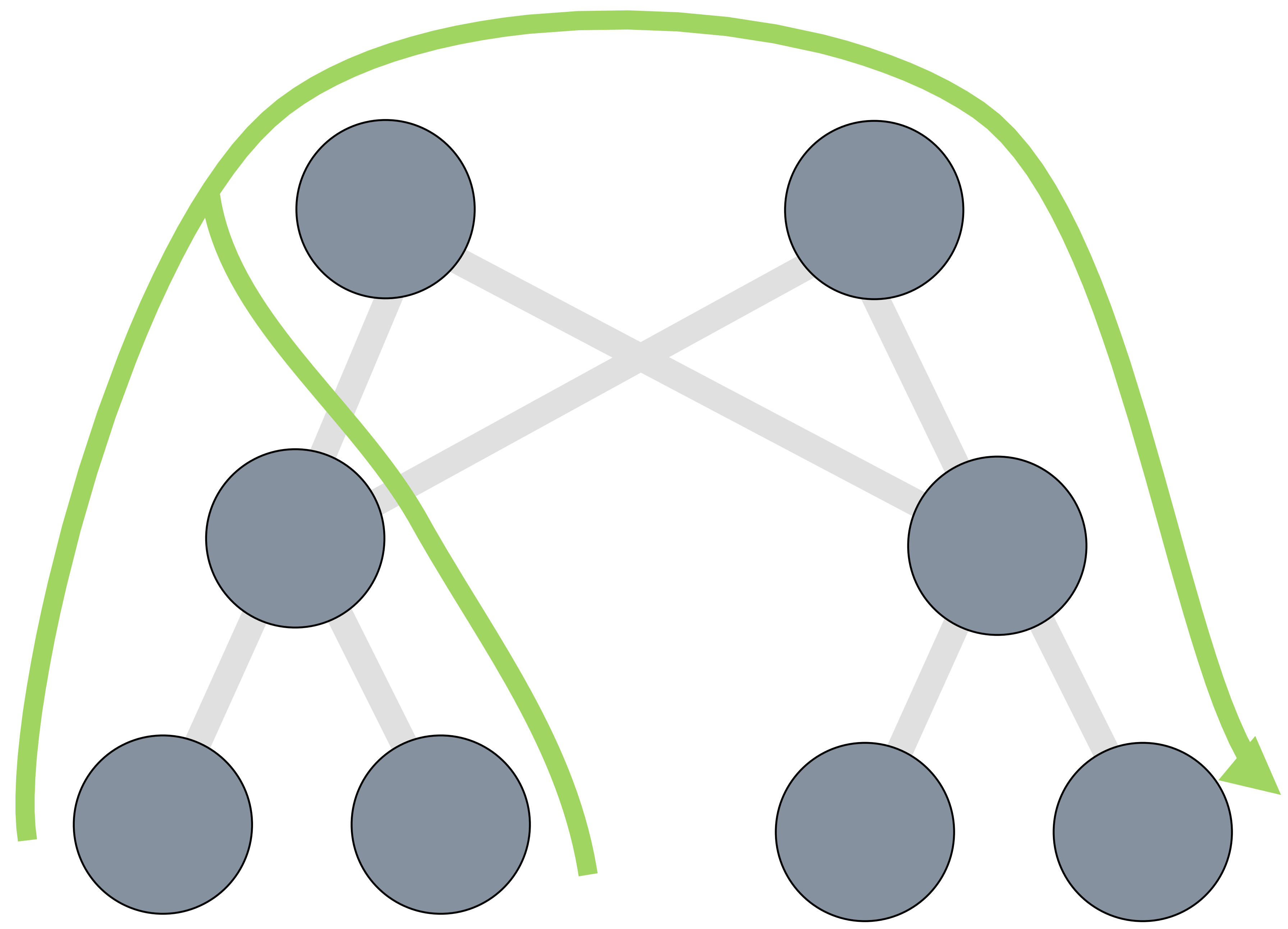
"Can router X always reach router Y"



Example Properties

> "Can router X always reach router Y"

> "Do all routers in a pod have equal length paths to a destination port?"

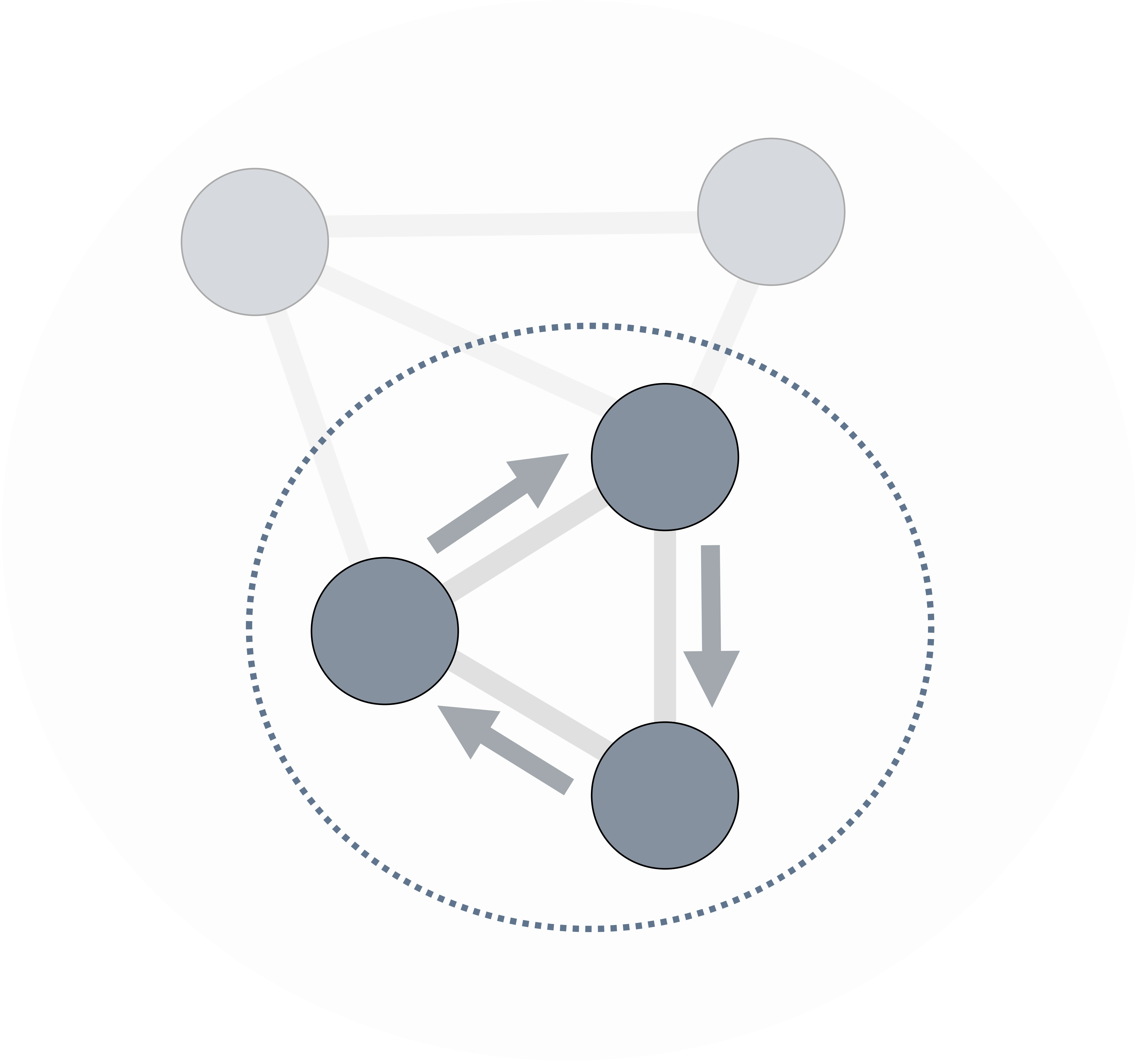


Example Properties

"Can router X always reach router Y"

"Do all routers in a pod have equal length paths to a destination port?"

"Can my network ever have loops?"



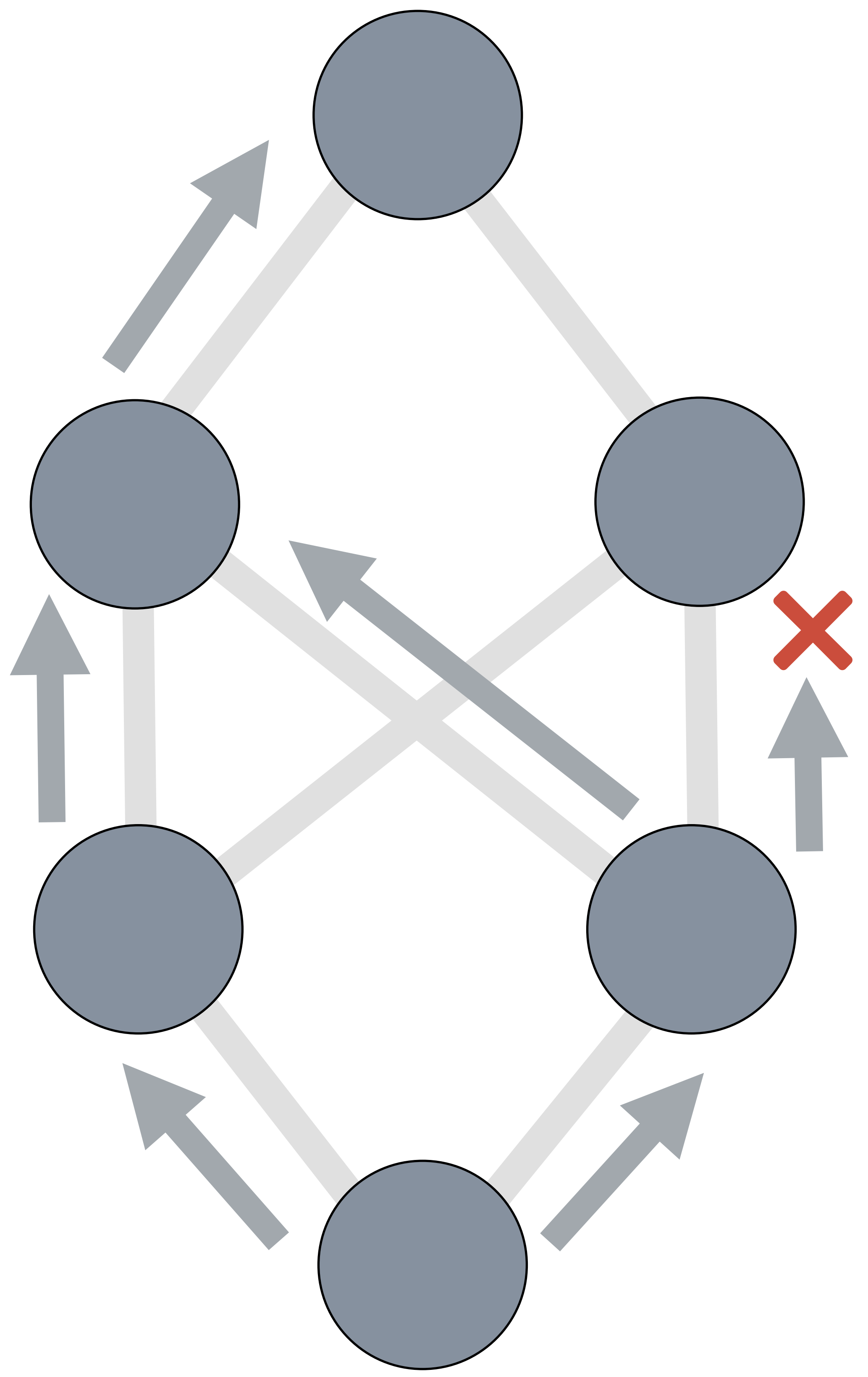
Example Properties

“Can router X always reach router Y”

“Do all routers in a pod have equal length paths to a destination port?”

“Can my network ever have loops?”

“Are multiple paths treated equally?”



Example Properties

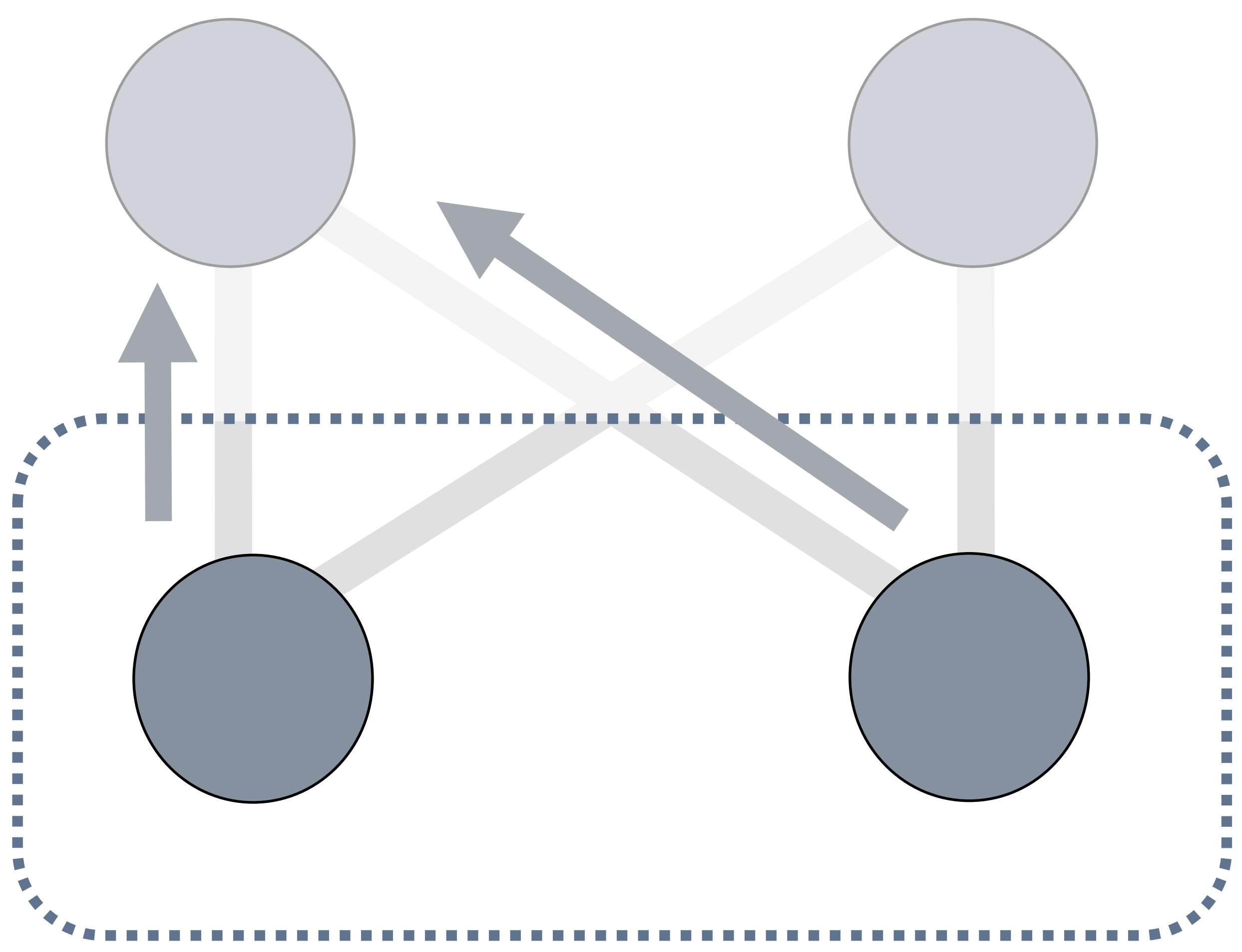
"Can router X always reach router Y"

"Do all routers in a pod have equal length paths to a destination port?"

"Can my network ever have loops?"

"Are multiple paths treated equally?"

"Do two routers serve equal roles?"



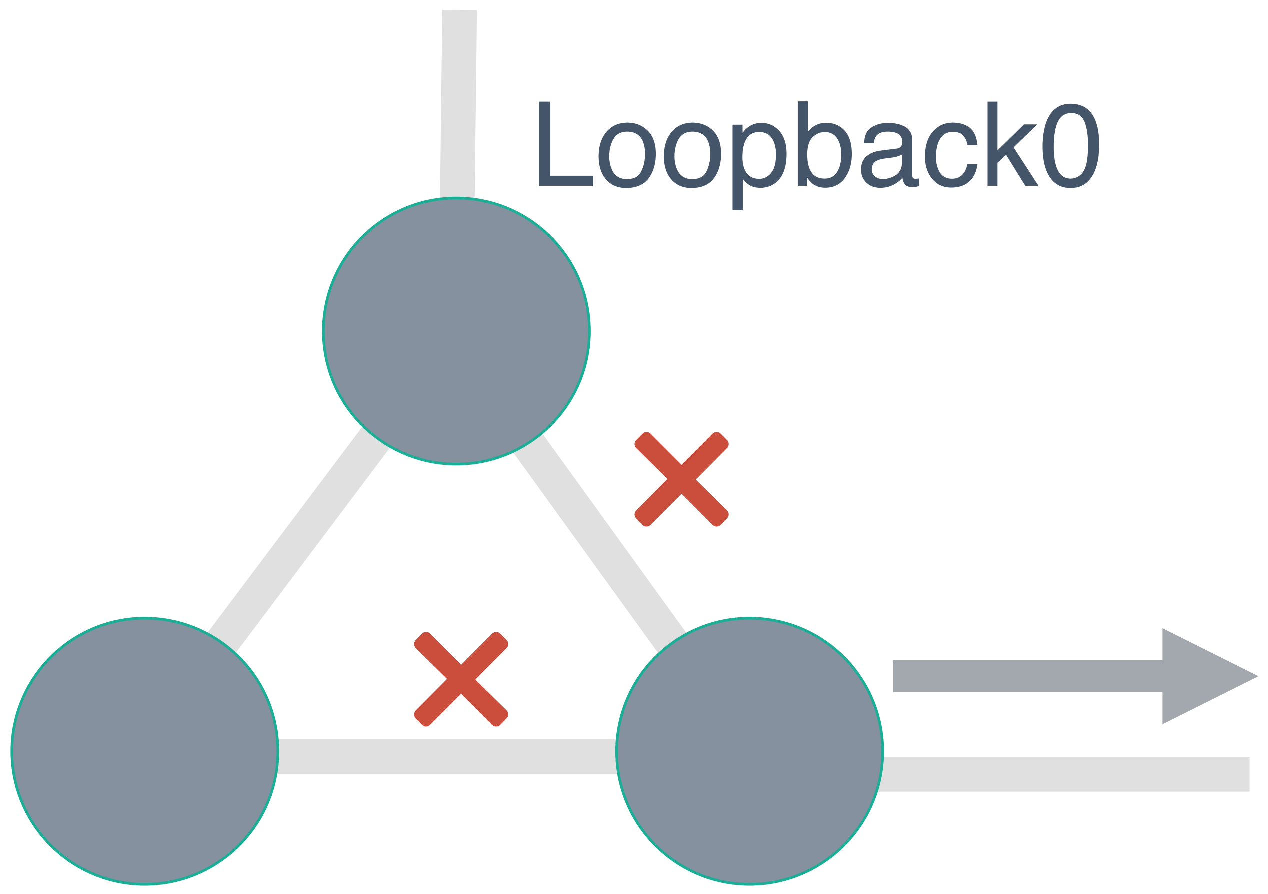
Supported Features

Features	Implemented	Continued…	
OSPF Intra-area	✓	iBGP	✓
OSPF Inter-area	✓	Route Reflectors	✓
eBGP Local-pref	✓	Static Routes	✓
eBGP Communities	✓	Route Redistribution	✓
eBGP MEDs	✓	Multipath Routing	✓
eBGP Path Prepending	✓	Access Control Lists	✓
eBGP Aggregation	✓	IPV6	✗

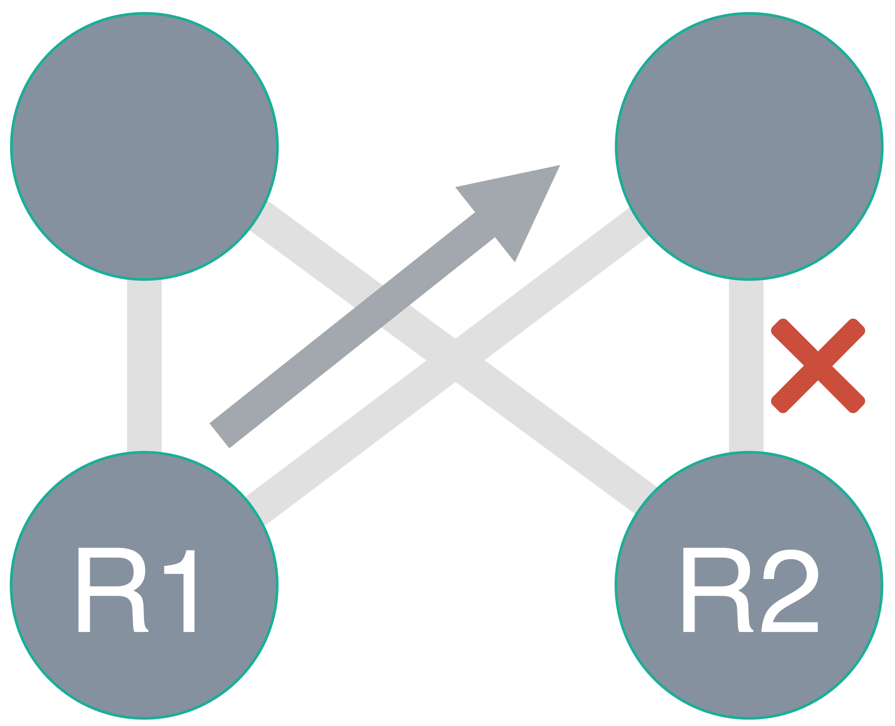
Evaluation: Bug Finding

Ran Minesweeper on **152 legacy data center networks**

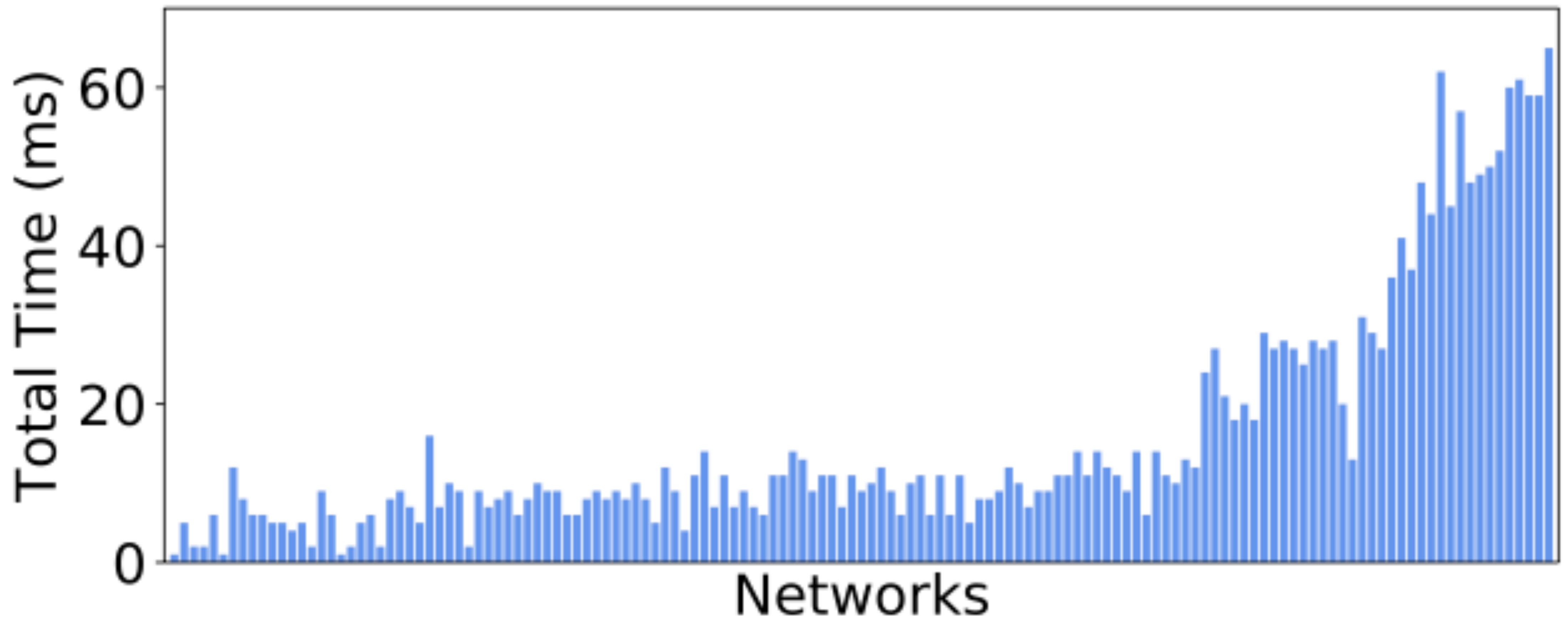
- Mangement interface reachability
  - Found 67 violations of the property
  - Each required a specific environment
  - Example: BGP peer sends /32 with length 2



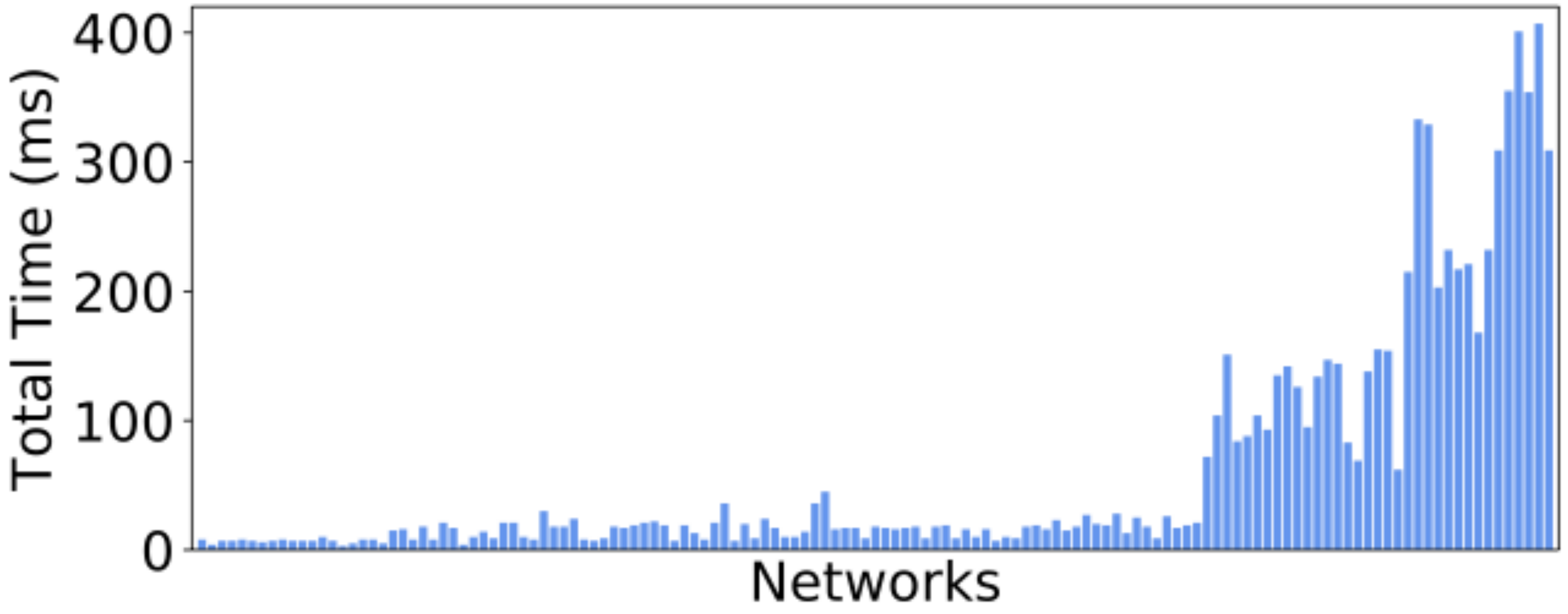
- Local equivalence of routers
  - Found 29 violations
  - Many caused by simple copy-paste errors
  - Example: ACL has missing entry



Evaluation: Scalability

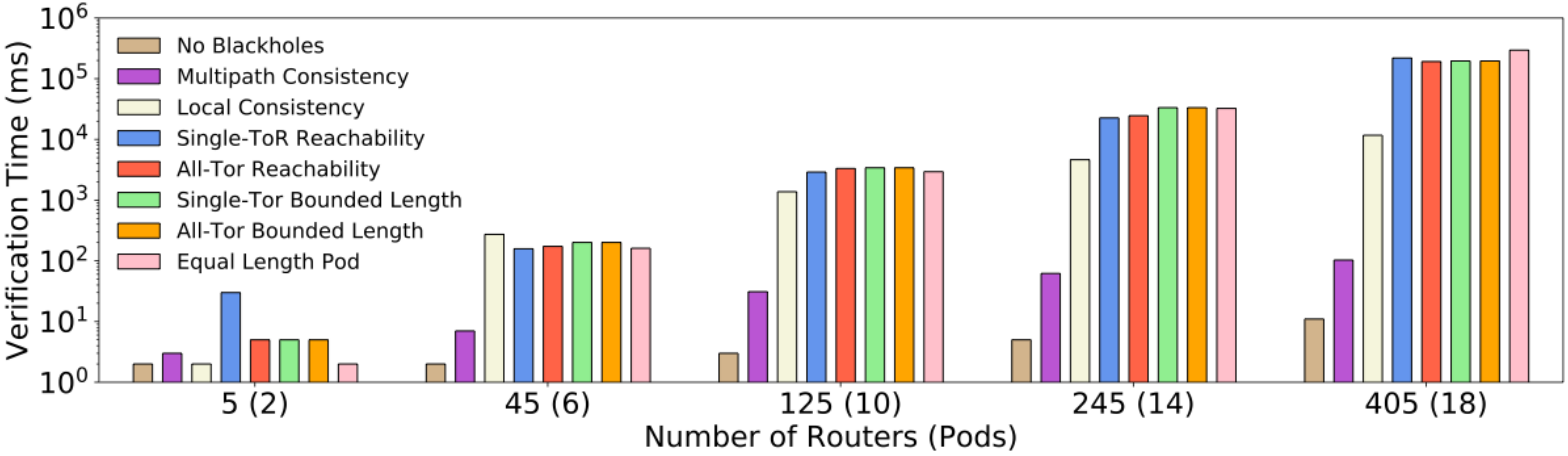


Management interface reachability



Local equivalence of routers
(For all n comparisons)

Evaluation: Scalability



Conclusion

Minesweeper is a general control plane verification tool

- Checks a wide variety of properties for all packets,
  all possible environments, and all combinations of k-failures
- Encodes the network as a hardware circuit and leverages modern off-the-shelf theorem provers
- Can find bugs in many real networks
- https://batfish.github.io/minesweeper/

Minesweeper

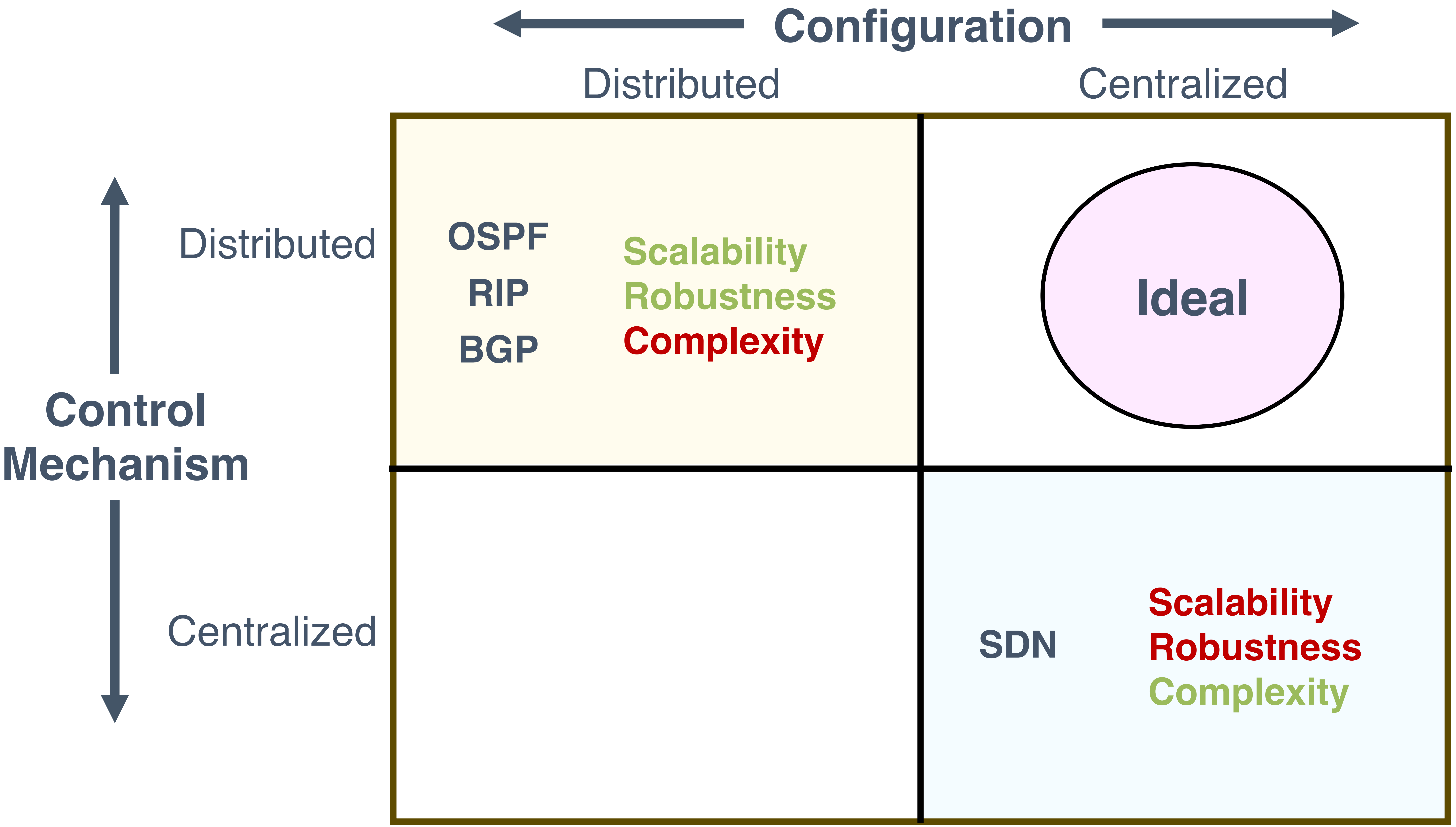
Find bugs in **legacy** networks

Propane

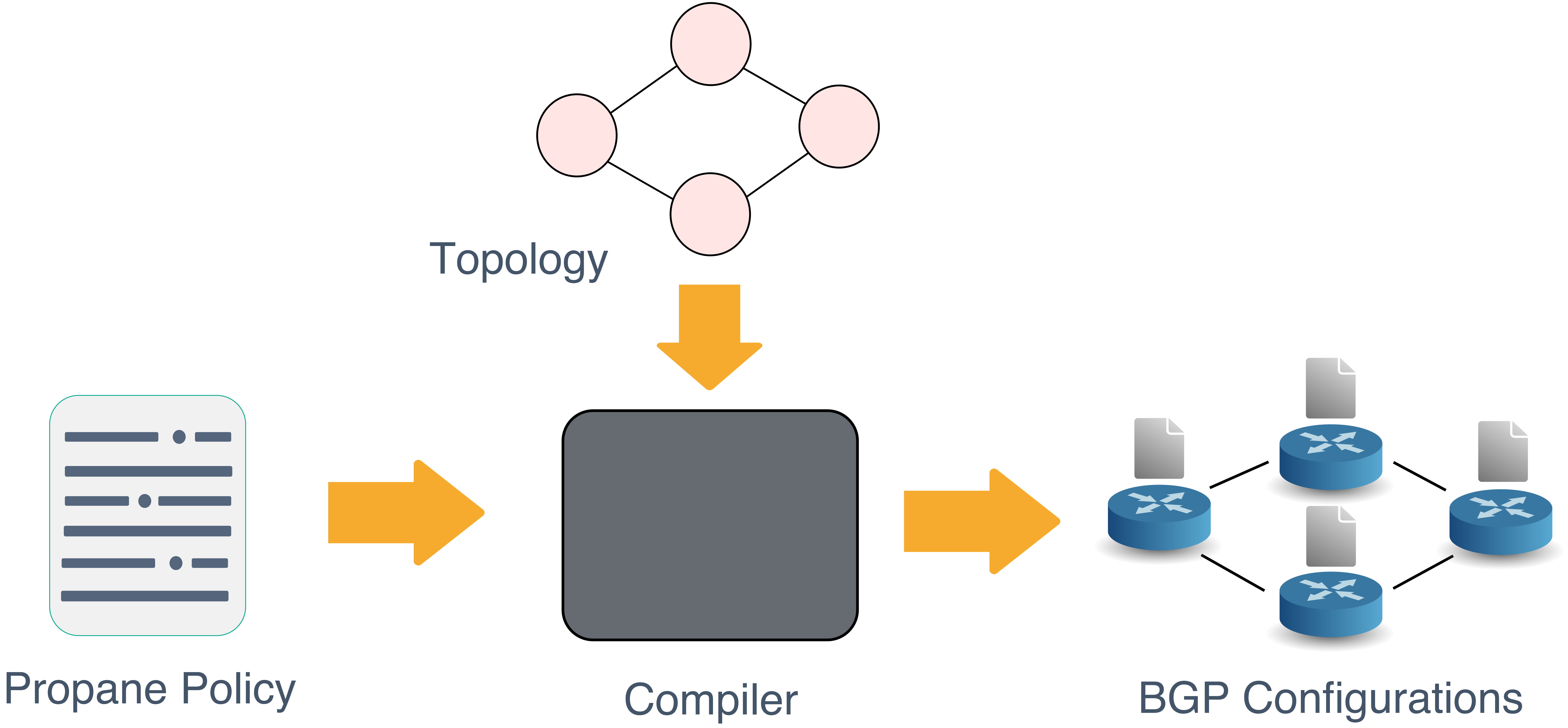
High-level design of **new** networks

Propane

Fundamental Tradeoff ?



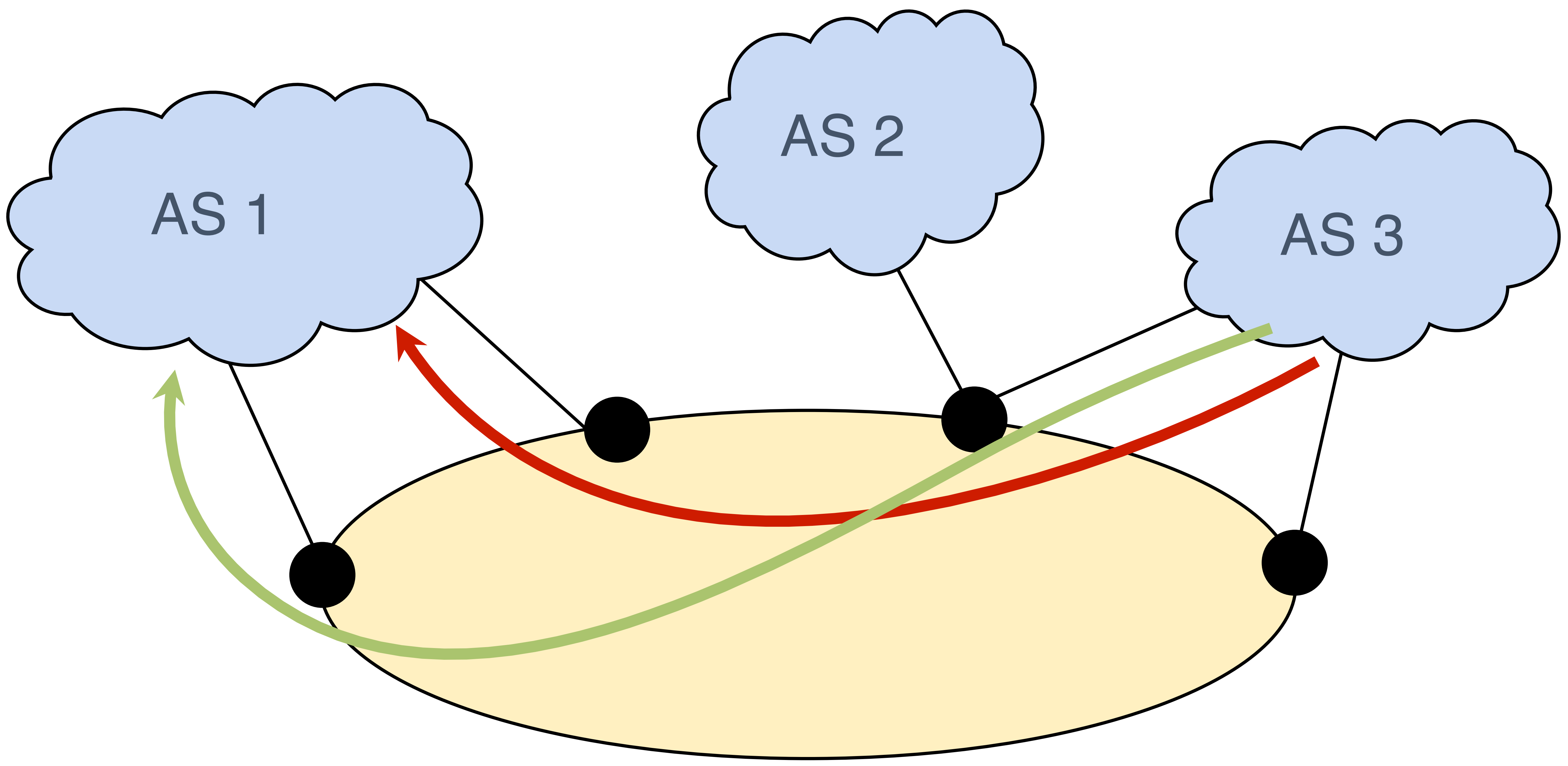
Propane Overview



Propane System

(1) Language for expressing **network-wide policy objectives** with:

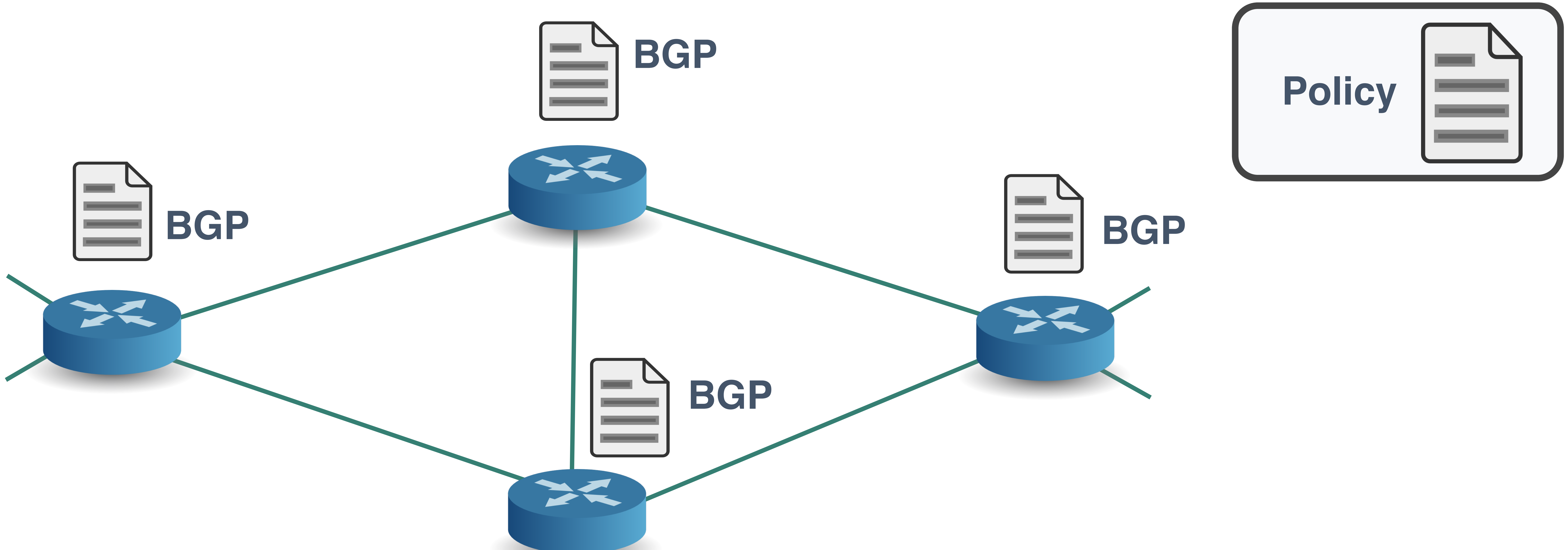
- Path **constraints** and **preferences**
- Uniform abstractions for **intra-** and **inter-**domain routing



Propane System

(2) Compiler for generating BGP configurations

- Guarantees **policy-compliance** for **all possible failures**

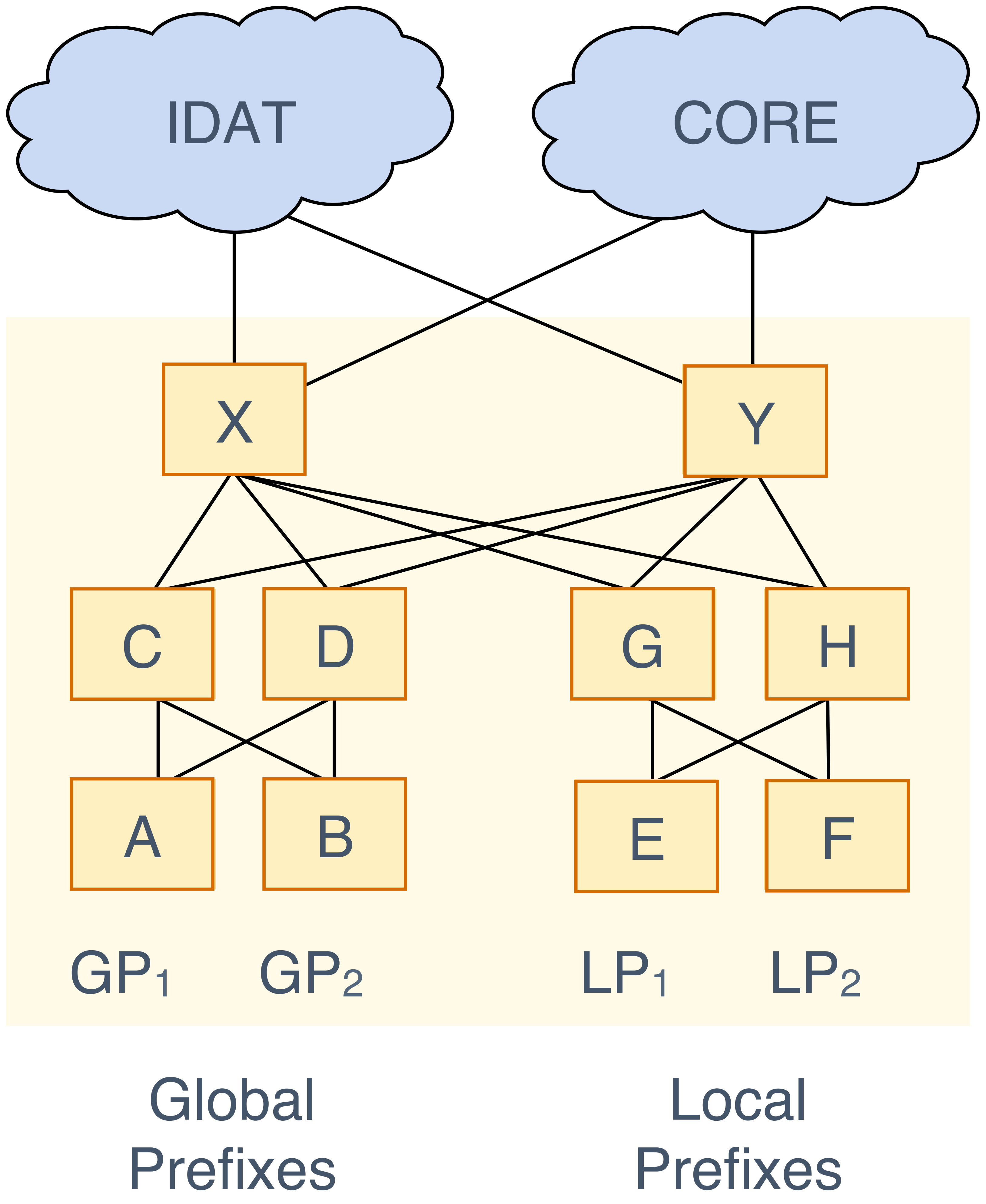


Example: A DC network

Policy Objectives

- Local prefixes reachable only internally
- Global prefixes reachable externally
- Aggregate global prefixes as GP
- Prefer leaving through $Peer_1$ over $Peer_2$
- Prevent transit traffic between peers

Demo!



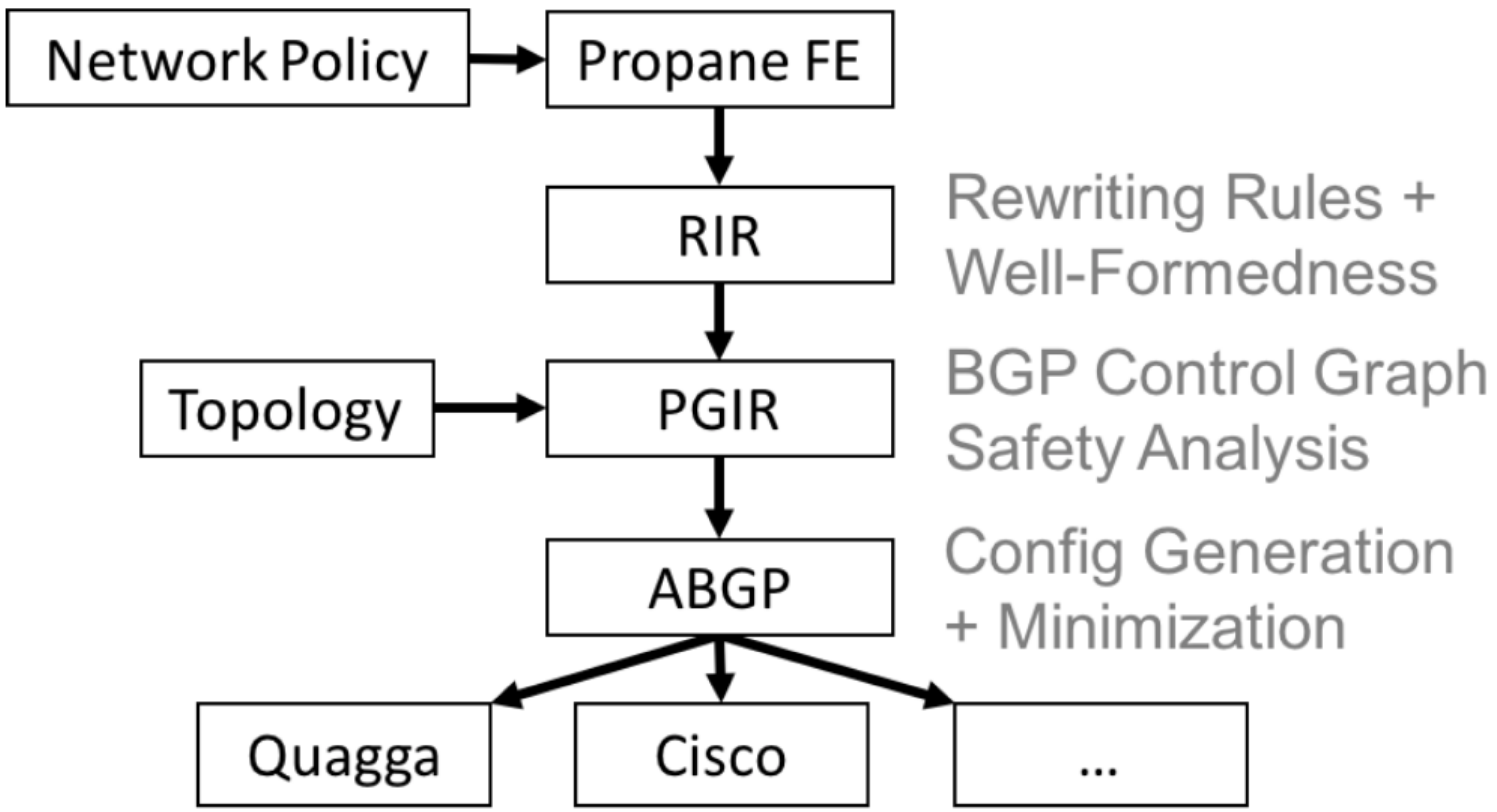
beckett:

demo.pro — demo (git: master)

```
define pfx1 = 1.0.0.0/24
define pfx2 = 1.0.1.0/24
define pfx3 = 2.0.0.0/24
define pfx4 = 2.0.1.0/24
define local_prefixes = (pfx3 or pfx4)
define peer = {IDAT, CORE}

define basic_routing = {
    pfx1 => end(A),
    pfx2 => end(B),
    pfx3 => end(E),
    pfx4 => end(F),
    true => end(out)
}

define main = basic_routing
```

Line: 16:28 | Propane | Tab Size: 4

How Compilation Works



Propane Compiler

- Generates Cisco and Quagga configs
- Includes a number of other analyses
  - Unused backup paths
  - Possible reachability issues
  - Aggregation-induced black holes
  - Unused prefixes / aggregates
- Can enable / disable MEDs, prepending, …

ryanbeckett — -bash — 86×27

```
Usage: propane [options]
       propane (--help | --version)

Options:
    -h, --help              Show this message.
    --version               Show the version of Propane.
    --policy FILE           Propane policy file.
    --topo FILE             Network topology file (xml).
    --output DIR            Specify output directory.
    --verbose               Display detailed information about fault-tolerance.
    --no-failures           Disable checks for aggregation safety
    --failures k            Guarantee k failure safety for aggregation.
    --check                 Only check for correctness, don't generate configs.
    --parallel              Enable parallel compilation.
    --naive                 Disable policy minimization.
    --stats                 Display compilation statistics in readable format.
    --csv                   Display compilation statistics in csv format.
    --anycast               Allow use of ip anycast.
    --med                   Allow use of the BGP MED attribute.
    --prepending            Allow use of AS path prepending.
    --noexport              Allow use of the BGP no-export community.
    --cbgp                  Generate C-BGP tests.
    --test                  Run compiler unit tests.
    --bench                 Generate benchmark policies.
    --debug                 Output debugging information.

beckett:~ ryanbeckett$
```

Evaluation

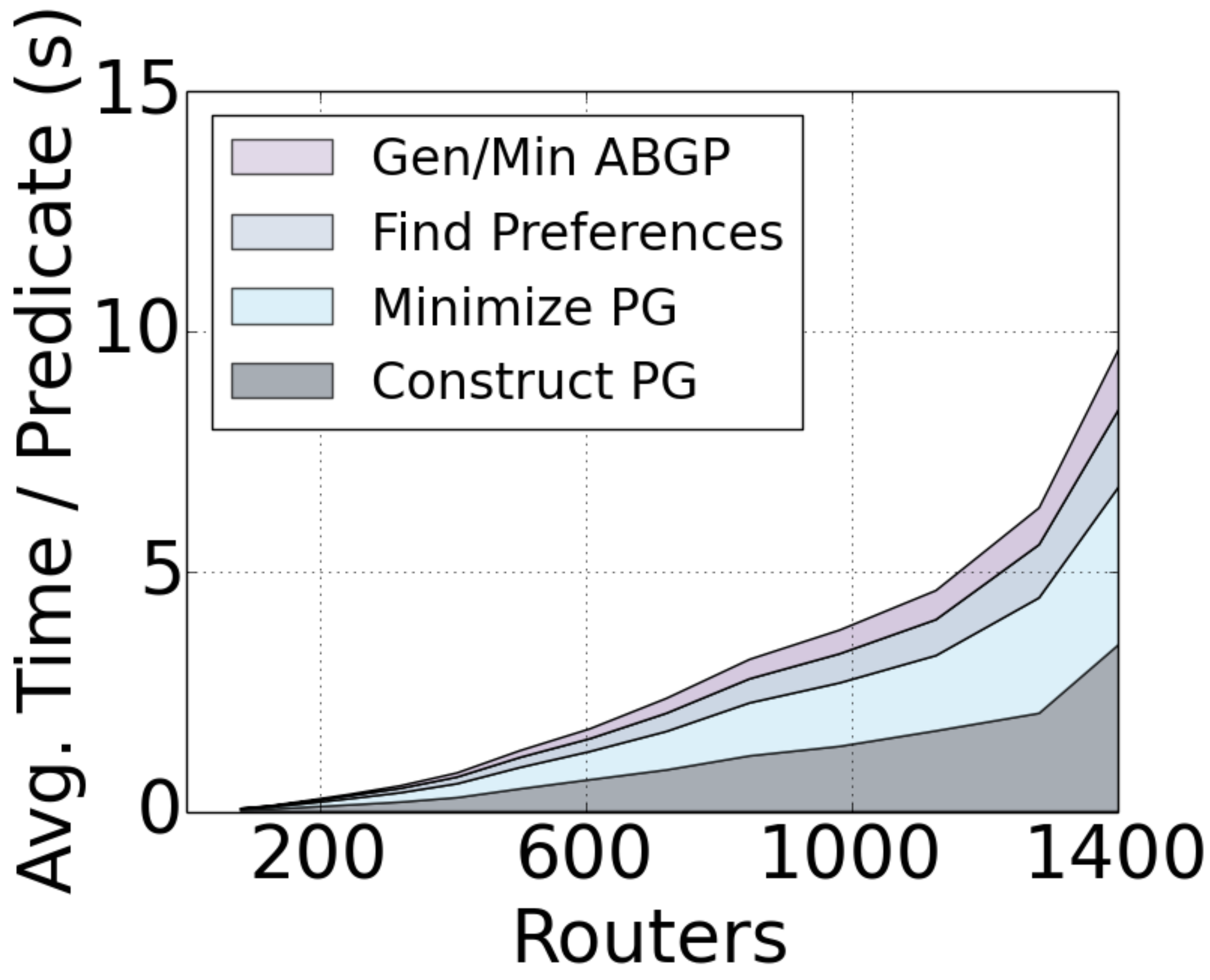
- Language expressiveness
  - Translated configurations from a large cloud provider
  - Policy described in English documents
  - Both data center and backbone networks
- Compiler performance
  - Used cloud provider's routing policy
  - Scaled the size of backbone and data center topologies

Language Expressiveness

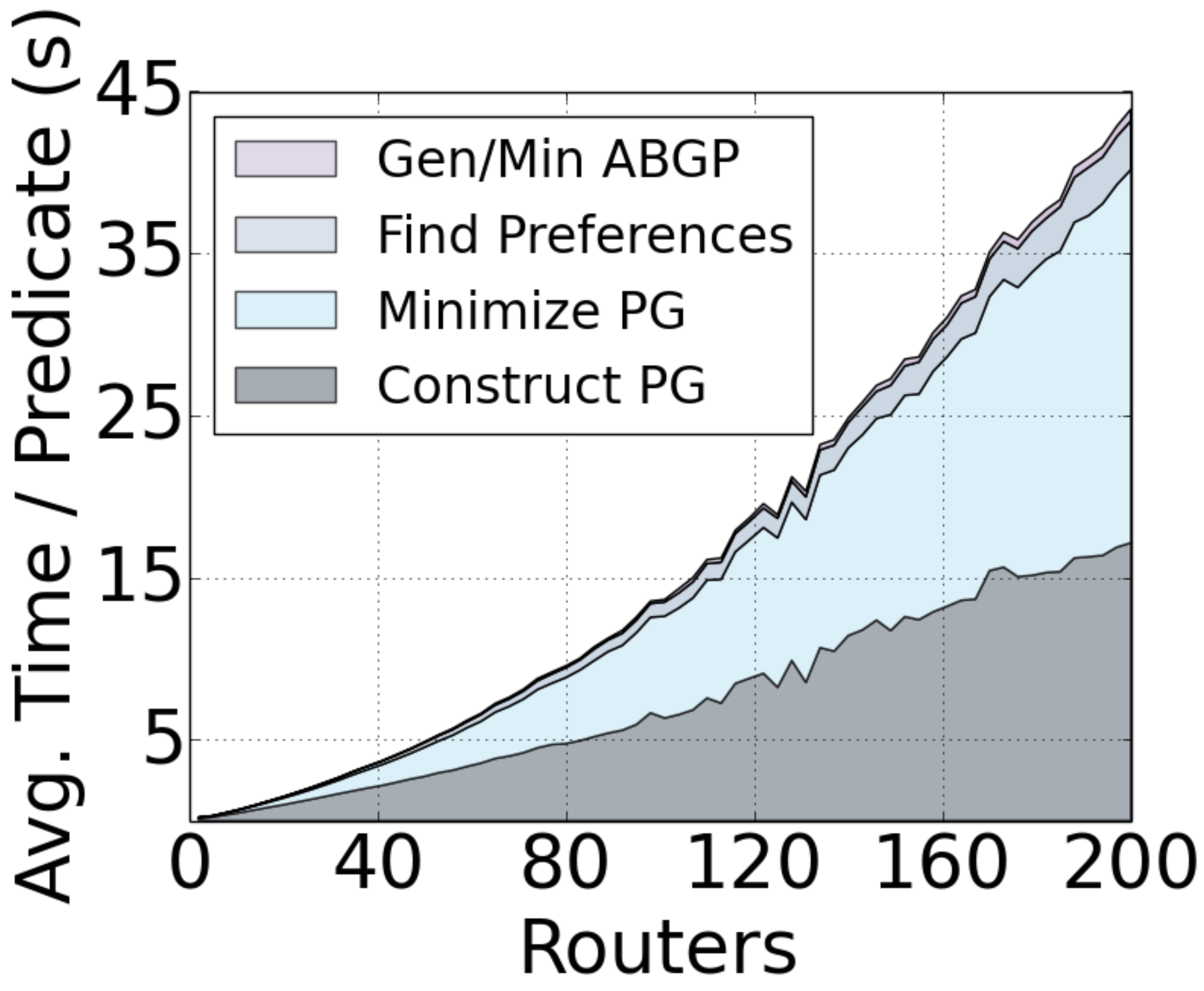
Not counting prefix / peer definitions

- Data center policy: **~30 lines** of Propane
- Backbone policy**: ~50 lines** of Propane
- Actual networks: **~1000s lines** of Configuration

Compiler Performance



Data center (< 9 min)



Backbone (< 3 min)

Conclusion

High-level language

- **Centralized** network programmability
- Constraints specify preferred paths and backup paths
- Core policy in 30-50 lines of Propane vs 1000s of config

Compiler

- **Distributed** implementation via BGP
- Static analysis guarantees policy compliance for **all failures**
- **Scales** to many large network topologies

http://www.propane-lang.org

Minesweeper

Find bugs in **legacy** networks

Propane

High-level design of **new** networks

rbeckett@princeton.edu