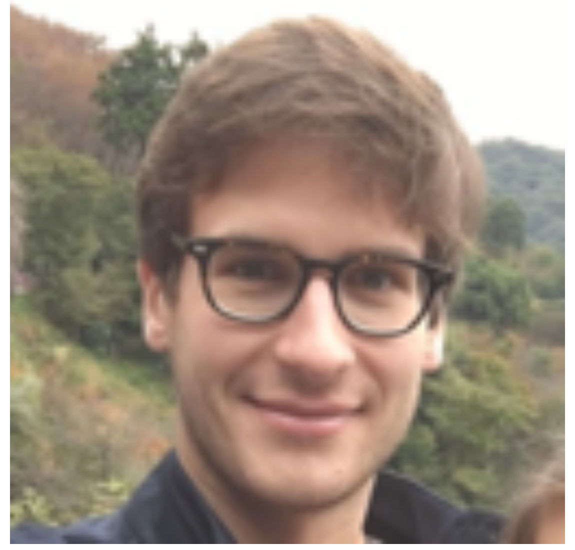
NetComplete: Practical Network-Wide Configuration Synthesis with Autocompletion






Ahmed El-Hassany | Petar Tsankov | Laurent Vanbever | Martin Vechev

SRL SOFTWARE RELIABILITY LAB

Networked Systems
ETH Zürich — seit 2015

ETH zürich

I shouldn't be the one giving this talk…

Ahmed El-Hassany

Third year PhD student @ETH Zürich

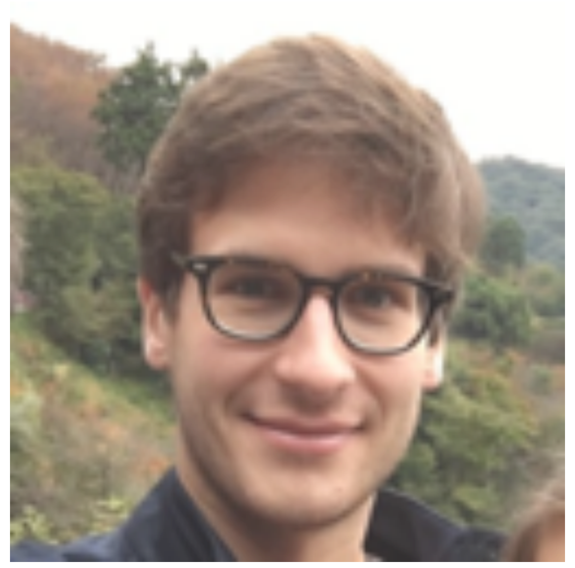
Papers at NSDI, SIGCOMM, PLDI, CAV, SOSR, …

Check him out at **hassany.ps**

NetComplete: Practical Network-Wide Configuration Synthesis with Autocompletion


Ahmed El-Hassany


Petar Tsankov


Laurent Vanbever


Martin Vechev

SRL SOFTWARE RELIABILITY LAB

Networked Systems
ETH Zürich — seit 2015

ETH zürich

Steve Uhlig

9 March at 14:30 ·

Curious if the Internet is also better during IETF/NANOG/RIPE...

Fewer heart attack patients die when top cardiologists are away at conferences, study finds

Heart attack patients are more likely to survive when top cardiologists are not in the hospital, a new study suggests.Researchers at Harvard Medical School...

FLIP.IT

Like Comment Share

Steve Uhlig
9 March at 14:30 ·

Curious if the Internet is also better during IETF/NANOG/RIPE...

Fewer heart attack patients die when top cardiologists are away at conferences, study finds

Heart attack patients are more likely to survive when top cardiologists are not in the hospital, a new study suggests.Researchers at Harvard Medical School...

FLIP.IT

Like Comment Share

Steve Uhlig

9 March at 14:30 ·

Curious if the Internet is also better during IETF/NANOG/RIPE...

Fewer heart attack patients die when top cardiologists are away at conferences, study finds

Heart attack patients are more likely to survive when top cardiologists are not in the hospital, a new study suggests.Researchers at Harvard Medical School...

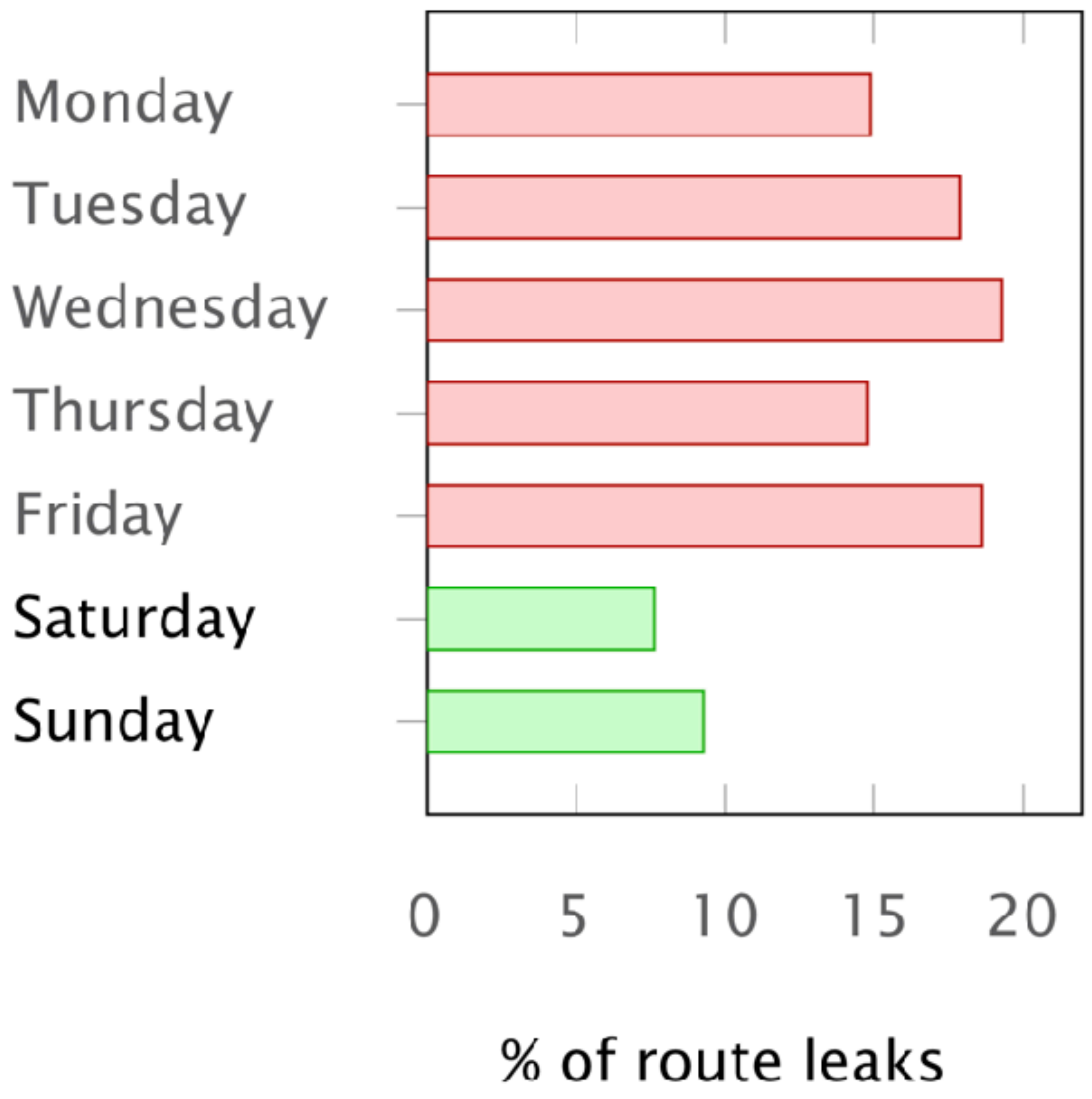
FLIP.IT

Like Comment Share

Yes.

Yes.

The Internet seems to be better off during week-ends…



source: Job Snijders (NTT)

“Human factors are responsible

for 50% to 80% of network outages”

Juniper Networks, *What’s Behind Network Downtime?*, 2008

The Register®
Biting the hand that feeds IT

Data Centre ▸ Networks

Google routing blunder sent Japan's Internet dark on Friday

Another big BGP blunder

By Richard Chirgwin 27 Aug 2017 at 22:35 40 SHARE ▼

Last Friday, someone in Google fat-thumbed a border gateway protocol (BGP) advertisement and sent Japanese Internet traffic into a black hole.

The trouble began when The Chocolate Factory "leaked" a big route table to Verizon, the result of which was traffic from Japanese giants like NTT and KDDI was sent to Google on the expectation it would be treated as transit.

Since Google doesn't provide transit services, as BGP Mon explains, that traffic either filled a link beyond its capacity, or hit an access control list, and disappeared.

The outage in Japan only lasted a couple of hours, but was so severe that Japan Times reports the country's Internal Affairs and Communications ministries want carriers to report on what went wrong.

BGP Mon dissects what went wrong here, reporting that more than

https://www.theregister.co.uk/2017/08/27/google_routing_blunder_sent_japans_internet_dark/

In August 2017

Someone in Google fat-thumbed a
Border Gateway Protocol (BGP) advertisement
and sent Japanese Internet traffic into a black hole.

In August 2017

Someone in Google fat-thumbed a
Border Gateway Protocol (BGP) advertisement
and sent Japanese Internet traffic into a black hole.

[…] Traffic from Japanese giants like NTT and KDDI
was sent to Google on the expectation
it would be treated as transit.

In August 2017

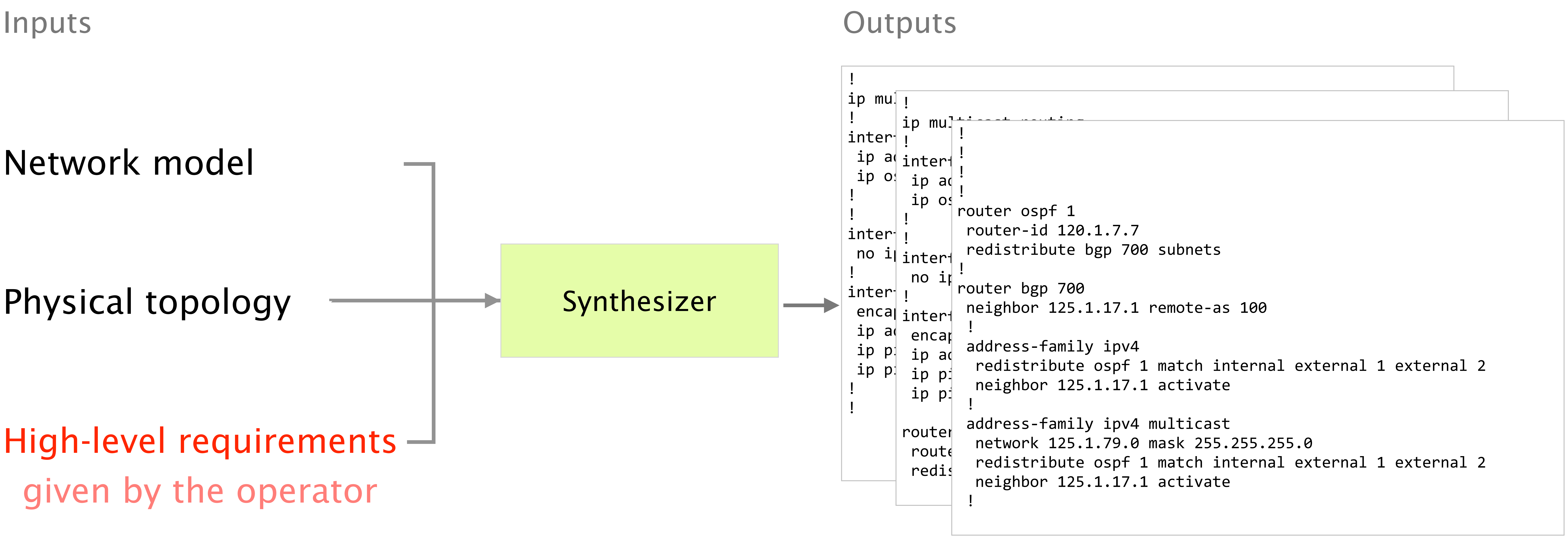
Someone in Google fat-thumbed a
Border Gateway Protocol (BGP) advertisement
and sent Japanese Internet traffic into a black hole.

[…] Traffic from Japanese giants like NTT and KDDI
was sent to Google on the expectation
it would be treated as transit.

The outage in Japan *only* lasted a couple of hours
but was so severe that […] the country's
Internal Affairs and Communications ministries
want carriers to report on what went wrong.

Configuration synthesis addresses this problem by deriving low-level configurations from high-level requirements

Configuration synthesis addresses this problem by deriving low-level configurations from high-level requirements



Configuration synthesis:
a booming research area!

Out of high-level requirements,
automatically derive…

Genesis [POPL’17]	forwarding rules
Propane [SIGCOMM’16] PropaneAT [PLDI’17]	BGP configurations
SyNET [CAV’17] Zeppelin [SIGMETRICS’18]	OSPF + BGP configurations

Synthesizing configuration is great, but comes with challenges preventing a wide adoption

Existing synthesizers…

Existing synthesizers…

Problem #1
interpretability

can produce configurations that
widely differ from humanly-generated ones

Existing synthesizers…

Problem #1
interpretability

can produce configurations that
widely differ from humanly-generated ones

Problem #2
continuity

can produce widely different configurations
given slightly different requirements

Existing synthesizers…

Problem #1
interpretability

can produce configurations that
widely differ from humanly-generated ones

Problem #2
continuity

can produce widely different configurations
given slightly different requirements

Problem #3
deployability

cannot flexibly adapt to operational requirements,
requiring configuration heterogeneity

A key issue is that synthesizers do not provide operators with a fine-grained control over the synthesized configurations

Introducing…

NetComplete

NetComplete allows network operators to flexibly express their intents through configuration sketches

A configuration with "holes"

```
interface TenGigabitEthernet1/1/1
  ip address ? ?
  ip ospf cost 10 < ? < 100

router ospf 100
  ?
  ...

router bgp 6500
  ...
  neighbor AS200 import route-map imp-p1
  neighbor AS200 export route-map exp-p1
  ...
ip community-list C1 permit ?
ip community-list C2 permit ?
```

```
route-map imp-p1 permit 10
  ?

route-map exp-p1 ? 10
  match community C2
route-map exp-p2 ? 20
  match community C1
...
```

```
interface TenGigabitEthernet1/1/1
  ip address ? ?
  ip ospf cost 10 < ? < 100

router ospf 100
  ?
  ...

router bgp 6500
  ...
  neighbor AS200 import route-map imp-p1
  neighbor AS200 export route-map exp-p1
  ...
ip community-list C1 permit ?
ip community-list C2 permit ?
```

Holes can identify specific attributes such as:

- IP addresses
- link costs
- BGP local preferences

```
interface TenGigabitEthernet1/1/1
  ip address ? ?
  ip ospf cost 10 < ? < 100

router ospf 100
  ?
  ...

router bgp 6500
  ...
  neighbor AS200 import route-map imp-p1
  neighbor AS200 export route-map exp-p1
  ...
ip community-list C1 permit ?
ip community-list C2 permit ?
```

```
route-map imp-p1 permit 10
  ?
route-map exp-p1 ? 10
  match community C2
route-map exp-p2 ? 20
  match community C1
...
```

Holes can also identify entire pieces of the configuration

NetComplete “autocompletes” the holes such that
the output configuration complies with the requirements

```
interface TenGigabitEthernet1/1/1
  ip address ? ?
  ip ospf cost 10 < ? < 100

router ospf 100
  ?
  ...

router bgp 6500
  ...
  neighbor AS200 import route-map imp-p1
  neighbor AS200 export route-map exp-p1
  ...
ip community-list C1 permit ?
ip community-list C2 permit ?
```

```
route-map imp-p1 permit 10
  ?

route-map exp-p1 ? 10
  match community C2
route-map exp-p2 ? 20
  match community C1
...
```

```
interface TenGigabitEthernet1/1/1
  ip address 10.0.0.1 255.255.255.254
  ip ospf cost 15

router ospf 100
  network 10.0.0.1 0.0.0.1 area 0.0.0.0


router bgp 6500
  ...
  neighbor AS200 import route-map imp-p1
  neighbor AS200 export route-map exp-p1
  ...
ip community-list C1 permit 6500:1
ip community-list C2 permit 6500:2
```

```
route-map imp-p1 permit 10
  set community 6500:1
  set local-pref 50
route-map exp-p1 permit 10
  match community C2
route-map exp-p2 deny 20
  match community C1
...
```

NetComplete reduces the autocompletion problem to a constraint satisfaction problem

First

Encode the

- protocol semantics
- high-level requirements
- partial configurations

as a logical formula (in SMT)

First Encode the

- protocol semantics
- high-level requirements
- partial configurations

as a logical formula (in SMT)

Then Use a solver (Z3) to find an assignment for the undefined configuration variables s.t. the formula evaluates to True

Main challenge:

Scalability

Insight #1

network-specific heuristics

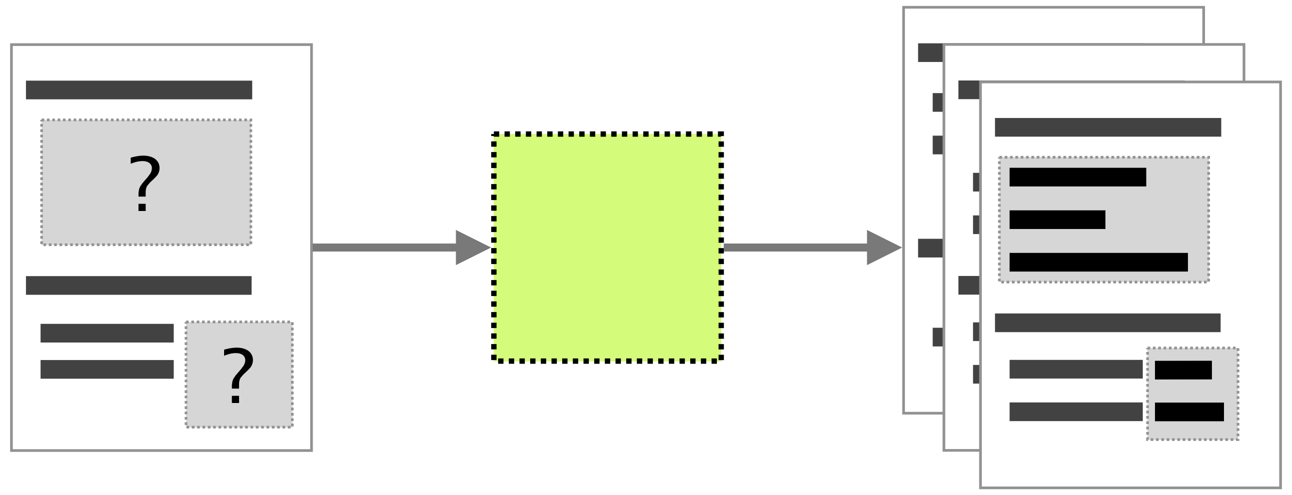
search space navigation

Insight #2

partial evaluation

search space reduction

NetComplete: Practical Network-Wide Configuration Synthesis with Autocompletion

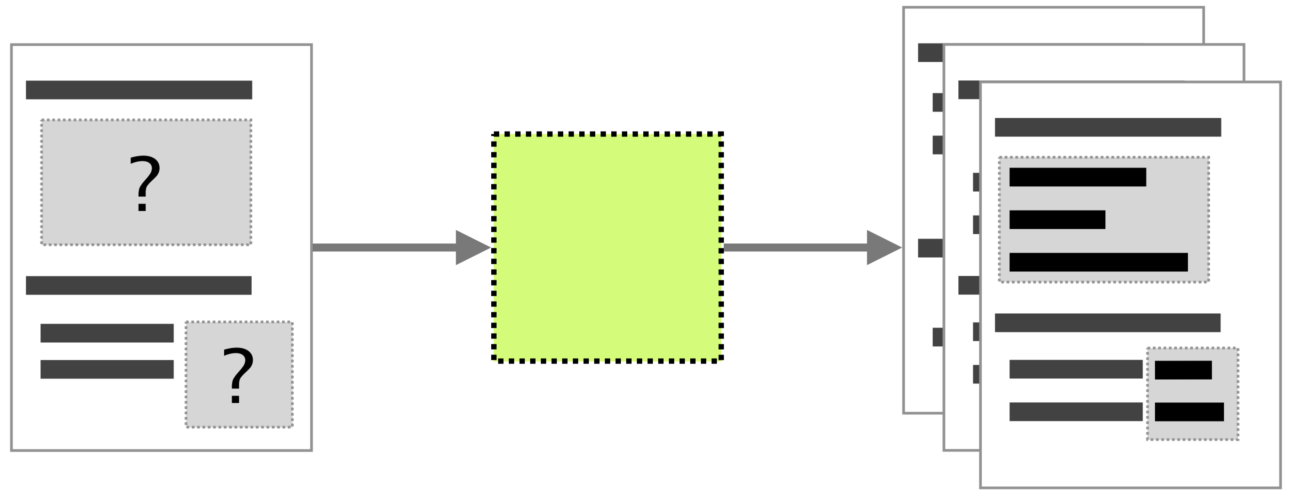


1 **BGP synthesis**
optimized encoding

2 **OSPF synthesis**
counter-examples-based

3 **Evaluation**
flexible, *yet* scalable

NetComplete: Practical Network-Wide Configuration Synthesis with Autocompletion



1 BGP synthesis

optimized encoding

OSPF synthesis

counter-examples-based

Evaluation

flexible, *yet* scalable

NetComplete autocompletes router-level BGP policies by encoding the desired BGP behavior as a logical formula

$$M \models \text{Reqs} \land \text{BGP}_{\text{protocol}} \land \text{Policies}$$

$$M \models \text{Reqs} \land \text{BGP}_{\text{protocol}} \land \text{Policies}$$

how should the network forward traffic

concrete, part of the input

$$M \models \text{Reqs} \wedge BGP_{protocol} \wedge \text{Policies}$$

$R1.BGP_{select}(A1,A2) \wedge$

$R1.BGP_{select}(A2,A3) \wedge \ldots$

concrete, protocol semantic

how do BGP routers select routes

$$M \models \text{Reqs} \wedge \text{BGP}_{\text{protocol}} \wedge \text{Policies}$$

$$BGP_{select}(X,Y) \Leftrightarrow (X.LocalPref > Y.LocalPref) \lor \ldots$$

$$M \models Reqs \land BGP_{protocol} \land Policies$$

$$M \models \mathtt{Reqs} \land \mathtt{BGP}_{\mathtt{protocol}} \land \mathtt{Policies}$$

how routes should be modified

symbolic, to be found

$$M \models \mathtt{Reqs} \wedge \mathtt{BGP}_{\mathtt{protocol}} \wedge \mathtt{Policies}$$

R1.SetLocalPref(A1) = VarX

R1.SetLocalPref(A2) = 200

Solving this logical formula consists in assigning each symbolic variable with a concrete value

$$BGP_{select}(X,Y) \Leftrightarrow (X.LocalPref > Y.LocalPref) \vee \ldots$$

$$M \models Reqs \wedge BGP_{protocol} \wedge Policies$$

$R1.BGP_{select}(A1,A2) \wedge$

$R1.BGP_{select}(A2,A3) \wedge \ldots$

R1.SetLocalPref(A1) = VarX

R1.SetLocalPref(A2) = 200

$BGP_{select}(X,Y) \Leftrightarrow (X.LocalPref > Y.LocalPref) \vee \ldots$

$$M \models Reqs \wedge BGP_{protocol} \wedge Policies$$

R1.BGP_{select}(A1,A2) ∧
R1.BGP_{select}(A2,A3) ∧ …

R1.SetLocalPref(A1) = VarX
R1.SetLocalPref(A2) = 200

$$BGP_{select}(X,Y) \Leftrightarrow (X.LocalPref > Y.LocalPref) \lor \ldots$$

$$VarX := 250 \quad — \quad M \models Reqs \land BGP_{protocol} \land Policies$$

$R1.BGP_{select}(A1,A2) \land$

$R1.BGP_{select}(A2,A3) \land \ldots$

$R1.SetLocalPref(A1) = VarX$

$R1.SetLocalPref(A2) = 200$

Naive encodings lead to complex constraints
that cannot be solved in a reasonable time

Naive encodings lead to complex constraints that cannot be solved in a reasonable time

$$M \models \mathtt{Reqs} \land \mathtt{BGP}_{\mathtt{protocol}} \land \mathtt{Policies}$$

challenges

BGP x OSPF

huge search space

Naive encodings lead to complex constraints
that cannot be solved in a reasonable time

$$M \models \mathrm{Reqs} \wedge \mathrm{BGP}_{\mathrm{protocol}} \wedge \mathrm{Policies}$$

challenges	BGP x OSPF	huge search space
solutions	iterative synthesis	partial evaluation

Naive encodings lead to complex constraints that cannot be solved in a reasonable time

$$M \models \text{Reqs} \land \text{BGP}_{\text{protocol}} \land \text{Policies}$$

challenges	BGP x OSPF	huge search space
solutions	iterative synthesis	partial evaluation

NetComplete encodes reduced policies by relying on the requirements and the sketches

NetComplete encodes reduced policies by relying on the requirements and the sketches

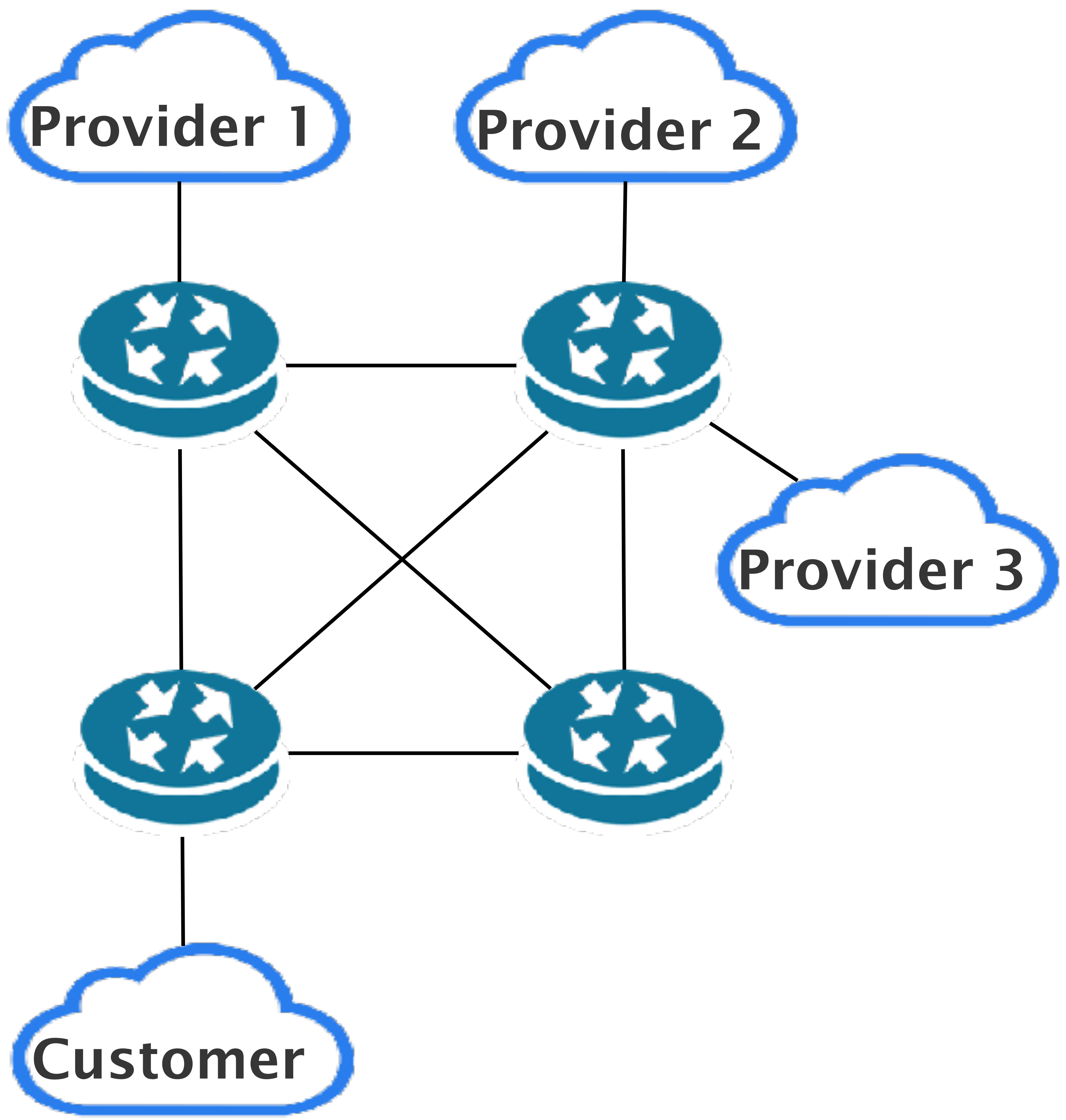
Step 1	Capture how announcements should propagate using the requirements
Output	BGP propagation graph

NetComplete encodes reduced policies by relying on the requirements and the sketches

Step 1	Capture how announcements should propagate using the requirements
Output	BGP propagation graph
Step 2	Combine the graph with constraints imposed by sketches via symbolic execution
Output	partially evaluated formulas

NetComplete relies on the requirements to figure out where BGP announcements should (not) propagate

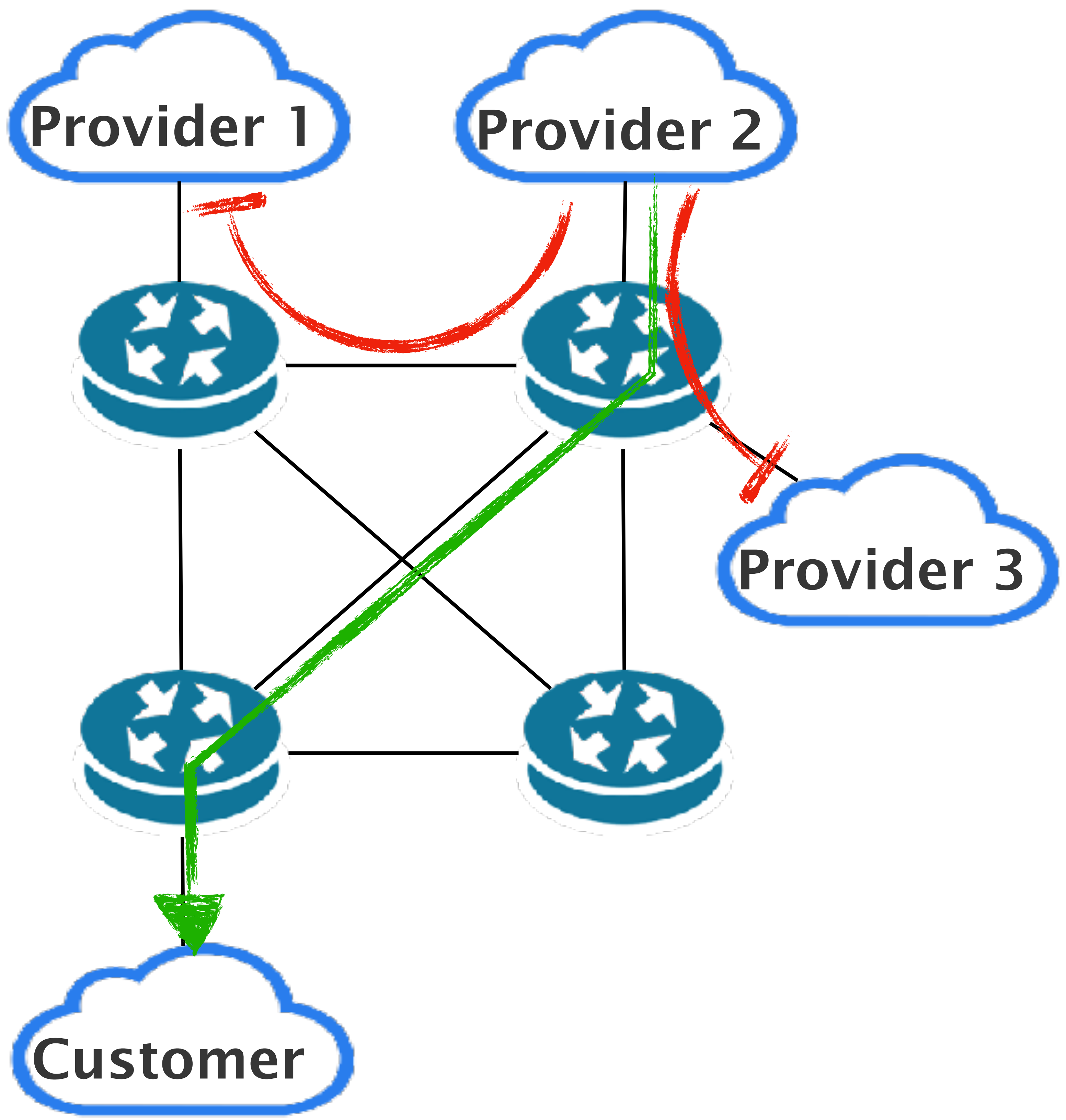
NetComplete relies on the requirements to figure out where BGP announcements should (not) propagate



Requirement

Only customers should be able to send traffic to Provider #2

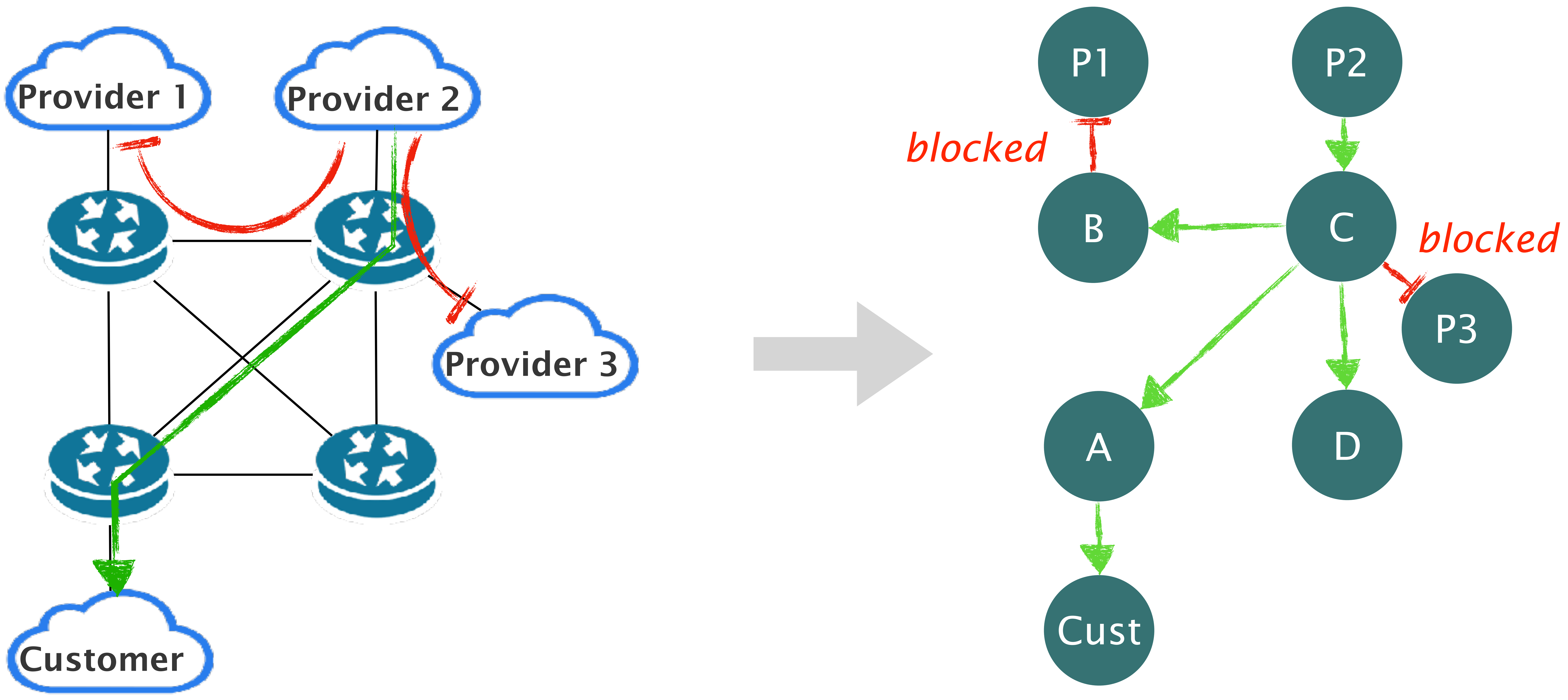
NetComplete relies on the requirements to figure out where BGP announcements should (not) propagate



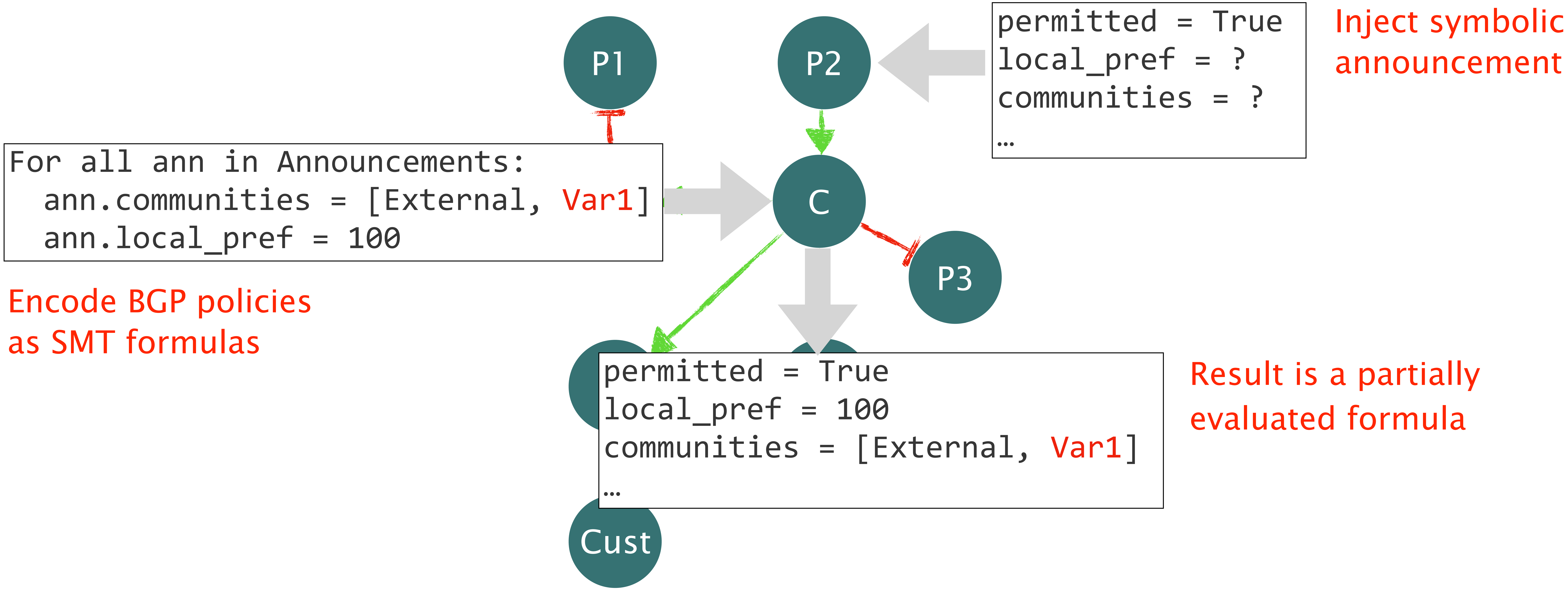
Requirement

Only customers should be able to send traffic to Provider #2

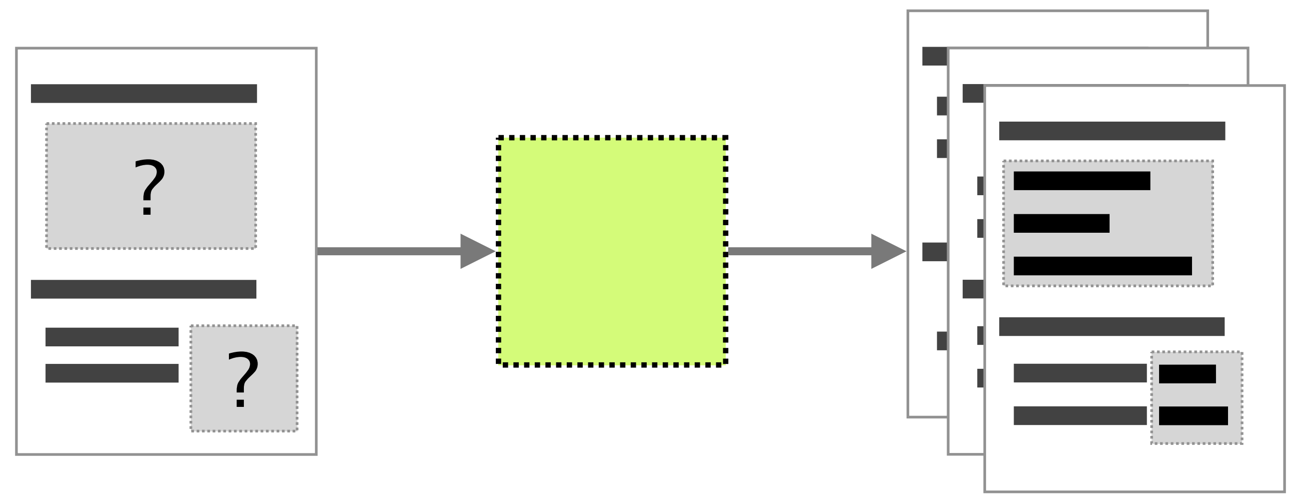
NetComplete computes one BGP propagation graph per equivalence class



NetComplete concretizes symbolic announcements by propagating them through the graph and sketches



NetComplete: Practical Network-Wide Configuration Synthesis with Autocompletion



BGP synthesis

optimized encoding

2 OSPF synthesis

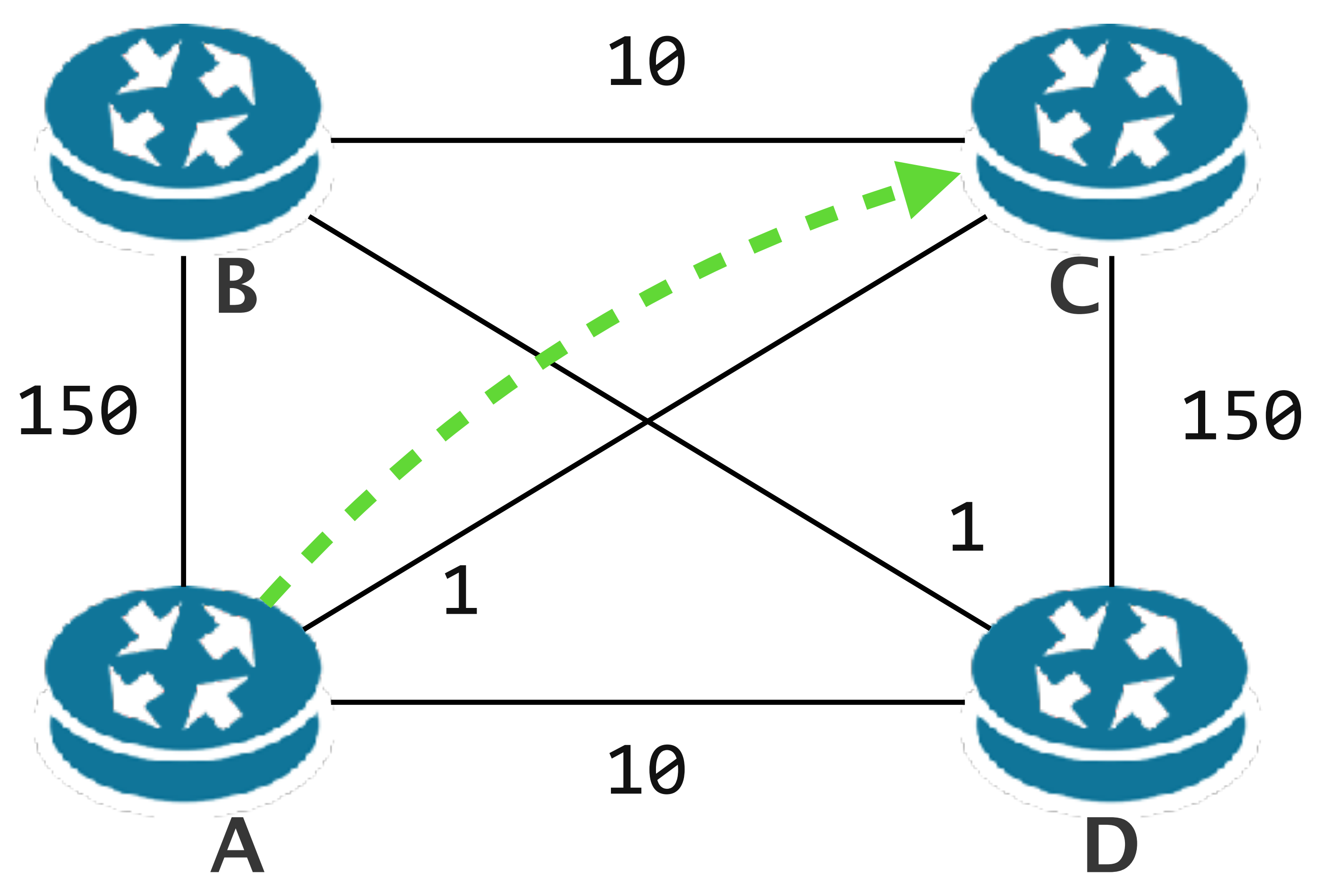
counter-examples-based

Evaluation

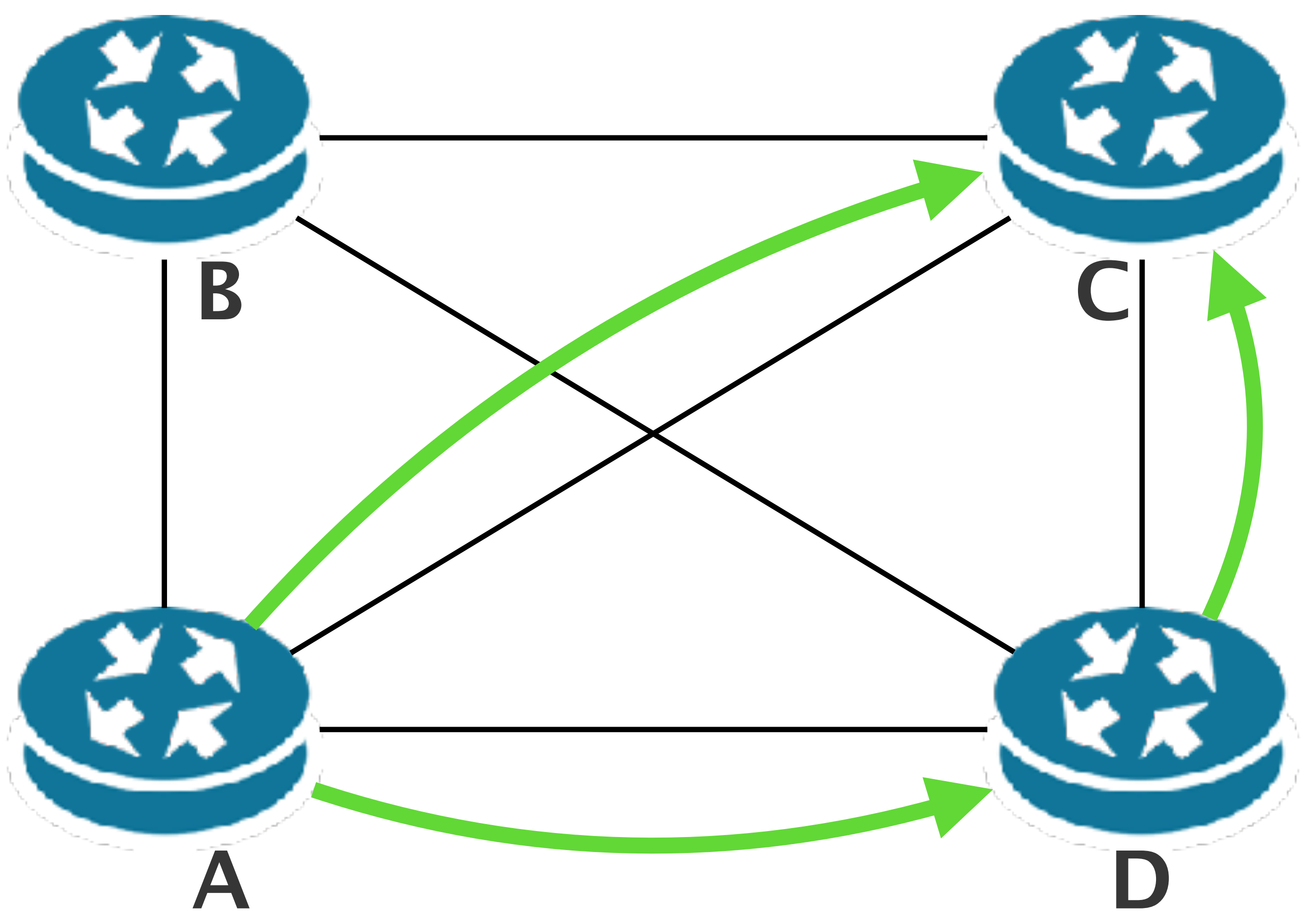
flexible, *yet* scalable

As for BGP, Netcomplete phrases the problem of finding weights as a constraint satisfaction problem

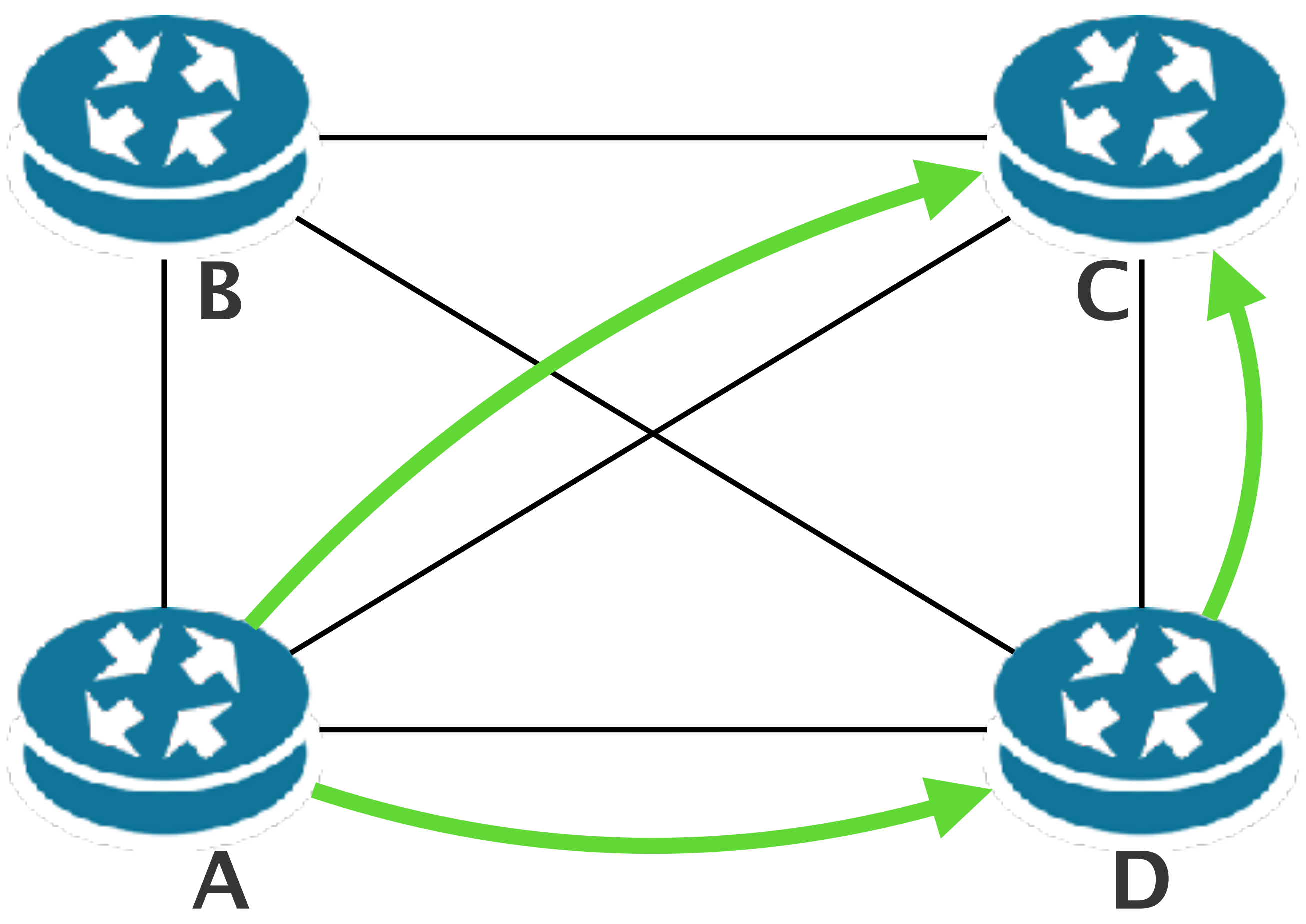
Consider this initial configuration in which the (A,C) traffic is forwarded along the direct link



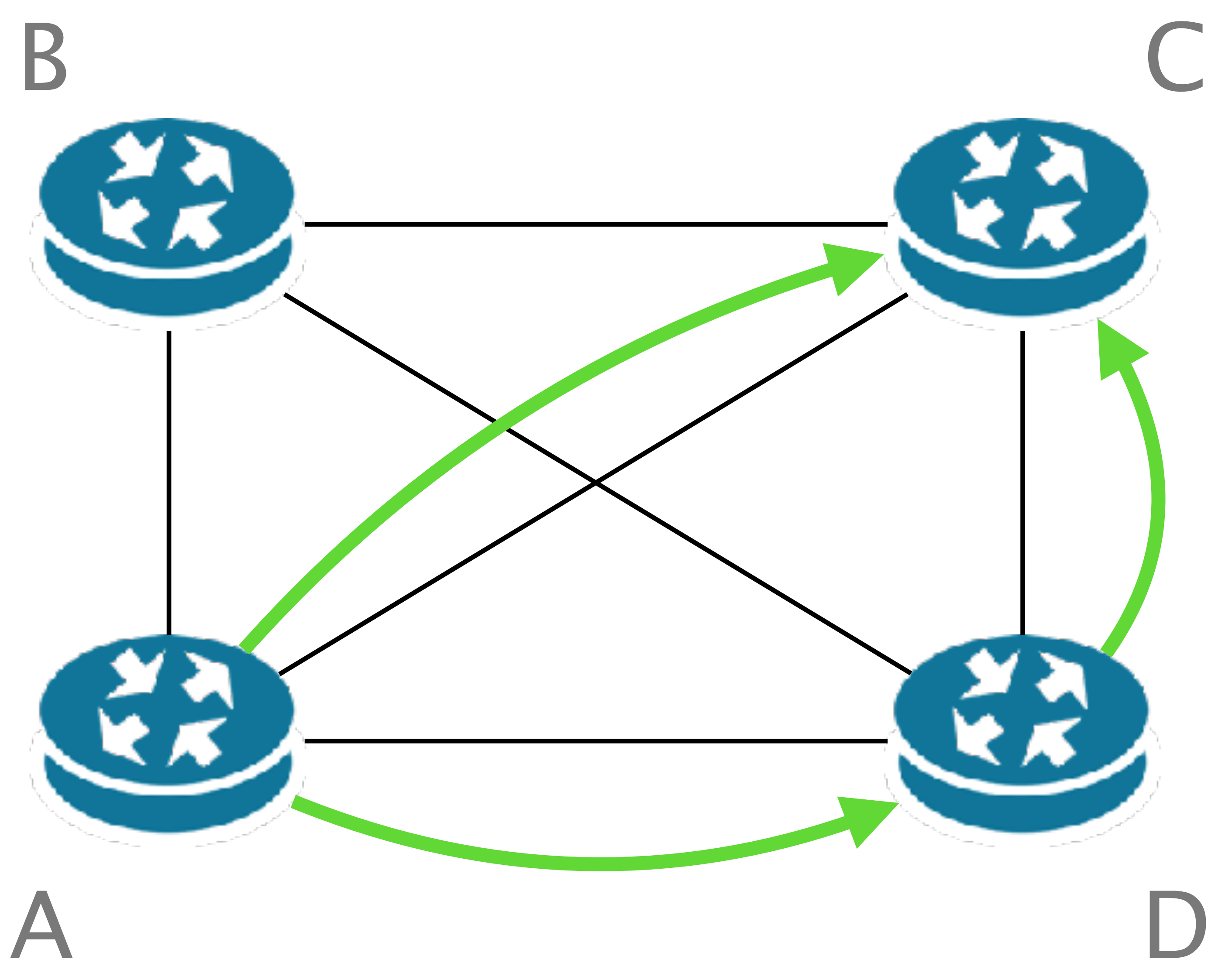
For performance reasons,
the operators want to enable load-balancing



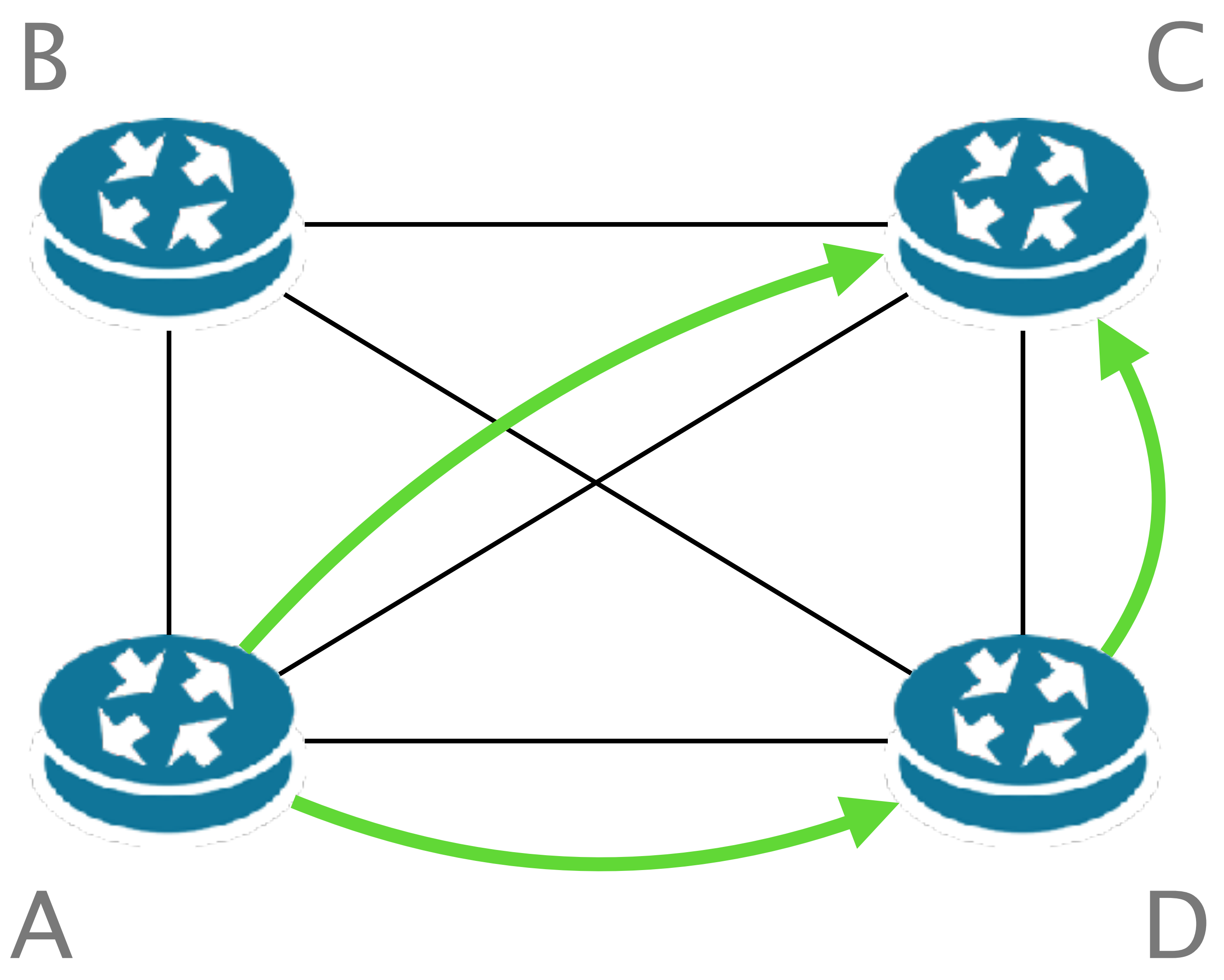
What should be the weights for this to happen?



input requirements

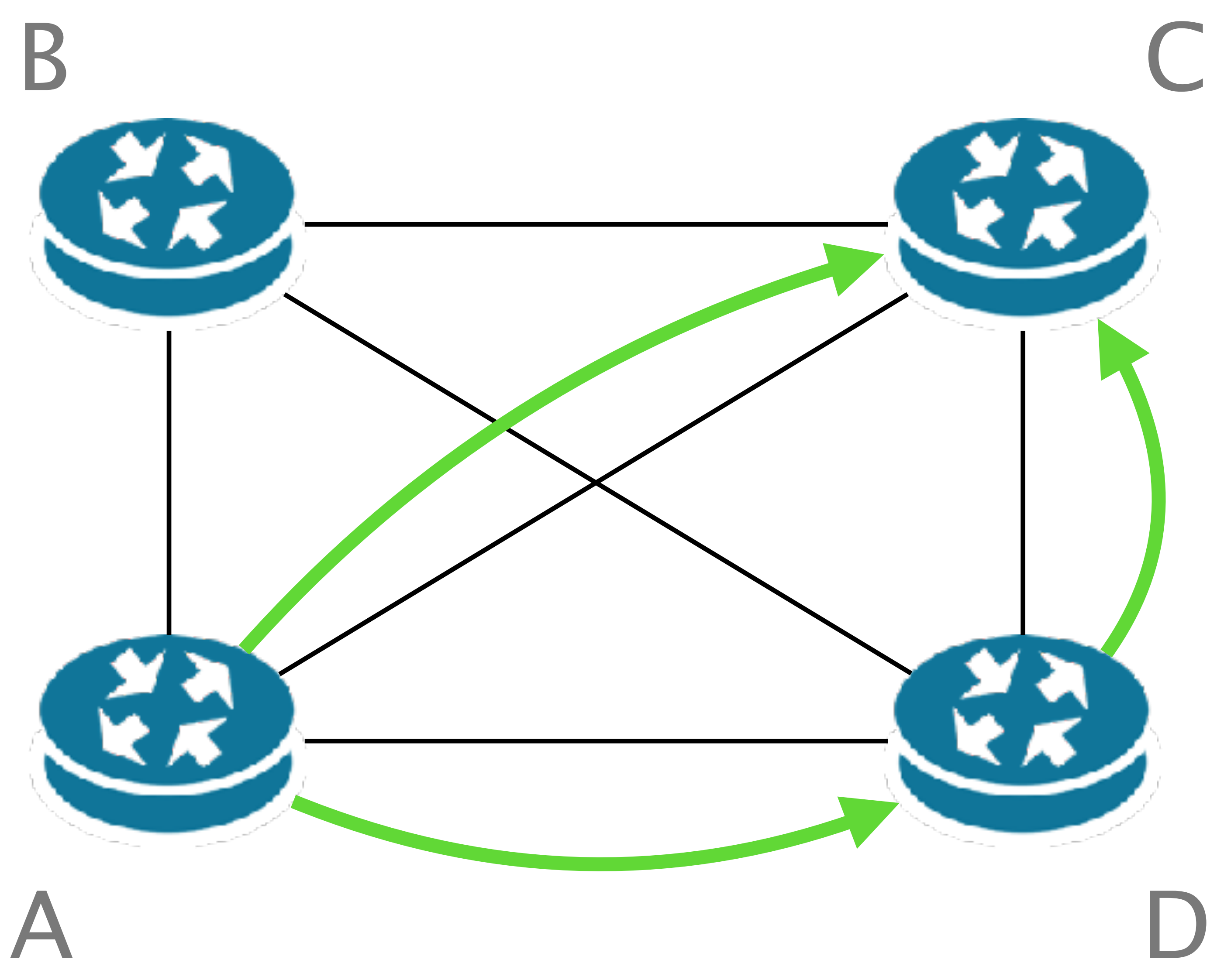


input requirements



synthesis procedure

input requirements



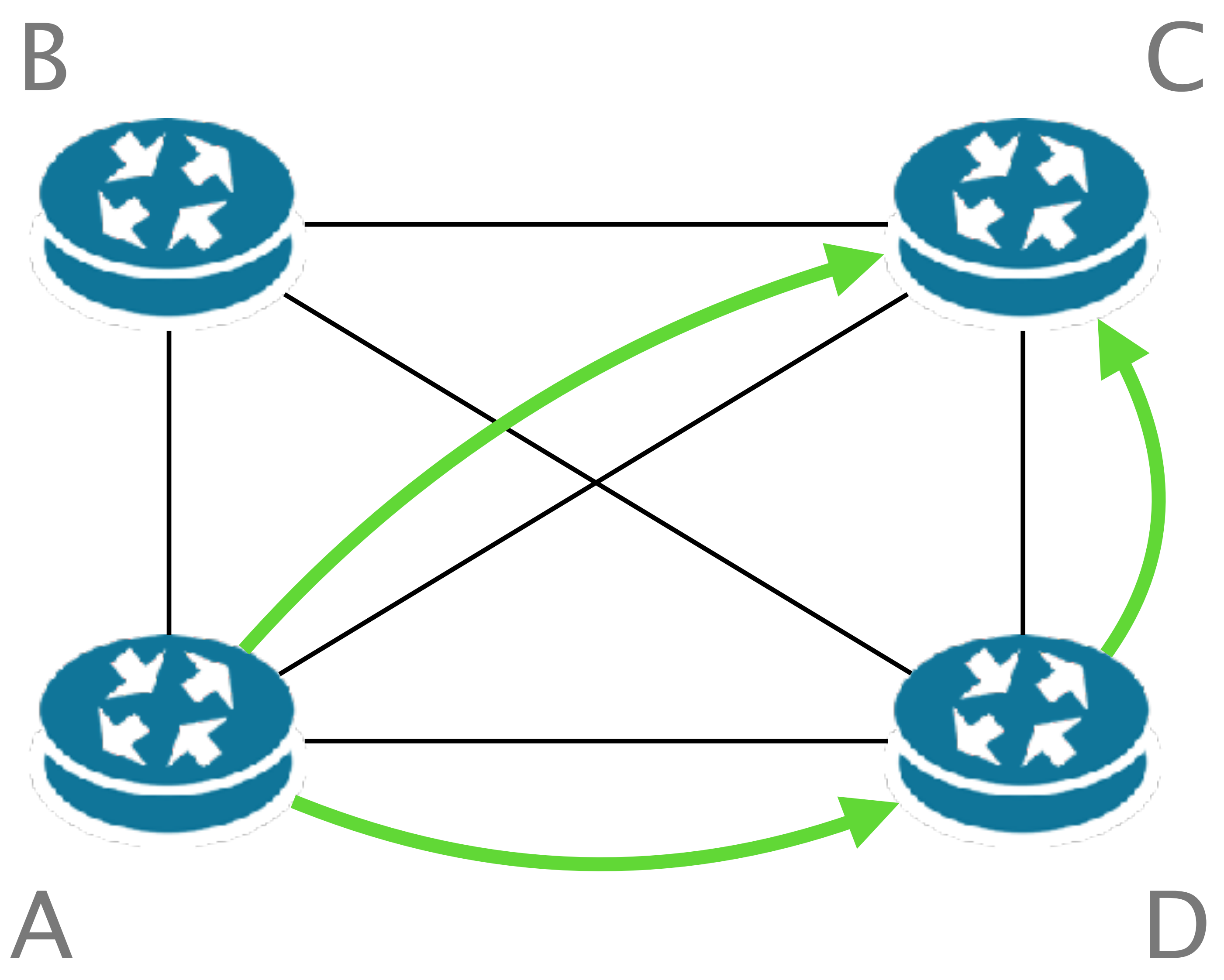
synthesis procedure

$$\forall X \in \mathrm{Paths}(A,C) \setminus \mathrm{Reqs}$$

↓

$$\mathrm{Cost}(A \rightarrow C) = \mathrm{Cost}(A \rightarrow D \rightarrow C) < \mathrm{Cost}(X)$$

input requirements



synthesis procedure

$\forall X \in Paths(A,C) \setminus Reqs$

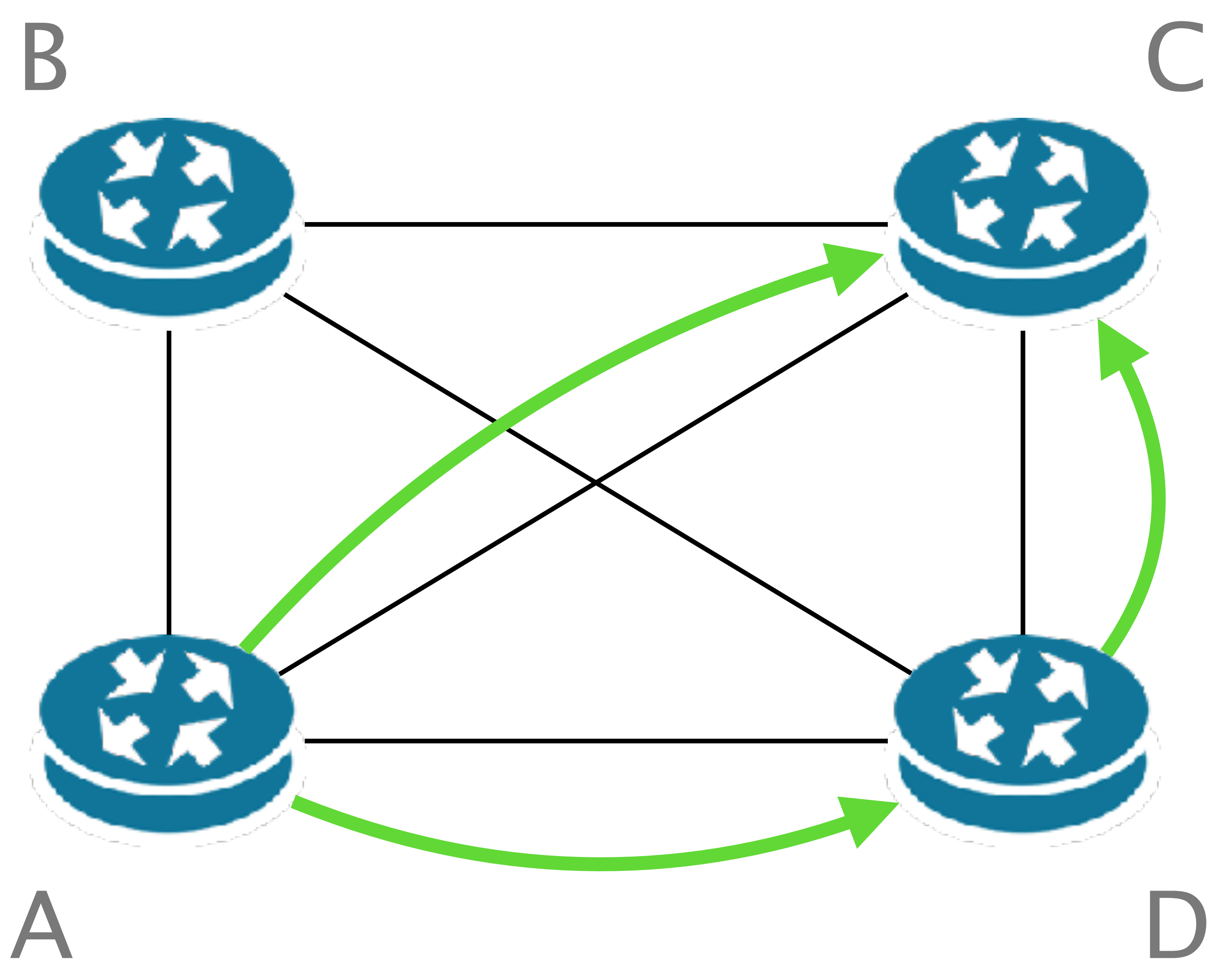
↓

$Cost(A \rightarrow C) = Cost(A \rightarrow D \rightarrow C) < Cost(X)$

↓

Solve

input requirements



synthesis procedure

$\forall X \in \text{Paths}(A,C) \setminus \text{Reqs}$

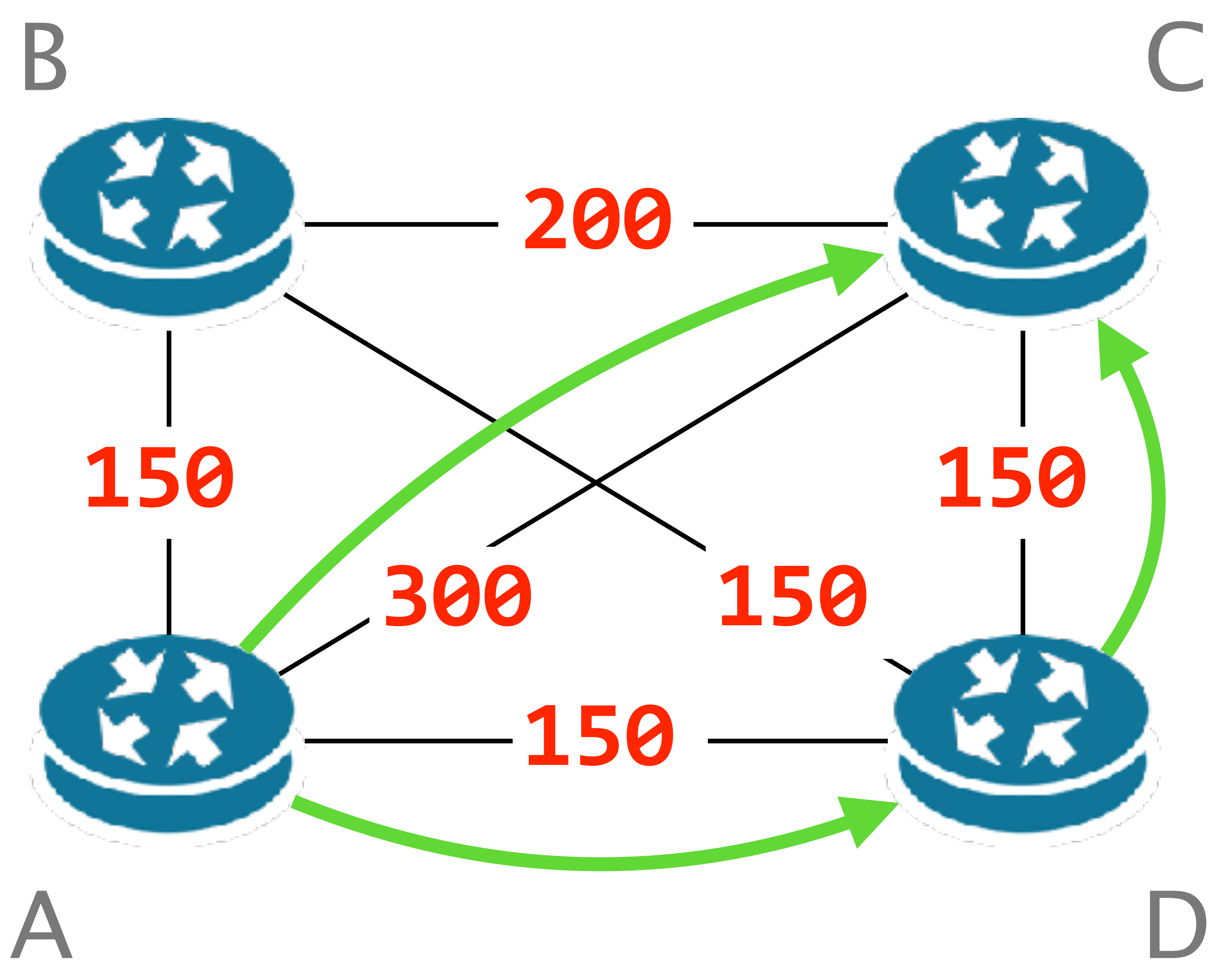
↓

$\text{Cost}(A \rightarrow C) = \text{Cost}(A \rightarrow D \rightarrow C) < \text{Cost}(X)$

↓

Solve

input requirements



Synthesized weights

synthesis procedure

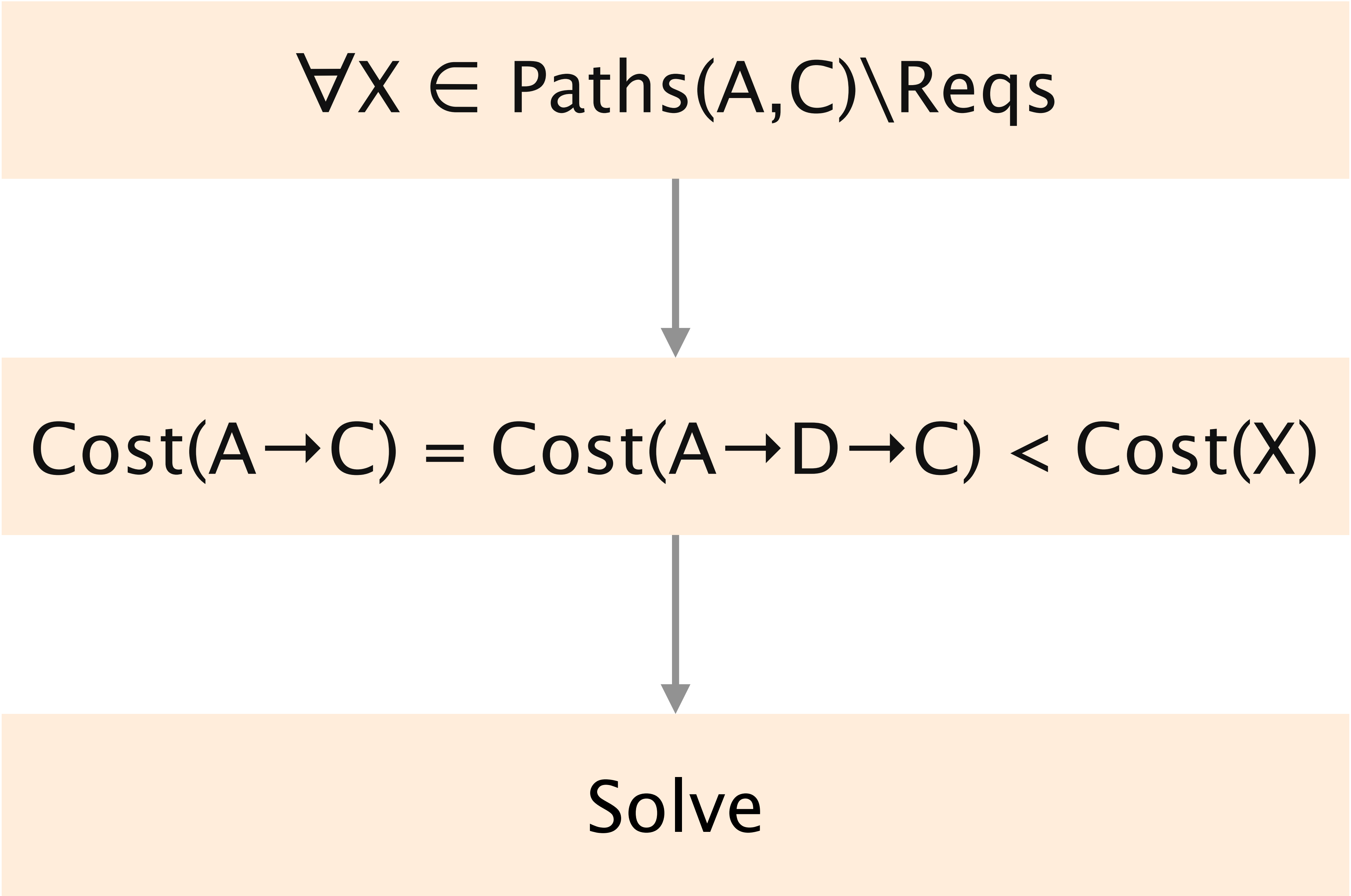
$\forall X \in \mathrm{Paths}(A,C) \setminus \mathrm{Reqs}$

$\mathrm{Cost}(A \rightarrow C) = \mathrm{Cost}(A \rightarrow D \rightarrow C) < \mathrm{Cost}(X)$

Solve

This was easy, but…

it does **not** scale



There can be an exponential number of paths between A and C...

$\forall X \in Paths(A,C) \setminus Reqs$

$Cost(A \to C) = Cost(A \to D \to C) < Cost(X)$

Solve

To scale, NetComplete leverages

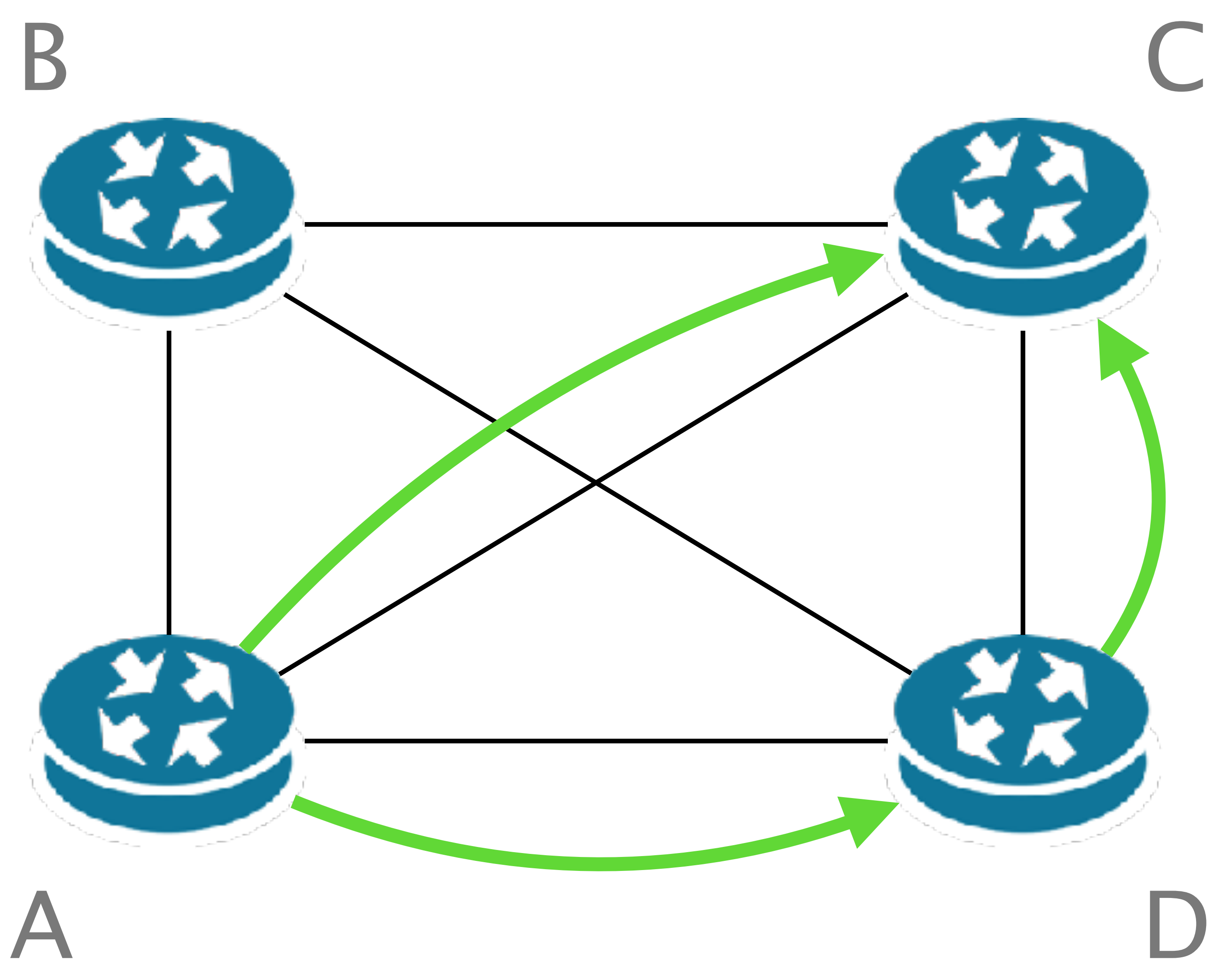
Counter-Example Guided Inductive Synthesis (CEGIS)

An contemporary approach to synthesis where

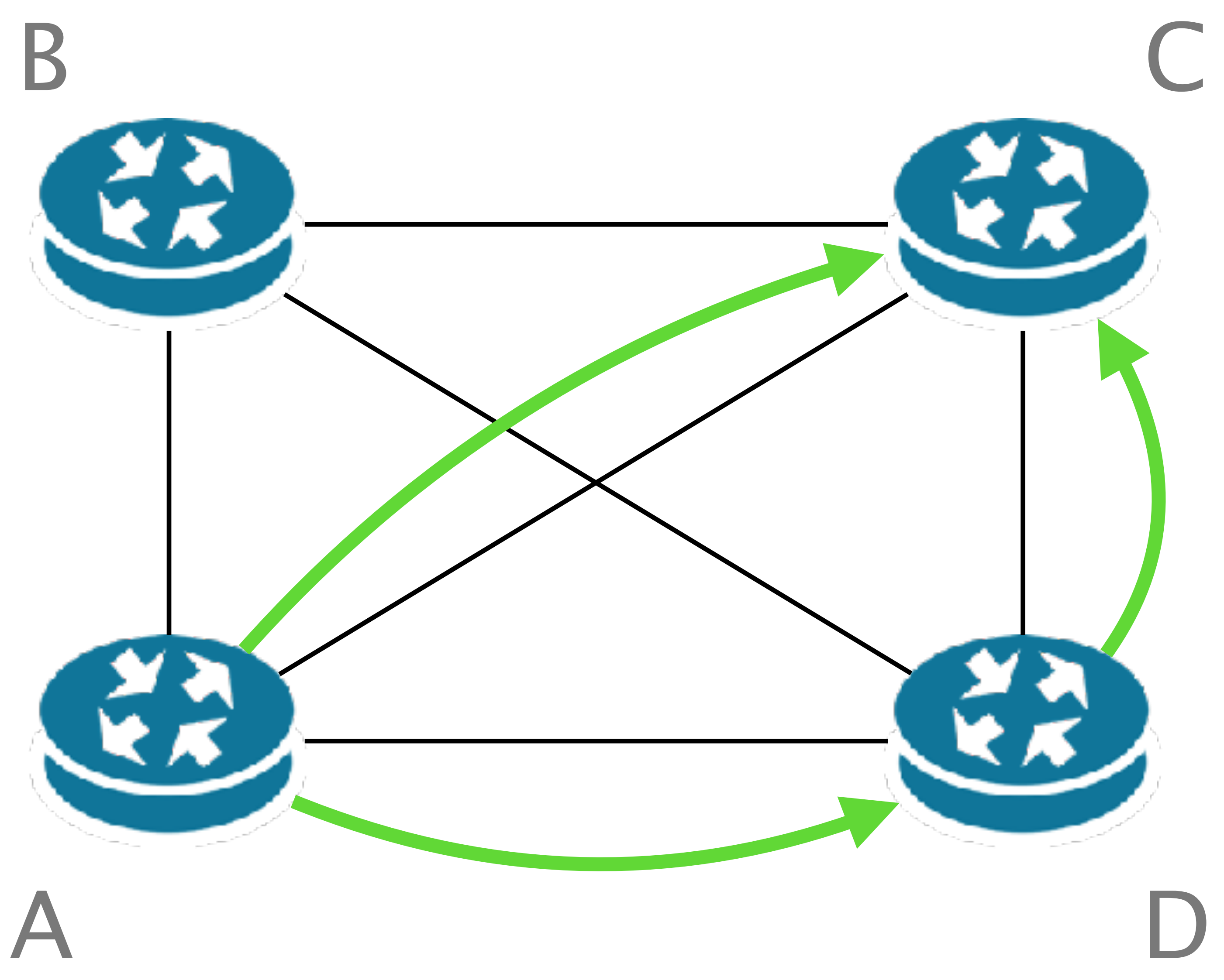
a solution is iteratively learned from counter-examples

While enumerating all paths is hard,
computing shortest paths given weights is easy!

input requirements

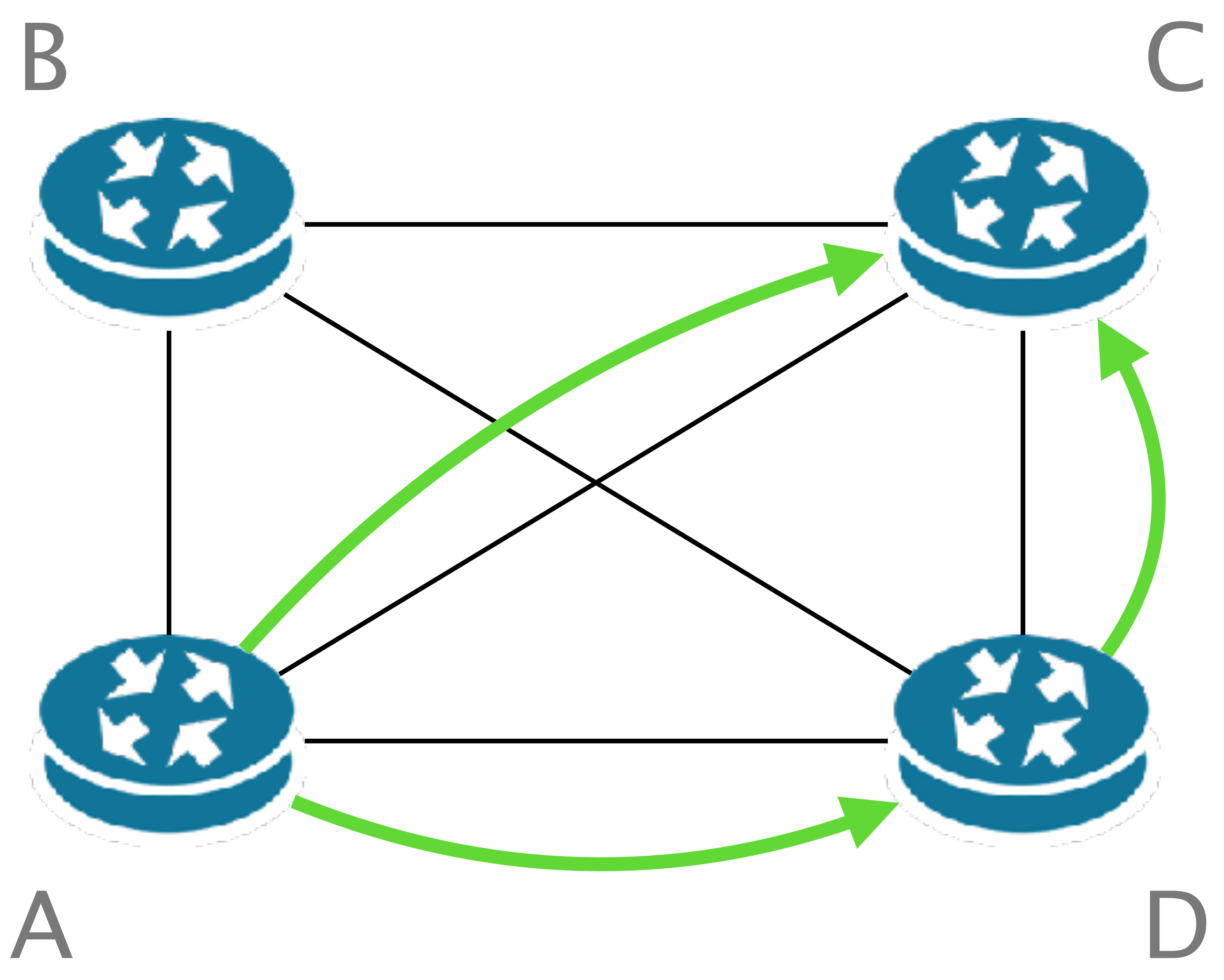


input requirements



synthesis procedure

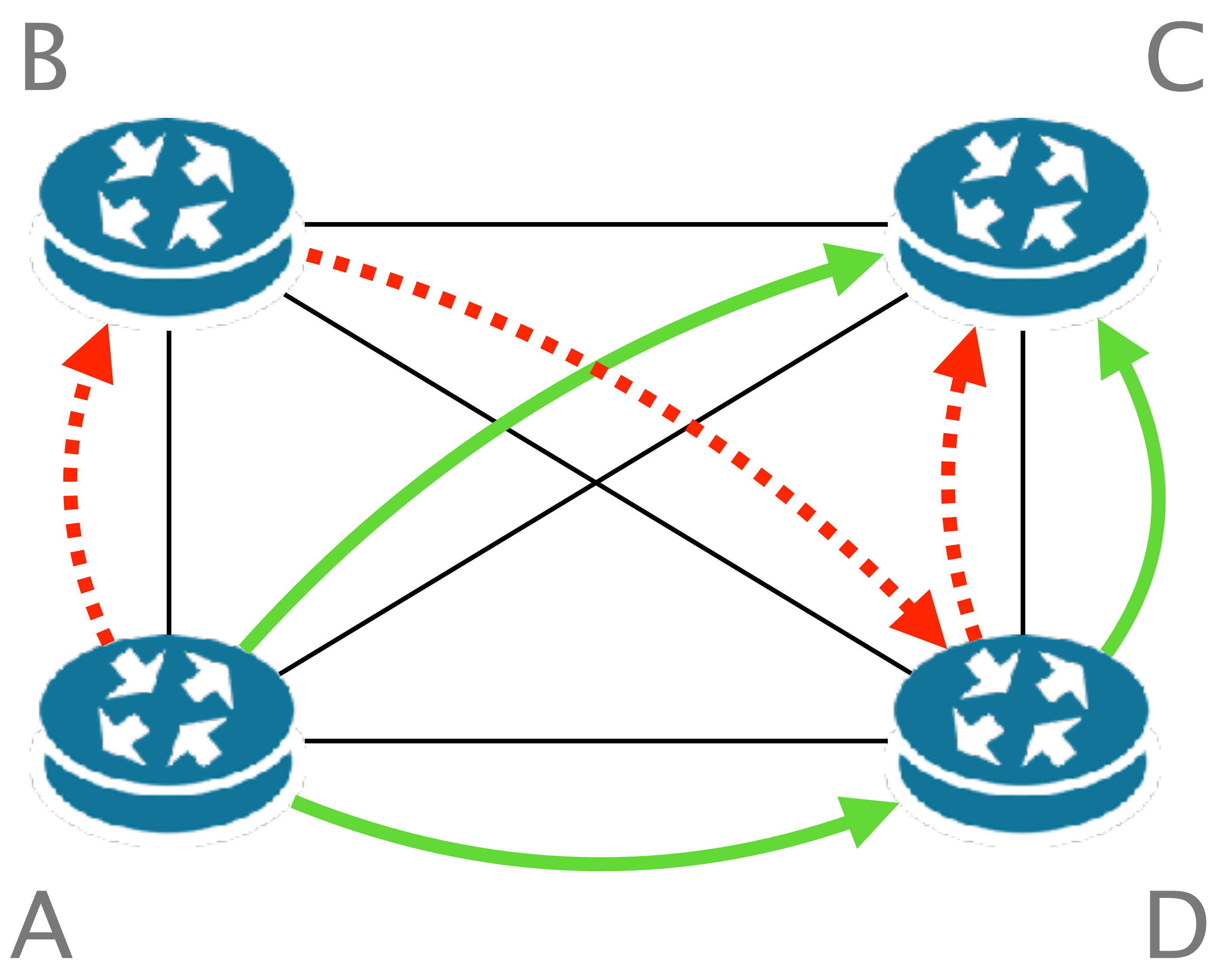
input requirements



synthesis procedure

$\forall X \in$ SamplePaths(A,C)\Reqs

input requirements

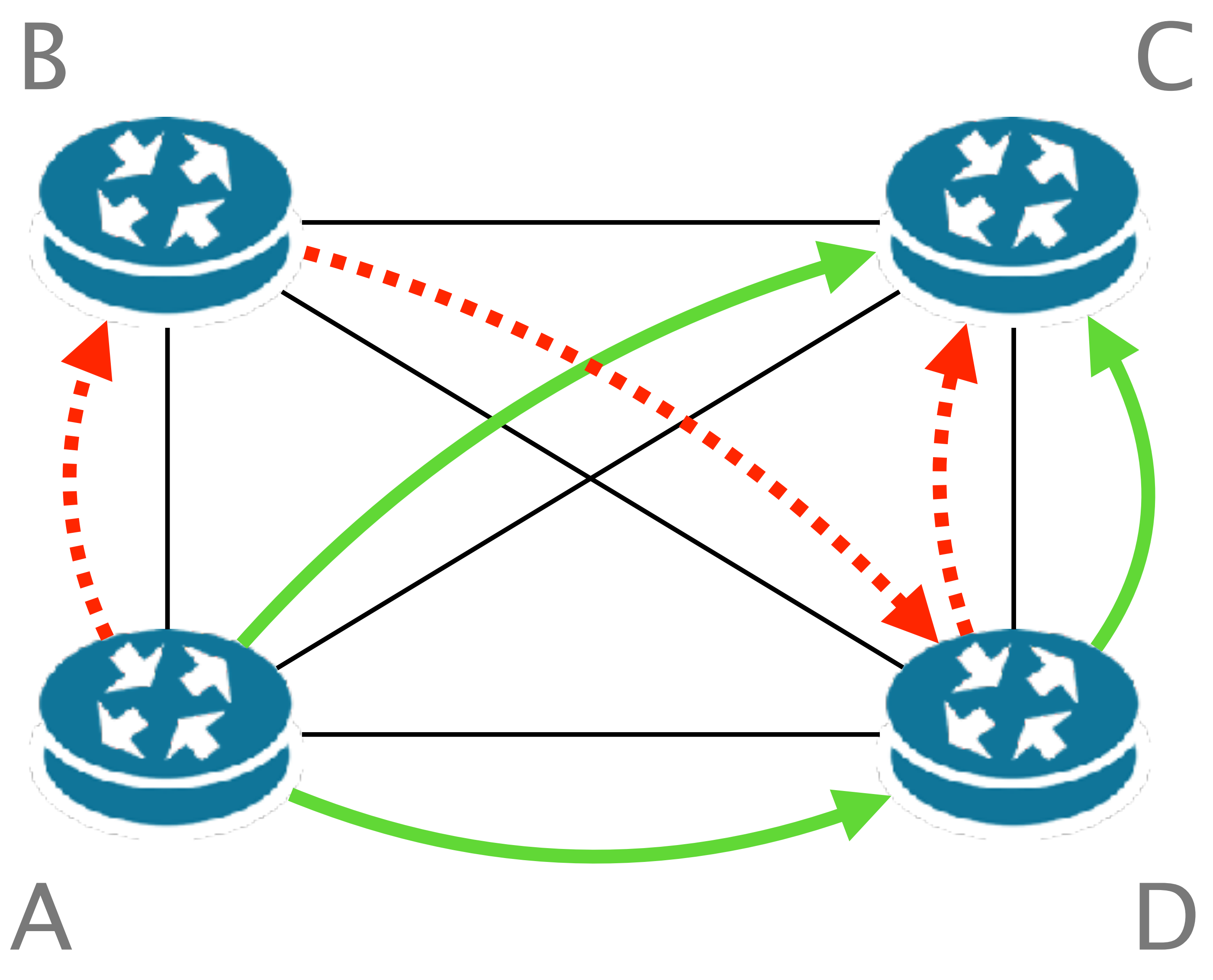


synthesis procedure

$\forall X \in$ SamplePaths(A,C)\Reqs

Sample: { [A,B,D,C] }

input requirements

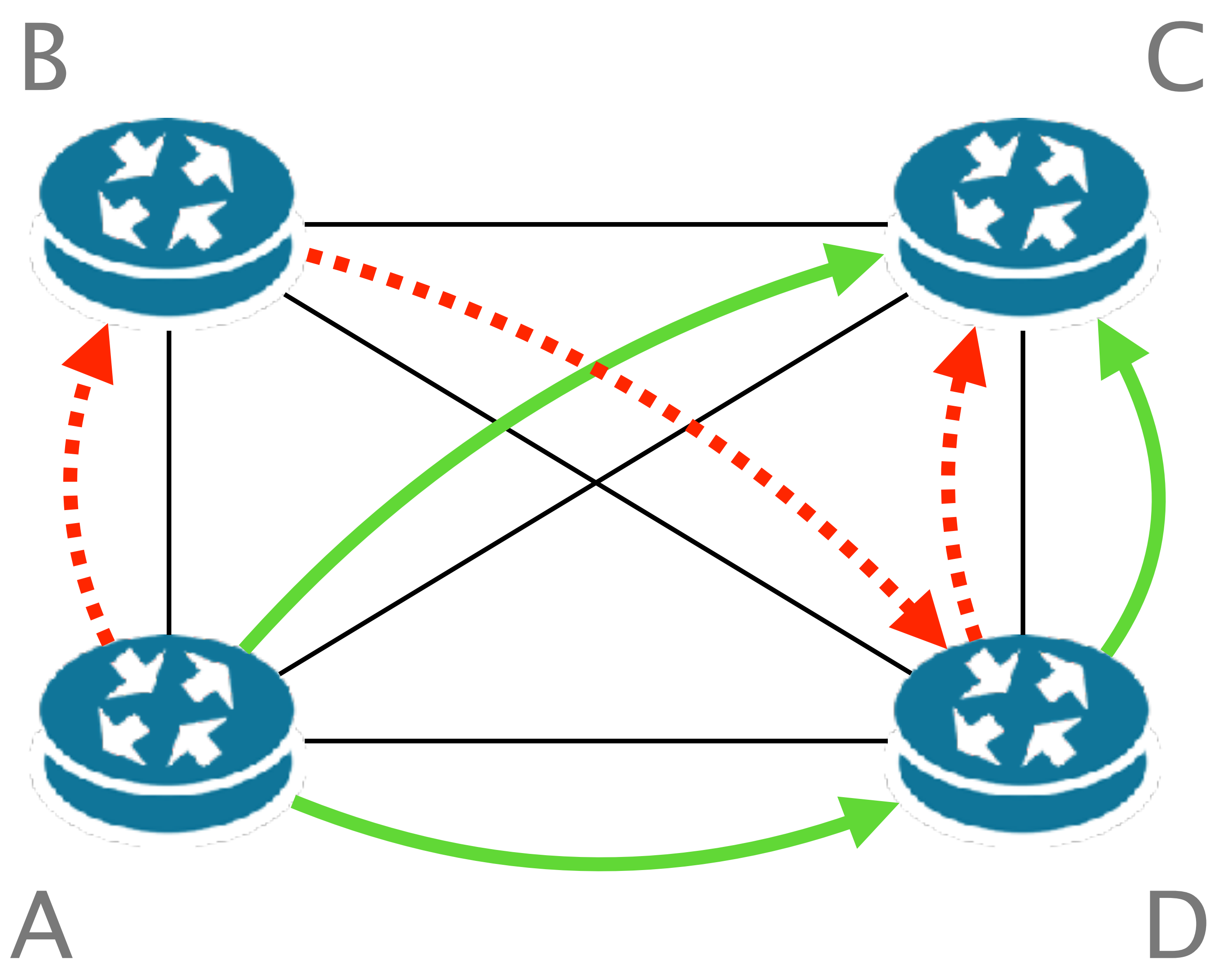


synthesis procedure

$\forall X \in \text{SamplePaths(A,C)} \setminus \text{Reqs}$

$\text{Cost}(A \rightarrow C) = \text{Cost}(A \rightarrow D \rightarrow C) < \text{Cost}(X)$

input requirements



synthesis procedure

∀X ∈ SamplePaths(A,C)\Reqs

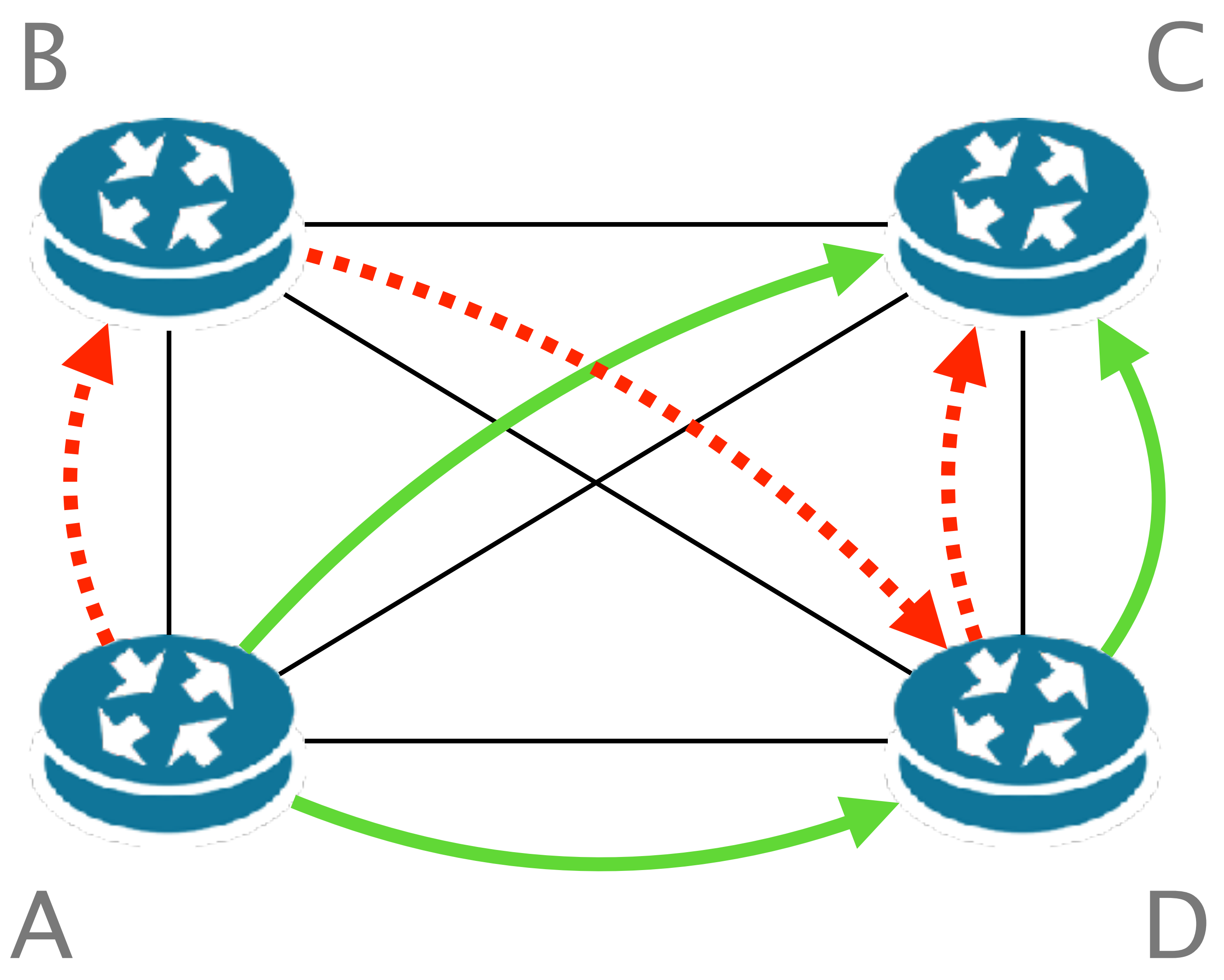
↓

Cost(A→C) = Cost(A→D→C) < Cost(X)

↓

Solve

input requirements



synthesis procedure

∀X ∈ SamplePaths(A,C)\Reqs

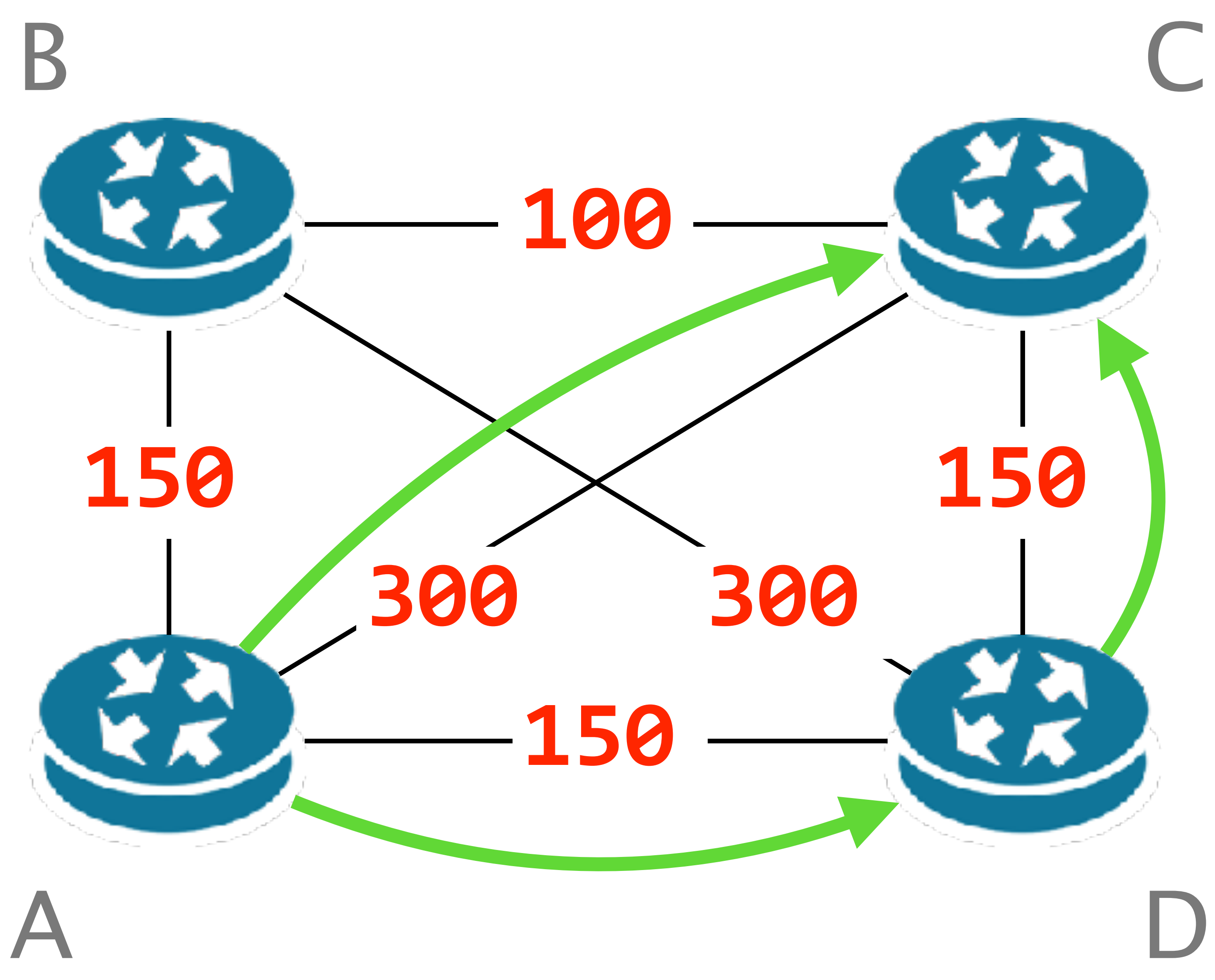
↓

Cost(A→C) = Cost(A→D→C) < Cost(X)

↓

Solve

input requirements



Synthesized weights

synthesis procedure

∀X ∈ SamplePaths(A,C)\Reqs

↓

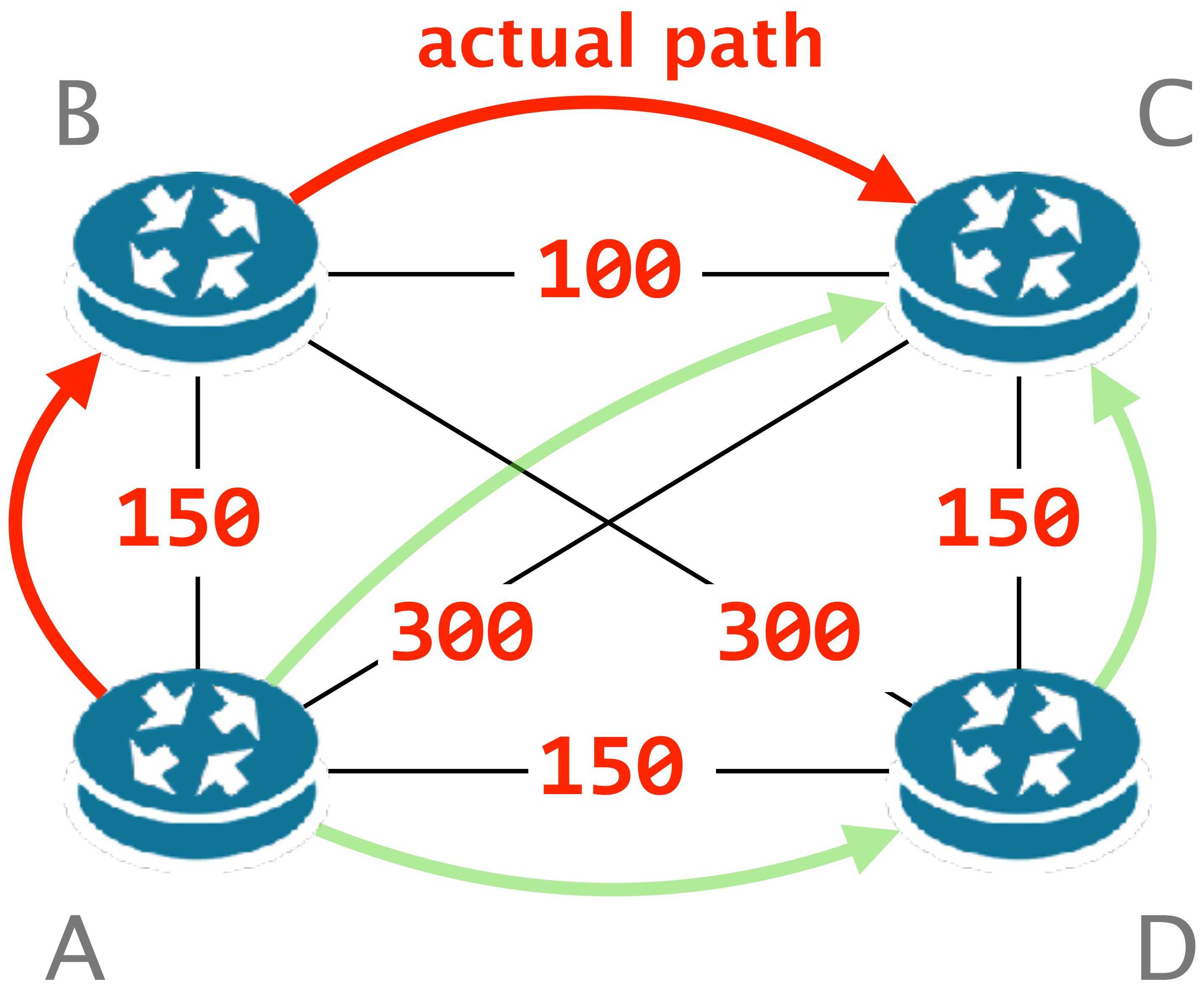
Cost(A→C) = Cost(A→D→C) < Cost(X)

↓

Solve

The synthesized weights are incorrect:

cost(A → B → C]) = 250 < cost(A → C) = 300



∀X ∈ SamplePaths(A,C)\Reqs

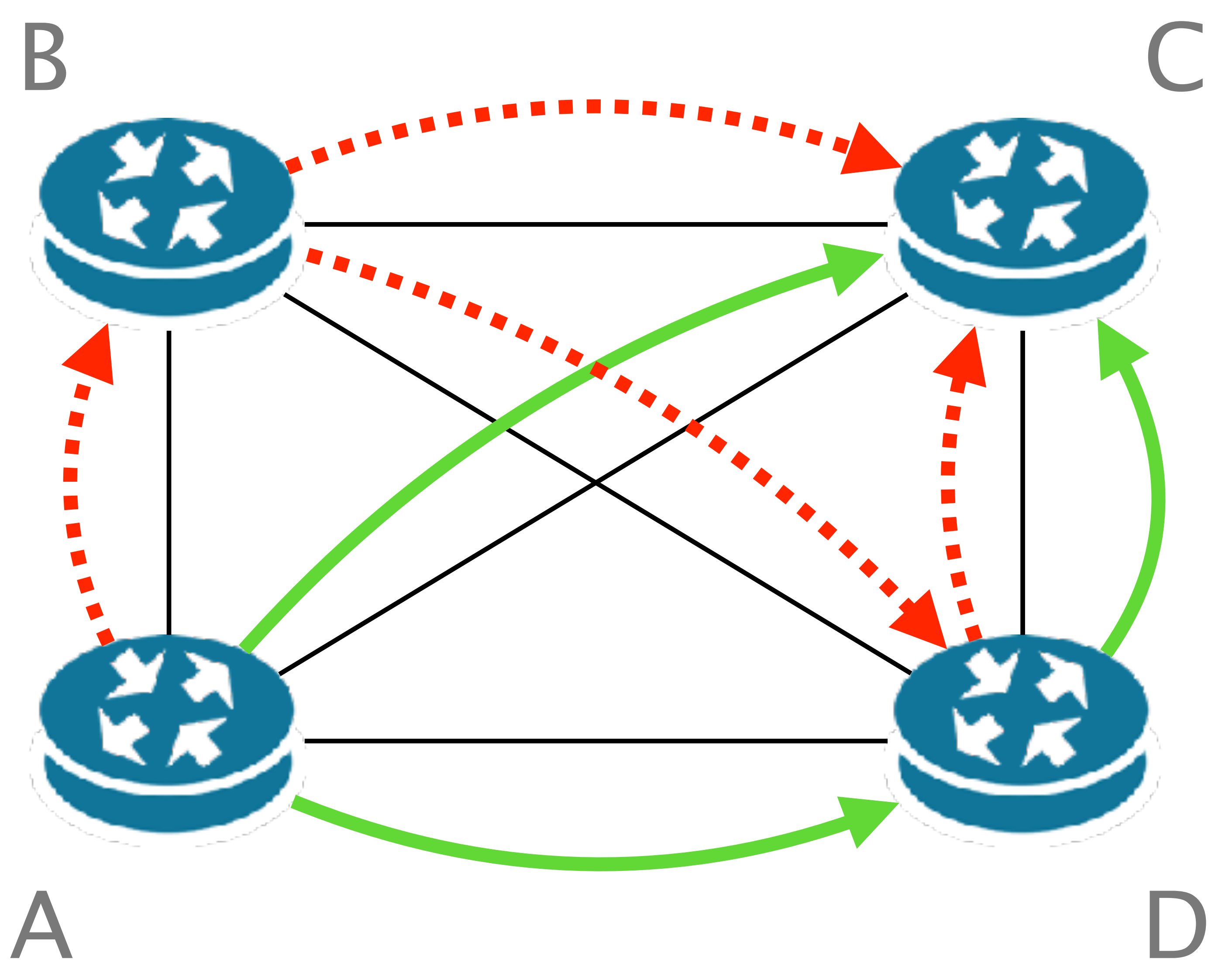
↓

Cost(A→C) = Cost(A→D→C) < Cost(X)

↓

Solve

We simply add the counter example to SamplePaths and repeat the procedure



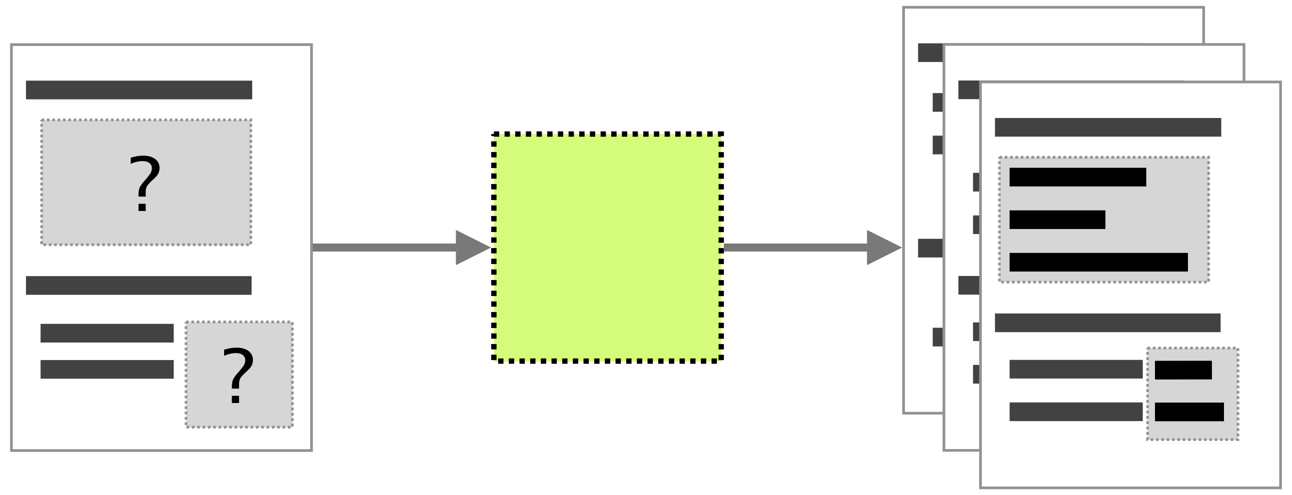
$\forall X \in \text{SamplePaths}(A,C) \setminus \text{Reqs}$

Sample: { [A,B,D,C] } U { [A,B,C] }

The entire procedure usually converges in few iterations

making it very fast in practice

NetComplete: Practical Network-Wide Configuration Synthesis with Autocompletion



BGP synthesis

optimized encoding

OSPF synthesis

counter-examples-based

3 Evaluation

flexible, *yet* scalable

Question #1

Can NetComplete synthesize large-scale configurations?

Question #2

How does the concreteness of the sketch influence the running time?

We fully implemented NetComplete and showed its practicality

Code	~10K lines of Python SMT-LIB v2 and Z3
Input	OSPF, BGP, static routes as partial and concrete configs
Output	Cisco-compatible configurations validated with actual Cisco routers

Methodology

Topology	15 topologies from Topology Zoo small, medium, and large
Requirement	Simple, Any, ECMP, and ordered (random) using OSPF/BGP
Sketch	Built from a fully concrete configuration from which we made a % of the variables symbolic

NetComplete synthesizes configurations
for large networks in few minutes

NetComplete synthesizes configurations for large networks in few minutes

	Network size	Reqs. type	Synthesis time
OSPF synthesis time (sec)	Large ~150 nodes	Simple	14s
		ECMP	13s
		Ordered	249s

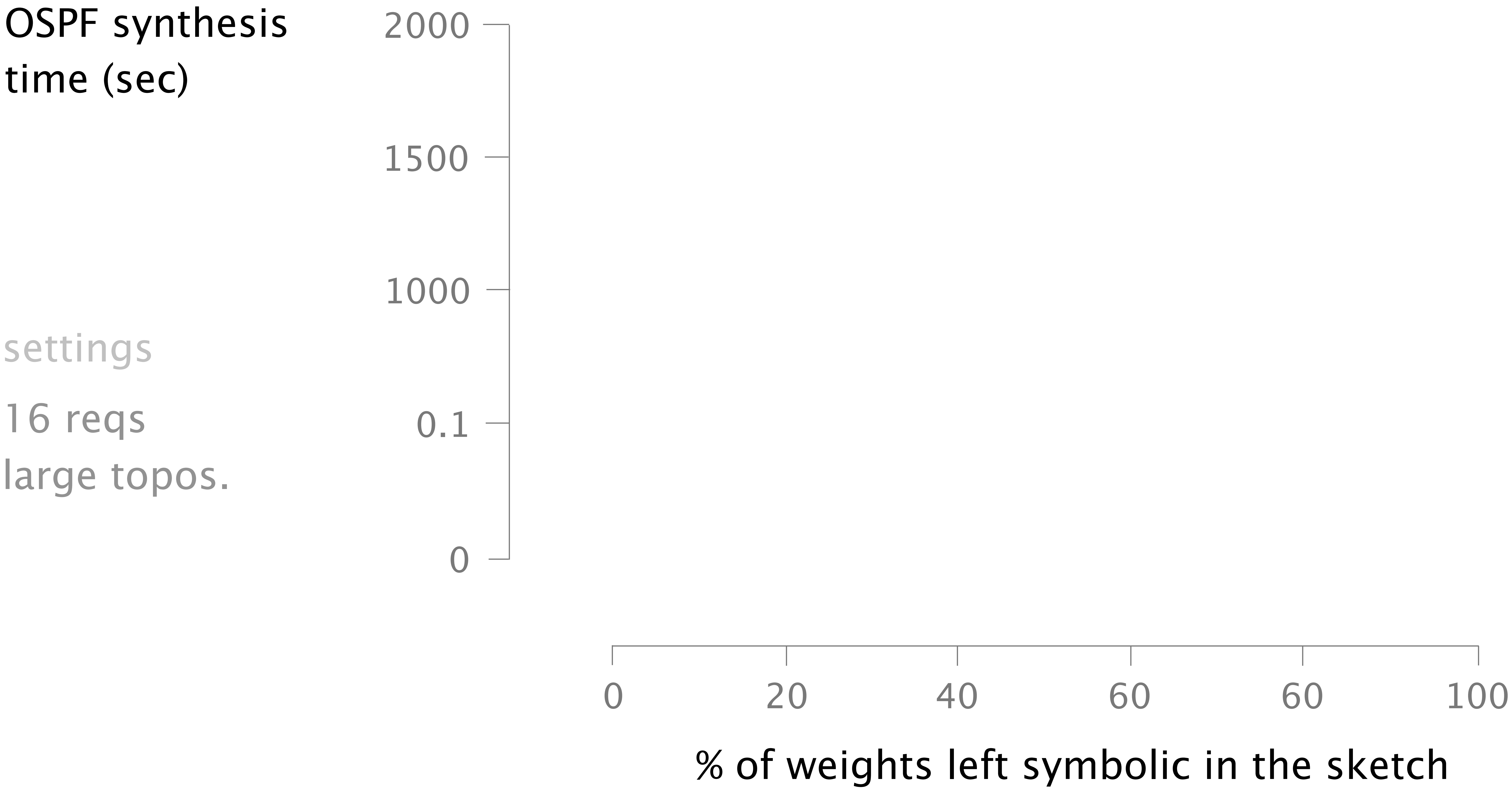
settings

16 reqs, 50% symbolic, 5 repet.

CEGIS enabled

Without CEGIS, OSPF synthesis is
>100x slower and often timeouts

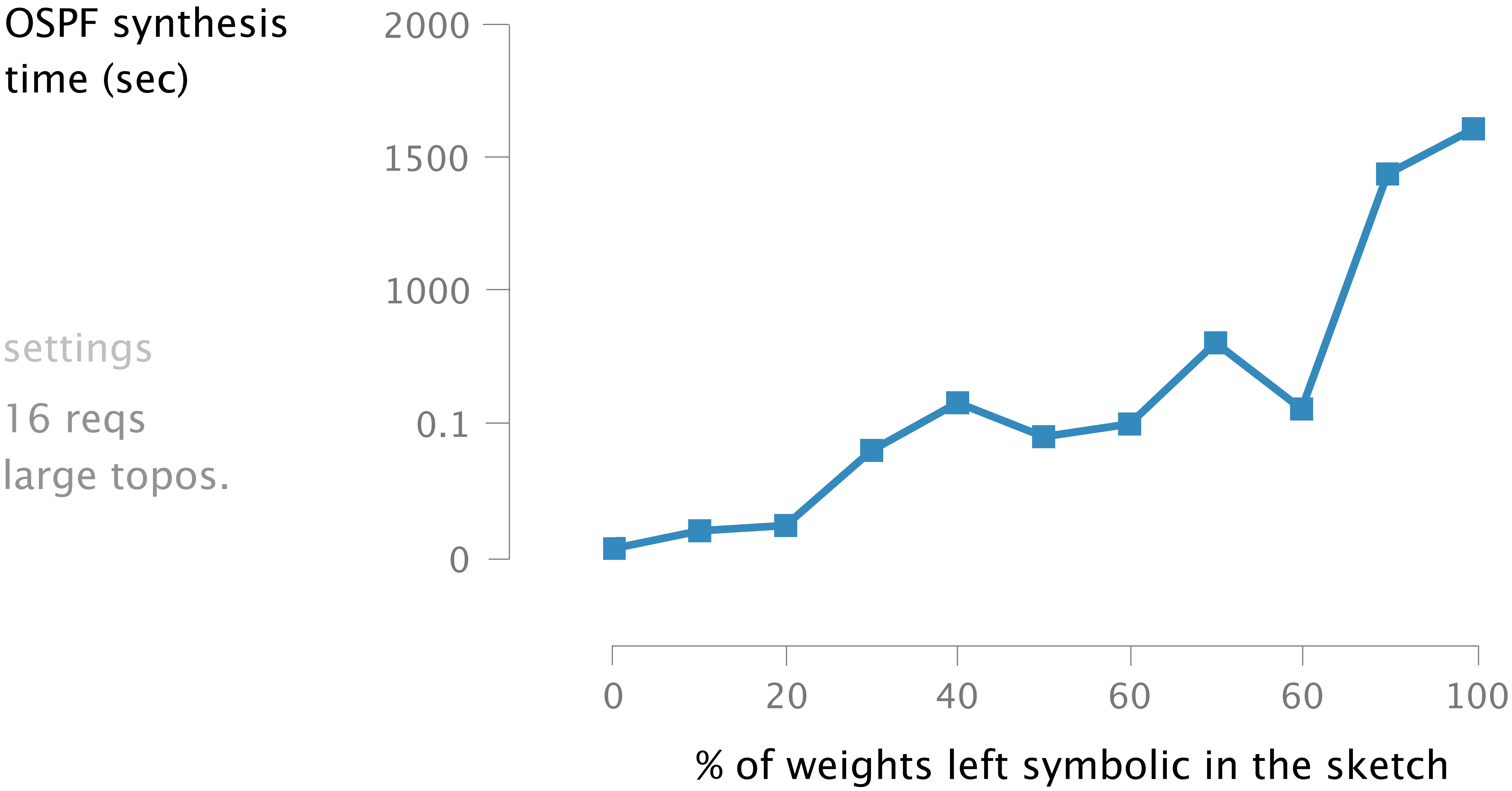
NetComplete synthesis time increases as the sketch becomes more symbolic



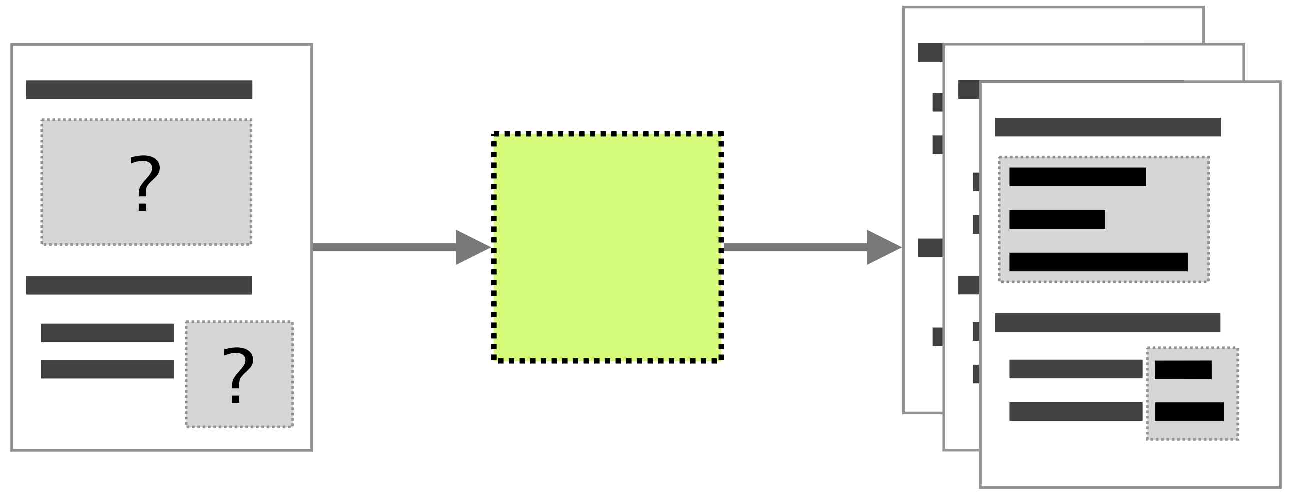
settings

16 reqs
large topos.

NetComplete synthesis time increases as the sketch becomes more symbolic



NetComplete: Practical Network-Wide Configuration Synthesis with Autocompletion



BGP synthesis

optimized encoding

OSPF synthesis

counter-examples-based

Evaluation

flexible, *yet* scalable

NetComplete: Practical Network-Wide Configuration Synthesis with Autocompletion

Autocompletes configurations with “holes”

leaving the concrete parts intact

Phrases the problem as constraints satisfaction

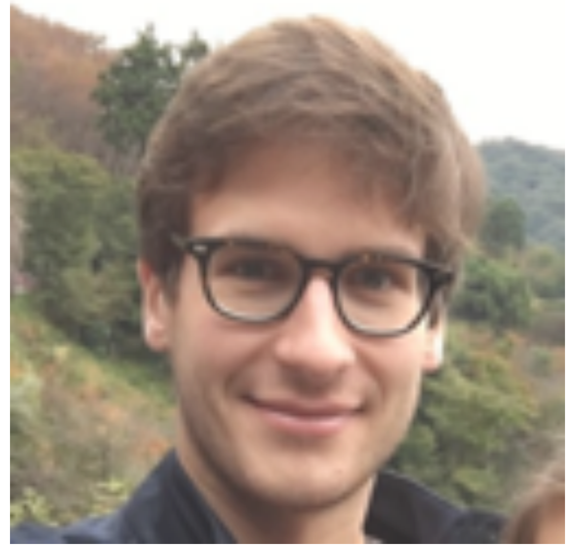
scales using network-specific heuristics & partial evaluation

Scales to realistic network size

synthesizes configurations for large network in minutes

NetComplete: Practical Network-Wide Configuration Synthesis with Autocompletion






Ahmed El-Hassany

Petar Tsankov

Laurent Vanbever

Martin Vechev

SRL SOFTWARE RELIABILITY LAB

Networked Systems ETH Zürich — seit 2015

ETH zürich