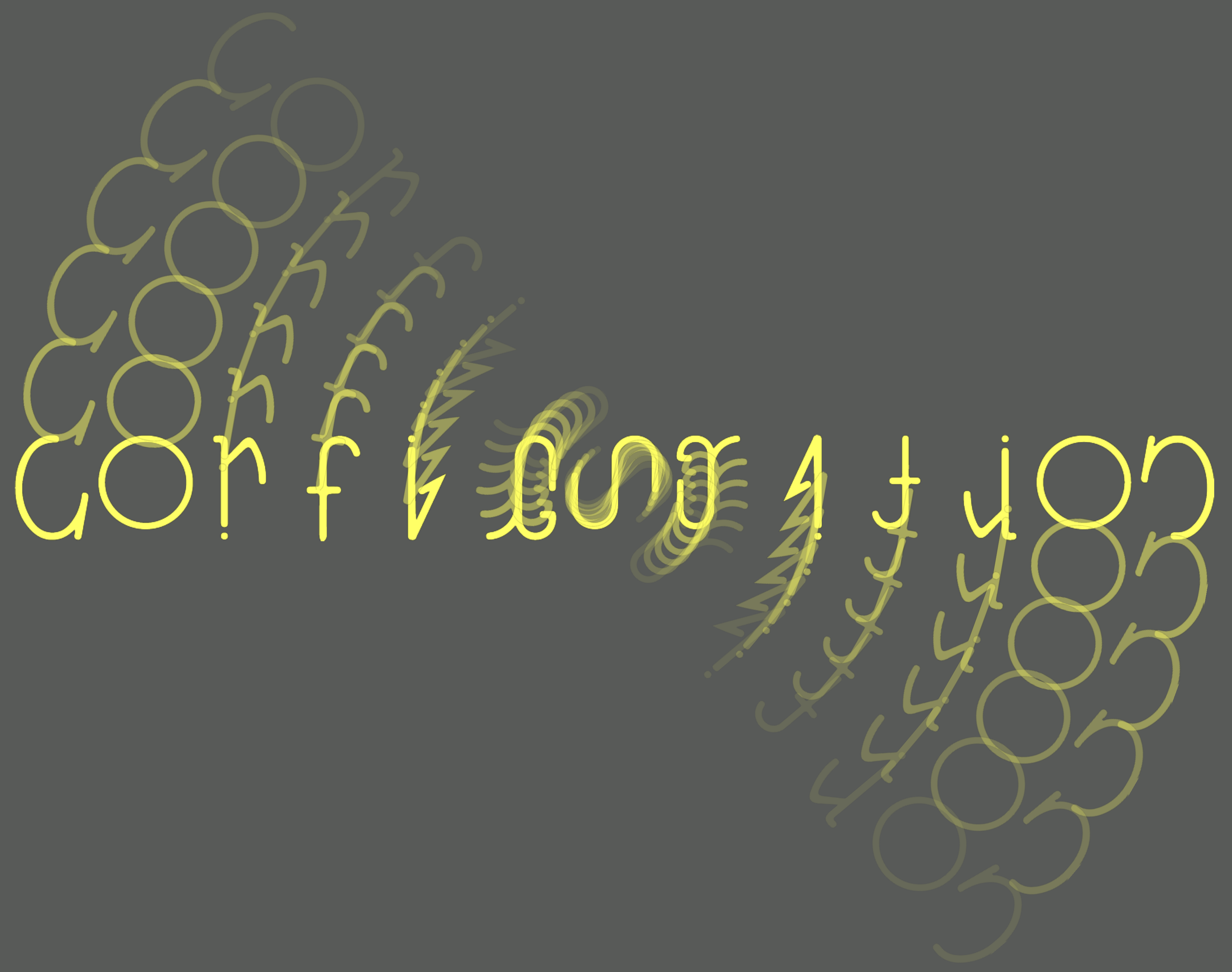
The three tales of (correct) network operations



Laurent Vanbever

nsg.ee.ethz.ch

CoNEXT

Wed Dec 8 2021

Date 29 April 2011 9:49pm

Date 29 April 2011 9:49pm

From sigcomm11-pc-chairs@acm.org

Date 29 April 2011 9:49pm

From sigcomm11-pc-chairs@acm.org

Subject Accepted paper #41 "Seamless Network-Wide IGP Migrations"

Body Dear Laurent Vanbever,

The ACM SIGCOMM 2011 Conference program committee is delighted to inform you that your paper #41 has been accepted to appear in the technical program in Toronto.

[...]

My first SIGCOMM paper (2011)

Seamless Network-Wide IGP Migrations

Laurent Vanbever*, Stefano Vissicchio†, Cristel Pelsser‡, Pierre Francois*, Olivier Bonaventure*

* Université catholique de Louvain † Roma Tre University ‡ Internet Initiative Japan
*{laurent.vanbever, pierre.francois, olivier.bonaventure} @uclouvain.be
†vissicch@dia.uniroma3.it ‡cristel@iij.ad.jp

ABSTRACT

Network-wide migrations of a running network, such as the replacement of a routing protocol or the modification of its configuration, can improve the performance, scalability, manageability, and security of the entire network. However, such migrations are an important source of concerns for network operators as the reconfiguration campaign can lead to long and service-affecting outages.

In this paper, we propose a methodology which addresses the problem of seamlessly modifying the configuration of commonly used link-state Interior Gateway Protocols (IGP). We illustrate the benefits of our methodology by considering several migration scenarios, including the addition or the removal of routing hierarchy in an existing IGP and the replacement of one IGP with another. We prove that a strict operational ordering can guarantee that the migration will not create IP transit service outages. Although finding a safe ordering is NP-complete, we describe techniques which efficiently find such an ordering and evaluate them using both real-world and inferred ISP topologies. Finally, we describe the implementation of a provisioning system which automatically performs the migration by pushing the configurations on the routers in the appropriate order, while monitoring the entire migration process.

Categories and Subject Descriptors: C.2.3 [Computer-Communication Networks]: Network Operations

General Terms: Algorithms, Management, Reliability

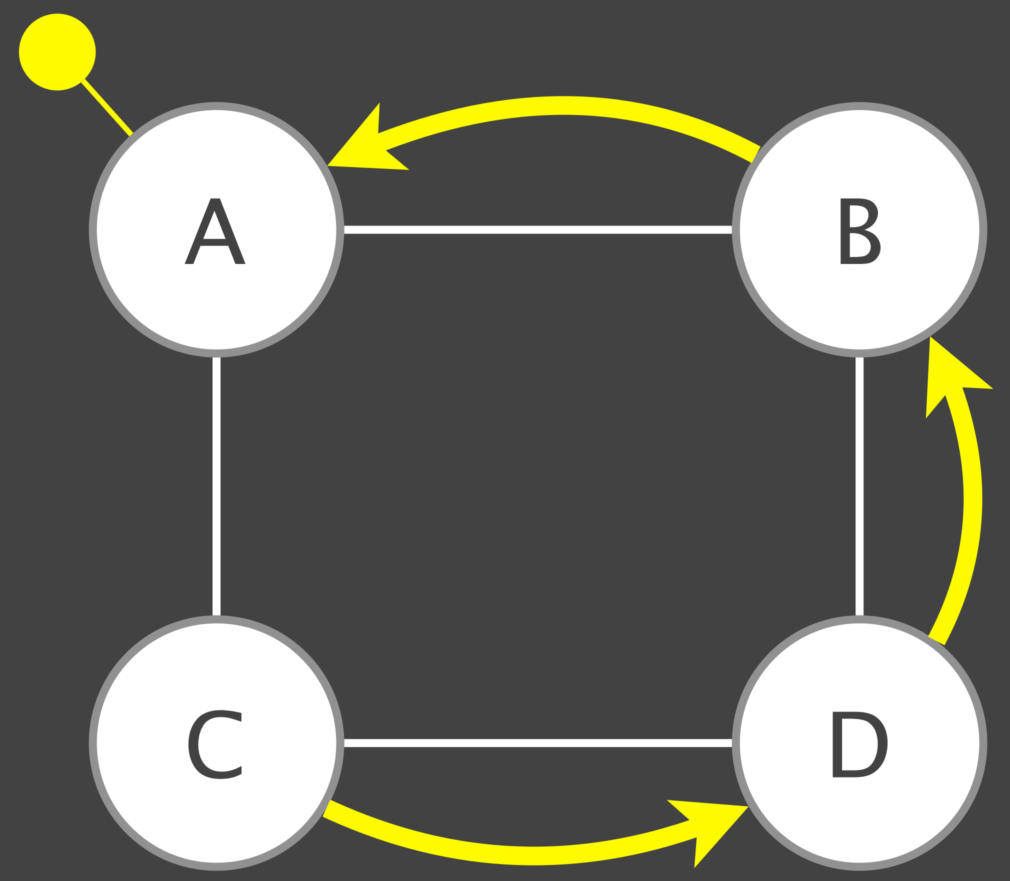
Keywords: Interior Gateway Protocol (IGP), configuration, migration, summarization, design guidelines

As the network grows or when new services have to be deployed, network operators often need to perform large-scale IGP reconfiguration [1]. Migrating an IGP is a complex process since all the routers have to be reconfigured in a proper manner. Simple solutions like restarting the network with the new configurations do not work since most of the networks carry traffic 24/7. Therefore, IGP migrations have to be performed gradually, while the network is running. Such operations can lead to significant traffic losses if they are not handled with care. Unfortunately, network operators typically lack appropriate tools and techniques to seamlessly perform large, highly distributed changes to the configuration of their networks. They also experience difficulties in understanding what is happening during a migration since complex interactions may arise between upgraded and non-upgraded routers. Consequently, as confirmed by many private communications with operators, large-scale IGP migrations are often avoided until they are absolutely necessary, thus hampering network evolvability and innovation.

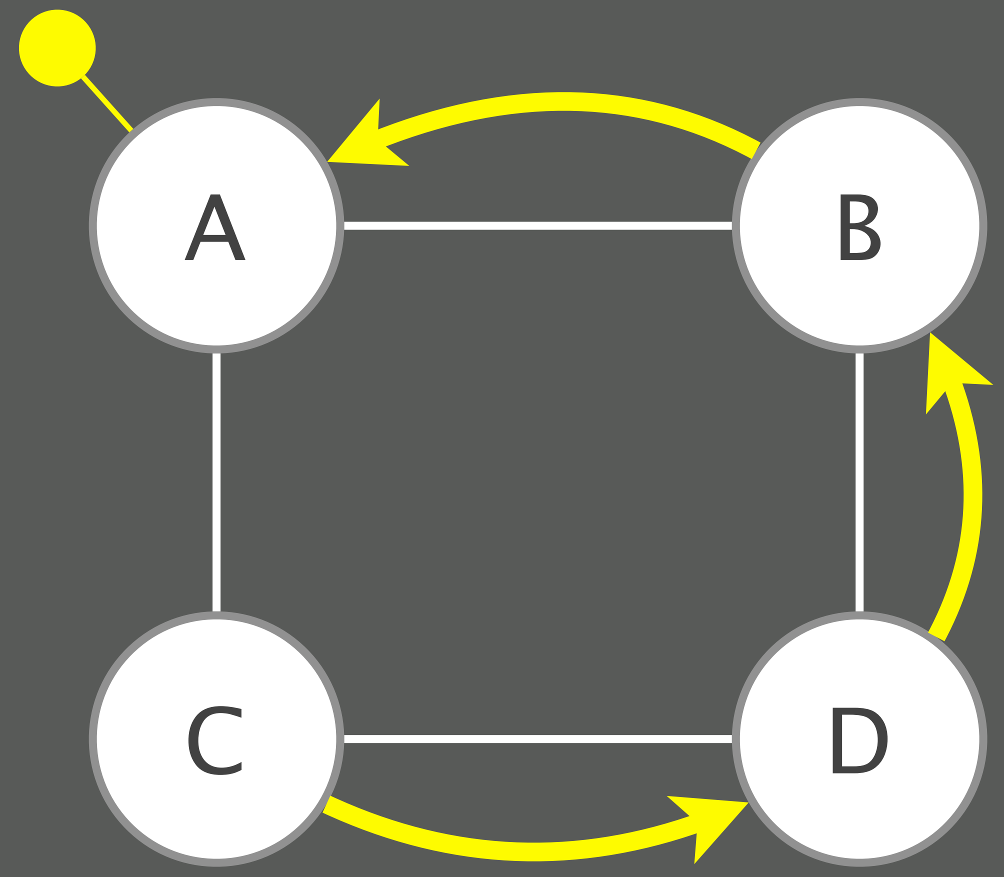
Most of the time, network operators target three aspects of the IGP when they perform large-scale migrations. First, they may want to replace the current protocol with another. For instance, several operators have switched from OSPF to IS-IS because IS-IS is known to be more secure against control-plane attacks [2, 3]. Operators may also want to migrate to an IGP that is not dependent on the address family (e.g., OSPFv3, IS-IS) in order to run only one IGP to route both IPv4 and IPv6 traffic [4, 3], or to change IGP in order to integrate new equipments which are not compliant with the adopted one [5]. Second, when the number of routers exceeds a certain critical mass, operators often introduce a hierarchy within their IGP to limit the control-plane

How do you reconfigure a network
without loosing reachability?

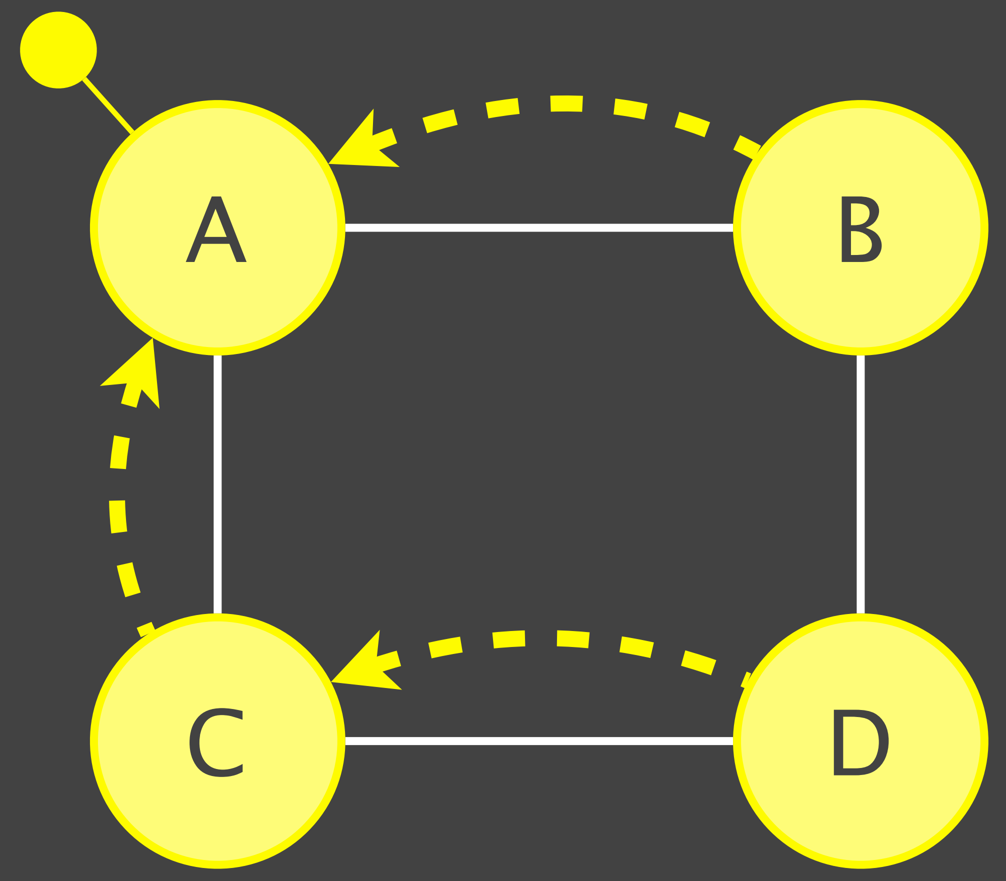
initial forwarding state



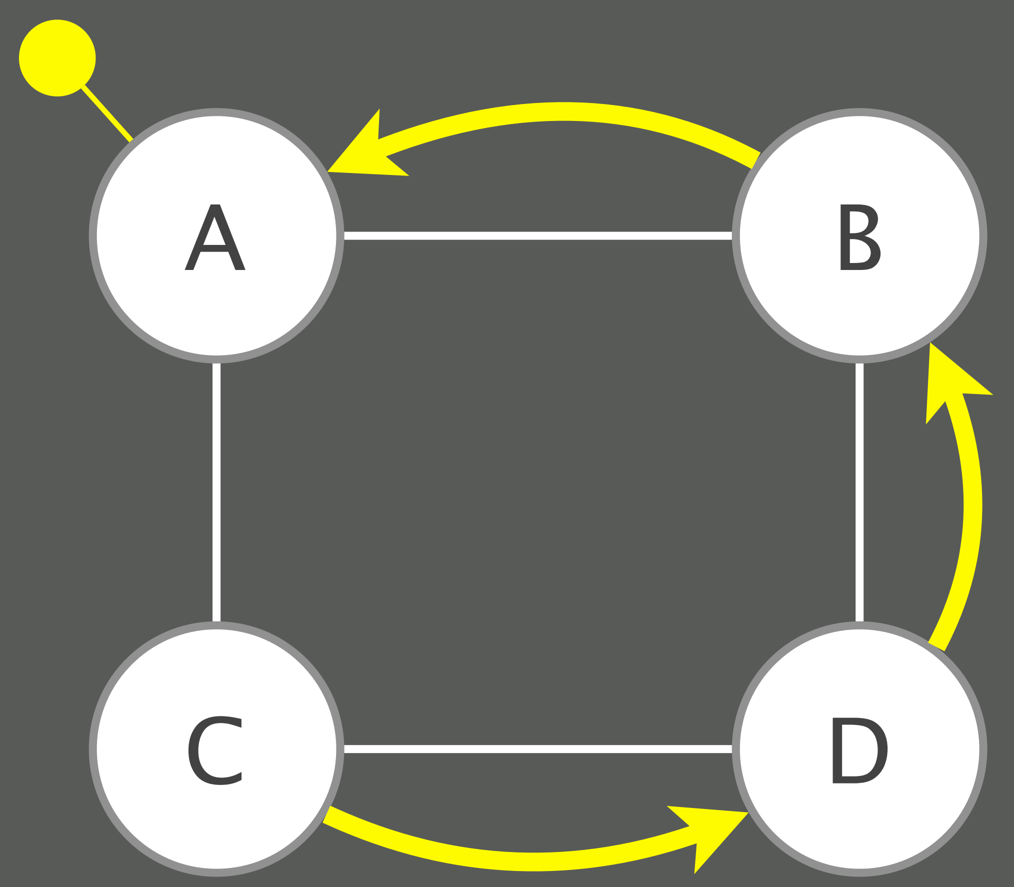
initial forwarding state



final forwarding state

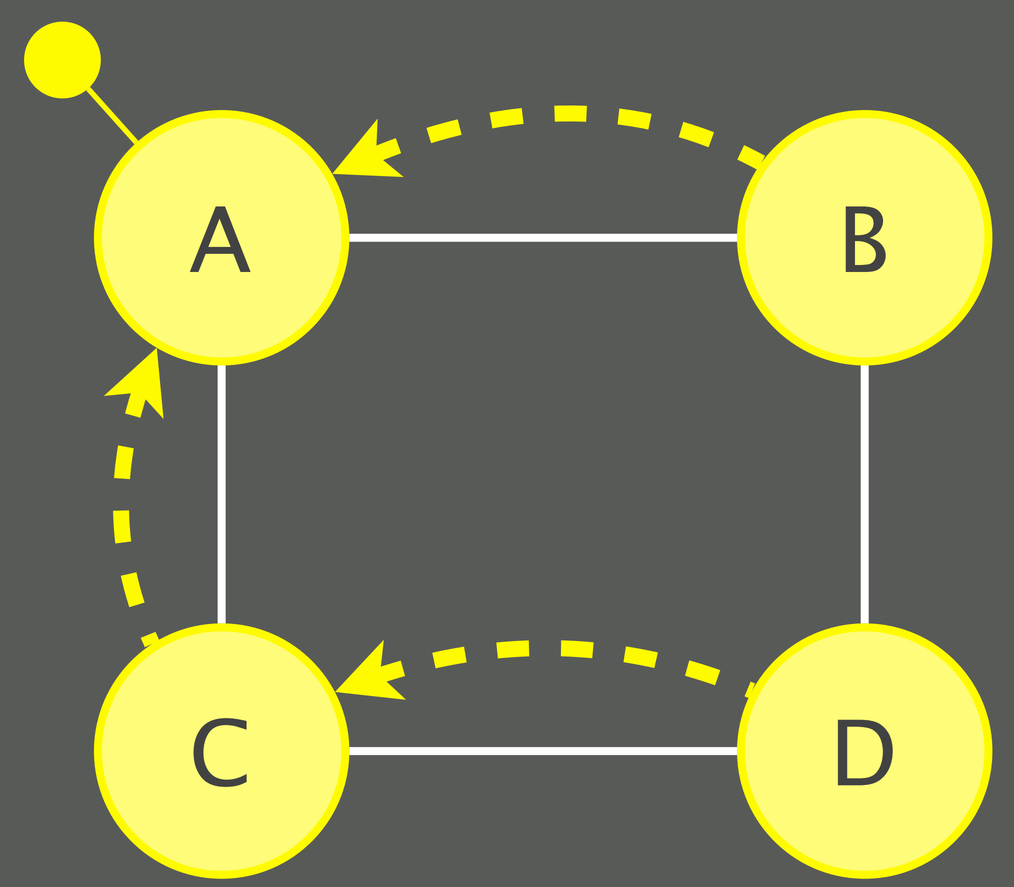


initial forwarding state

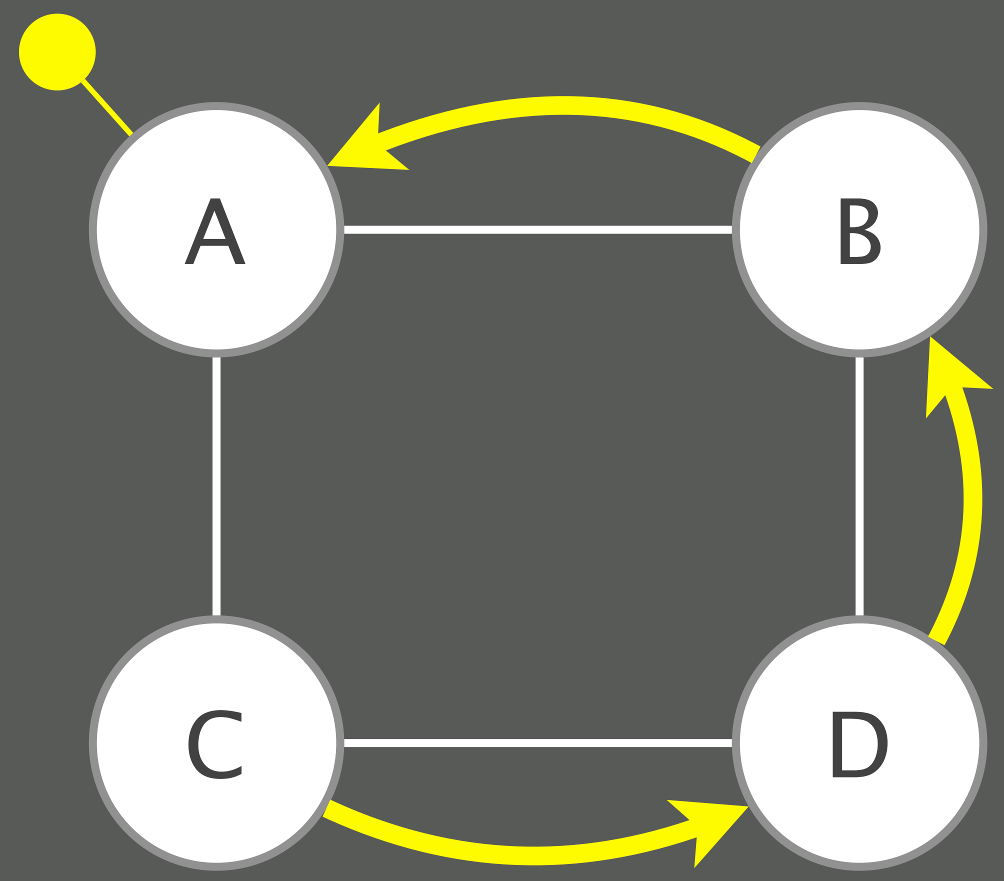


How do you reconfigure a network without loosing reachability?

final forwarding state

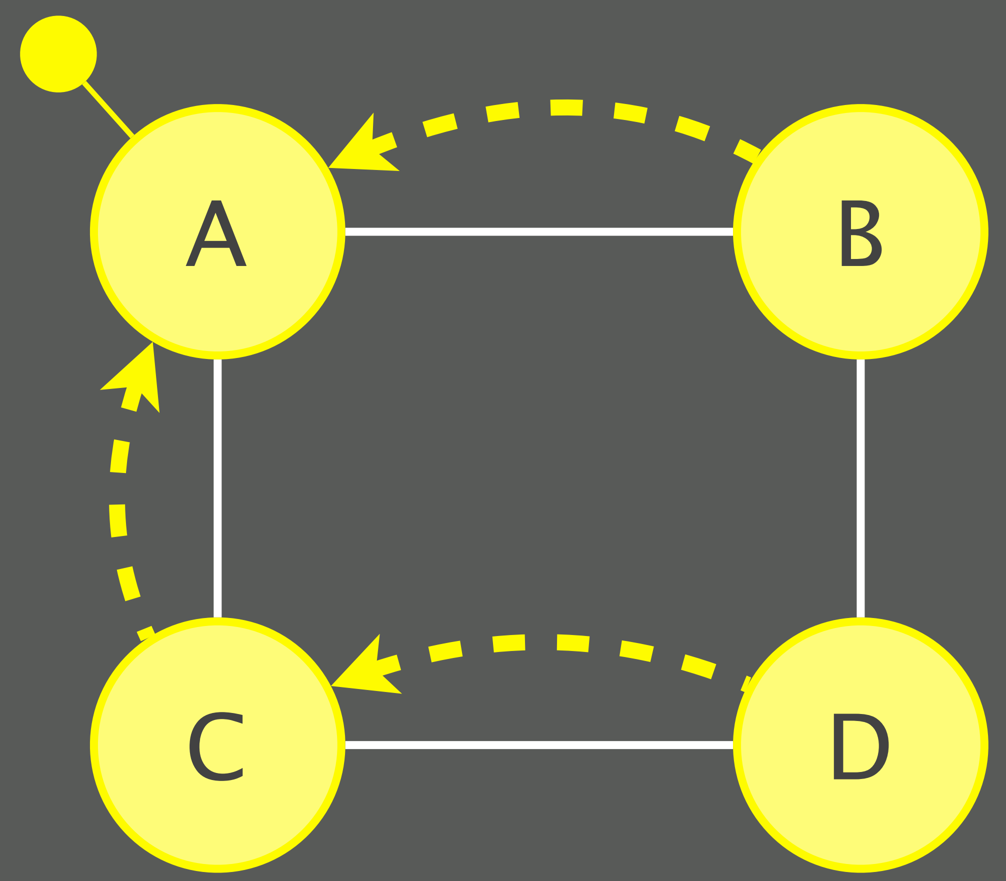


initial forwarding state



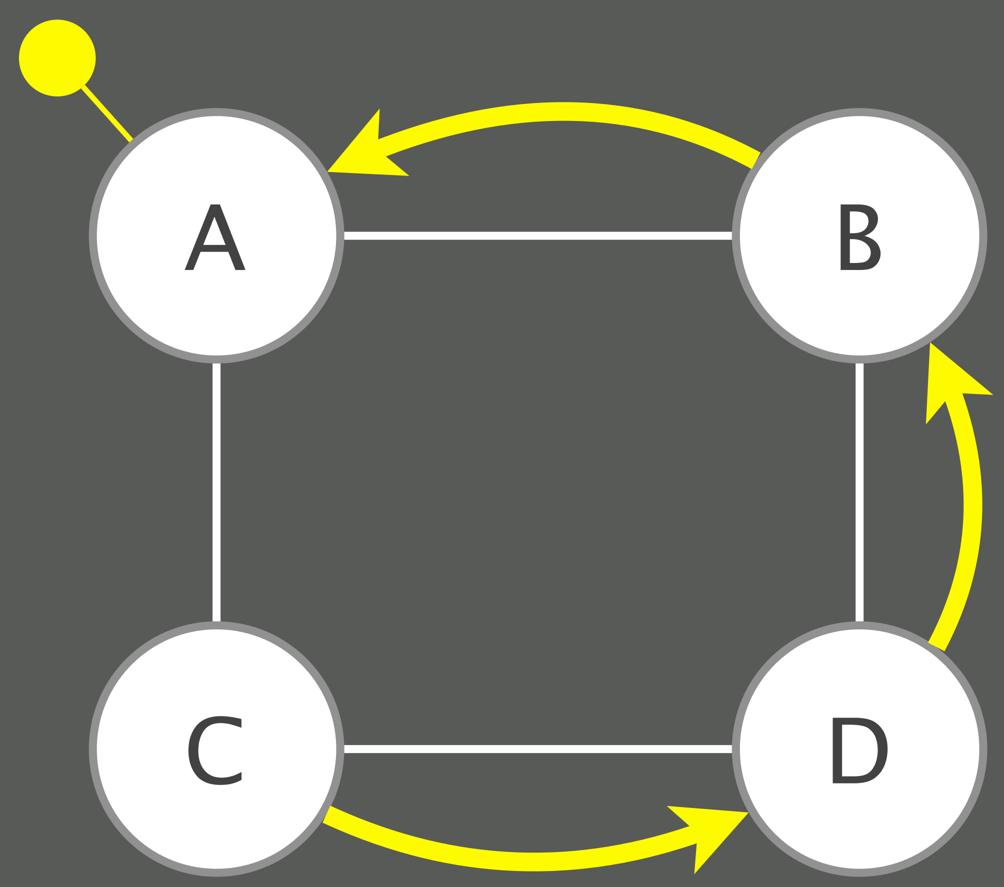
intermediate
forwarding state

final forwarding state

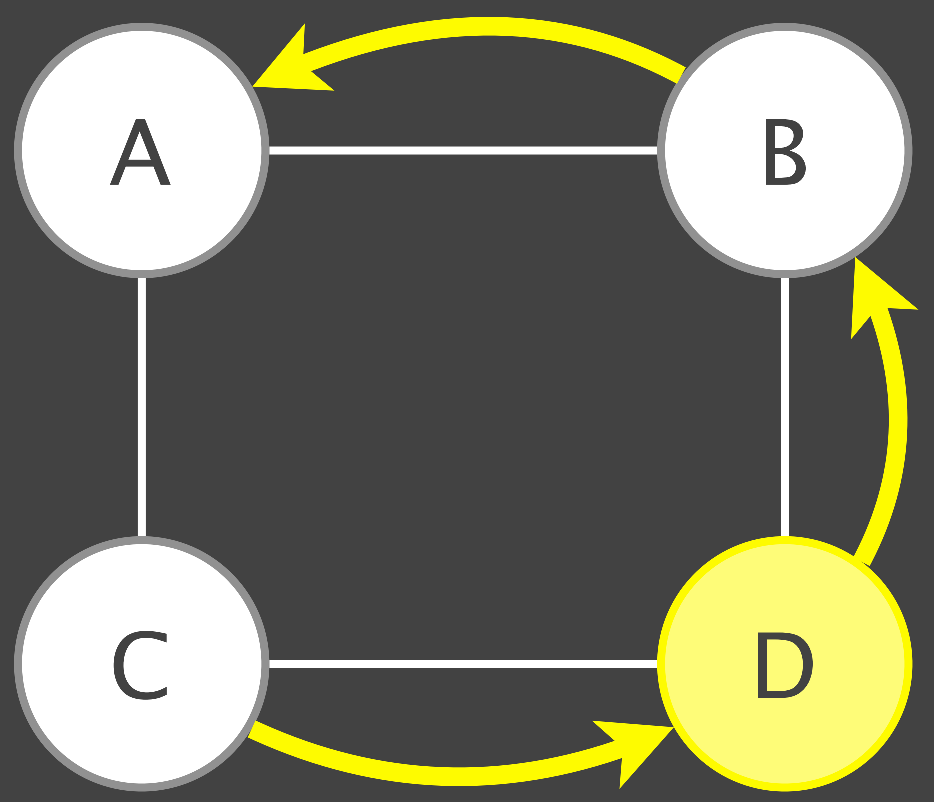



initial forwarding state
A
B
C
D
intermediate
forwarding state
A
B
C
D
final forwarding state
A
B
C
D

initial forwarding state

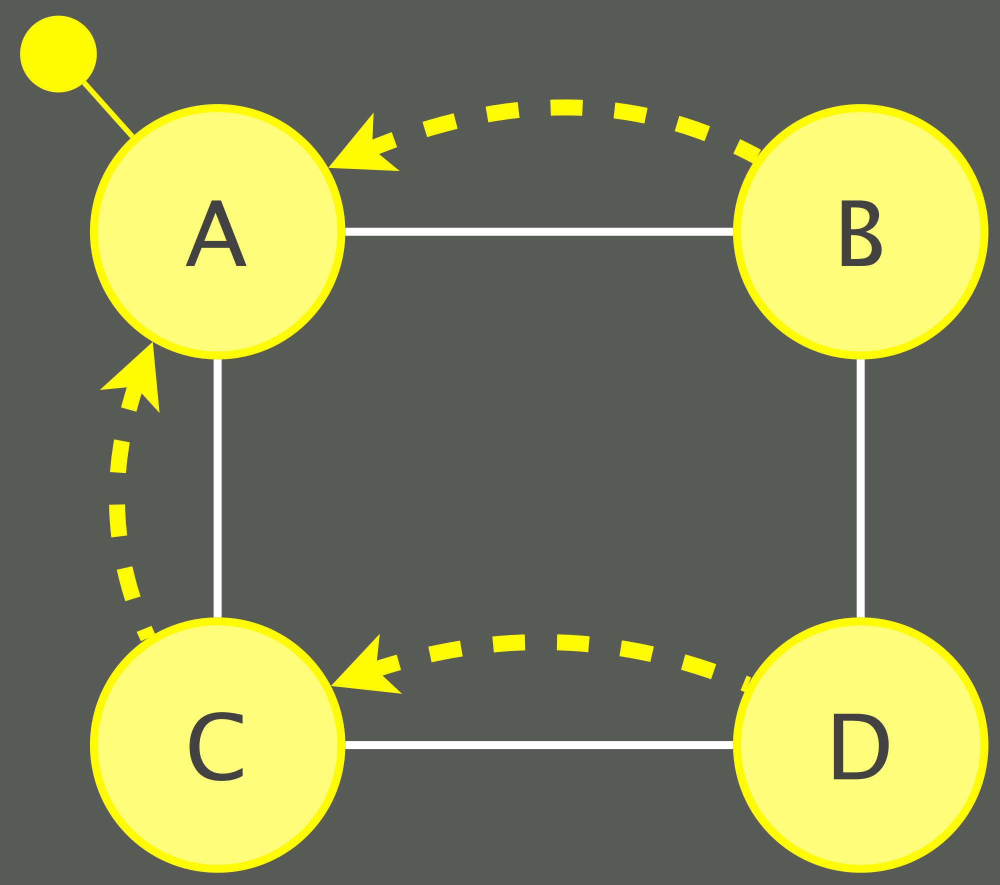


intermediate
forwarding state

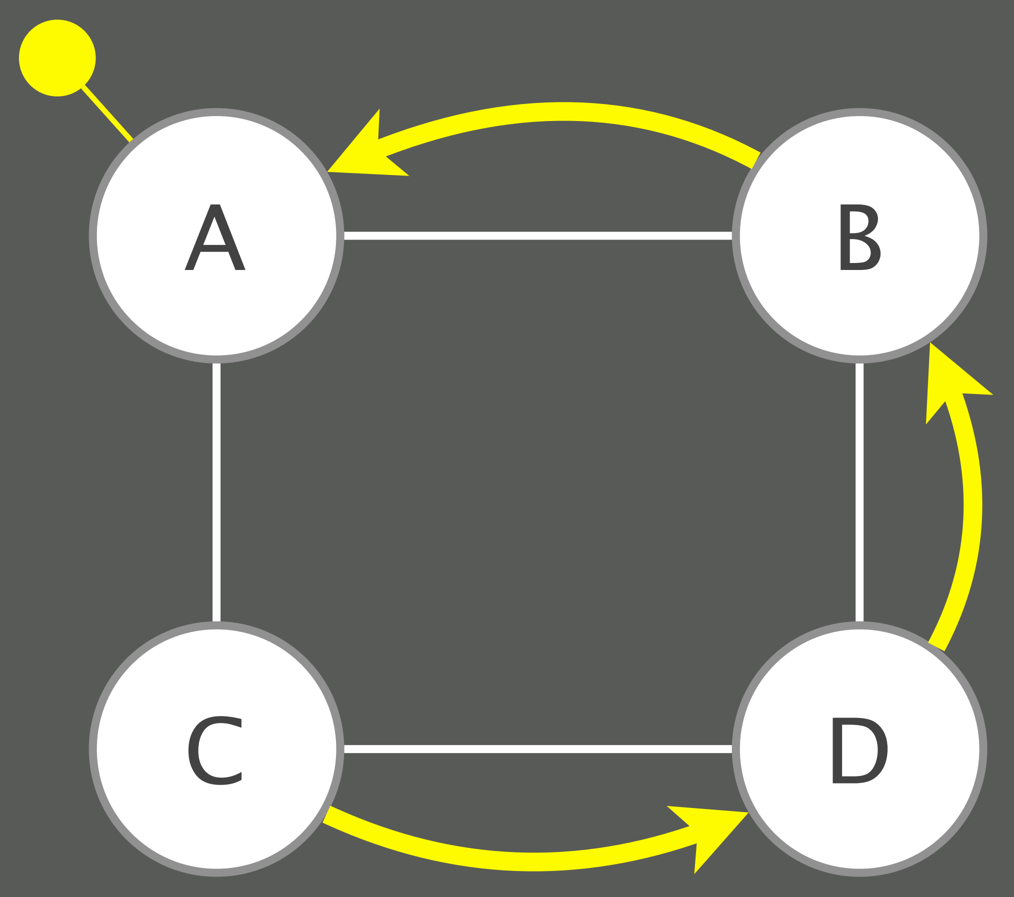


What if we reconfigure D first?

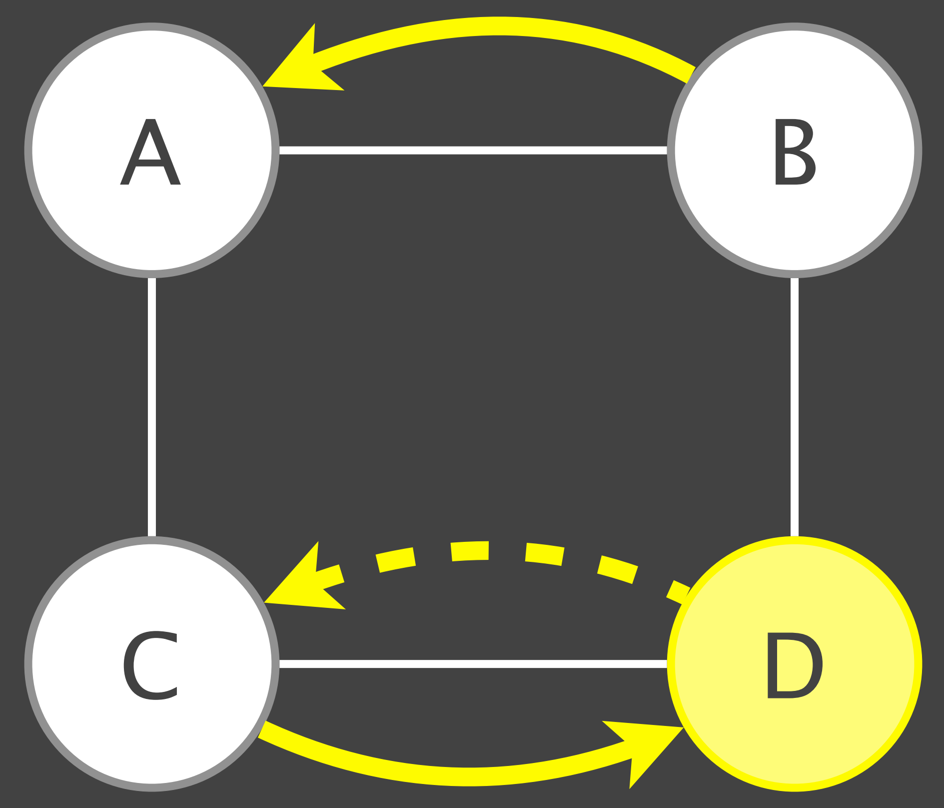
final forwarding state



initial forwarding state

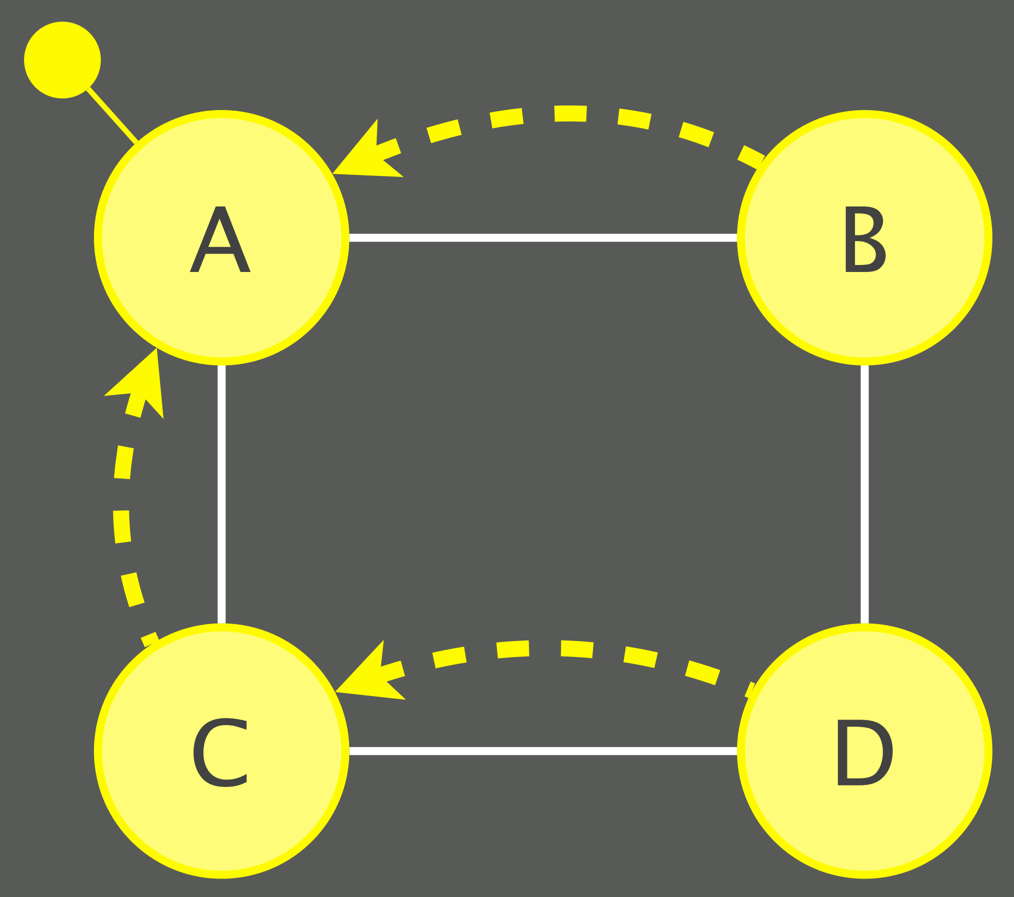


intermediate
forwarding state

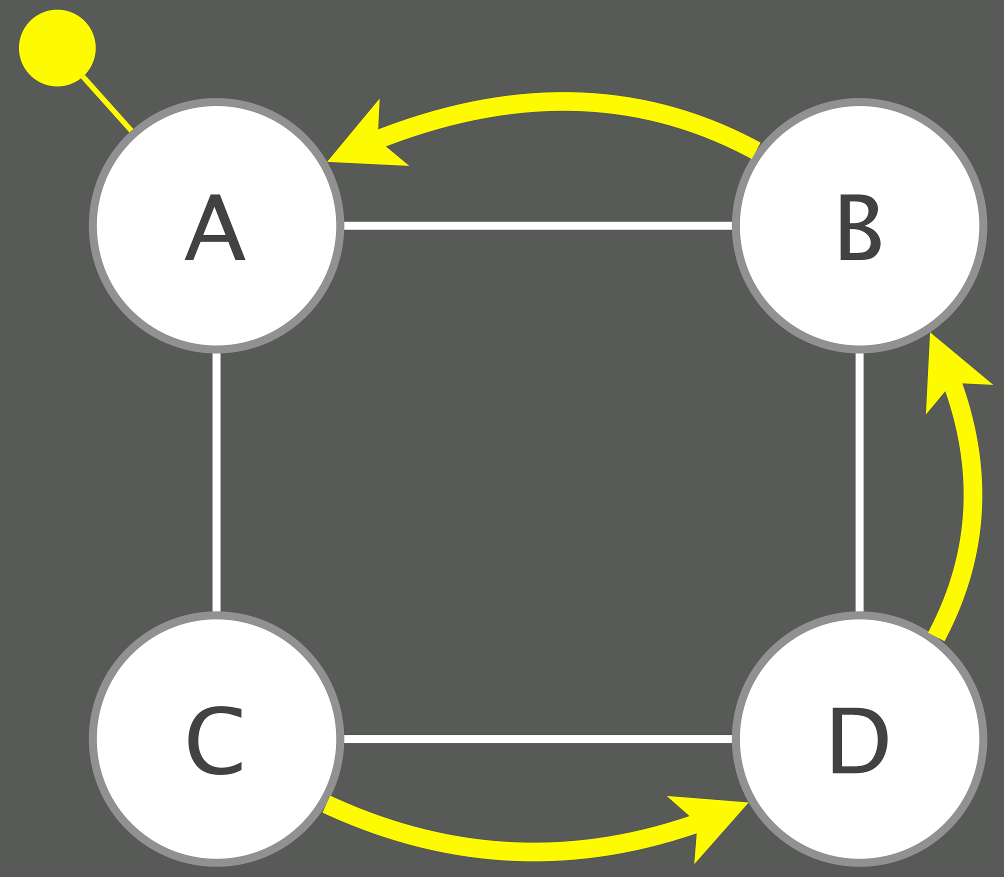


What if we reconfigure D first?

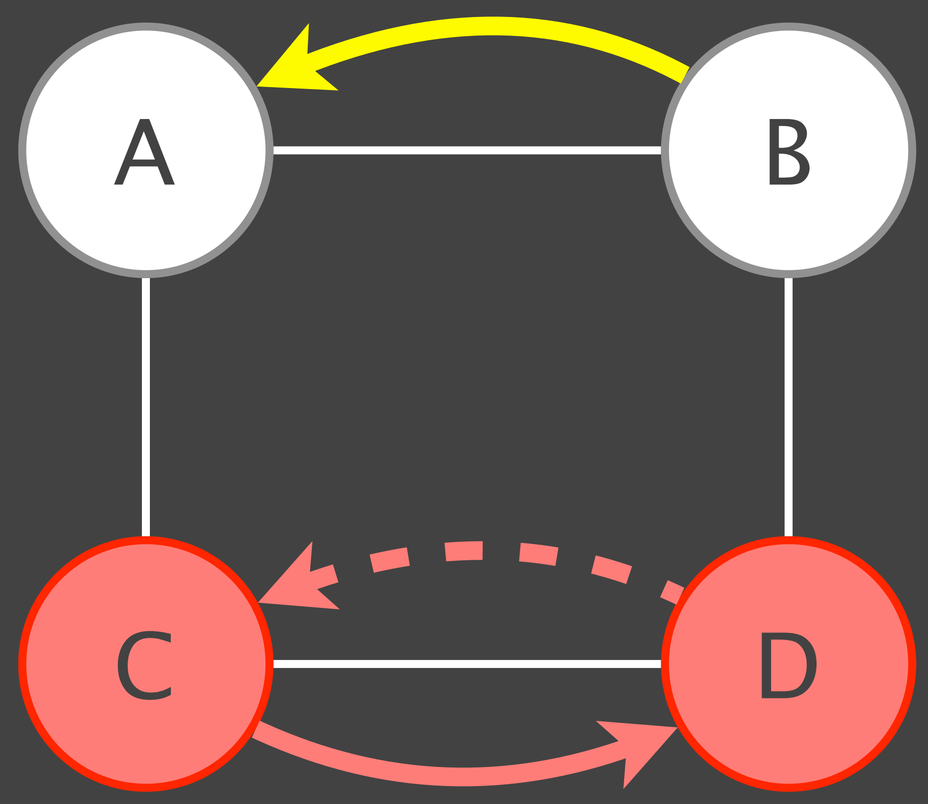
final forwarding state



initial forwarding state



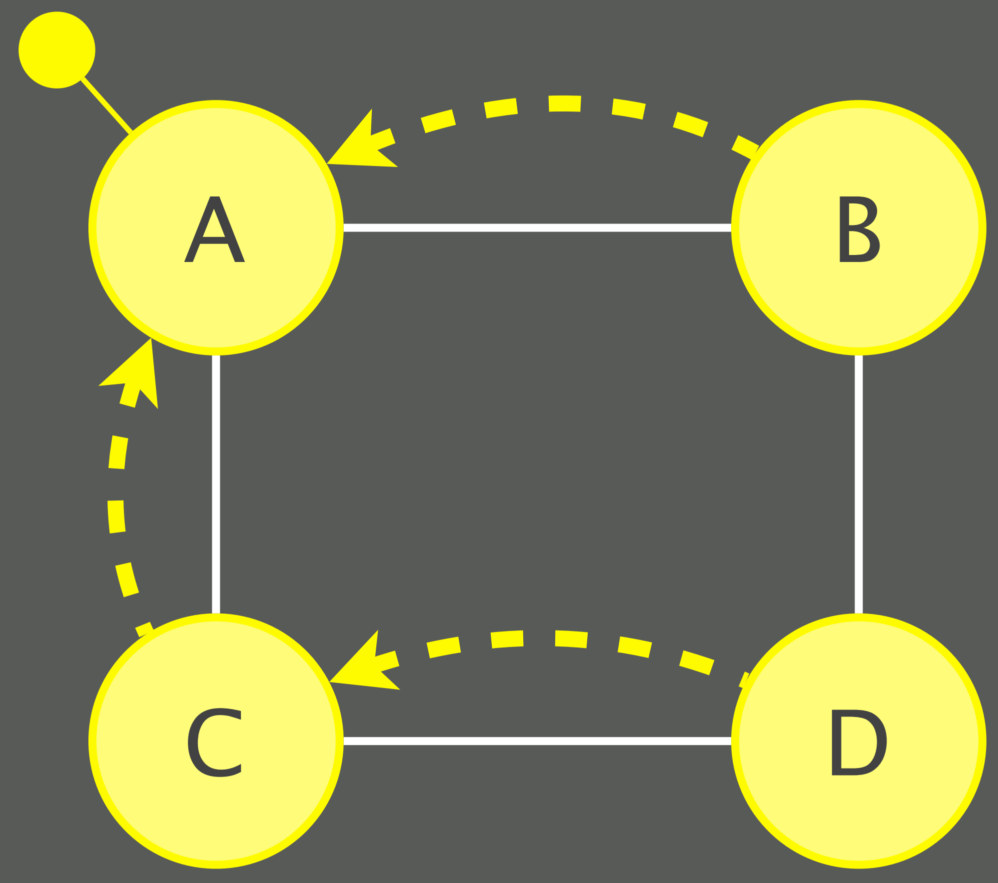
intermediate
forwarding state



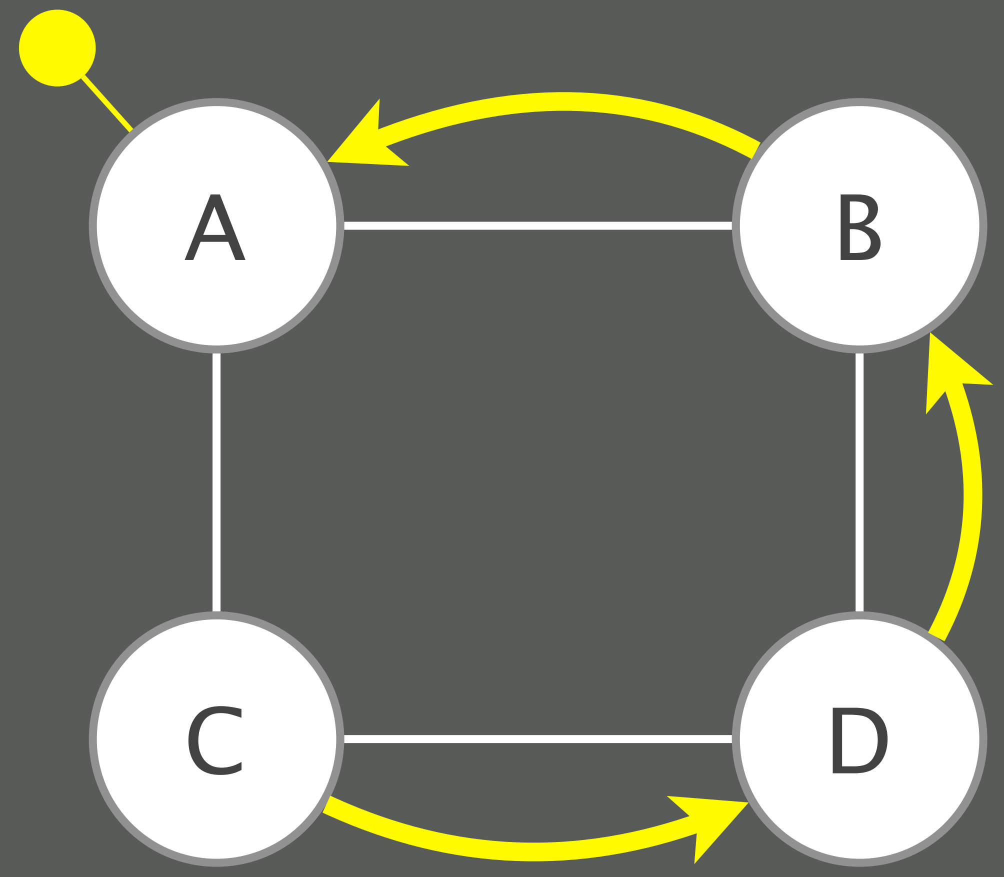
What if we reconfigure D first?

We create a forwarding loop

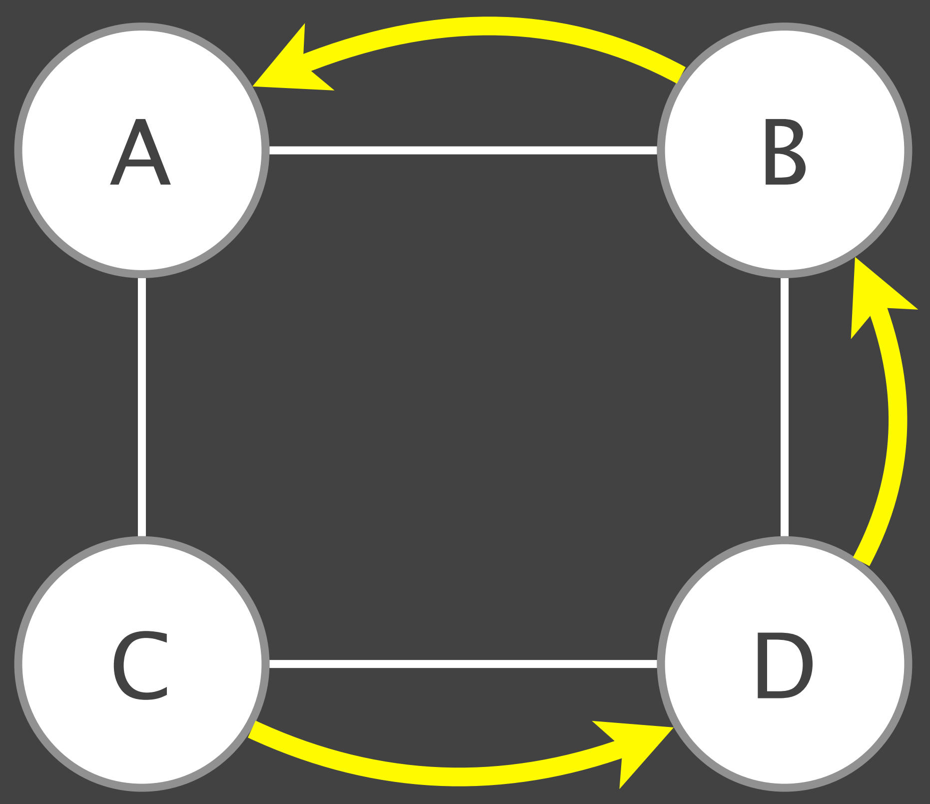
final forwarding state



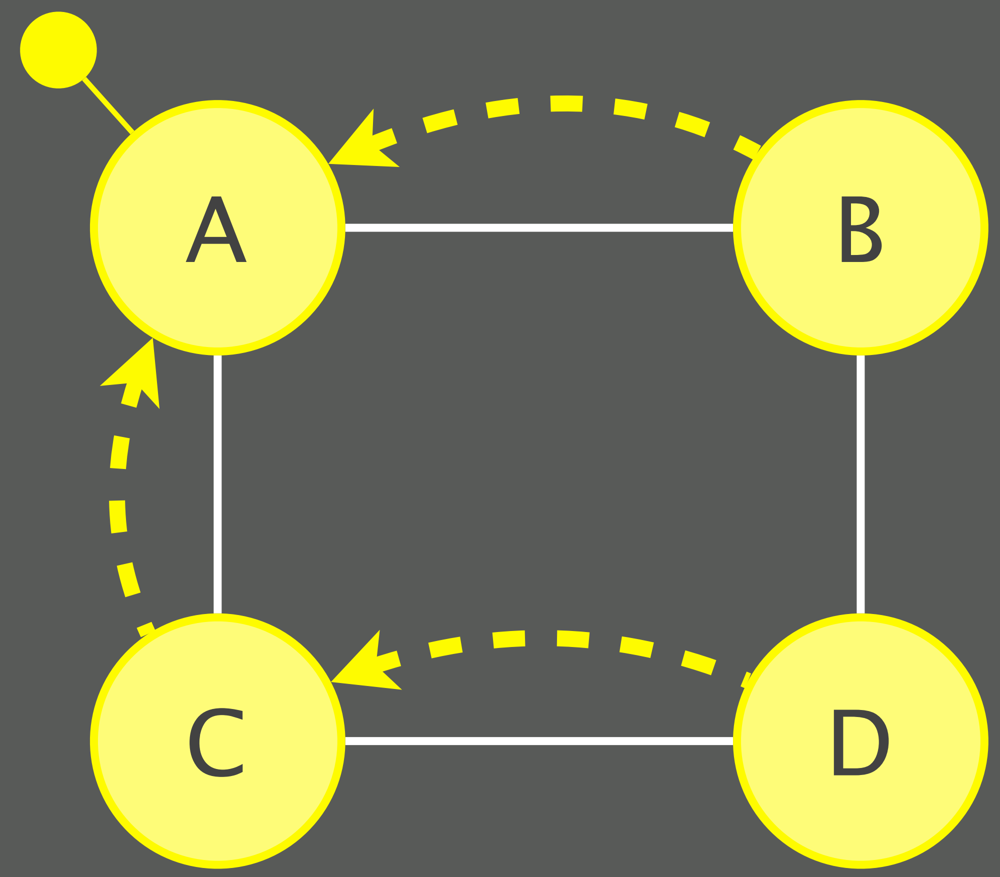
initial forwarding state



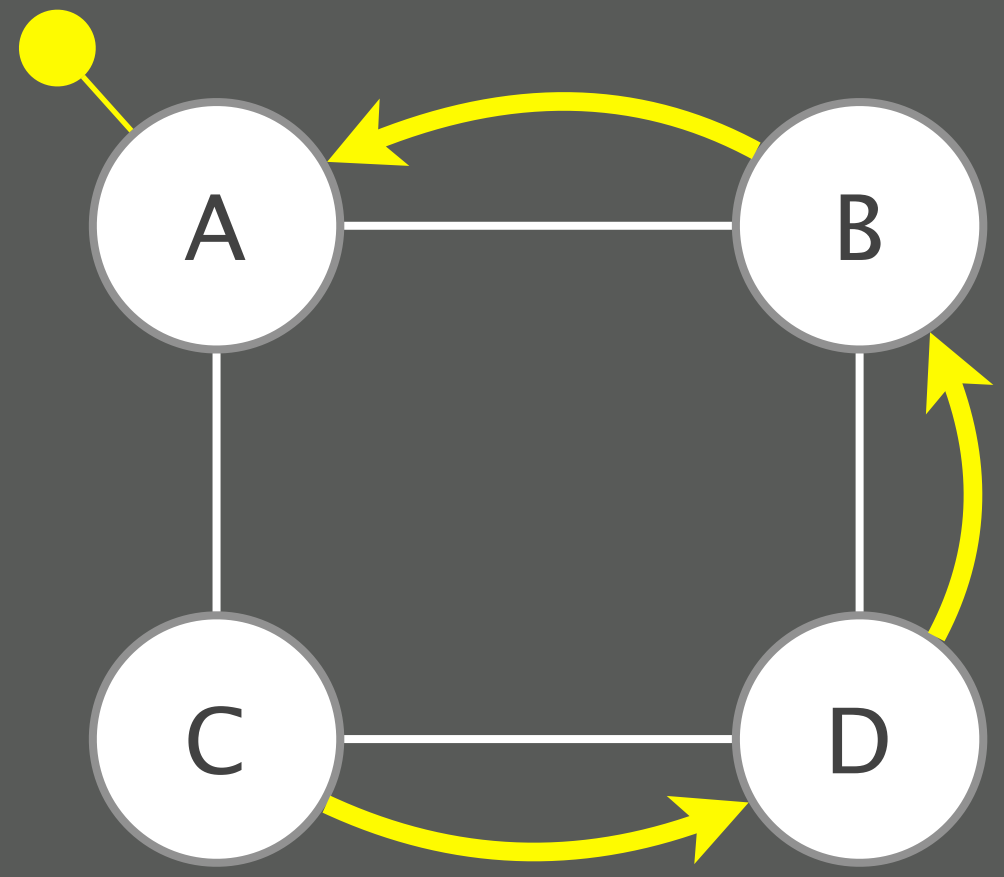
intermediate
forwarding state



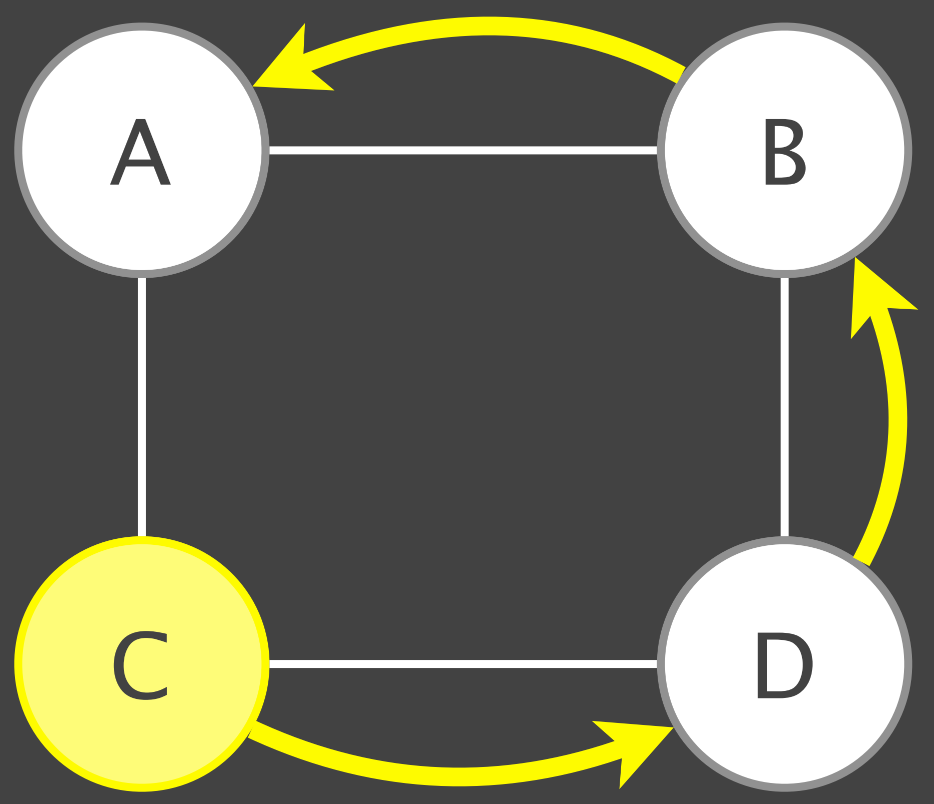
final forwarding state



initial forwarding state

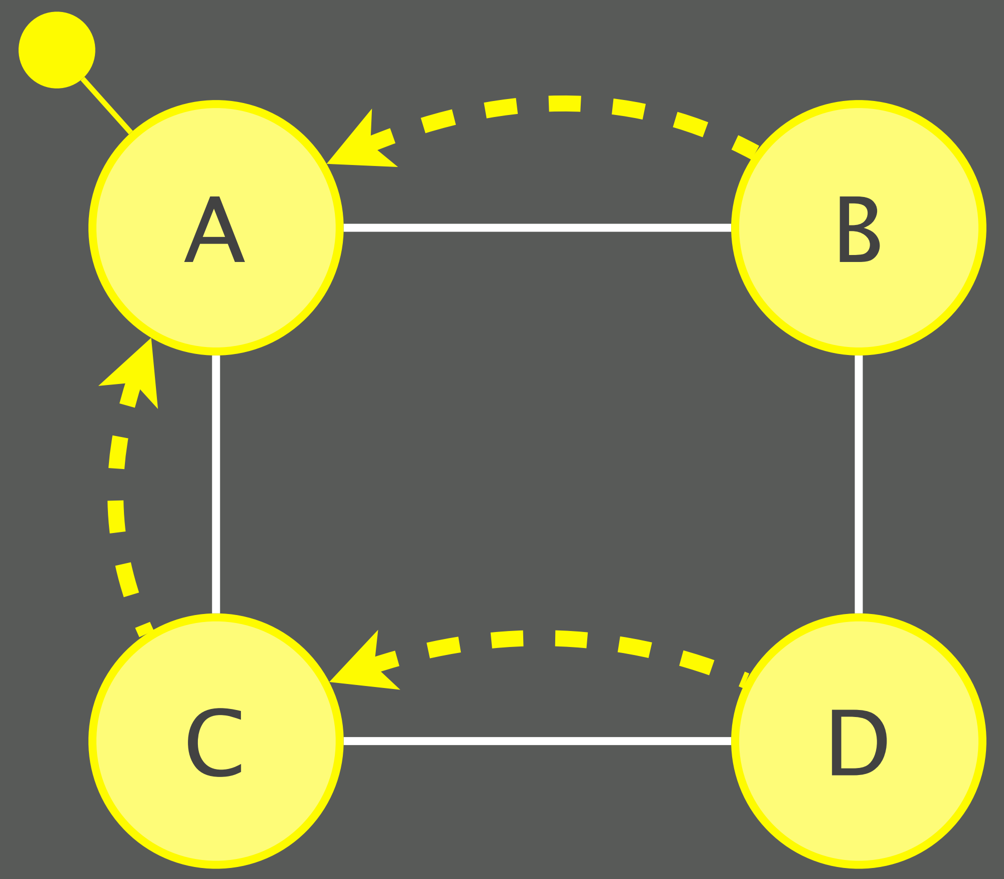


intermediate
forwarding state

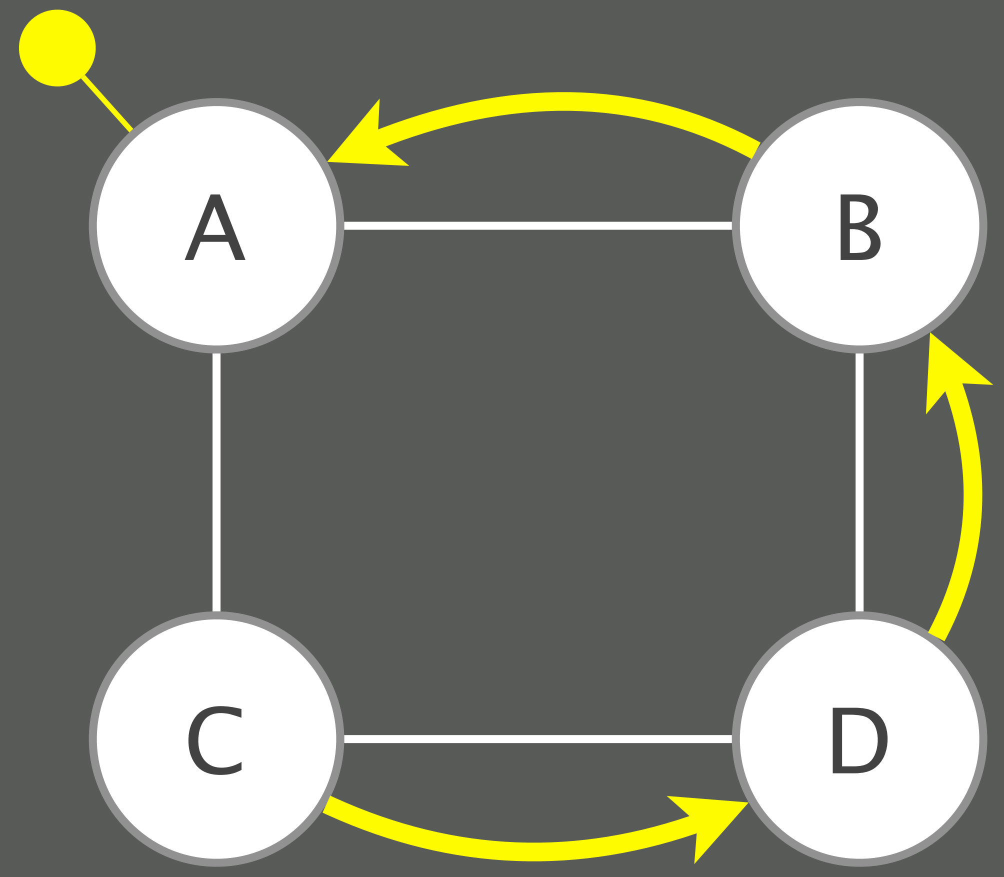


What if we reconfigure C first?

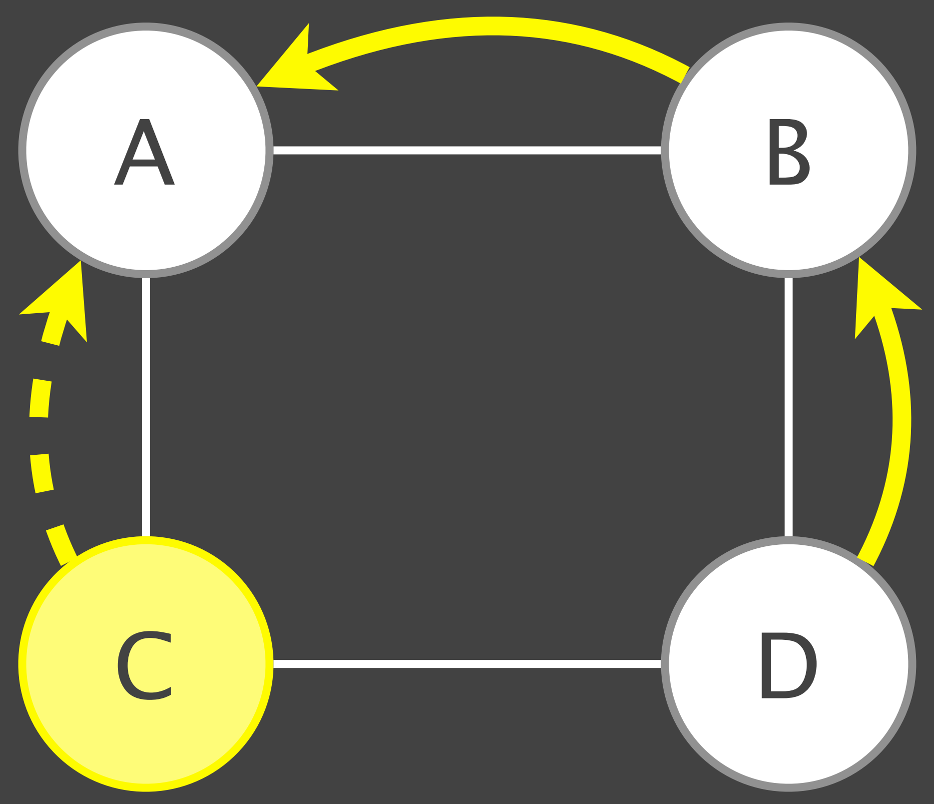
final forwarding state



initial forwarding state



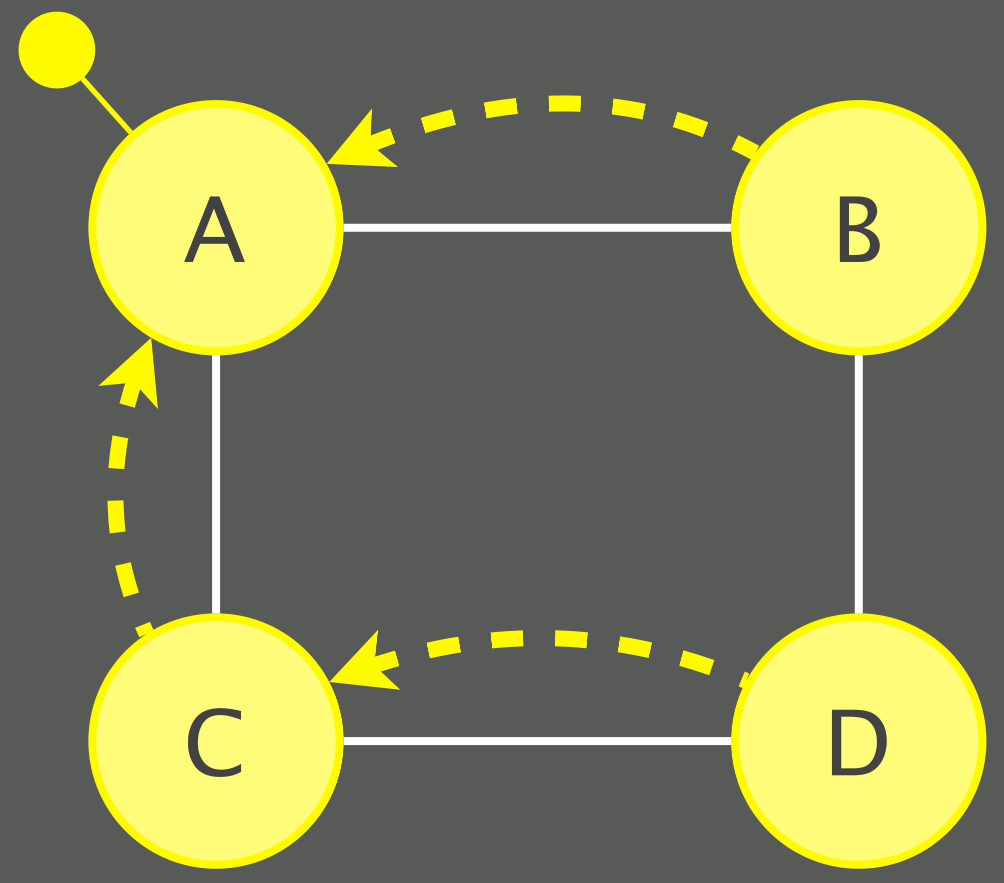
intermediate forwarding state



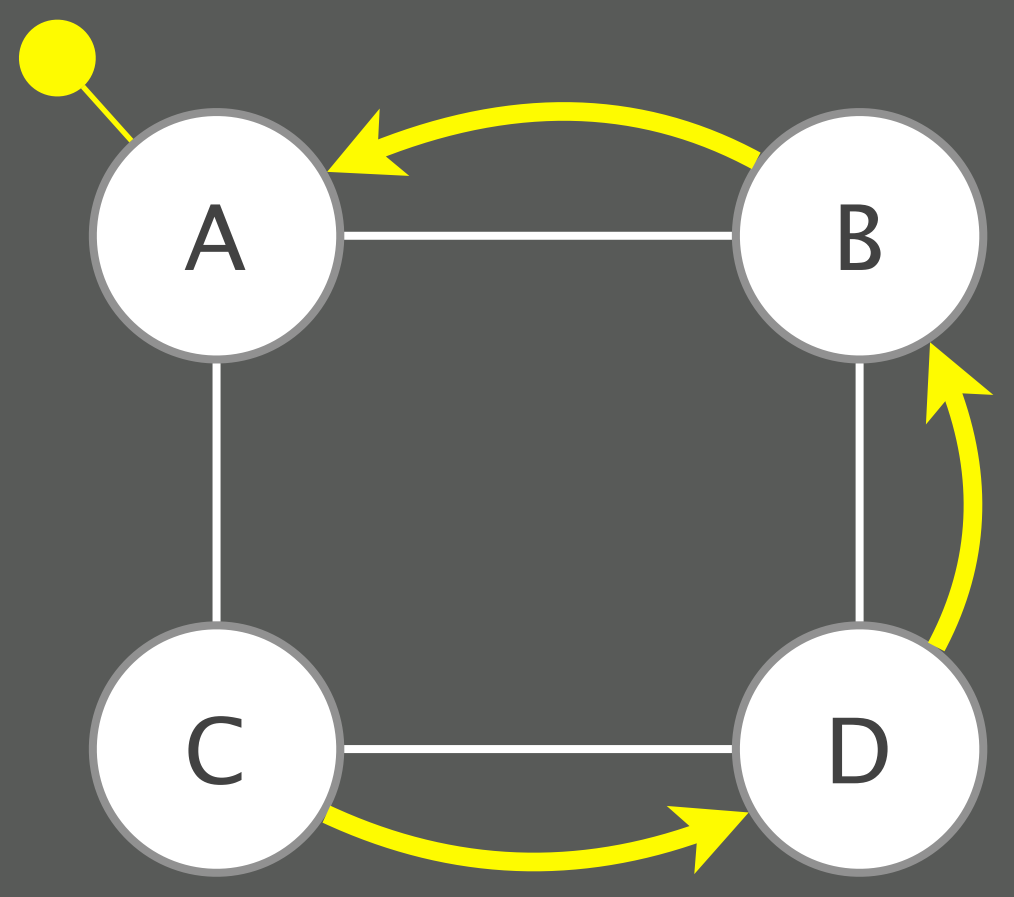
What if we reconfigure C first?

Works!

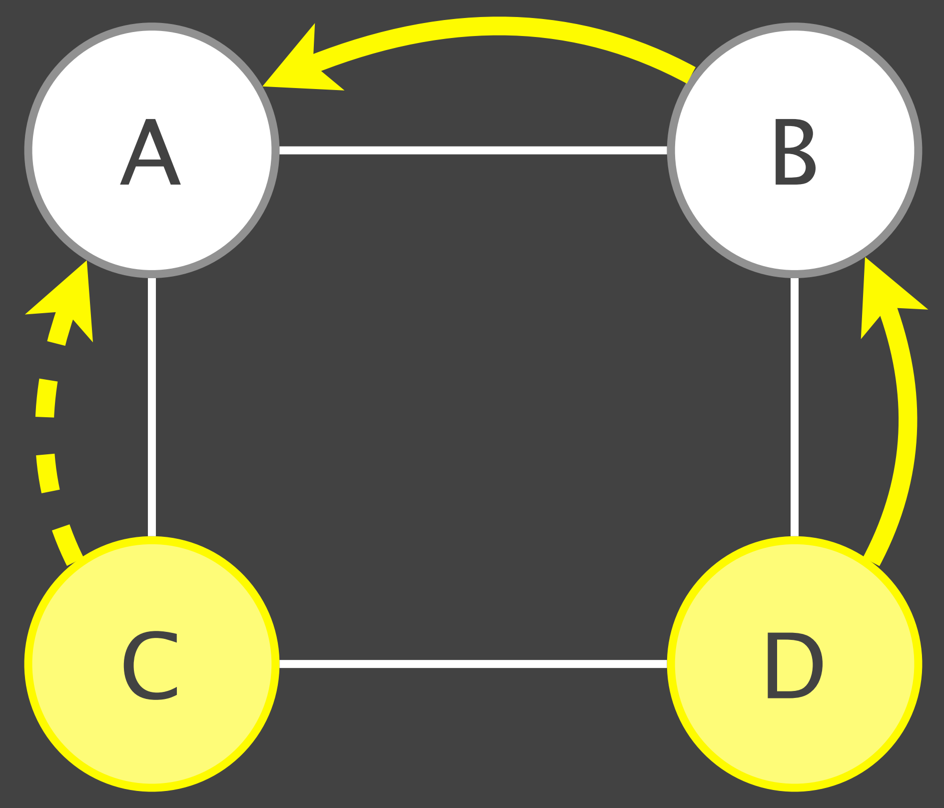
final forwarding state



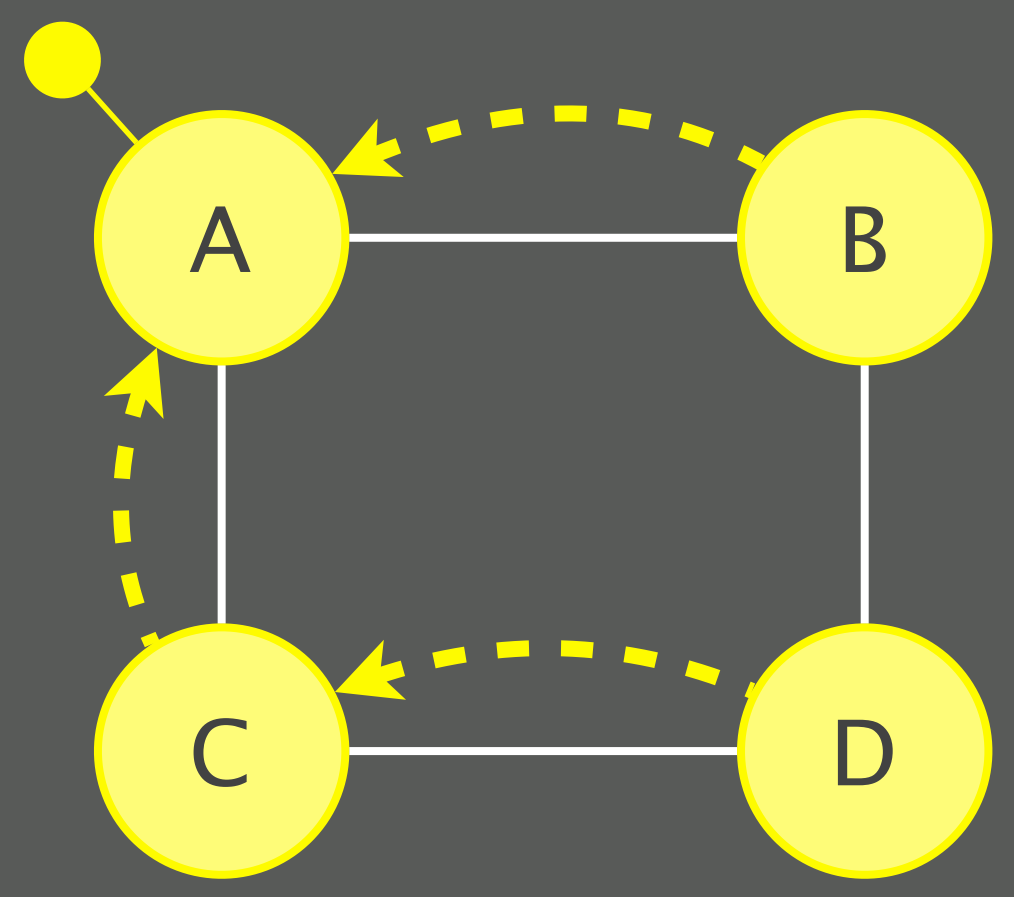
initial forwarding state

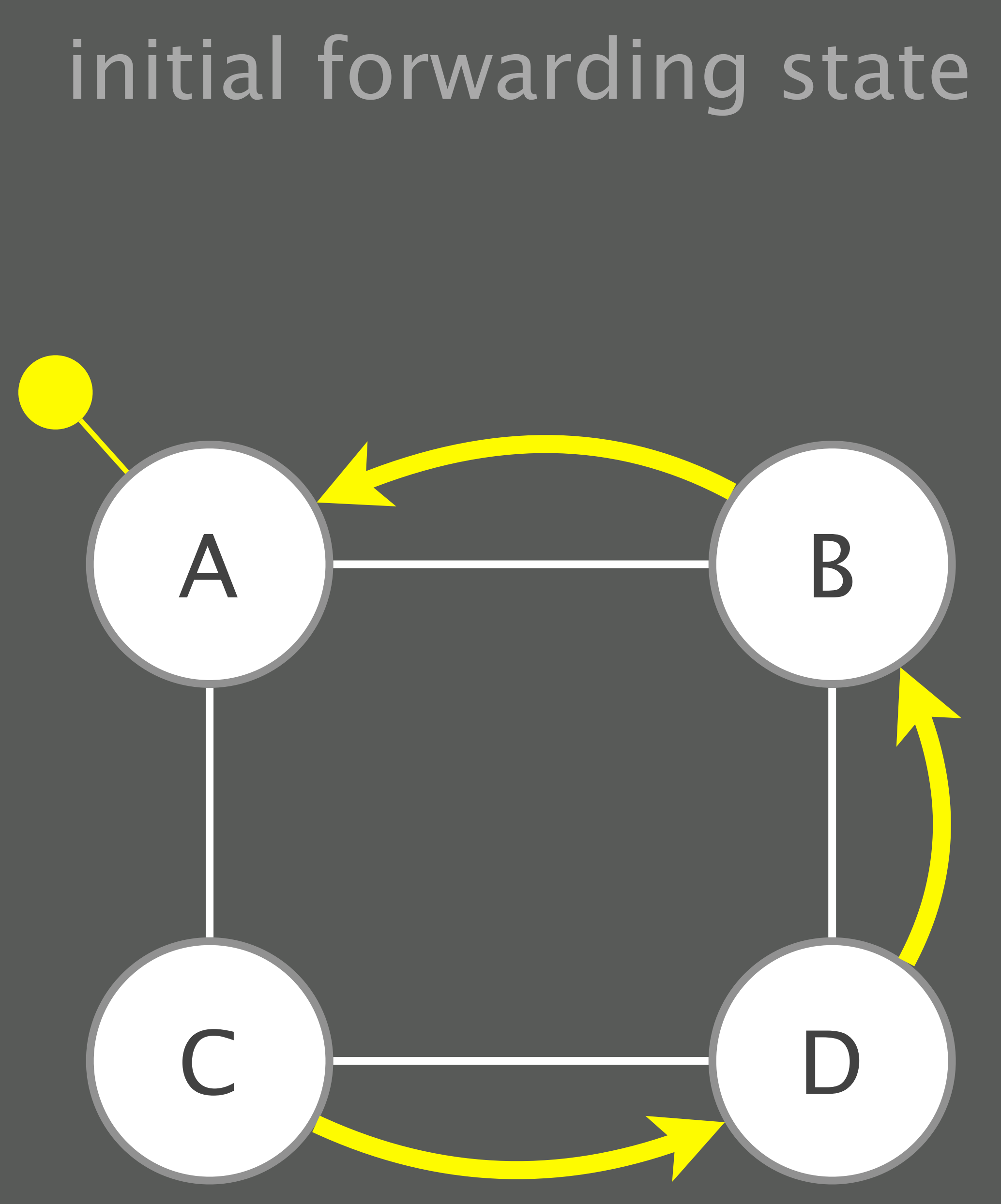


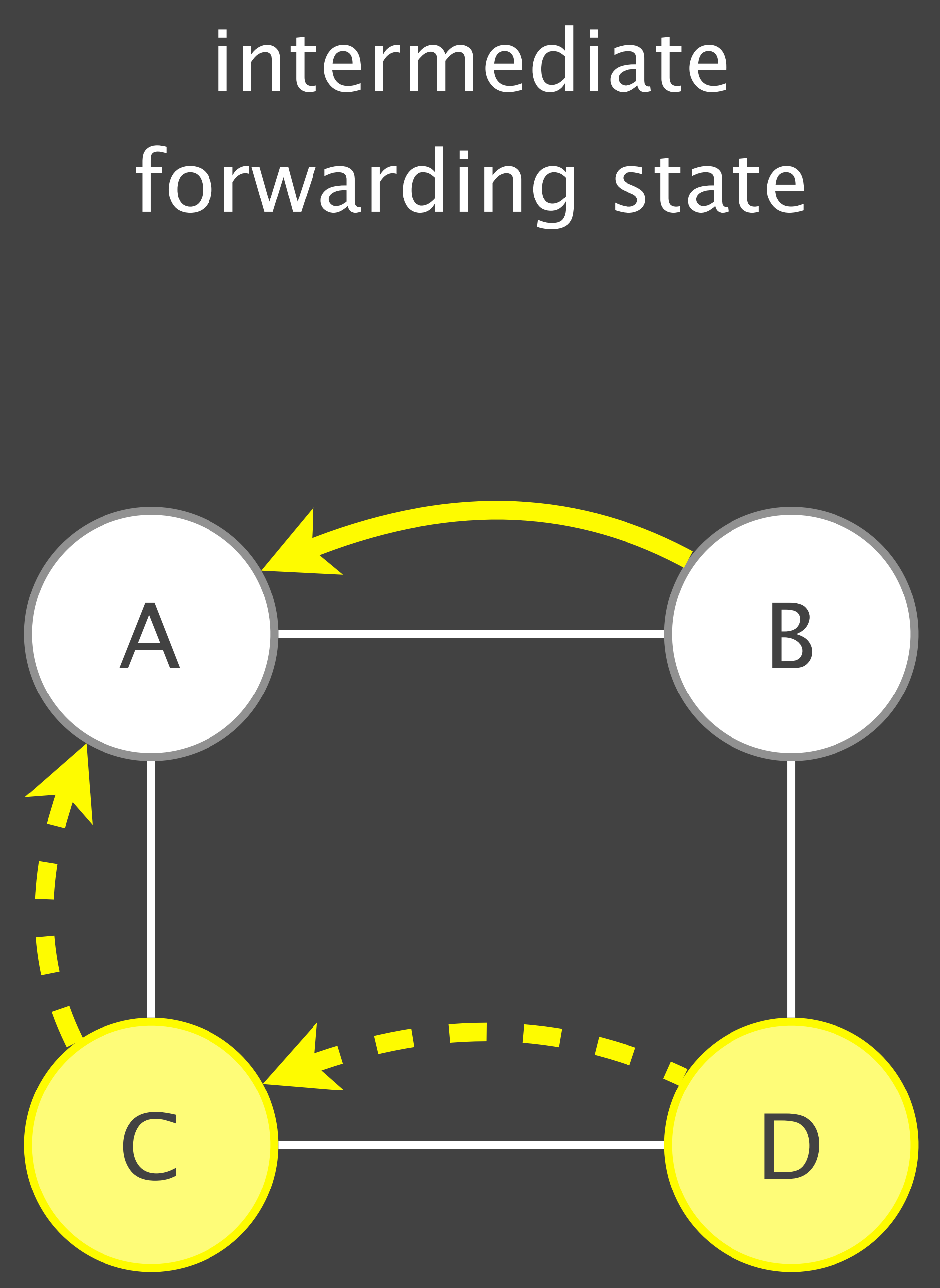
intermediate
forwarding state

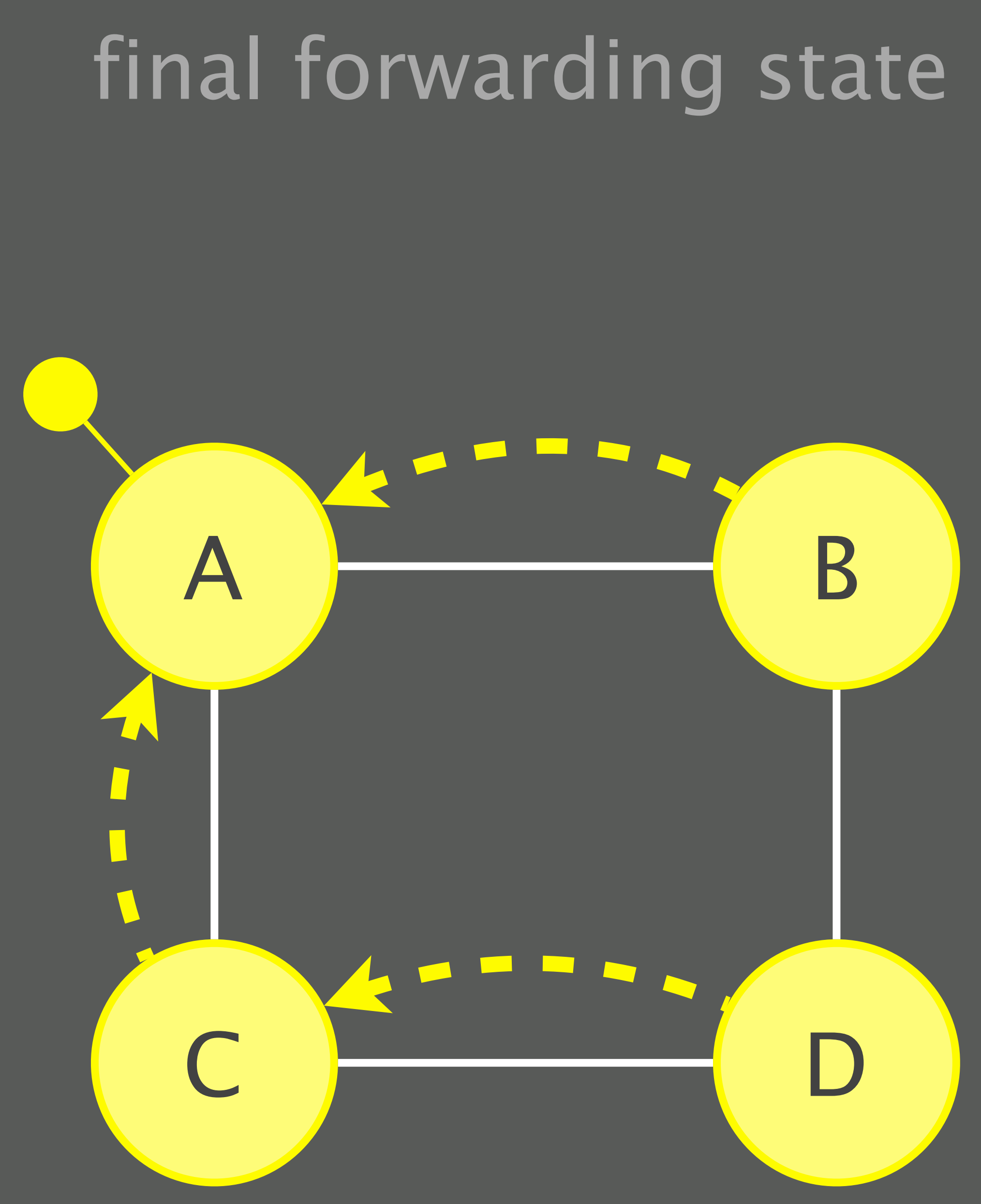


final forwarding state

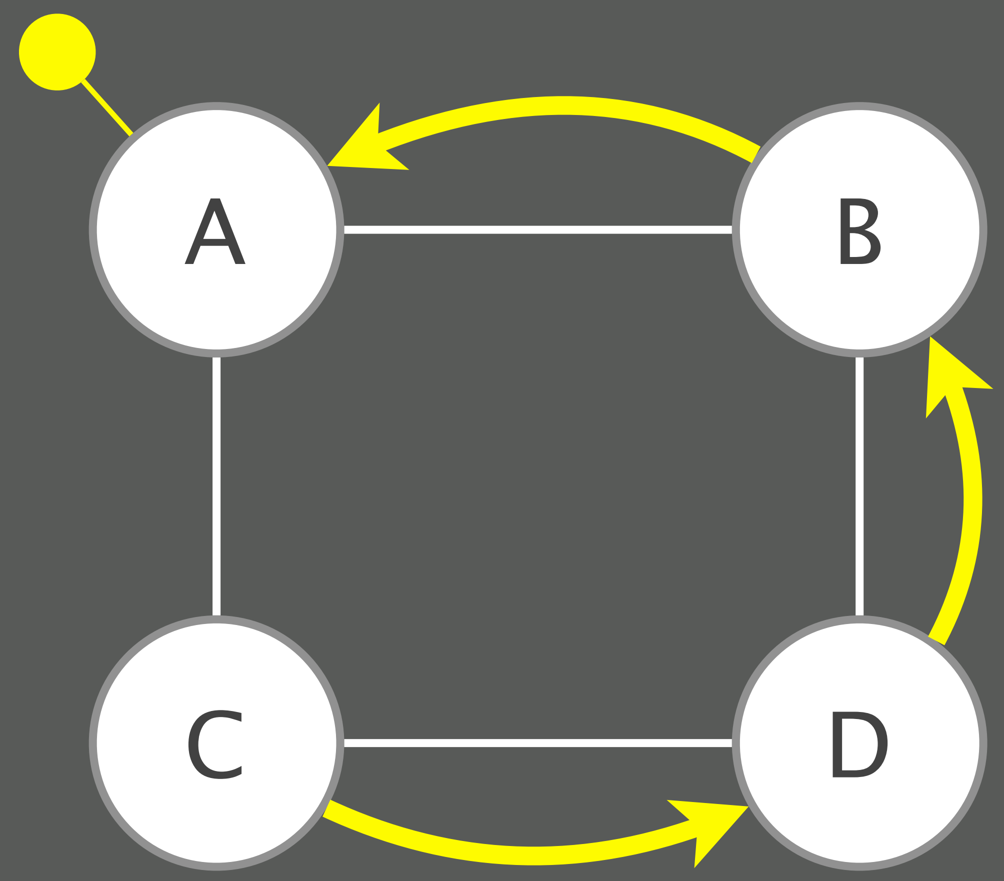



initial forwarding state
A
B
C
D

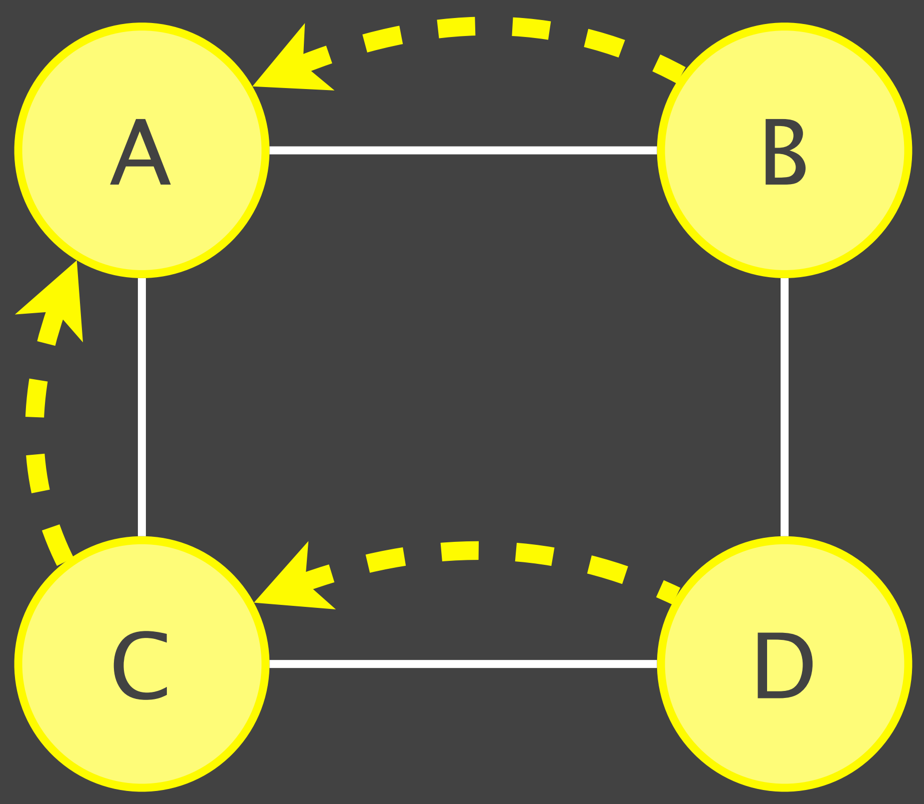

intermediate
forwarding state
A
B
C
D


final forwarding state
A
B
C
D

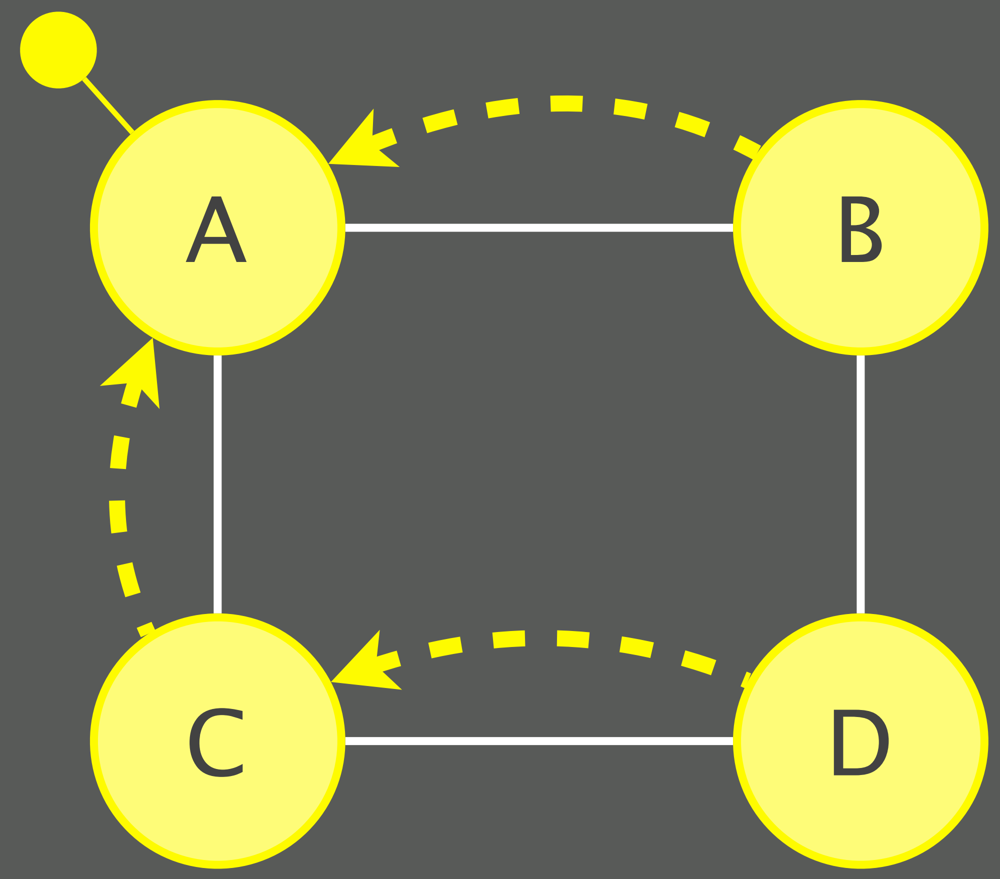
initial forwarding state



intermediate forwarding state



final forwarding state



How do you reconfigure a network
without loosing reachability?

This was easy to compute
for *one* destination, but…

How do you reconfigure a network
without loosing reachability?

This was easy to compute
for *one* destination, but…

what if you have many?

Finding an ordering preserving reachability is hard

Contributions

Prove that finding an ordering is NP-complete
by reducing from the 3-SAT problem

Design practical algorithms and heuristics
based on necessary/sufficient conditions

Implement an orchestration system
which applies the updates to a live network

My first SIGCOMM paper (2011)

Seamless Network-Wide IGP Migrations

Laurent Vanbever*, Stefano Vissicchio†, Cristel Pelsser‡, Pierre Francois*, Olivier Bonaventure*

* Université catholique de Louvain † Roma Tre University ‡ Internet Initiative Japan
*{laurent.vanbever, pierre.francois, olivier.bonaventure} @uclouvain.be
†vissicch@dia.uniroma3.it ‡cristel@iij.ad.jp

ABSTRACT

Network-wide migrations of a running network, such as the replacement of a routing protocol or the modification of its configuration, can improve the performance, scalability, manageability, and security of the entire network. However, such migrations are an important source of concerns for network operators as the reconfiguration campaign can lead to long and service-affecting outages.

In this paper, we propose a methodology which addresses the problem of seamlessly modifying the configuration of commonly used link-state Interior Gateway Protocols (IGP). We illustrate the benefits of our methodology by considering several migration scenarios, including the addition or the removal of routing hierarchy in an existing IGP and the replacement of one IGP with another. We prove that a strict operational ordering can guarantee that the migration will not create IP transit service outages. Although finding a safe ordering is NP-complete, we describe techniques which efficiently find such an ordering and evaluate them using both real-world and inferred ISP topologies. Finally, we describe the implementation of a provisioning system which automatically performs the migration by pushing the configurations on the routers in the appropriate order, while monitoring the entire migration process.

Categories and Subject Descriptors: C.2.3 [Computer-Communication Networks]: Network Operations

General Terms: Algorithms, Management, Reliability

Keywords: Interior Gateway Protocol (IGP), configuration, migration, summarization, design guidelines

As the network grows or when new services have to be deployed, network operators often need to perform large-scale IGP reconfiguration [1]. Migrating an IGP is a complex process since all the routers have to be reconfigured in a proper manner. Simple solutions like restarting the network with the new configurations do not work since most of the networks carry traffic 24/7. Therefore, IGP migrations have to be performed gradually, while the network is running. Such operations can lead to significant traffic losses if they are not handled with care. Unfortunately, network operators typically lack appropriate tools and techniques to seamlessly perform large, highly distributed changes to the configuration of their networks. They also experience difficulties in understanding what is happening during a migration since complex interactions may arise between upgraded and non-upgraded routers. Consequently, as confirmed by many private communications with operators, large-scale IGP migrations are often avoided until they are absolutely necessary, thus hampering network evolvability and innovation.

Most of the time, network operators target three aspects of the IGP when they perform large-scale migrations. First, they may want to replace the current protocol with another. For instance, several operators have switched from OSPF to IS-IS because IS-IS is known to be more secure against control-plane attacks [2, 3]. Operators may also want to migrate to an IGP that is not dependent on the address family (e.g., OSPFv3, IS-IS) in order to run only one IGP to route both IPv4 and IPv6 traffic [4, 3], or to change IGP in order to integrate new equipments which are not compliant with the adopted one [5]. Second, when the number of routers exceeds a certain critical mass, operators often introduce a hierarchy within their IGP to limit the control-plane

Our last SIGCOMM paper (2021)

Snowcap: Synthesizing Network-Wide Configuration Updates

Tibor Schneider
ETH Zurich, Switzerland
sctibor@ethz.ch

Rüdiger Birkner
ETH Zurich, Switzerland
rbirkner@ethz.ch

Laurent Vanbever
ETH Zurich, Switzerland
lvanbever@ethz.ch

ABSTRACT

Large-scale reconfiguration campaigns tend to be nerve-racking for network operators as they can lead to significant network downtimes, decreased performance, and policy violations. Unfortunately, existing reconfiguration frameworks often fall short in practice as they either only support a small set of reconfiguration scenarios or simply do not scale.

We address these problems with Snowcap, the first network reconfiguration framework which can synthesize configuration updates that comply with arbitrary hard and soft specifications, and involve arbitrary routing protocols. Our key contribution is an efficient search procedure which leverages counter-examples to efficiently navigate the space of configuration updates. Given a reconfiguration ordering which violates the desired specifications, our algorithm automatically identifies the problematic commands so that it can avoid this particular order in the next iteration.

We fully implemented Snowcap and extensively evaluated its scalability and effectiveness on real-world topologies and typical, large-scale reconfiguration scenarios. Even for large topologies, Snowcap finds a valid reconfiguration ordering with minimal side-effects (i.e., traffic shifts) within a few seconds at most.

CCS CONCEPTS

• **Networks** → **Network management**; **Network reliability**; *Network simulations*; • **Theory of computation** → Modal and temporal logics; Logic and verification;

KEYWORDS

Network analysis, Configuration, Migration

ACM Reference Format:
Tibor Schneider, Rüdiger Birkner, and Laurent Vanbever. 2021. **Snow-**

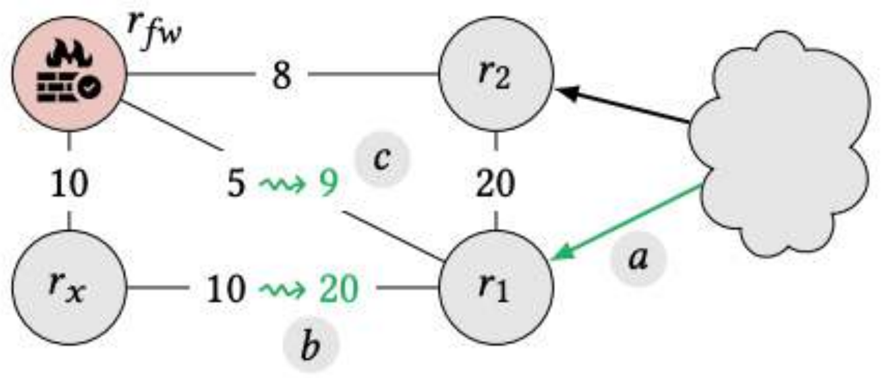


Figure 1: This scenario consists of adding an eBGP session (a) and adapting two link weights: (b) and (c), while: *(i)* ensuring traffic from r_x always flows via r_{fw}; and *(ii)* minimizing traffic shifts. Two orderings achieve both goals: (b c a) and (c b a).

1 INTRODUCTION

Network operators reconfigure their network literally every day [17, 27, 39, 40, 45]. In a Tier-1 ISP for example, network operators modify their BGP configurations up to ≈20 times per day on average [45].

While most of these reconfigurations are small (e.g., adding a new BGP session), a non-negligible fraction is large-scale. Common examples include switching routing protocols (e.g., from OSPF to IS-IS [19]), adopting a more scalable routing organization (e.g., route reflection [37]), or absorbing another network [23]. As an illustration, Google's data center networks have undergone no less than 5 large-scale configuration changes within the last decade [36].

Small or large, network reconfigurations consist in modifying the configuration of one or more network devices. Due to the distributed nature of networks, applying all reconfiguration commands atomically—on all devices—is impossible. Instead, the network necessarily transitions through a series of intermediate configurations, each of which inducing possibly distinct routing and forwarding states. Doing so the network might temporarily violate important

Have we just come
full circle?

SIGCOMM 2011

Seamless Network-Wide IGP Migrations

Laurent Vanbever*, Stefano Vissicchio†, Cristel Pelsser‡, Pierre Francois*, Olivier Bonaventure*

* Université catholique de Louvain † Roma Tre University ‡ Internet Initiative Japan
*{laurent.vanbever, pierre.francois, olivier.bonaventure} @uclouvain.be
†vissicch@dia.uniroma3.it ‡cristel@iij.ad.jp

ABSTRACT

Network-wide migrations of a running network, such as the replacement of a routing protocol or the modification of its configuration, can improve the performance, scalability, manageability, and security of the entire network. However, such migrations are an important source of concerns for network operators as the reconfiguration campaign can lead to long and service-affecting outages.

In this paper, we propose a methodology which addresses the problem of seamlessly modifying the configuration of commonly used link-state Interior Gateway Protocols (IGP). We illustrate the benefits of our methodology by considering several migration scenarios, including the addition or the removal of routing hierarchy in an existing IGP and the replacement of one IGP with another. We prove that a strict operational ordering can guarantee that the migration will not create IP transit service outages. Although finding a safe ordering is NP-complete, we describe techniques which efficiently find such an ordering and evaluate them using both real-world and inferred ISP topologies. Finally, we describe the implementation of a provisioning system which automatically performs the migration by pushing the configurations on the routers in the appropriate order, while monitoring the entire migration process.

Categories and Subject Descriptors: C.2.3 [Computer-Communication Networks]: Network Operations

General Terms: Algorithms, Management, Reliability

Keywords: Interior Gateway Protocol (IGP), configuration, migration, summarization, design guidelines

1. INTRODUCTION

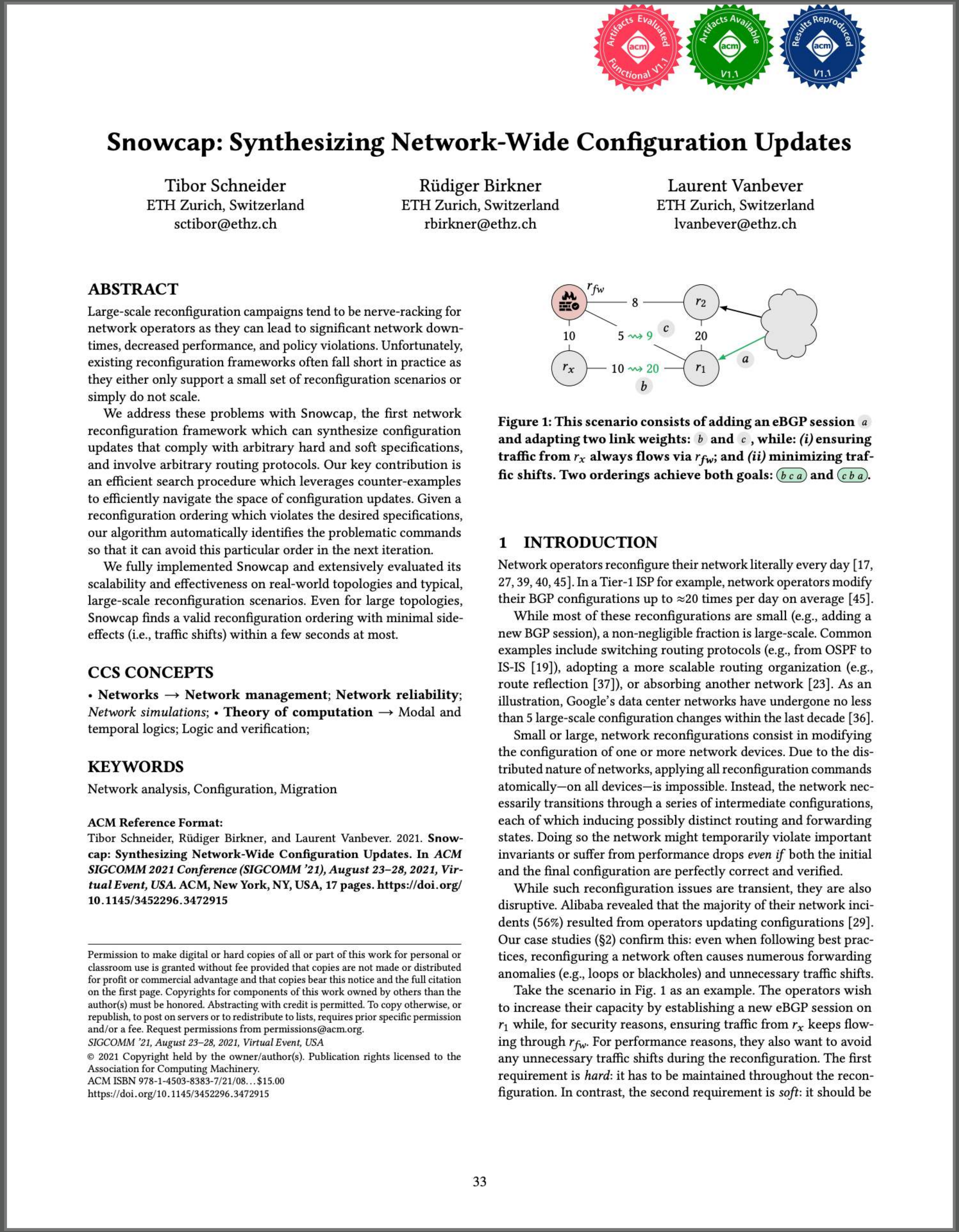
Among all network routing protocols, link-state Interior Gateway Protocols (IGPs), like IS-IS and OSPF, play a critical role. Indeed, an IGP enables end-to-end reachability between any pair of routers within the network of an Autonomous System (AS). Many other routing protocols, like BGP, LDP or PIM, also rely on an IGP to properly work. As the network grows or when new services have to be deployed, network operators often need to perform large-scale IGP reconfiguration [1]. Migrating an IGP is a complex process since all the routers have to be reconfigured in a proper manner. Simple solutions like restarting the network with the new configurations do not work since most of the networks carry traffic 24/7. Therefore, IGP migrations have to be performed gradually, while the network is running. Such operations can lead to significant traffic losses if they are not handled with care. Unfortunately, network operators typically lack appropriate tools and techniques to seamlessly perform large, highly distributed changes to the configuration of their networks. They also experience difficulties in understanding what is happening during a migration since complex interactions may arise between upgraded and non-upgraded routers. Consequently, as confirmed by many private communications with operators, large-scale IGP migrations are often avoided until they are absolutely necessary, thus hampering network evolvability and innovation.

Most of the time, network operators target three aspects of the IGP when they perform large-scale migrations. First, they may want to replace the current protocol with another. For instance, several operators have switched from OSPF to IS-IS because IS-IS is known to be more secure against control-plane attacks [2, 3]. Operators may also want to migrate to an IGP that is not dependent on the address family (e.g., OSPFv3, IS-IS) in order to run only one IGP to route both IPv4 and IPv6 traffic [4, 3], or to change IGP in order to integrate new equipments which are not compliant with the adopted one [5]. Second, when the number of routers exceeds a certain critical mass, operators often introduce a hierarchy within their IGP to limit the control-plane stress [6, 7]. Removing a hierarchy might also be needed, for instance, to better support some traffic engineering extensions [8]. Another reason operators introduce hierarchy is to have more control on route propagation by tuning the way routes are propagated from one portion of the hierarchy to another [1]. Third, network operators also modify the way the IGP learns or announces the prefixes by introducing or removing route summarization. Route summarization is an efficient way to reduce the number of entries in the routing tables of the routers as IGP networks can currently track as many as 10,000 prefixes [9]. Route summarization also helps improving the stability by limiting the visibility of local events. Actually, some IGP migrations combine several of these scenarios, such as the migration from a hierarchical OSPF to a flat IS-IS [2]. There have also been cases where, after having performed a migration, the network no

SIGCOMM'11, August 15–19, 2011, Toronto, Ontario, Canada.
Copyright 2011 ACM 978-1-4503-0797-0/11/08 ...$10.00.

vs

SIGCOMM 2021



Snowcap: Synthesizing Network-Wide Configuration Updates

Tibor Schneider, ETH Zurich, Switzerland, sctibor@ethz.ch
Rüdiger Birkner, ETH Zurich, Switzerland, rbirkner@ethz.ch
Laurent Vanbever, ETH Zurich, Switzerland, lvanbever@ethz.ch

ABSTRACT

Large-scale reconfiguration campaigns tend to be nerve-racking for network operators as they can lead to significant network downtimes, decreased performance, and policy violations. Unfortunately, existing reconfiguration frameworks often fall short in practice as they either only support a small set of reconfiguration scenarios or simply do not scale.

We address these problems with Snowcap, the first network reconfiguration framework which can synthesize configuration updates that comply with arbitrary hard and soft specifications, and involve arbitrary routing protocols. Our key contribution is an efficient search procedure which leverages counter-examples to efficiently navigate the space of configuration updates. Given a reconfiguration ordering which violates the desired specifications, our algorithm automatically identifies the problematic commands so that it can avoid this particular order in the next iteration.

We fully implemented Snowcap and extensively evaluated its scalability and effectiveness on real-world topologies and typical, large-scale reconfiguration scenarios. Even for large topologies, Snowcap finds a valid reconfiguration ordering with minimal side-effects (i.e., traffic shifts) within a few seconds at most.

CCS CONCEPTS

• **Networks** → **Network management**; **Network reliability**; *Network simulations*; • **Theory of computation** → Modal and temporal logics; Logic and verification;

KEYWORDS

Network analysis, Configuration, Migration

ACM Reference Format:
Tibor Schneider, Rüdiger Birkner, and Laurent Vanbever. 2021. **Snowcap: Synthesizing Network-Wide Configuration Updates. In *ACM SIGCOMM 2021 Conference (SIGCOMM '21), August 23–28, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3452296.3472915**

Figure 1: This scenario consists of adding an eBGP session (a) and adapting two link weights: (b) and (c), while: *(i)* ensuring traffic from r_x always flows via r_{fw}; and *(ii)* minimizing traffic shifts. Two orderings achieve both goals: (b c a) and (c b a).

1 INTRODUCTION

Network operators reconfigure their network literally every day [17, 27, 39, 40, 45]. In a Tier-1 ISP for example, network operators modify their BGP configurations up to ≈20 times per day on average [45].

While most of these reconfigurations are small (e.g., adding a new BGP session), a non-negligible fraction is large-scale. Common examples include switching routing protocols (e.g., from OSPF to IS-IS [19]), adopting a more scalable routing organization (e.g., route reflection [37]), or absorbing another network [23]. As an illustration, Google's data center networks have undergone no less than 5 large-scale configuration changes within the last decade [36].

Small or large, network reconfigurations consist in modifying the configuration of one or more network devices. Due to the distributed nature of networks, applying all reconfiguration commands atomically—on all devices—is impossible. Instead, the network necessarily transitions through a series of intermediate configurations, each of which inducing possibly distinct routing and forwarding states. Doing so the network might temporarily violate important invariants or suffer from performance drops *even if* both the initial and the final configuration are perfectly correct and verified.

While such reconfiguration issues are transient, they are also disruptive. Alibaba revealed that the majority of their network incidents (56%) resulted from operators updating configurations [29]. Our case studies (§2) confirm this: even when following best practices, reconfiguring a network often causes numerous forwarding anomalies (e.g., loops or blackholes) and unnecessary traffic shifts.

Take the scenario in Fig. 1 as an example. The operators wish to increase their capacity by establishing a new eBGP session on r_1 while, for security reasons, ensuring traffic from r_x keeps flowing through r_{fw}. For performance reasons, they also want to avoid any unnecessary traffic shifts during the reconfiguration. The first requirement is *hard*: it has to be maintained throughout the reconfiguration. In contrast, the second requirement is *soft*: it should be

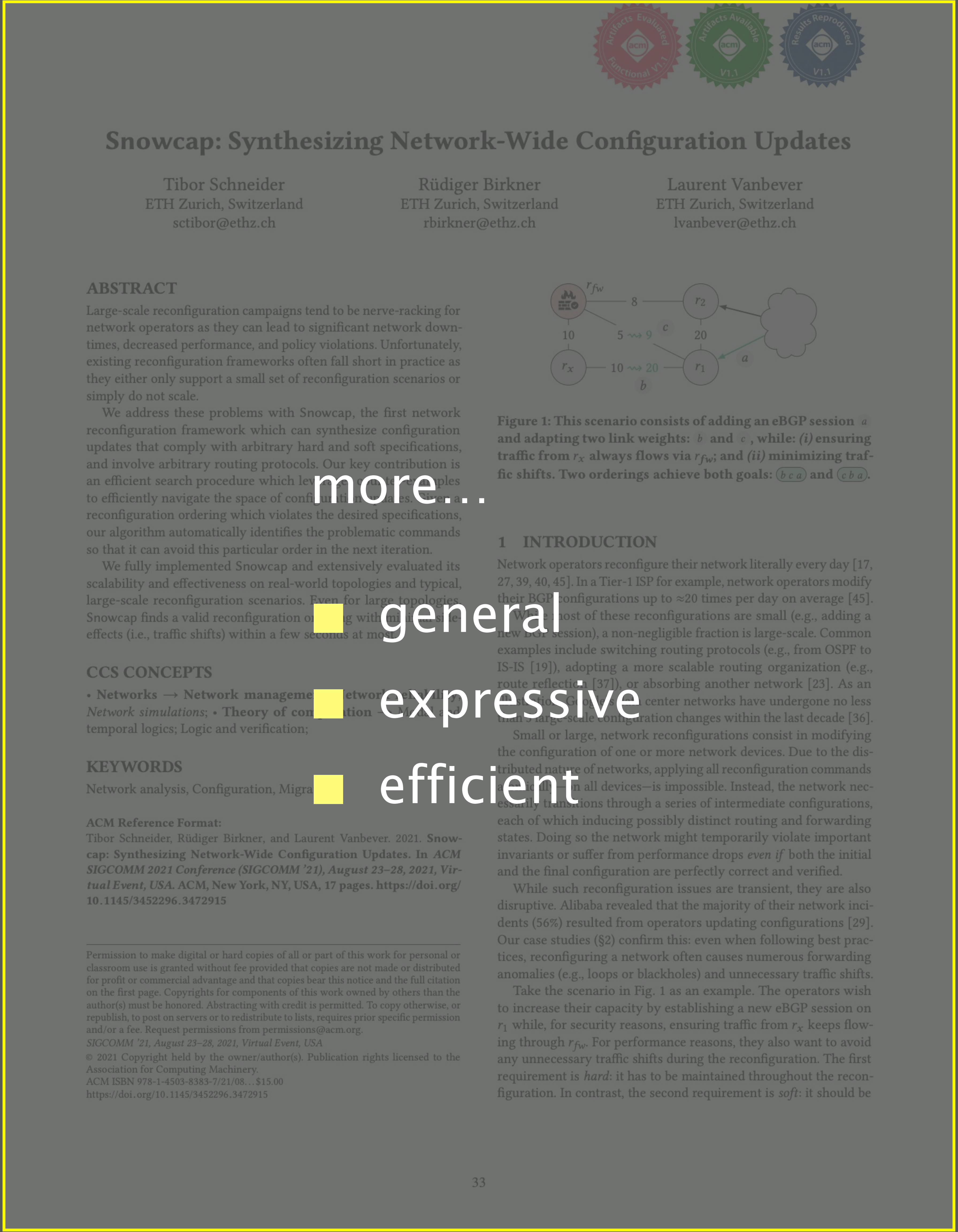
SIGCOMM '21, August 23–28, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8383-7/21/08...$15.00
https://doi.org/10.1145/3452296.3472915

Seamless Network-Wide IGP Migrations

Laurent Vanbever, Stefano Vissicchio, Cristel Pelsser, Pierre Francois, Olivier Bonaventure

vs



Snowcap: Synthesizing Network-Wide Configuration Updates

Tibor Schneider, Rüdiger Birkner, Laurent Vanbever
ETH Zurich, Switzerland

more…

- general
- expressive
- efficient

Seamless Network-Wide IGP Migrations

Laurent Vanbever, Stefano Vissicchio, Cristel Pelsser, Pierre Francois, Olivier Bonaventure

vs

Snowcap: Synthesizing Network-Wide Configuration Updates

Tibor Schneider, Rüdiger Birkner, Laurent Vanbever

more…

- general
- expressive
- efficient

reason about distributed network computations reason about

distributed network computations

Distributed computations rule over network forwarding behavior

distributed
algorithms

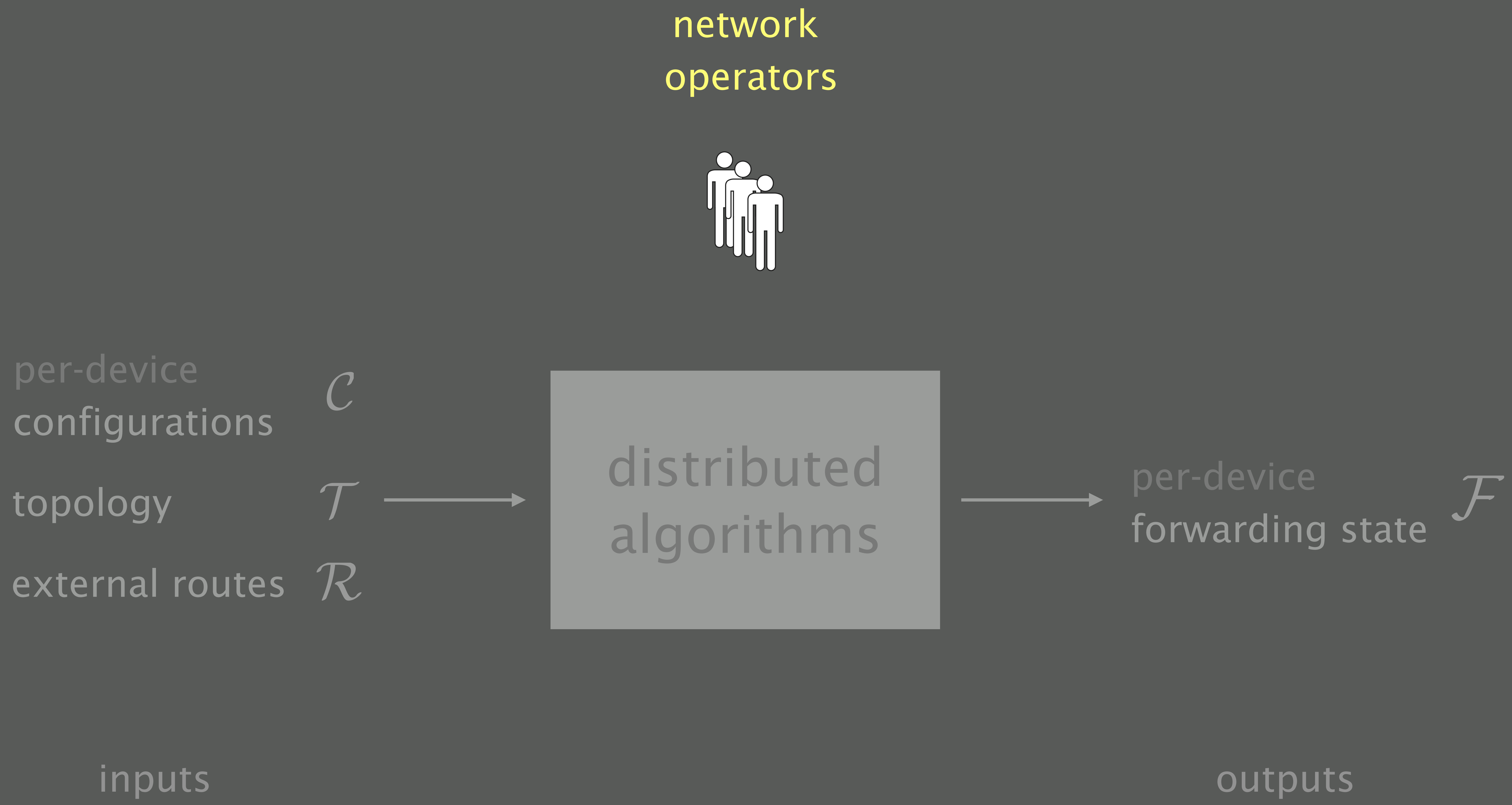
distributed algorithms

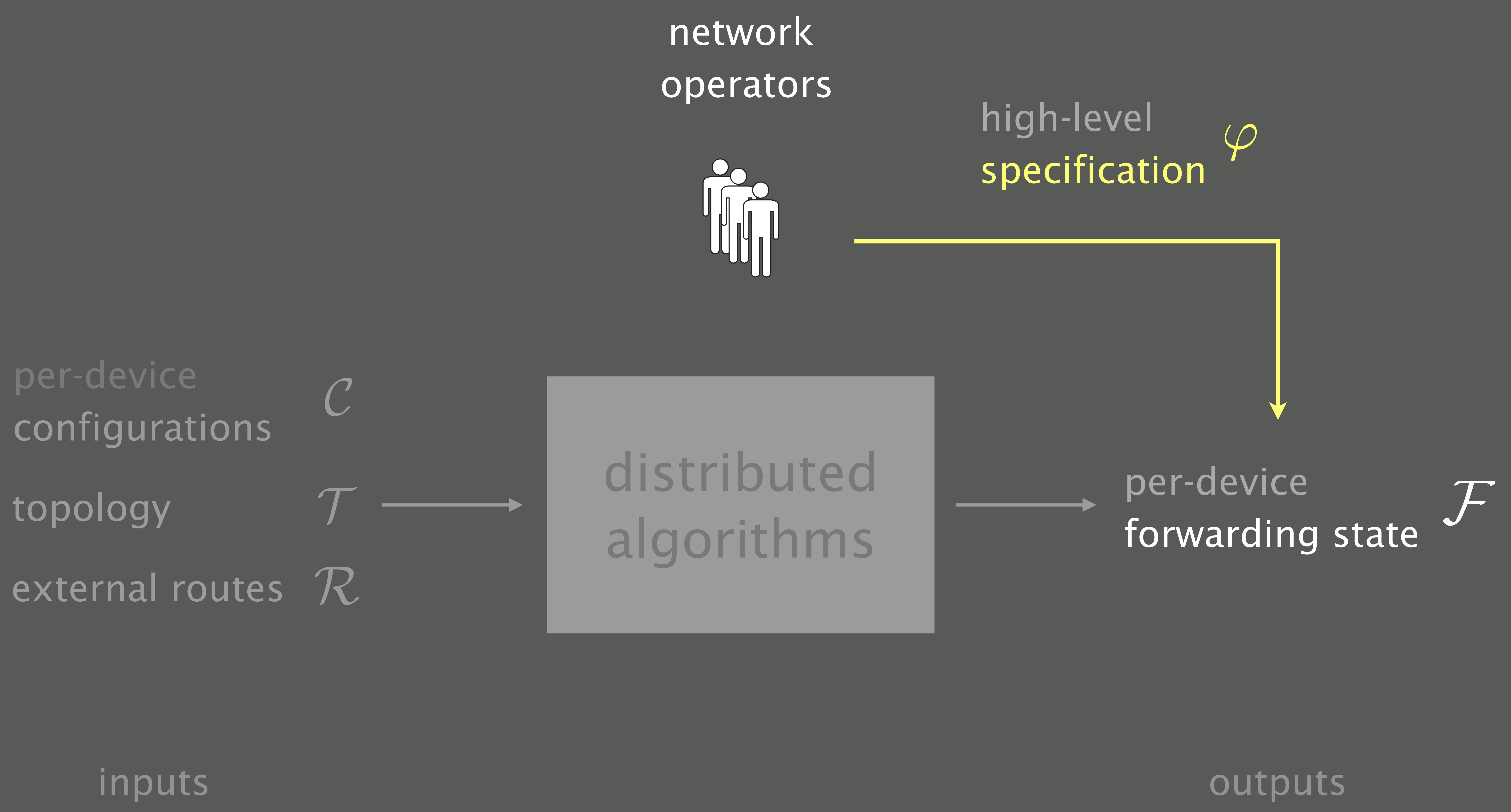
per-device

forwarding state $\mathcal{F}$

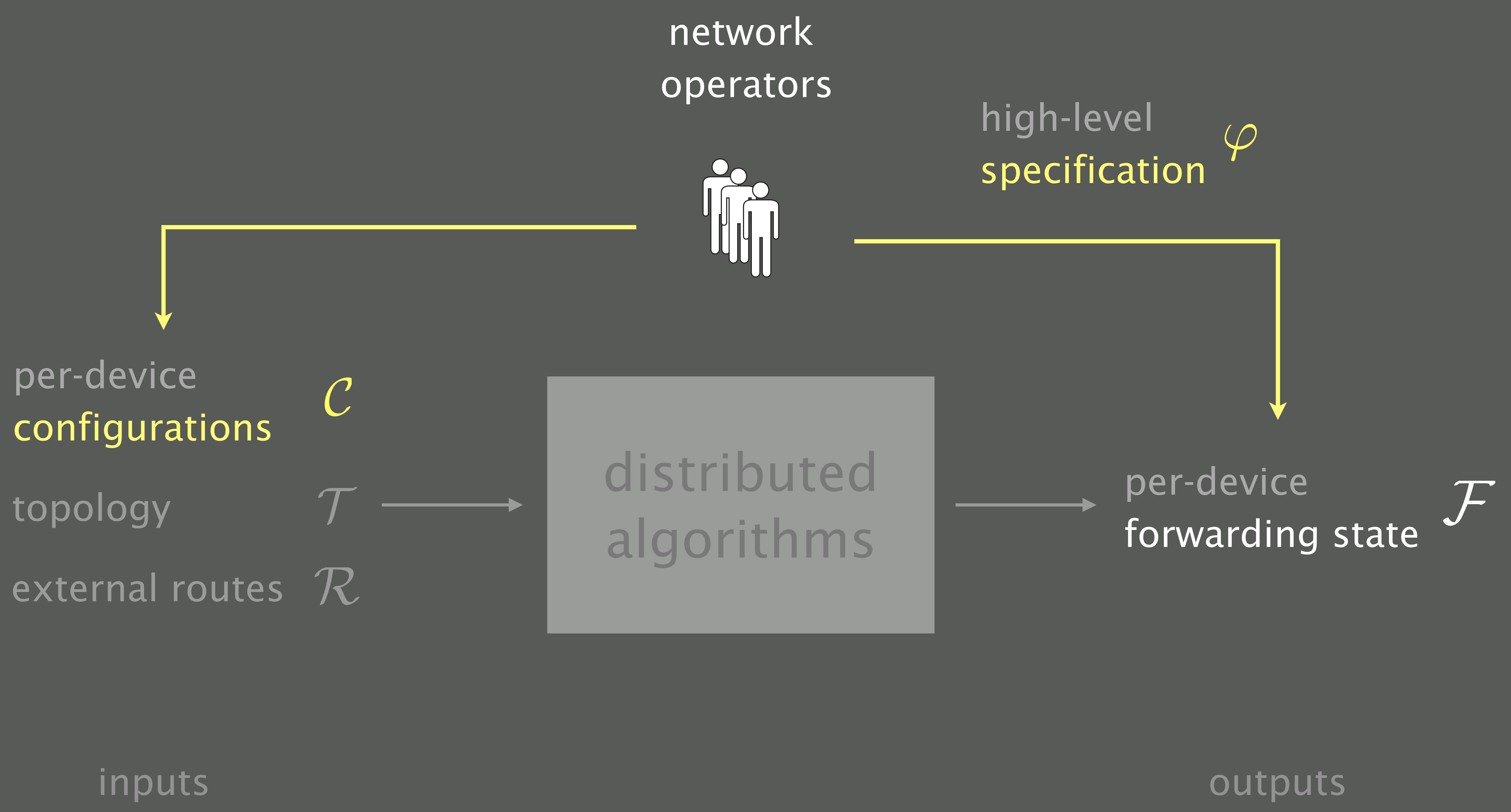
outputs


per-device
configurations
$\mathcal{C}$
topology
$\mathcal{T}$
external routes
$\mathcal{R}$
distributed
algorithms
per-device
forwarding state
$\mathcal{F}$
inputs
outputs

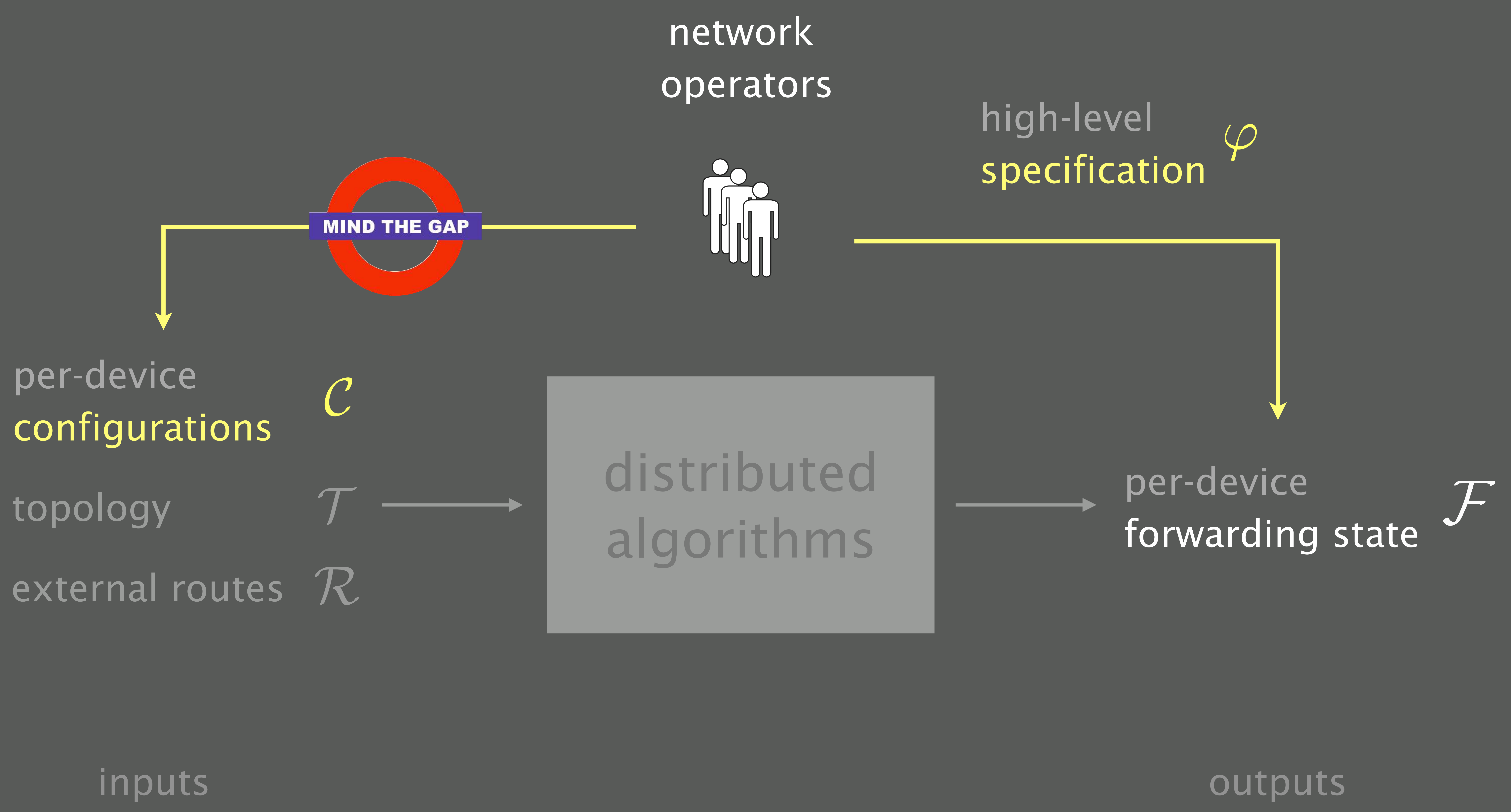

network
operators
per-device
configurations $\mathcal{C}$
topology $\mathcal{T}$
external routes $\mathcal{R}$
distributed
algorithms
per-device
forwarding state $\mathcal{F}$
inputs
outputs



network
operators
high-level
specification
φ
per-device
configurations
$\mathcal{C}$
topology
$\mathcal{T}$
external routes
$\mathcal{R}$
distributed
algorithms
per-device
forwarding state
$\mathcal{F}$
inputs
outputs



network
operators
high-level
specification
φ
per-device
configurations
$\mathcal{C}$
topology
$\mathcal{T}$
external routes
$\mathcal{R}$
distributed
algorithms
per-device
forwarding state
$\mathcal{F}$
inputs
outputs


network
operators
high-level
specification φ
MIND THE GAP
per-device
configurations $\mathcal{C}$
topology $\mathcal{T}$
external routes $\mathcal{R}$
distributed
algorithms
per-device
forwarding state $\mathcal{F}$
inputs
outputs

Facebook blames major outage ×

verdict.co.uk/outage-facebook-instagram-and-whatsapp-are-down/

Menu Search

VERDICT

Sign in

SOCIAL MEDIA AND ONLINE

Facebook blames major outage on “configuration changes”: Rivals gloat

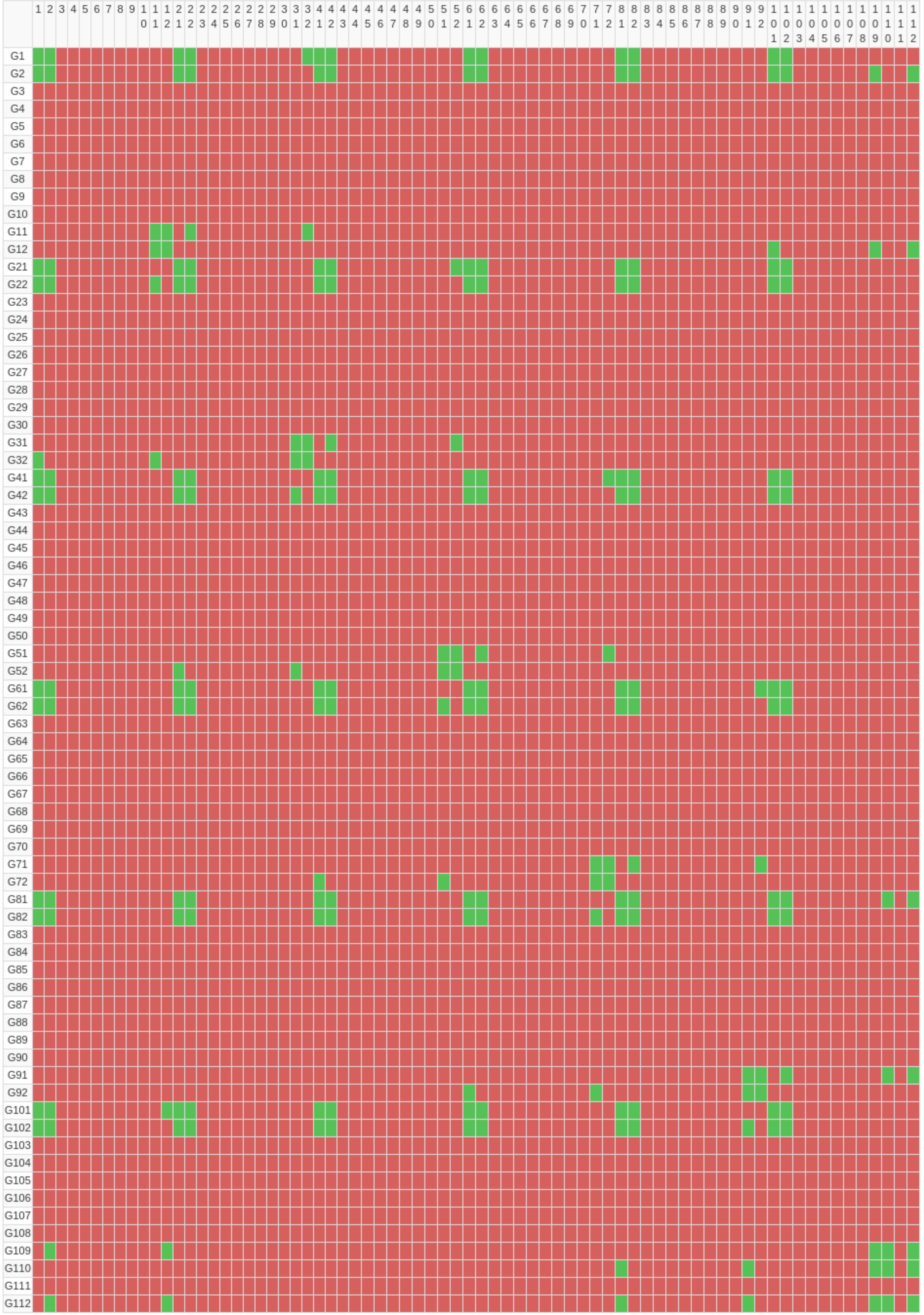
Eric Johansson | 4th October 2021 (Last Updated October 5th, 2021 11:57)

Need more proof?

Ask our students!

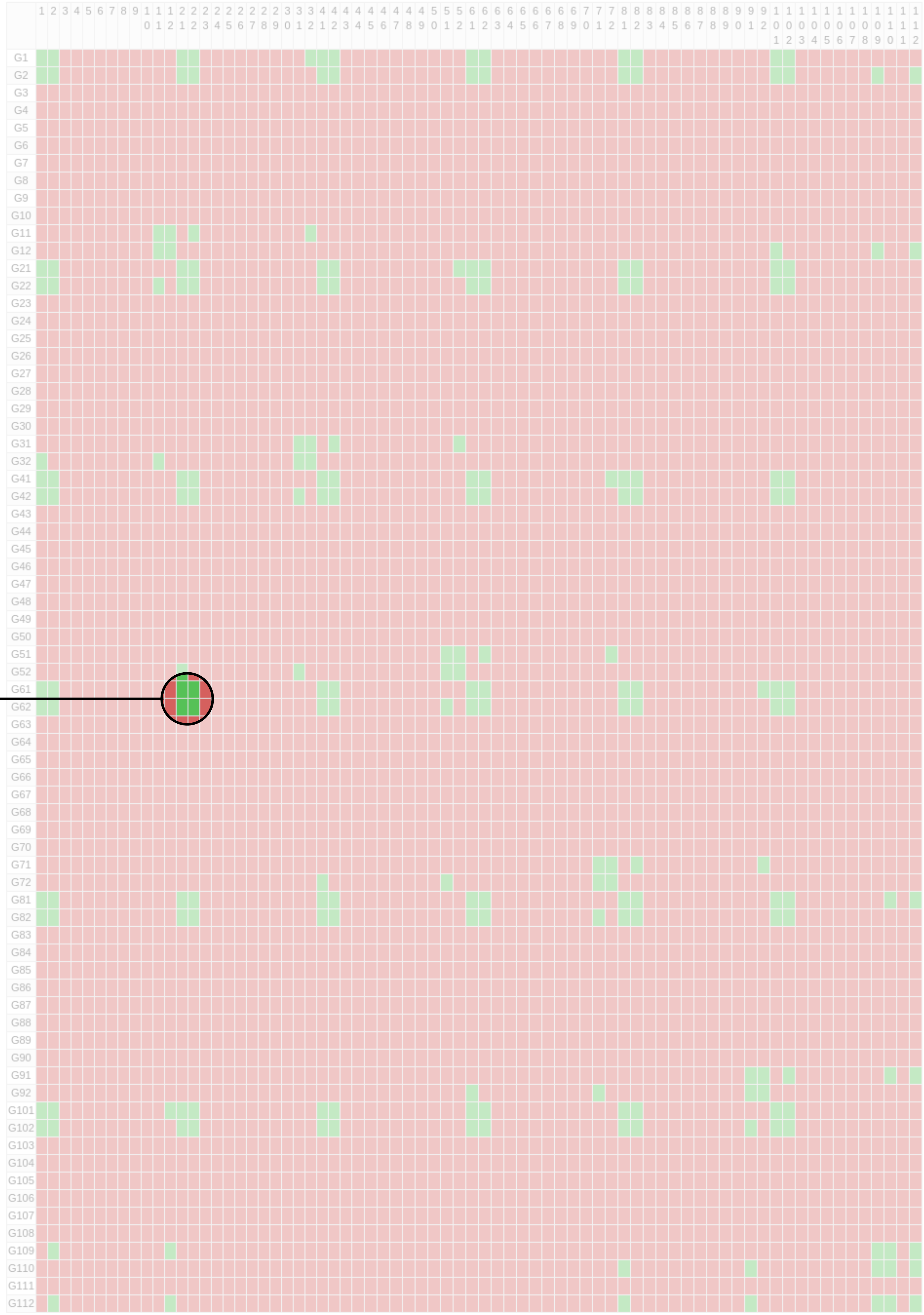
Pre-COVID Mini-Internet hackathon @ETH Zürich

Connectivity statistics (2021)



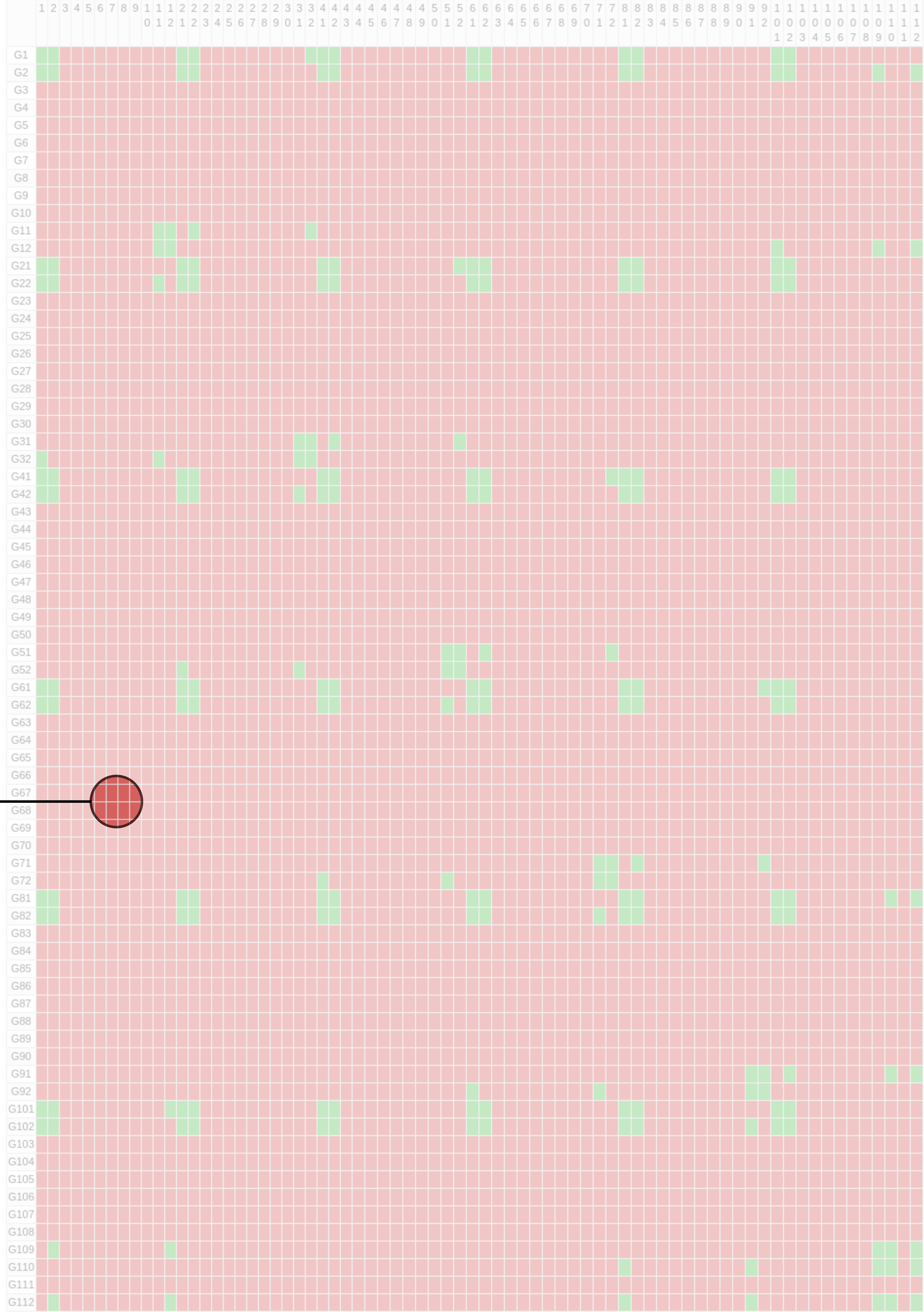
group$_i$ can reach group$_j$

there is a working path



group$_i$ cannot reach group$_j$

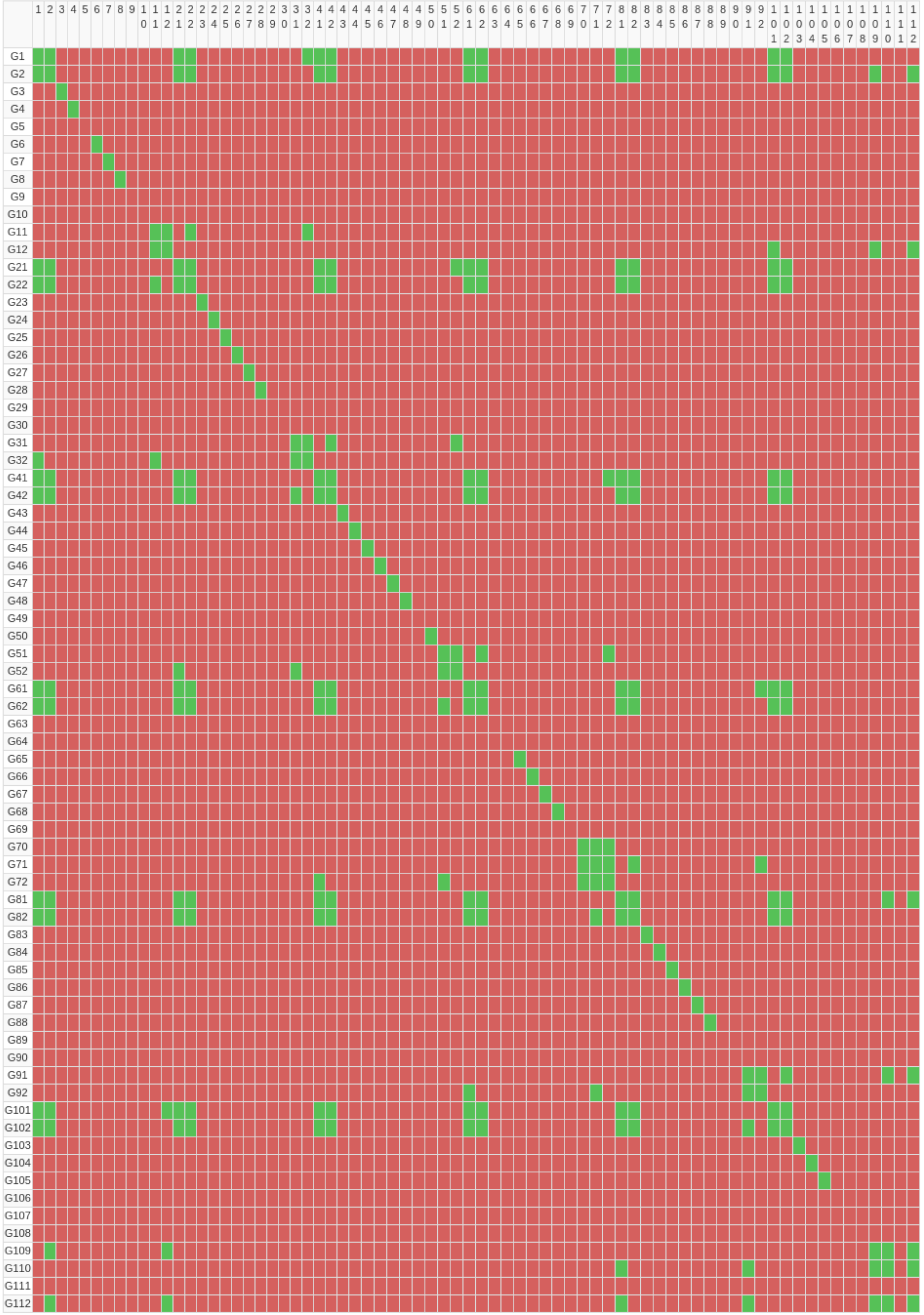
there is an outage



Connectivity statistics (2021)

initial ~10%

final



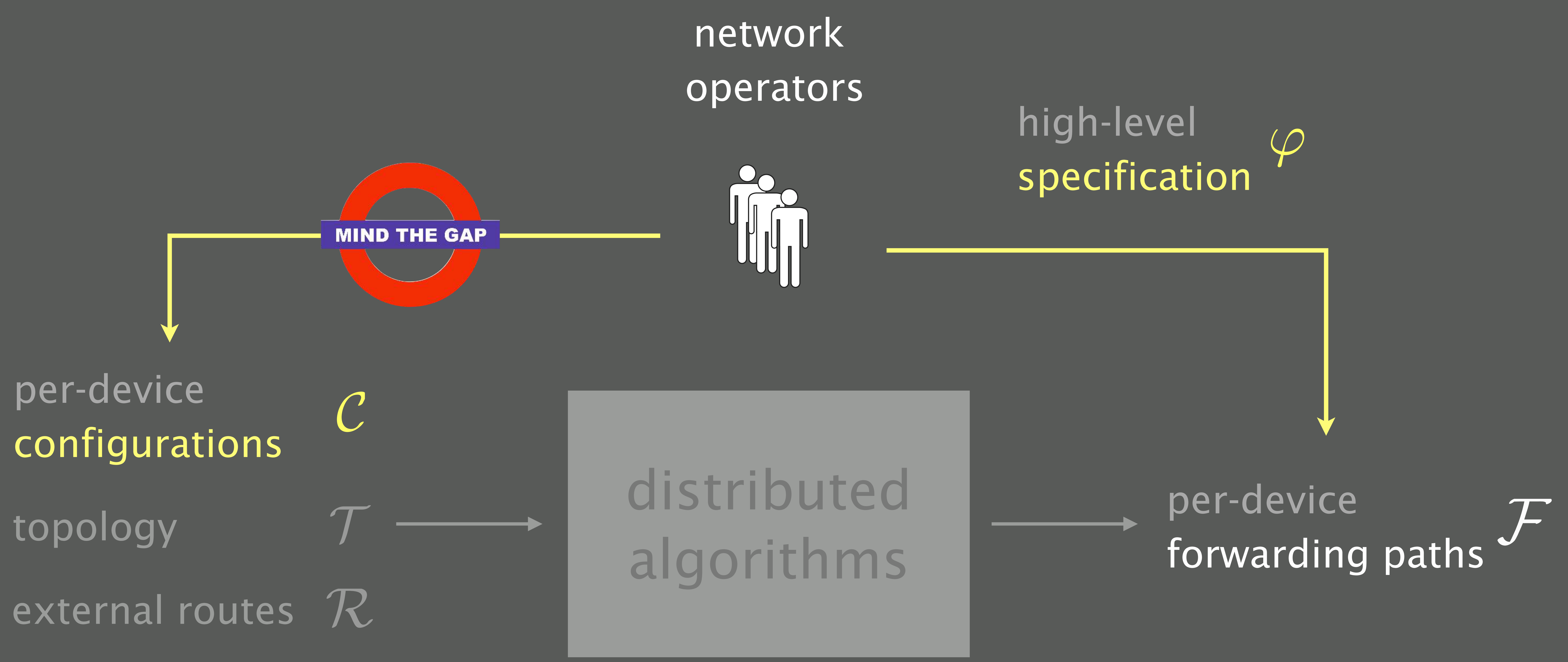
Connectivity statistics (2021)

initial ~10%

final **~98%**

highest since 2016! ◡̈


nsg-ethz/mini_internet_project



network
operators
high-level
specification φ
MIND THE GAP
per-device
configurations $\mathcal{C}$
topology $\mathcal{T}$
external routes $\mathcal{R}$
distributed
algorithms
per-device
forwarding paths $\mathcal{F}$

We've aimed at helping operators bridging this gap considering three directions

We've aimed at helping operators bridging this gap considering three directions

Verification

Synthesis

Reconfiguration

We've aimed at helping operators bridging this gap considering three directions

Given specification φ
and

Verification

Synthesis

Reconfiguration

We've aimed at helping operators bridging this gap considering three directions

Given specification φ

and

Verification

configuration $\mathcal{C}$

Synthesis

Reconfiguration

We've aimed at helping operators bridging this gap considering three directions

Given specification φ and

Return

Verification

configuration $\mathcal{C}$

Synthesis

Reconfiguration

We've aimed at helping operators bridging this gap considering three directions

	Given specification φ and	Return
Verification	configuration $\mathcal{C}$	$\mathcal{C} \models \varphi$ ✓ / ✗
Synthesis		
Reconfiguration		

We've aimed at helping operators bridging this gap considering three directions

	Given specification φ and	Return
Verification	configuration $\mathcal{C}$	$\mathcal{C} \models \varphi$ ✓ / ✗
Synthesis	$\varnothing$	$\mathcal{C} \models \varphi$
Reconfiguration		

We've aimed at helping operators bridging this gap considering three directions

	Given specification φ and	Return
Verification	configuration $\mathcal{C}$	$\mathcal{C} \models \varphi$ ✓ ✗
Synthesis	$\varnothing$	$\mathcal{C} \models \varphi$
Reconfiguration	initial and final configuration $\mathcal{C}_i$ $\mathcal{C}_f$	$\mathcal{C}_i, \mathcal{C}_a, \mathcal{C}_b, \ldots \mathcal{C}_f \models \varphi$

The three tales of (correct) network operations

1 **Verification**

going forward

2 **Synthesis**

going backward

3 **Reconfiguration**

going sideways

The three tales of (correct) network operations

1 **Verification**

going forward

Synthesis

going backward

Reconfiguration

going sideways

Probabilistic Verification of Network Configurations


Samuel Steffen


Timon Gehr


Petar Tsankov


Laurent Vanbever

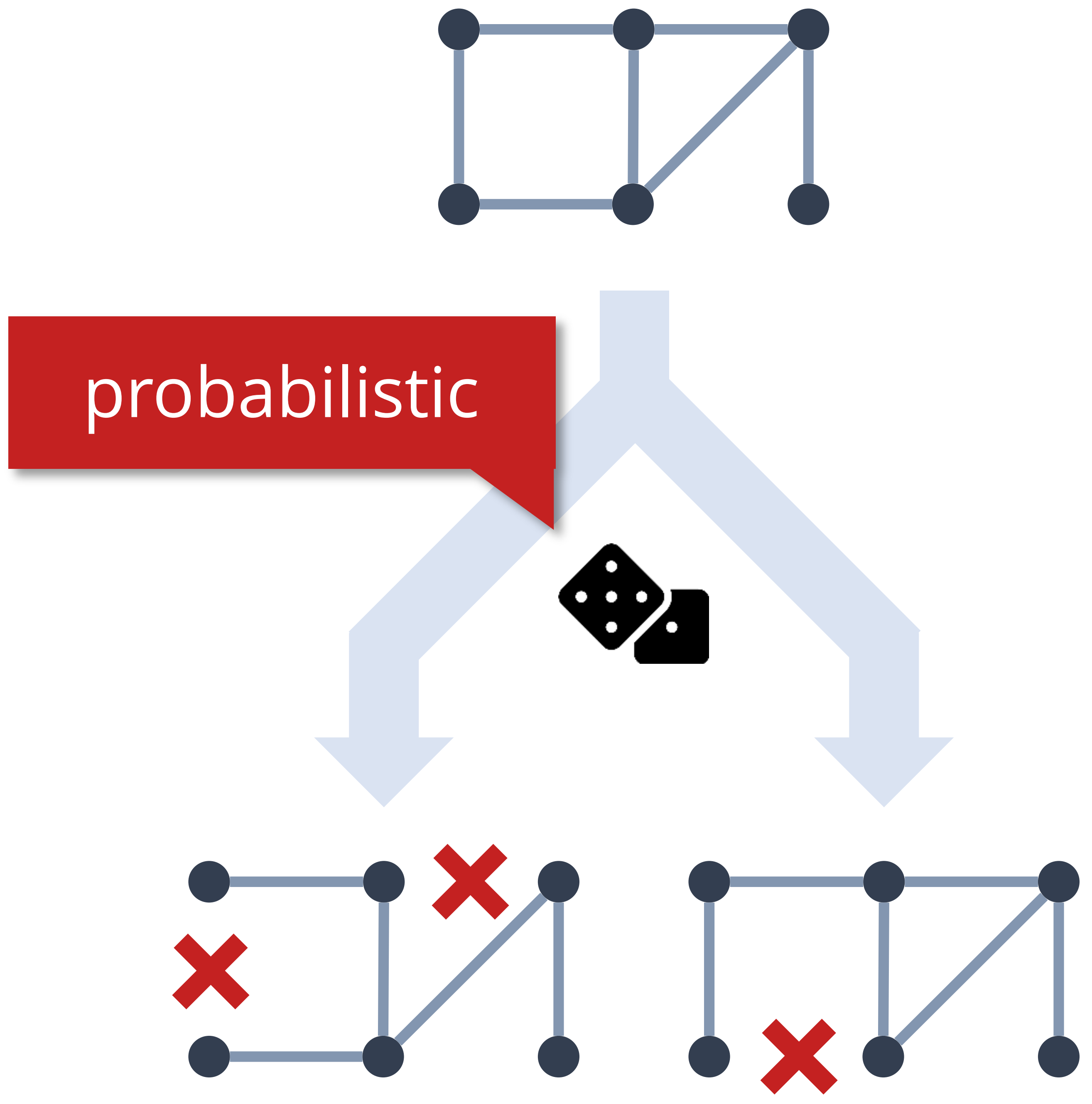

Martin Vechev

ETH zürich

Networked Systems

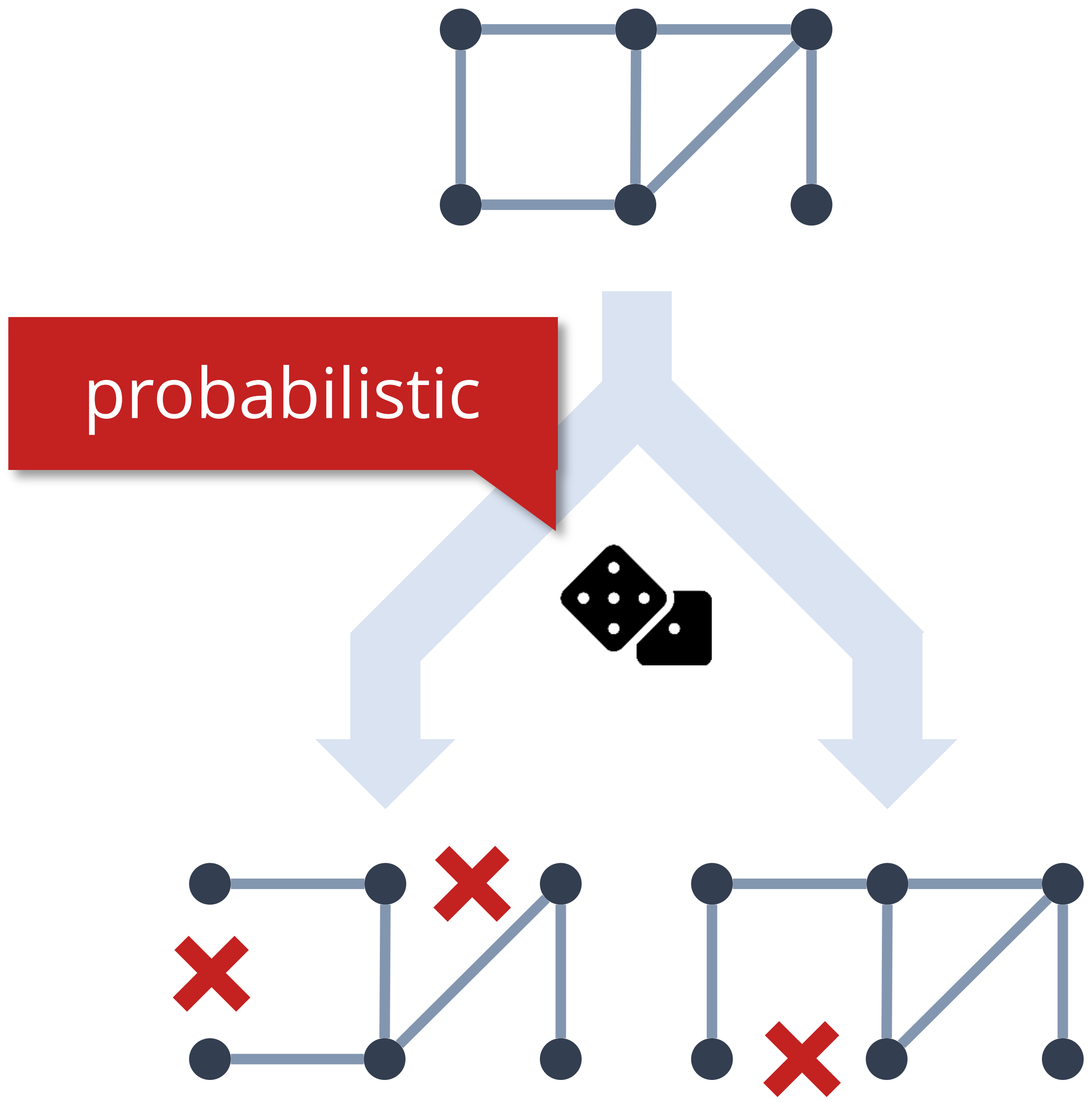
SRILAB

Probabilistic Verification



What is the *probability* of ?

Probabilistic **Verification**

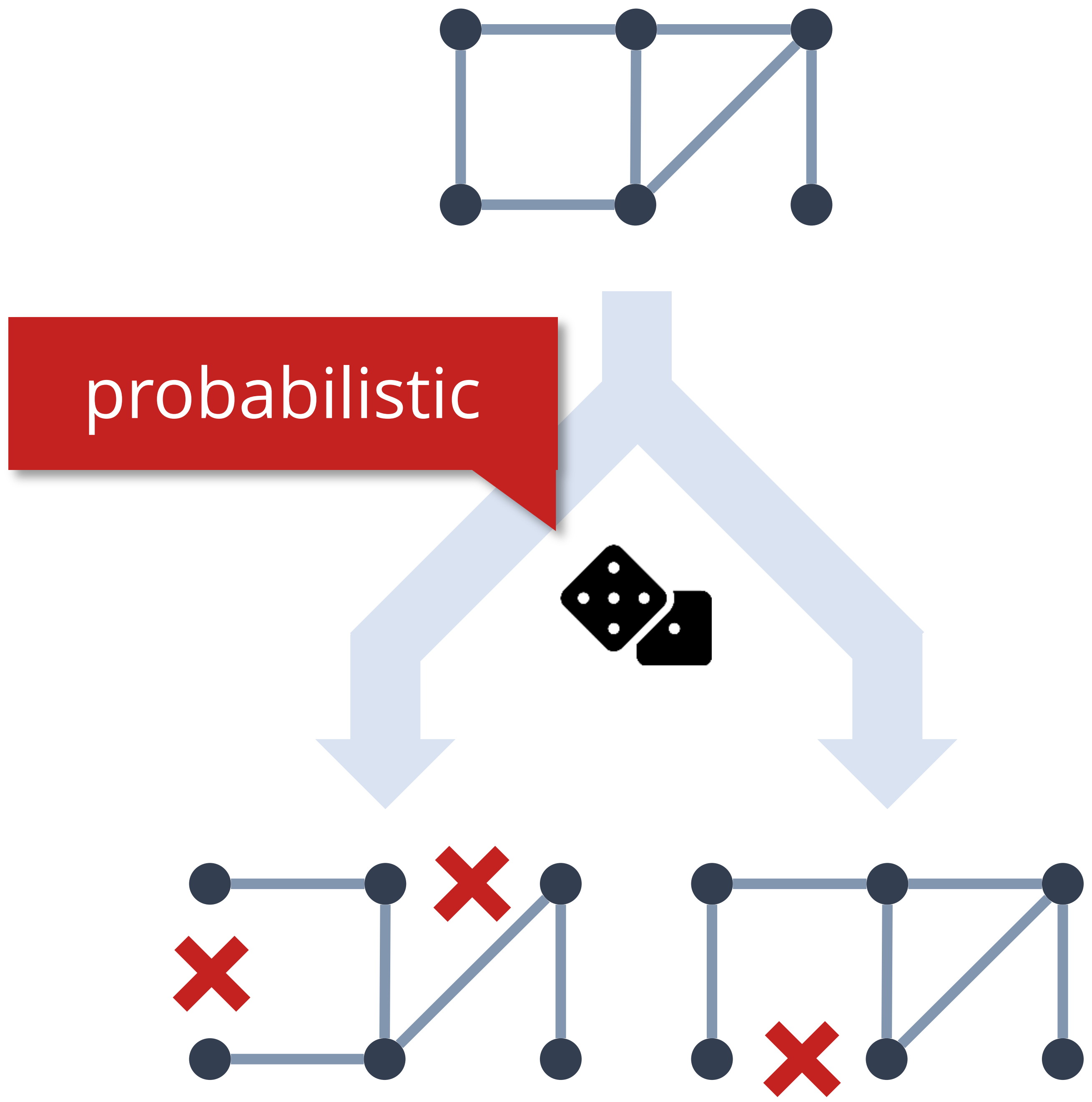


What is the *probability* of ?

Service Level Agreements (SLA)
“99.99% reachability”

Traffic Engineering
“80% load-balanced”

Probabilistic Verification



What is the *probability* of ?

high precision required

Service Level Agreements (SLA)
“99.99% reachability”

Traffic Engineering
“80% load-balanced”

Attempts: Exploring Failures

Attempts: Exploring Failures

Partial exploration

1 107 359

#scenarios for *four 9s*,
191 links, $p_{\text{link failure}} = 0.001$

Attempts: Exploring Failures

Too expensive

Partial exploration

1 107 359

#scenarios for *four 9s*,
191 links, $p_{\text{link failure}}$ = 0.001

Attempts: Exploring Failures

Partial exploration

1 107 359

#scenarios for *four 9s*,
191 links, $p_{\text{link failure}} = 0.001$

Estimation via sampling

738 M

Hoeffding, $\alpha = 0.95$

Attempts: Exploring Failures

Too expensive

Partial exploration

Estimation via sampling

1 107 359

#scenarios for *four 9s*,
191 links, $p_{\text{link failure}} = 0.001$

738 M

Hoeffding, $\alpha = 0.95$

Attempts: Exploring Failures

Too expensive

Partial exploration

1 107 359

#scenarios for *four 9s*,
191 links, $p_{\text{link failure}} = 0.001$

Estimation via sampling

738 M

Hoeffding, $\alpha = 0.95$

NetDice

1 854

≈600x reduction

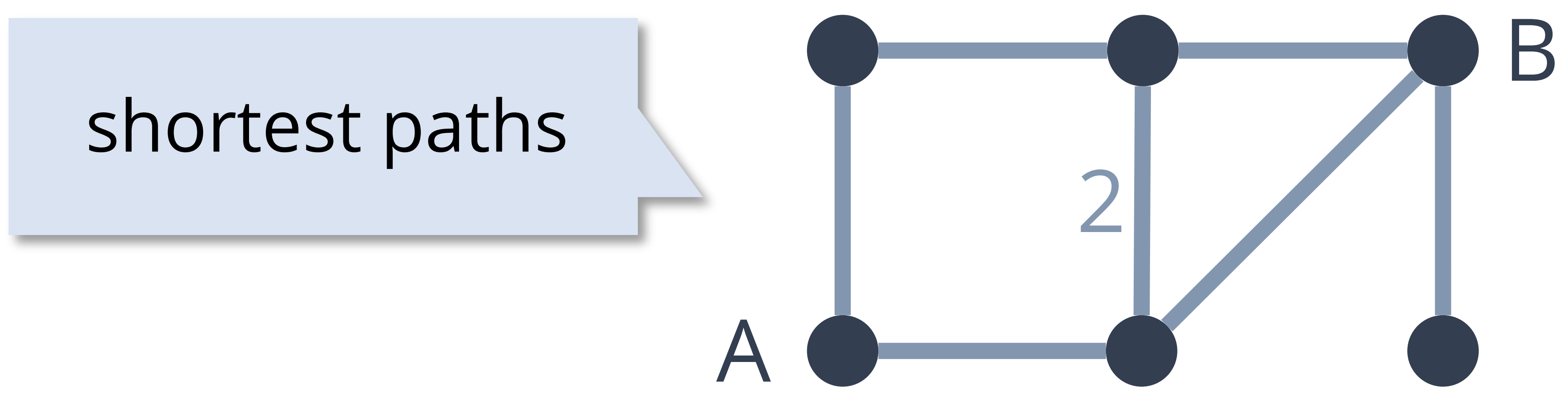
Overview

NetDice

- BGP + IGP support
- High accuracy
- Scalable

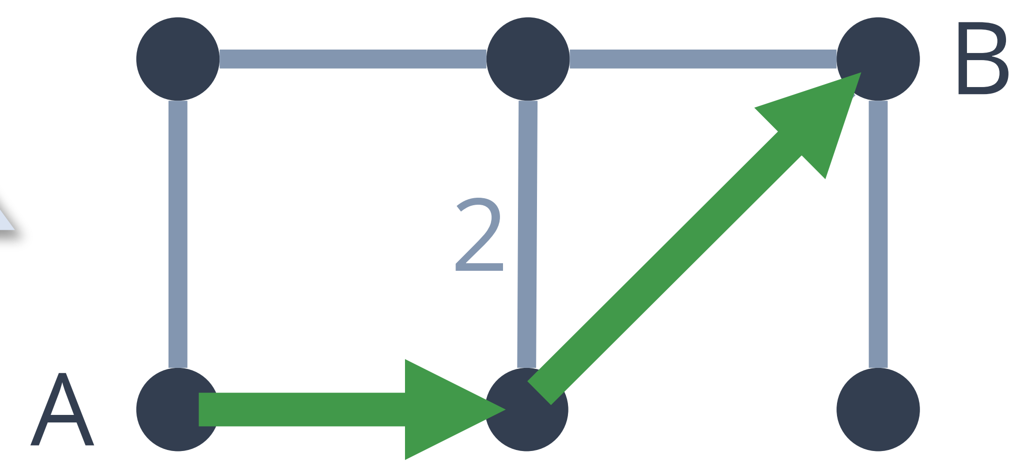
Pruning Failures

Key Idea

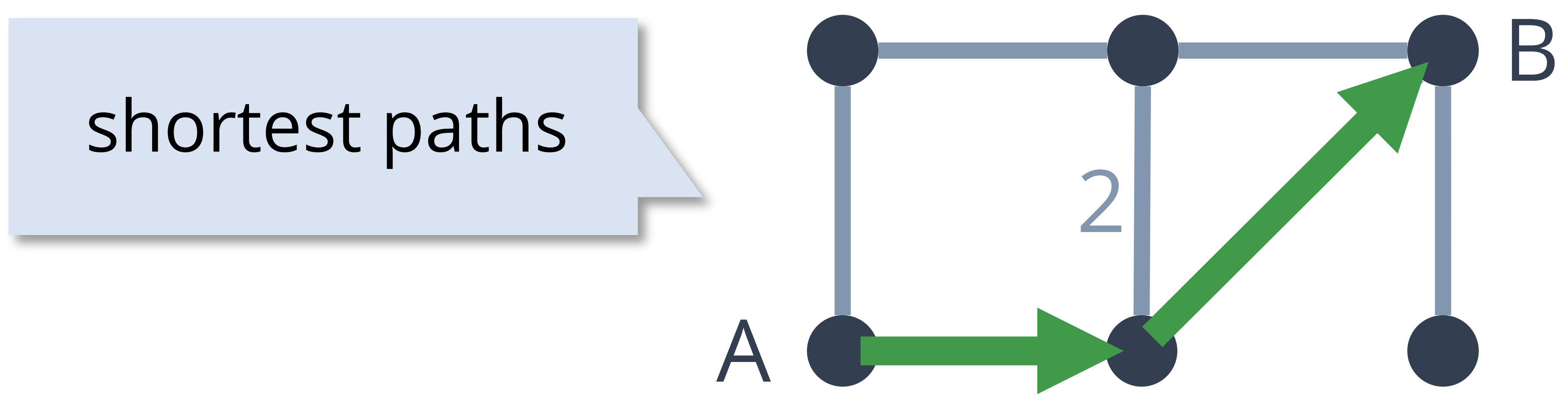


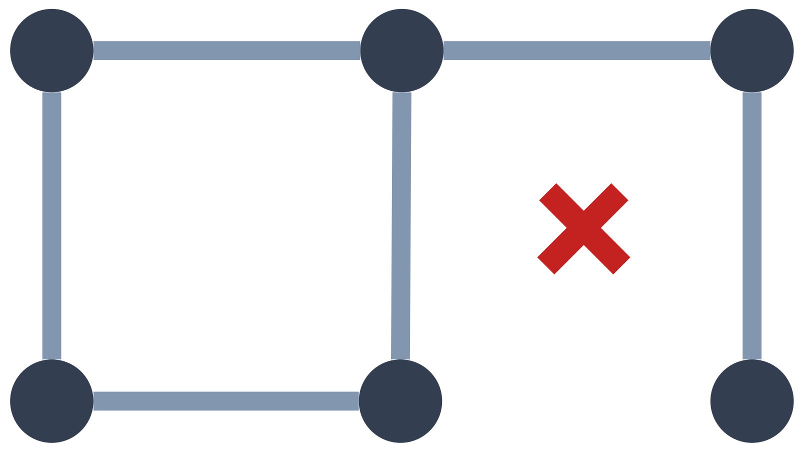
Key Idea

shortest paths



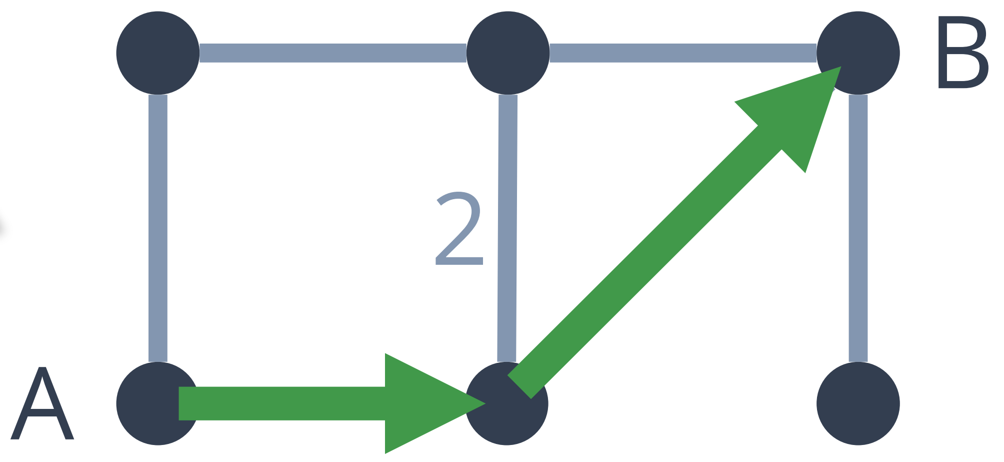
Key Idea

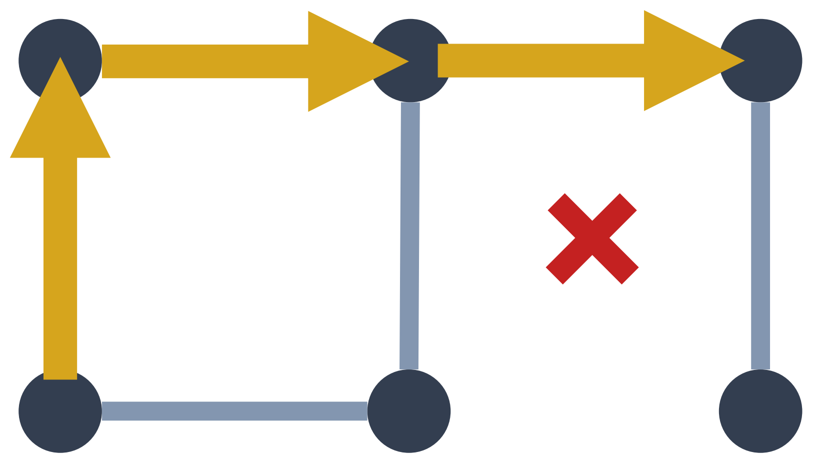




Key Idea

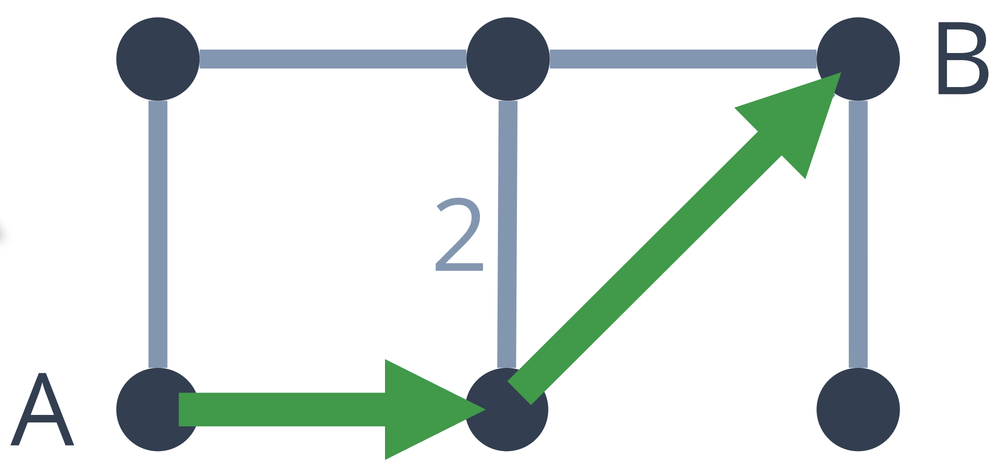
shortest paths





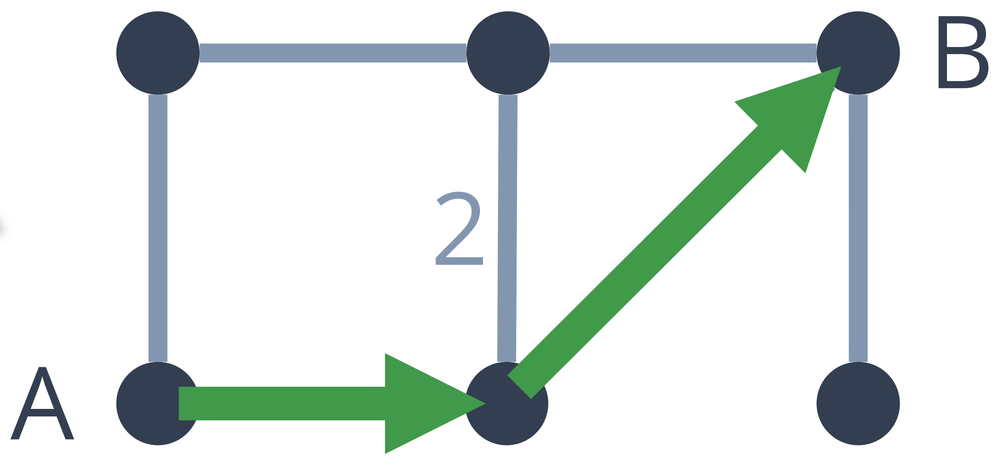
Key Idea

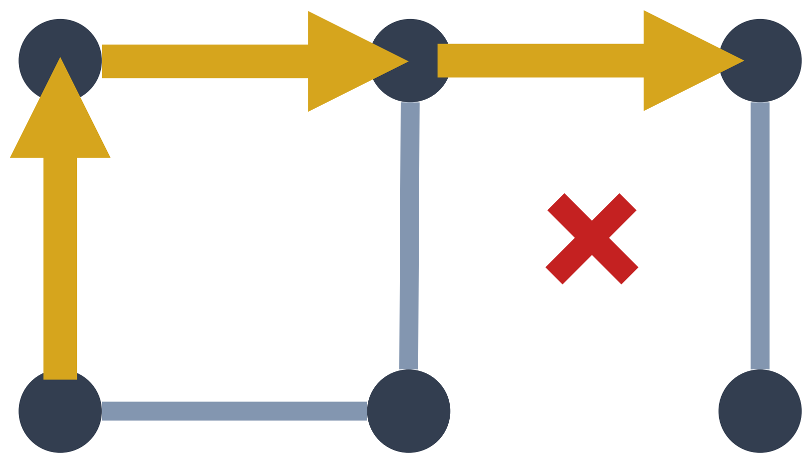
shortest paths

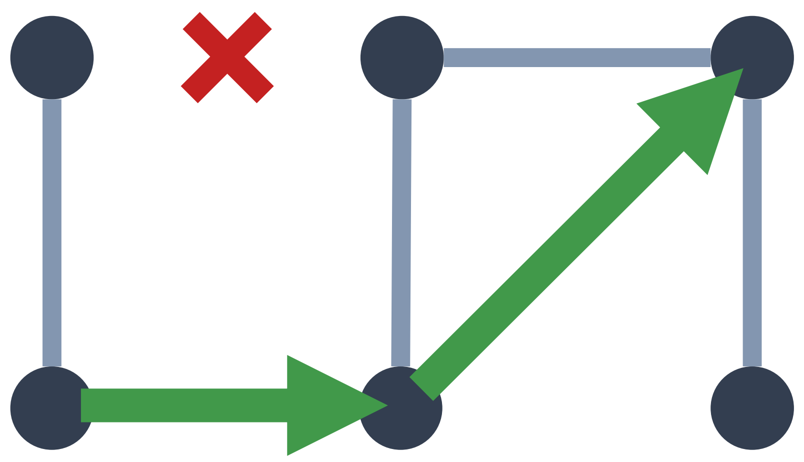


Key Idea

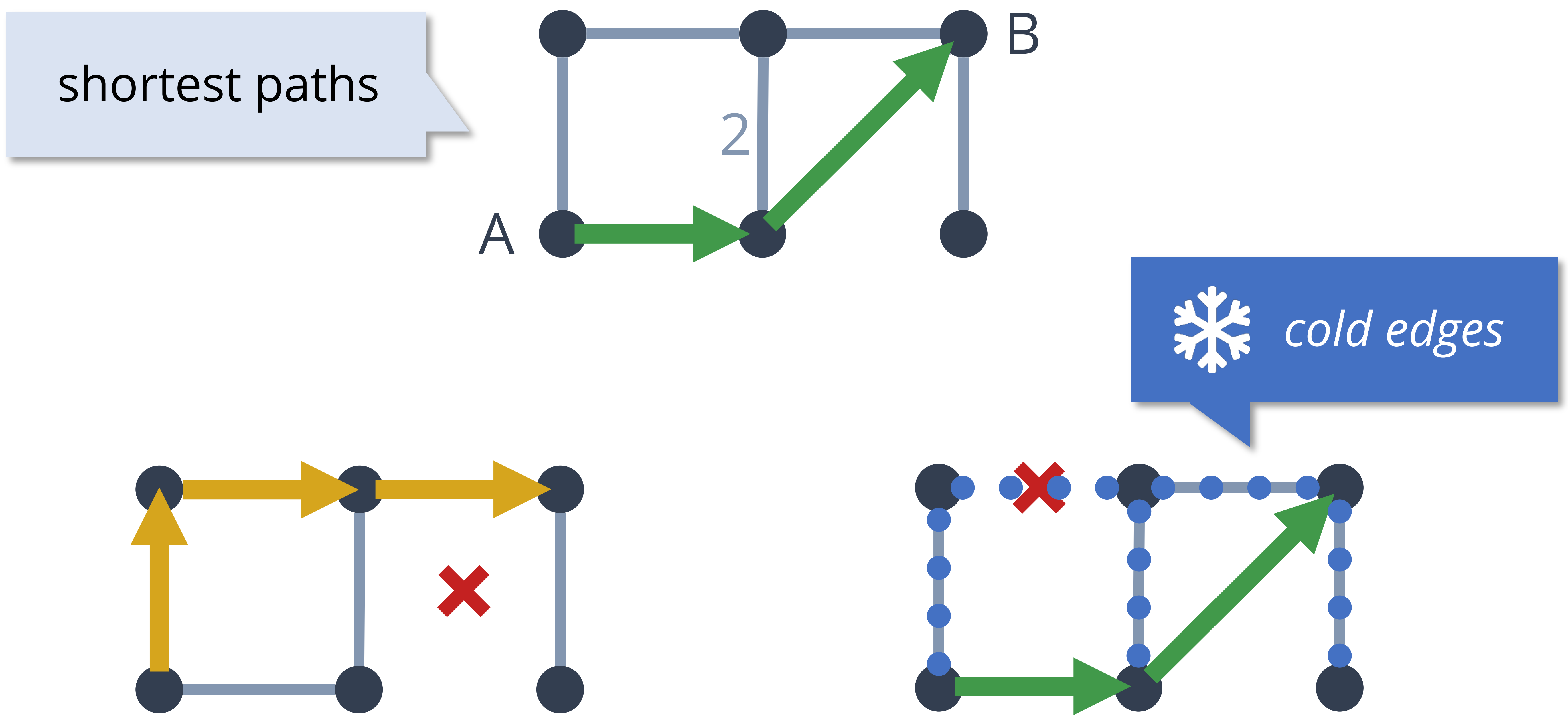
shortest paths





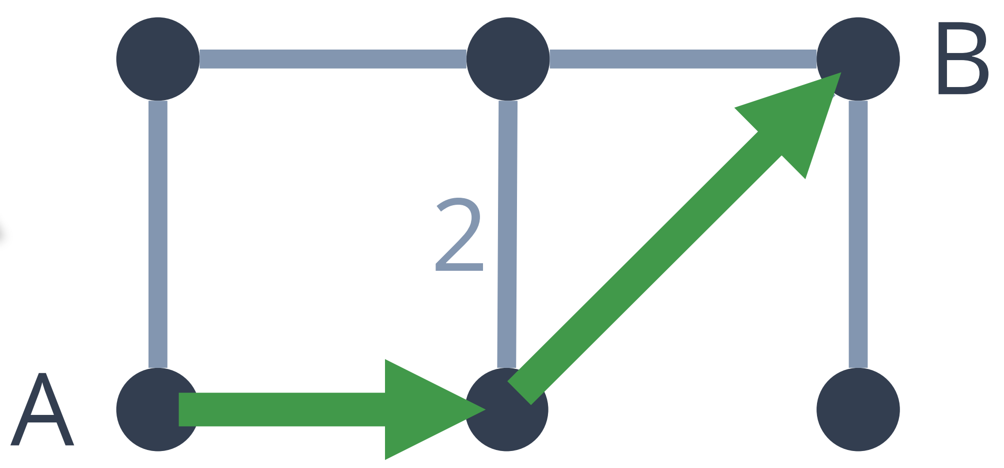


Key Idea



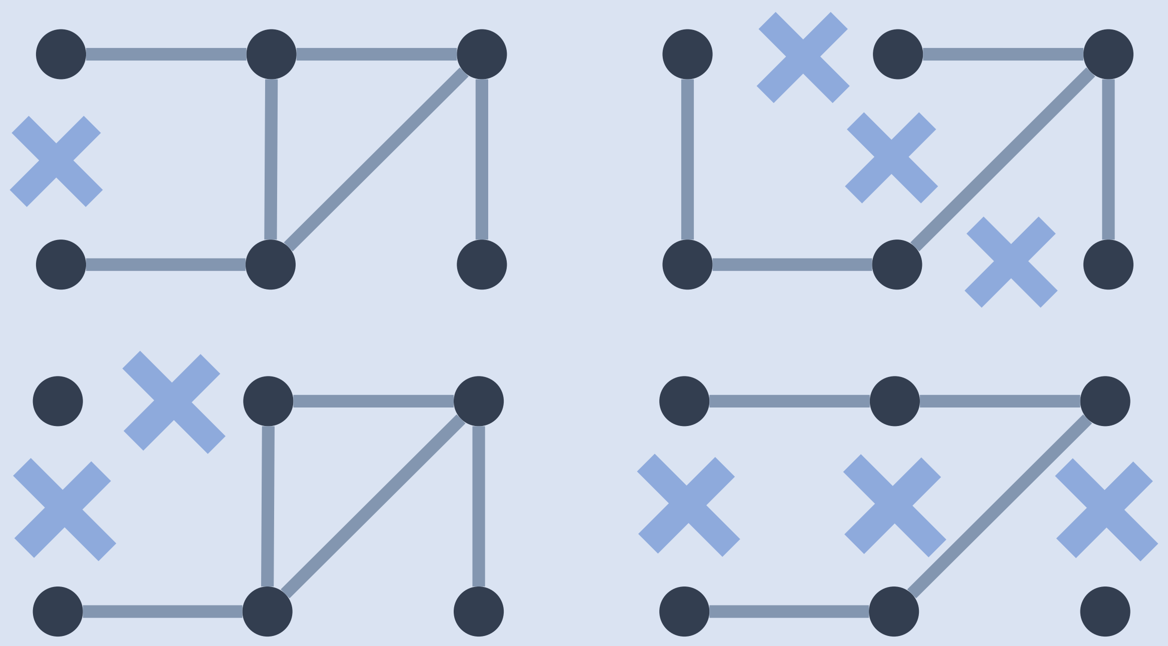
Key Idea

shortest paths

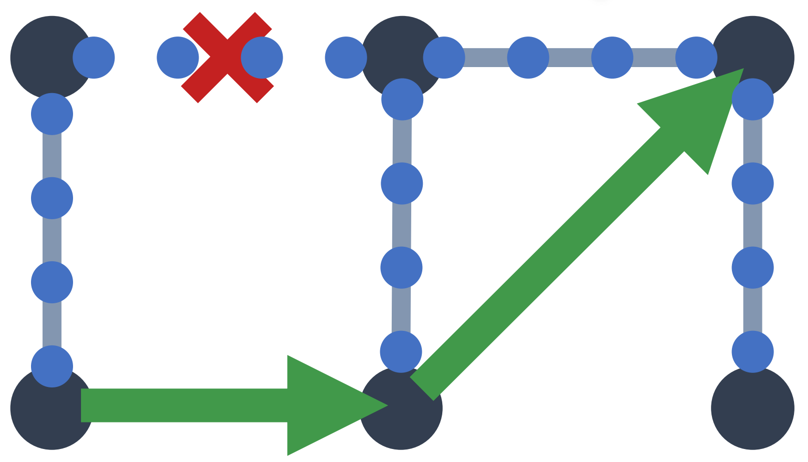


cold edges

Scenarios with same forwarding graph (32 total):

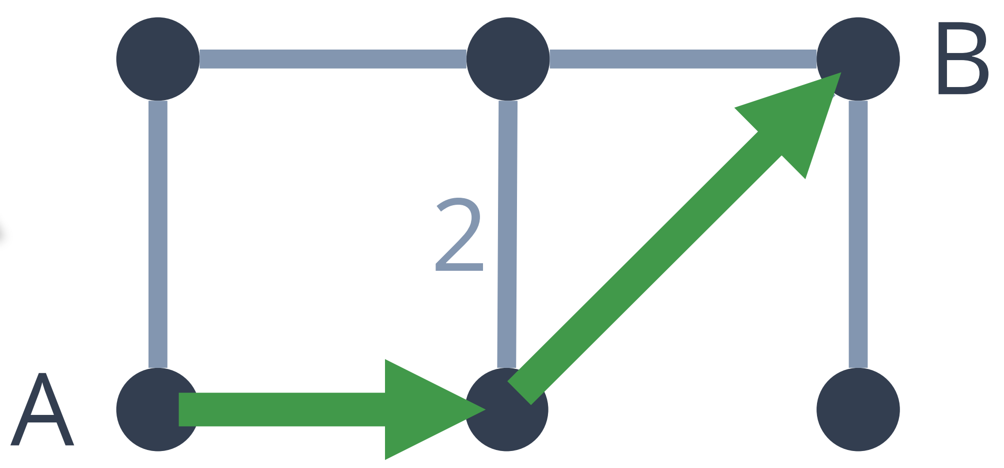


...



Key Idea

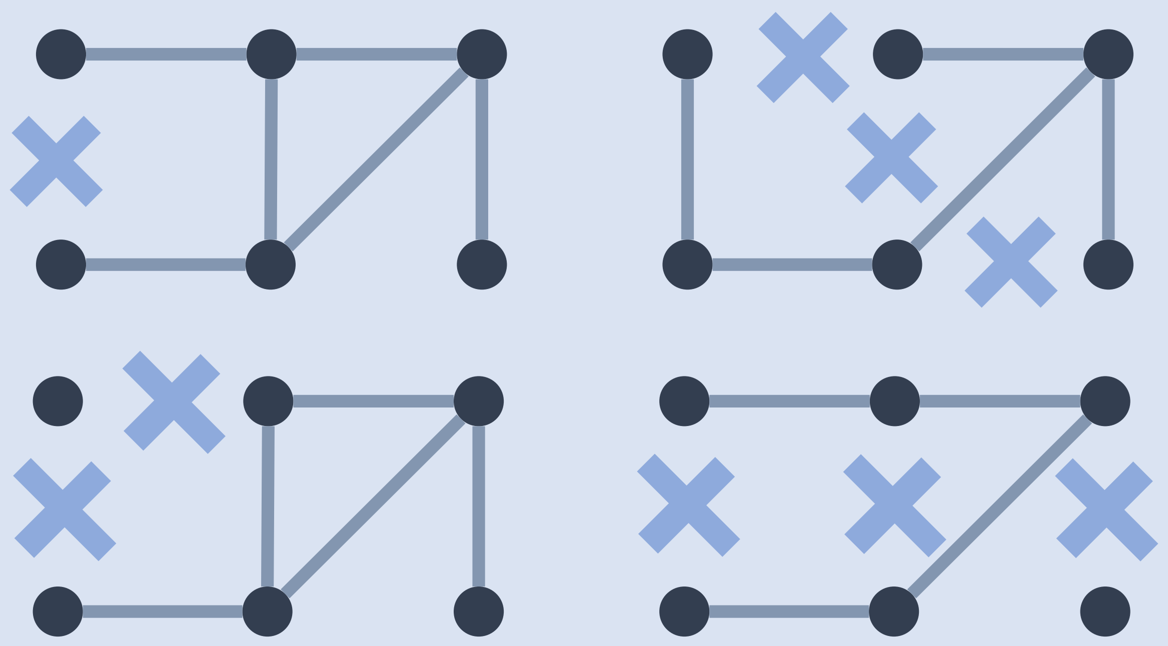
shortest paths



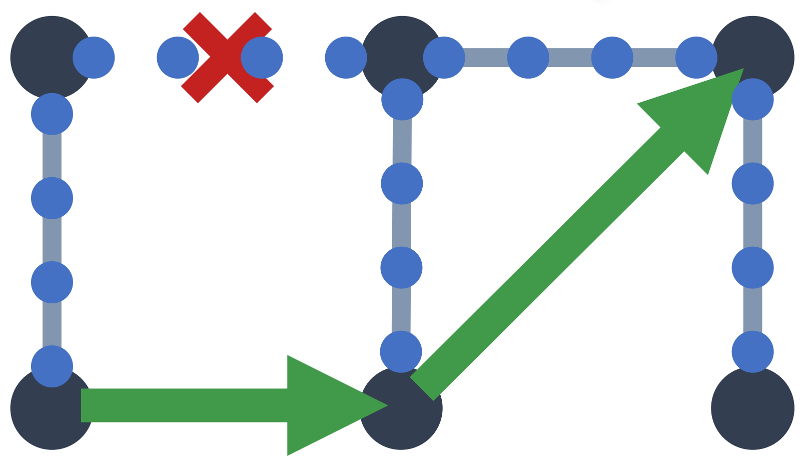
How to find these?

cold edges

Scenarios with same forwarding graph (32 total):



...



❄ for BGP

Algorithm 3 Hot edges for BGP

```
procedure HotBgp(u, d, E_fwd, L)
  X ← nodes in the same partition as u under L
  Br_L ← Top3(Br, X)                              ▷ BGP pre-processing (§4.2)
  Rr_L ← Rr ∩ X
  H ← AllSp(Rr_L, Br_L, L)                        ▷ all shortest paths (Alg. 2)
  D ← {u}                                         ▷ decision points
          ∪ {y | (x, y) ∈ Static_d ∩ E_fwd}
          ∪ {y | (x, y) ∈ E_fwd ∧ Nh_d(x) ≠ Nh_d(y)}
  for each x ∈ D do
    H ← H ∪ Sp_L(x, Nh_d(x))                      ▷ shortest path x → Nh_d(x)
  H ← H ∪ (Static_d ∩ E_fwd)                      ▷ traversed static routes
  if Rr_L = ∅ then
    H ← H ∪ AllSp({u}, Br_L)                      ▷ ensure connectivity
  return H
```

see paper

❄ for BGP

Algorithm 3 Hot edges for BGP

procedure HotBgp(u, d, E_{fwd}, L)
  $X \leftarrow$ nodes in the same partition as u under L
  $\text{Br}_L \leftarrow \text{Top3}(\text{Br}, X)$ ▹ BGP pre-processing (§4.2)
  $\text{Rr}_L \leftarrow \text{Rr} \cap X$
  $\mathcal{H} \leftarrow \text{AllSp}(\text{Rr}_L, \text{Br}_L, L)$ ▹ all shortest paths (Alg. 2)
  $\mathcal{D} \leftarrow \{u\}$ ▹ decision points
    $\cup \{y \mid (x, y) \in \text{Static}_d \cap E_{\text{fwd}}\}$
    $\cup \{y \mid (x, y) \in E_{\text{fwd}} \wedge \text{Nh}_d(x) \neq \text{Nh}_d(y)\}$
  for each $x \in \mathcal{D}$ **do**
    $\mathcal{H} \leftarrow \mathcal{H} \cup \text{Sp}_L(x, \text{Nh}_d(x))$ ▹ shortest path $x \rightarrow \text{Nh}_d(x)$
  $\mathcal{H} \leftarrow \mathcal{H} \cup (\text{Static}_d \cap E_{\text{fwd}})$ ▹ traversed static routes
  if $\text{Rr}_L = \emptyset$ **then**
    $\mathcal{H} \leftarrow \mathcal{H} \cup \text{AllSp}(\{u\}, \text{Br}_L)$ ▹ ensure connectivity
  return $\mathcal{H}$

see paper

network partitions

route reflection

dependence on
IGP costs

❄ for BGP

Algorithm 3 Hot edges for BGP

```
procedure HotBgp(u, d, E_fwd, L)
  X ← nodes in the same partition as u under L
  Br_L ← Top3(Br, X)                                   ▷ BGP pre-processing (§4.2)
  Rr_L ← Rr ∩ X
  H ← AllSp(Rr_L, Br_L, L)                             ▷ all shortest paths (Alg. 2)
  D ← {u}                                              ▷ decision points
        ∪ {y | (x, y) ∈ Static_d ∩ E_fwd}
        ∪ {y | (x, y) ∈ E_fwd ∧ Nh_d(x) ≠ Nh_d(y)}
  for each x ∈ D do
    H ← H ∪ Sp_L(x, Nh_d(x))                           ▷ shortest path x → Nh_d(x)
  H ← H ∪ (Static_d ∩ E_fwd)                           ▷ traversed static
  if Rr_L = ∅ then
    H ← H ∪ AllSp({u}, Br_L)                           ▷ ensure conn ivity
  return H
```

see paper

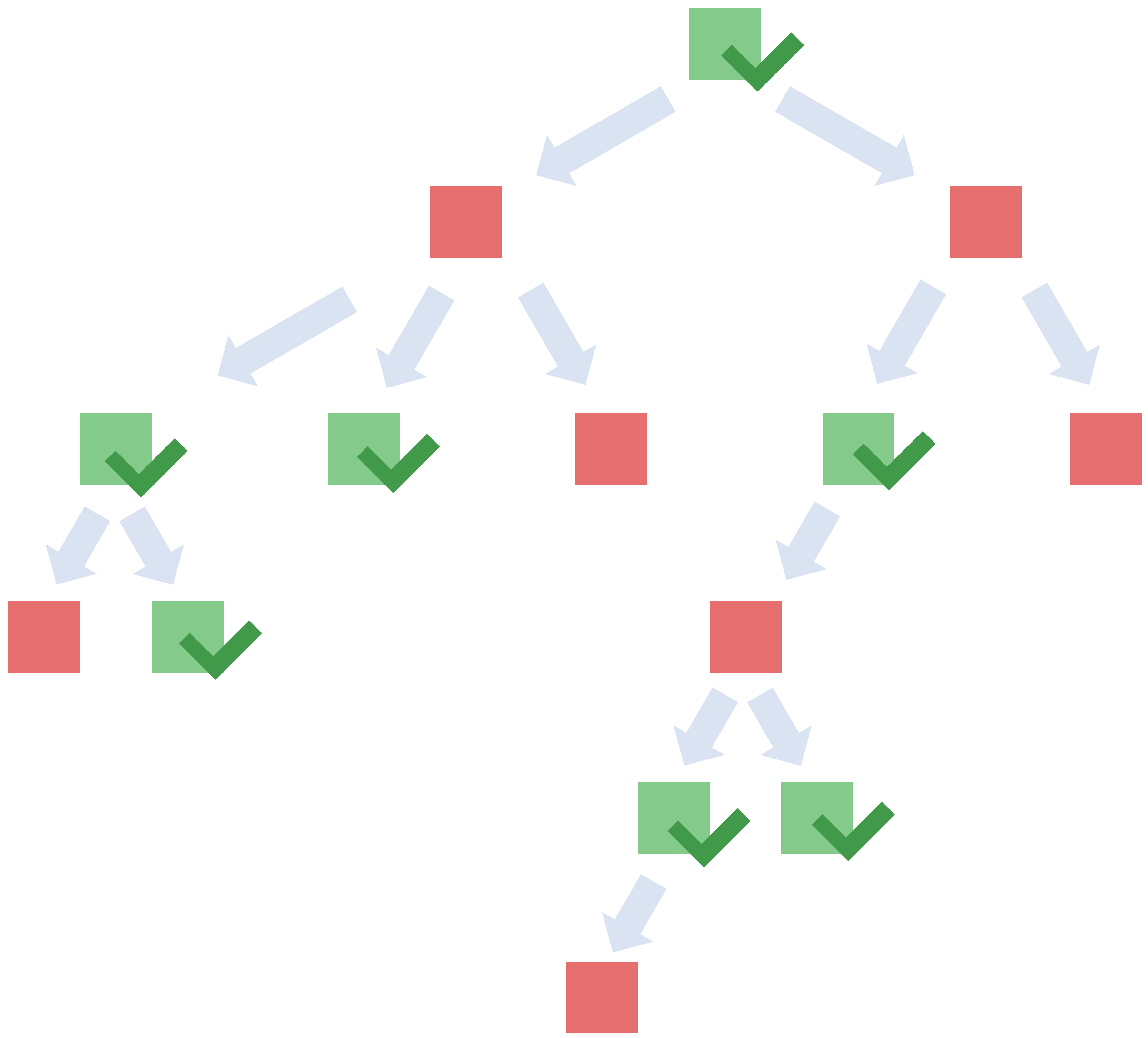
network partitions

route reflection

dependence on IGP costs

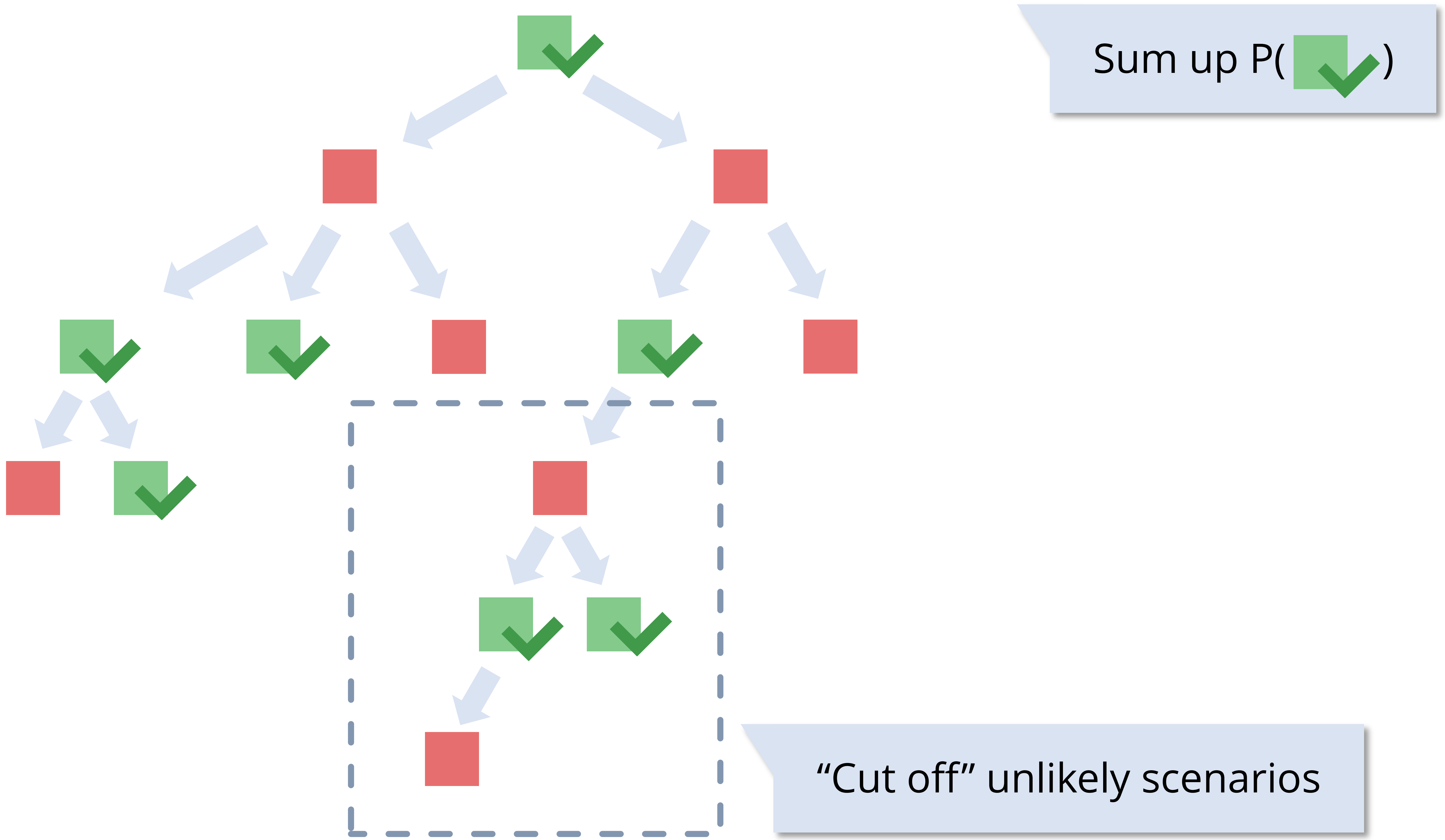
with correctness proof

Failure Exploration

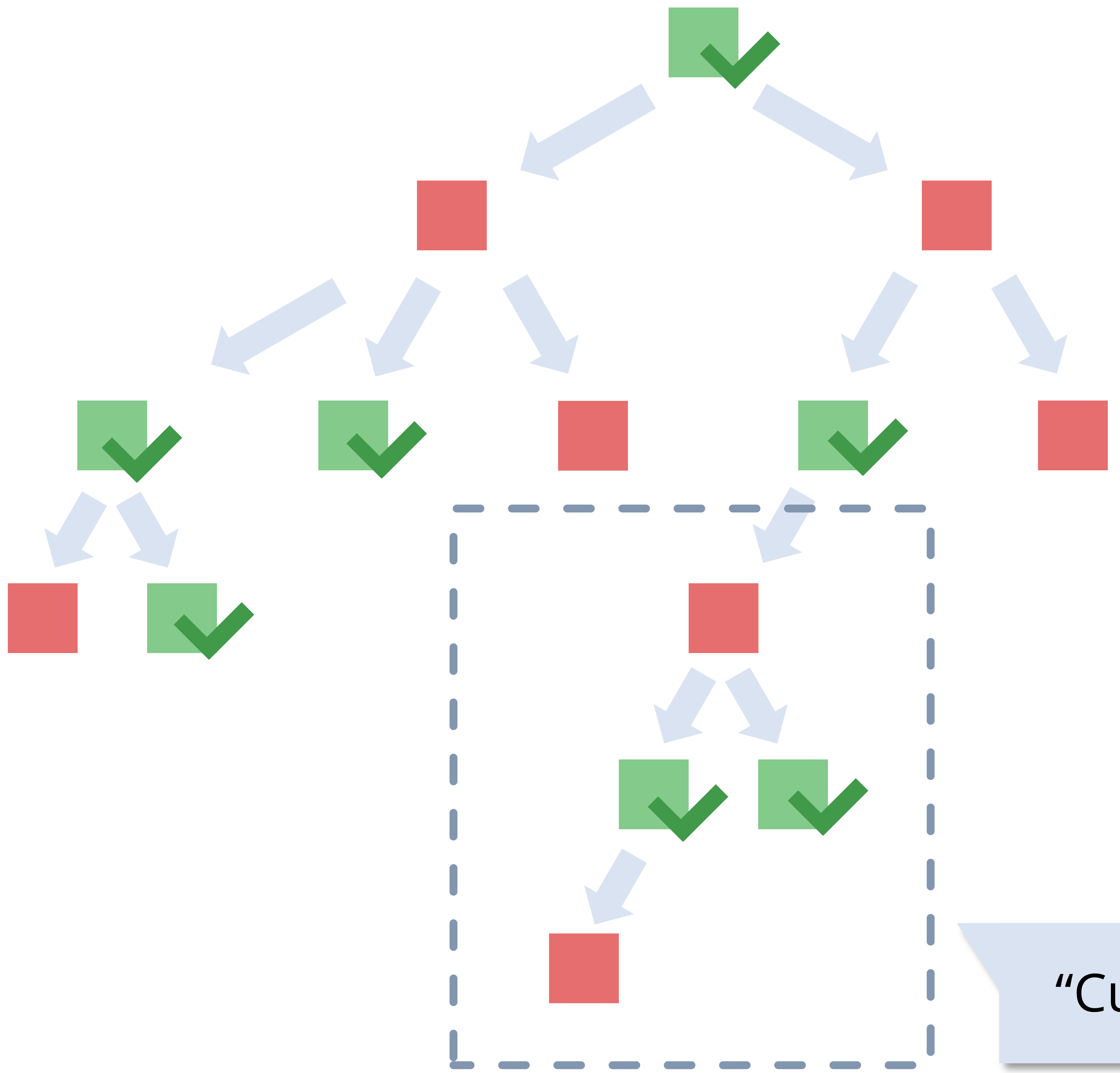


Sum up P(✓)

Failure Exploration



Failure Exploration



Sum up P(✓)

Efficiency depends on #❄

Very efficient in practice

"Cut off" unlikely scenarios

Implementation

Net Dice

nsg-ethz/netdice

- Reachability
- Path length
- Egress
- Waypointing
- Isolation
- Load balancing
- Congestion
- ...

Runtime

Single-flow (e.g. Reachability)

> *Few minutes* for *100s* of links for *four 9*s

> For 80% of scenarios, > 50% of links are ❄

Runtime

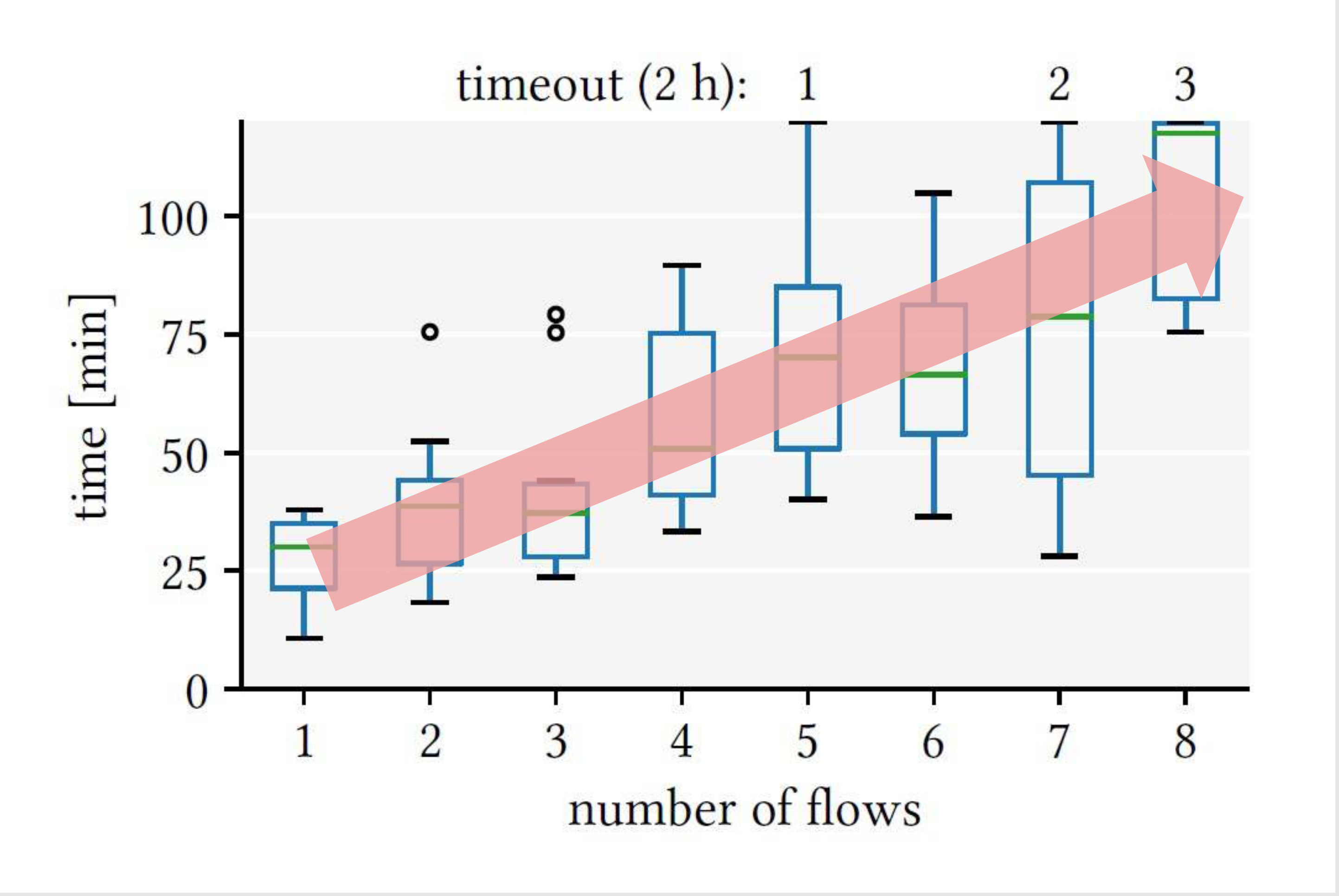
Single-flow (e.g. Reachability)

Few minutes for *100s* of links for *four 9s*

For 80% of scenarios, > 50% of links are ❄

Multi-flow (e.g. Isolation)

Performance degrades gracefully



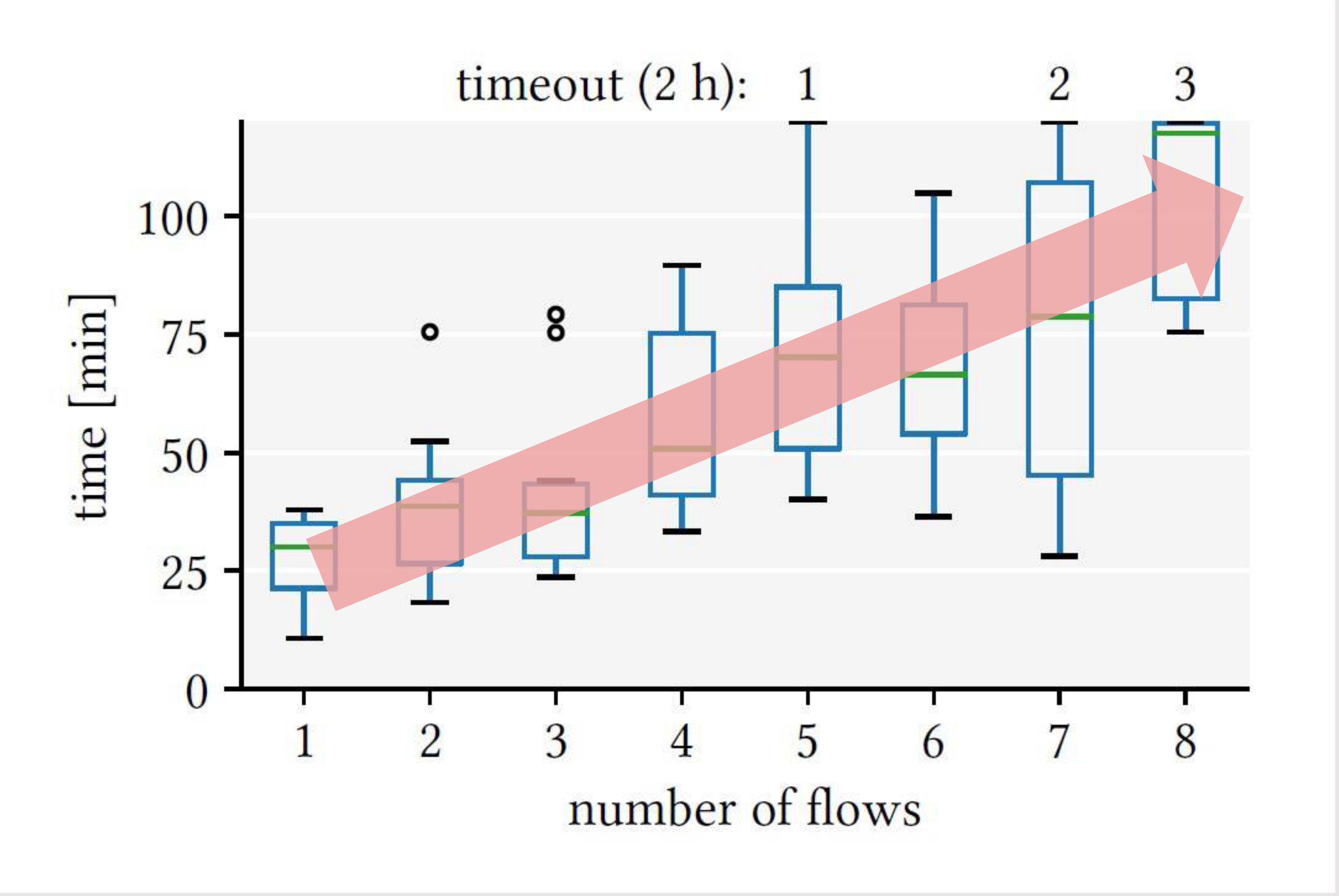
Runtime

Also analyzed real ISP config

Single-flow (e.g. Reachability)

Few minutes for *100s* of links for *four 9s*

For 80% of scenarios, > 50% of links are ❄



Multi-flow (e.g. Isolation)

Performance degrades gracefully

Runtime

Single-flow (e.g. Reachability)

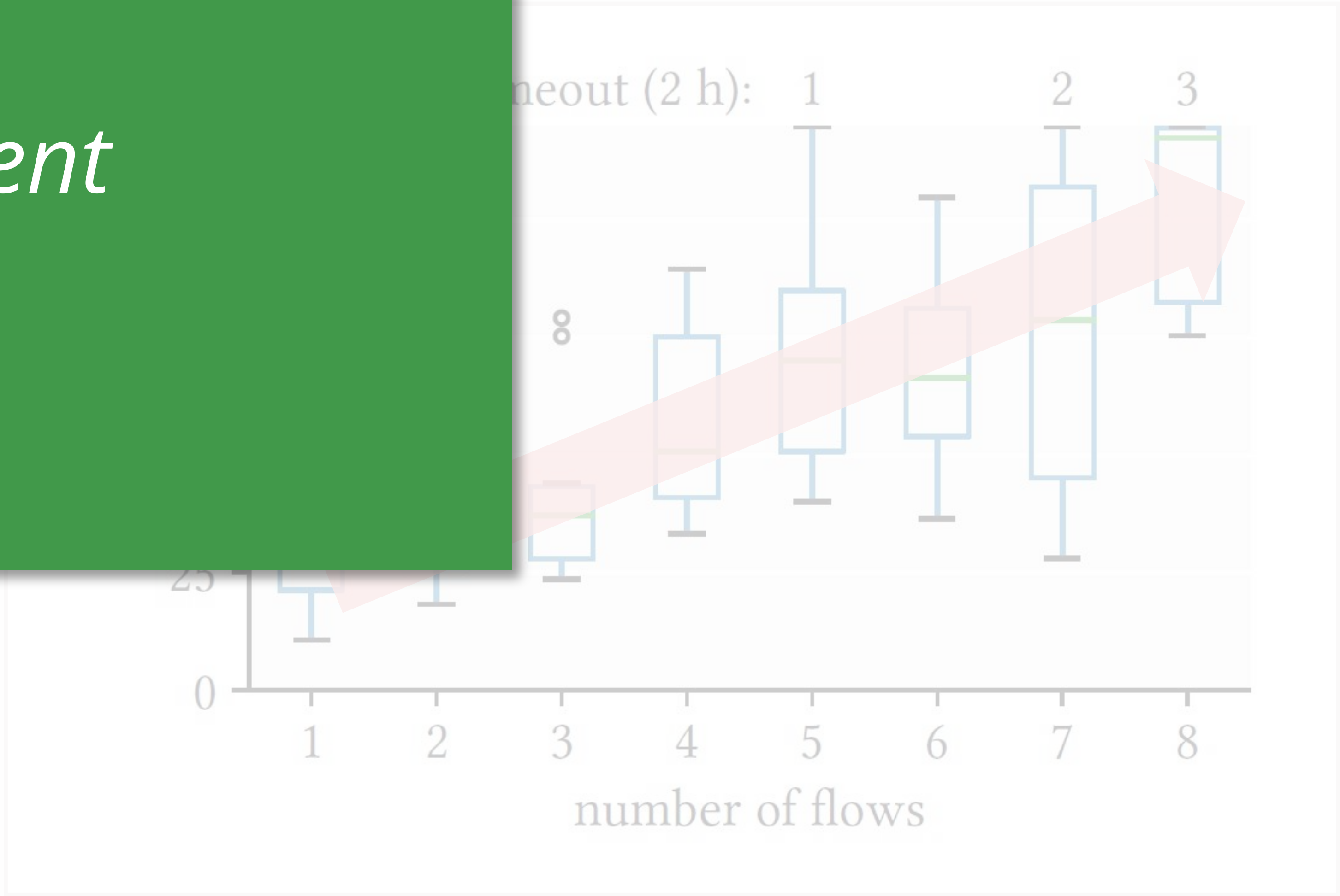
Few minutes for 10

For 80% of scenarios

Multi-flow (e.g. Isolation)

Performance degrades gracefully

Also analyzed real ISP config



NetDice is
precise and *efficient*

The three tales of (correct) network operations

Verification

going forward

2 Synthesis

going backward

Reconfiguration

going sideways

NetComplete: Practical Network-Wide Configuration Synthesis with Autocompletion




Ahmed El-Hassany Petar Tsankov Martin Vechev Laurent Vanbever

USENIX Symposium on Networked Systems Design and Implementation. April 2018.

NetComplete takes as inputs configuration sketches together with a set of high-level requirements

NetComplete takes as inputs configuration sketches together with a set of high-level requirements

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-p1 ? 20
  match community C1
...
```

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-p1 ? 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-p1 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

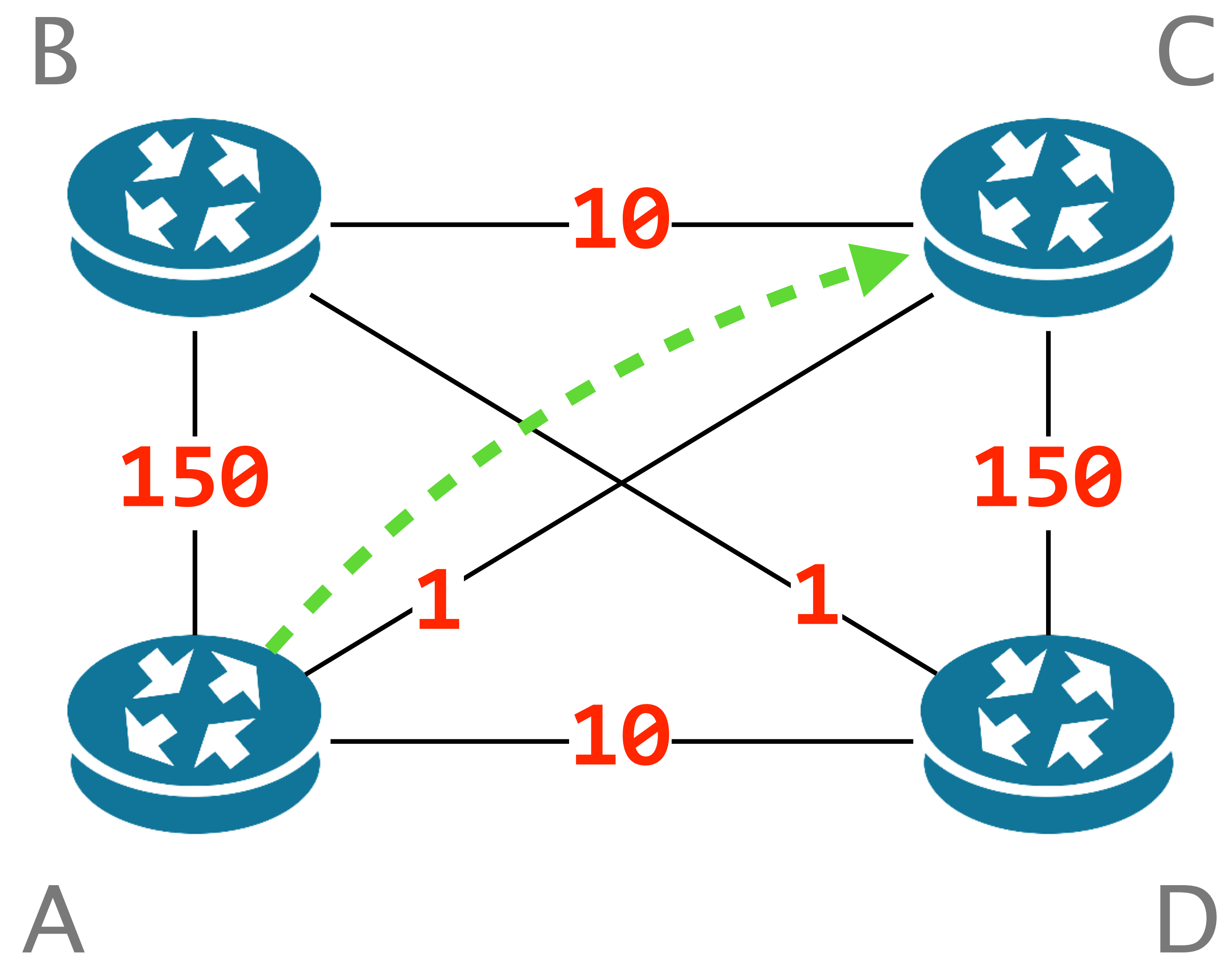
search space navigation

Insight #2

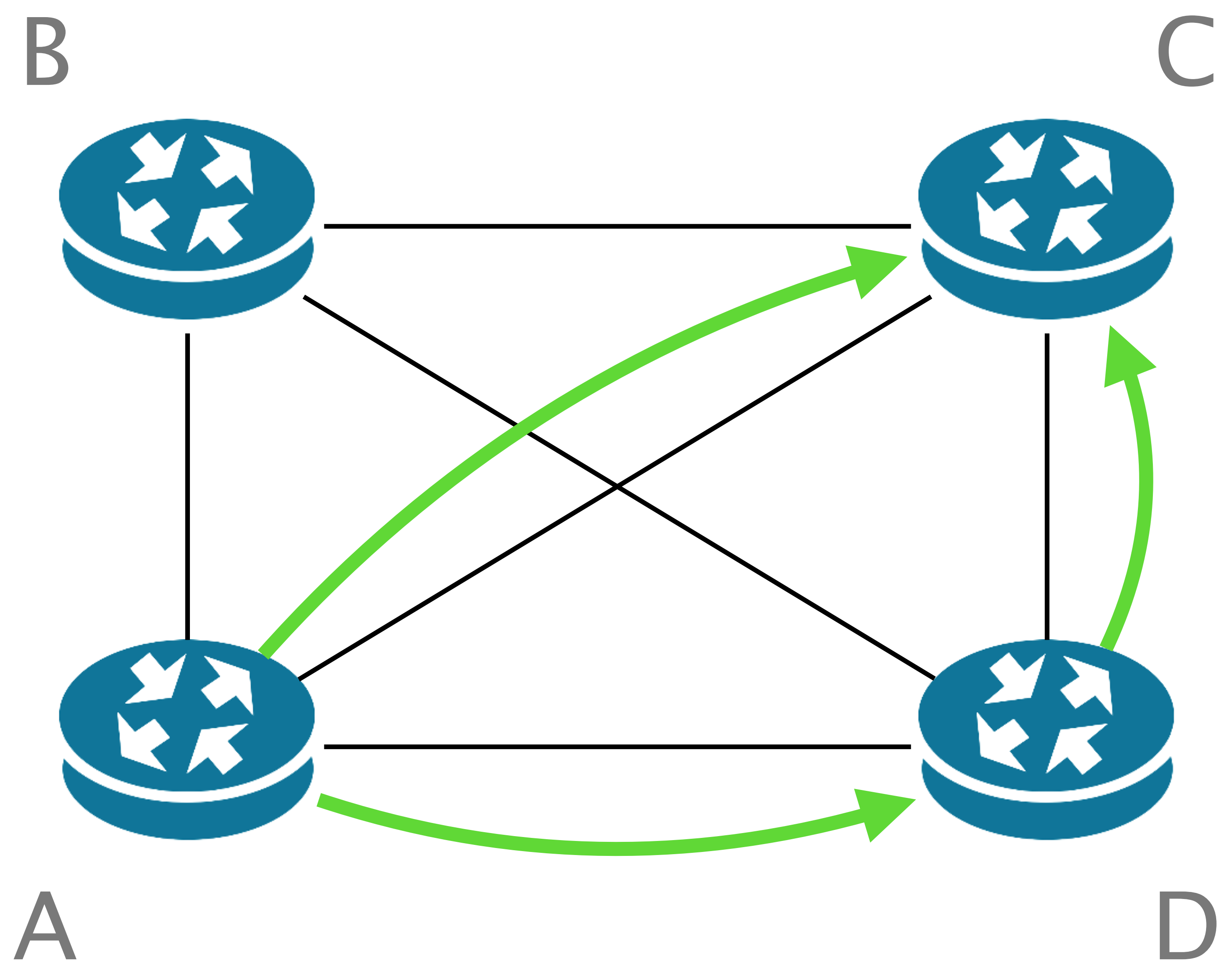
partial evaluation

search space reduction

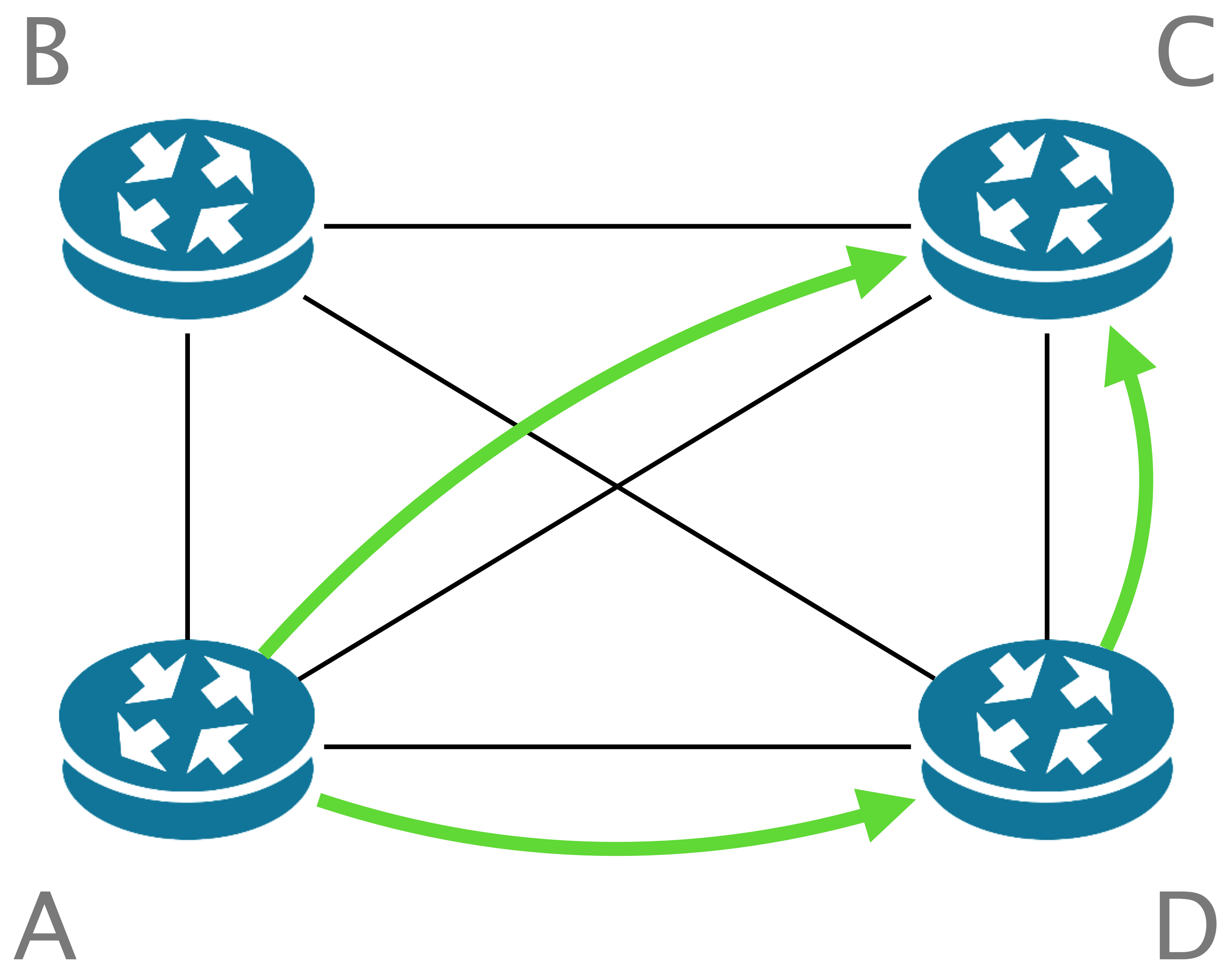
Consider this initial configuration in which (A,C) traffic is forwarded along the direct link



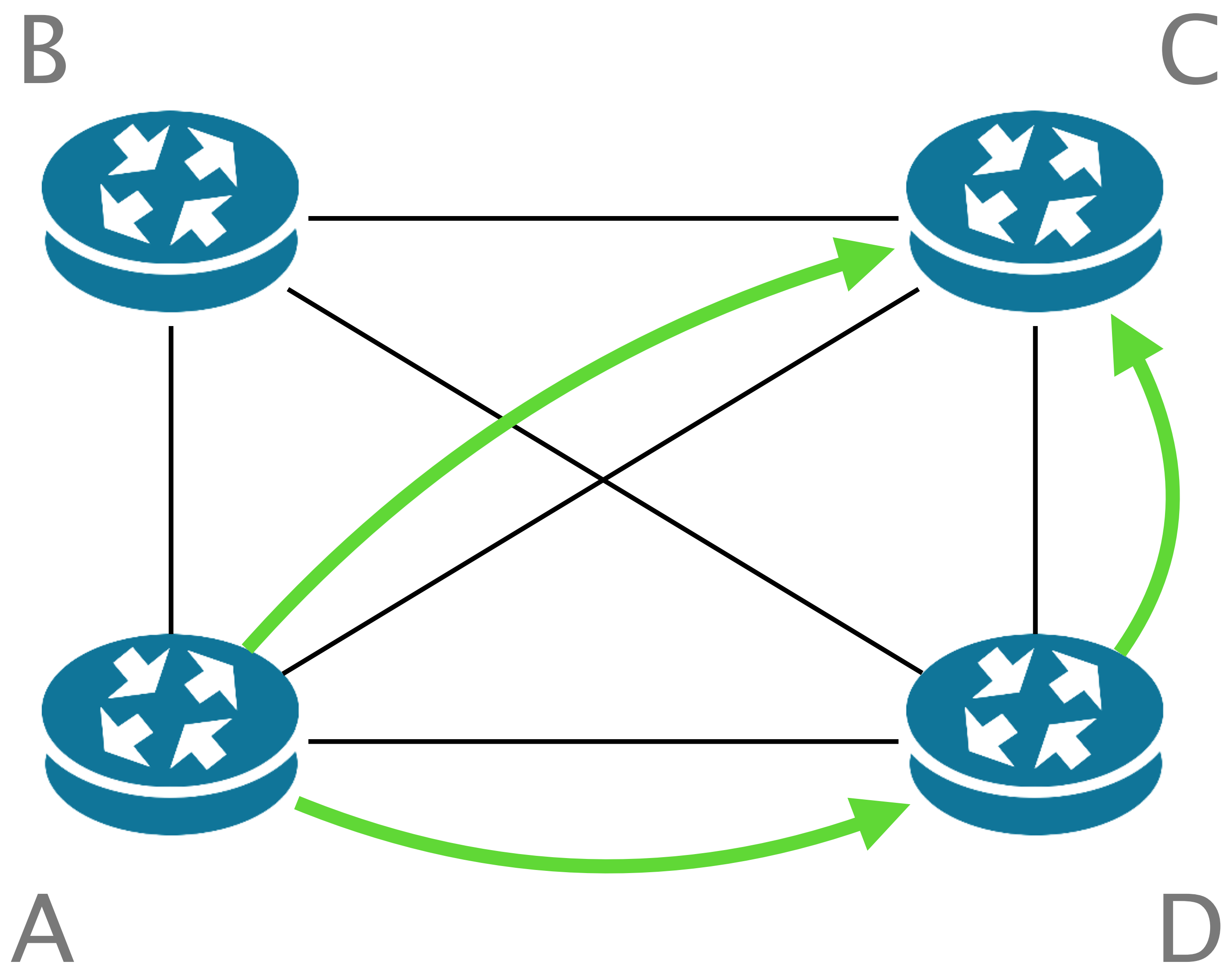
For performance reasons,
the operators want to enable load-balancing



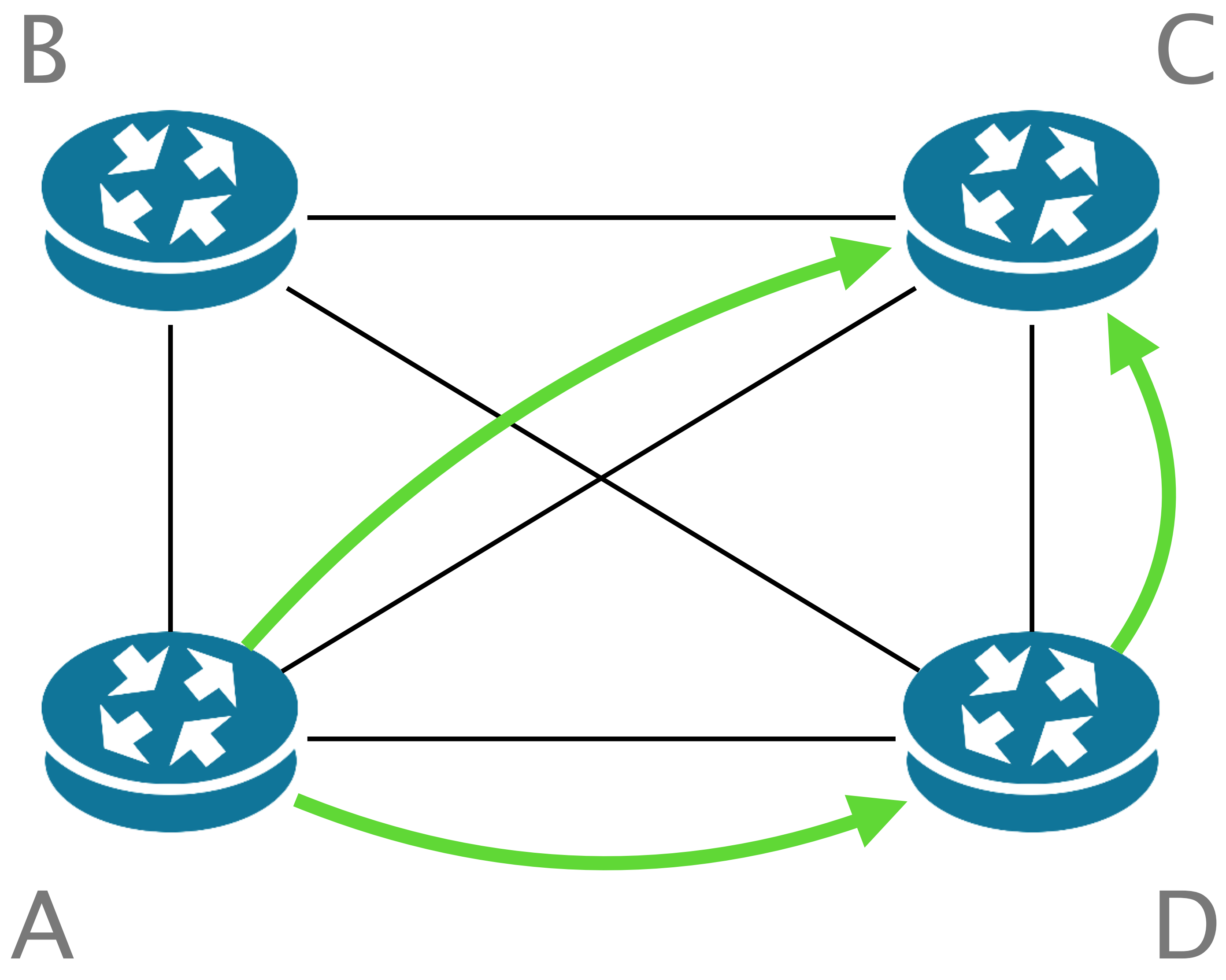
What should be the weights for this to happen?



input requirements

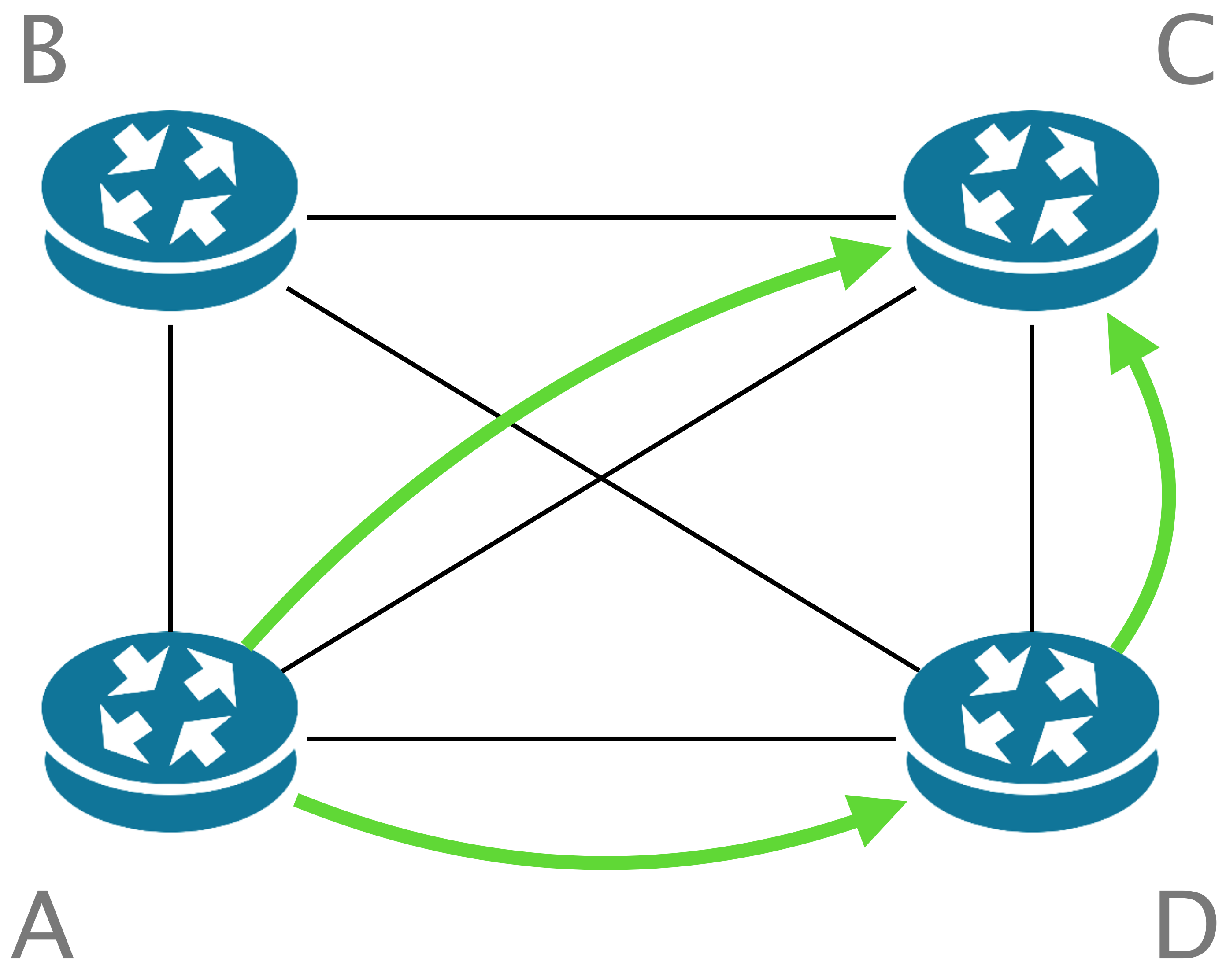


input requirements



synthesis procedure

input requirements

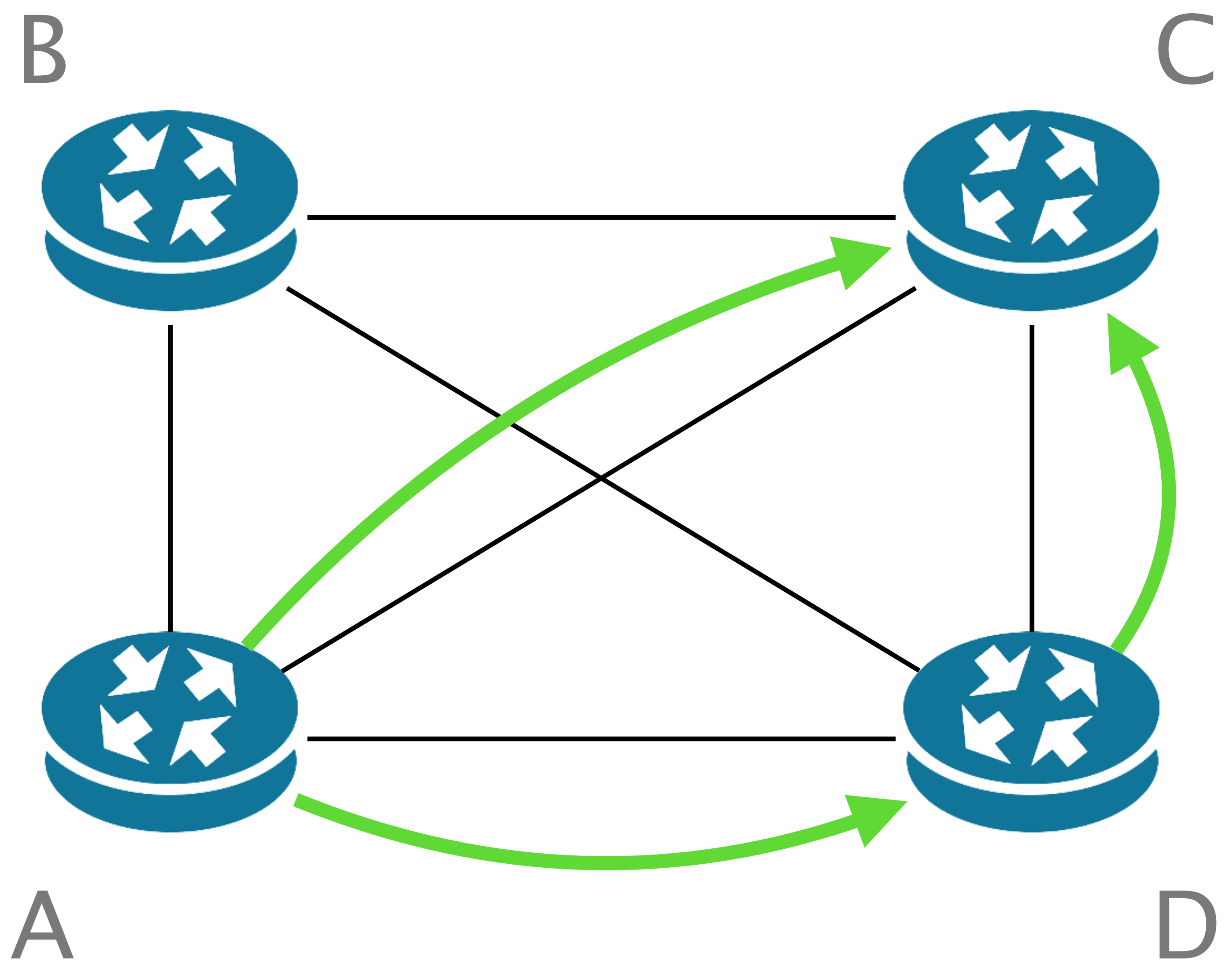


synthesis procedure

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

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

input requirements



synthesis procedure

∀X ∈ Paths(A,C)\Reqs

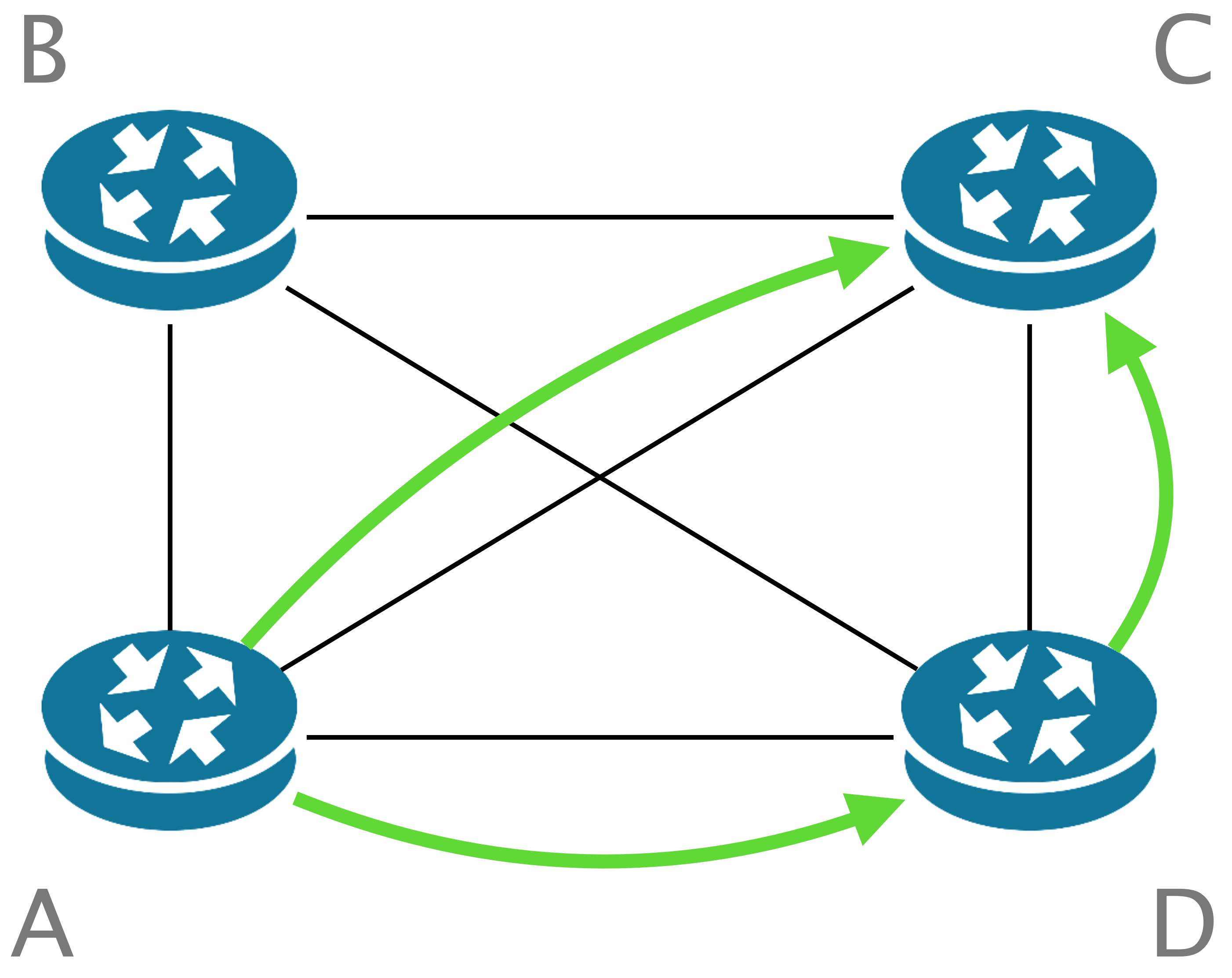
↓

Cost(A→C) = Cost(A→D→C) < Cost(X)

↓

Solve

input requirements



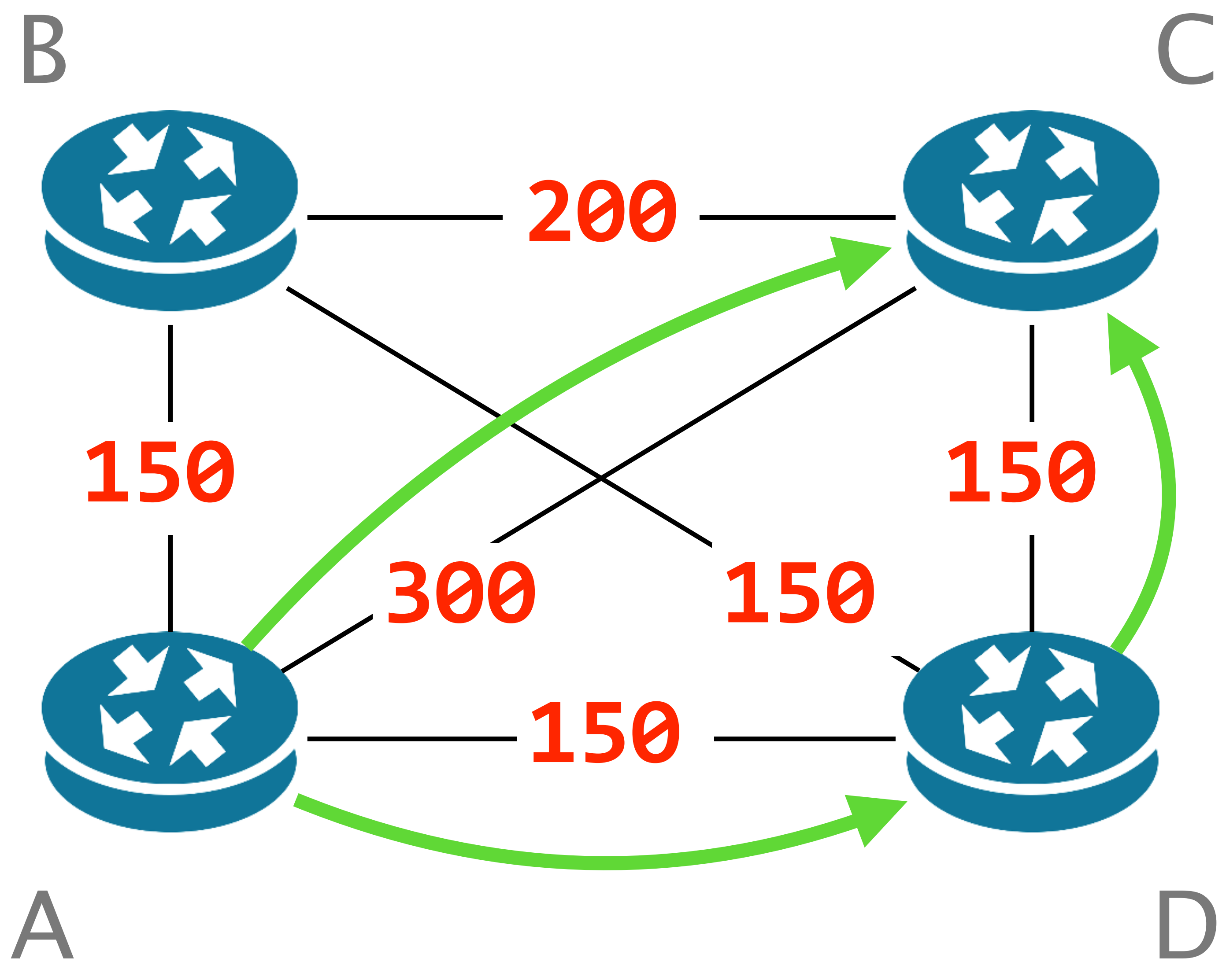
synthesis procedure

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

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

Solve

input requirements



Synthesized weights

synthesis procedure

∀X ∈ Paths(A,C)\Reqs

↓

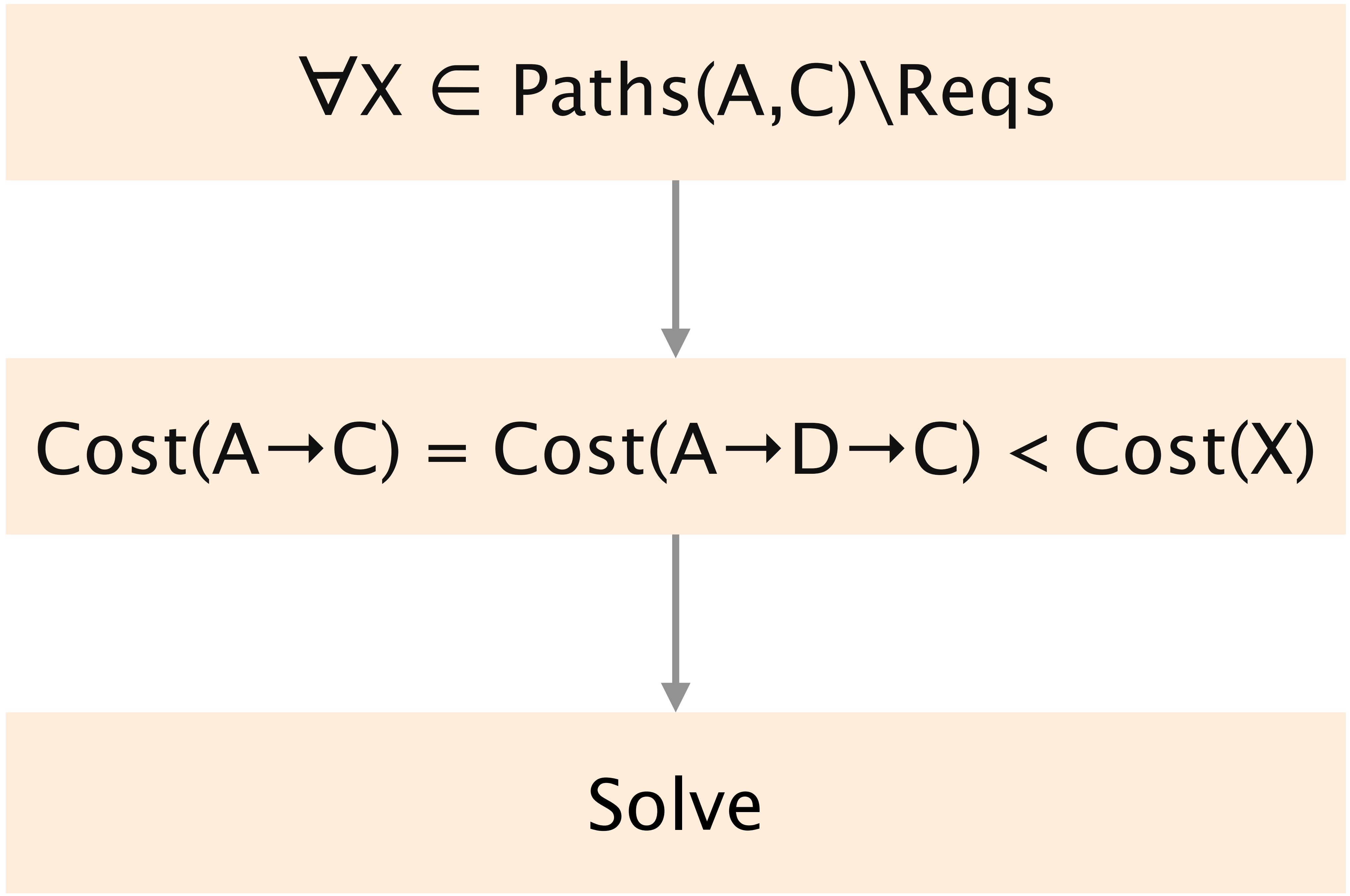
Cost(A→C) = Cost(A→D→C) < Cost(X)

↓

Solve

This was easy, but…

it does **not** scale



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

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

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

Solve

To scale, NetComplete leverages
Counter-Example Guided Inductive Synthesis (CEGIS)

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!

Instead of considering all paths between X and Y

Instead of considering all paths between X and Y

CEGIS
Part 1

Consider a random subset S of them and synthesize the weights considering S only

Instead of considering all paths between X and Y

CEGIS Part 1 — Consider a random subset S of them and synthesize the weights considering S only

intuition — **Fast** as S is small compared to all paths

Instead of considering all paths between *X* and *Y*

CEGIS Part 1

Consider a random subset *S* of them and synthesize the weights considering *S* only

intuition

Fast as *S* is small compared to all paths

but synthesized weights can be wrong

Instead of considering all paths between *X* and *Y*

CEGIS
Part 1

Consider a random subset *S* of them and
synthesize the weights considering *S* only

CEGIS
Part 2

Check whether the weights found comply
with the requirements over all paths

If so return
Else take a counter-example (a path)
that violates the Reqs and add it to *S*

Repeat.

Instead of considering all paths between *X* and *Y*

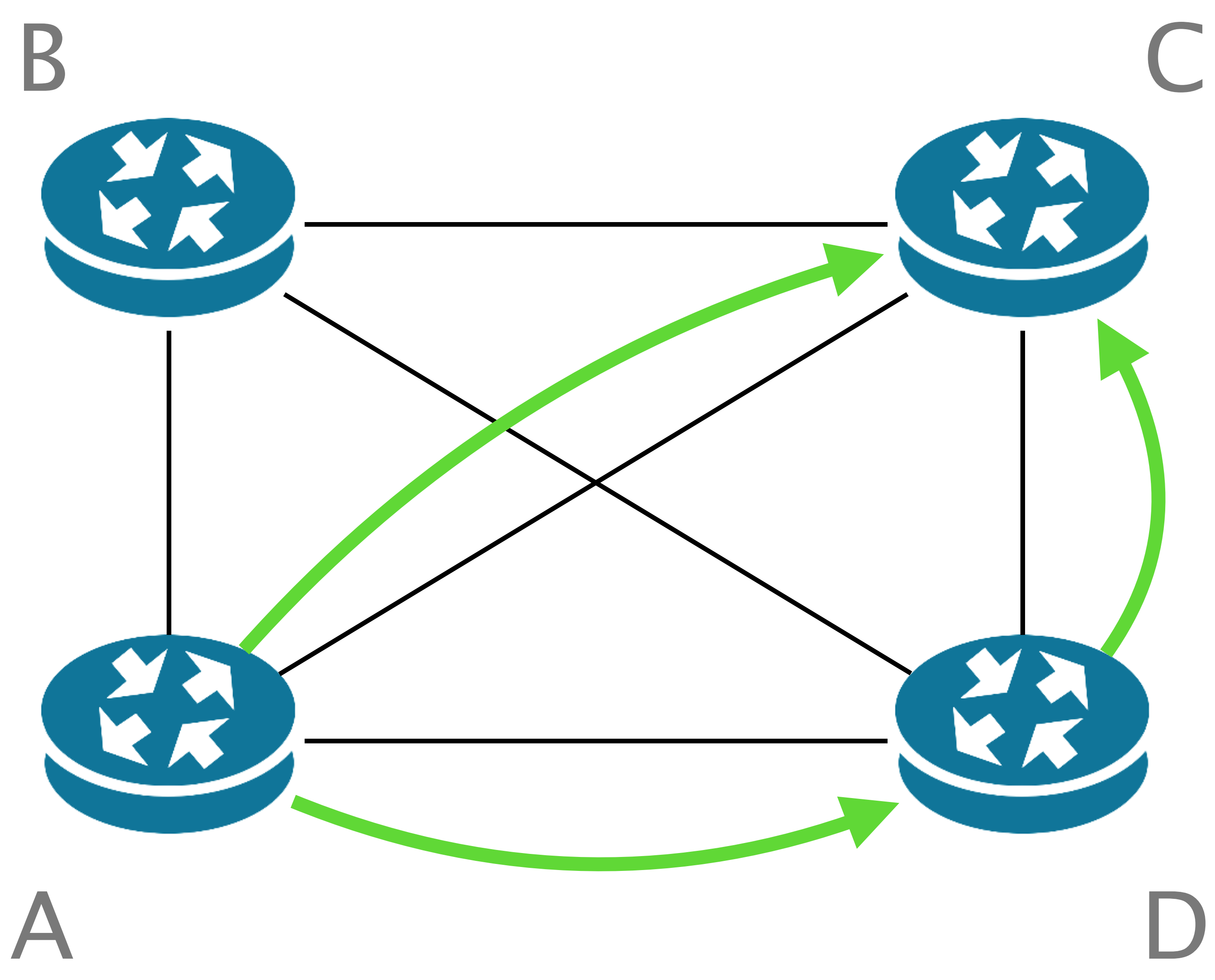
CEGIS Part 1 — Consider a random subset *S* of them and synthesize the weights considering *S* only

CEGIS Part 2 — **Check** whether the weights found comply with the requirements **over all paths**

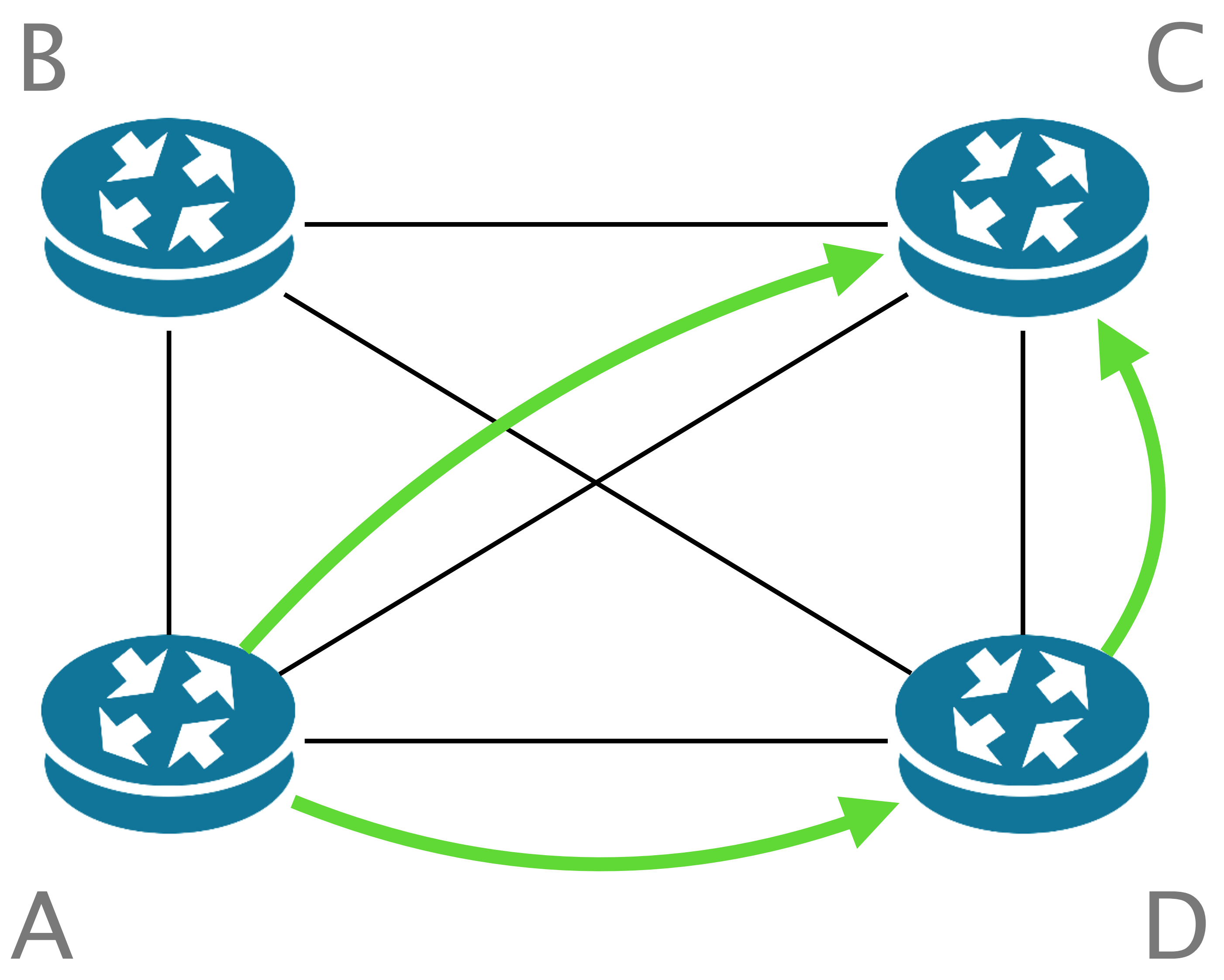
intuition — **Fast too**

simple shortest-path computation

input requirements

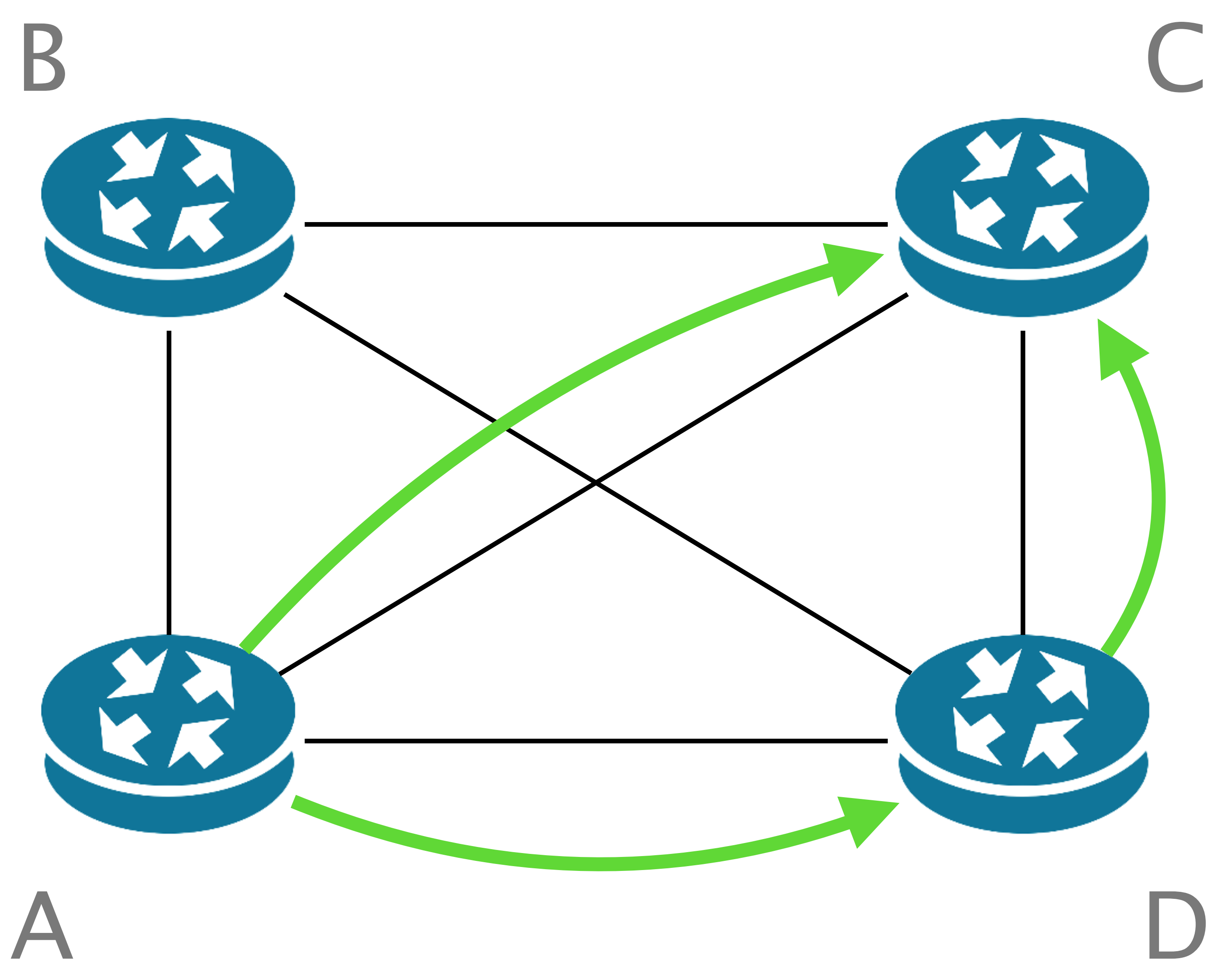


input requirements



synthesis procedure

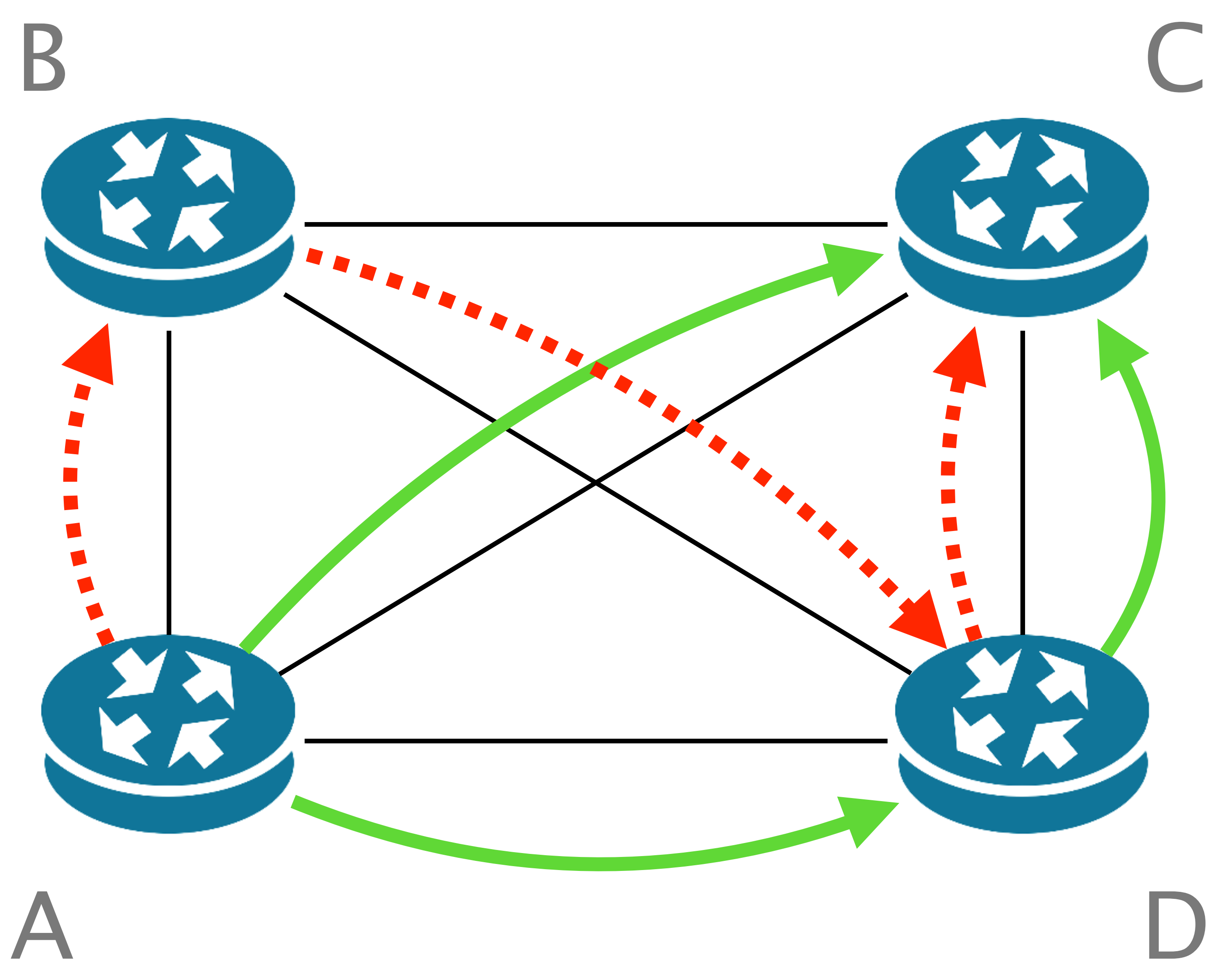
input requirements



synthesis procedure

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

input requirements

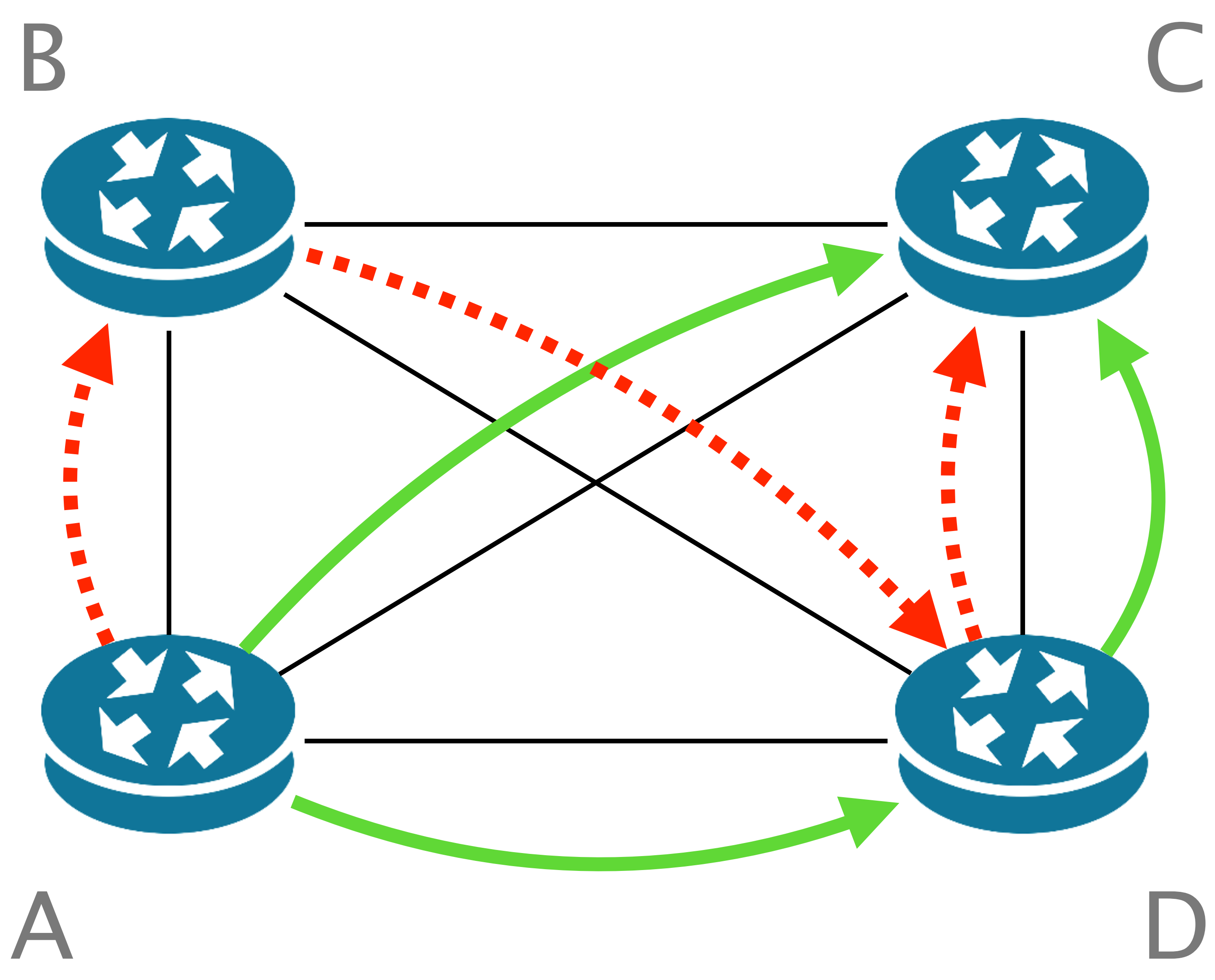


synthesis procedure

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

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

input requirements



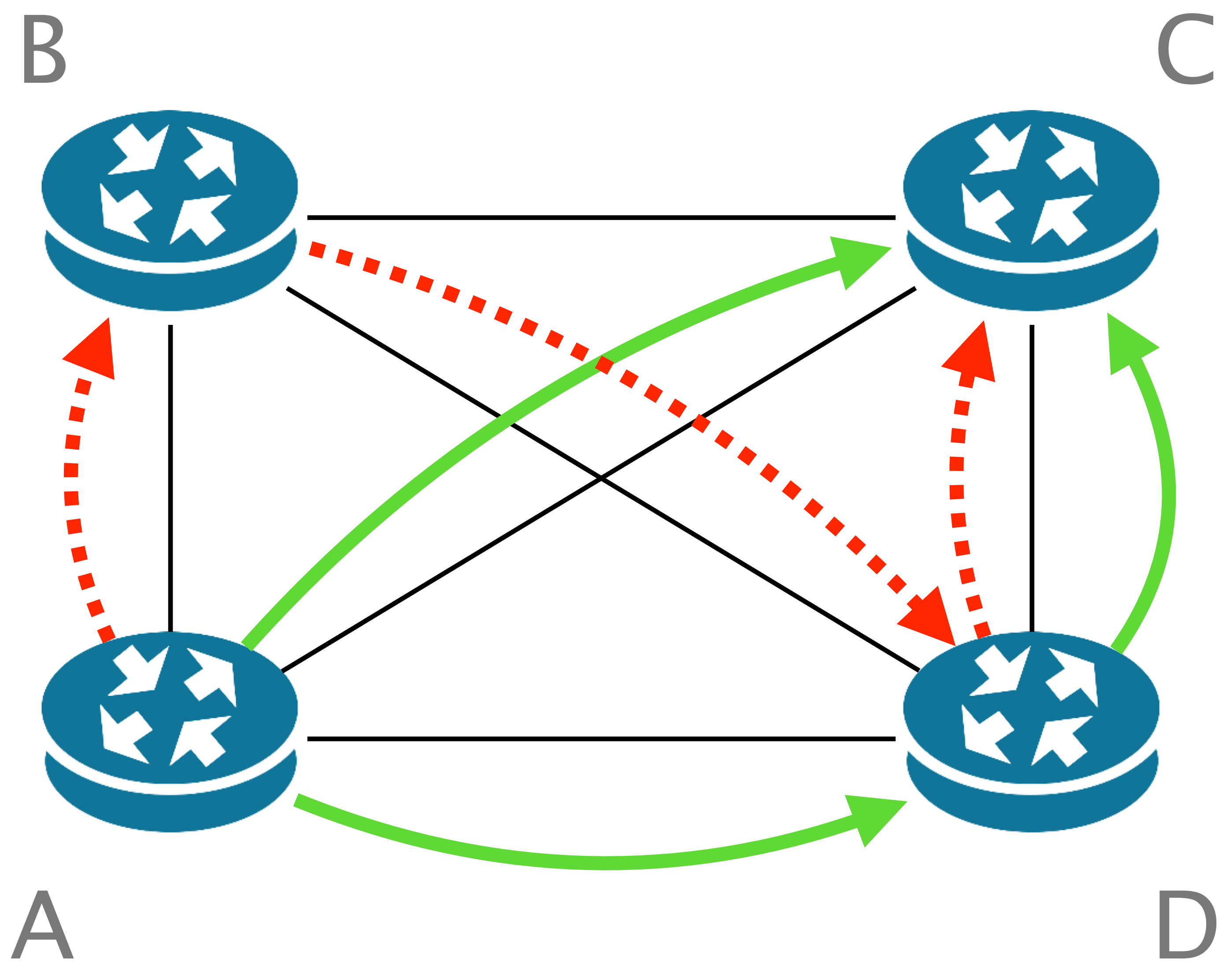
synthesis procedure

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

↓

Cost(A→C) = Cost(A→D→C) < Cost(X)

input requirements



synthesis procedure

∀X ∈ SamplePaths(A,C)\Reqs

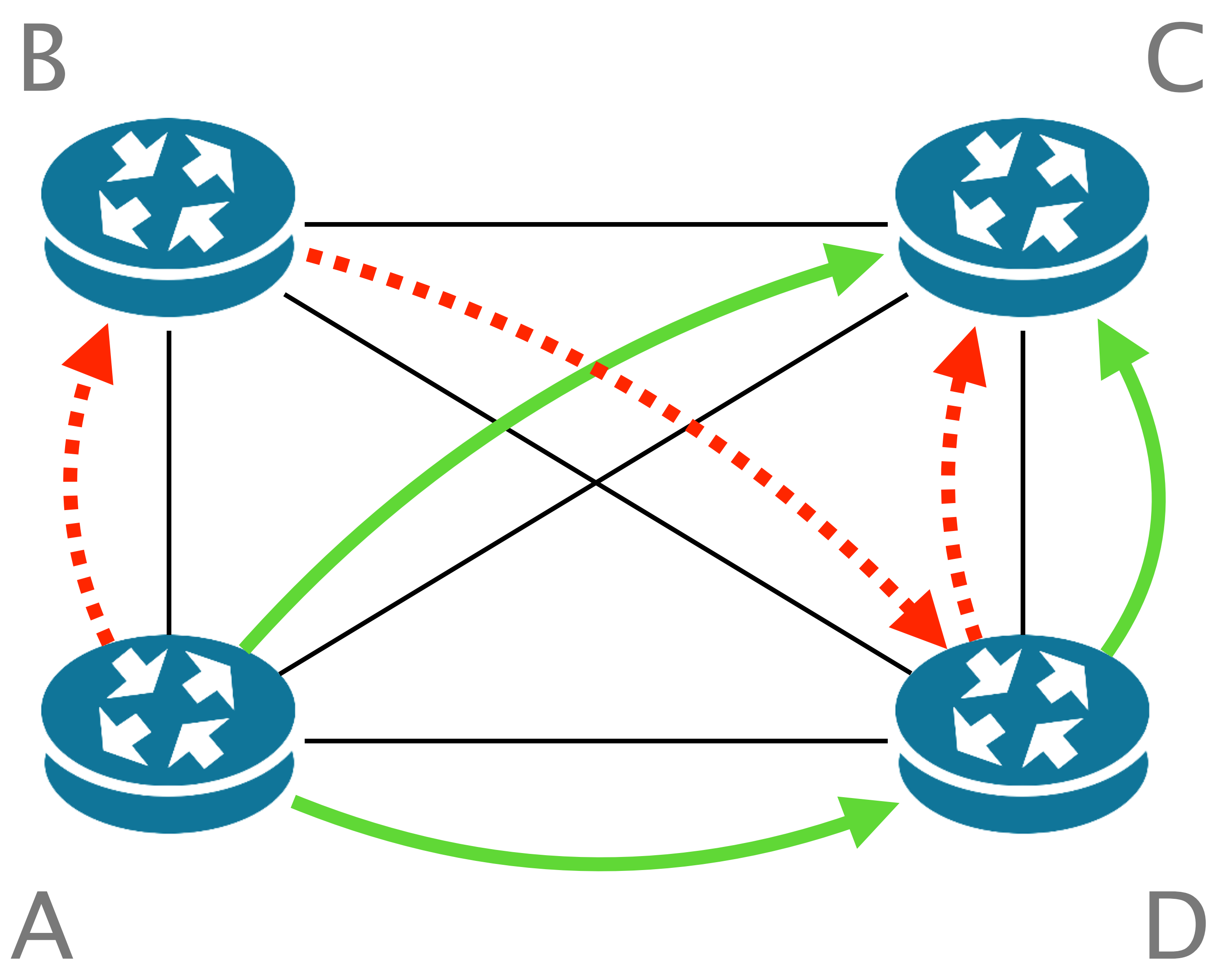
↓

Cost(A→C) = Cost(A→D→C) < Cost(X)

↓

Solve

input requirements



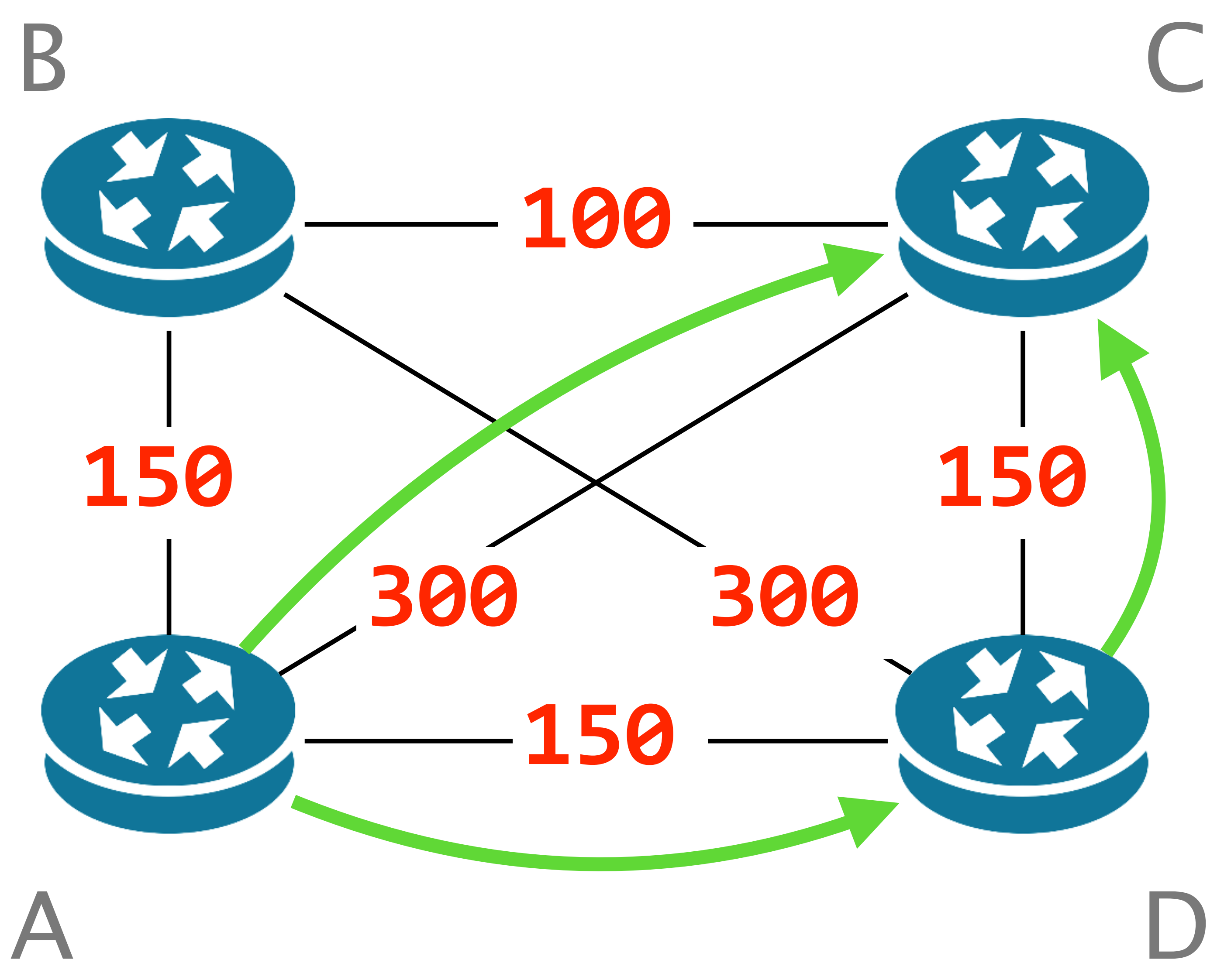
synthesis procedure

∀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

↓

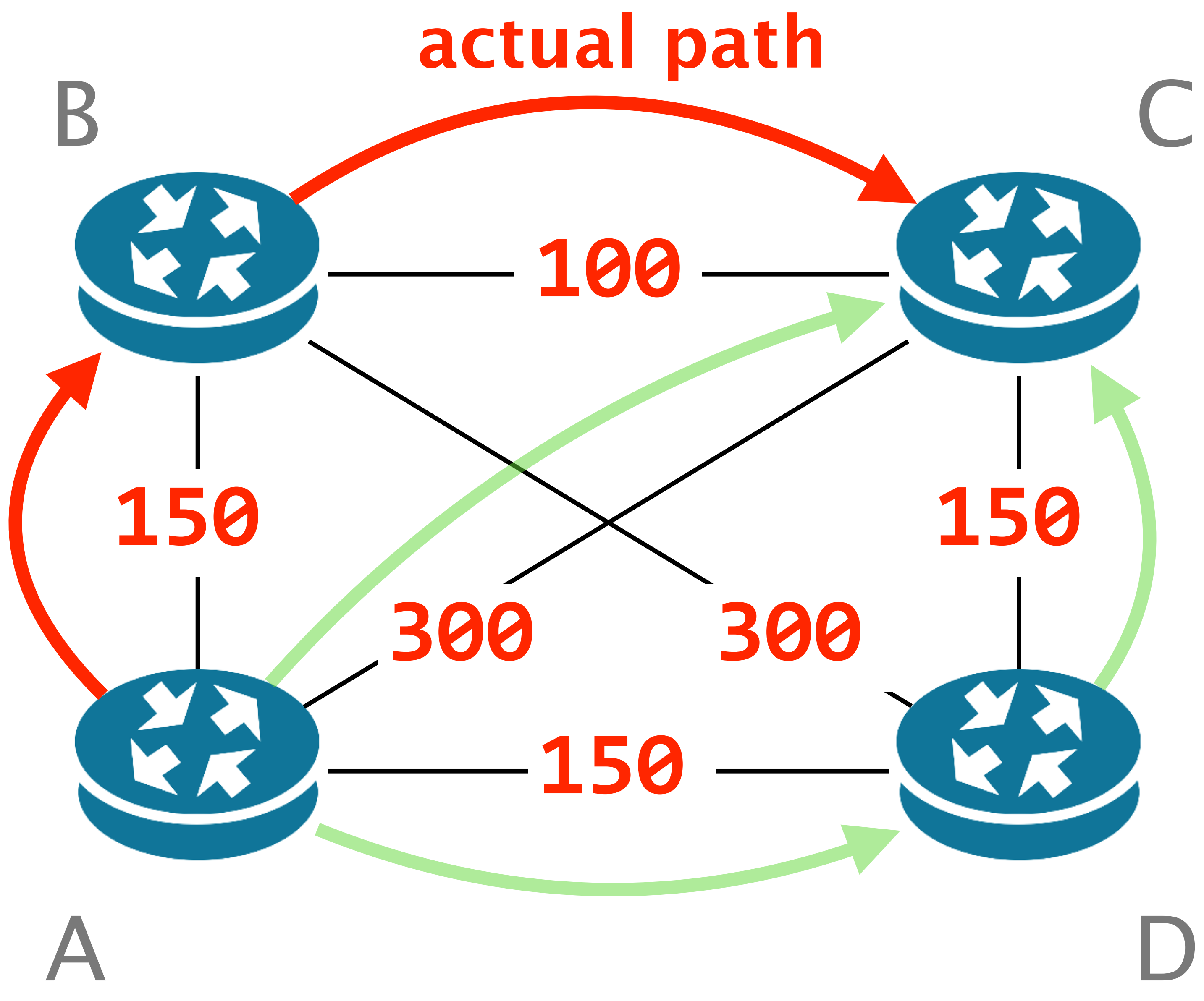
Cost(A→C) = Cost(A→D→C) < Cost(X)

↓

Solve

The synthesized weights are incorrect:

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



∀X ∈ SamplePaths(A,C)\Reqs

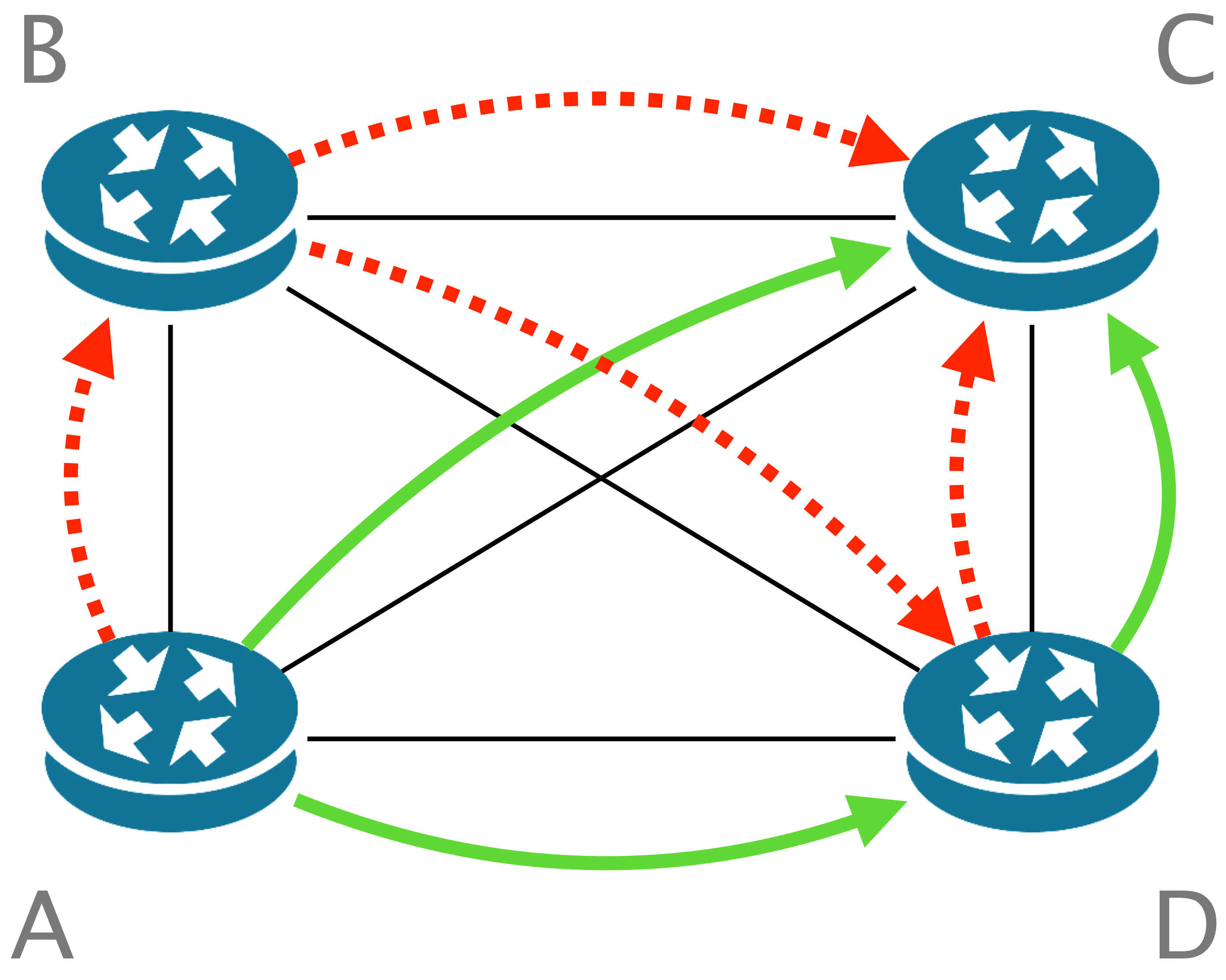
↓

Cost(A→C) = Cost(A→D→C) < Cost(X)

↓

Solve

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



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

Sample: { [A,B,D,C] } ∪ { [A,B,C] }

The entire procedure usually converges in few iterations making it very fast in practice

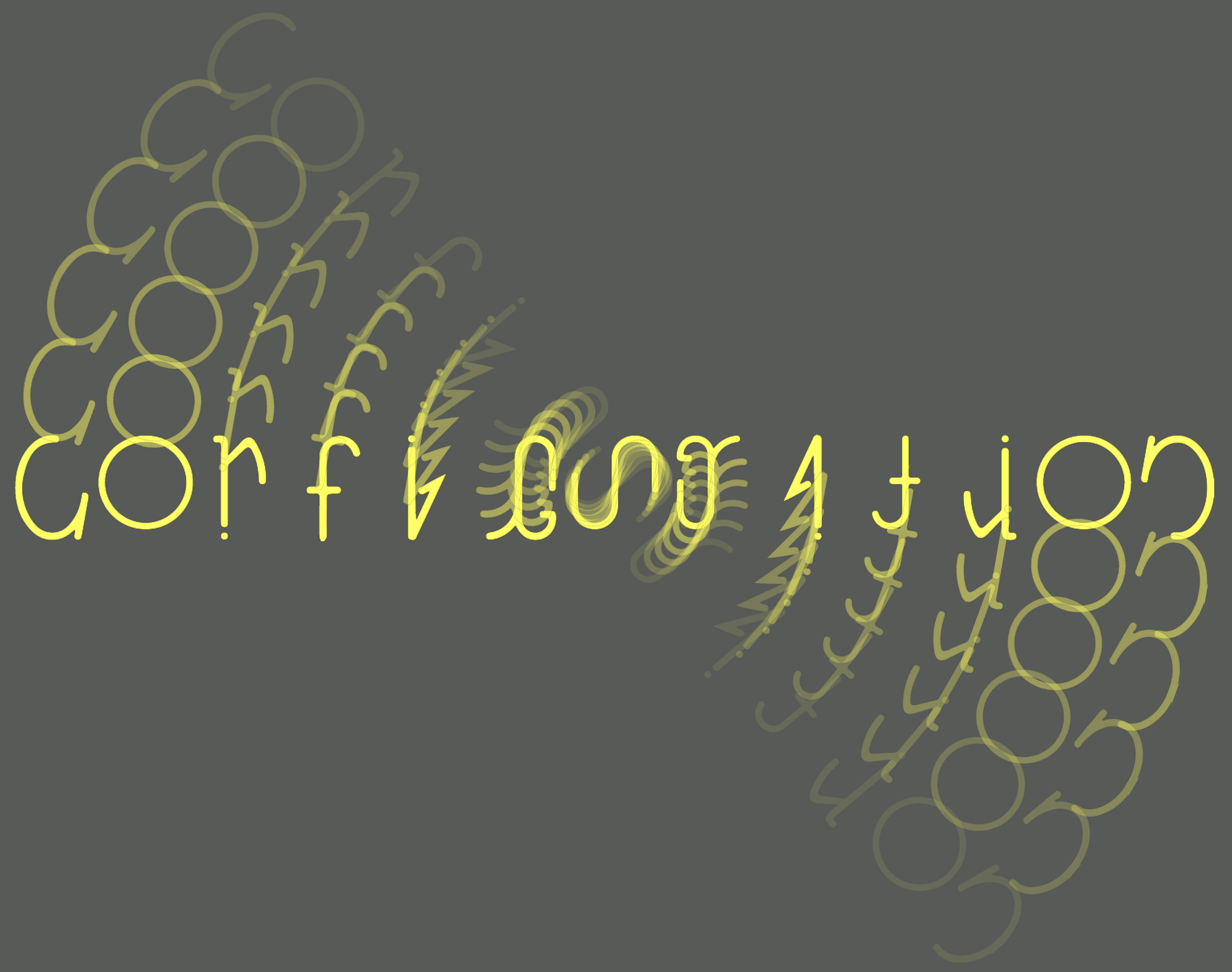
	Network size	Reqs. type	Synthesis time
OSPF synthesis time (sec)	Large ~150 nodes	Simple	14s
		ECMP	13s
		Ordered	249s

settings

16 reqs, 50% symbolic, 5 repet.

CEGIS enabled

The three tales of (correct) network operations



Verification

going forward

Synthesis

going backward

3 Reconfiguration

going sideways

ETH zürich

Networked Systems
ETH Zürich — seit 2015

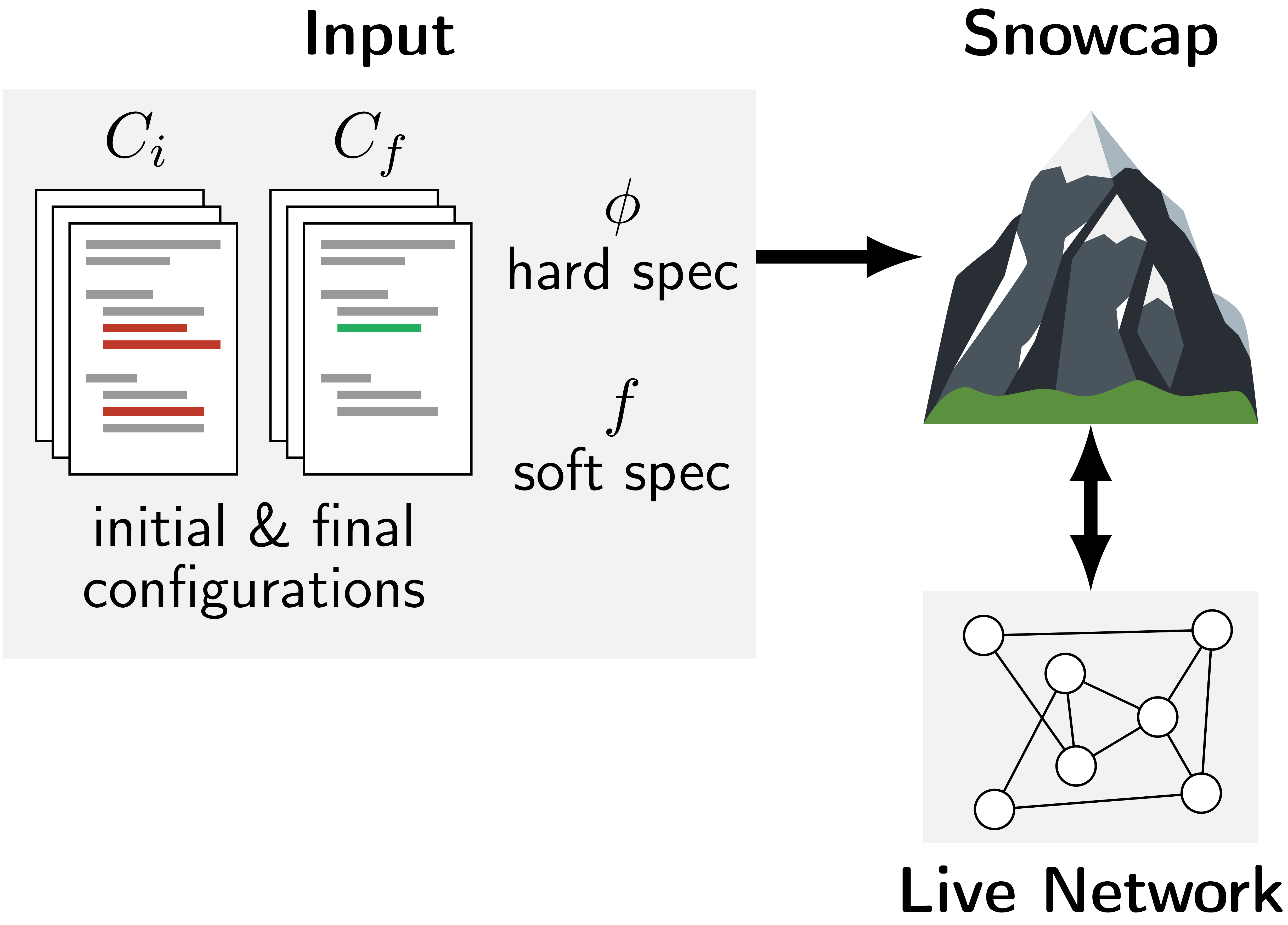
Snowcap: Synthesizing Network-Wide Configuration Updates

Tibor Schneider Rüdiger Birkner Laurent Vanbever

SIGCOMM'21, August 24, 2021

Artifacts Available · Artifacts Evaluated Functional · Results Reproduced

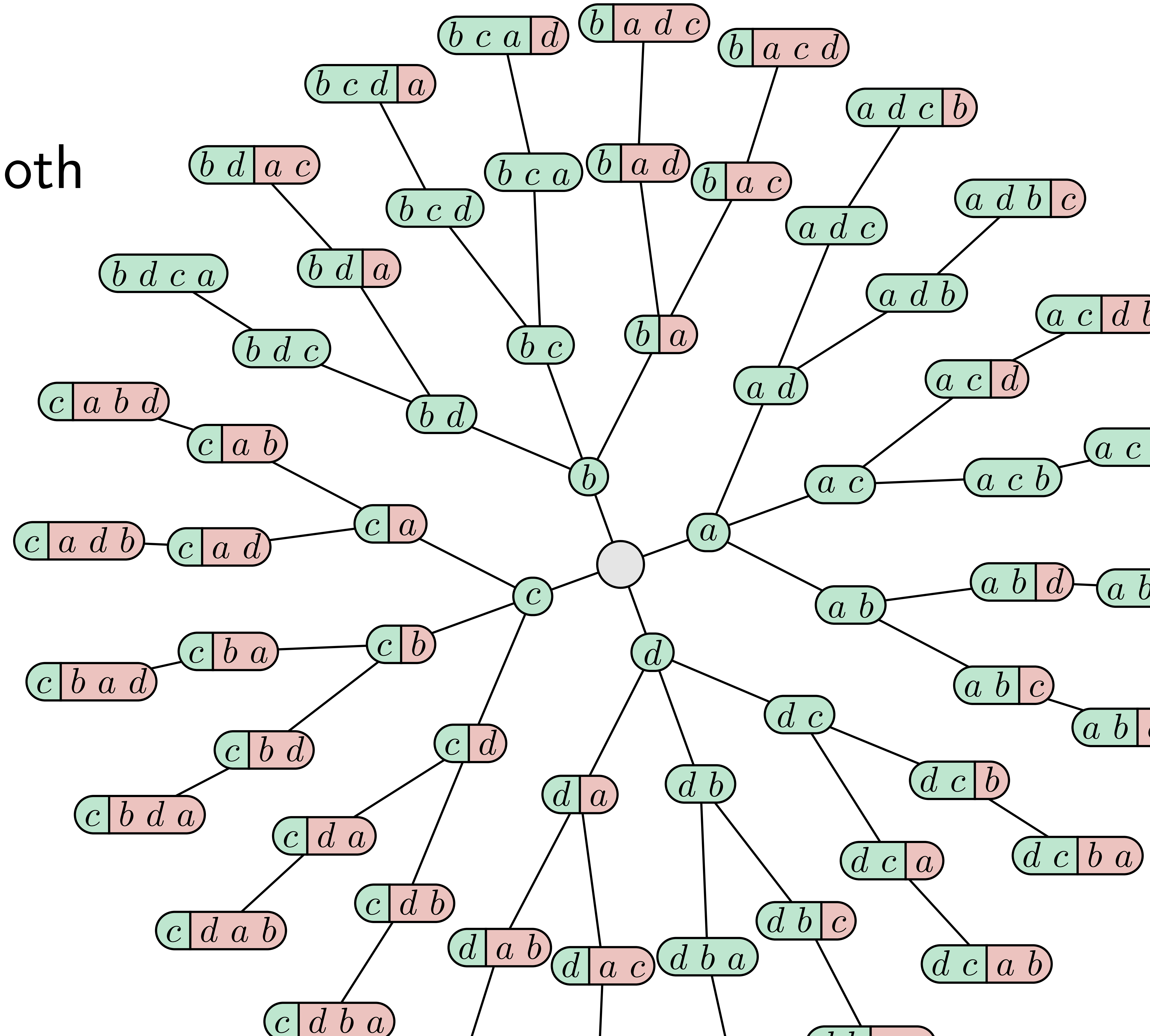
Snowcap performs network reconfigurations automatically and safely



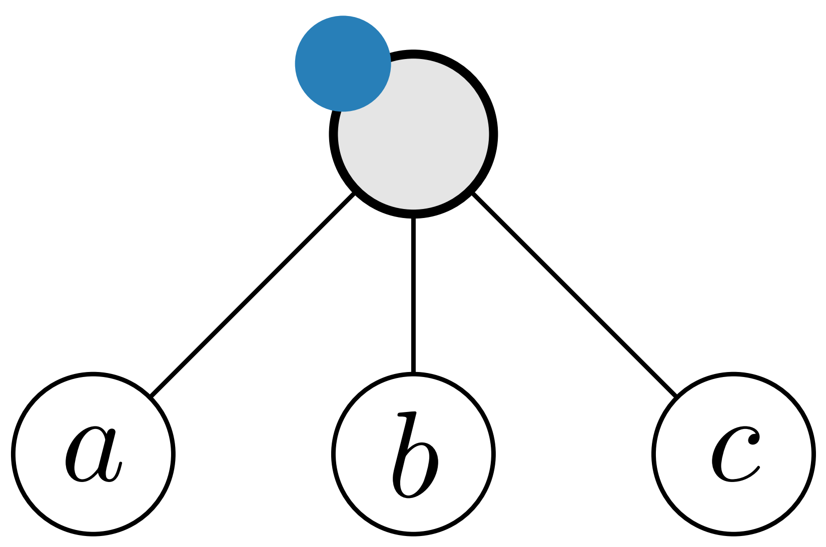
It's all about navigating the search space of possible reconfiguration orderings

The search space is both

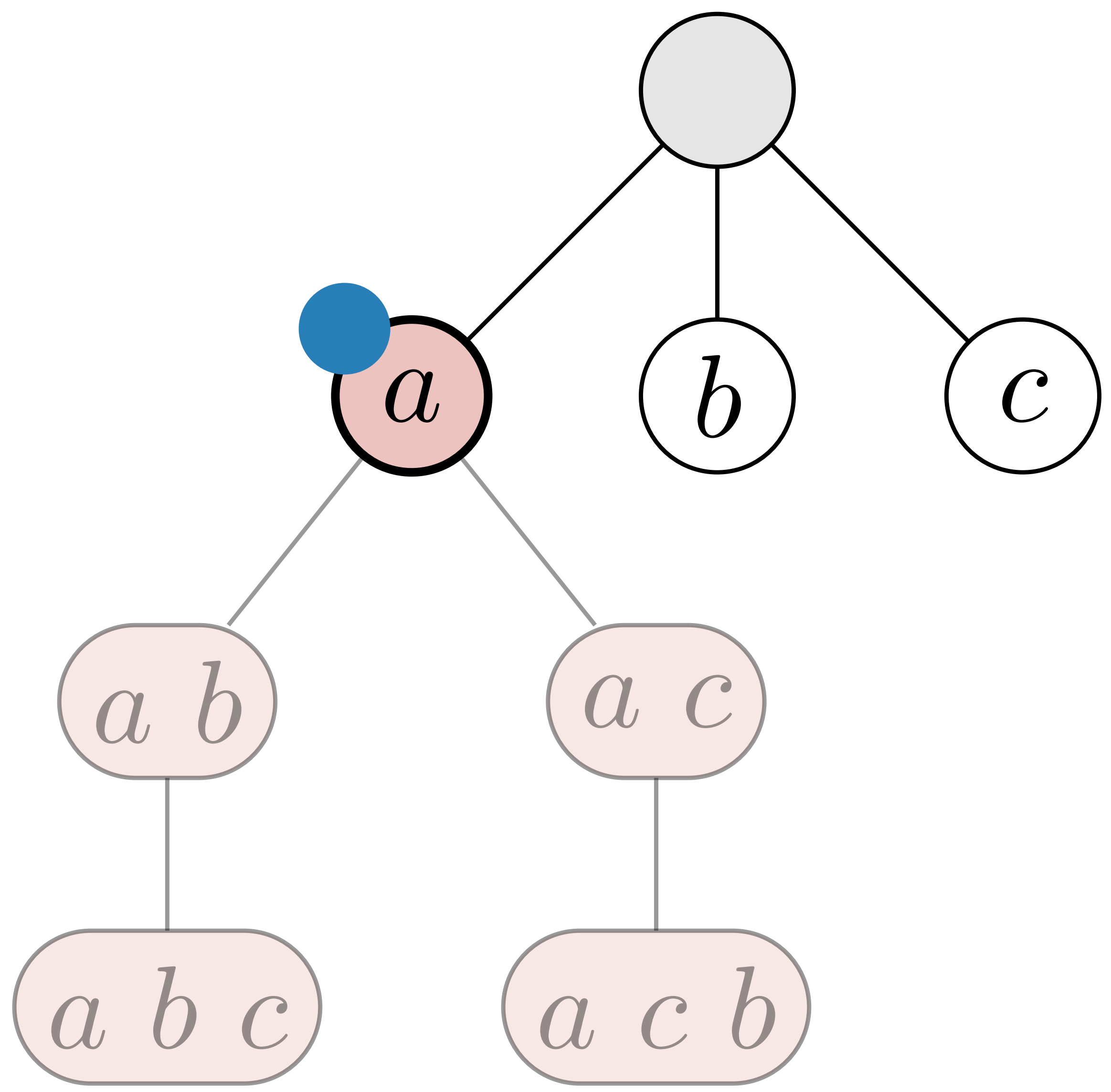
- **sparse**; and
- **huge**.



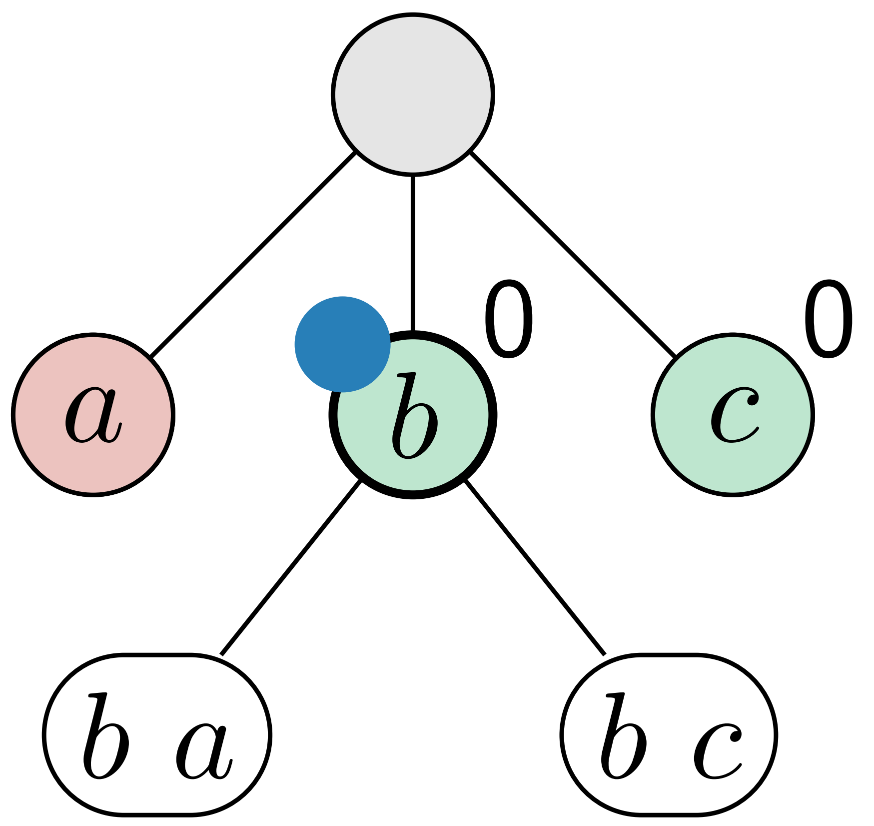
The exploration algorithm is based on DFS traversal



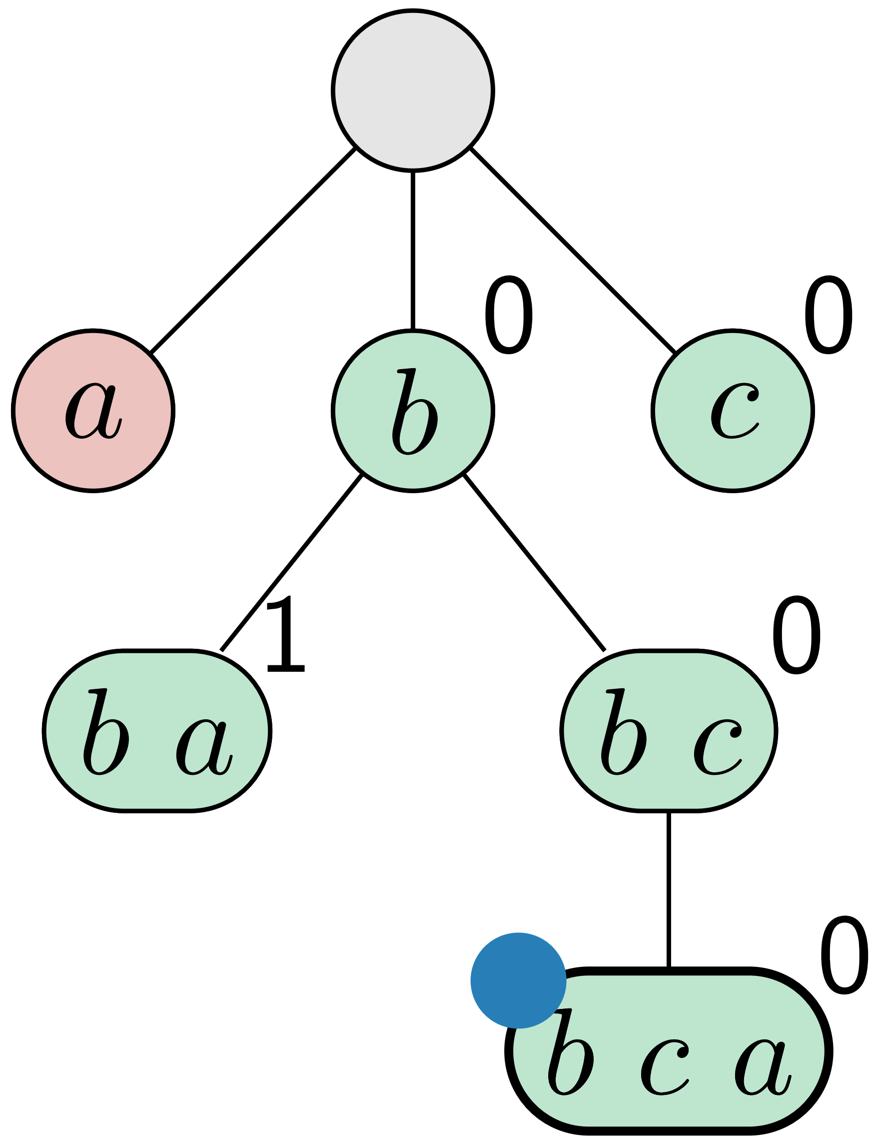
Sequences with a known, bad prefix are not explored



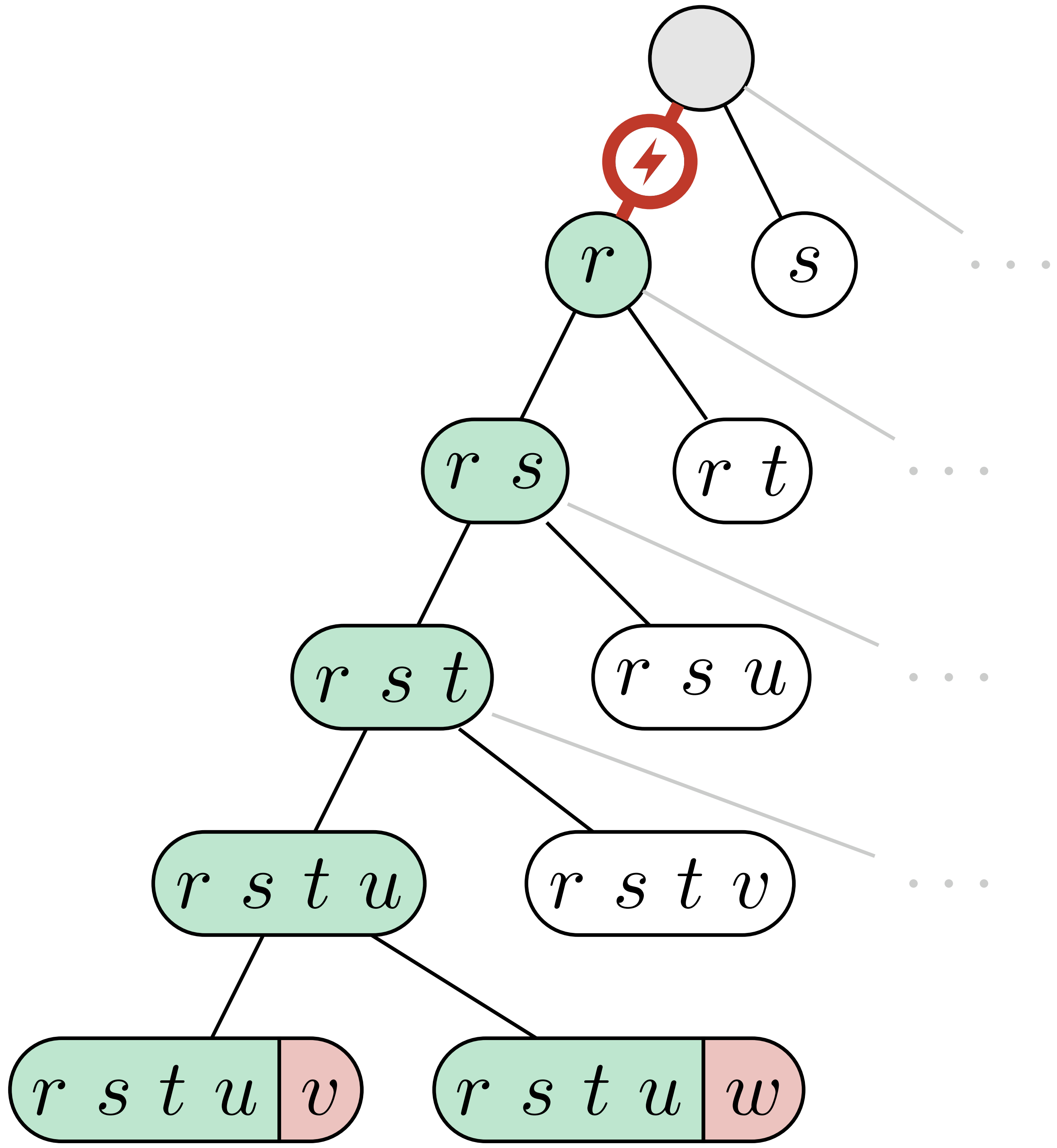
Greedy minimization of the cost function



Greedy minimization of the cost function



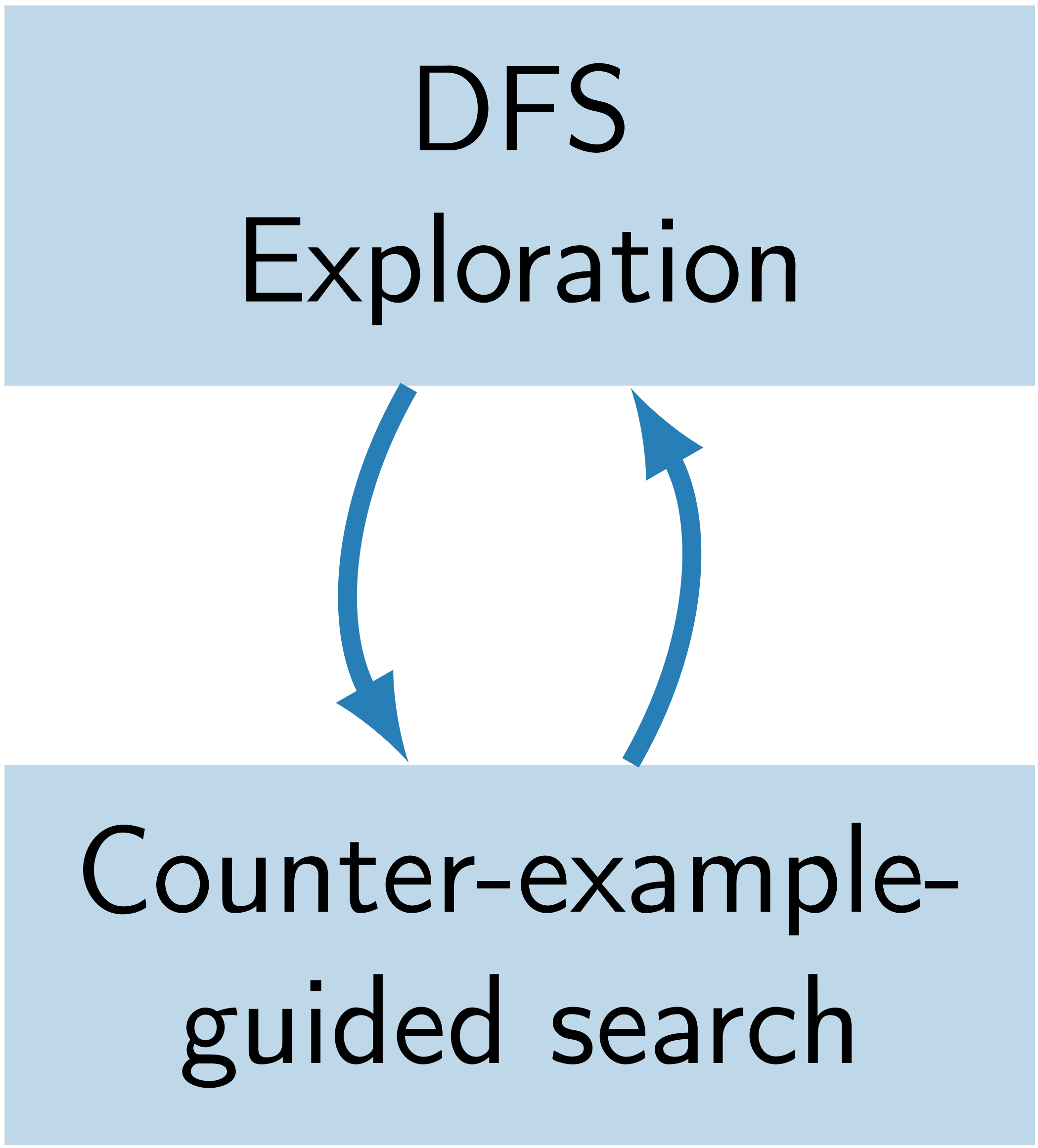
DFS Exploration works well in *most* cases



However: What if we get stuck? Bad decision **early** may cause problems **later**.

→ Actively find the problem!

Snowcap uses counter-example-guided search to resolve difficult dependencies



Snowcap . . .

- performs normal exploration **until a dead end**
- follows a **divide-and-conquer** approach

We evaluate Snowcap on a wide range of topologies and migration scenarios

- $\approx$ 80 Topologies from Topology Zoo
- Common migration scenarios
- Random link weights and iBGP topologies.

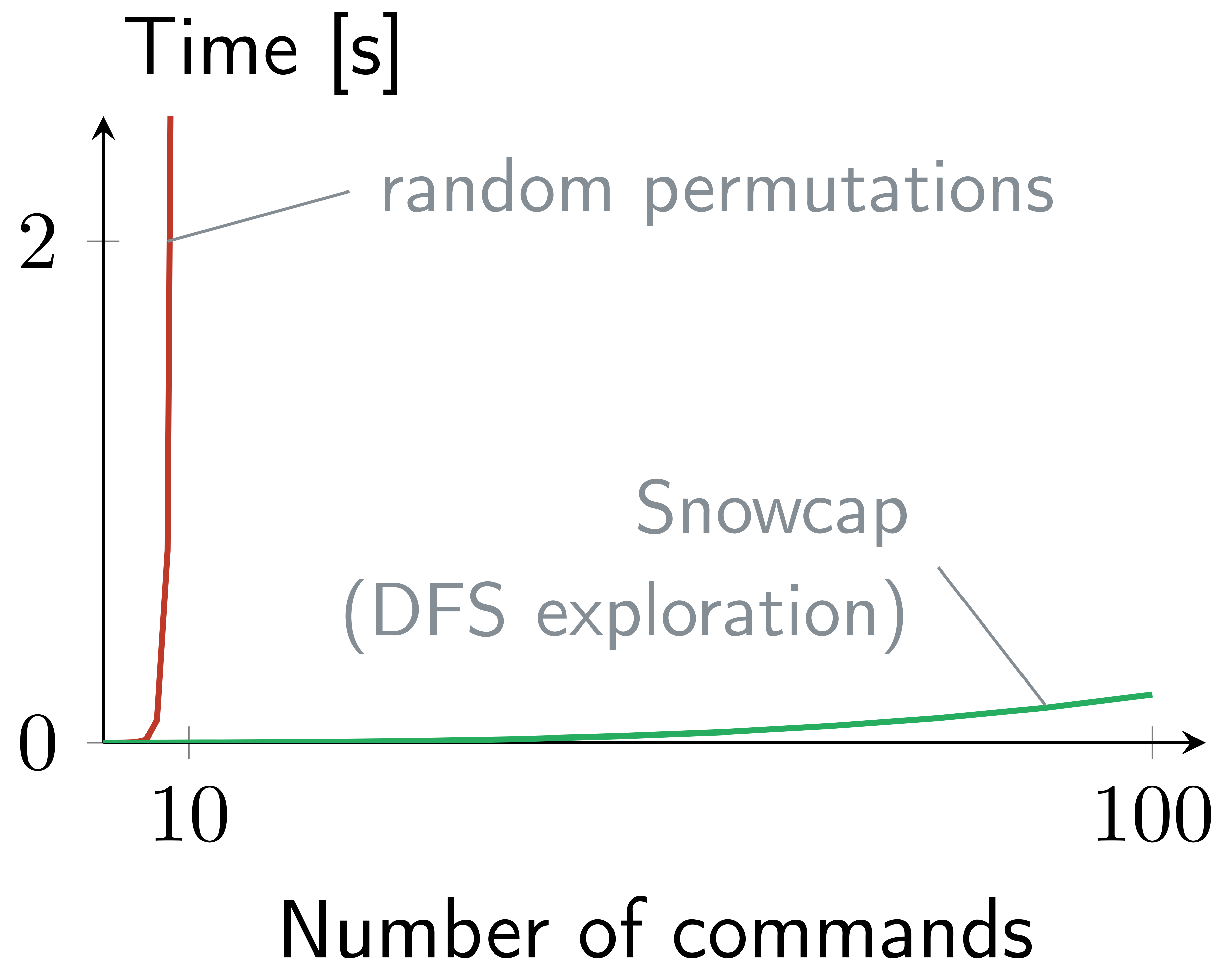
Snowcap finds solutions within seconds

Migration from iBGP full-mesh to route-reflection.

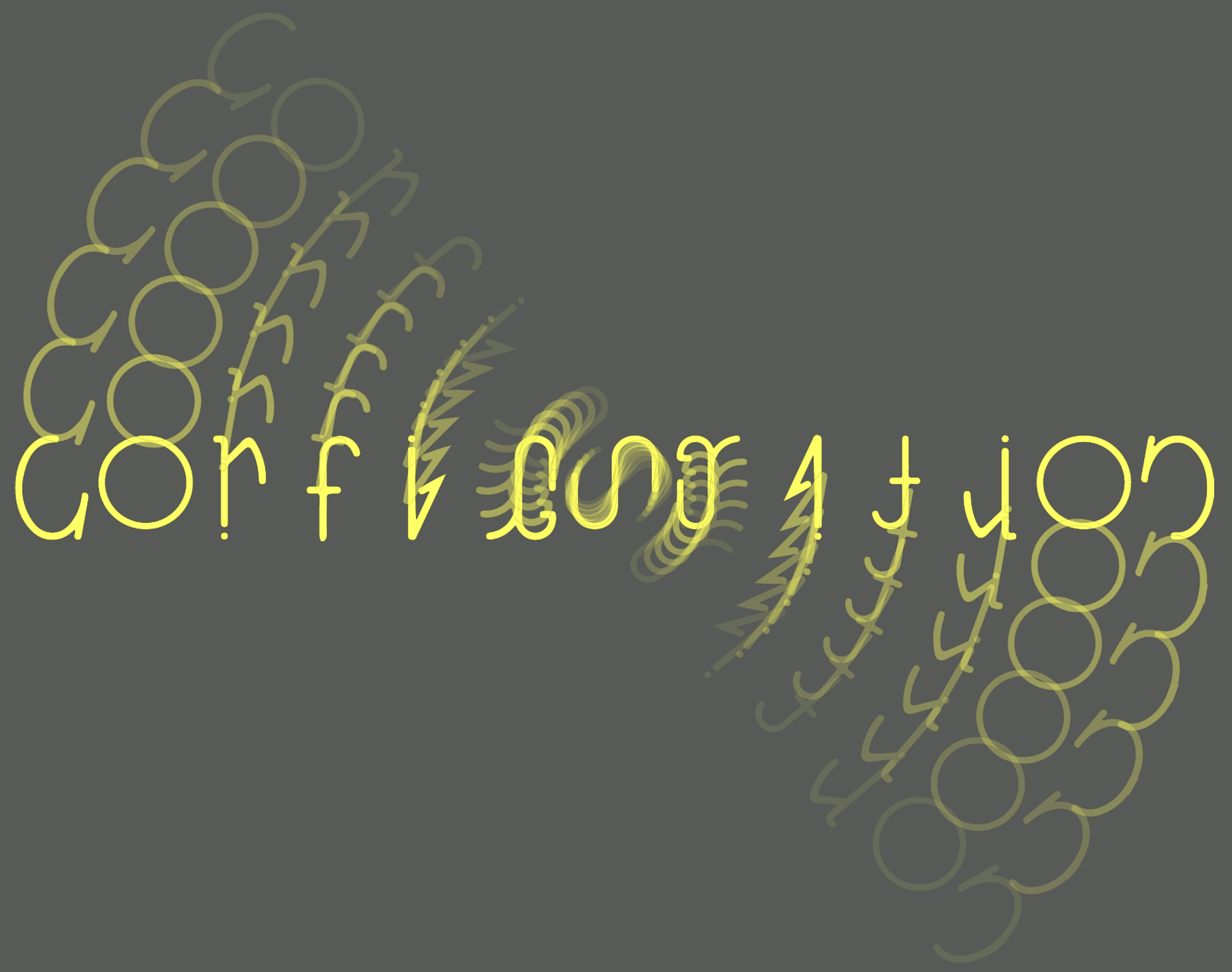
	$\geq 50\%$ chance to violate reachability	time
Random order	$\mathbf{70\%}$	
Best practice order	$\mathbf{25\%}$	
Snowcap	$\mathbf{0\%}$	at most $12s^*$

*for 3081 commands on 82 routers.

Snowcap's runtime scales very well with increasing complexity



The three tales of (correct) network operations



Verification

going forward

Synthesis

going backward

Reconfiguration

going sideways

We have only scratched the surface when it comes to analyzing network computation

Complexity

Simplicity

Learnability

We have only scratched the surface when it comes to analyzing network computation

Complexity

Simplicity

Learnability

What's the computational complexity of configuration verification and synthesis?

Yes. SMT solving works, but is it *really* needed?

We have only scratched the surface when it comes to analyzing network computation

Complexity

Simplicity

What's the *simplest* computation that can do it all?

and hopefully is easier to verify / synthesize for?

Learnability

We have only scratched the surface when it comes to analyzing network computation

Complexity

Simplicity

Learnability — Can we *learn* how to invert network computations?

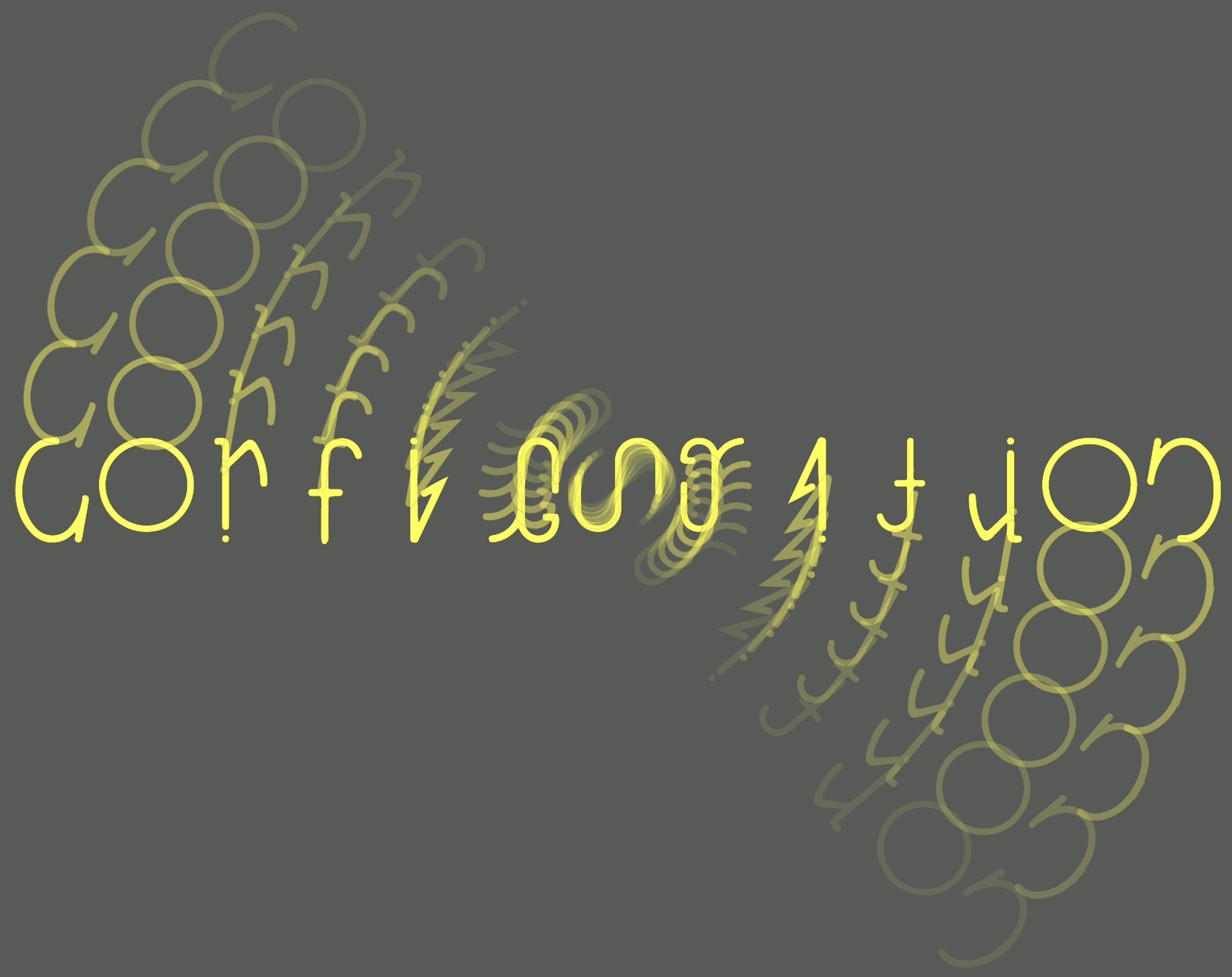
instead of writing inverse models by hands


Robert College
reusch
Campbell's
TOMATO

Merci à tous!

+ all NSG alumnis, collaborators, mentors (esp. Olivier Bonaventure and Jennifer Rexford), and colleagues!!

The three tales of (correct) network operations



Laurent Vanbever

nsg.ee.ethz.ch

CoNEXT

Wed Dec 8 2021