The three tales of (correct) network operations

Laurent Vanbever
nsg.ee.ethz.ch

CoNEXT
Wed Dec 8 2021
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.

[...]
Seamless Network-Wide IGP Migrations

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

* Université catholique de Louvain 1 Roma Tre University 2 Internet Initiative Japan
{(laurent.vanbever, pierre.francois, olivier.bonaventure) @uclouvain.be
vissicchio@dia.uniroma3.it cristel@iiij.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 losing reachability?
initial forwarding state
initial forwarding state

final forwarding state
How do you reconfigure a network without losing reachability?
initial forwarding state

intermediate forwarding state

final forwarding state
What if we reconfigure D first?
What if we reconfigure D first?
We create a forwarding loop

What if we reconfigure D first?

We create a forwarding loop
initial forwarding state

intermediate forwarding state

final forwarding state
What if we reconfigure C first?
What if we reconfigure C first?  
Works!
How do you reconfigure a network without losing reachability?

This was easy to compute for one destination, but...
How do you reconfigure a network without losing 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
Seamless Network-Wide IGP Migrations

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

* Université catholique de Louvain 1 Roma Tre University 2 Internet Initiative Japan
{(laurent.vanbever, pierre.francois, olivier.bonaventure) @ uclouvain.be
vissicchio@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 evolution 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
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

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

Figure 1: This scenario consists of adding an eBGP session a and adapting two link weights: b and c, while: (i) ensuring traffic from r2 always flows via r3; and (ii) minimizing traffic shifts. Two orderings achieve both goals: b2c and c2b.
Have we just come full circle?
Seamless Network-Wide IGP Migrations

Laurent Vanberven, Stefano Vissicchio, Cristel Peltier, Pierre François, Olivier Bonaventure

Université catholique de Louvain / Roma Tre University / Internet Initiative Japan

Abstract
Network-wide migrations of a running network, such as the deployment of a new network protocol or the alteration of its configuration, require careful planning to avoid disrupting service and causing security issues. However, the process of deploying an IGP migration is complicated by the need to coordinate updates across the entire network. In this paper, we propose a methodology that addresses the problem of smoothly modifying the configuration of a network-wide Interior Gateway Protocol (IGP). We illustrate the benefits of our methodology by considering the upgrade to Open Shortest Path First (OSPF) over an existing 802.11x, with minimal disruption to traffic flows. Our approach involves a staged migration process that minimizes the impact on the network by sequentially updating different subsets of the infrastructure. This approach allows for a controlled and gradual transition, ensuring that network traffic is not disrupted and minimizing the risk of service outages.

SIGCOMM 2011

Snowcast: Synthesizing Network-Wide Configuration Updates

Tiber Schneider, ETIS, Dandia, Switzerland

Ridger Bieden, ETIS, Dandia, Switzerland

Laurent Vanberven, ETIS, Dandia, Switzerland

Abstract
Large-scale configuration campaigns tend to be slow and costly for network operators as they can lead to significant network downtime, decreased performance, and policy violations. Furthermore, existing migration frameworks often fall short in practice as they are either too complex or just a single migration scenario or simply do not scale. We address these problems with snowcast, the first network-wide migration framework that 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 arrange route-examples and network-routing configurations ordering which validates the desired specifications, our algorithm automatically identifies the problematic constraints so that it can avoid this particular order in the next iteration. We fully implemented snowcast and extensively evaluated its scalability and effectiveness on real-world topologies and protocols, large-scale migration scenarios. For large topologies, snowcast finds a valid migration ordering with minimal side-effects (e.g., traffic re-flows) within few seconds at most.

CSC Concepts:
• Networks – Network management; Network reliability; Network configuration; • Theory of computation – Model and temporal logic; Logic and verification;

Keywords
ACM Reference Format

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Figure 1. This scenario consists of adding an eBGP session to a and adapting two link weights: a1 and a2. While (unloading traffic from a2, always flows over eBGP, and b1 remaining traffic for eBGP sessions). Two orders achieve both goals. The and

SIGCOMM 2021
Seamless Network-Wide IGP Migrations

Laurent Vanbever, Stefano Vlassischo, Cristel Pelletier, Pierre Franczos, Olivier Bonaventure

Université catholique de Louvain / Rome University / Internet Initiative Japan

vlassischo@dia.uolouvain2l.be / pelletier@i2r.2louvrain2lbe / franczos@i2r.2louvrain2lbe / bonaventure@i2r.2louvrain2lbe

ABSTRACT

Network-wide migration of a running network, such as the migration of a routing protocol or the configuration of a network, can have serious consequences for network availability, manageability, and stability of the network system. However, each network component is responsible for its own migration, which means that the coordination required by a network-wide migration is out of reach of standard IGP operations. To address this problem, we propose a novel approach for network-wide IGP migrations. Our approach is based on the use of a single, general, and expressive tool, called Cross Site Communication Network (XCN), which provides a general framework for coordinating and managing network-wide migrations. Our tool allows for the description of a network-wide migration as a set of commands that can be executed in a coordinated manner.

In the next section, we describe the IGP migration process in detail, using a case study to illustrate the potential of the tool. We also discuss the limitations and challenges of implementing the tool in practice.

CSC Concepts:
Networks, Network management

Keywords:
Cross Site Communication Network

1. INTRODUCTION

Many network components have their own independent IGP formation or IGP operation, which means that each component cannot be coordinated in a unified manner. The coordination of a network-wide migration, on the other hand, can be achieved through a unified management tool that can be used to coordinate the migration of all network components. Our proposal is based on the use of a single, general, and expressive tool, called Cross Site Communication Network (XCN), which provides a general framework for coordinating and managing network-wide migrations. Our tool allows for the description of a network-wide migration as a set of commands that can be executed in a coordinated manner.

In the next section, we describe the IGP migration process in detail, using a case study to illustrate the potential of the tool. We also discuss the limitations and challenges of implementing the tool in practice.

Figure 1: This scenario consists of adding and removing links 10 and 11, and adapting traffic weights 1, 5 and 2, while (1) ensuring network operability, (2) maintaining traffic flow, (3) adapting network components to changing demands, and (4) monitoring network performance. The XCN framework enables network operators to manage these tasks efficiently and effectively.

1 INTRODUCTION

Network operators configure their network locally (e.g., by adding or removing links and nodes, by changing link weights, etc.). As a result, the network configuration is complex and difficult to manage. In addition, network operators are often required to maintain the network in a stable and efficient manner, which can be challenging due to the complexity of the network configuration. The goal of this study is to develop a framework for the efficient and effective management of network-wide IGP migrations. Our framework is based on the use of a single, general, and expressive tool, called Cross Site Communication Network (XCN), which provides a general framework for coordinating and managing network-wide migrations. Our tool allows for the description of a network-wide migration as a set of commands that can be executed in a coordinated manner.

In the next section, we describe the IGP migration process in detail, using a case study to illustrate the potential of the tool. We also discuss the limitations and challenges of implementing the tool in practice.

Figure 1: This scenario consists of adding and removing links 10 and 11, and adapting traffic weights 1, 5 and 2, while (1) ensuring network operability, (2) maintaining traffic flow, (3) adapting network components to changing demands, and (4) monitoring network performance. The XCN framework enables network operators to manage these tasks efficiently and effectively.

1 INTRODUCTION

Network operators configure their network locally (e.g., by adding or removing links and nodes, by changing link weights, etc.). As a result, the network configuration is complex and difficult to manage. In addition, network operators are often required to maintain the network in a stable and efficient manner, which can be challenging due to the complexity of the network configuration. The goal of this study is to develop a framework for the efficient and effective management of network-wide IGP migrations. Our framework is based on the use of a single, general, and expressive tool, called Cross Site Communication Network (XCN), which provides a general framework for coordinating and managing network-wide migrations. Our tool allows for the description of a network-wide migration as a set of commands that can be executed in a coordinated manner.

In the next section, we describe the IGP migration process in detail, using a case study to illustrate the potential of the tool. We also discuss the limitations and challenges of implementing the tool in practice.

Figure 1: This scenario consists of adding and removing links 10 and 11, and adapting traffic weights 1, 5 and 2, while (1) ensuring network operability, (2) maintaining traffic flow, (3) adapting network components to changing demands, and (4) monitoring network performance. The XCN framework enables network operators to manage these tasks efficiently and effectively.

1 INTRODUCTION

Network operators configure their network locally (e.g., by adding or removing links and nodes, by changing link weights, etc.). As a result, the network configuration is complex and difficult to manage. In addition, network operators are often required to maintain the network in a stable and efficient manner, which can be challenging due to the complexity of the network configuration. The goal of this study is to develop a framework for the efficient and effective management of network-wide IGP migrations. Our framework is based on the use of a single, general, and expressive tool, called Cross Site Communication Network (XCN), which provides a general framework for coordinating and managing network-wide migrations. Our tool allows for the description of a network-wide migration as a set of commands that can be executed in a coordinated manner.

In the next section, we describe the IGP migration process in detail, using a case study to illustrate the potential of the tool. We also discuss the limitations and challenges of implementing the tool in practice.

Figure 1: This scenario consists of adding and removing links 10 and 11, and adapting traffic weights 1, 5 and 2, while (1) ensuring network operability, (2) maintaining traffic flow, (3) adapting network components to changing demands, and (4) monitoring network performance. The XCN framework enables network operators to manage these tasks efficiently and effectively.

1 INTRODUCTION

Network operators configure their network locally (e.g., by adding or removing links and nodes, by changing link weights, etc.). As a result, the network configuration is complex and difficult to manage. In addition, network operators are often required to maintain the network in a stable and efficient manner, which can be challenging due to the complexity of the network configuration. The goal of this study is to develop a framework for the efficient and effective management of network-wide IGP migrations. Our framework is based on the use of a single, general, and expressive tool, called Cross Site Communication Network (XCN), which provides a general framework for coordinating and managing network-wide migrations. Our tool allows for the description of a network-wide migration as a set of commands that can be executed in a coordinated manner.

In the next section, we describe the IGP migration process in detail, using a case study to illustrate the potential of the tool. We also discuss the limitations and challenges of implementing the tool in practice.

Figure 1: This scenario consists of adding and removing links 10 and 11, and adapting traffic weights 1, 5 and 2, while (1) ensuring network operability, (2) maintaining traffic flow, (3) adapting network components to changing demands, and (4) monitoring network performance. The XCN framework enables network operators to manage these tasks efficiently and effectively.

1 INTRODUCTION

Network operators configure their network locally (e.g., by adding or removing links and nodes, by changing link weights, etc.). As a result, the network configuration is complex and difficult to manage. In addition, network operators are often required to maintain the network in a stable and efficient manner, which can be challenging due to the complexity of the network configuration. The goal of this study is to develop a framework for the efficient and effective management of network-wide IGP migrations. Our framework is based on the use of a single, general, and expressive tool, called Cross Site Communication Network (XCN), which provides a general framework for coordinating and managing network-wide migrations. Our tool allows for the description of a network-wide migration as a set of commands that can be executed in a coordinated manner.

In the next section, we describe the IGP migration process in detail, using a case study to illustrate the potential of the tool. We also discuss the limitations and challenges of implementing the tool in practice.

Figure 1: This scenario consists of adding and removing links 10 and 11, and adapting traffic weights 1, 5 and 2, while (1) ensuring network operability, (2) maintaining traffic flow, (3) adapting network components to changing demands, and (4) monitoring network performance. The XCN framework enables network operators to manage these tasks efficiently and effectively.
Seamless Network-Wide IGP Migrations

Laurent Vanhever, Stéphane Vlasschok, Cristel Peltier, Pierre Françoise, Olivier Bonaventure

Université catholique de Louvain / Rome 1 University / Internet Initiative Japan
(laurent.vanhever,pierre.franoise,olivier.bonaventure)@ulb.ac.be

vs

Snowcap: Synthesizing Network-Wide Configuration Updates

Tobias Schneider
ETH Zurich, Switzerland

Rüdiger Rehman
ETH Zurich, Switzerland

Laurent Vanhever
ETH Zurich, Switzerland

lvanhever@ethz.ch

ABSTRACT

Large-scale network configuration is an ever-evolving task for network operators as they need to deploy significant network changes, which are often performed in parallel, to avoid service disruptions. Existing configuration frameworks often fail short in practice as they only consider individual best-effort configuration sessions. In this paper, we present Snowcap, a distributed network configuration tool that enables operators to reason about the configuration of distributed network computations as simple as daily code commits.

1 INTRODUCTION

Network operators are facing a significant daily (10-30) configuration task. For example, network operators may deploy new services that will be available for next day (24). These services are often deployed in parallel, which means that their deployment must consider the interdependencies between them.

vs

general

more...

expressive
efficient

reason about distributed network computations
reason about

distributed network computations
Distributed computations rule over network forwarding behavior
distributed algorithms
distributed algorithms \rightarrow \text{per-device forwarding state } \mathcal{F} \rightarrow \text{outputs}
distributed algorithms

\( C \) per-device configurations

\( T \) topology

\( R \) external routes

\( \mathcal{F} \) per-device forwarding state

inputs

outputs
network operators

distributed algorithms

c \rightarrow \mathcal{T} \rightarrow \mathcal{R} \rightarrow \mathcal{F}

per-device configurations

topology

external routes

inputs

outputs

per-device forwarding state
Network operators

High-level specification \( \varphi \)

Inputs
- Per-device configurations \( C \)
- Topology \( T \)
- External routes \( R \)

Outputs
- Per-device forwarding state \( F \)
network operators

high-level specification $\varphi$

distributed algorithms

c
$\Gamma$
$\mathcal{R}$

per-device configurations

topology

external routes

inputs

outputs

per-device forwarding state $\mathcal{F}$
- topology
- external routes

MIND THE GAP

C

\( R \)

\( \mathcal{D} \)

network operators

high-level specification \( \mathcal{O} \)

per-device forwarding state \( \mathcal{F} \)

inputs

outputs

per-device configurations
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! 😊
network operators

MIND THE GAP

per-device configurations

topology

external routes

c \( \mathcal{C} \)

\( \mathcal{R} \) \rightarrow distributed algorithms

\( \mathcal{F} \) \rightarrow per-device forwarding paths

high-level specification \( \varphi \)
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** $\varphi$
- Verification
- Synthesis
- Reconfiguration
We've aimed at helping operators bridging this gap considering three directions

Given specification $\varphi$
and

Verification configuration $C$

Synthesis

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

- **Given specification** \( \varphi \)
- **Verification** configuration \( C \)
- **Synthesis**
- **Reconfiguration**
- **Return**
We've aimed at helping operators bridging this gap considering three directions.

Verification configuration $C$

Synthesis

Reconfiguration

Given specification $\varphi$ and

Return $C \models \varphi$
We've aimed at helping operators bridging this gap considering three directions:

- **Verification**
  - Given specification $\varphi$
  - Configuration $C$
  - $C \models \varphi$

- **Synthesis**
  - $\emptyset$

- **Reconfiguration**
We've aimed at helping operators bridging this gap considering three directions:

<table>
<thead>
<tr>
<th>Verification</th>
<th>Given specification $\varphi$ and configuration $C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthesis</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>Reconfiguration</td>
<td>initial and final configuration $C_i, C_a, C_b, \ldots C_f \models \varphi$</td>
</tr>
</tbody>
</table>

Given a specification $\varphi$, we aim to find a configuration $C$ that satisfies $\varphi$. If $C \models \varphi$, we consider the initial configuration $C_i$. If $C \not\models \varphi$, we move to the next configuration $C_a$, until we find a configuration $C_f$ that satisfies $\varphi$. The process of moving configurations is denoted by the arrows $\Rightarrow$. The final configuration $C_f$ is verified to satisfy $\varphi$. This approach helps operators bridge the gap between given specifications and returned configurations.
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
2. Synthesis  
going backward
3. Reconfiguration  
going sideways
Probabilistic Verification of Network Configurations

Samuel Steffen
Timon Gehr
Petar Tsankov
Laurent Vanbever
Martin Vechev

ETHzürich
Networked Systems
SRILAB
What is the probability of '.middle'?
**Probabilistic Verification**

What is the *probability* of \( \text{\text{}} \) ?

- **Service Level Agreements (SLA)**
  "99.99% reachability"

- **Traffic Engineering**
  "80% load-balanced"
Probabilistic Verification

What is the probability of ?

Service Level Agreements (SLA)
“99.99% reachability”

Traffic Engineering
“80% load-balanced”

probabilistic

high precision required
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

Partial exploration

1 107 359

#scenarios for four 9s, 191 links, p_{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

Partial exploration

Estimation via sampling

1 107 359 #scenarios for four 9s, 191 links, \( p_{\text{link failure}} = 0.001 \)

738 M Hoeffding, \( \alpha = 0.95 \)

Too expensive
Attempts: Exploring Failures

Partial exploration

Estimation via sampling

1,107,359

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

Too expensive

738 M

Hoeffding, $\alpha = 0.95$

≈600x reduction

Net Dice

1,854

Too expensive
Overview

Net Dice

- BGP + IGP support
- High accuracy
- Scalable
Pruning Failures
Key Idea

shortest paths
Key Idea

shortest paths
Key Idea

shortest paths
Key Idea

shortest paths
Key Idea

shortest paths
Key Idea

shortest paths
Key Idea

shortest paths

cold edges
Key Idea

Scenarios with same forwarding graph (32 total):

shortest paths

cold edges
Key Idea

Scenarios with same forwarding graph (32 total):

shortest paths

How to find these?

cold edges
for BGP

Algorithm 3 Hot edges for BGP

1: procedure HotBGP(u, d, E_{fwd}, L)
2: $X \leftarrow$ nodes in the same partition as $u$ under $L$
3: $BRL \leftarrow$ Top3($BR, X$) \Comment{BGP pre-processing (§4.2)}
4: $RRL \leftarrow RR \cap X$
5: $H \leftarrow$ AllSp($RRL, BRL, L$) \Comment{all shortest paths (Alg. 2)}
6: $D \leftarrow \{ u \}$ \Comment{decision points}
7: $\cup \{ y \mid (x, y) \in Static_d \cap E_{fwd} \}$
8: $\cup \{ y \mid (x, y) \in E_{fwd} \land NH_d(x) \neq NH_d(y) \}$
9: for each $x \in D$ do
10: $H \leftarrow H \cup SP_{L}(x, NH_d(x))$ \Comment{shortest path $x \rightarrow NH_d(x)$}
11: $H \leftarrow H \cup (\text{Static}_d \cap E_{fwd})$ \Comment{traversed static routes}
12: if $RRL = \emptyset$ then
13: $H \leftarrow H \cup \text{AllSp}(\{ u \}, BRL)$ \Comment{ensure connectivity}
14: return $H$
Algorithm 3: Hot edges for BGP

1: procedure HotBGP(u, d, E\text{fwd}, L)
2:  \( X \leftarrow \) nodes in the same partition as \( u \) under \( L \)
3:  \( BRL \leftarrow \text{Top3}(BR, X) \) \( \triangleright \) BGP pre-processing (§4.2)
4:  \( RRL \leftarrow RR \cap X \)
5:  \( H \leftarrow \text{AllSp}(RRL, BRL, L) \) \( \triangleright \) all shortest paths (Alg. 2)
6:  \( D \leftarrow \{u\} \) \( \triangleright \) decision points
7:  \( \cup \left\{ y \mid (x, y) \in \text{STATIC}_d \cap E_{\text{fwd}} \right\} \)
8:  \( \cup \left\{ y \mid (x, y) \in E_{\text{fwd}} \land NH_d(x) \neq NH_d(y) \right\} \)
9:  \textbf{for each} \( x \in D \) \textbf{do}
10:  \( H \leftarrow H \cup \text{SP}_L(x, NH_d(x)) \) \( \triangleright \) shortest path \( x \rightarrow NH_d(x) \)
11:  \( H \leftarrow H \cup (\text{STATIC}_d \cap E_{\text{fwd}}) \) \( \triangleright \) traversed static routes
12:  \textbf{if} \( RRL = \emptyset \) \textbf{then}
13:  \( H \leftarrow H \cup \text{AllSp}(\{u\}, BRL) \) \( \triangleright \) ensure connectivity
14:  \textbf{return} \( H \)
for BGP

Algorithm 3 Hot edges for BGP

1: procedure HotBgp(u, d, E_{fwd}, L)
2: \( X \leftarrow \) nodes in the same partition as \( u \) under \( L \)
3: \( BRL \leftarrow \text{Top3}(Br, X) \)  \( \triangleright \) BGP pre-processing (§4.2)
4: \( RRL \leftarrow RR \cap X \)
5: \( \mathcal{H} \leftarrow \text{AllSp}(RRL, BRL, L) \)  \( \triangleright \) all shortest paths (Alg. 2)
6: \( \mathcal{D} \leftarrow \{u\} \)  \( \triangleright \) decision points
7: \( \cup \{y \mid (x, y) \in \text{STATIC}_d \cap E_{fwd}\} \)
8: \( \cup \{y \mid (x, y) \in E_{fwd} \land NH_d(x) \neq NH_d(y)\} \)
9: for each \( x \in \mathcal{D} \) do
10: \( \mathcal{H} \leftarrow \mathcal{H} \cup \text{Sp}_L(x, NH_d(x)) \)  \( \triangleright \) shortest path \( x \rightarrow NH_d(x) \)
11: \( \mathcal{H} \leftarrow \mathcal{H} \cup (\text{STATIC}_d \cap E_{fwd}) \)  \( \triangleright \) traversed static
12: if \( RRL = \emptyset \) then
13: \( \mathcal{H} \leftarrow \mathcal{H} \cup \text{AllSp}(\{u\}, BRL) \)  \( \triangleright \) ensure connectivity
14: return \( \mathcal{H} \)

see paper
network partitions
route reflection
dependence on IGP costs
with correctness proof
Failure Exploration

Sum up $P(\_\_\_\_\_\_\_\_\_\_\_\_\_)$
Failure Exploration

Sum up $P(\ )$

"Cut off" unlikely scenarios
Failure Exploration

- Sum up $P(\square)$
- Efficiency depends on #
- "Cut off" unlikely scenarios

Very efficient in practice
Implementation

[Diagram with the following text:

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

Additional text:

nsg-net/hz/netdice

---

Implementation

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

Single-flow (e.g. Reachability)

Few minutes for 100s of links for four 9s

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

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

Also analyzed real ISP config

![Graph showing performance degradation over number of flows](image)
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

NetDice is precise and efficient

Also analyzed real ISP config

Runtime
The three tales of (correct) network operations

Verification
  going forward

Synthesis
  going backward

Reconfiguration
  going sideways
NetComplete: Practical Network-Wide Configuration Synthesis with Autocompletion

Ahmed El-Hassany  Petar Tsankov  Martin Vechev  Laurent Vanbever

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

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

ip community-list C1 permit 6500:1
ip community-list C2 permit 6500:2
NetComplete reduces the autocompletion problem to a *constraint satisfaction problem*.
First, encode the following:
- protocol semantics
- high-level requirements as a logical formula (in SMT)
- partial configurations
Use a solver (Z3) to find an assignment for the undefined configuration variables s.t. the formula evaluates to True

First

- protocol semantics
- Encode the high-level requirements as a logical formula (in SMT)
- partial configurations

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

\[\text{Cost}(A \rightarrow C) = \text{Cost}(A \rightarrow D \rightarrow C) < \text{Cost}(X)\]
∀ \( X \in \text{Paths}(A,C) \setminus \text{Reqs} \)

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

Solve
∀ X ∈ Paths(A,C) \ Reqs
Cost(A → C) = Cost(A → D → C) < Cost(X)

Solve
∀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...

\[ \forall X \in \text{Paths}(A,C) \backslash \text{Reqs} \]

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

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 $Y$

<table>
<thead>
<tr>
<th>CEGIS</th>
<th>Consider a random subset $S$ of them and synthesize the weights considering $S$ only</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part 1</td>
<td></td>
</tr>
<tr>
<td>intuition</td>
<td><strong>Fast as $S$ is small compared to all paths</strong></td>
</tr>
</tbody>
</table>
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**

**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
∀ \(X \in \text{SamplePaths}(A,C) \setminus \text{Reqs}\)
∀ \( X \in \text{SamplePaths}(A,C) \setminus \text{Reqs} \)

Sample: \{ [A,B,D,C] \}
∀ \( X \in \text{SamplePaths}(A,C) \setminus \text{Reqs} \)

\[
\text{Cost}(A \rightarrow C) = \text{Cost}(A \rightarrow D \rightarrow C) < \text{Cost}(X)
\]
∀ X ∈ SamplePaths(A,C)\\Reqs
Cost(A → C) = Cost(A → D → C) < Cost(X)
Solve
∀X ∈ SamplePaths(A,C) \ Reqs

Cost(A → C) = Cost(A → D → C) < Cost(X)

Solve
∀ \( X \in \text{SamplePaths}(A,C) \setminus \text{Reqs} \)

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

Solve
The synthesized weights are incorrect:
\[
\text{cost}(A \rightarrow B \rightarrow C) = 250 < \text{cost}(A \rightarrow C) = 300
\]

\[
\forall X \in \text{SamplePaths}(A,C) \text{\textbackslash Reqs}
\]

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

Solve
We simply add the counter example to SamplePaths and repeat the procedure.
The entire procedure usually converges in few iterations making it very fast in practice.

<table>
<thead>
<tr>
<th>Network size</th>
<th>Reqs. type</th>
<th>Synthesis time</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSPF synthesis time (sec)</td>
<td>Large</td>
<td>Simple</td>
</tr>
<tr>
<td>~150 nodes</td>
<td>ECMP</td>
<td>14s</td>
</tr>
<tr>
<td>Ordered</td>
<td>13s</td>
<td></td>
</tr>
<tr>
<td>settings</td>
<td>249s</td>
<td></td>
</tr>
</tbody>
</table>

16 reqs, 50% symbolic, 5 repet.
CEGIS enabled
The three tales of (correct) network operations

Verification
going forward

Synthesis
going backward

Reconfiguration
going sideways
Snowcap: Synthesizing Network-Wide Configuration Updates

Tibor Schneider    Rüdiger Birkner    Laurent Vanbever

SIGCOMM’21, August 24, 2021
Snowcap performs network reconfigurations automatically and safely.

Input

- $C_i$: initial configurations
- $C_f$: final configurations
- $\phi$: hard spec
- $f$: soft spec

Output

- Snowcap
- Live Network

Compute the difference set of commands. Describe how network properties change.
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.

<table>
<thead>
<tr>
<th></th>
<th>≥ 50% chance to violate reachability</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random order</td>
<td>70%</td>
<td></td>
</tr>
<tr>
<td>Best practice order</td>
<td>25%</td>
<td></td>
</tr>
<tr>
<td>Snowcap</td>
<td>0%</td>
<td>at most 12s*</td>
</tr>
</tbody>
</table>

*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**

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

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

**Simplicity**

**Learnability**
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.

**Learnability**
Can we *learn* how to invert network computations instead of writing inverse models by hands?
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