Self-Driving Networks

Breaking new ground in network automation

Laurent Vanbever

nsg.ee.ethz.ch

ETH Zurich

May 29 2019

1962

Paul Baran (1926–2011)

American electrical engineer

created the notion of a distributed network

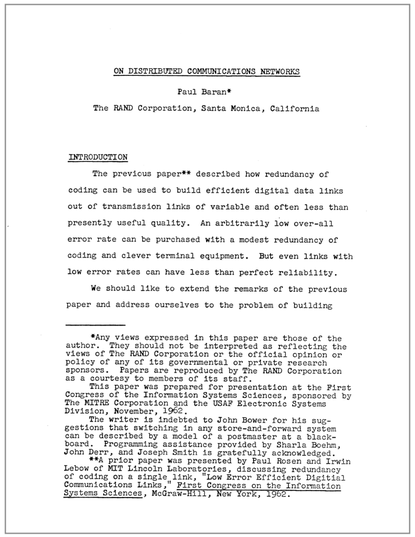
which could maintain communications

in the face of a thermonuclear attack

inventor of packet switching
(with Donald Davies)

On distributed communication networks

Paul Baran—Sept. 1962



ON DISTRIBUTED COMMUNICATIONS NETWORKS

Paul Baran*

The RAND Corporation, Santa Monica, California

INTRODUCTION

The previous paper** described how redundancy of coding can be used to build efficient digital data links out of transmission links of variable and often less than presently useful quality. An arbitrarily low over-all error rate can be purchased with a modest redundancy of coding and clever terminal equipment. But even links with low error rates can have less than perfect reliability.

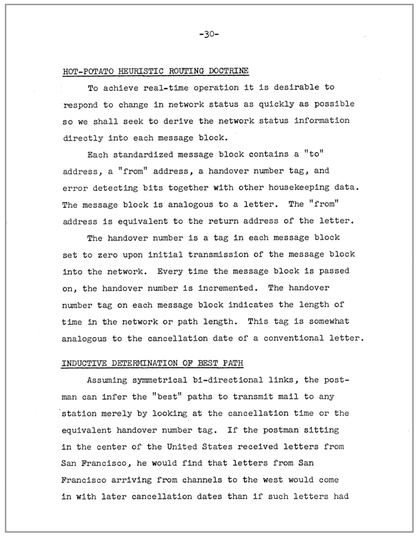
We should like to extend the remarks of the previous paper and address ourselves to the problem of building

*Any views expressed in this paper are those of the author. They should not be interpreted as reflecting the views of The RAND Corporation or the official opinion or policy of any of its governmental or private research sponsors. Papers are reproduced by The RAND Corporation as a courtesy to members of its staff.

This paper was prepared for presentation at the First Congress of the Information Systems Sciences, sponsored by The MITRE Corporation and the USAF Electronic Systems Division, November, 1962.

The writer is indebted to John Bower for his suggestions that switching in any store-and-forward system can be described by a model of a postmaster at a blackboard. Programming assistance provided by Sharla Boehm, John Derr, and Joseph Smith is gratefully acknowledged.

**A prior paper was presented by Paul Rosen and Irwin Lebow of MIT Lincoln Laboratories, discussing redundancy of coding on a single link, "Low Error Efficient Digitial Communications Links," First Congress on the Information Systems Sciences, McGraw-Hill, New York, 1962.



-30-

HOT-POTATO HEURISTIC ROUTING DOCTRINE

To achieve real-time operation it is desirable to respond to change in network status as quickly as possible so we shall seek to derive the network status information directly into each message block.

Each standardized message block contains a "to" address, a "from" address, a handover number tag, and error detecting bits together with other housekeeping data. The message block is analogous to a letter. The "from" address is equivalent to the return address of the letter.

The handover number is a tag in each message block set to zero upon initial transmission of the message block into the network. Every time the message block is passed on, the handover number is incremented. The handover number tag on each message block indicates the length of time in the network or path length. This tag is somewhat analogous to the cancellation date of a conventional letter.

INDUCTIVE DETERMINATION OF BEST PATH

Assuming symmetrical bi-directional links, the postman can infer the "best" paths to transmit mail to any station merely by looking at the cancellation time or the equivalent handover number tag. If the postman sitting in the center of the United States received letters from San Francisco, he would find that letters from San Francisco arriving from channels to the west would come in with later cancellation dates than if such letters had

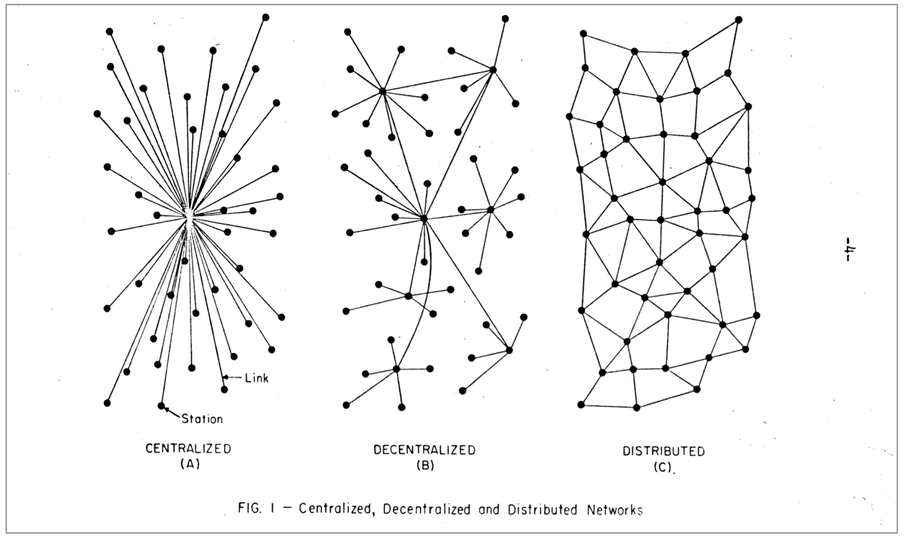


FIG. 1 — Centralized, Decentralized and Distributed Networks

Introduces the concept of

- switching
- distributed routing
- reliable transport

among many others…

Thanks to Paul Barlan et al.

the Internet infrastructure is extremely resilient

ability to route packets around failures

(not necessarily fast though)

Thanks to Paul Barlan et al.

the Internet infrastructure is extremely resilient

… at least when it comes to thermonuclear wars…

Thanks to Paul Barlan et al.

the Internet infrastructure is extremely resilient

… at least when it comes to thermonuclear wars…

but how does it fare against, say…

Thanks to Paul Barlan et al.

the Internet infrastructure is extremely resilient

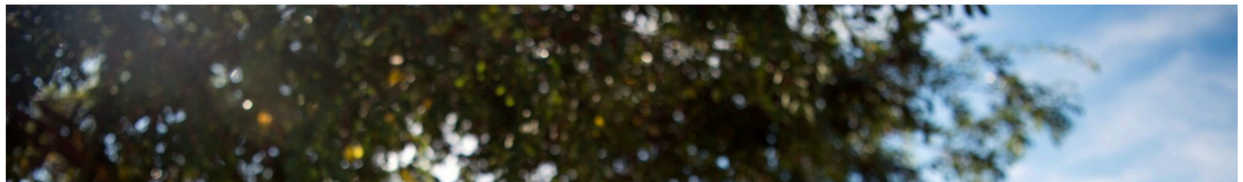
… at least when it comes to thermonuclear wars…

but how does it fare against, say… us humans?

Google accidentally broke the internet throughout Japan

A mistake led to internet outages for about half of the country.

Mallory Locklear, @mallorylocklear
08.28.17 in Internet



27 August 2017

Someone in Google made a configuration mistake which caused Google Chicago to wrongly advertise 160k IP prefixes to its neighbors.

Someone in Google made a configuration mistake which caused Google Chicago to wrongly advertise 160k IP prefixes to its neighbors.

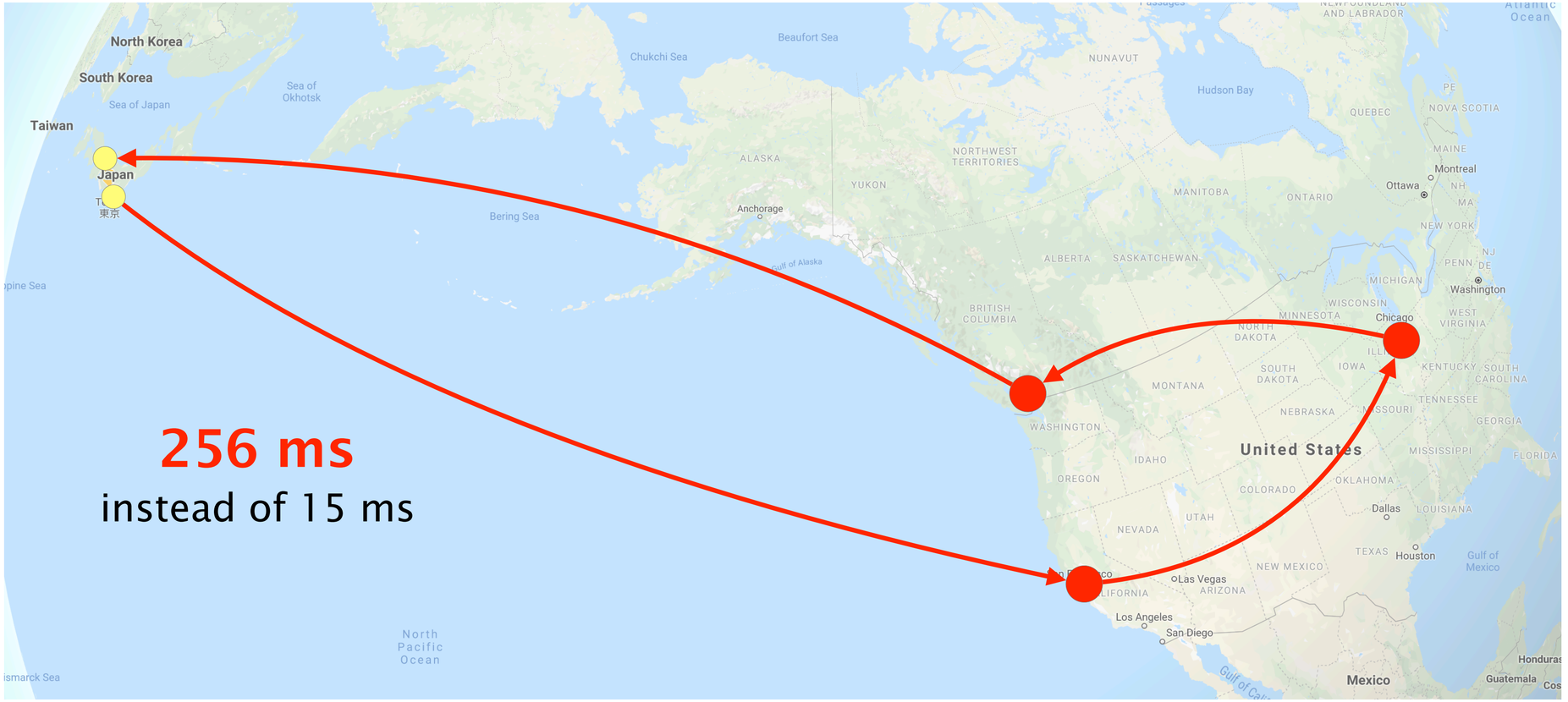
These advertisements propagated in the Internet and got picked up by Japanese giants (IIJ and KDDI) which started to direct local Japanese traffic to Google.

Tokyo to Nagoya

(normally)

Tokyo to Nagoya via… Chicago?!

A **17x** increased latency!



Someone in Google made a configuration mistake
which caused Google Chicago to wrongly advertise
160k IP prefixes to its neighbors.

These advertisements propagated in the Internet
and got picked up by Japanese giants (NTT and KDDI)
which started to direct local, Japanese traffic to Google.

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

This is far from being an isolated event...

Forbes / Personal Finance

The Little Black Book of Billionaire Secrets

JUL 8, 2015 @ 03:36 PM 11,261 VIEWS

United Airlines Blames Router for Grounded Flights

'Configuration Error' Blamed for AWS Outage

By David Ramel ■ 08/12/2015

The summer of network misconfigurations

Amazon's massive AWS outage was caused by human error

One incorrect command and the whole internet suffers.

By Jason Del Rey | @DelRey | Mar 2, 2017, 2:20pm EST

Data Centre ▸ Networks

Level3 switch config blunder for US-wide VoIP blackout

Widespread internet outages affected Comcast, Spectrum, Verizon and AT&T customers

BY: CNN

Data Centre ▸ Networks

CloudFlare apologizes for Telia screwing you over

Unhappy about

By Kieren McCarthy in Sa

Facebook struggles to deal with epic outage


By Donie O'Sullivan and Heather Kelly, CNN Business

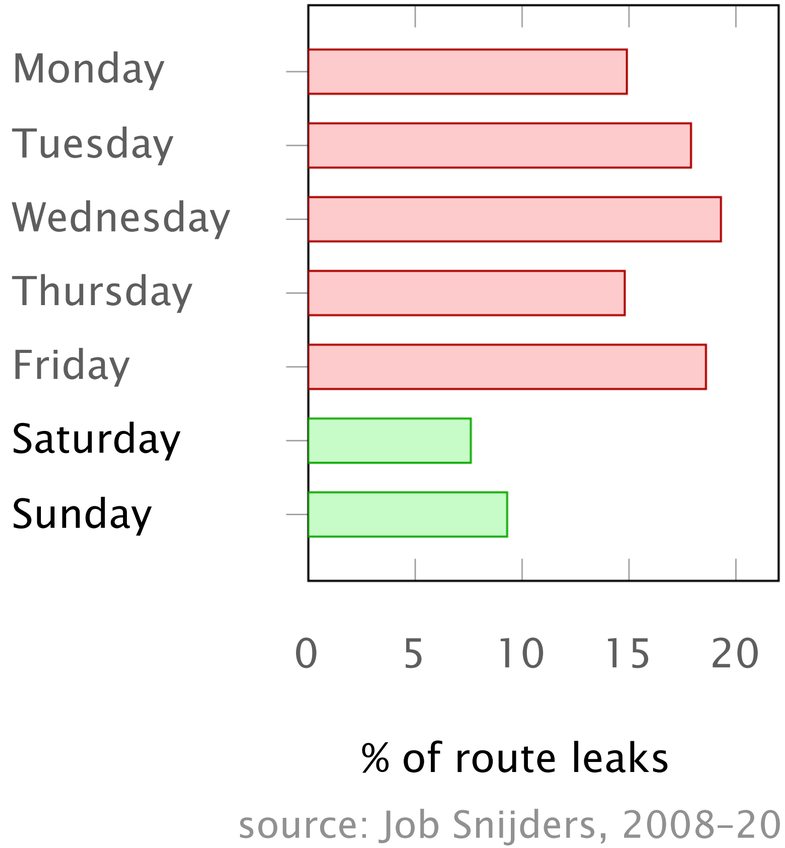
Updated 0654 GMT (1454 HKT) March 14, 2019

"Human factors are responsible

for 50% to 80% of network outages"

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

A perhaps ironic consequence is that the Internet works better during the week-ends



source: Job Snijders, 2008–2016

Need more proof?

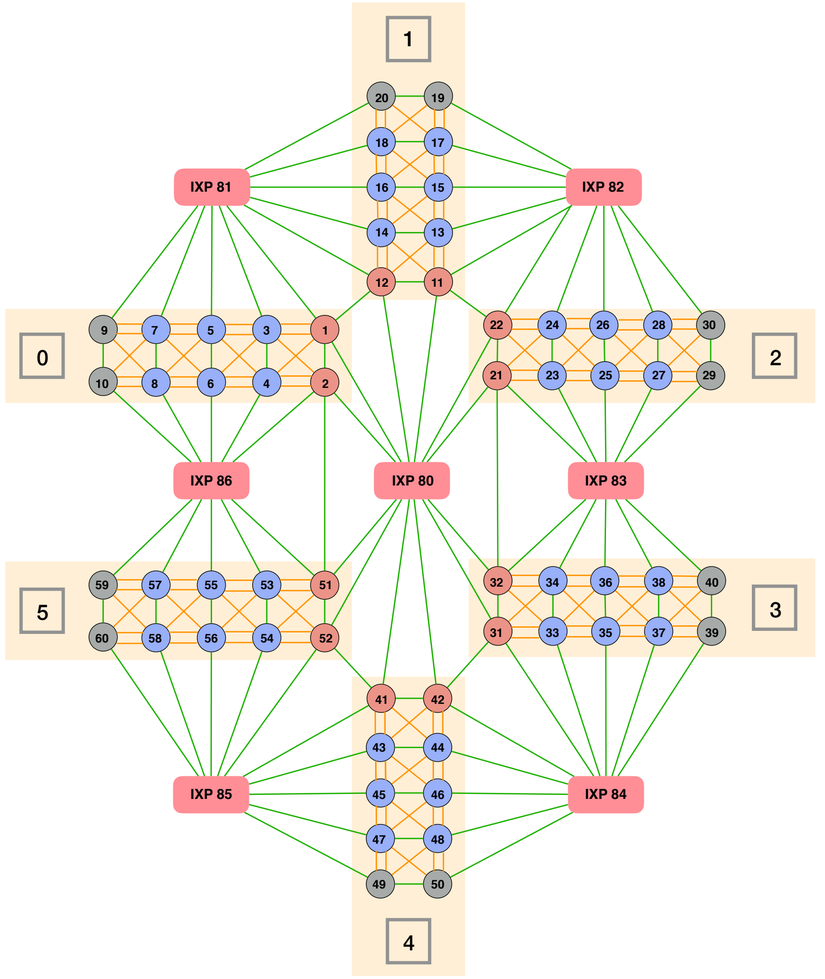
Ask our students!

Yearly Communication Networks "Hackathon"

Each year, D-ITET students build and operate their very own Internet infrastructure

Internet topology 2019

~30 groups, 100 students

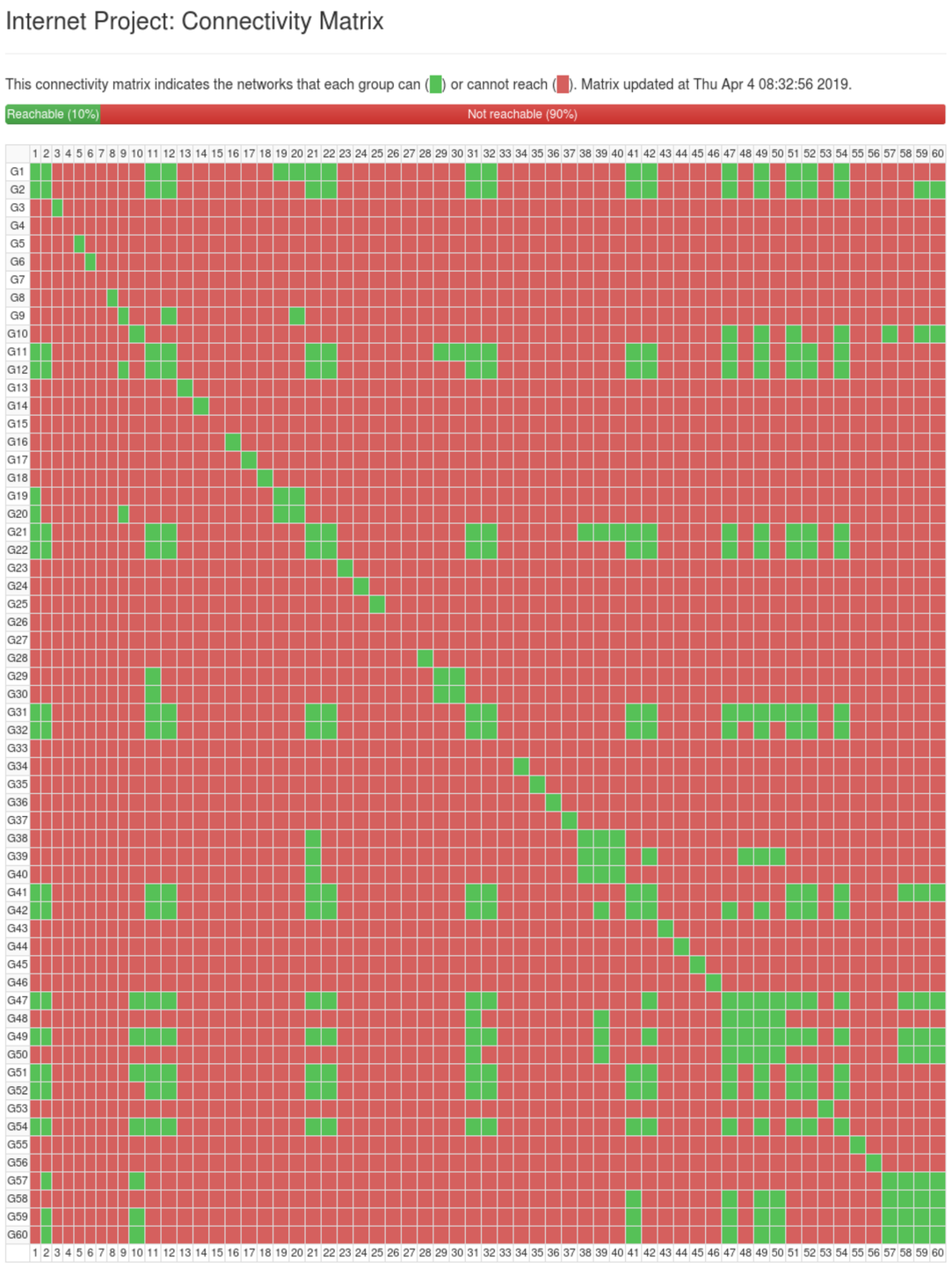


ITET's Internet connectivity

initial ~10%

final ~97%

highest since 2016! ☺

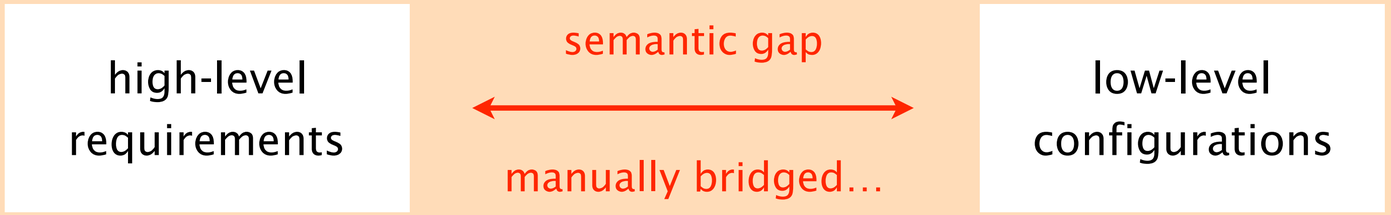


Why ?!?

Configuring networks today is hard

because of a fundamental semantic gap

Configuring networks today is hard because of a fundamental semantic gap



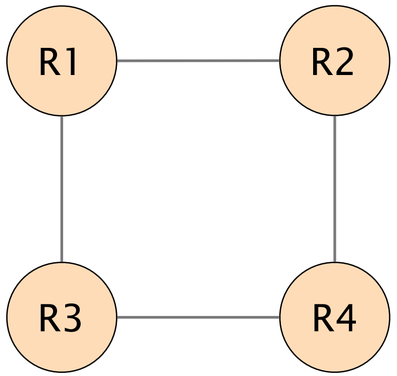
induce
network-wide behavior

High-level requirements encode operational objectives

connectivity	guarantee	reachability
		traffic crosses a firewall
	prevent	X from reaching Y
business/ performance	minimize	transit costs
		congestion
		convergence time
	maximize	load-balancing

Operators implement these requirements by manually configuring their network devices

setting parameter values



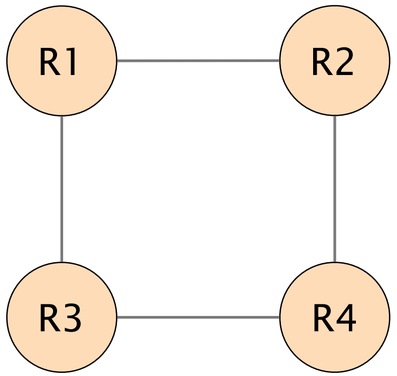
Operators implement these requirements by manually configuring their network devices

R1's configuration

```
id          BRUX
lo0         10.0.0.1
...
paramX      OSPF
 weight (1,2)  10
 weight (1,3)  20
```

R2's configuration

```
id          PARIS
lo0         10.0.0.2
...
paramX      OSPF
 weight (2,1)  10
 weight (2,4)  20
```



R3's configuration

```
id          LON
lo0         10.0.0.3
...
paramX      OSPF
 weight (3,1)  10
 weight (3,4)  20
```

R4's configuration

```
id          ZRH
lo0         10.0.0.4
...
paramX      OSPF
 weight (4,1)  10
 weight (4,2)  20
```

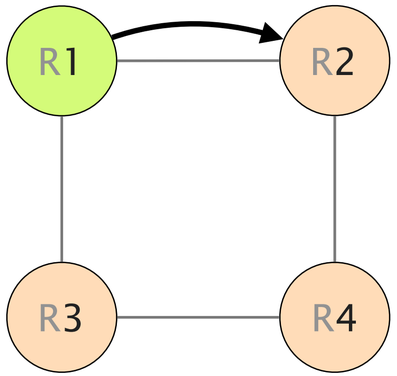
These parameters constraint the behavior of the distributed routing protocols each node runs

R1's configuration

id	BRUX
lo0	10.0.0.1
...	
paramX	OSPF
weight (1,2)	10
weight (1,3)	20

Eurovision

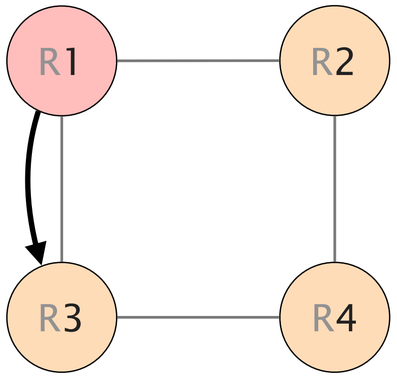
video traffic



R1's configuration

id	BRUX
lo0	10.0.0.1
...	
paramX	OSPF
weight (1,2)	100
weight (1,3)	20

Eurovision
video traffic



Defining the routing behavior of a large network requires writing millions of lines of configuration

Configuration length

avg.	14k	LoC per device
max.	97k	

200+ devices, large European network

"Human factors are responsible

for 50% to 80% of network outages"

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

“Human factors are responsible

for [redacted] of [redacted]”

“Human factors are responsible

for >90% of car accidents”

NHTSA, National Motor Vehicle Crash Causation Survey, February 2015

Enters…

Self-Driving Car

Enters…

Self-Driving Car

Wikipedia

A vehicle that is capable of
sensing its environment and
moving with little or no human input

Enters…

Self-Driving Car

Wikipedia — A vehicle that is capable of sensing its environment and moving with little or no human input

Promise — A drastic reduction in the number of (fatal) accidents

Enters…

Self-Driving Car

Enters…

Self-Driving

Enters…

Self-Driving Network

Enters…

Self-Driving Network

Definition

A network that is capable of sensing its environment and adapting its behavior accordingly with little or no human input

Enters…

Self-Driving Network

Definition

A network that is capable of
sensing its environment and
adapting its behavior accordingly
with little or no human input

Questions

How do we **build** and deploy such networks?

Until recently, innovating in networks was hard because devices were completely locked down

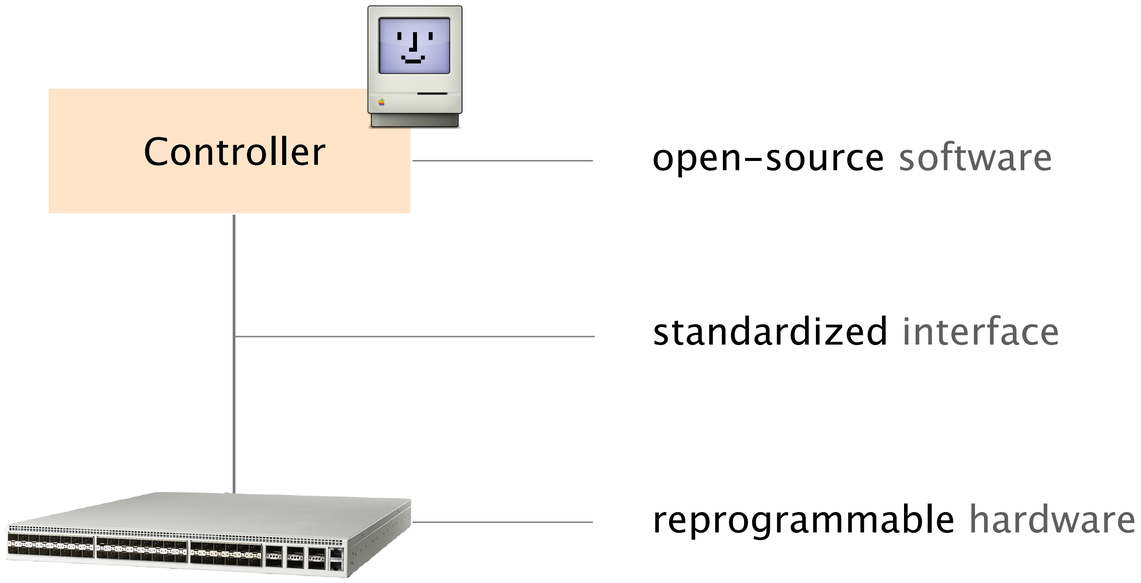
closed software

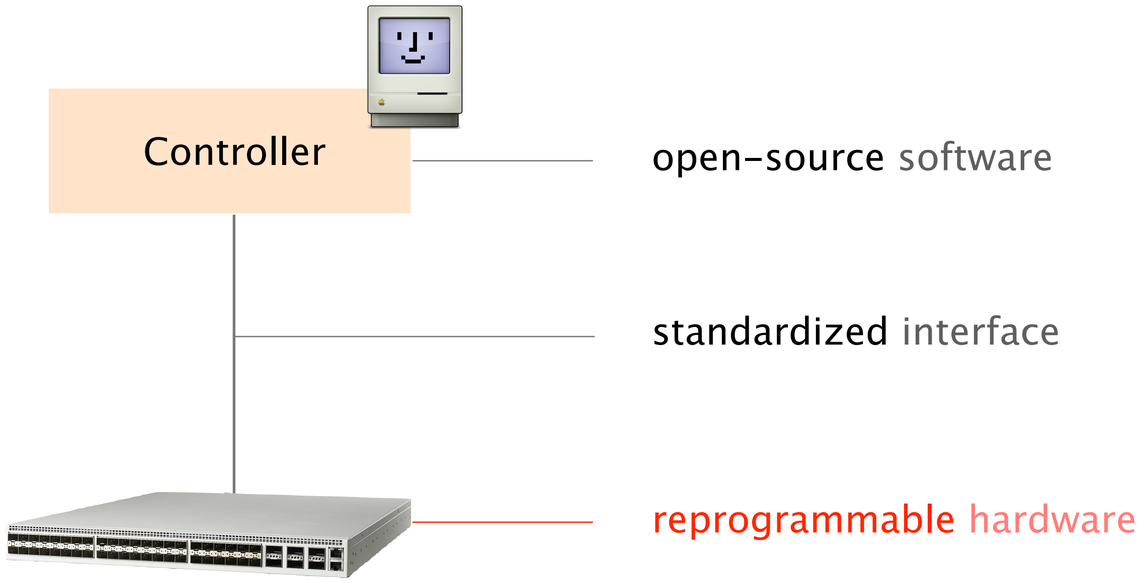
closed hardware

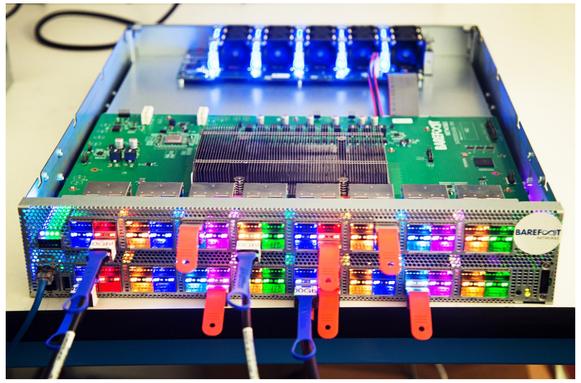
Cisco™ device

Things are changing though!

Networks are on the verge of a paradigm shift towards deep programmability

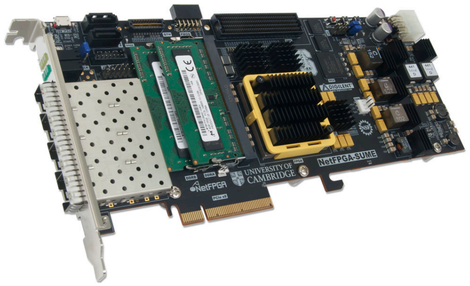



Controller
open-source software
standardized interface
reprogrammable hardware

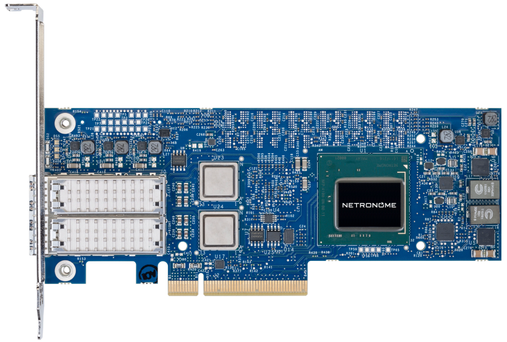


Barefoot Tofino 12.8 Tbps

programmable network switch



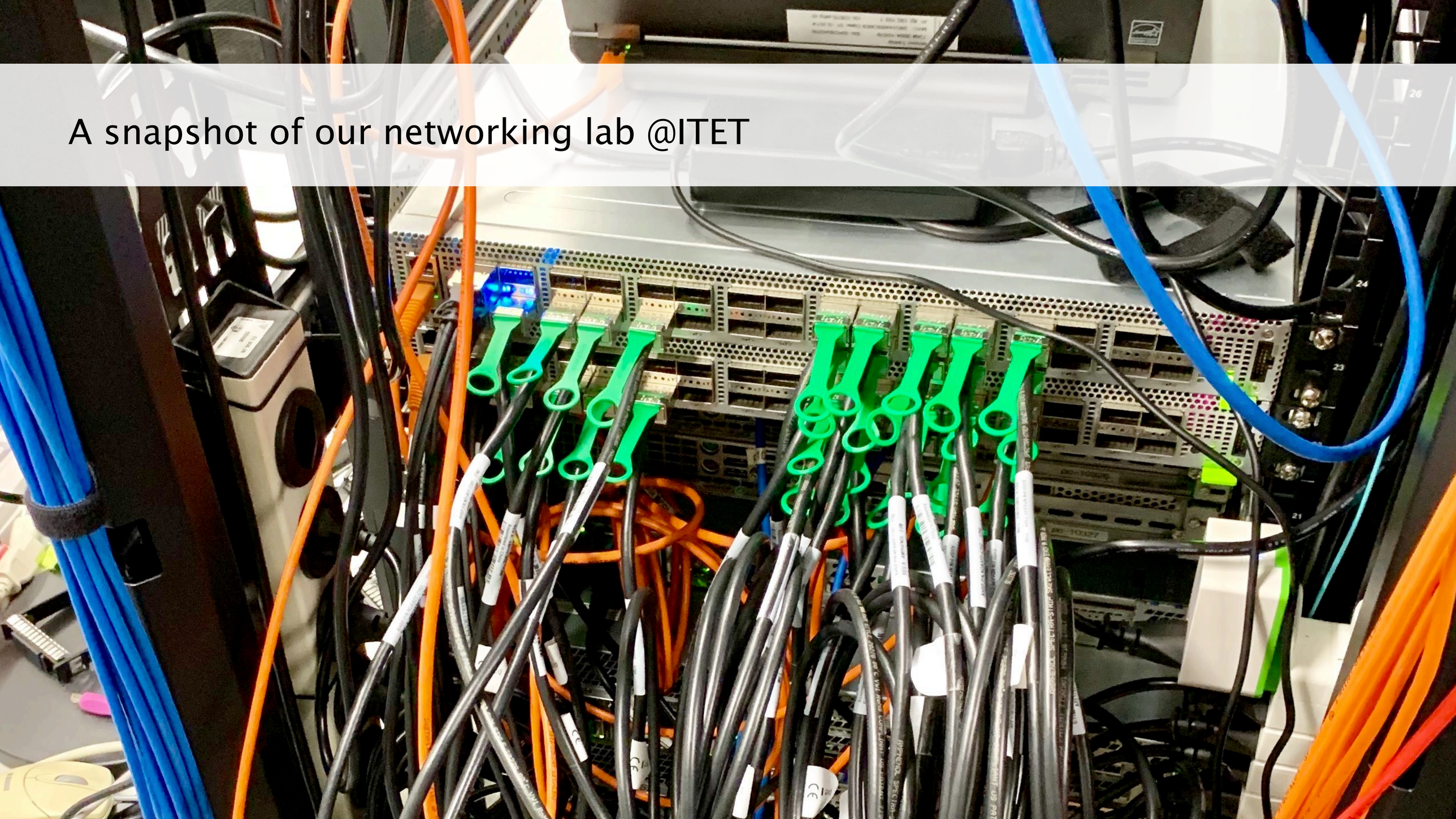
NetFPGA SUME



Netronome Agilio CX

100 Gbps programmable network card

A snapshot of our networking lab @ITET



Programmable networks can be made self-driving

Programmable devices can

- measure
- perform statistical inference
- adapt their forwarding decisions

at line rate, on a per-packet basis

Enters…

Self-Driving Network

Definition

A network that is capable of

sensing its environment and

adapting its behavior accordingly

with little or no human input

Questions

How do we build and **deploy** such networks?

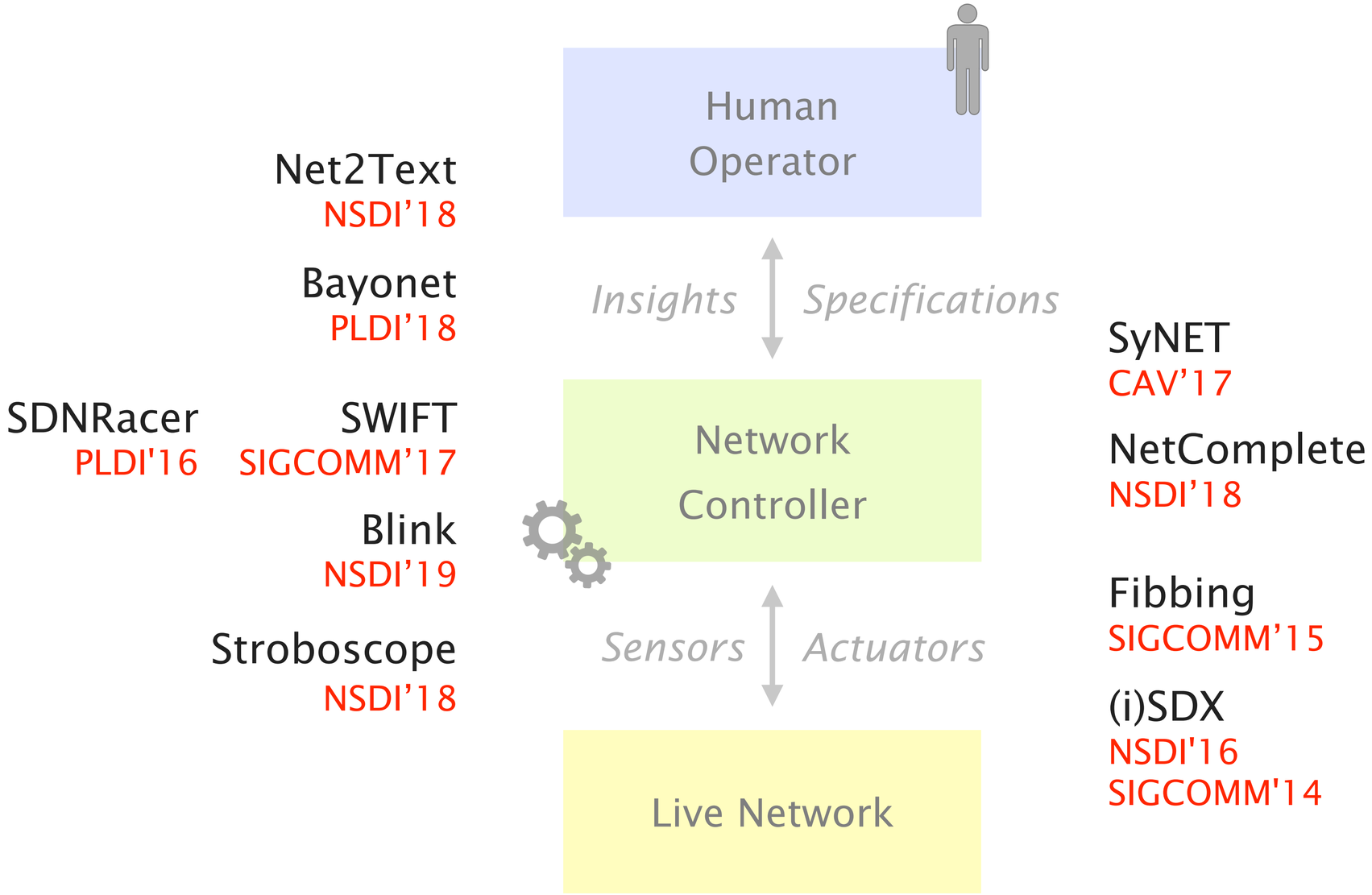
Levels of autonomy in self-driving cars

level 0	no automation	
1	driver assistance	human monitors the environment
2	partial automation	
3	conditional automation	system monitors human as fallback
4	high automation	
5	full automation	no more human

Self-Driving Networks

Breaking new ground in network automation

1	operator assistance	Part I assisting operators
2	partial automation	
3	conditional automation	Part II taking control
4	high automation	
5	full automation	too futuristic? ☺

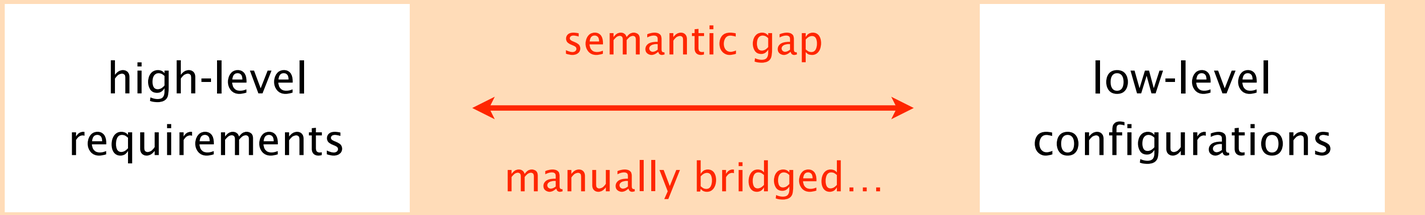

Human Operator
Insights
Specifications
Network Controller
Sensors
Actuators
Live Network
Net2Text
NSDI'18
Bayonet
PLDI'18
SDNRacer
PLDI'16
SWIFT
SIGCOMM'17
Blink
NSDI'19
Stroboscope
NSDI'18
SyNET
CAV'17
NetComplete
NSDI'18
Fibbing
SIGCOMM'15
(i)SDX
NSDI'16
SIGCOMM'14

Self-Driving Networks

Breaking new ground in network automation

1	operator assistance	Part I assisting operators
2	partial automation	
3	conditional automation	Part II taking control
4	high automation	
5	full automation	too futuristic? ☺

Part I: Assisting operators in configuring their networks



induce
network-wide behavior

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 assists network operators in bridging the semantic gap between requirements and low-level configurations

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 as a logical formula (in SMT)
- partial configurations

First Encode the
- protocol semantics
- 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

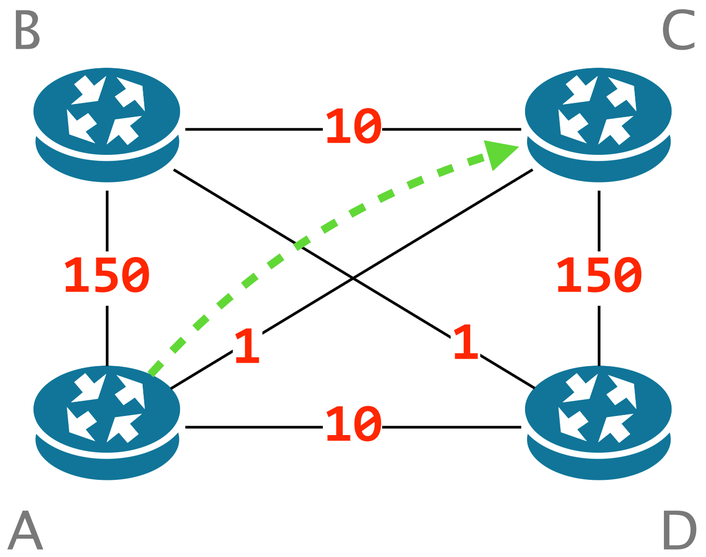
search space navigation

Insight #2

partial evaluation

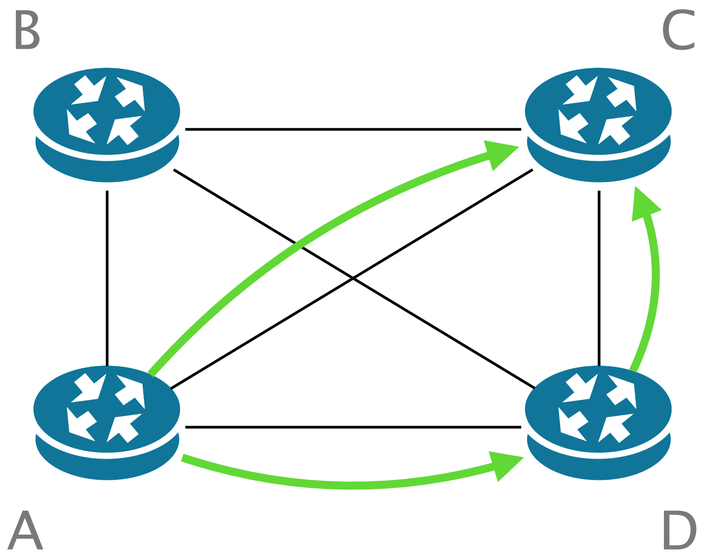
search space reduction

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

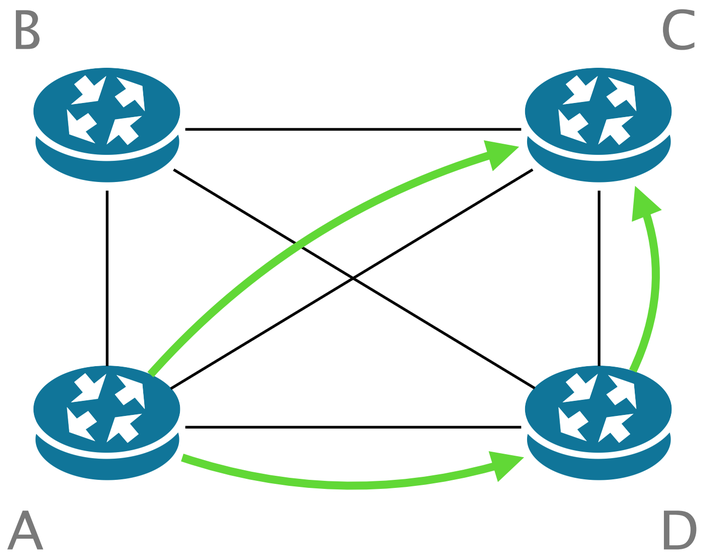


For performance reasons,

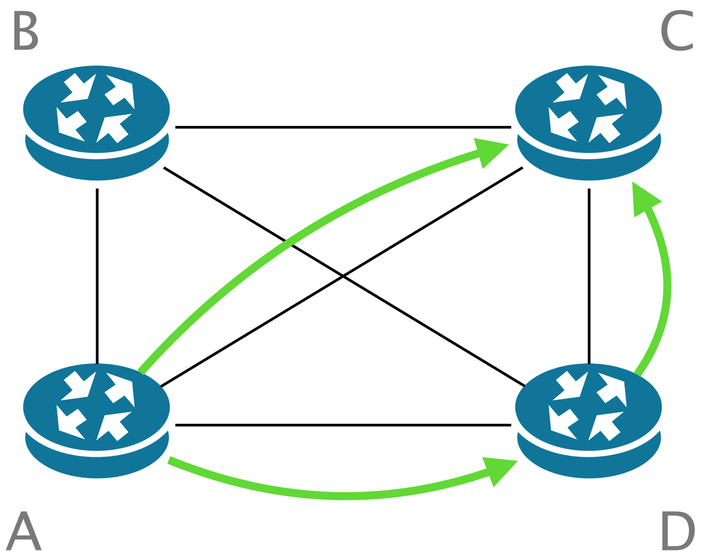
the operators want to enable load-balancing



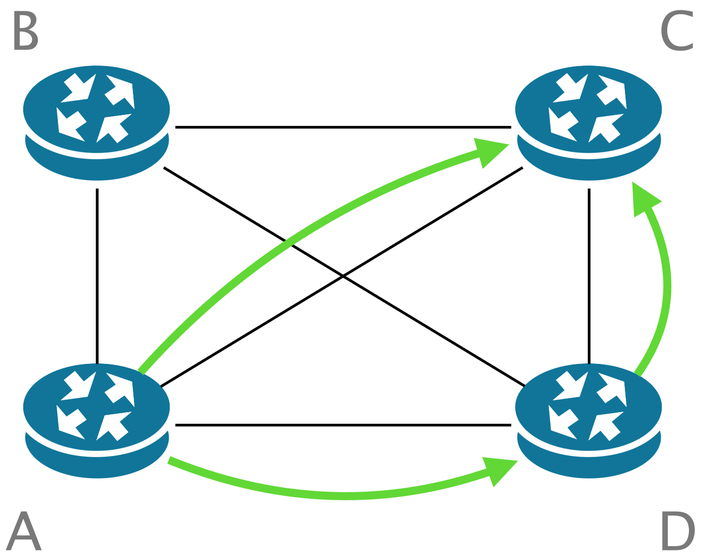
What should be the weights for this to happen?



input requirements

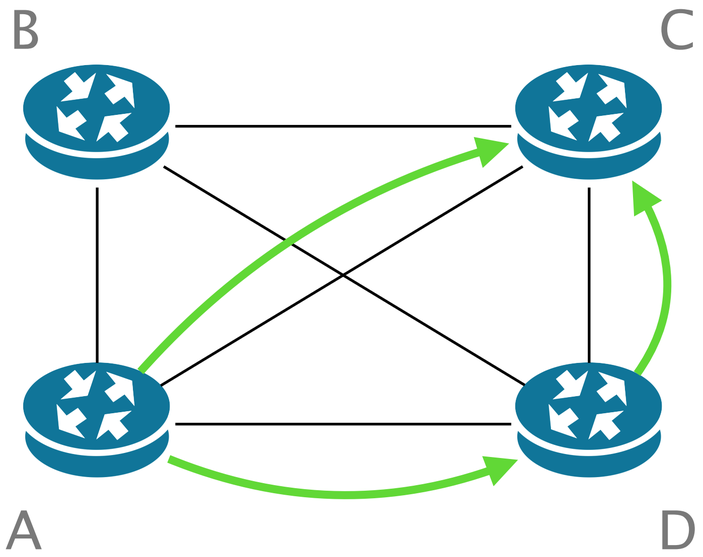


input requirements



synthesis procedure

input requirements



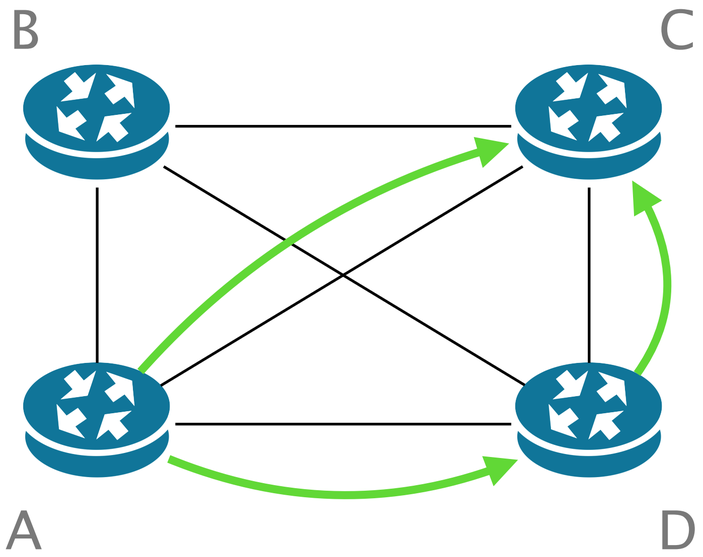
synthesis procedure

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

$$\downarrow$$

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

input requirements



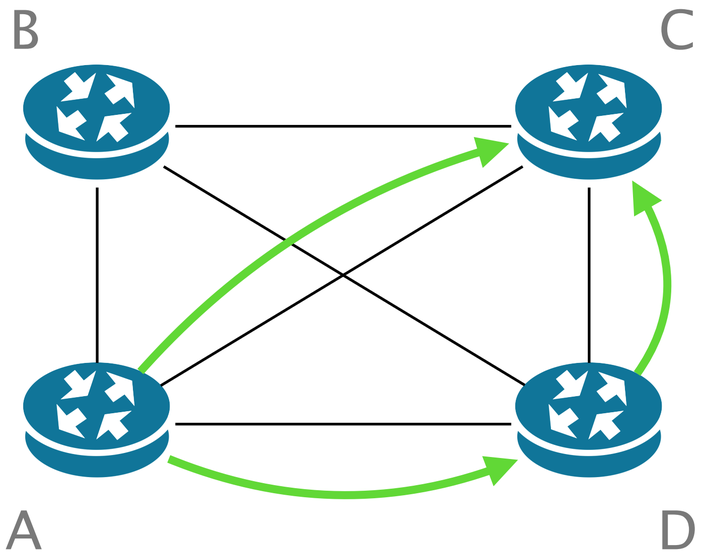
synthesis procedure

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

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

Solve

input requirements



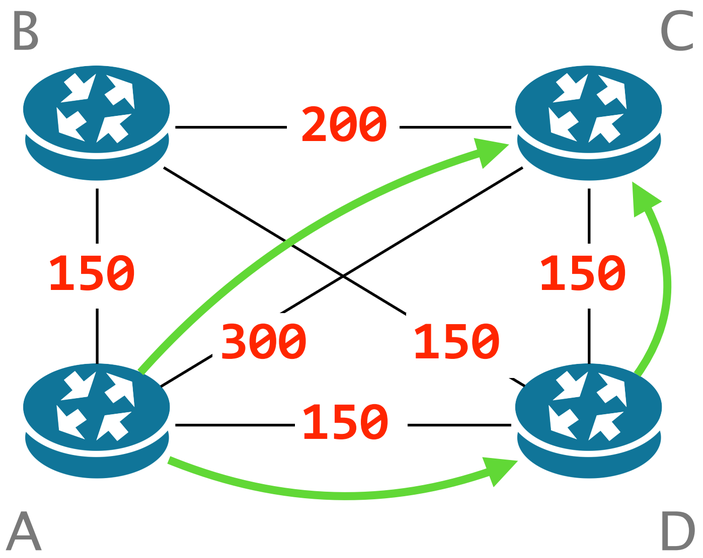
synthesis procedure

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

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

Solve

input requirements



Synthesized weights

synthesis procedure

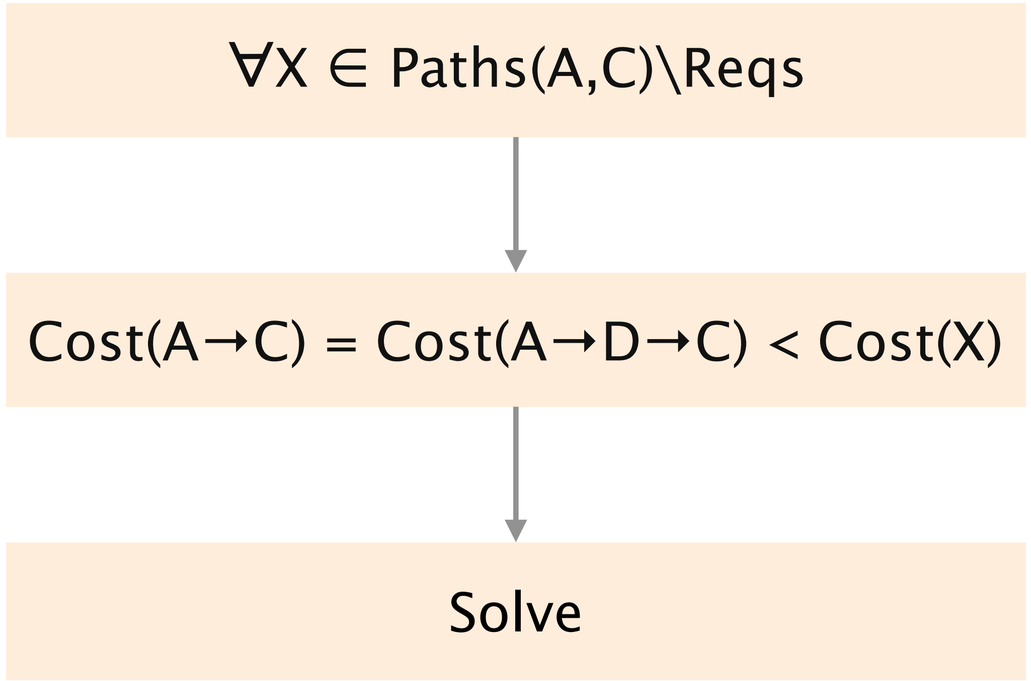
$\forall X \in Paths(A,C) \backslash Reqs$

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

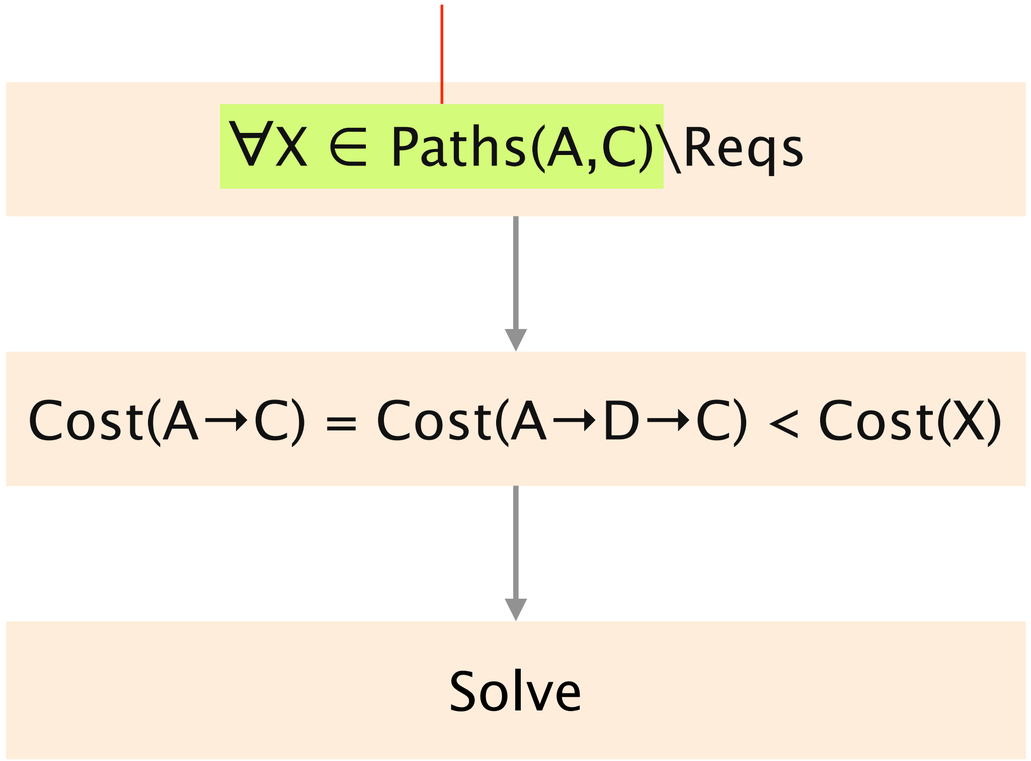
Solve

This was easy, but…

it does **not** scale



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



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 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 Req and add it to S

Repeat.

Instead of considering all paths between X and Y

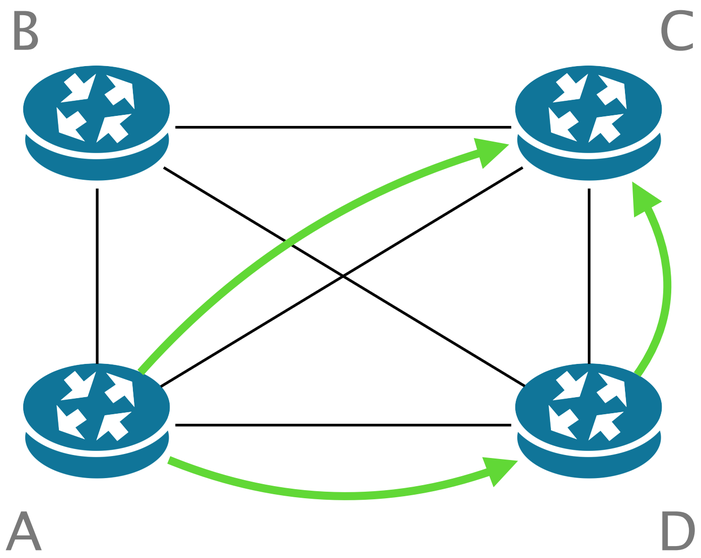
CEGIS Part 1: Consider a random subset S of them and synthesize the weights considering S only

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

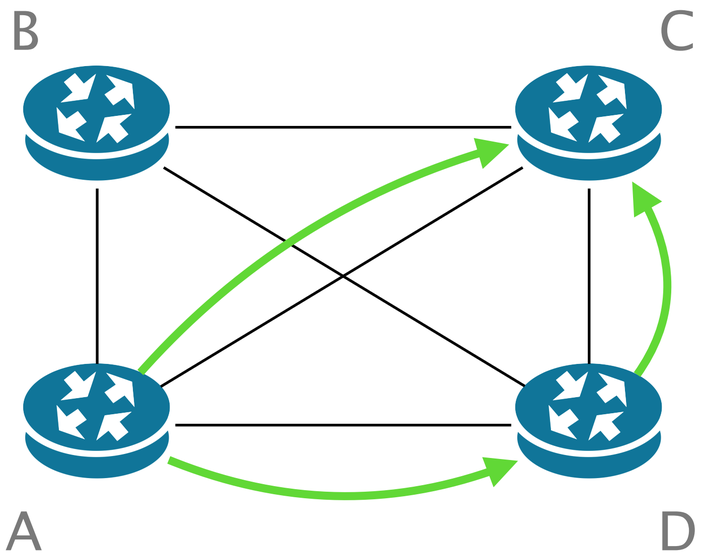
intuition: **Fast too**

simple shortest-path computation

input requirements

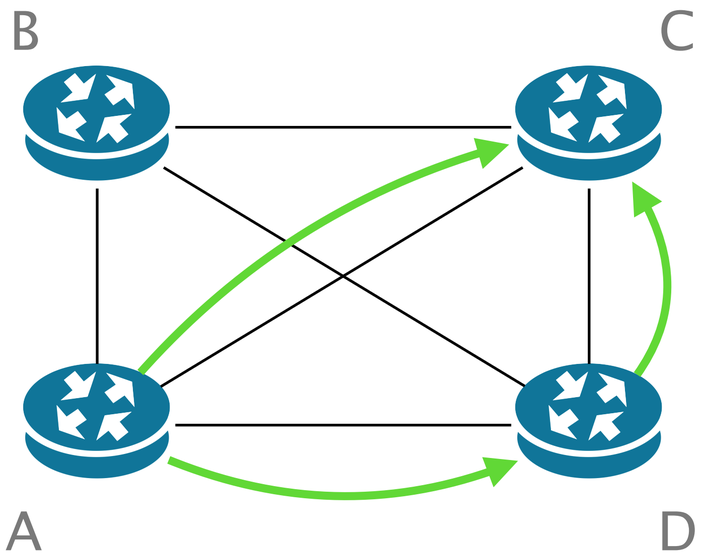


input requirements



synthesis procedure

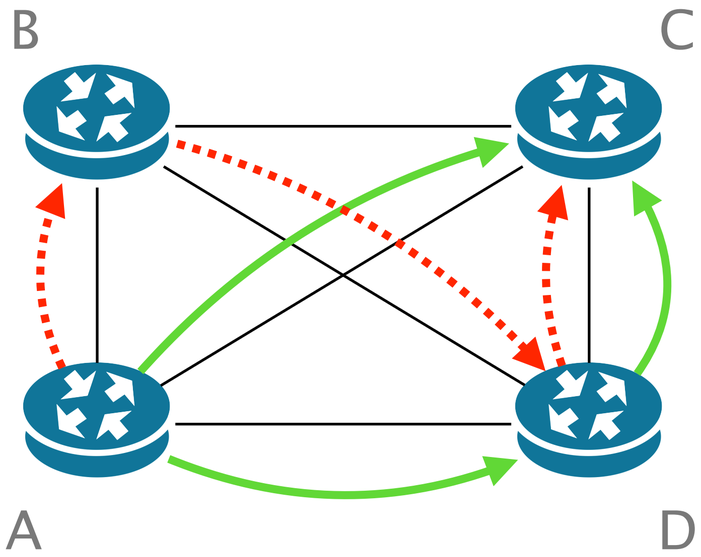
input requirements



synthesis procedure

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

input requirements



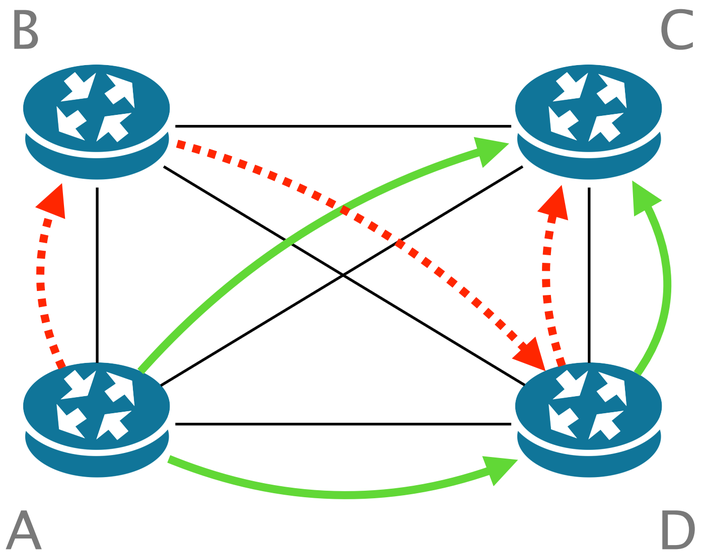
synthesis procedure

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

↓

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

input requirements



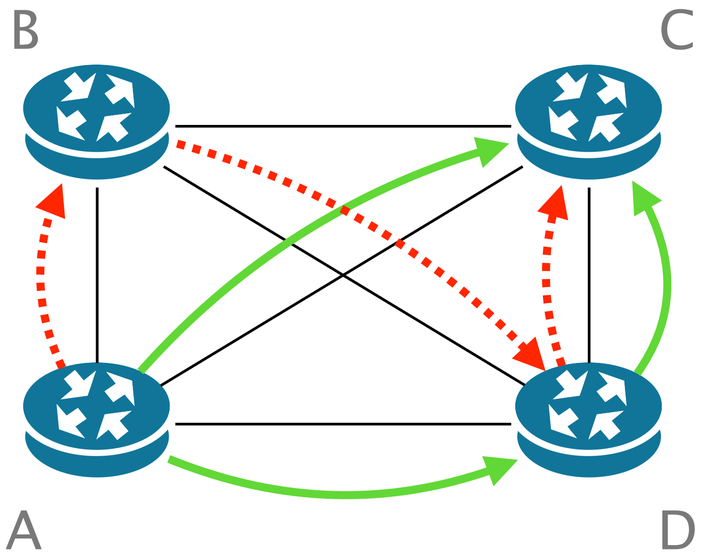
synthesis procedure

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

$$\downarrow$$

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

input requirements



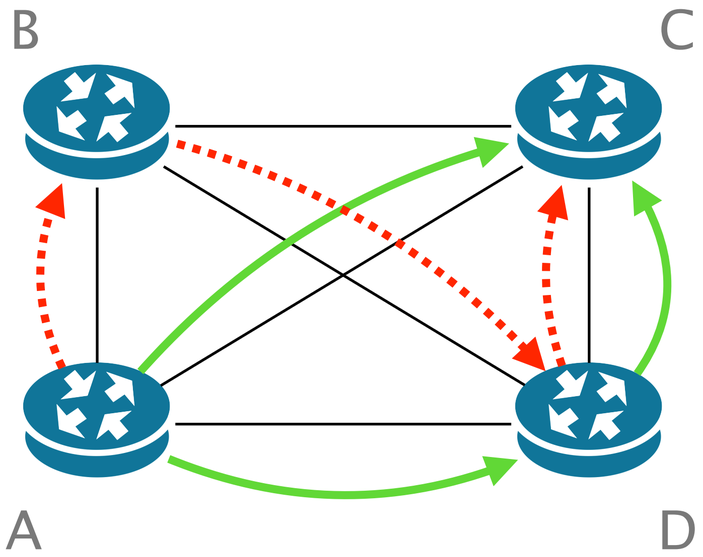
synthesis procedure

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

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

Solve

input requirements



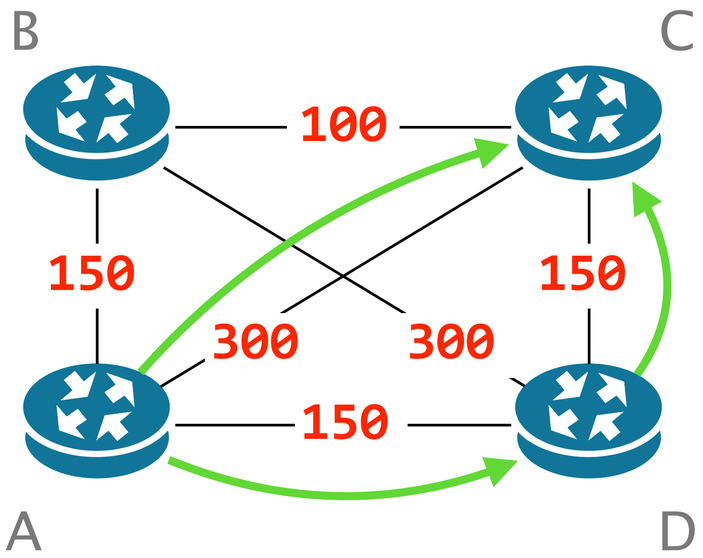
synthesis procedure

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

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

Solve

input requirements



Synthesized weights

synthesis procedure

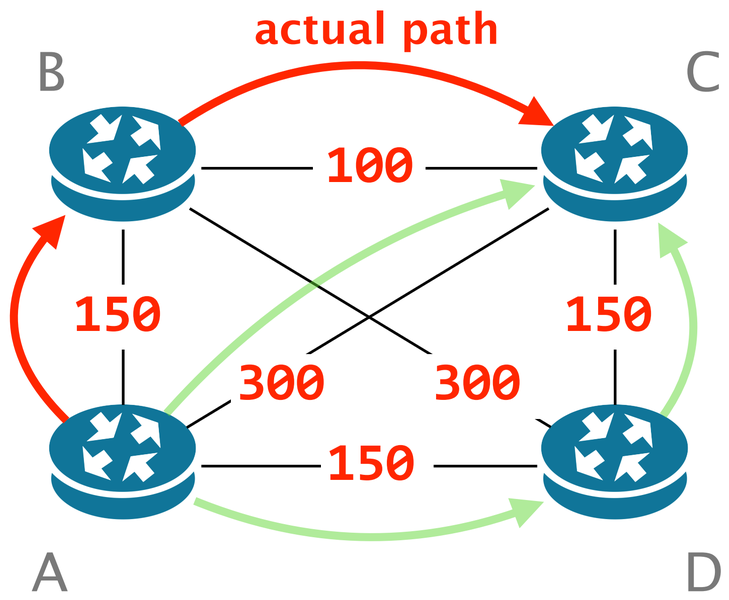
$\forall X \in \text{SamplePaths}(A,C) \setminus \text{Reqs}$

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

Solve

The synthesized weights are incorrect:

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



∀X ∈ SamplePaths(A,C)\Reqs

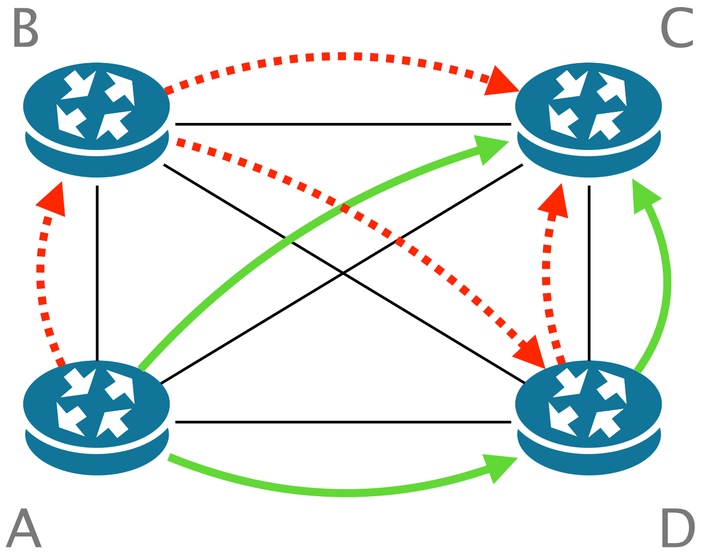
↓

Cost(A→C) = Cost(A→D→C) < Cost(X)

↓

Solve

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



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

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

The entire procedure usually converges in few iterations

making it very fast in practice

Self-Driving Networks

Breaking new ground in network automation

1	operator assistance	Part I assisting operators
2	partial automation	
3	conditional automation	Part II taking control
4	high automation	
5	full automation	too futuristic? ☺

Part II: Taking control

applications

frameworks

methods

data-driven convergence

monitoring/inference

some of our challenges lying ahead

Part II: Taking control

applications

frameworks

methods

data-driven

convergence

Blink

Fast Connectivity Recovery Entirely in the Data Plane


Thomas Holterbach


Edgar Costa Molero


Maria Apostolaki

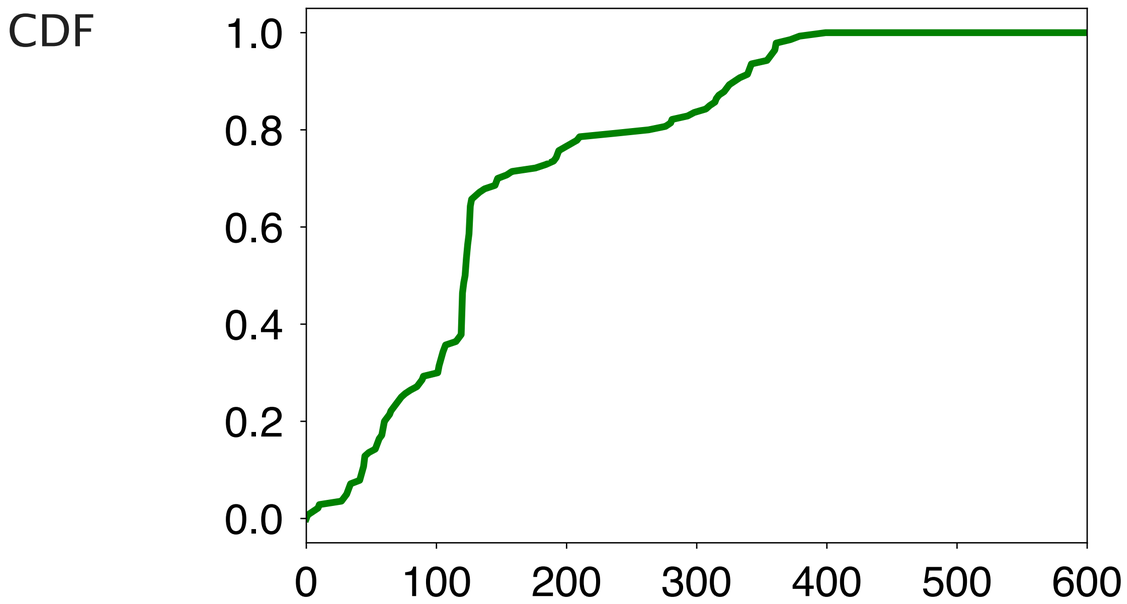

Alberto Dainotti


Stefano Vissicchio


Laurent Vanbever

USENIX Symposium on Networked Systems Design and Implementation. February 2019.

Internet control plane can take minutes to converge upon remote failures



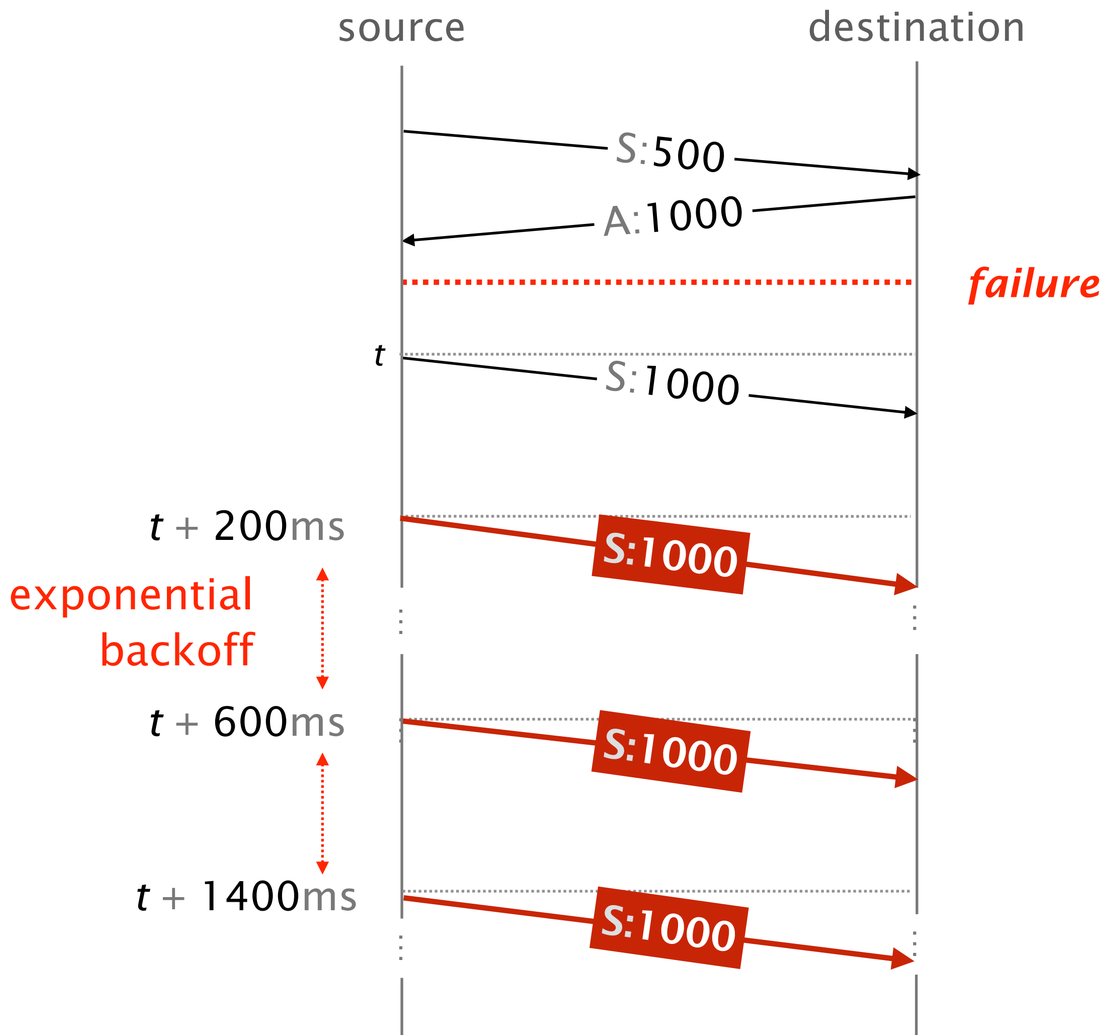
Second elapsed between the data plane failure and the control plane learning about it

While the control plane can take minutes to learn about a failure,

the actual Internet traffic is affected almost instantaneously

How about we track this signal instead?

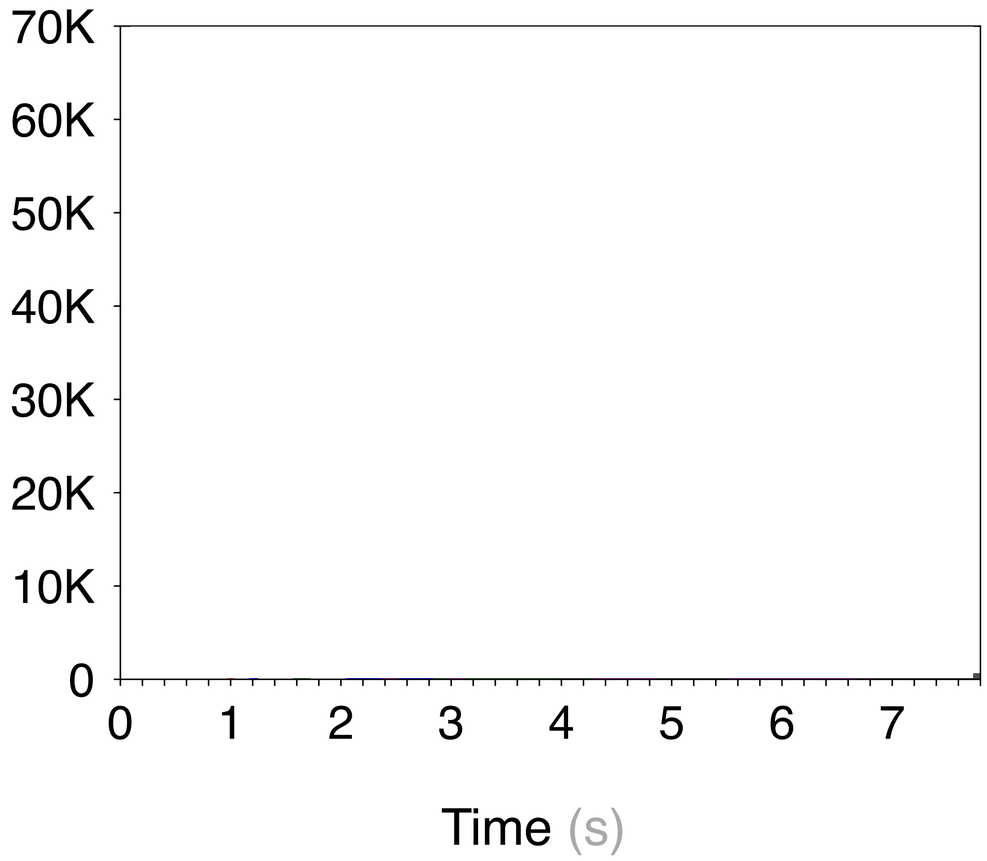
Internet traffic end-points retransmit packets upon experiencing packet drops

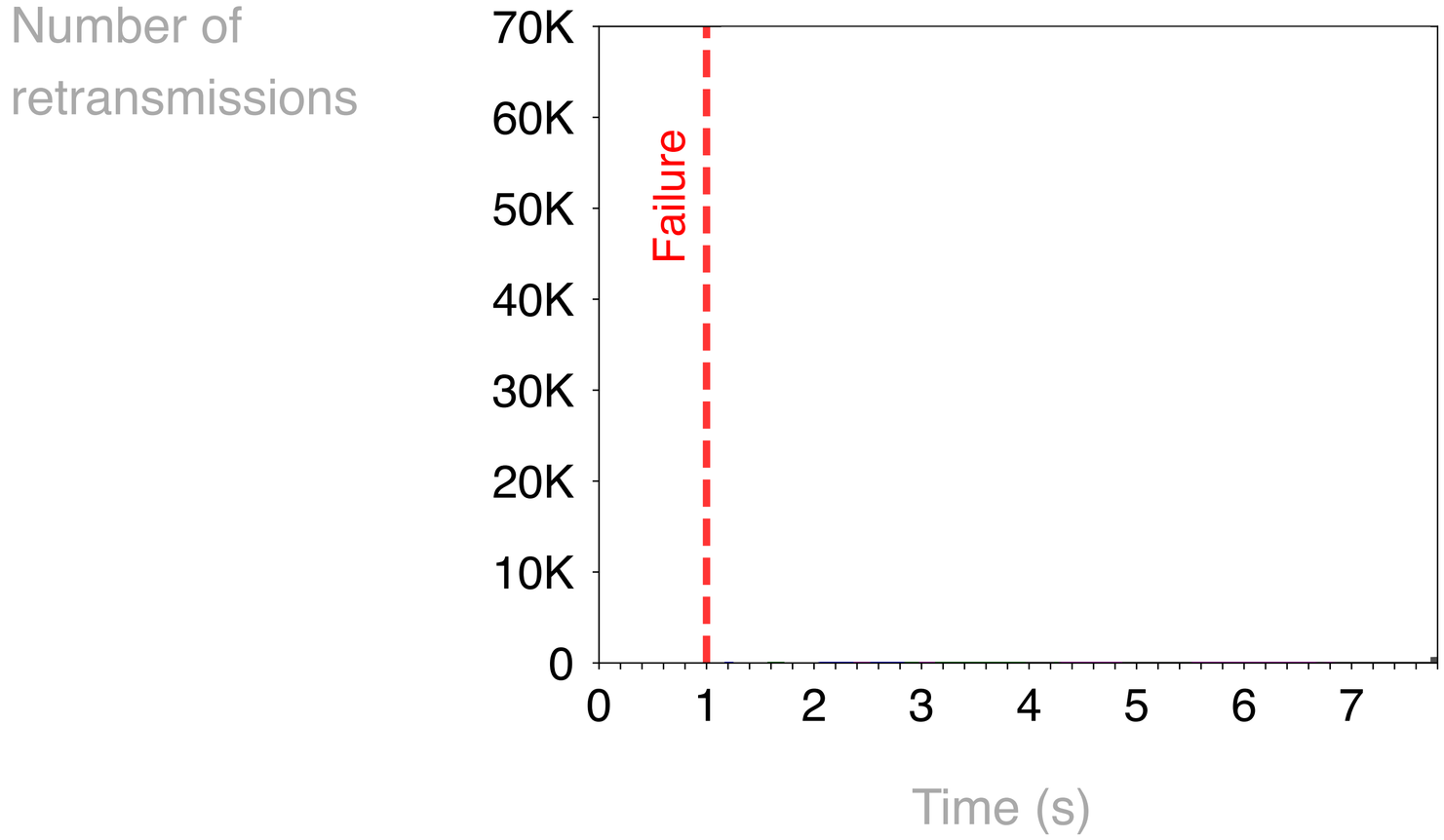


Many end-points retransmitting leads to

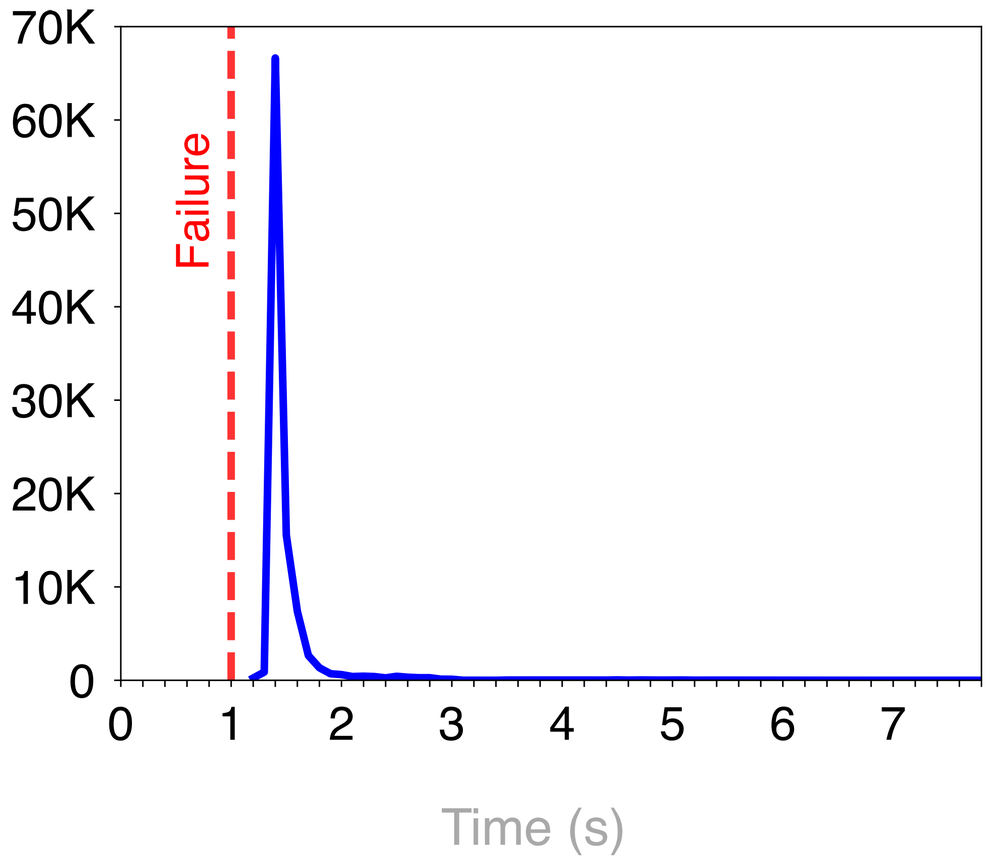
waves of retransmission

Number of retransmissions

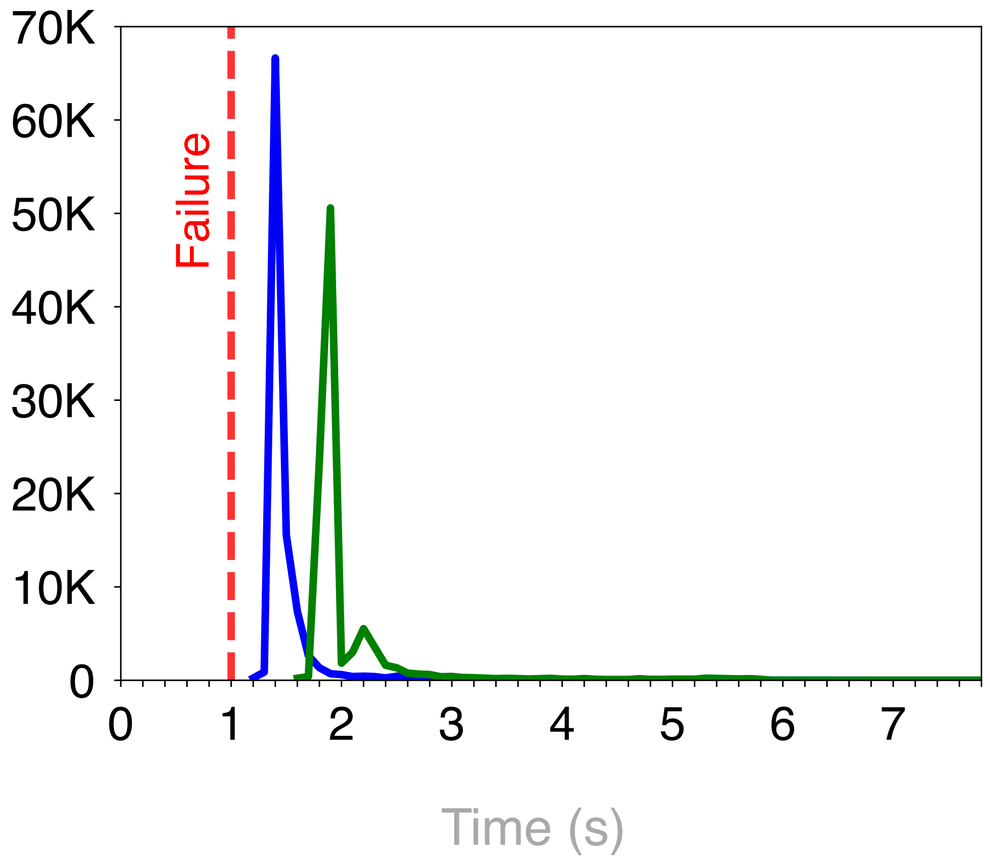



Number of retransmissions
70K
60K
50K
40K
30K
20K
10K
0
Failure
0
1
2
3
4
5
6
7
Time (s)

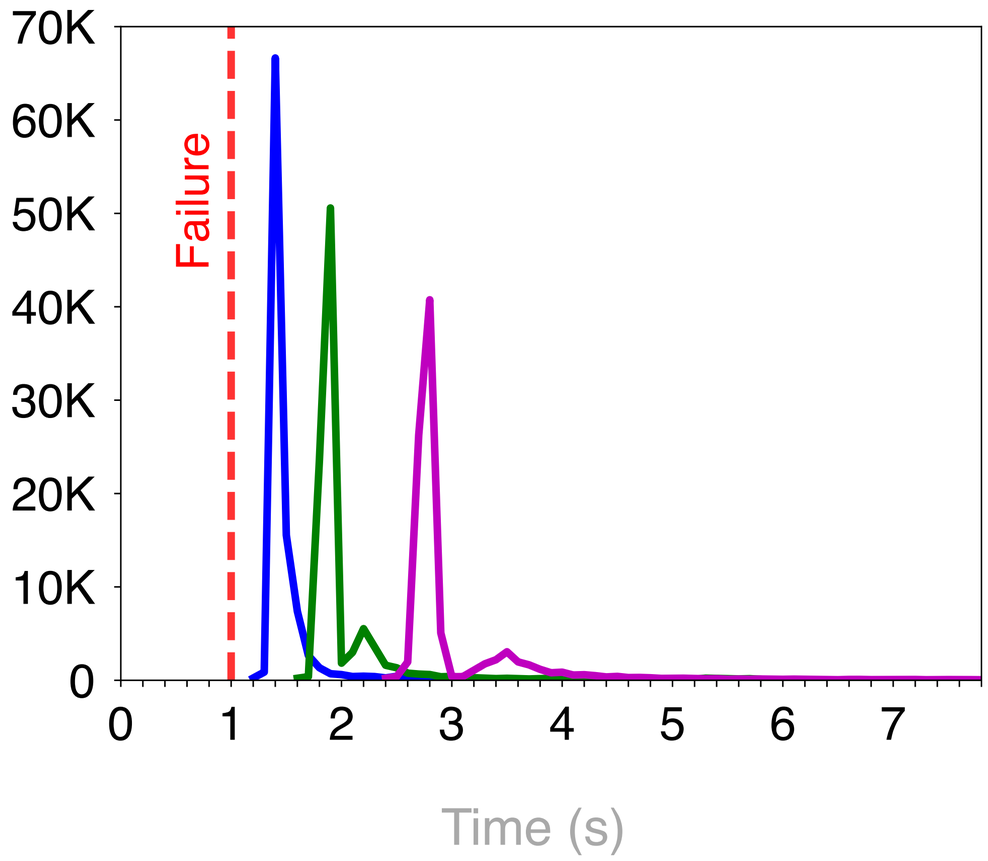
Number of retransmissions



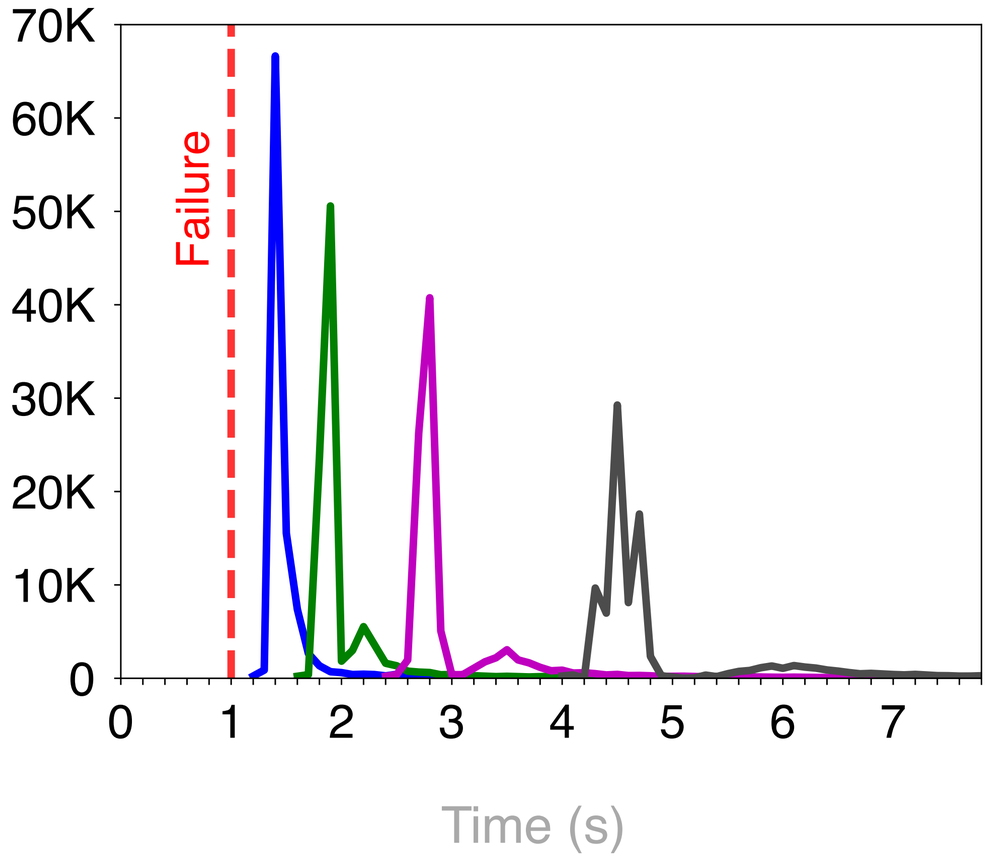
Number of retransmissions



Number of retransmissions



Number of retransmissions



Tracking this signal in the data plane is challenging

Challenges

Signal is noisy

packets loss are routinely observed

Signal fades away quickly

due to the exponential backoff

Signal is compounded over many small ones

requires per-connection tracking, which is hard to scale

We solve these challenges by considering a subset of the signal that we carefully craft for maximal signal-to-noise ratio

Solutions

Signal is noisy

Focus on retransmissions caused by bursty losses

Signal fades away quickly

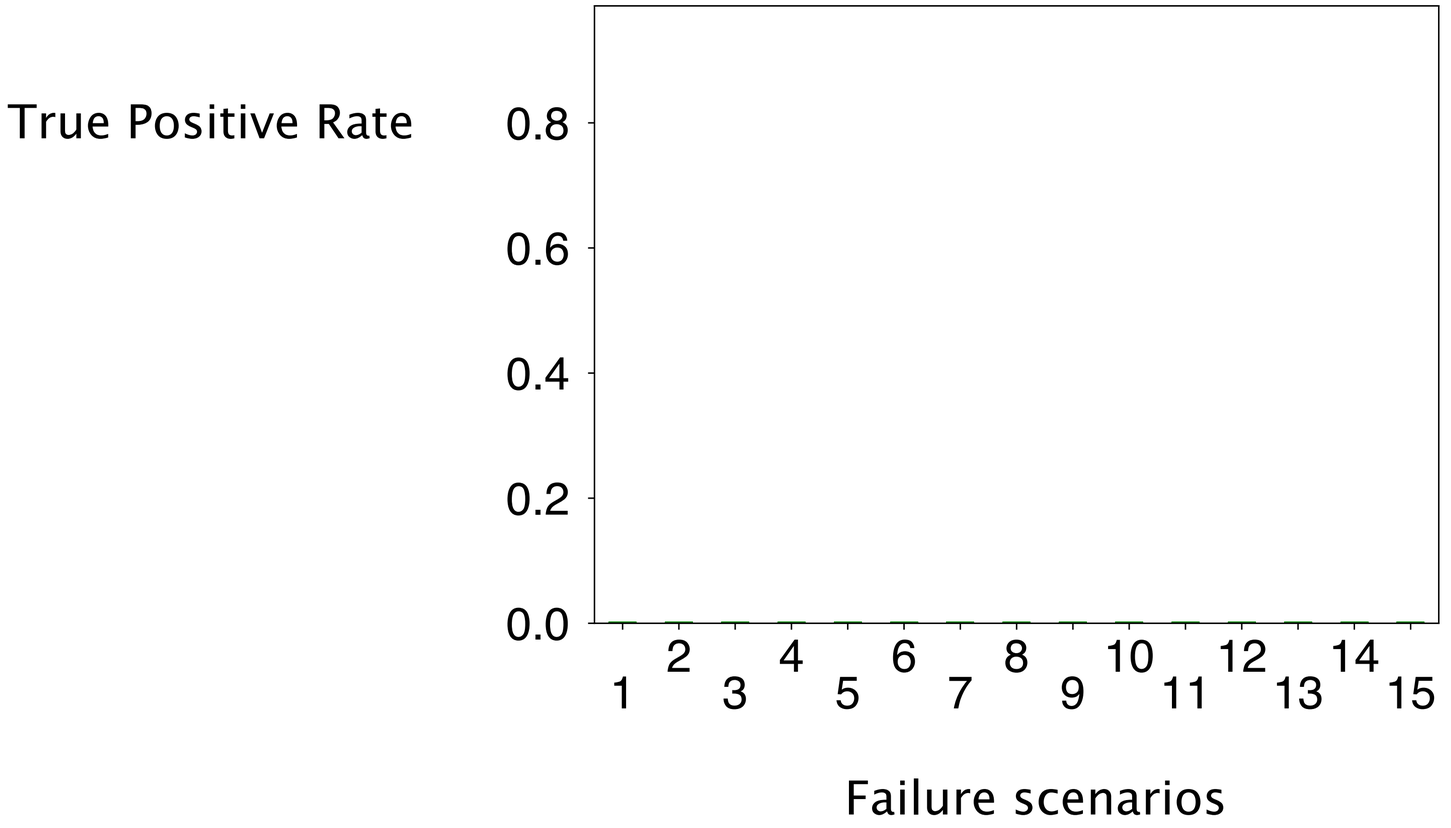
Focus on active flows (fast to retransmit after a failure)

Signal is compounded over many small ones

Rely on scalable data structures and sampling

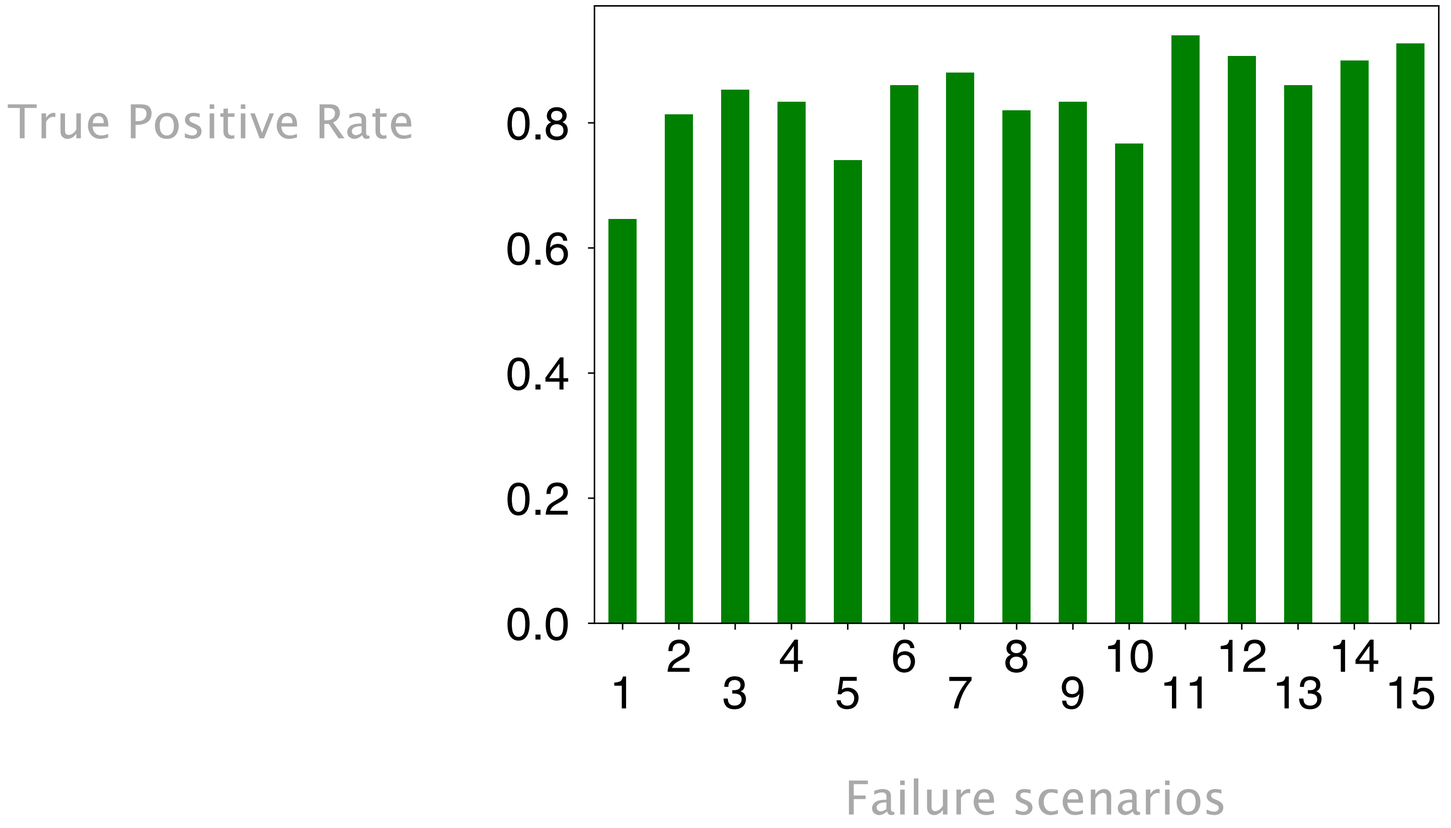
Blink works well in practice,

with an inference accuracy above 80% in most cases



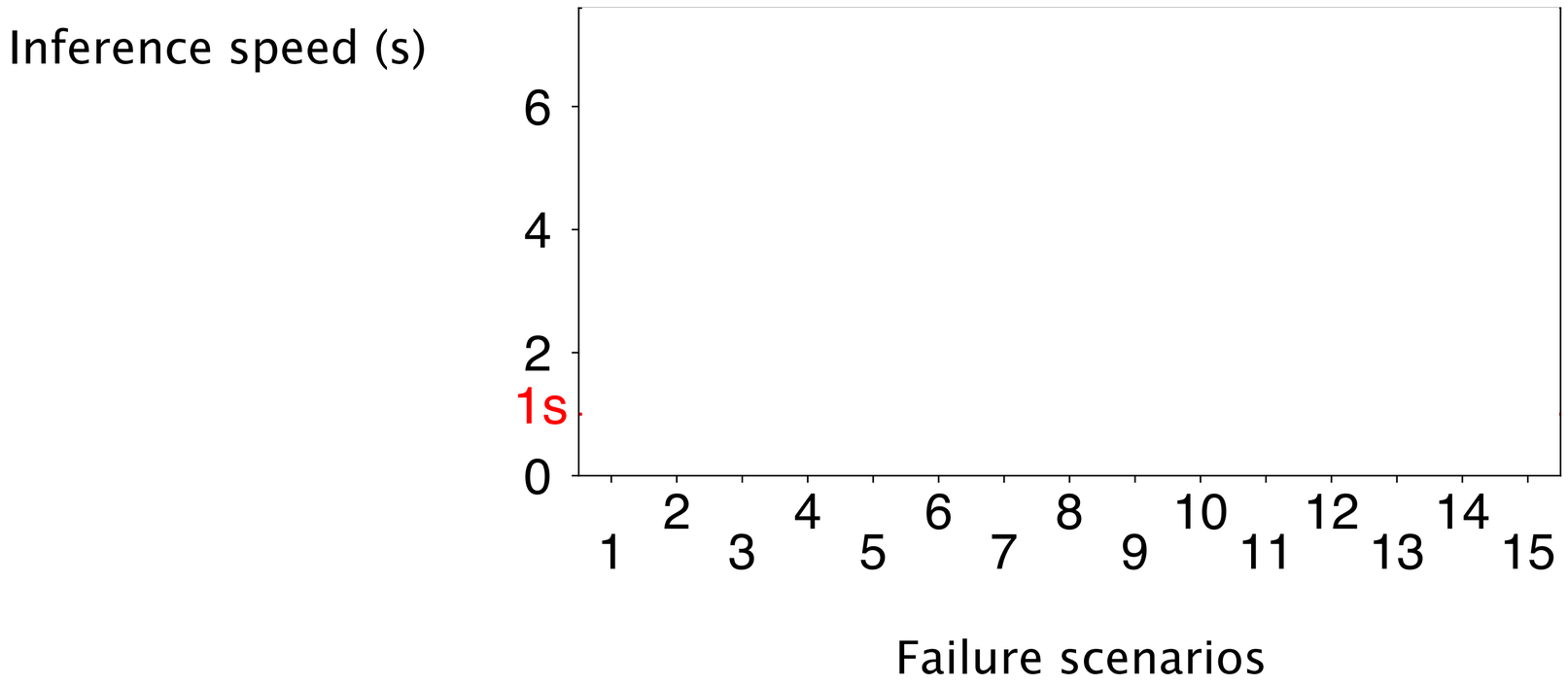
Blink works well in practice,

with an inference accuracy above 80% in most cases



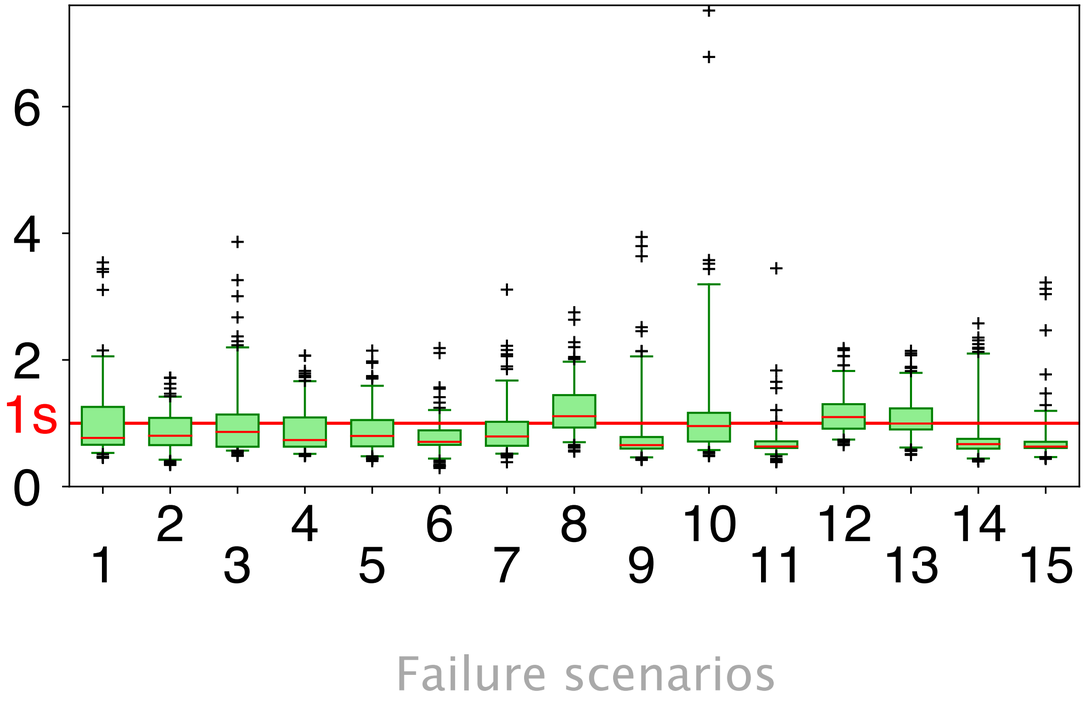
Blink works well in practice,

with an inference speed below 1 second in most cases



Blink works well in practice,

with an inference speed below 1 second in most cases



We intend to generalize the signals we track,
for a wider variety of applications

reliability	reroute around fine-grained failures
performance	steer traffic along the most performant path
security	detect unwanted traffic redirection

Part II: Taking control

applications

frameworks

methods

monitoring/inference

FitNets: An Adaptive Framework to Learn Accurate Traffic Distributions

Alexander Dietmüller

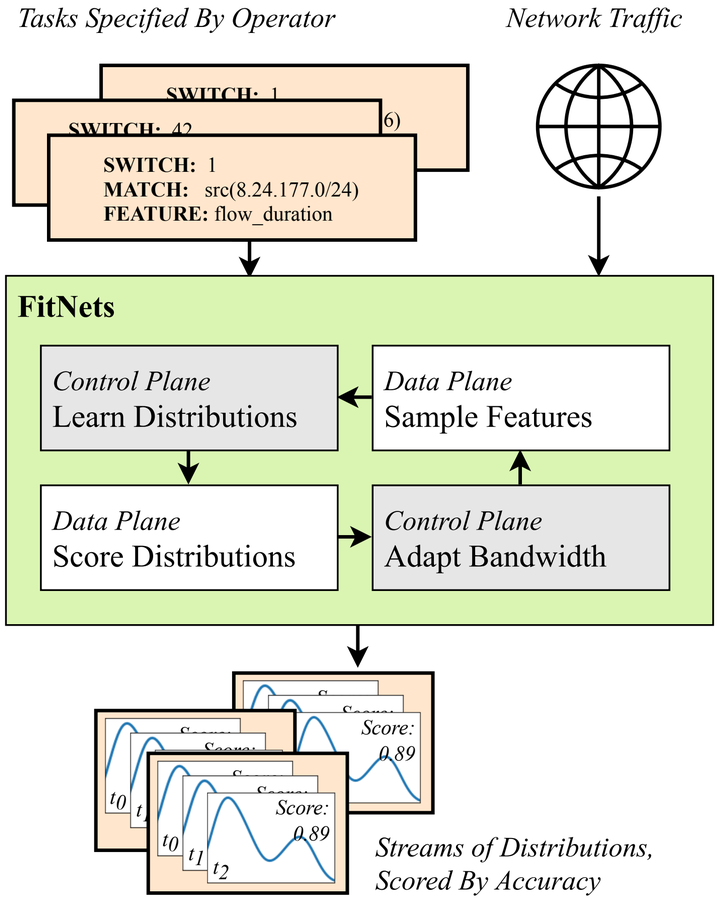
Albert
Gran Alcoz

Laurent Vanbever

FitNets

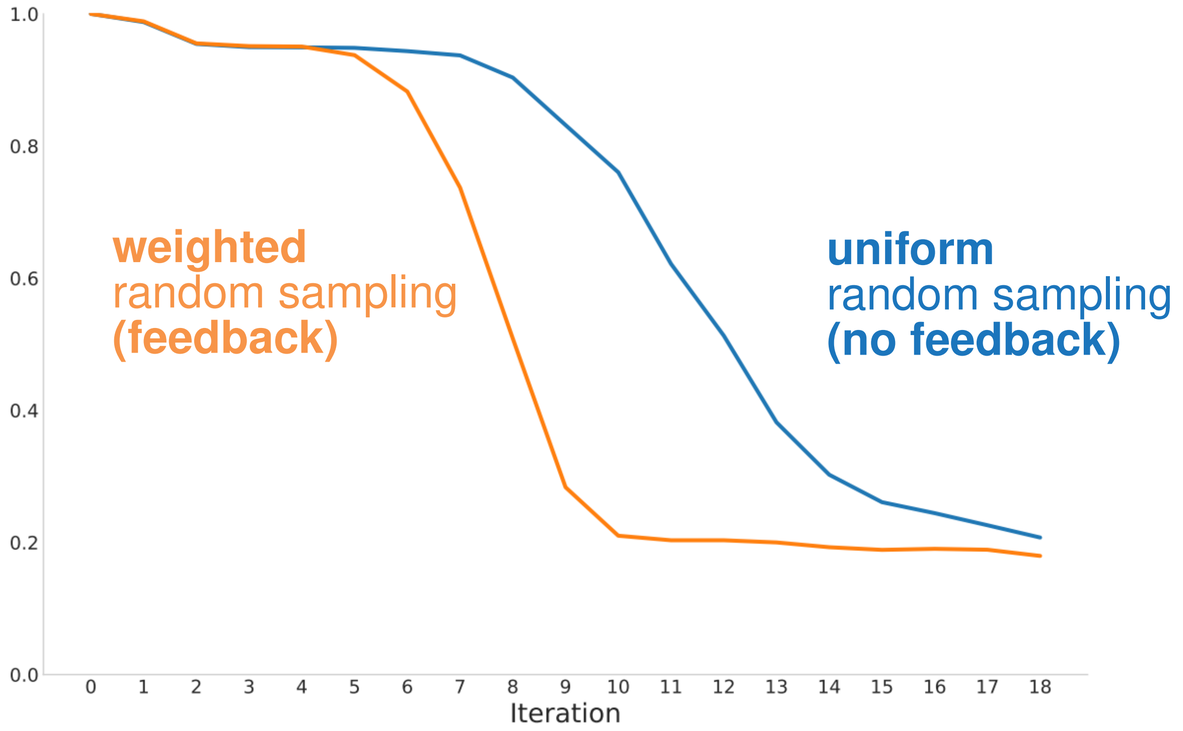
An adaptive network monitoring framework

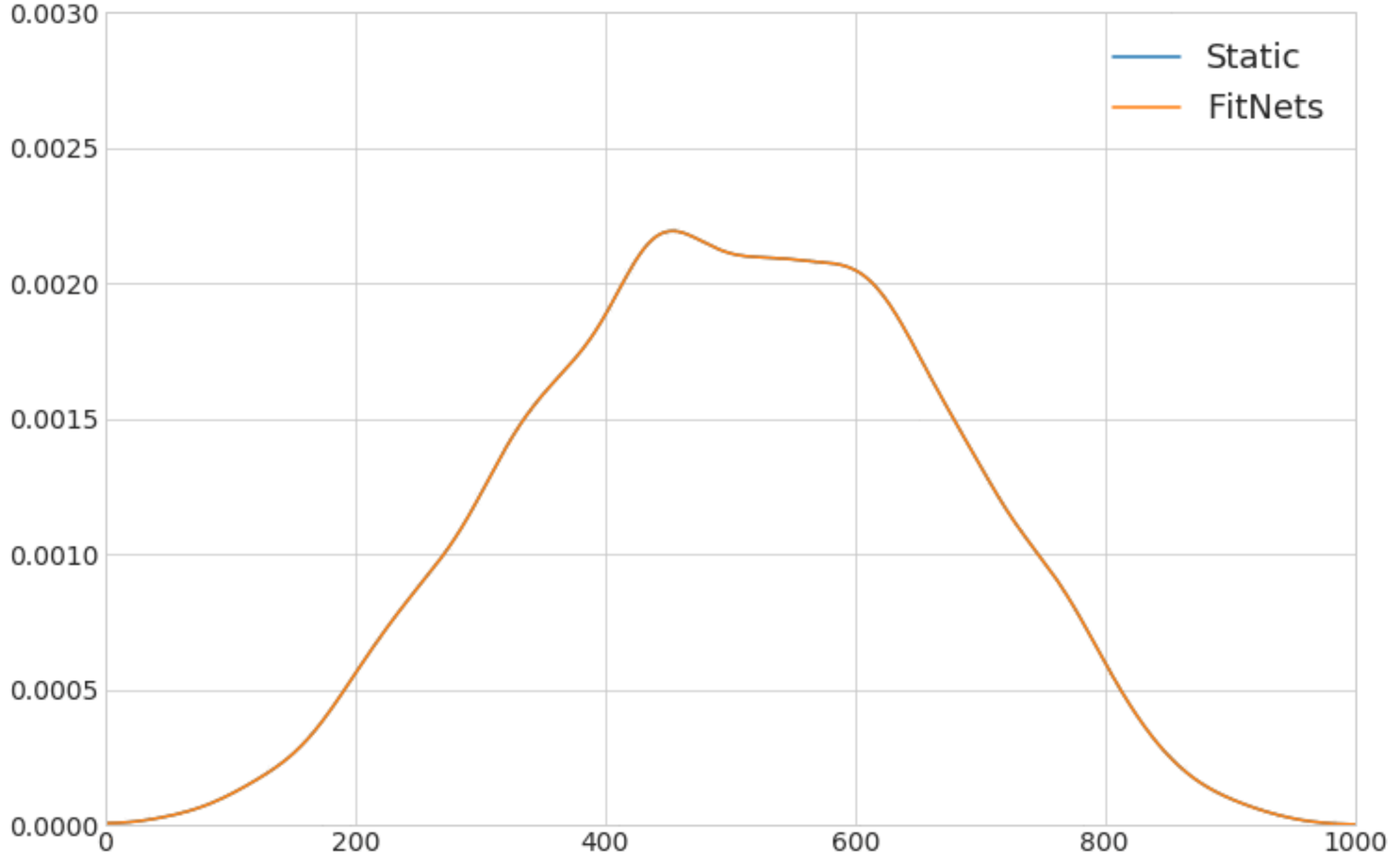
- computes traffic distribution
  in the control plane, on traffic samples
- scores these distributions
  in the data plane, on *all* traffic
- adapts the sampling rate
  according to the score



FitNets learns distribution better and faster than simply randomly sampling traffic

Distance from
true distribution
(Jensen-Shannon)




0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
200
400
600
800
1000
Static
FitNets

Part II: Taking control

applications

frameworks

methods

monitoring/inference

pForest: In-Network Inference with Random Forests

Coralie Busse-Grawitz

Roland Meier

Alexander Dietmüller

Tobias Bühler

Laurent Vanbever

Why (and How) Networks Should Run Themselves

Nick Feamster and Jennifer Rexford
Princeton University

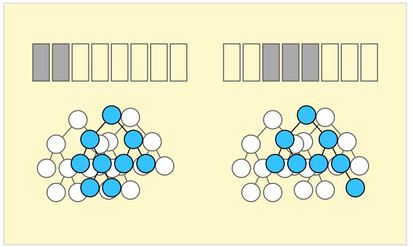
Abstract

The proliferation of networked devices, systems, and applications that we depend on every day makes managing networks more important than ever. The increasing security, availability, and performance demands of these applications suggest that these increasingly difficult network management

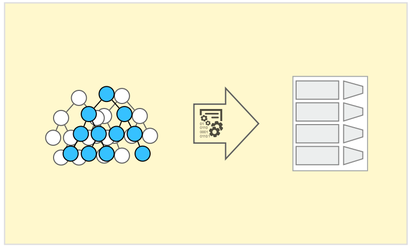
relationships between them and user quality of experience become increasingly complex. Twenty years ago, we had some hope of (and success in) creating clean, closed-form models of individual protocols, applications, and systems [4, 24]; today, many of these are too complicated for closed-form analysis. Prediction problems such as determining how search query response time would vary in response to the placement

Performing automated, real-time inference: The past ten years has demonstrated significant promise in using machine learning to both detect and predict network attacks; we must build on the increasing amount of work in automated infer-

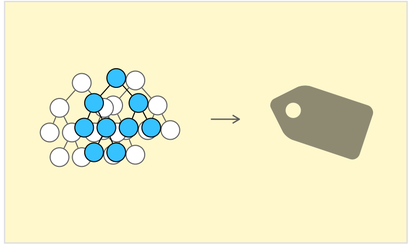
pForest: In-Network Inference with Random Forests



Optimizing random forest models
for programmable network devices



Compiling random forest models
to programmable network devices



Performing runtime classification
at Tbps

Early, accurate & efficient classification of an ongoing flow as an optimization problem

Given a labeled dataset $\mathcal{F}$ and a threshold τ

find a classifier C such that

- accuracy(C) >= τ
- C fits in programmable network devices

while

- minimizing memory usage
- maximizing classification speed

Part II: Taking control

applications

frameworks

methods

some of our challenges
lying ahead

Building and operating truly self-driving networks will require us to overcome fundamental challenges

data

correctness

interpretability

data

correctness

interpretability

data is network-dependent

poor generalization capabilities

interesting events are rare

few interesting examples (if any)

data can easily be polluted

by anyone with a Internet connection…

data

correctness

interpretability

how do we…

ensure correctness?

network should be reachable

guarantee stability?

agents can clash with each other

reason about optimality?

data-driven > non-data-driven (?)

data

correctness

interpretability

how can operators…

reason about self-driving networks?

especially in partial deployment

debug their self-driving networks?

manual override

Self-Driving Networks

Breaking new ground in network automation

1 operator assistance

2 partial automation

Part I
assisting operators

3 conditional automation

4 high automation

Part II
taking control

5 full automation

too futuristic? ☺

Huge thanks to my research group!

Ahmed

Roland

Mirja

Alex

Edgar

Rüdiger

Tobias

Albert

Thomas

Maria

Merci à tous!

Group

Maria Apostolaki
Rüdiger Birkner
Tobias Bühler
Edgar Costa Molero
Alexander Dietmüller
Ahmed El-Hassany
Albert Gran Alcoz
Thomas Holterbach
Roland Meier

Bernhard Ager
David Gugelmann
Mirja Kühlewind
Brian Trammell

Collaborators

Alberto Dainotti
Dana Drachsler Cohen
Nick Feamster
Arpit Gupta
Markus Happe
Vincent Lenders
Jennifer Rexford
Michael Schapira
Petar Tsankov
Stefano Vissicchio
Martin Vechev
Aviv Zohar

+ many more

Colleagues @ITET

Lothar Thiele
Roger Wattenhofer

Gian-Luca Bona (committee)
Amos Lapidoth
John Lygeros

+ the entire D-ITET faculty & staff

Colleagues @ETH

Schulleitung
Rektorat
Professoren Ruderteam

+ ***many*** more

Self-Driving Networks

Breaking new ground in network automation

Laurent Vanbever

nsg.ee.ethz.ch

ETH Zurich

May 29 2019