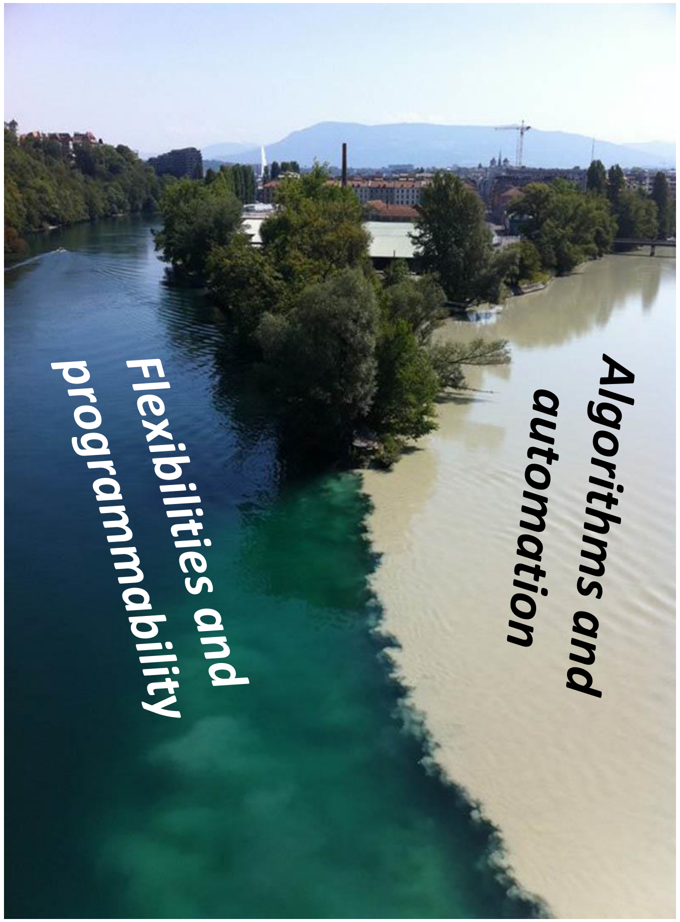
Self-Driving Networks: Use Cases, Approaches, and Research Challenges

Stefan Schmid

"We cannot direct the wind,
but we can adjust the sails."
(Folklore)

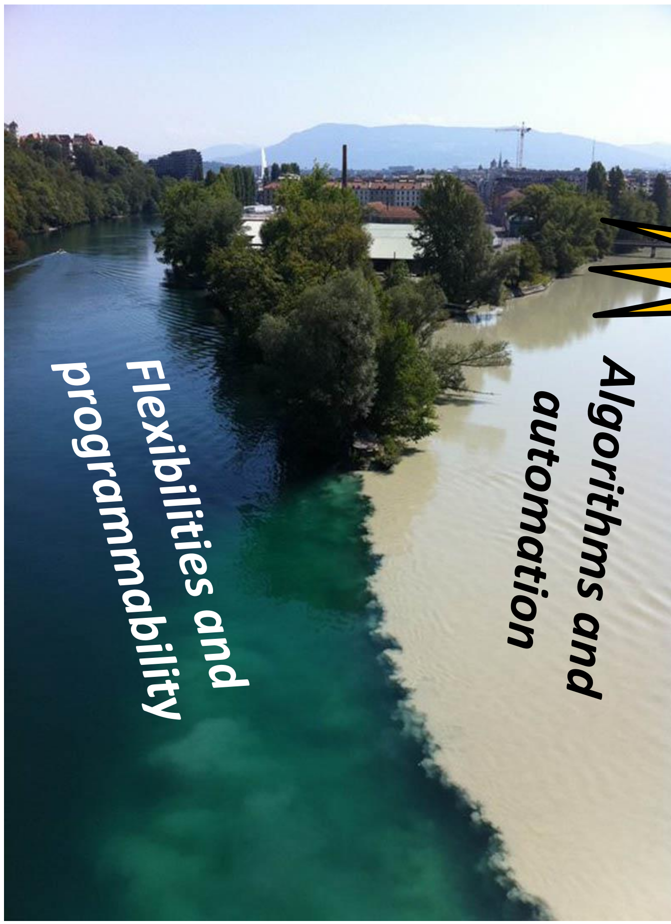
Acknowledgements:

It`s a Great Time to Be a Networking Researcher!



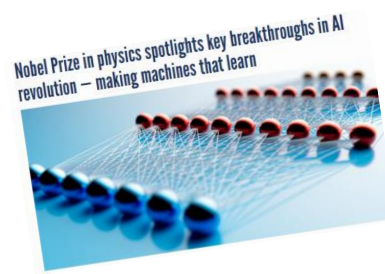
Rhone and Avre (Switzerland)

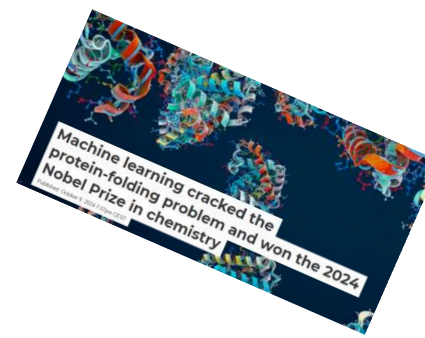
It`s a Great Time to Be a Networking Researcher!



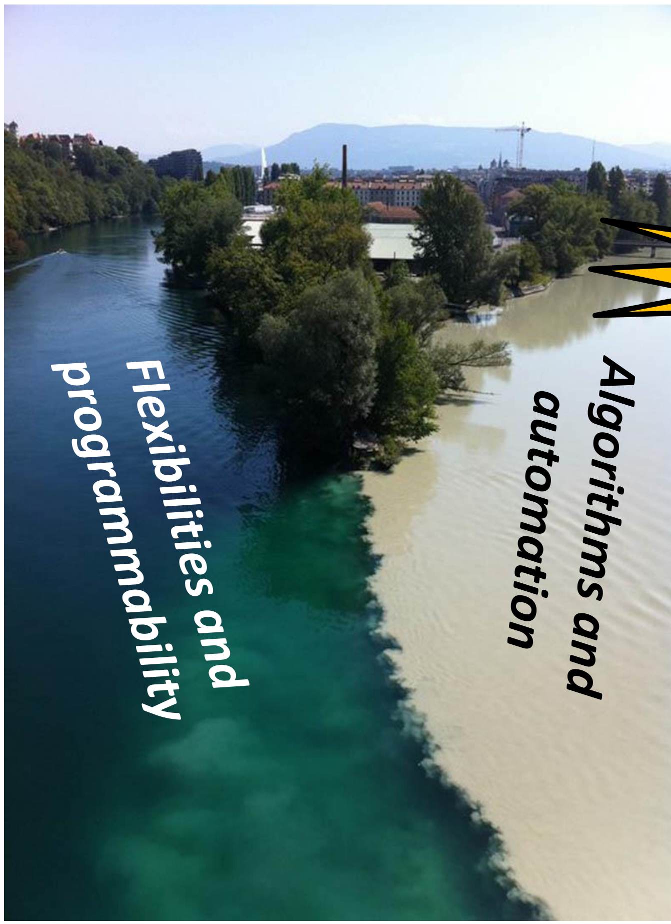
Rhone and Avre (Switzerland)

AI/ML everywhere!



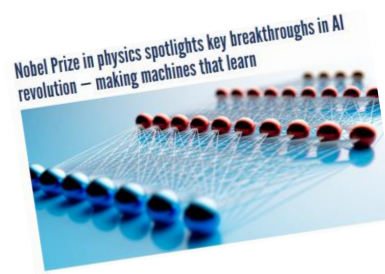


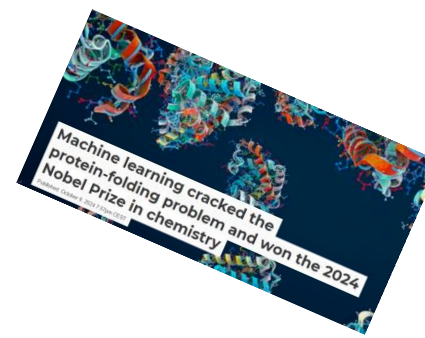
It`s a Great Time to Be a Networking Researcher!



Rhone and Avre (Switzerland)

AI/ML everywhere!





Economics and literature?

It`s a Great Time to Be a Networking Researcher!

Credits: George Varghese

It`s a Great Time to Be a Networking Researcher!

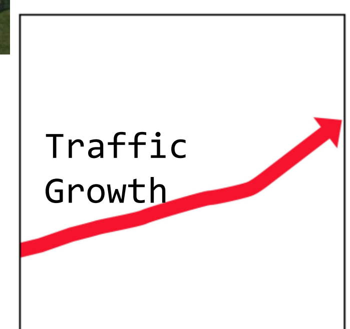
Enables and motivates **self-driving networks**!

Credits: George Varghese

Explosive Traffic

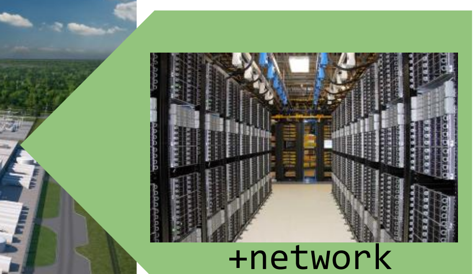
Datacenters (“hyper-scale”)

Interconnecting networks:
a **critical infrastructure**
of our digital society.



Explosive Traffic

Datacenters (“hyper-scale”)



Interconnecting networks:
a **critical infrastructure**
of our digital society.

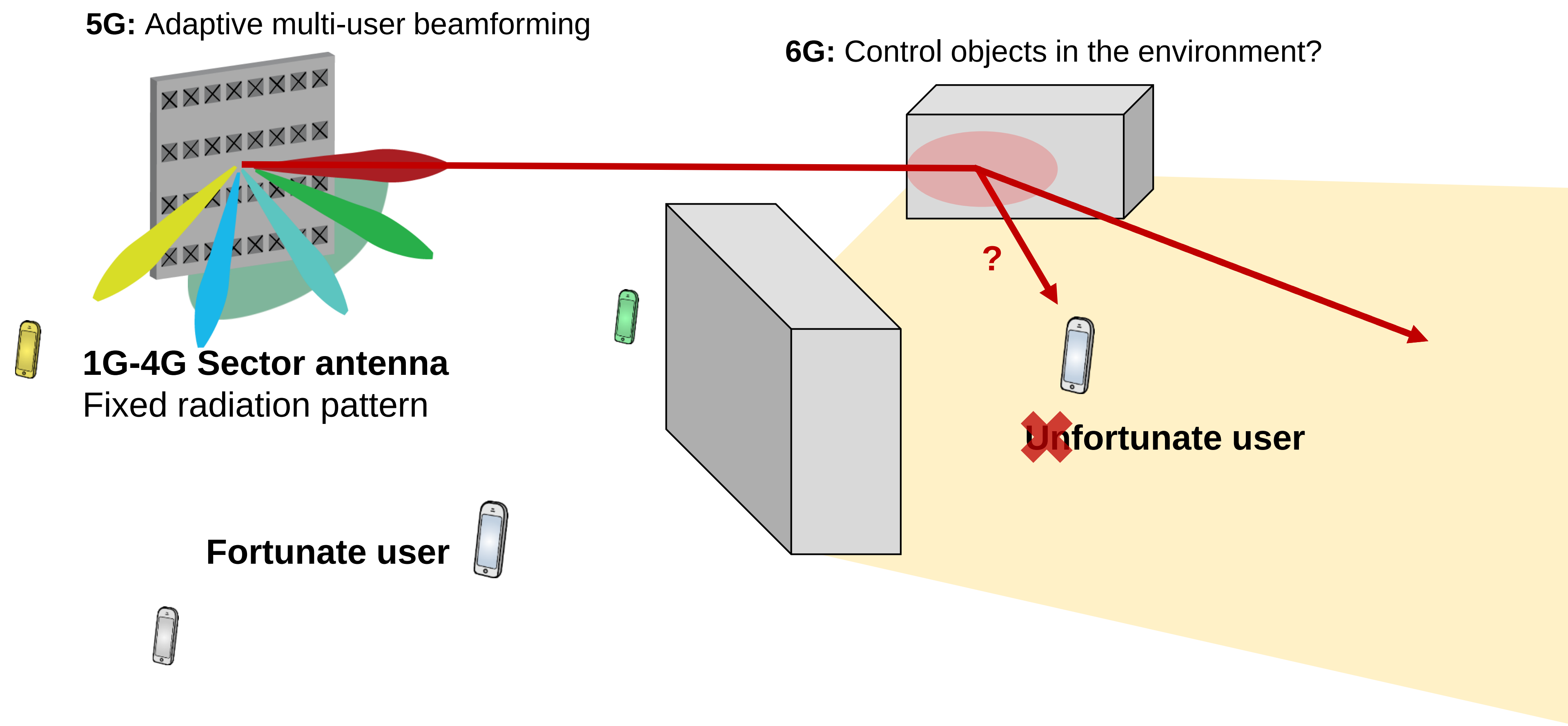
Credits: Marco Chiesa

Fast growing traffic also in…

… wireless and mobile

From generation to generation more…

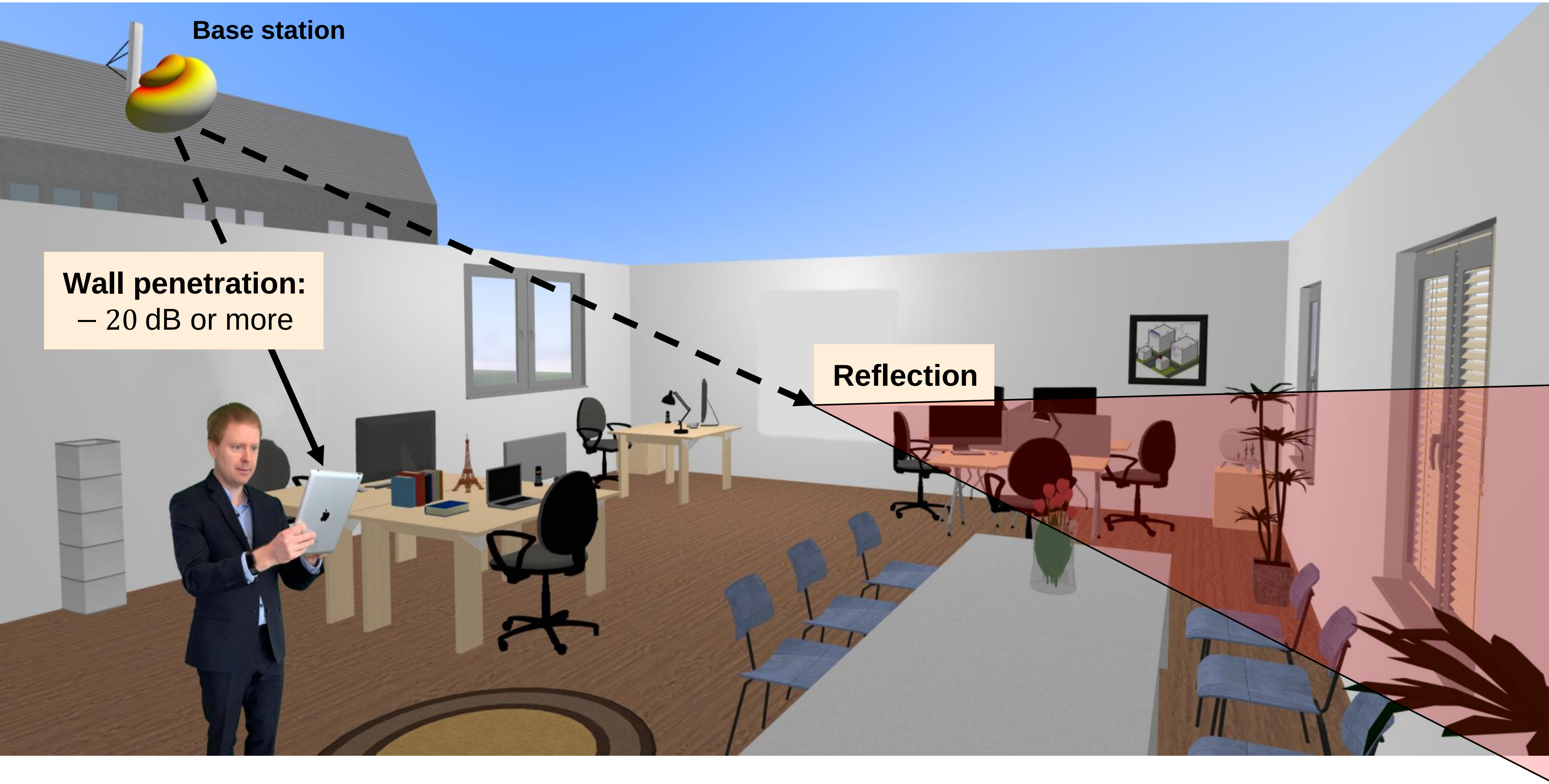
Exciting Flexibilities



credit: Emil Björnson, Christos Liaskos

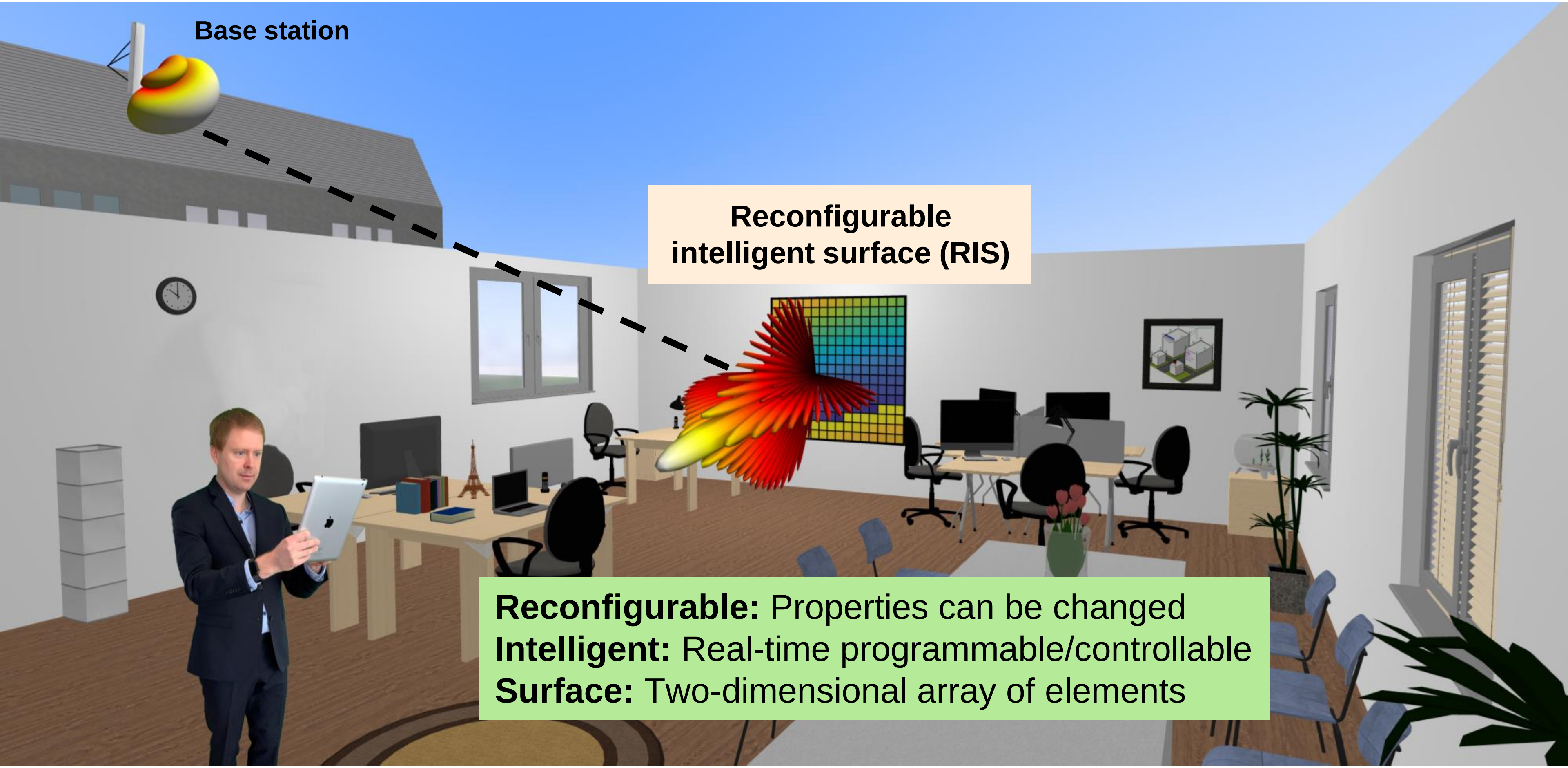
Traditionally limited by

Line of Sight Only



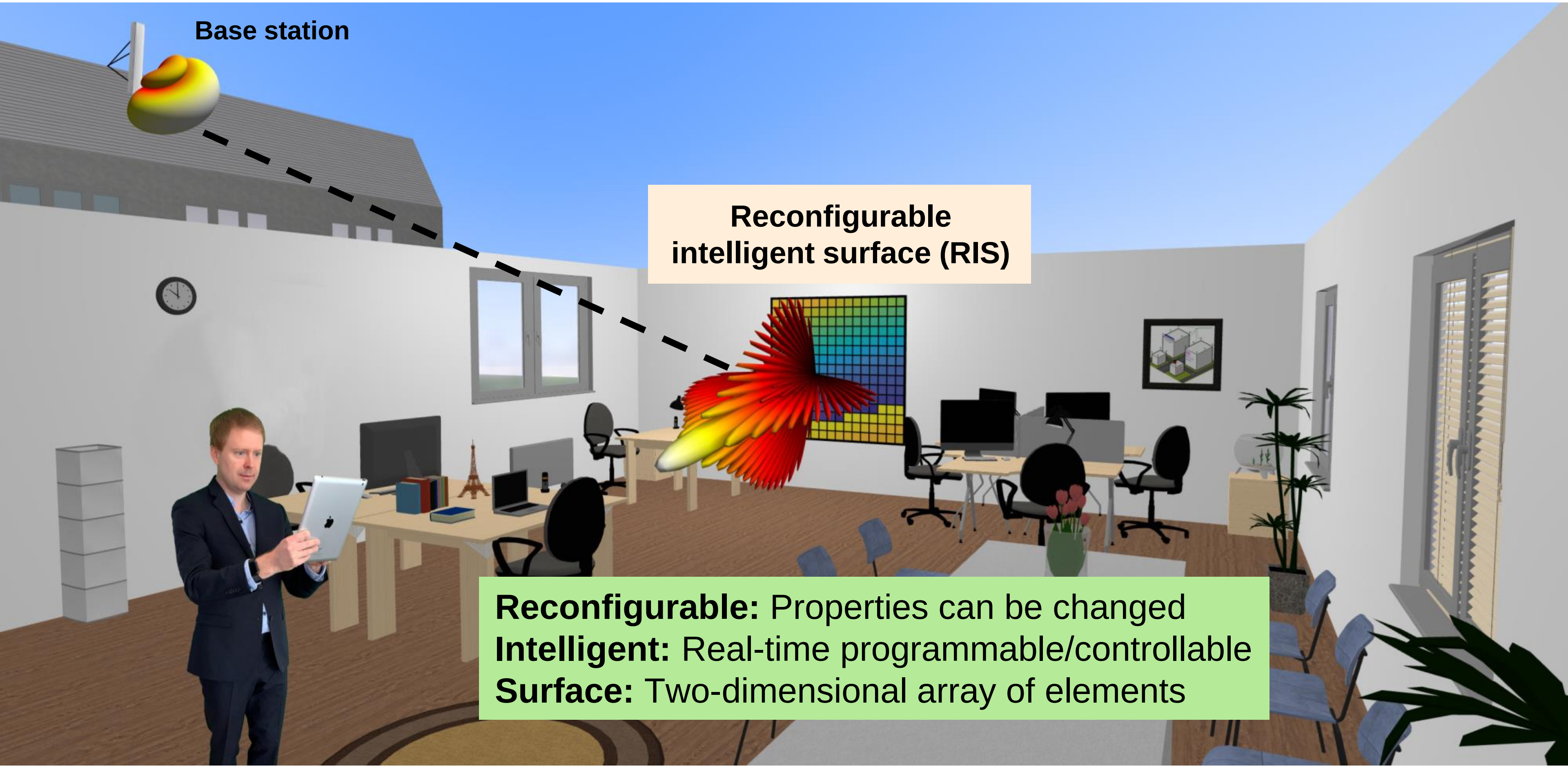
credit: Emil Björnson

Reconfigurable Intelligent Surfaces: Extend to Virtual Line of Sight



credit: Emil Björnson

Reconfigurable Intelligent Surfaces: Extend to Virtual Line of Sight



Literature: Software-Defined Reconfigurable Intelligent Surfaces: From Theory to End-to-End Implementation. Liaskos et al. Proceedings IEEE, 2022.

Great opportunities but come with…

Challenges

- With growing ***demand*** for networks, also increasing ***dependability***
- Important step toward dependable networks: ***modelling***…
- … and ***automation*** (also using formal methods)!
- Contributions from the ICIN community critical

It’s high time for computer-aided designs!

Reality vs Requirements

Today, dependability requirements stand in contrast with reality:

Countries disconnected

Data Centre ▸ Networks

Google routing blunder sent Japan's Internet dark on Friday

Another big BGP blunder

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

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

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

Passengers stranded

British Airways' latest Total Inability To Support Upwardness of Planes* caused by Amadeus system outage

Stuck on the ground awaiting a load sheet? Here's why

By Gareth Corfield 19 Jul 2018 at 11:16 109 SHARE ▼

Even 911 affected

Officials: Human error to blame in Minn. 911 outage

According to a press release, CenturyLink told department of public safety that human error by an employee of a third party vendor was to blame for the outage

Aug 16, 2018

Duluth News Tribune

SAINT PAUL, Minn. — The Minnesota Department of Public Safety Emergency Communication Networks division was told by its 911 provider that an Aug. 1 outage was caused by human error.

Even tech-savvy companies struggle:

GoDaddy.com github SOCIAL CODING amazon web services

Credits: Laurent Vanbever, Nate Foster

It’s high time for computer-aided designs!

Reality vs Requirements

Today, dependability requirements stand in contrast with reality:

Countries disconnected

Data Centre ▸ Networks

Google routing blunder sent Japan's Internet dark on Friday

Another big BGP blunder

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

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

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

Passengers stranded

British Airways' latest Total Inability To Support Upwardness of Planes* caused by Amadeus system outage

Stuck on the ground awaiting a load sheet? Here's why

By Gareth Corfield 19 Jul 2018 at 11:16 109 SHARE ▼

BA flights around the world were grounded as a result of the Amadeus outage

Even 911 affected

Officials: Human error to blame in Minn. 911 outage

According to a press release, CenturyLink told department of public safety that human error by an employee of a third party vendor was to blame for the outage

Aug 16, 2018

Duluth News Tribune

SAINT PAUL, Minn. — The Minnesota Department of Public Safety Emergency Communication Networks division was told by its 911 provider that an Aug. 1 outage was caused by human error.

Mainly: human errors!

Even tech-savvy companies struggle:

GoDaddy.com github SOCIAL CODING amazon web services

It’s high time for computer-aided designs!

Reality vs Requirements

Today, dependability requirements stand in contrast with reality:

Countries disconnected

Data Centre ▸ Networks

Google routing blunder sent Japan's Internet dark on Friday

Another big BGP blunder

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

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

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

Passengers stranded

British Airways' latest Total Inability To Support Upwardness of Planes* caused by Amadeus system outage

Stuck on the ground awaiting a load sheet? Here's why

By Gareth Corfield 19 Jul 2018 at 11:16 109 SHARE ▼

Even 911 affected

Officials: Human error to blame in Minn. 911 outage

According to a press release, CenturyLink told department of public safety that human error by an employee of a third party vendor was to blame for the outage

Aug 16, 2018

Duluth News Tribune

SAINT PAUL, Minn. — The Minnesota Department of Public Safety Emergency Communication Networks division was told by its 911 provider that an Aug. 1 outage was caused by human error.

Mainly: human errors!

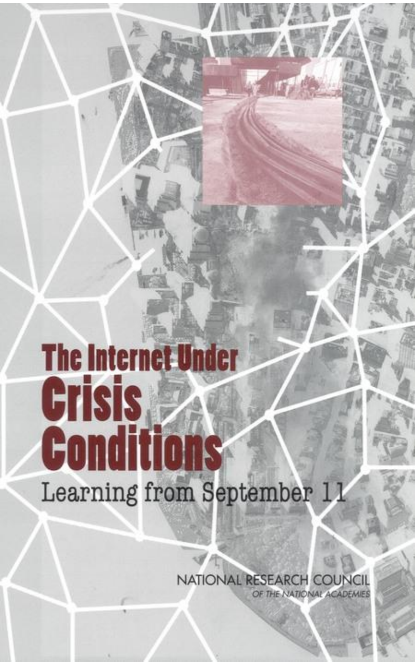
Even tech-savvy companies struggle:

Go Daddy.com github SOCIAL CODING amazon web services

Wireless particularly challenging to model!

An Anecdote

- Report by the National Research Council about ***9/11/2001 attacks***
- While the core Internet infrastructure installed in the WTC was down, the overall Internet was ***more stable*** than usual
- … because operators stopped touching network devices?!



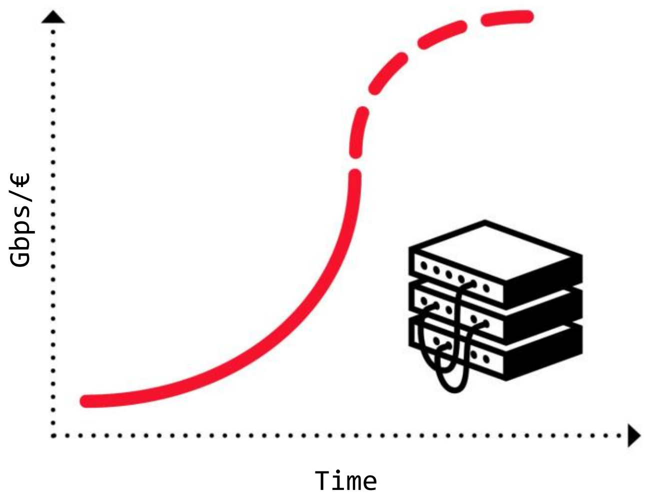
Roadmap

- Performance: Self-adjusting datacenter networks
- Modelling: How to model workloads, such as ML workloads?
- Dependability: Self-correcting MPLS networks
- More Use cases for self-driving networks

Datacenters Today

Huge Infrastructure, Inefficient Use

- Network equipment reaching capacity limits
  - ⇀ Transistor density rates stalling
  - ⇀ "End of **Moore's Law** in networking"
- Hence: more equipment, larger networks
- Resource intensive and: **inefficient**



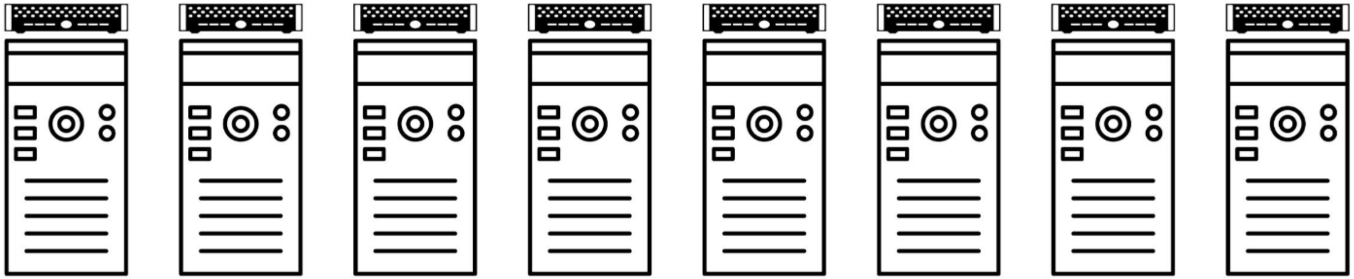
[1] Source: Microsoft, 2019

Annoying for companies, **opportunity** for researchers!

Root Cause

Fixed and Demand-Oblivious Topology

How to interconnect?

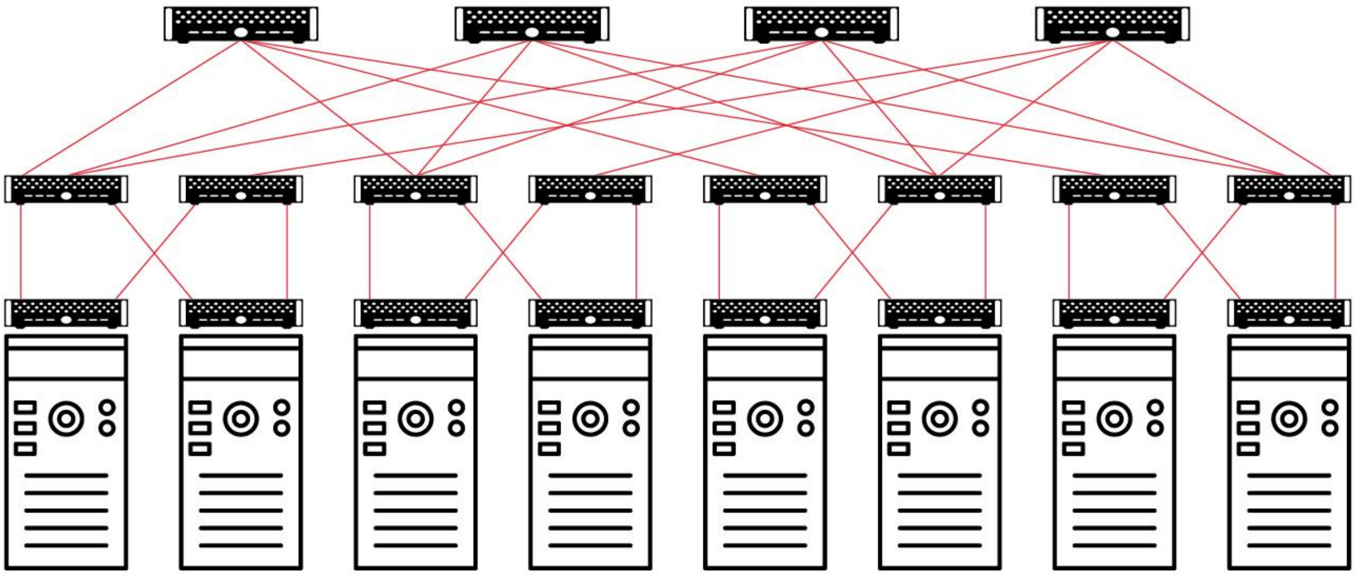


Root Cause

Fixed and Demand-Oblivious Topology

⇢ Example: fat-tree topology (bi-regular)

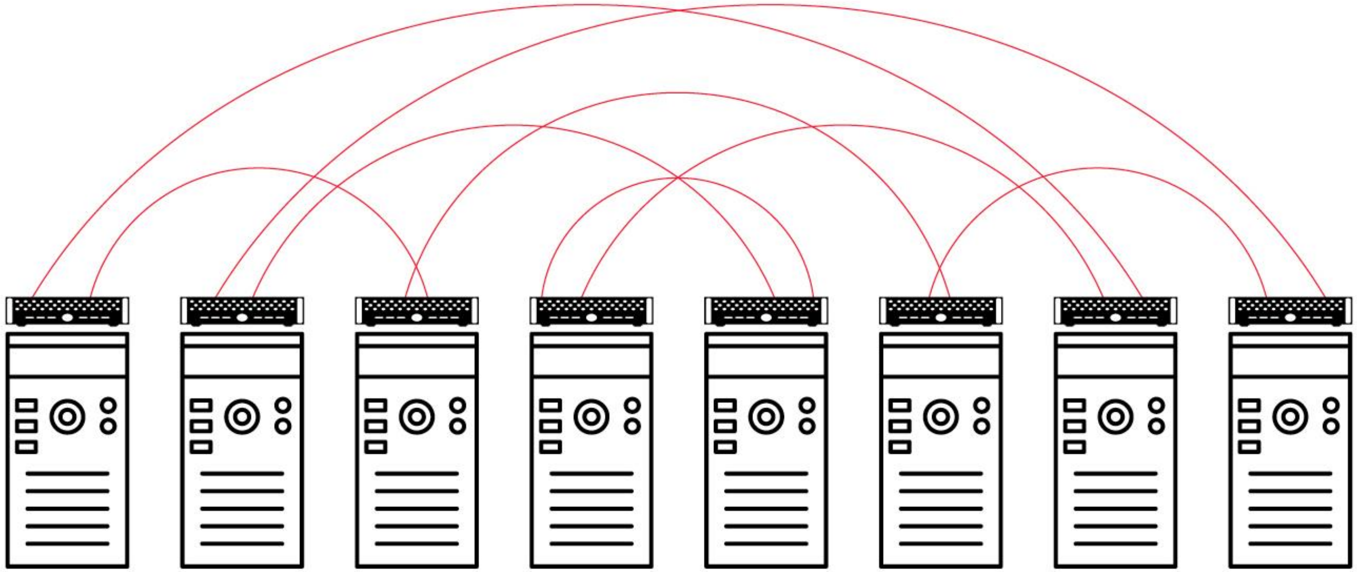
→ 2 types of switches: top-of-rack (ToR) connect to hosts, additional switches connecting switches to increase throughput



Root Cause

Fixed and Demand-Oblivious Topology

- Example: expander topology (uni-regular)
  - → Only 1 type of switches: lower installation and management overheads

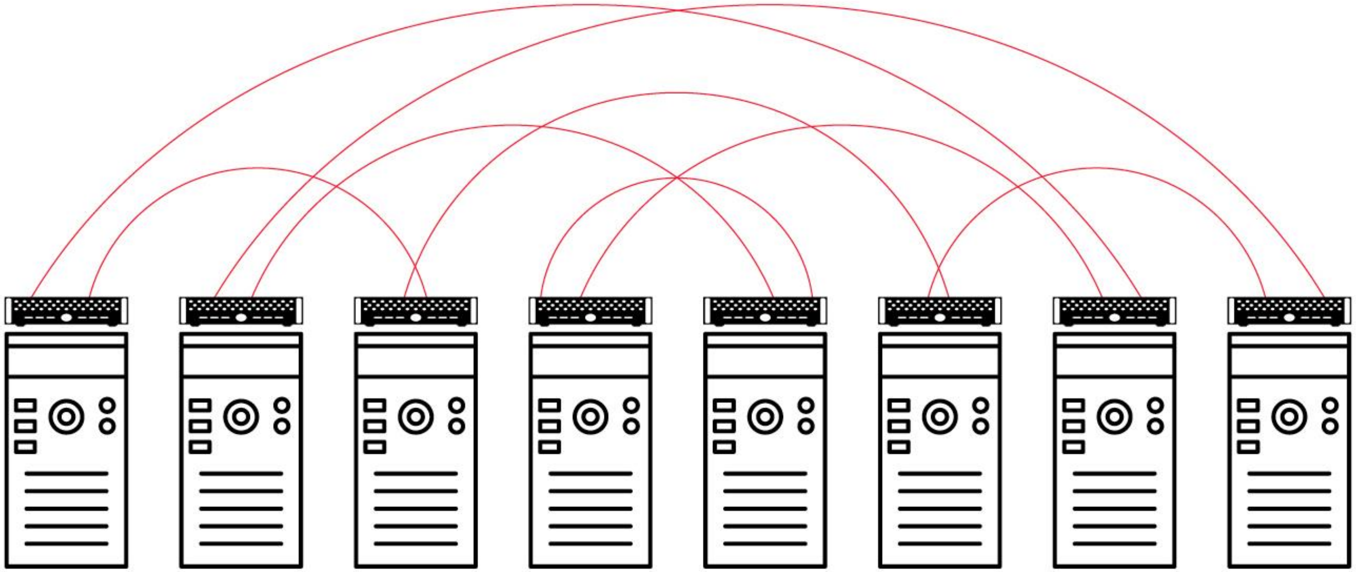


Root Cause

Fixed and Demand-Oblivious Topology

Highway which ignores actual traffic: frustrating!

- Example: expander topology (uni-regular)
  - → Only 1 type of switches: lower installation and management overheads



Root Cause

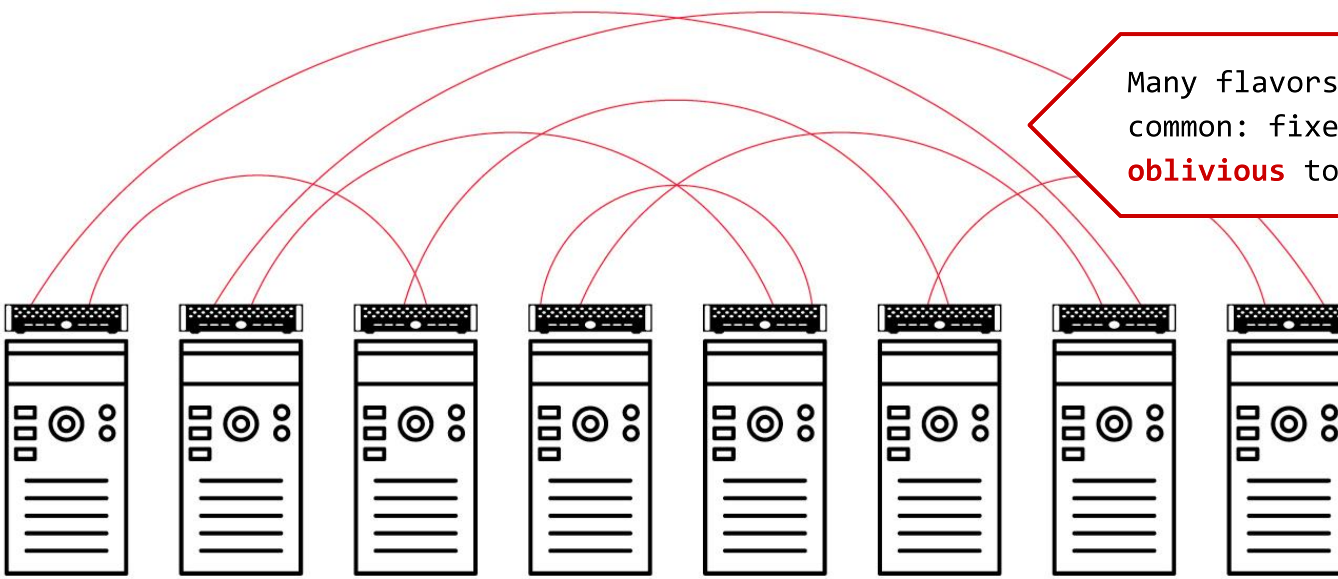
Fixed and Demand-Oblivious Topology

Highway which ignores actual traffic: frustrating!

⇢ Example: expander topology (uni-regular)

→ Only 1 type of switches:

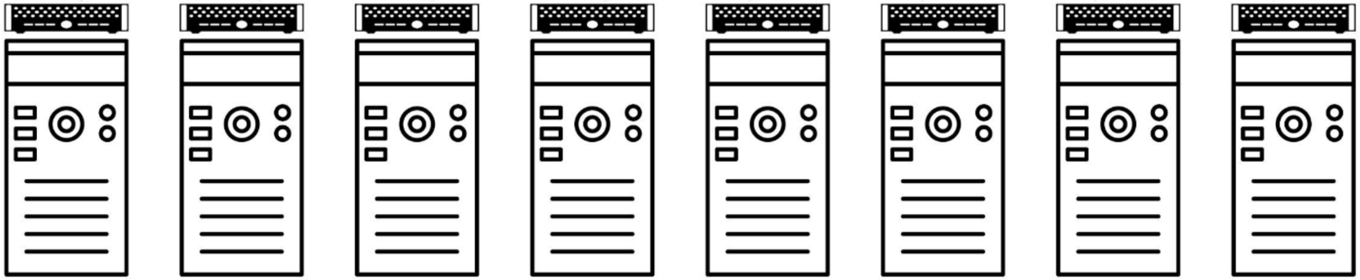
lower installation and management overheads



Many flavors, but in common: fixed and **oblivious** to actual demand.

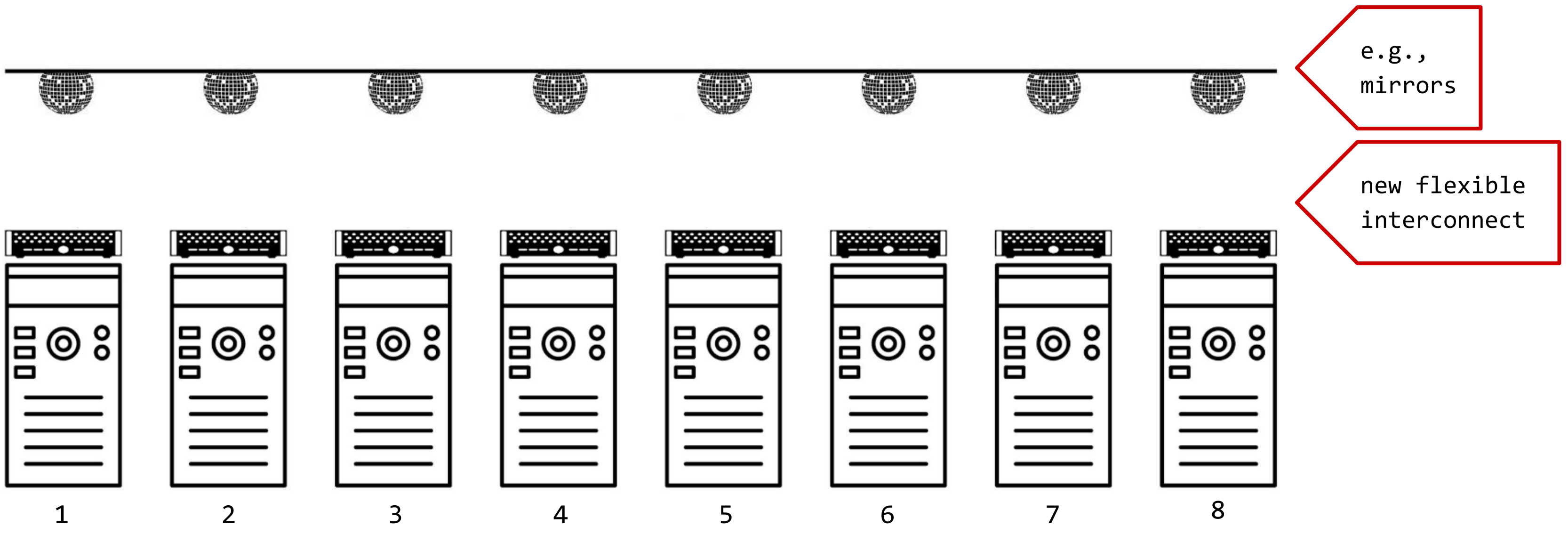
A Vision

Flexible and Demand-Aware Topologies



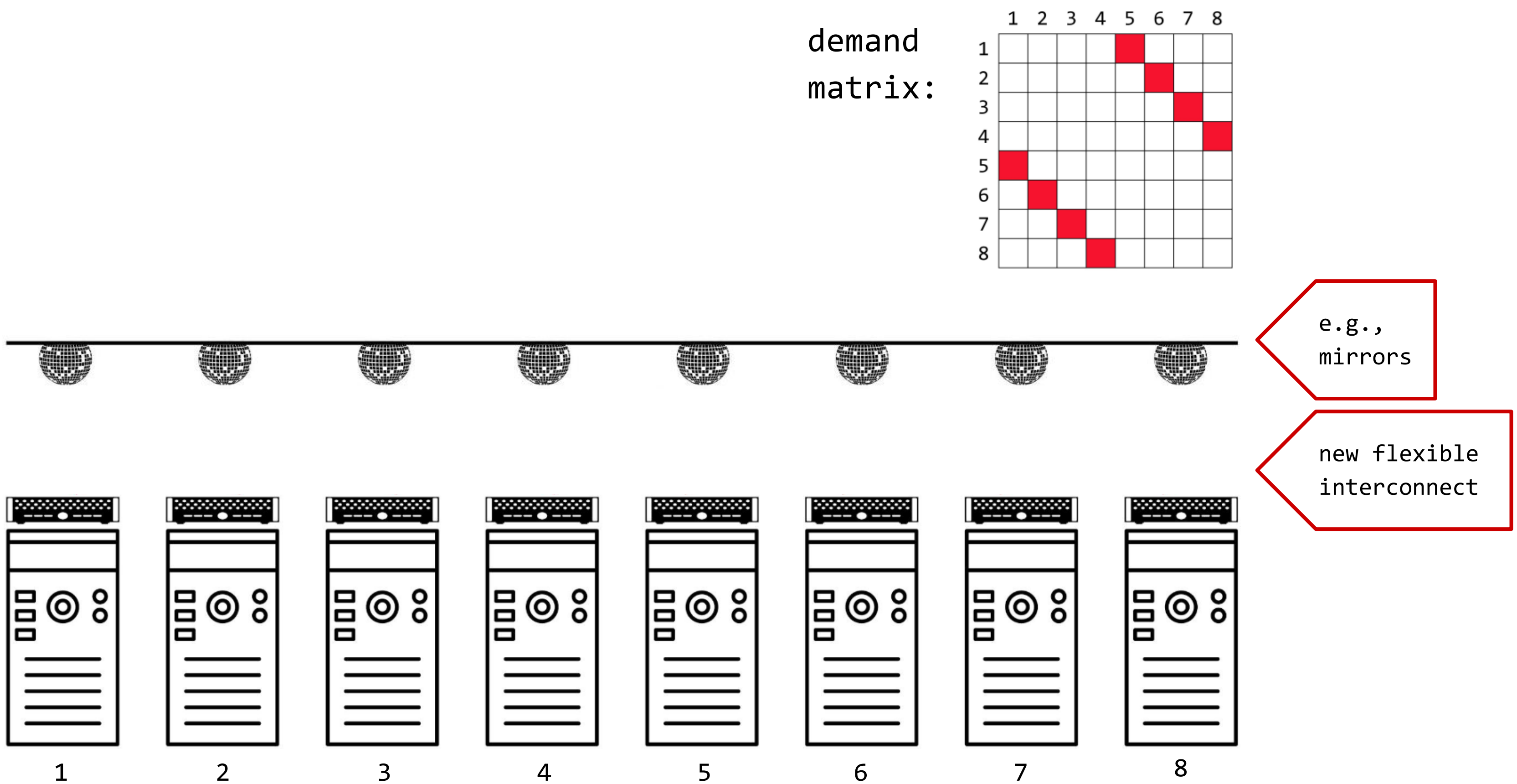
A Vision

Flexible and Demand-Aware Topologies



A Vision

Flexible and Demand-Aware Topologies

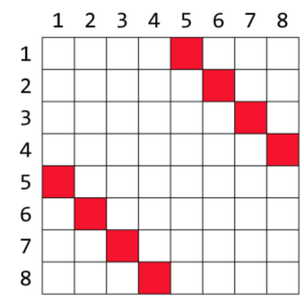


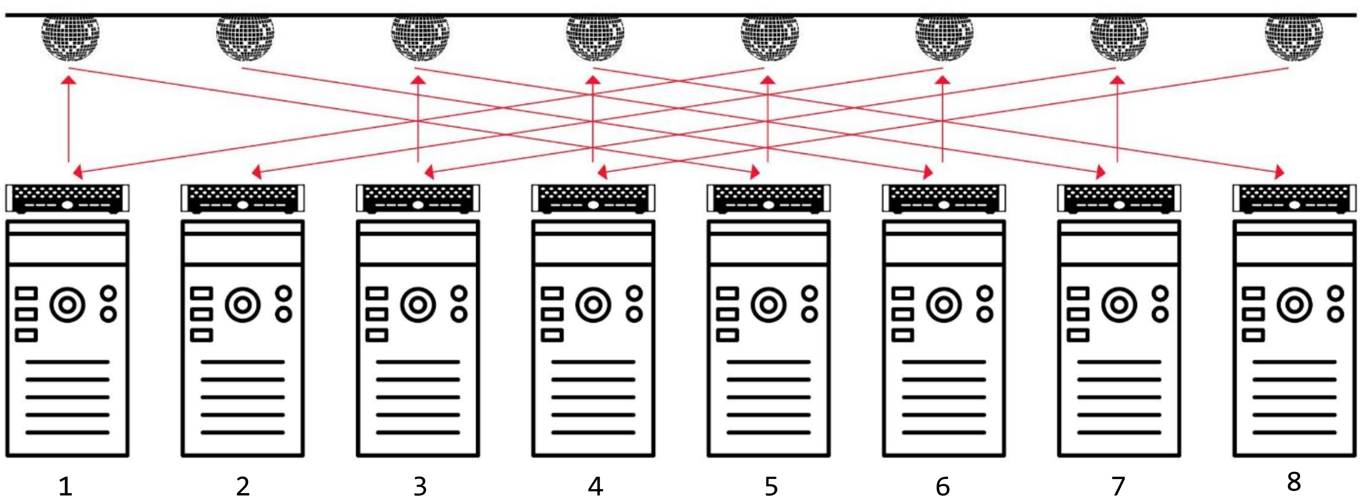
A Vision

Flexible and Demand-Aware Topologies

Matches demand

demand matrix:



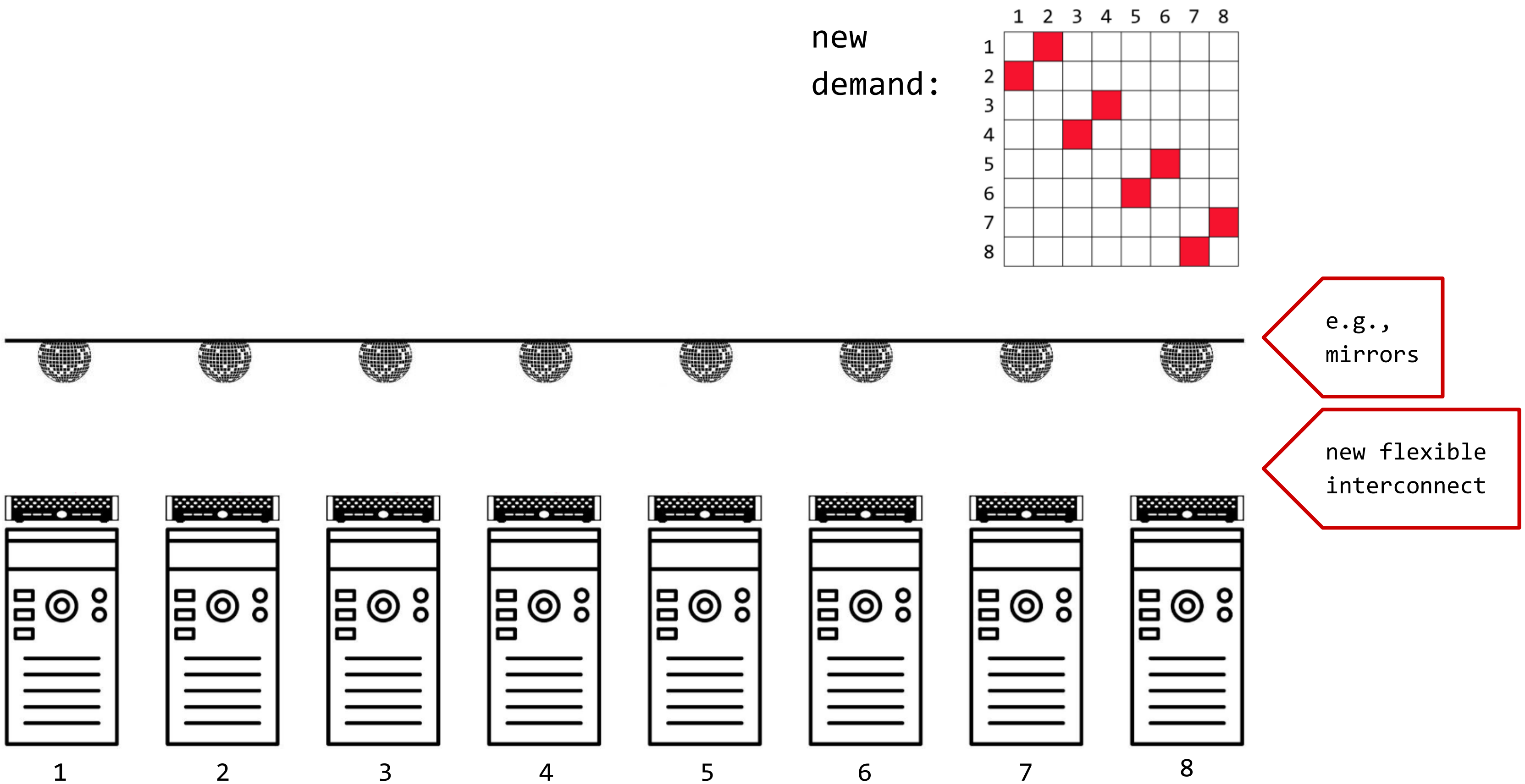


e.g., mirrors

new flexible interconnect

A Vision

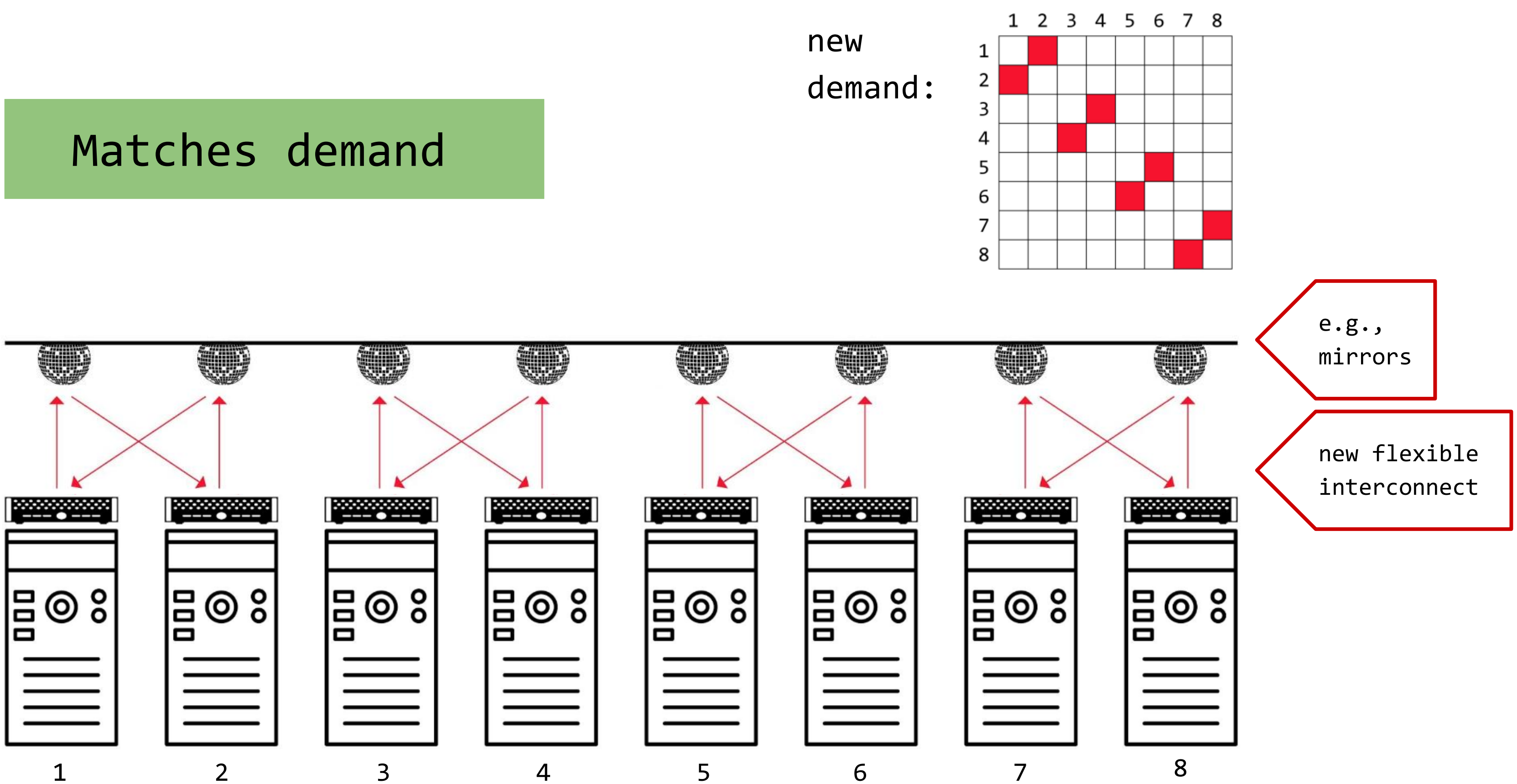
Flexible and Demand-Aware Topologies



A Vision

Flexible and Demand-Aware Topologies

Matches demand

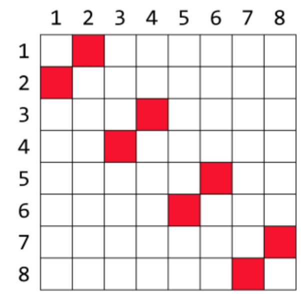


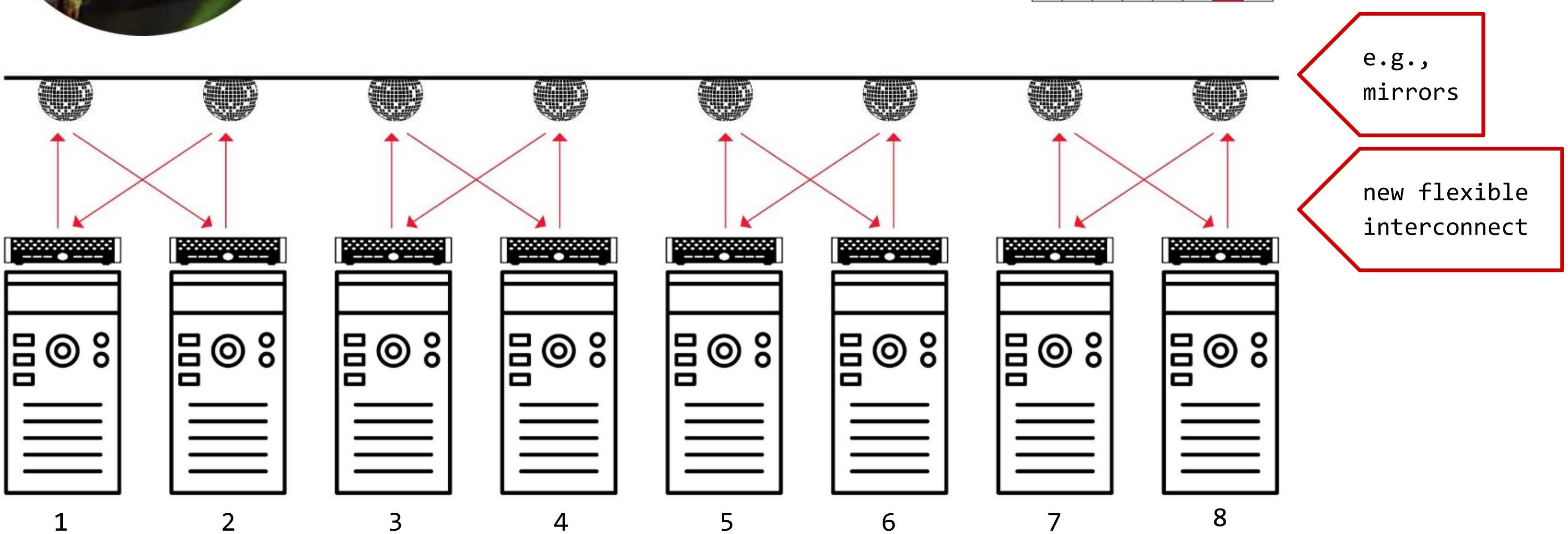
A Vision

Flexible and Demand-Aware Topologies

Self-Adjusting Networks

new demand:





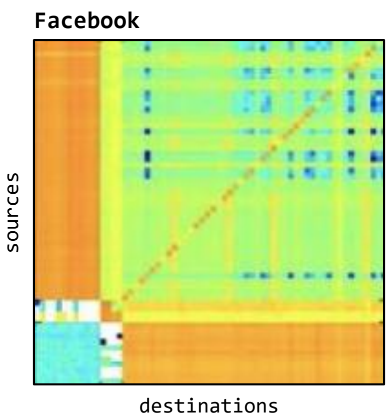
“Yin and Yang”-Networking!

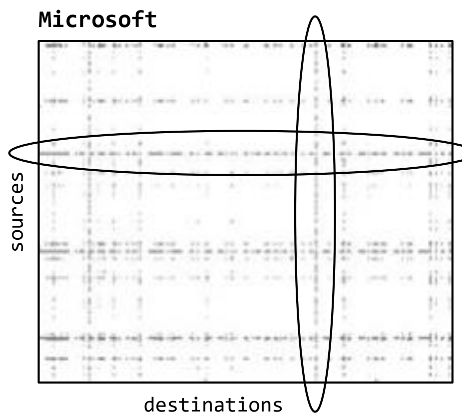
The Motivation

Much Structure in the Demand

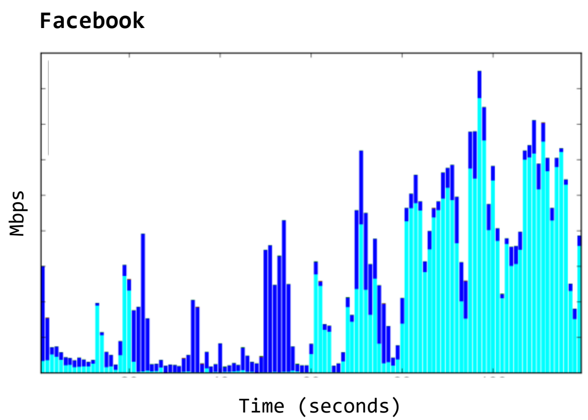
Empirical studies:

traffic matrices sparse and skewed





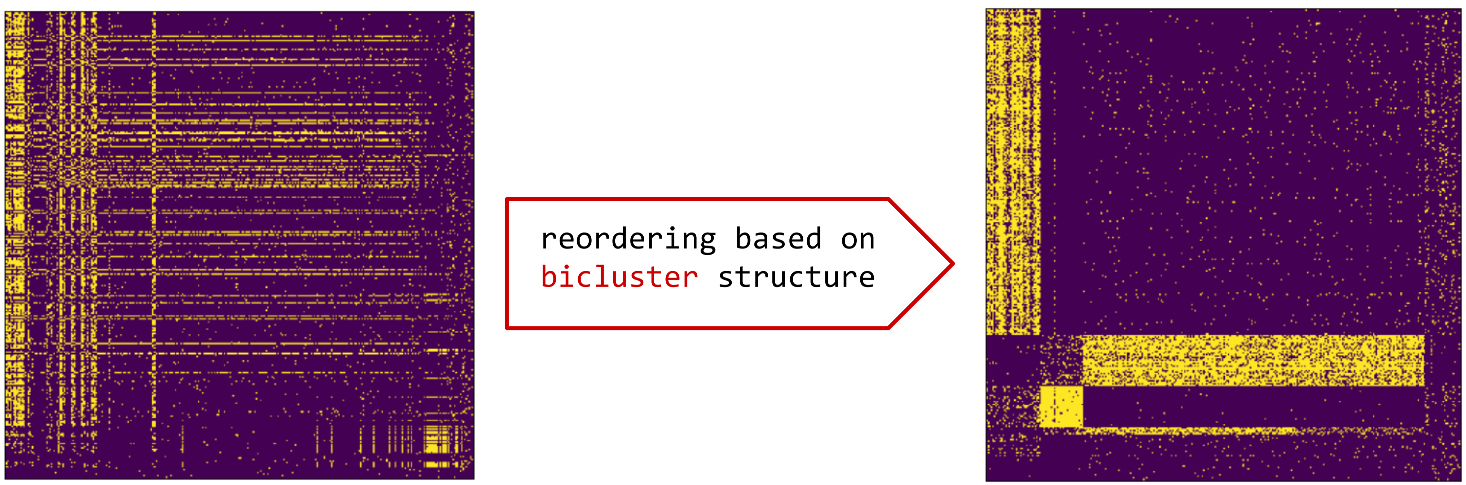
traffic bursty over time



The hypothesis: can be exploited.

Traffic is also clustered:

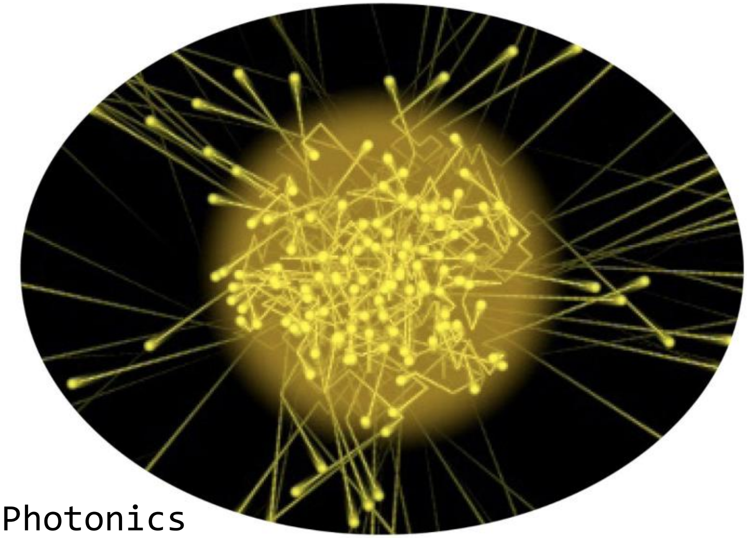
Small Stable Clusters



Opportunity: *exploit* with little reconfigurations!

Literature: Analyzing the Communication Clusters in Datacenters. Foerster et al. WWW Conference, 2023.

Sounds Crazy? Emerging Enabling Technology.



Photonics

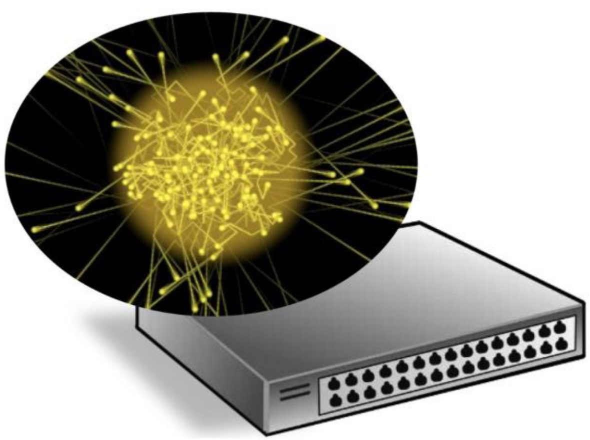
H2020:
“Photonics one of only five key enabling technologies for future prosperity.”

US National Research Council:
“Photons are the new Electrons.”

Enabler

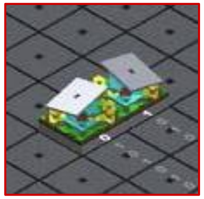
Novel Reconfigurable Optical Switches

- **Spectrum** of prototypes
  - Different sizes, different reconfiguration times
  - From our ACM **SIGCOMM** workshop OptSys



Prototype 1

Moving antenna (ms)



Prototype 2

Moving mirrors (mus)

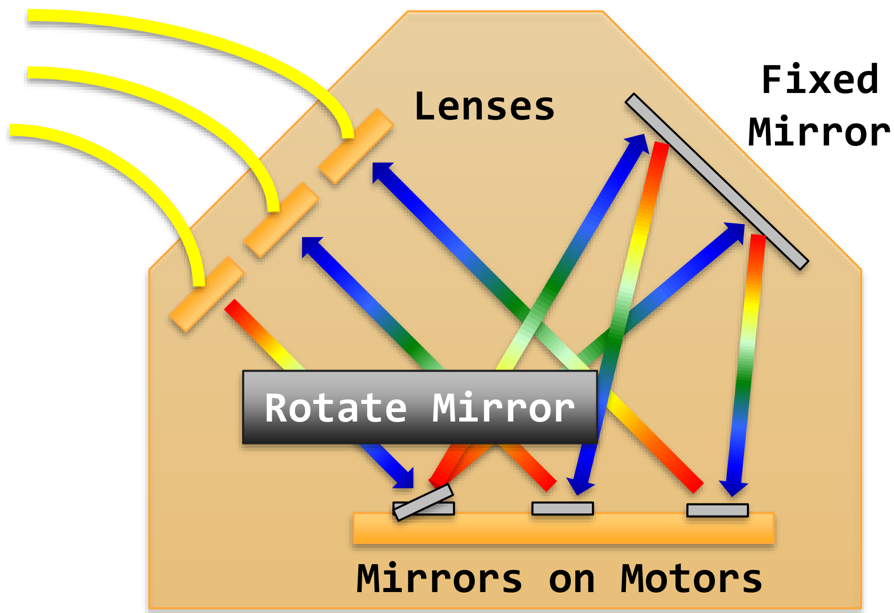
Prototype 3

Changing lambdas (ns)

Example

Optical Circuit Switch

- ⇢ Optical Circuit Switch rapid adaption of physical layer
  - → Based on rotating mirrors

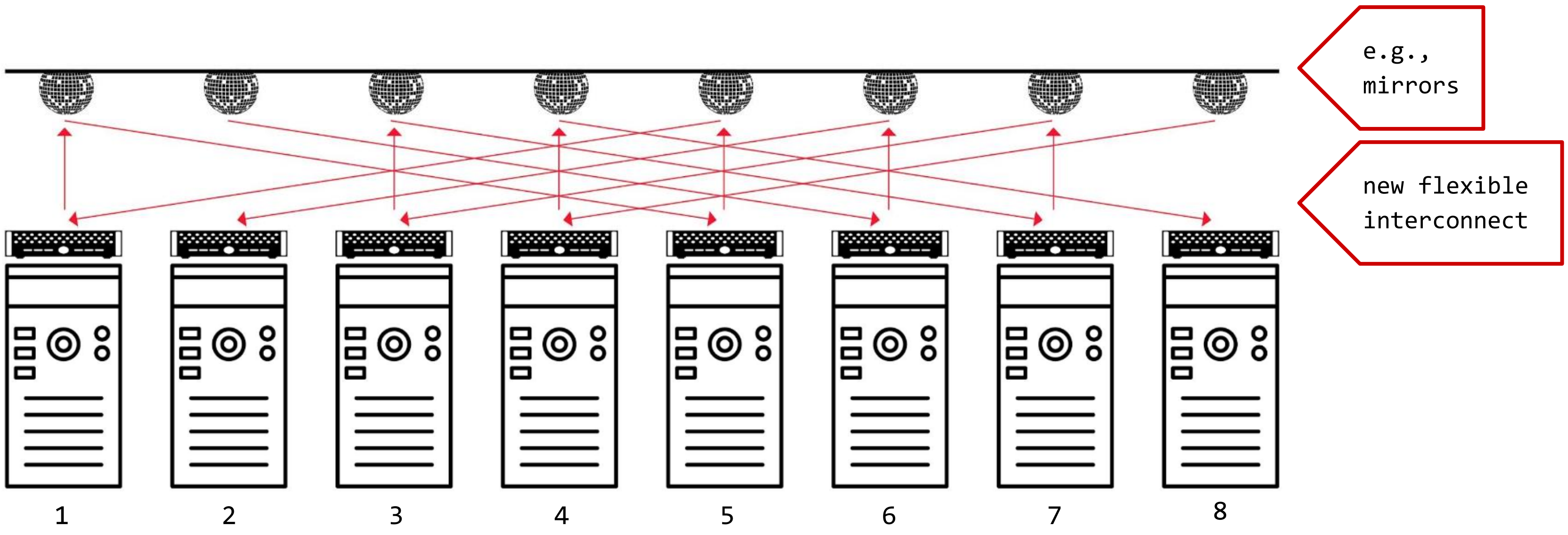


Optical Circuit Switch

By Nathan Farrington, SIGCOMM 2010

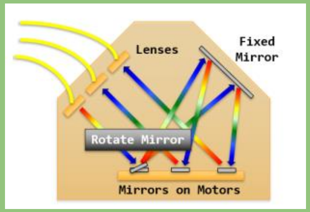
Recall: Our Vision

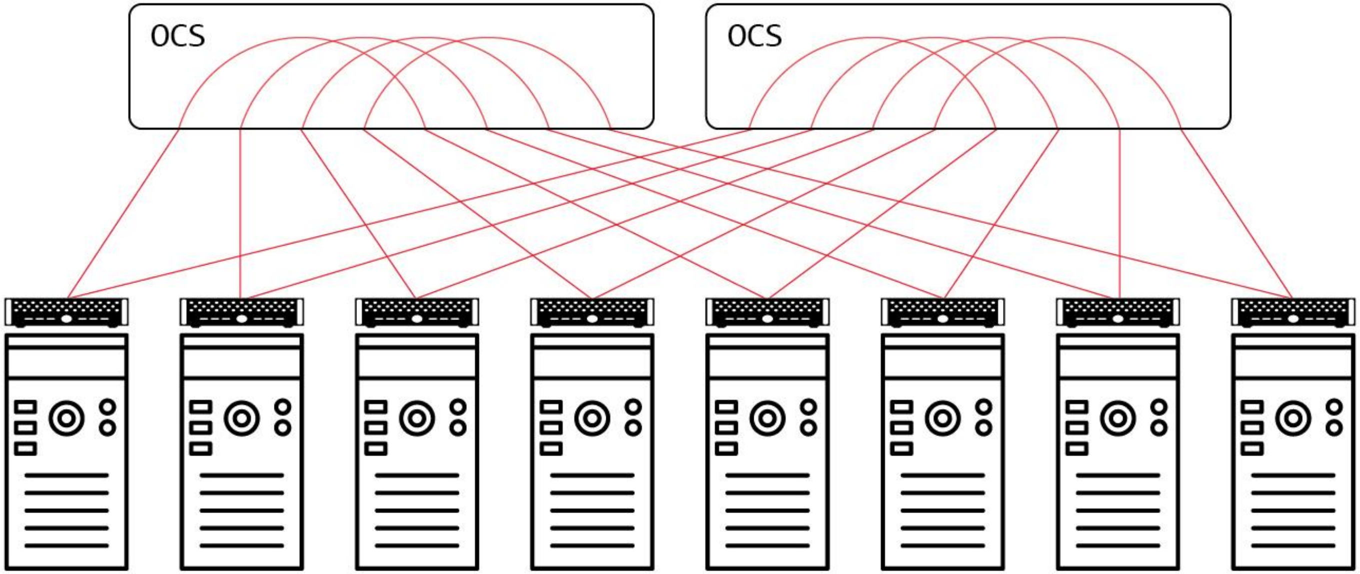
Using Mirrors and Lasers



Realization

with Optical Circuit Switches (OCS)





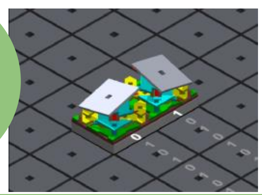
First Deployments

E.g., Google

The Big Picture

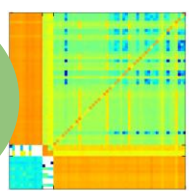
New!

Flexibility



More!

Structure



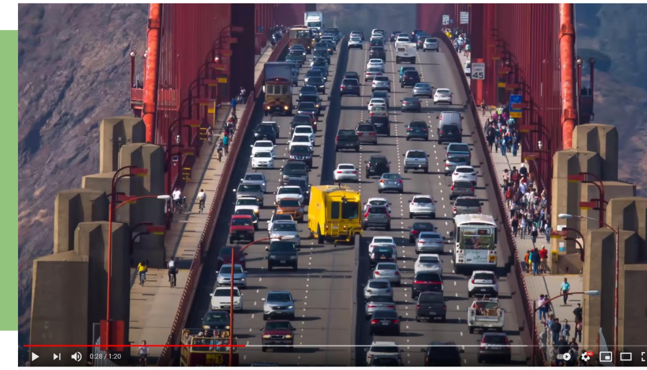
Self-Adjusting Networks

Efficiency

Now is the time!

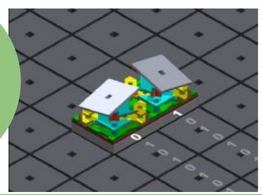
The Big Picture

Like "Golden Gate Zipper" for datacenters.



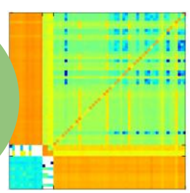
New!

Flexibility



More!

Structure



Self-Adjusting Networks

Efficiency

Now is the time!

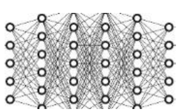
Unique Position

Demand-Aware, Self-Adjusting Systems

Everywhere, but mainly in software

Algorithmic trading

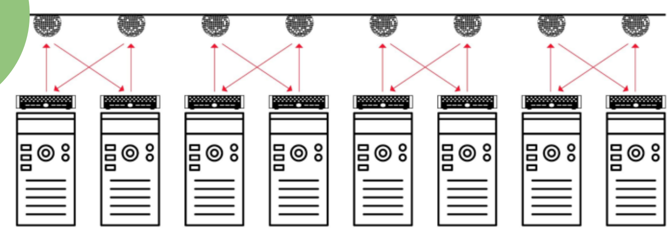
Recommender systems



Neural networks

VS

Our focus in this talk: in hardware



First basic question:

How to measure and model structure in workloads?

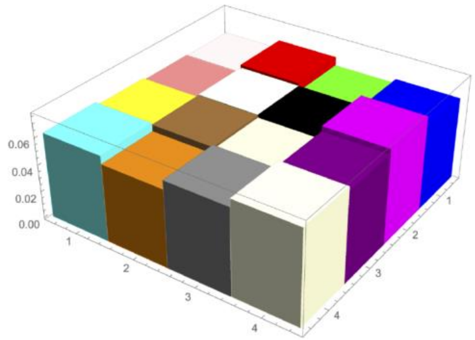
A first insight: related to entropy.

Intuition

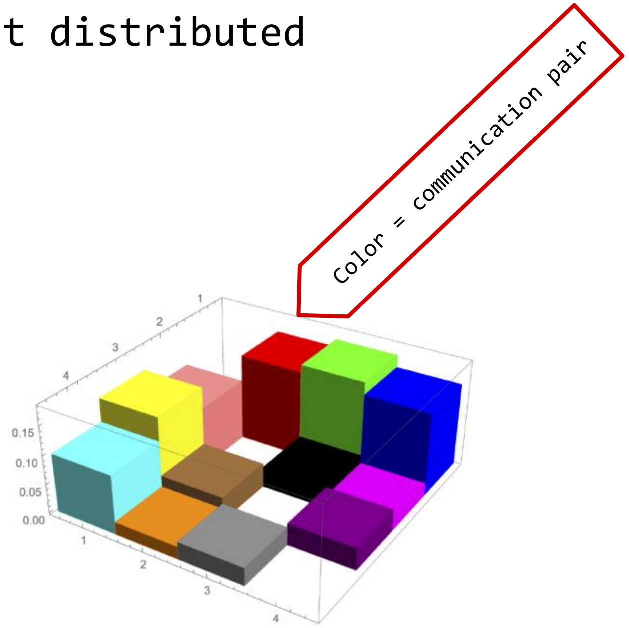
Which demand has more structure?

⇢ Traffic matrices of two different distributed ML applications

- ⇀ GPU-to-GPU



VS

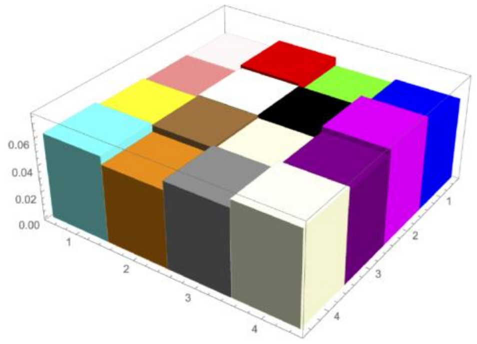


Intuition

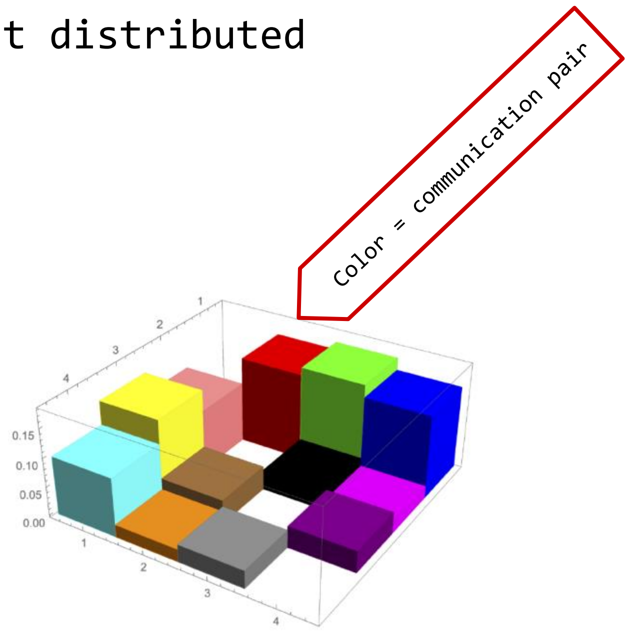
Which demand has more structure?

⇢ Traffic matrices of two different distributed ML applications

⇀ GPU-to-GPU



VS



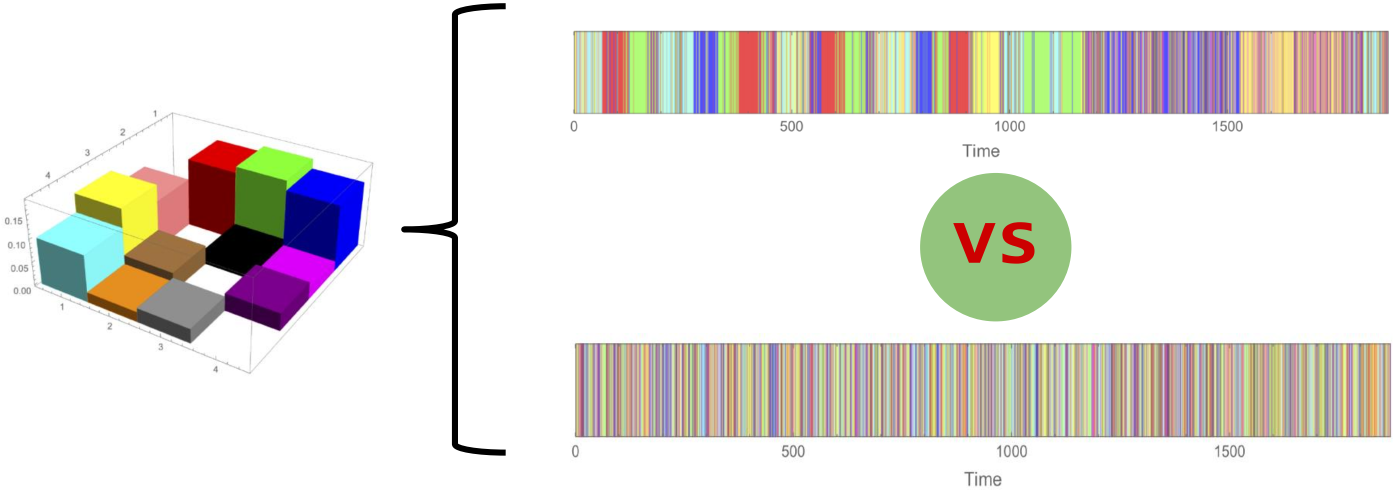
More uniform **More structure**

Intuition

Spatial vs temporal structure

⇢ Two different ways to generate same traffic matrix:
  ⇀ Same non-temporal structure

⇢ Which one has more structure?

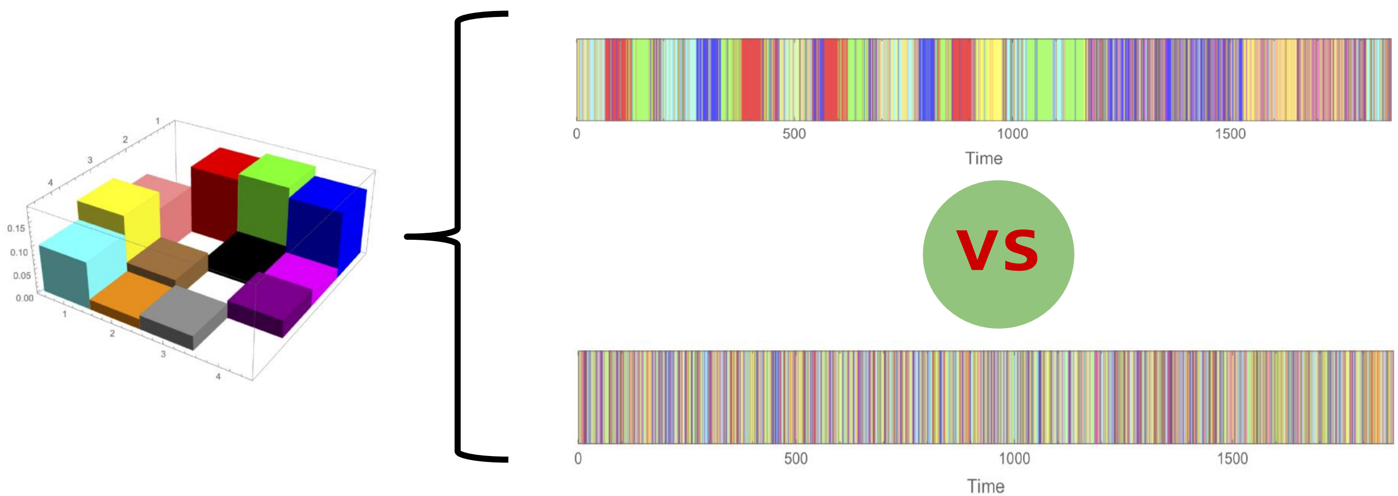


Intuition

Spatial vs temporal structure

- Two different ways to generate same traffic matrix:
  - Same non-temporal structure

- Which one has more structure?

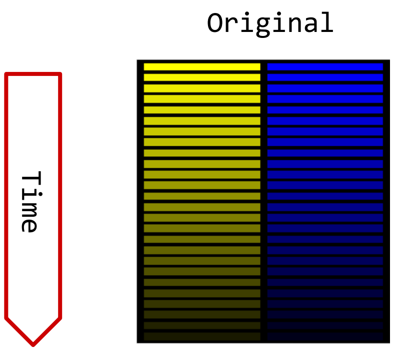


Systematically?

Trace Complexity

Information-Theoretic Approach

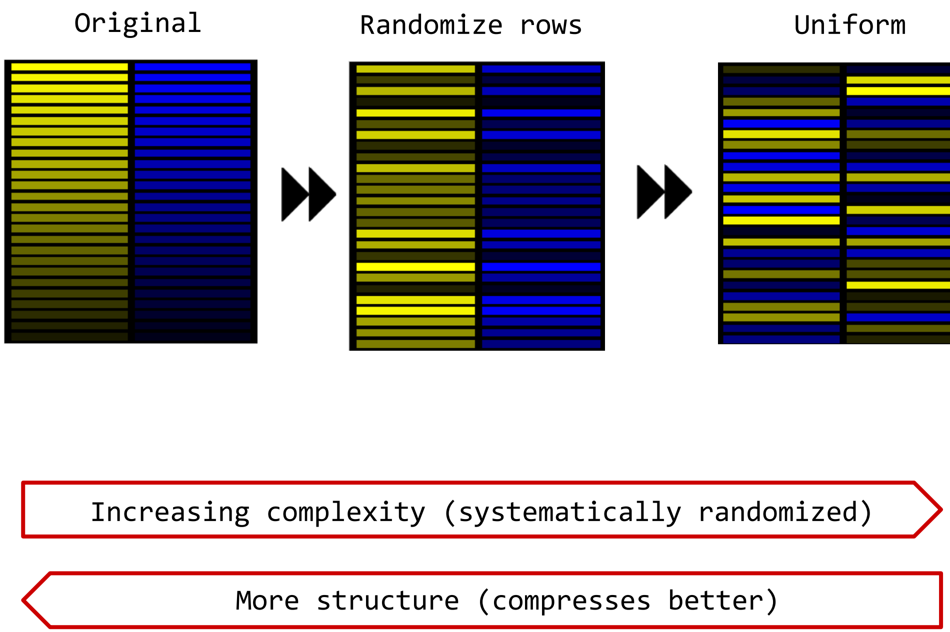
“Shuffle&Compress”



Trace Complexity

Information-Theoretic Approach

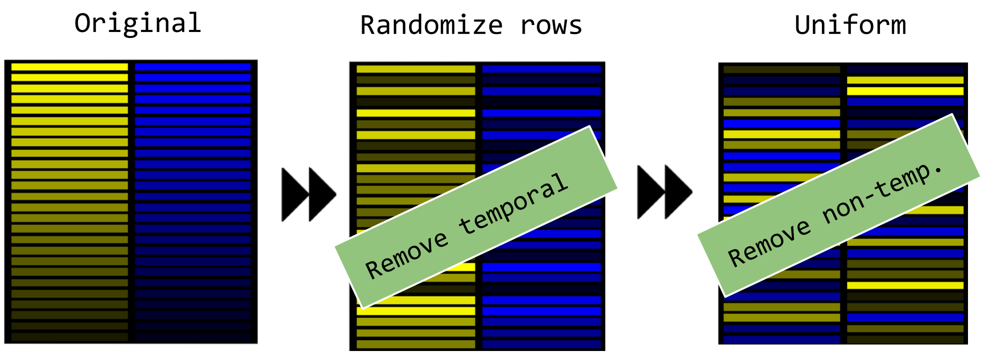
“Shuffle&Compress”



Trace Complexity

Information-Theoretic Approach

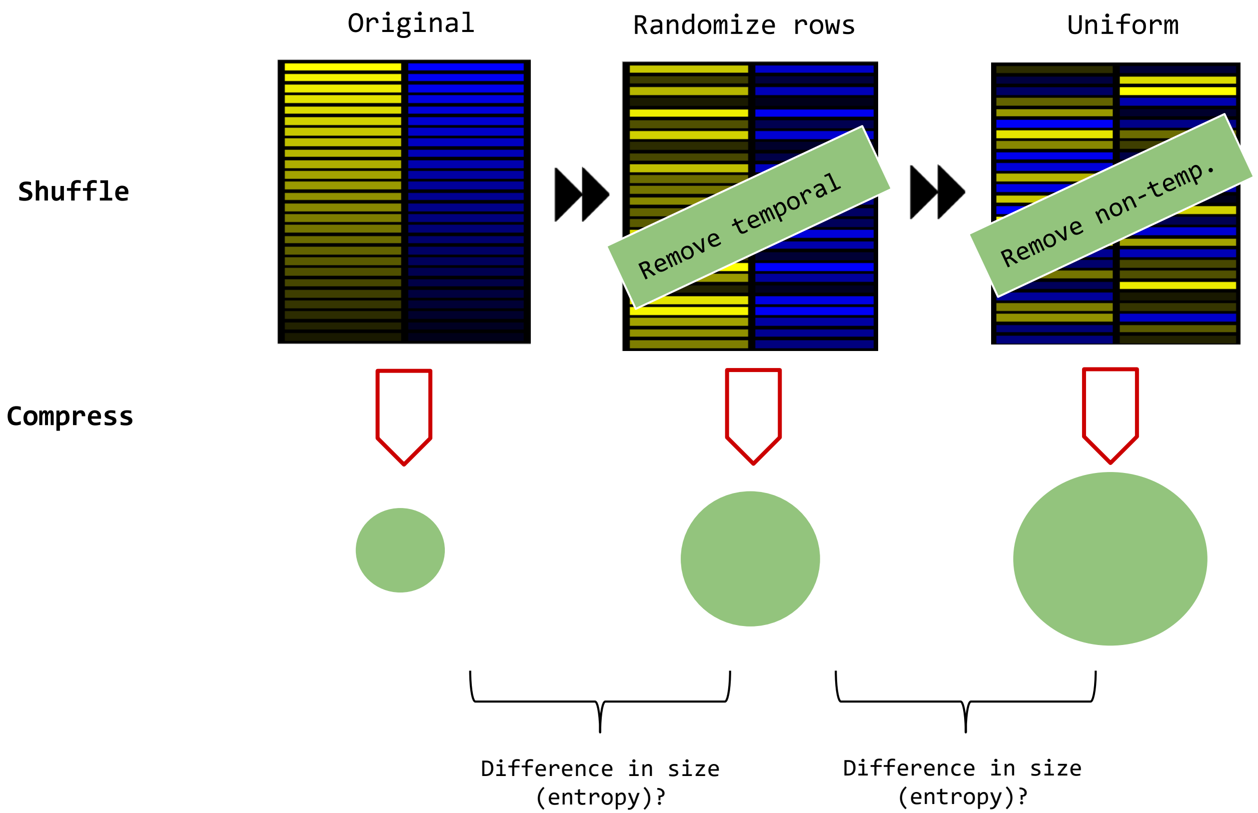
“Shuffle&Compress”



Trace Complexity

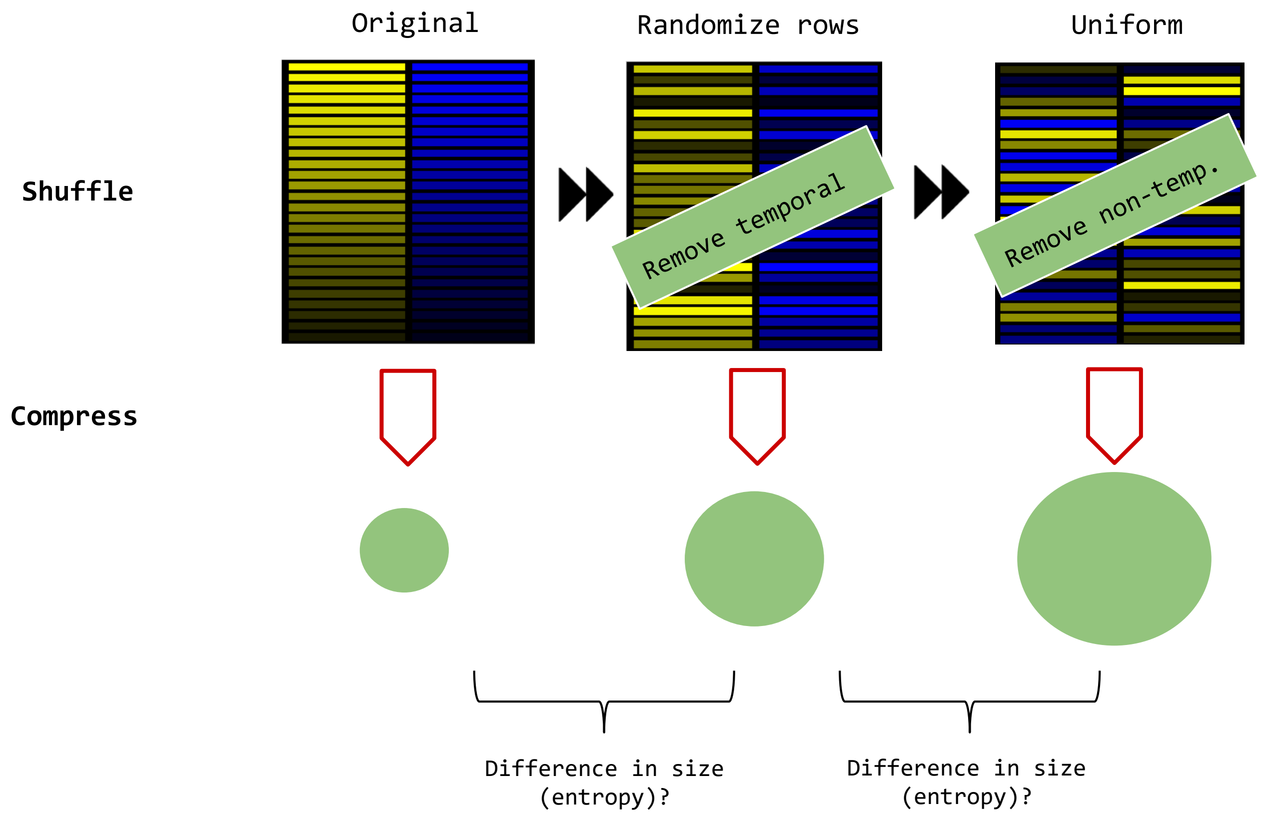
Information-Theoretic Approach

"Shuffle&Compress"



Trace Complexity

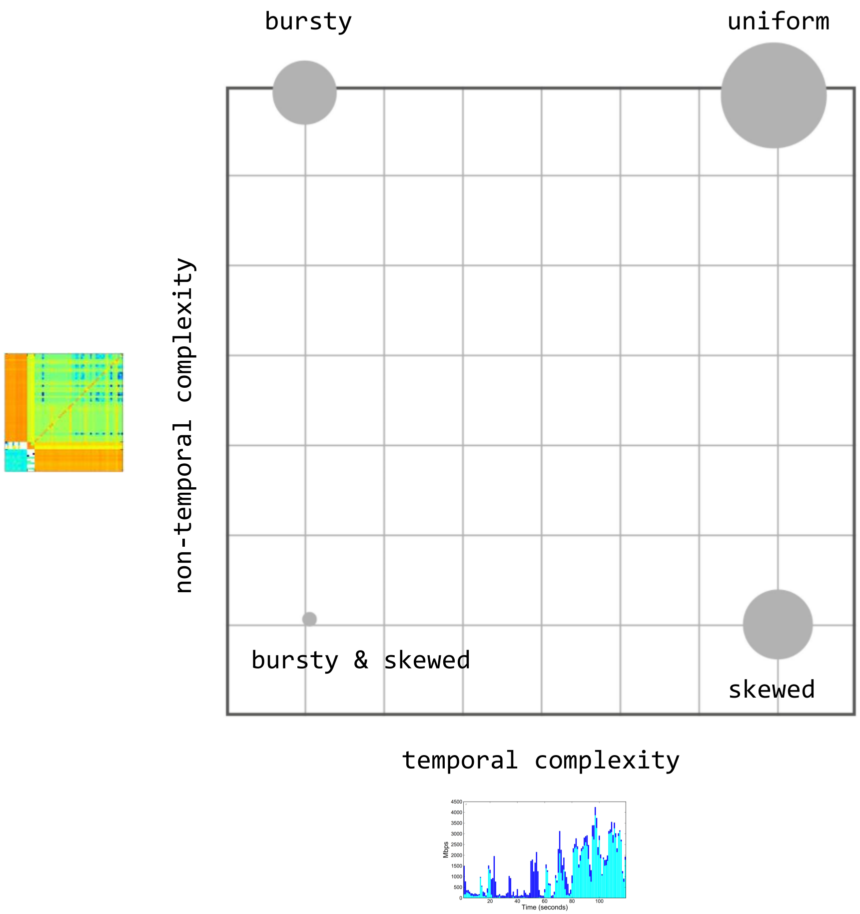
Information-Theoretic Approach
“Shuffle&Compress”



Can be used to define 2-dimensional complexity map!

Avin et al. (Sigmetrics'2020)

Complexity Map

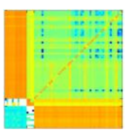


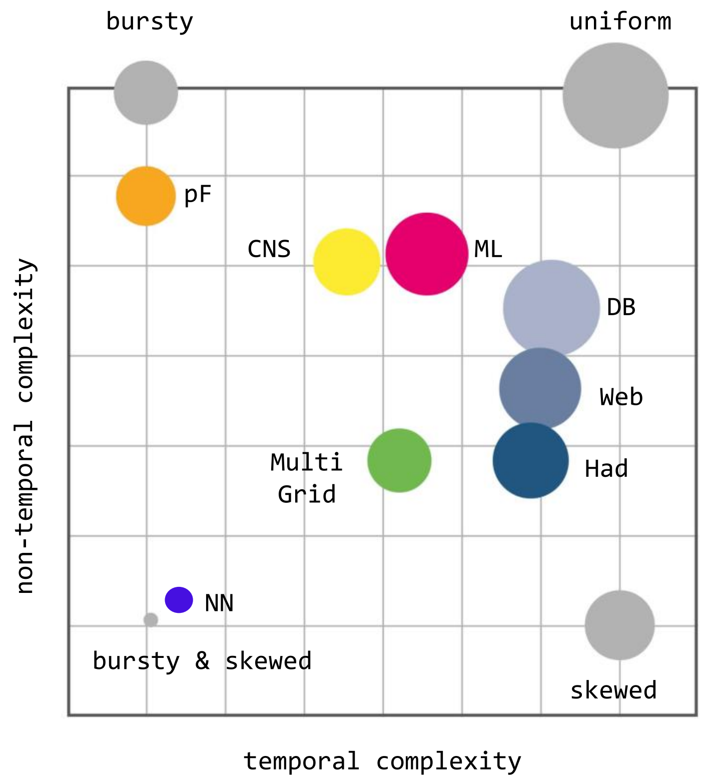
No structure

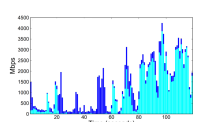
Our **approach:** iterative **randomization and compression** of trace to identify dimensions of structure.

Avin et al. (Sigmetrics'2020)

Complexity Map





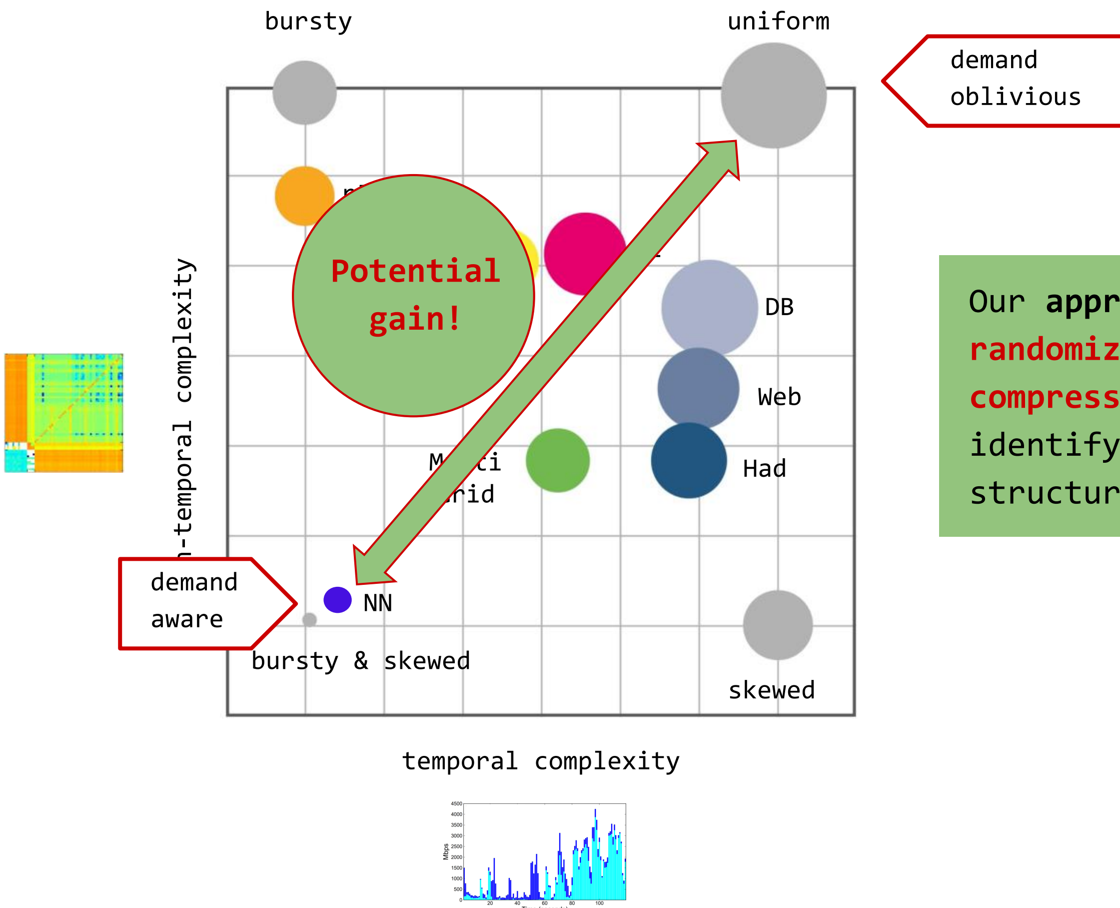


Our **approach**: iterative **randomization and compression** of trace to identify dimensions of structure.

Different structures!

Avin et al. (Sigmetrics’2020)

Complexity Map

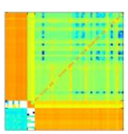


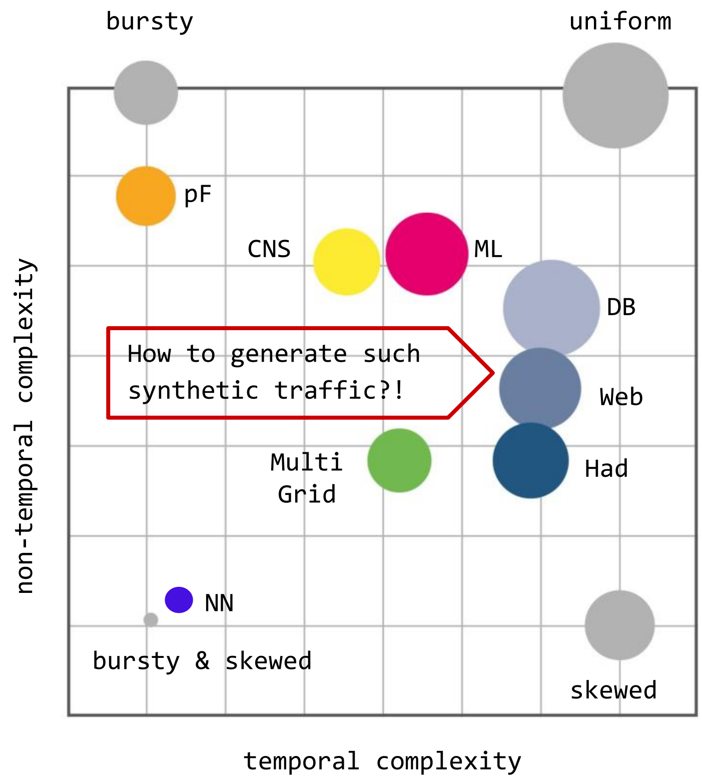
Our **approach**: iterative **randomization and compression** of trace to identify dimensions of structure.

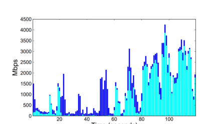
Literature: On the Complexity of Traffic Traces and Implications. Avin et al., ACM SIGMETRICS, 2020.

Avin et al. (Sigmetrics’2020)

Complexity Map





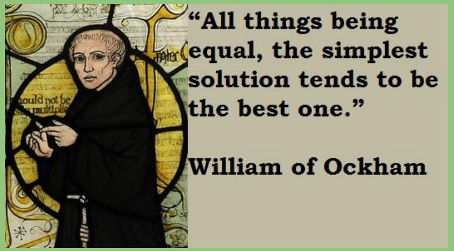


Our **approach**: iterative **randomization and compression** of trace to identify dimensions of structure.

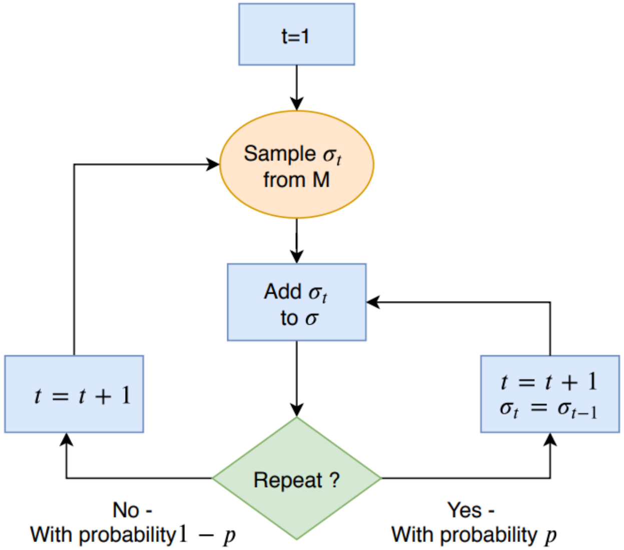
Literature: On the Complexity of Traffic Traces and Implications. Avin et al., ACM SIGMETRICS, 2020.

From Analysis to

Synthesis

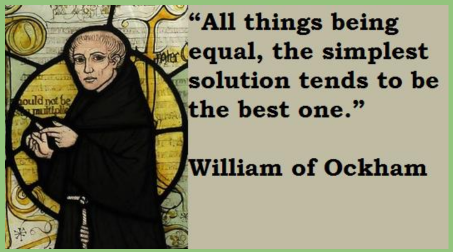


- Complexity map is just 2-dimensional: many ways to synthesize any point on map
- Most simple ("Occam's razor"):
  - ***Spatial distribution:*** empirical traffic matrix M (or synthetic distribution, e.g. Zipf)
  - ***Temporal distribution:*** repeat with probability p (can be computed analytically from data)
- Resulting ***Markov process*** generates corresponding disk on complexity map
  - ***Stationary distribution*** corresponds to M
  - Temporary pattern matches ***entropy rate***

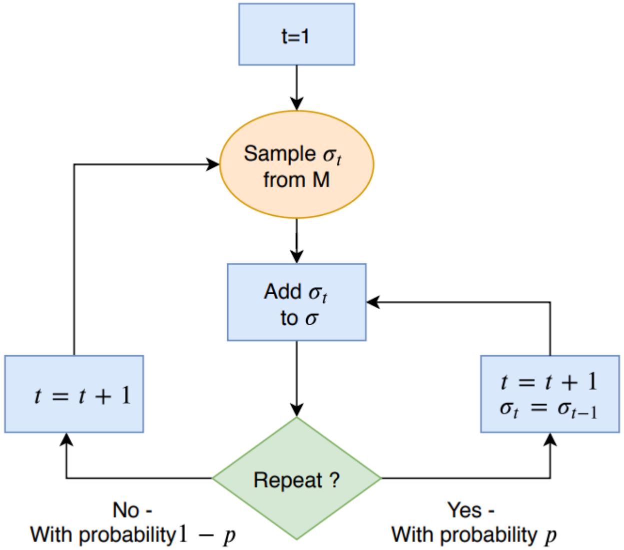


From Analysis to

Synthesis



- Complexity map is just 2-dimensional: many ways to synthesize any point on map
- Most simple ("Occam's razor"):
  - ***Spatial distribution:*** empirical traffic matrix M (or synthetic distribution, e.g. Zipf)
  - ***Temporal distribution:*** repeat with probability p (can be computed analytically from data)
- Resulting ***Markov process*** generates corresponding disk on complexity map
  - ***Stationary distribution*** corresponds to M
  - Temporary pattern matches ***entropy rate***



Literature: On the Complexity of Traffic Traces and Implications. Avin et al., ACM SIGMETRICS, 2020.

Further Reading

On the Complexity of Traffic Traces and Implications
Chen Avin, Manya Ghobadi, Chen Griner, and Stefan Schmid.
ACM **SIGMETRICS** and ACM Performance Evaluation Review (**PER**), Boston, Massachusetts, USA, June 2020.

Analyzing the Communication Clusters in Datacenters
Klaus-Tycho Foerster, Thibault Marette, Stefan Neumann, Claudia Plant, Ylli Sadikaj, Stefan Schmid, and Yllka Velaj.
The Web Conference (**WWW**), Austin, Texas, USA, April 2023.

Network Traffic Characteristics of Machine Learning Frameworks Under the Microscope
Johannes Zerwas, Kaan Aykurt, Stefan Schmid, and Andreas Blenk. 17th International Conference on Network and Service Management (**CNSM**), Izmir, Turkey, October 2021.

Website: trace-collection.net

The Natural Question:

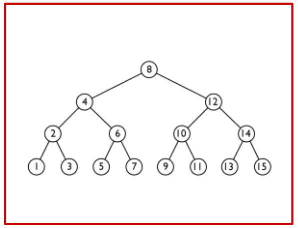
Given This Structure, What Can Be Achieved? Metrics and Algorithms?

Also depends on entropy of the demand!

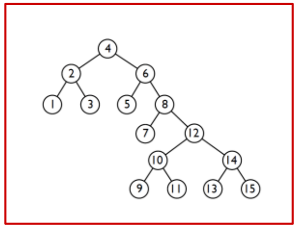
Insight:

Connection to Datastructures

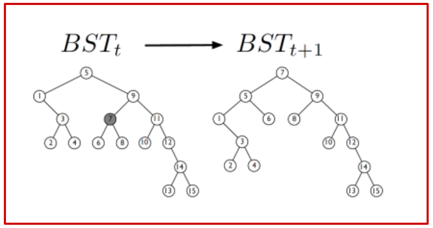
Traditional BST



Demand-aware BST



Self-adjusting BST

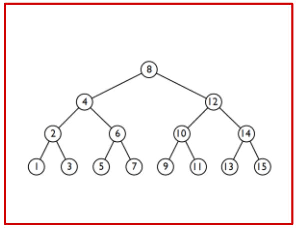


More structure: improved **access cost**

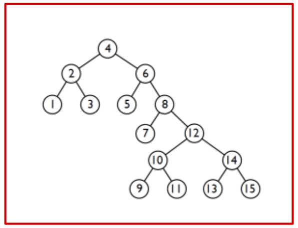
Insight:

Connection to Datastructures & Coding

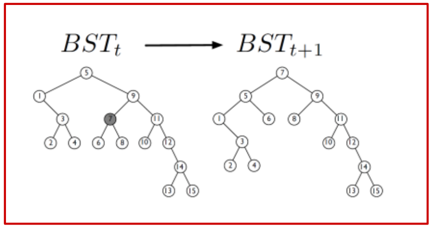
Traditional BST
(Worst-case coding)



Demand-aware BST
(Huffman coding)



Self-adjusting BST
(Dynamic Huffman coding)



More structure: improved **access cost** / shorter **codes**

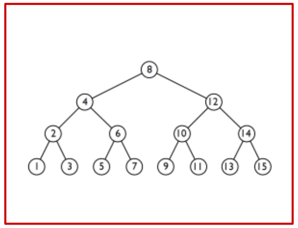
Insight:

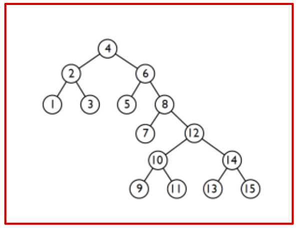
Connection to Datastructures & Coding

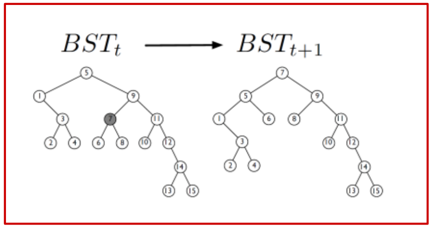
Traditional BST
(Worst-case coding)

Demand-aware BST
(Huffman coding)

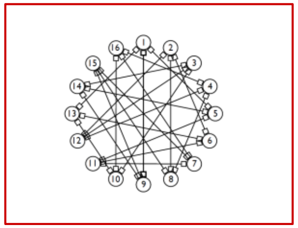
Self-adjusting BST
(Dynamic Huffman coding)

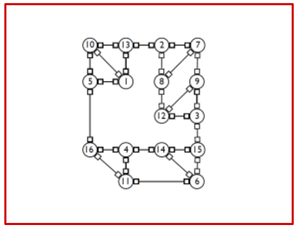


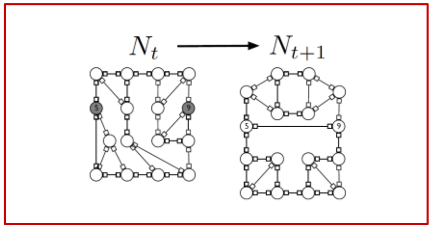




More structure: improved **access cost** / shorter **codes**





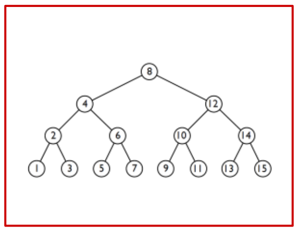


Similar **benefits**?

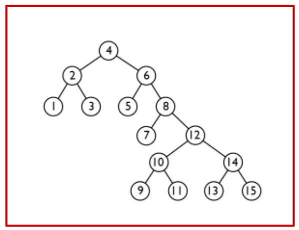
Insight:

Connection to Datastructures & Coding

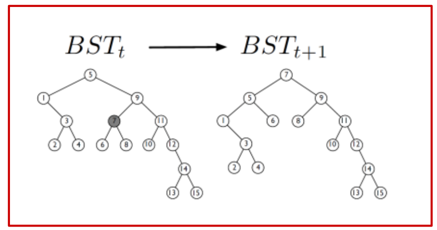
Traditional BST
(Worst-case coding)



Demand-aware BST
(Huffman coding)

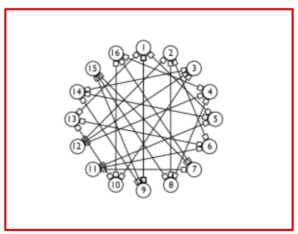


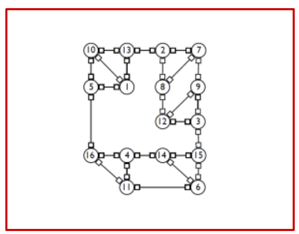
Self-adjusting BST
(Dynamic Huffman coding)

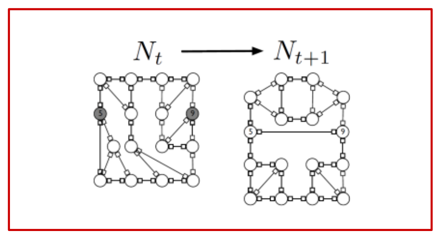


More than an analogy!

More structure: improved **access cost** / shorter **codes**







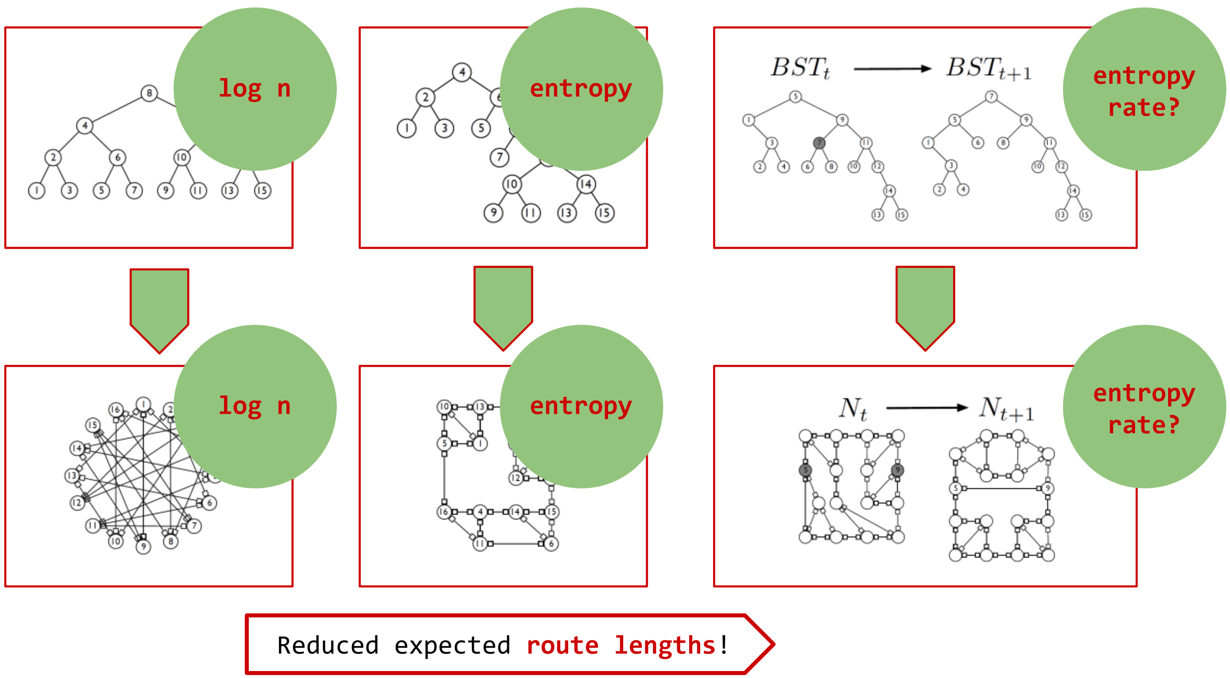
Similar **benefits**?

Insight:

Connection to Datastructures & Coding

More than an analogy!

Traditional BST (Worst-case coding) | Demand-aware BST (Huffman coding) | Self-adjusting BST (Dynamic Huffman coding)

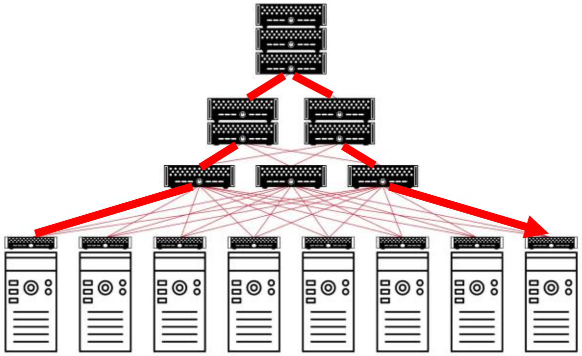


Generalize methodology:
... and transfer entropy bounds and algorithms of data-structures to networks.

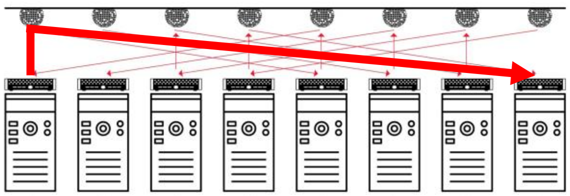
First result:
Demand-aware networks of asymptotically optimal route lengths.

Reality more complicated

⇀ Self-adjusting networks may be really useful to serve large flows (elephant flows): avoiding multi-hop routing



vs

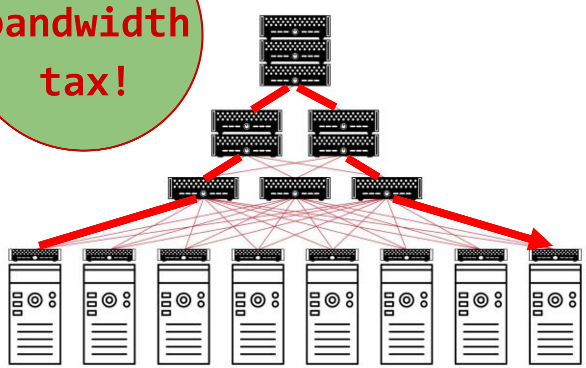


6 hops **1 hop**

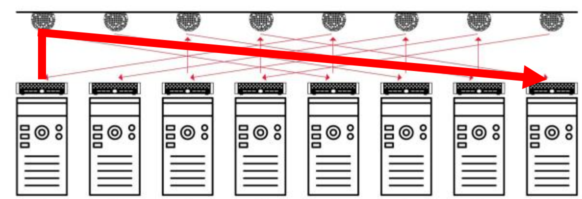
Reality more complicated

→ Self-adjusting networks may be really useful to serve large flows (elephant flows): avoiding multi-hop routing

bandwidth tax!



vs



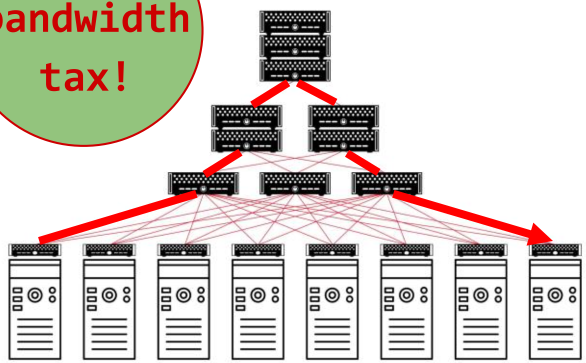
6 hops

1 hop

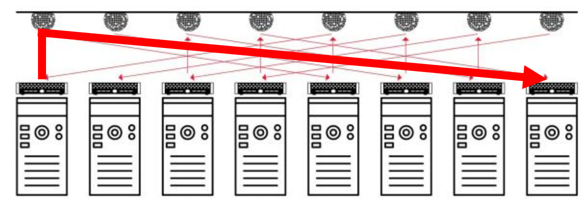
Reality more complicated

→ Self-adjusting networks may be really useful to serve large flows (elephant flows): avoiding multi-hop routing

bandwidth tax!



vs



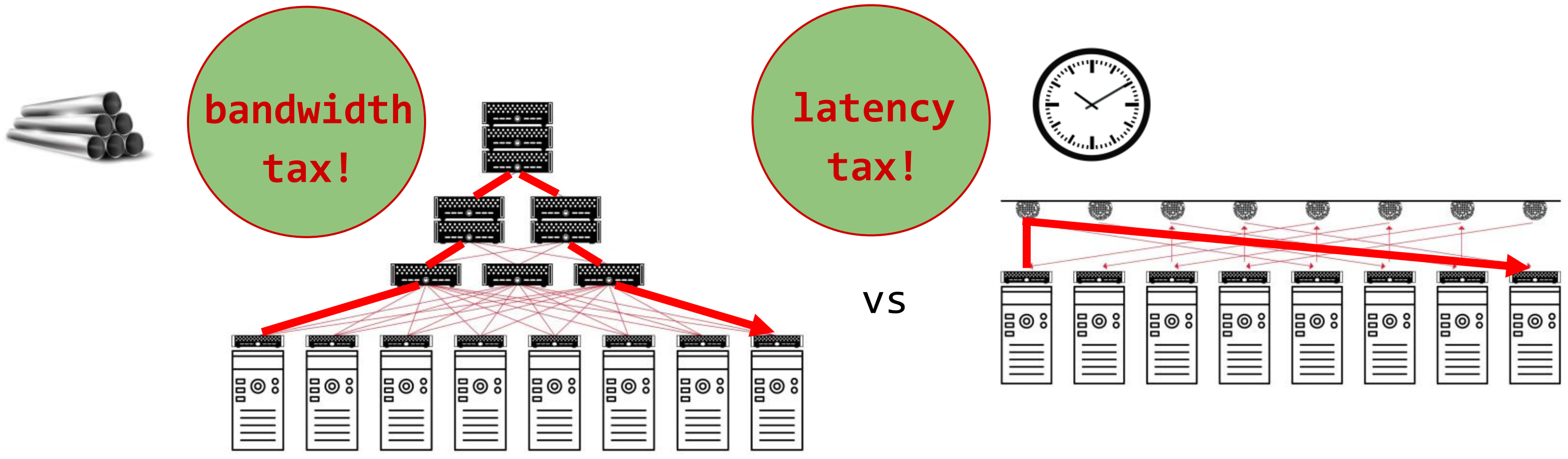
6 hops

1 hop

→ However, requires optimization and adaption, which takes time

Reality more complicated

→ Self-adjusting networks may be really useful to serve large flows (elephant flows): avoiding multi-hop routing



6 hops **1 hop**

→ However, requires optimization and adaption, which takes time

Indeed, it is more complicated than that…

Challenge: Traffic Diversity

Diverse patterns:

⇀ Shuffling/Hadoop:
all-to-all

⇀ All-reduce/ML: ring or
tree traffic patterns

⇀ Elephant flows

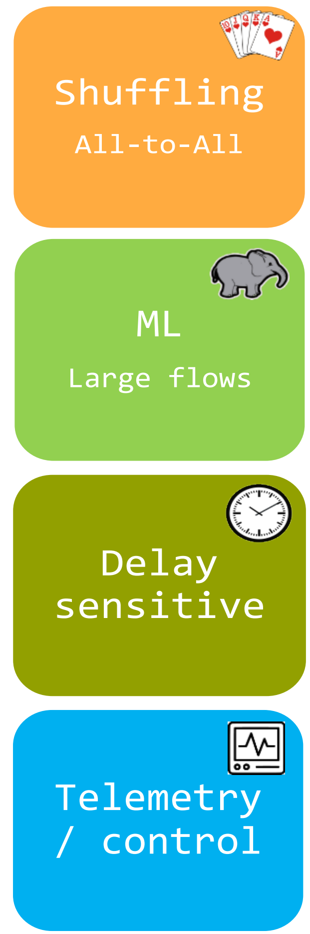
⇀ Query traffic: skewed

⇀ Mice flows

⇀ Control traffic: does not evolve
but has non-temporal structure

Diverse requirements:

⇀ ML is bandwidth hungry,
small flows are latency-
sensitive



Opportunity: Tech Diversity

Diverse topology components:

→ demand-oblivious and

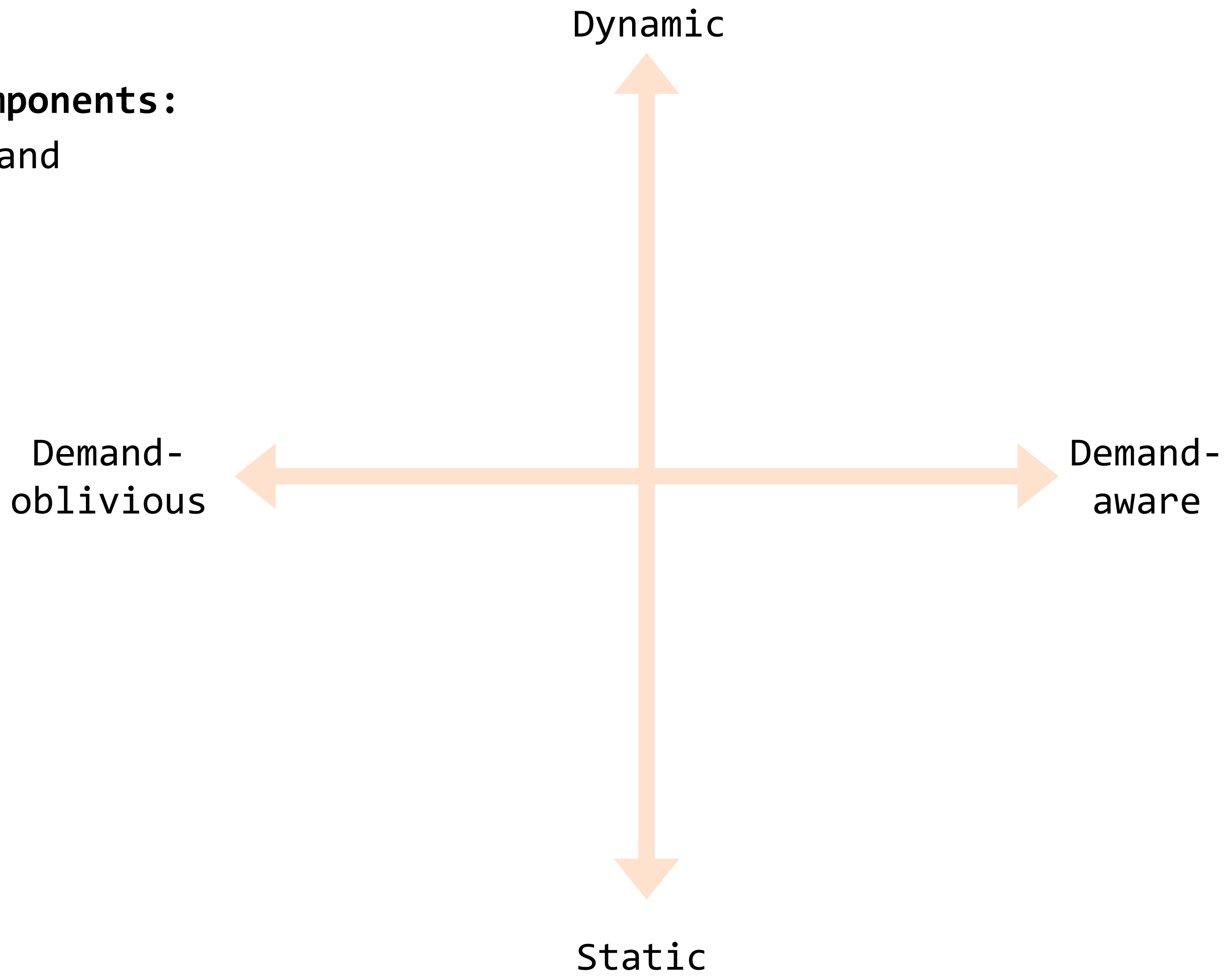
demand-aware

Opportunity: Tech Diversity

Diverse topology components:

⇀ demand-oblivious and
demand-aware

⇀ static vs dynamic

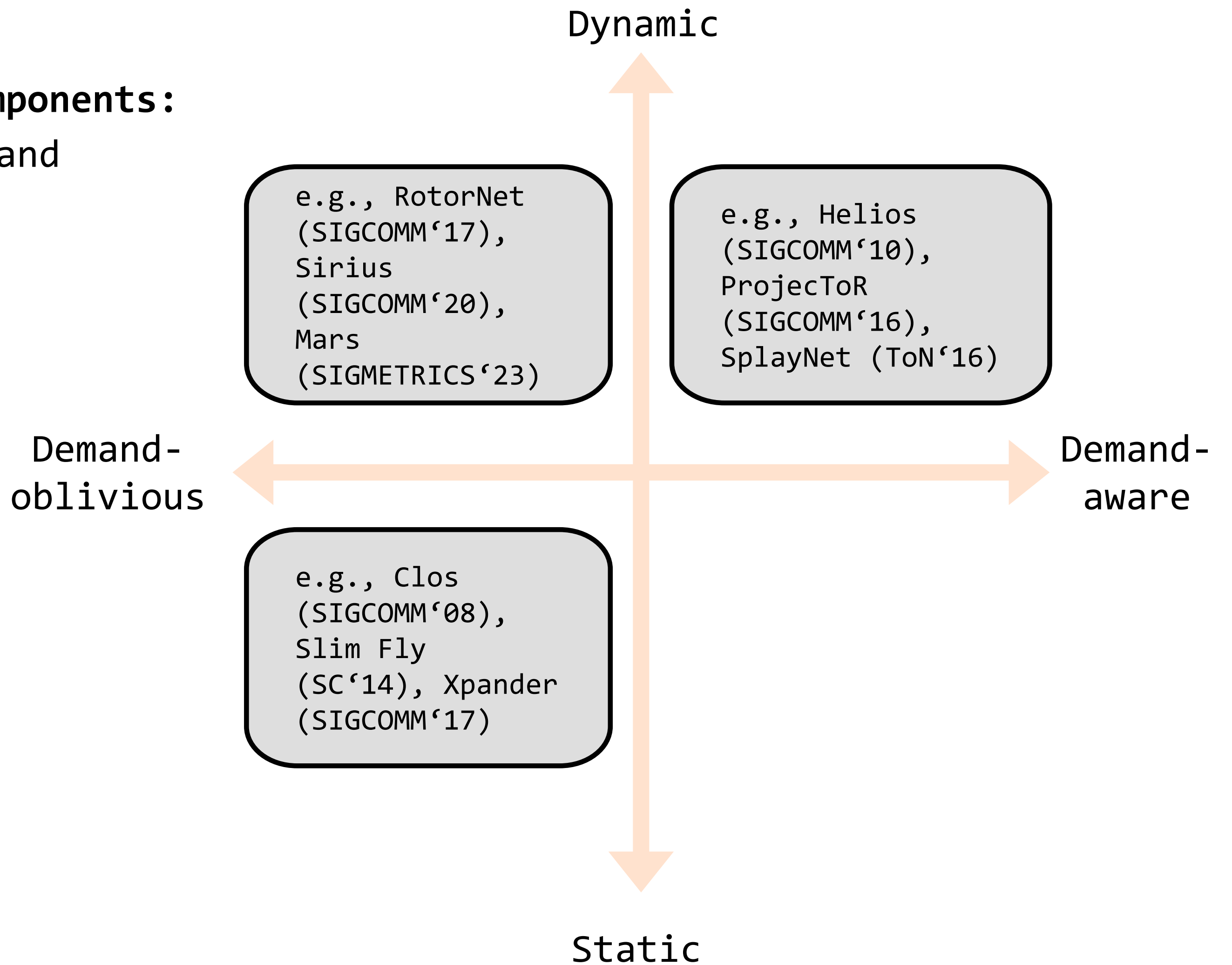


Opportunity: Tech Diversity

Diverse topology components:

→ demand-oblivious and demand-aware

→ static vs dynamic

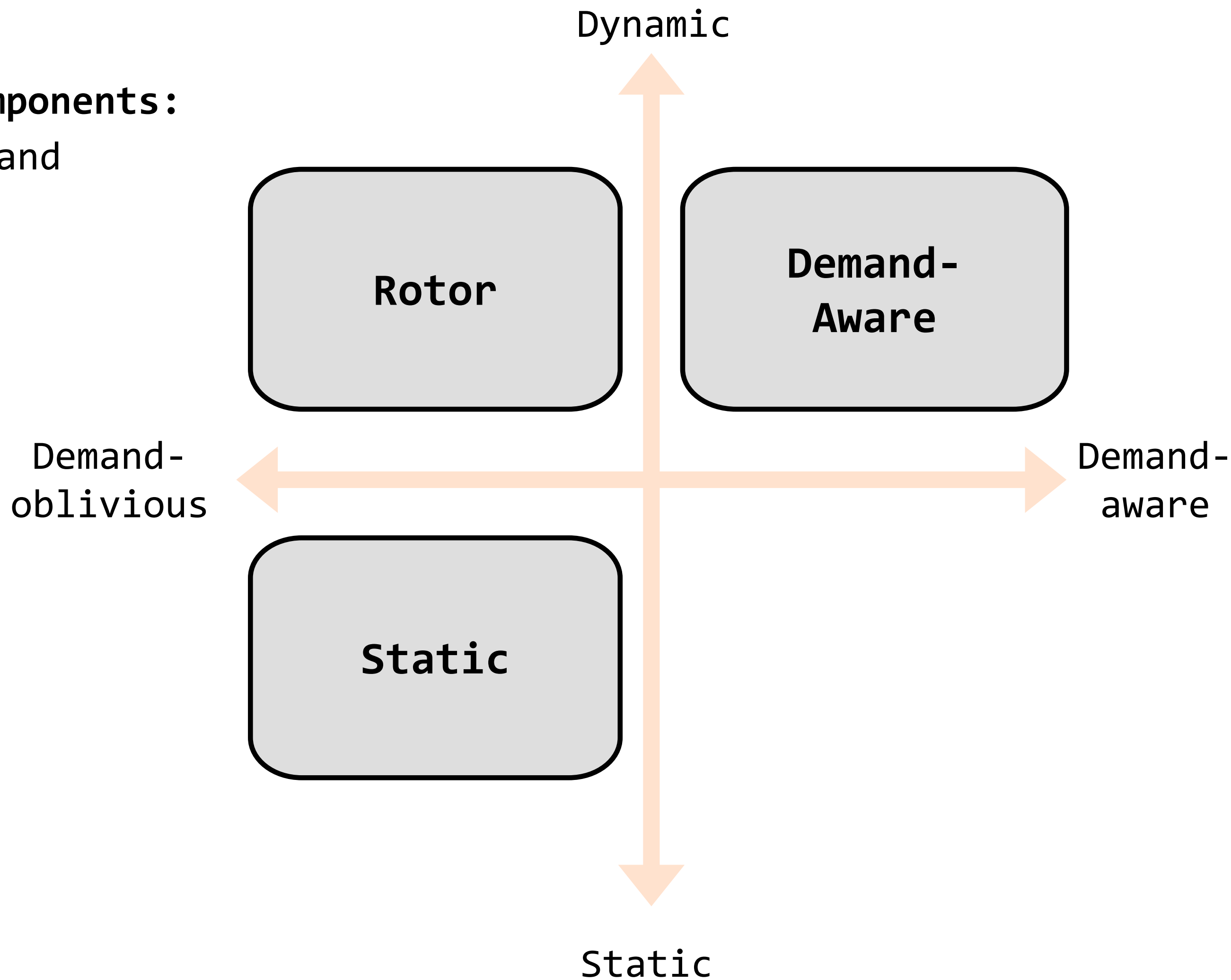


Opportunity: Tech Diversity

Diverse topology components:

⇀ demand-oblivious and demand-aware

⇀ static vs dynamic

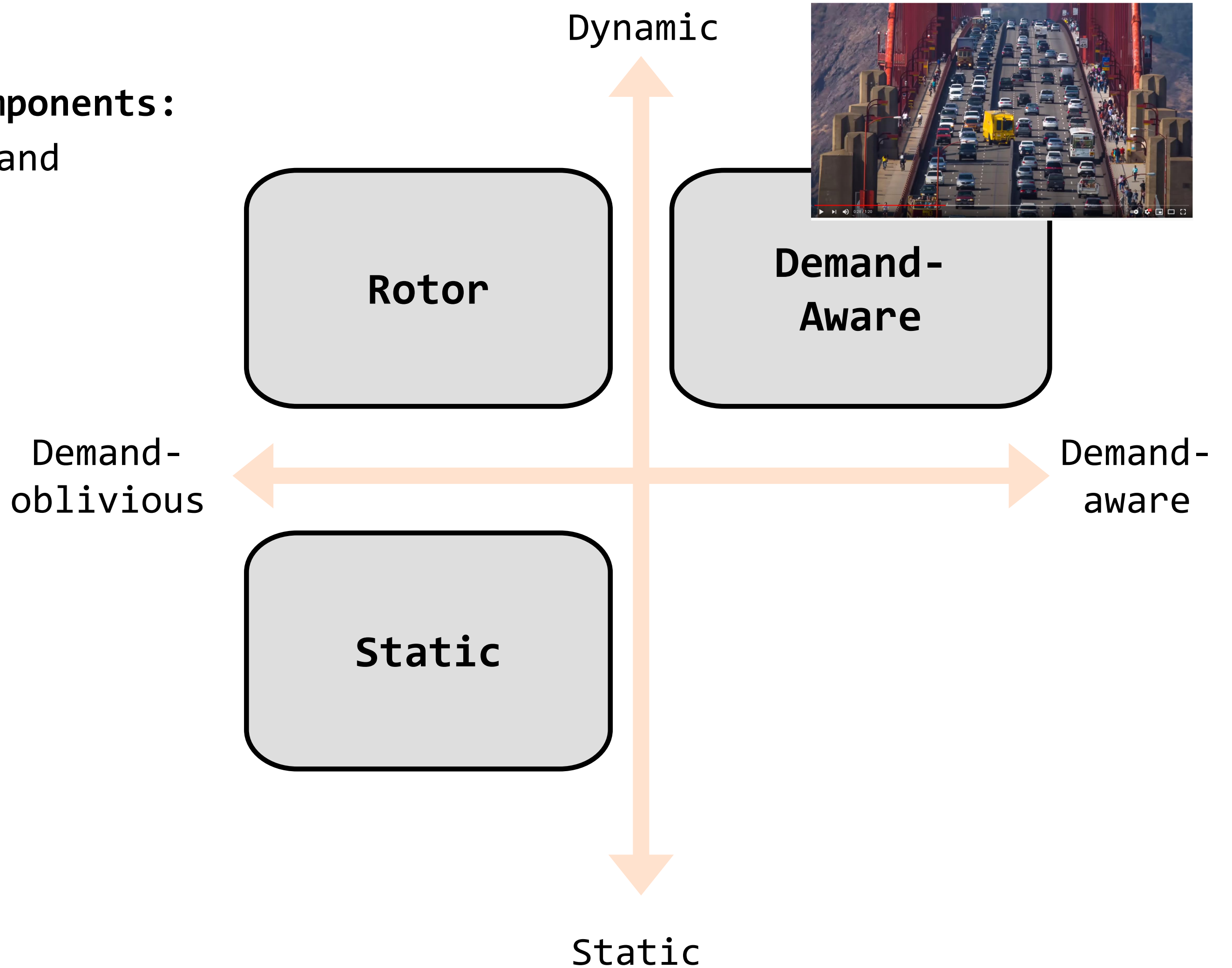


Opportunity: Tech Diversity

Diverse topology components:

→ demand-oblivious and demand-aware

→ static vs dynamic

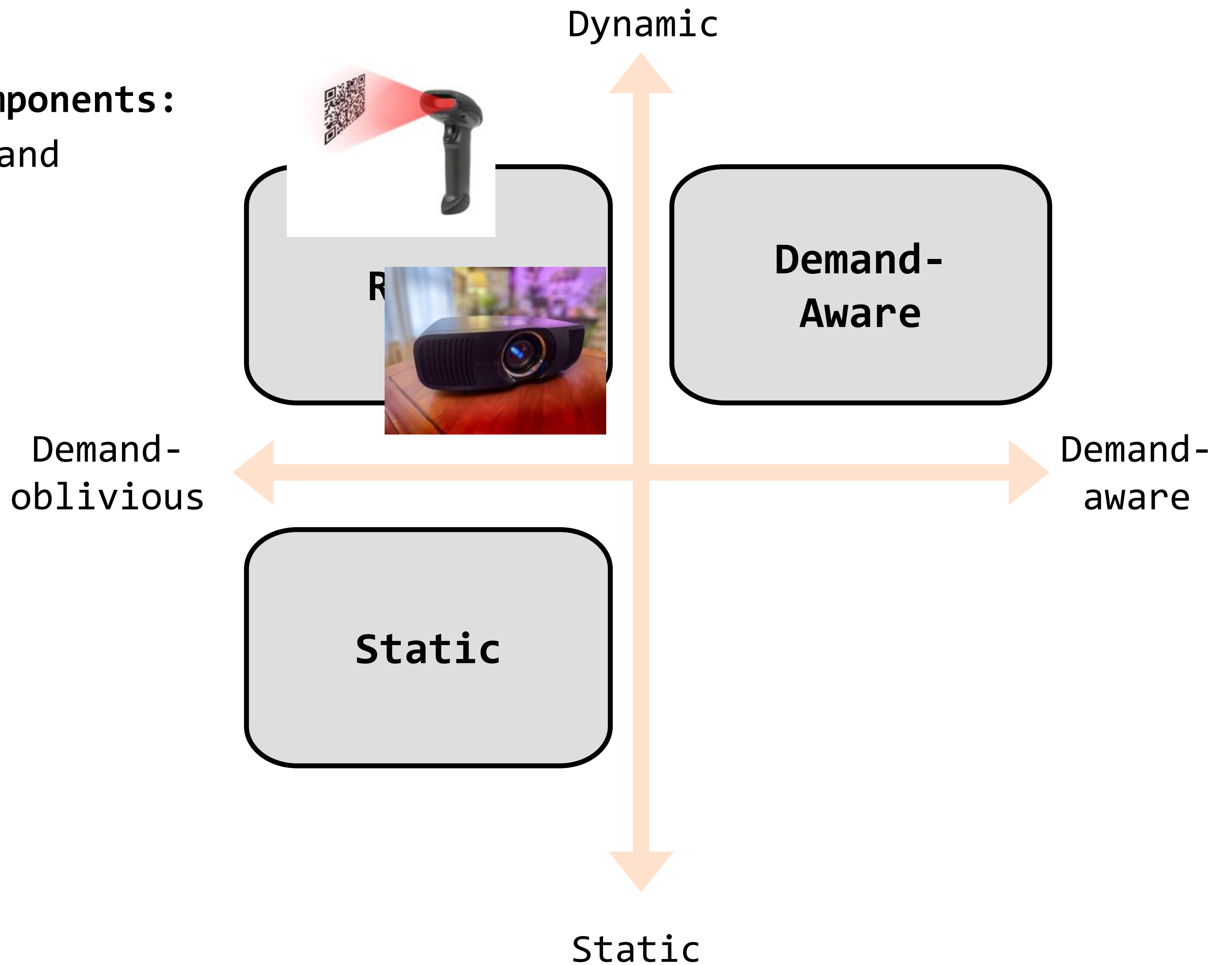


Opportunity: Tech Diversity

Diverse topology components:

⇀ demand-oblivious and demand-aware

⇀ static vs dynamic



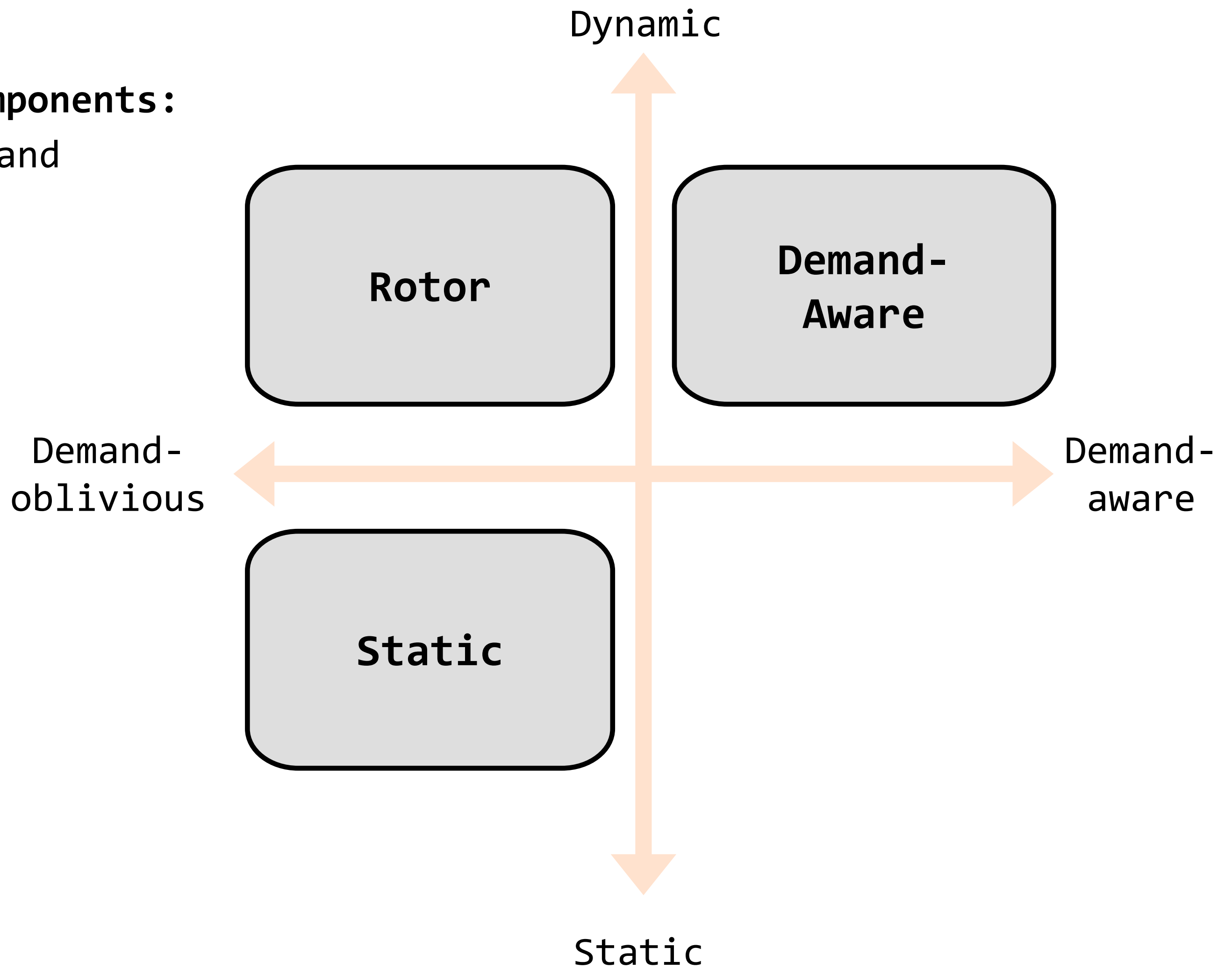
Opportunity: Tech Diversity

Diverse topology components:

→ demand-oblivious and demand-aware

→ static vs dynamic

Which approach is best?

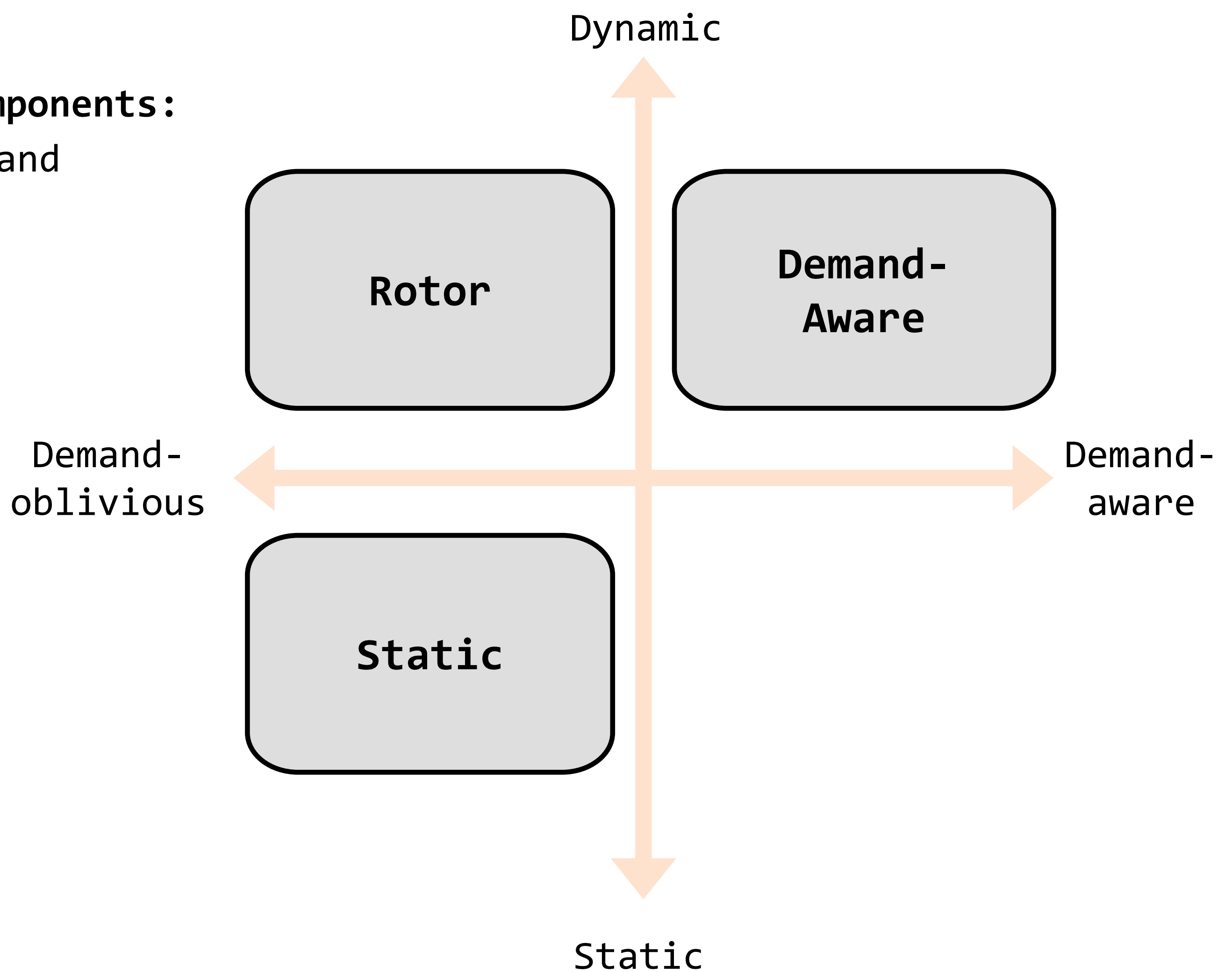


Opportunity: Tech Diversity

Diverse topology components:

⇀ demand-oblivious and
demand-aware

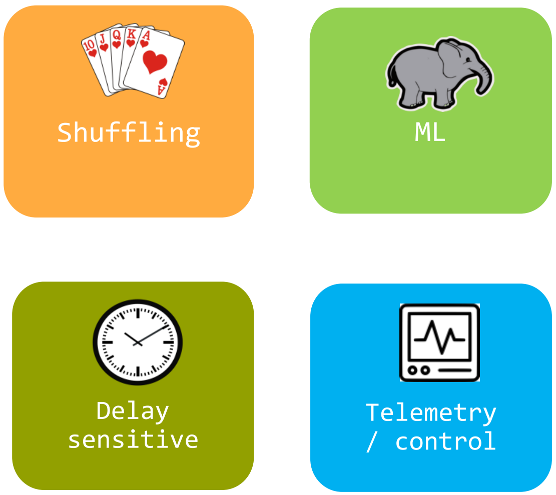
⇀ static vs dynamic



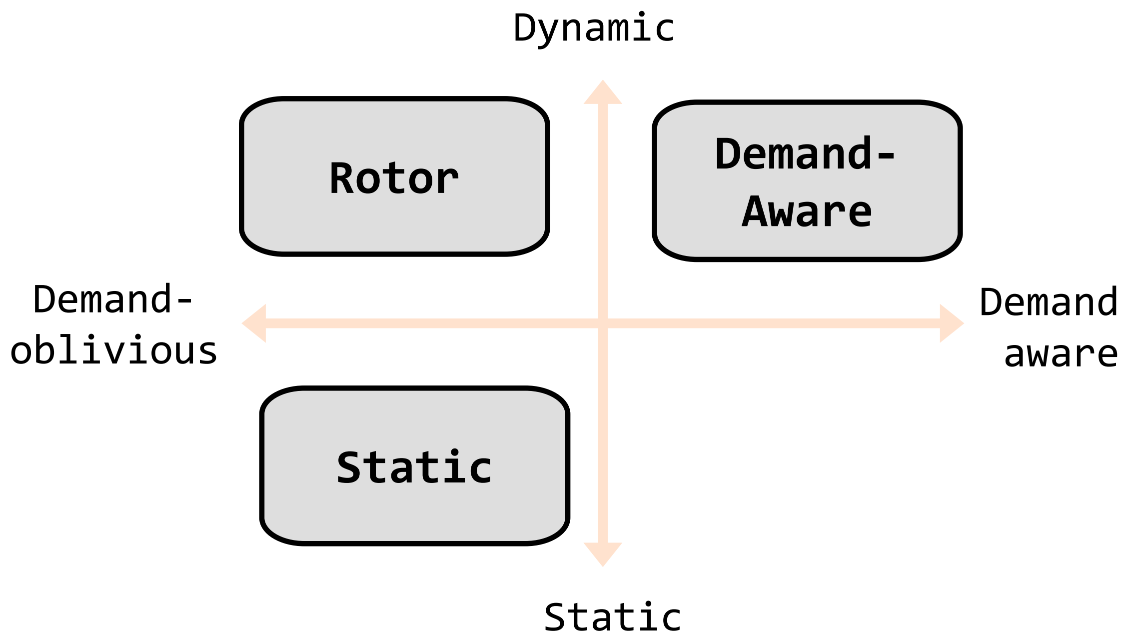
Which approach is best?

As always in CS:
It depends…

Examples:
Match or Mismatch?

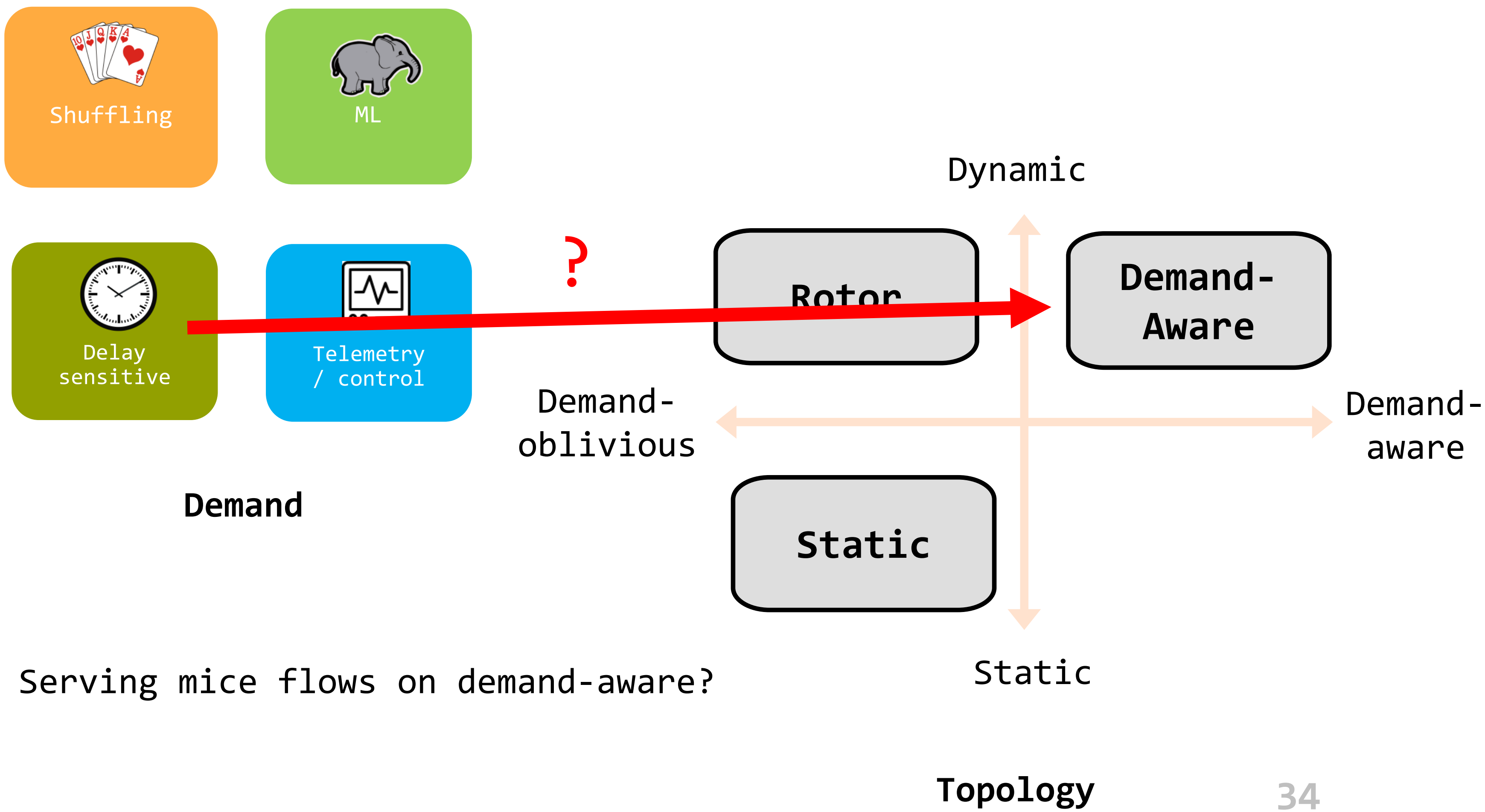


Demand



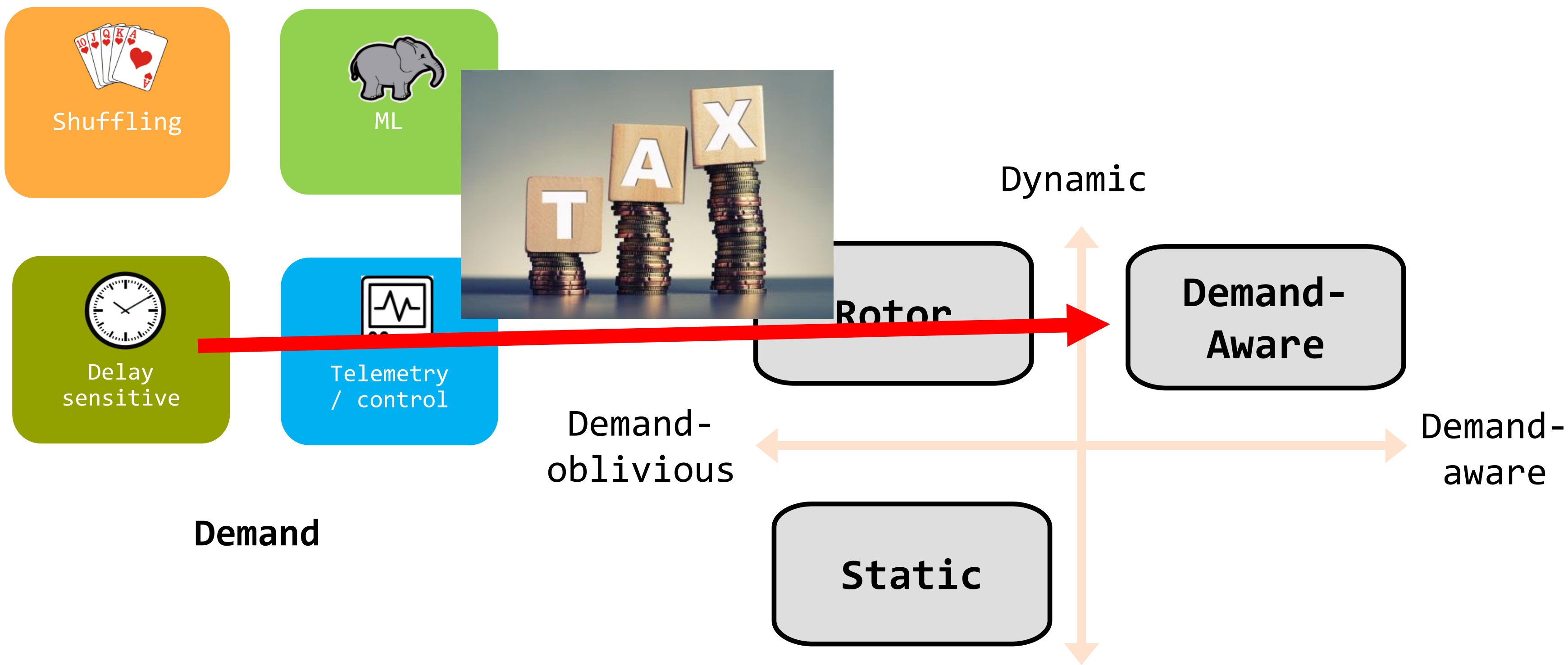
Topology

Examples:
Match or Mismatch?



Serving mice flows on demand-aware?

Examples:
Match or Mismatch?



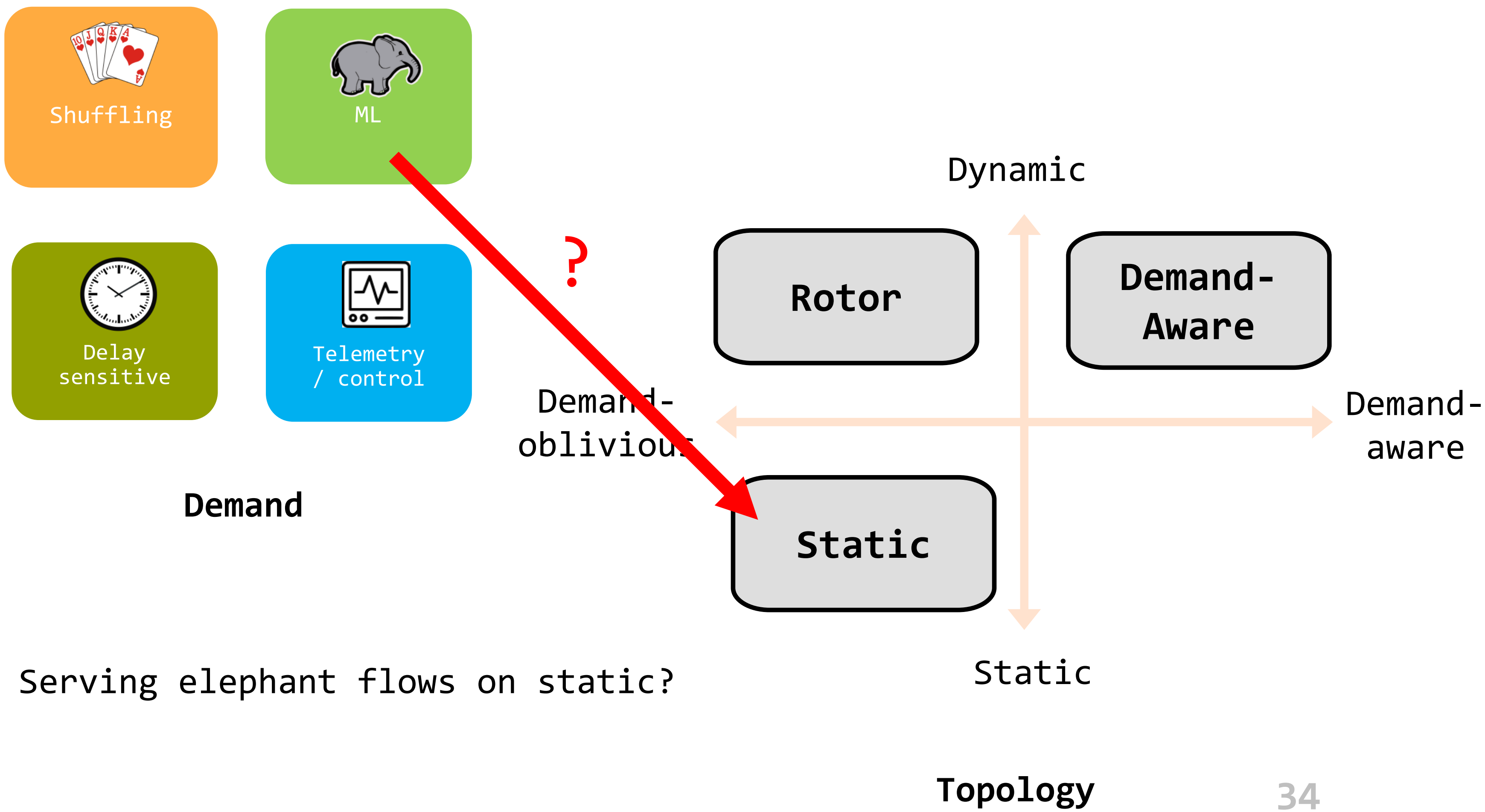
Serving mice flows on demand-aware?

Bad idea! Latency tax.

Static

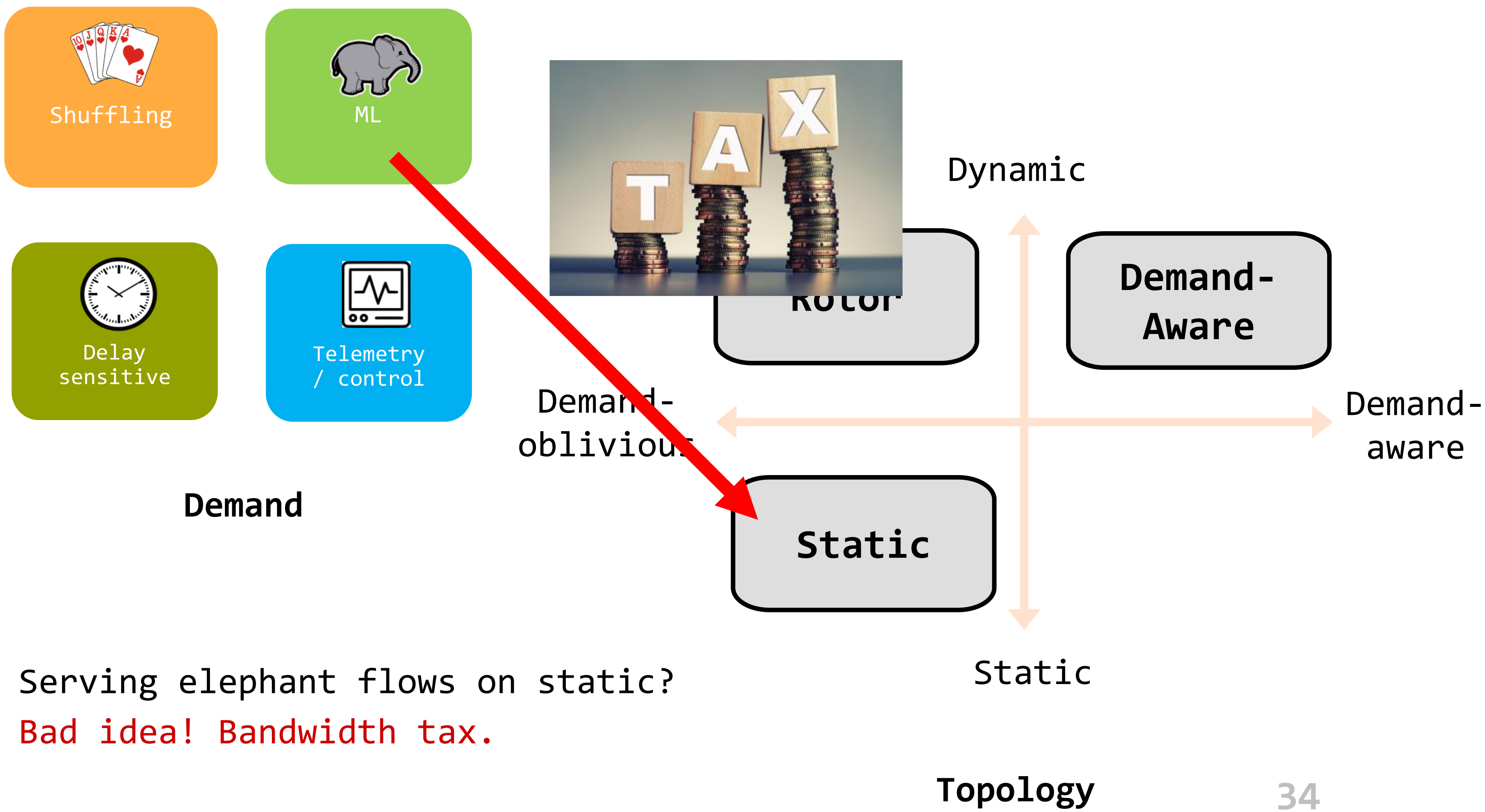
Topology

Examples:
Match or Mismatch?



Serving elephant flows on static?

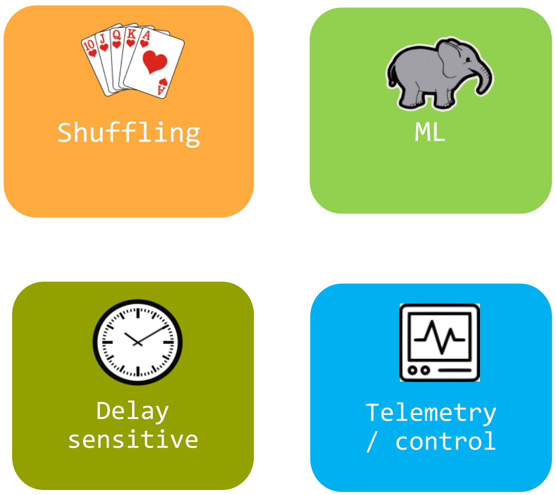
Examples:
Match or Mismatch?



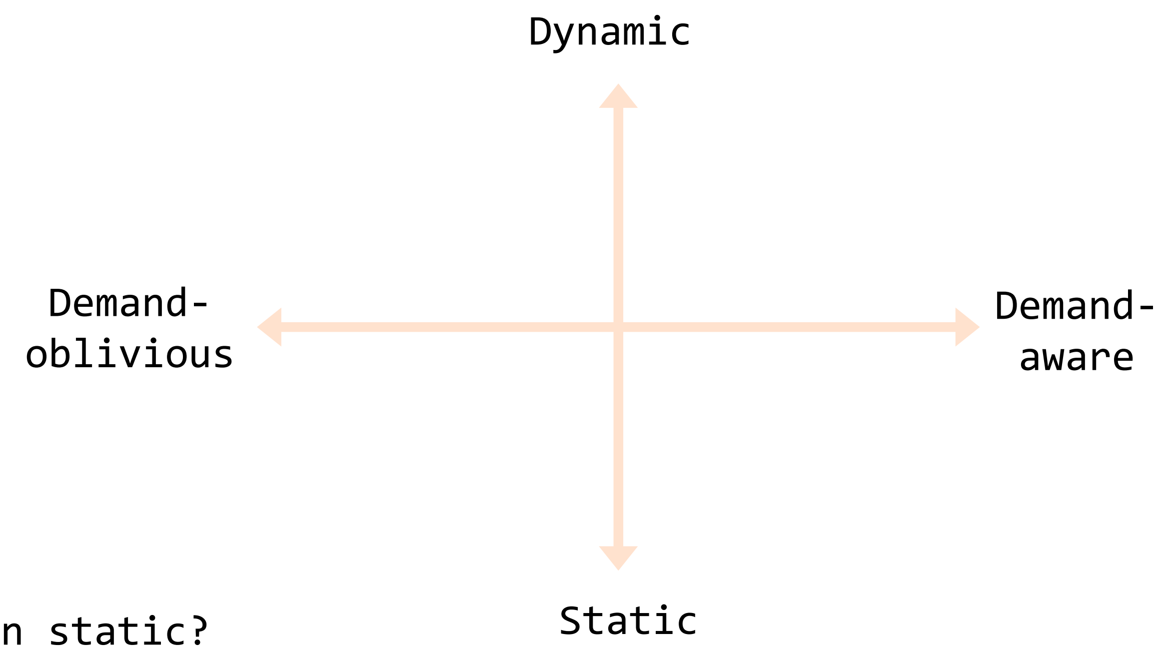
Serving elephant flows on static?

Bad idea! Bandwidth tax.

Examples:
Match or Mismatch?



Demand

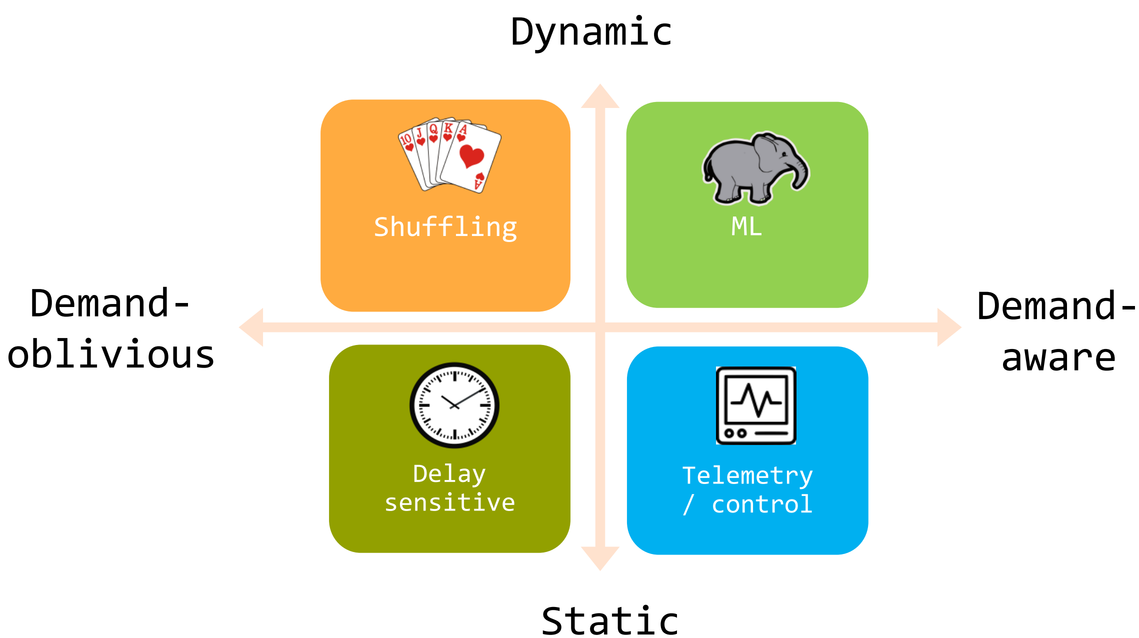


Serving elephant flows on static?

Bad idea! Bandwidth tax.

Topology

Optimal Solution: It's a Match!



We have a first approach:
Cerberus*** serves traffic on the “best topology”! (Optimality open)

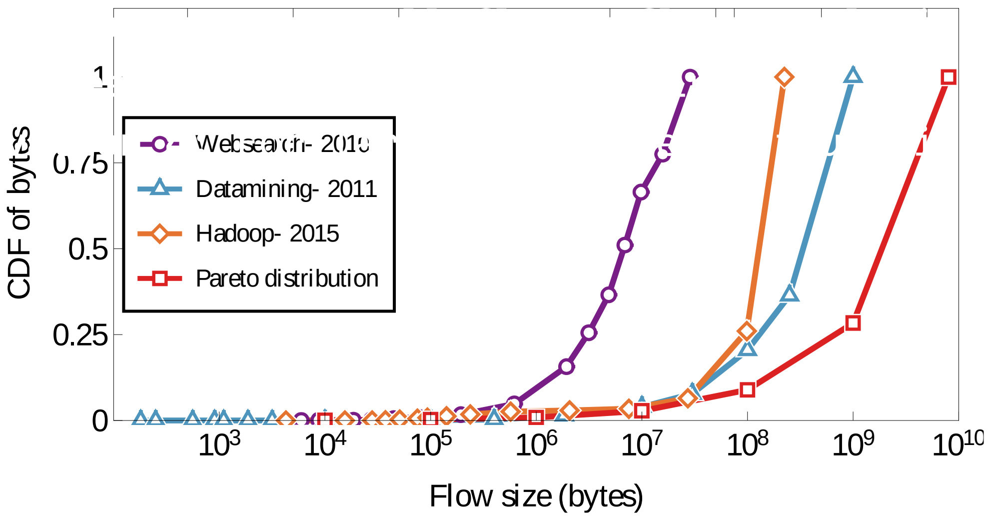
* Cerberus: The Power of Choices in Datacenter Topology Design. Griner et al. ACM SIGMETRICS, 2022.

Flow Size Matters

On what should topology type depend? We argue: flow size.

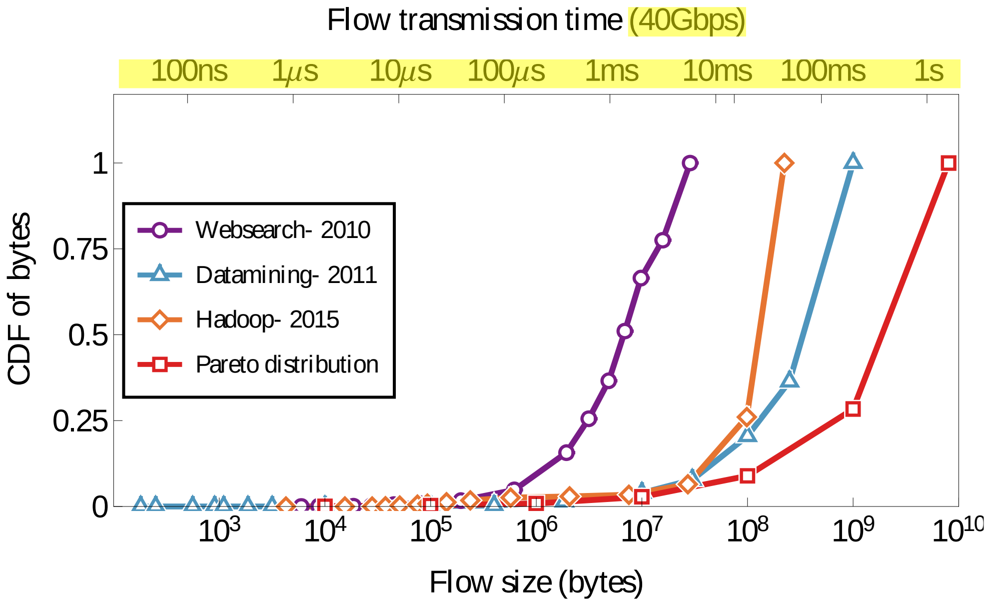
Flow Size Matters

On what should topology type depend? We argue: flow size.



⇢ **Observation 1:** Different apps have different flow size distributions.

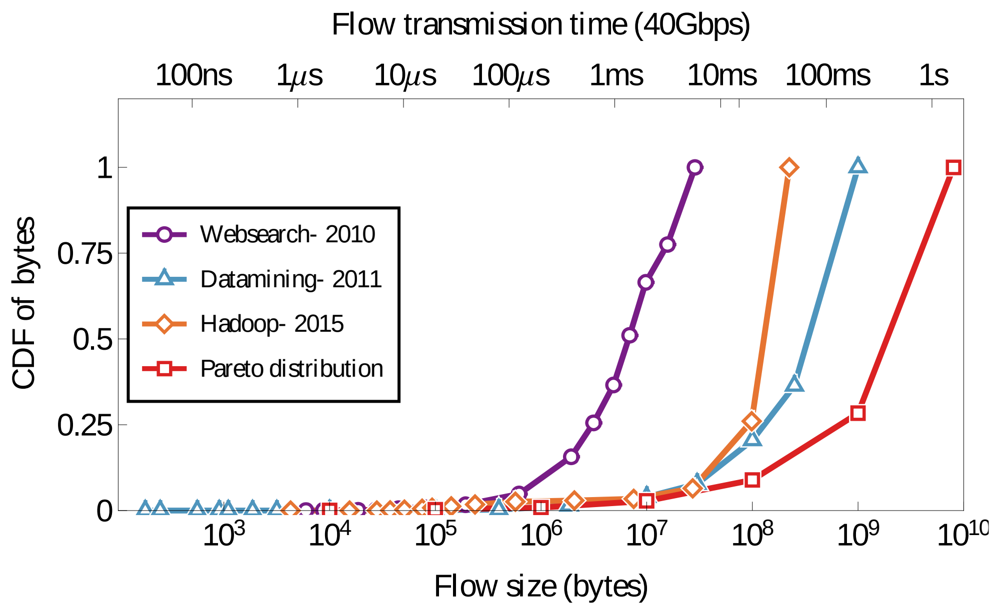
Flow Size Matters



⇢ **Observation 1:** Different apps have different flow size distributions.

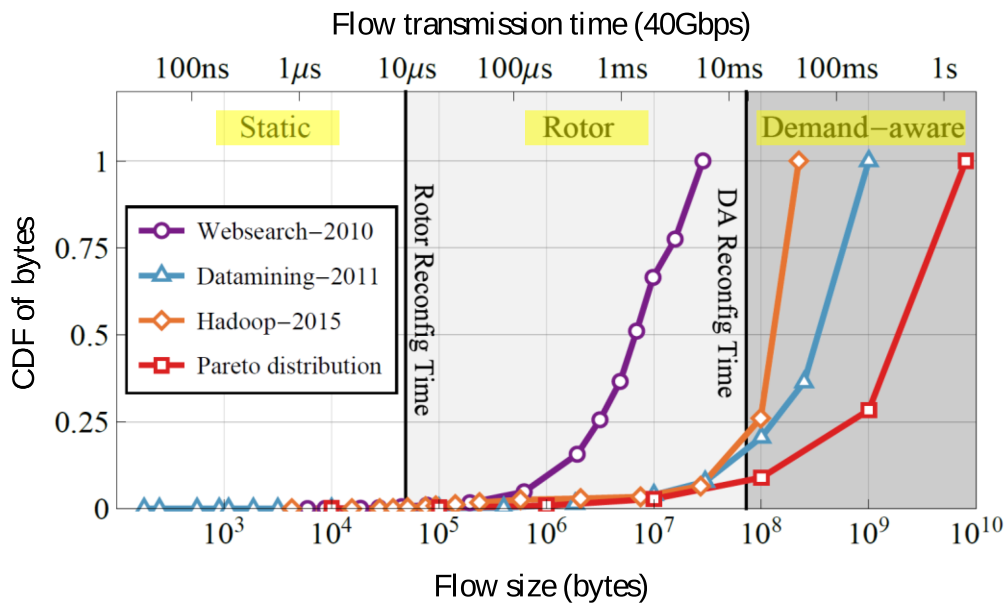
⇢ **Observation 2:** The transmission time of a flow depends on its size.

Flow Size Matters



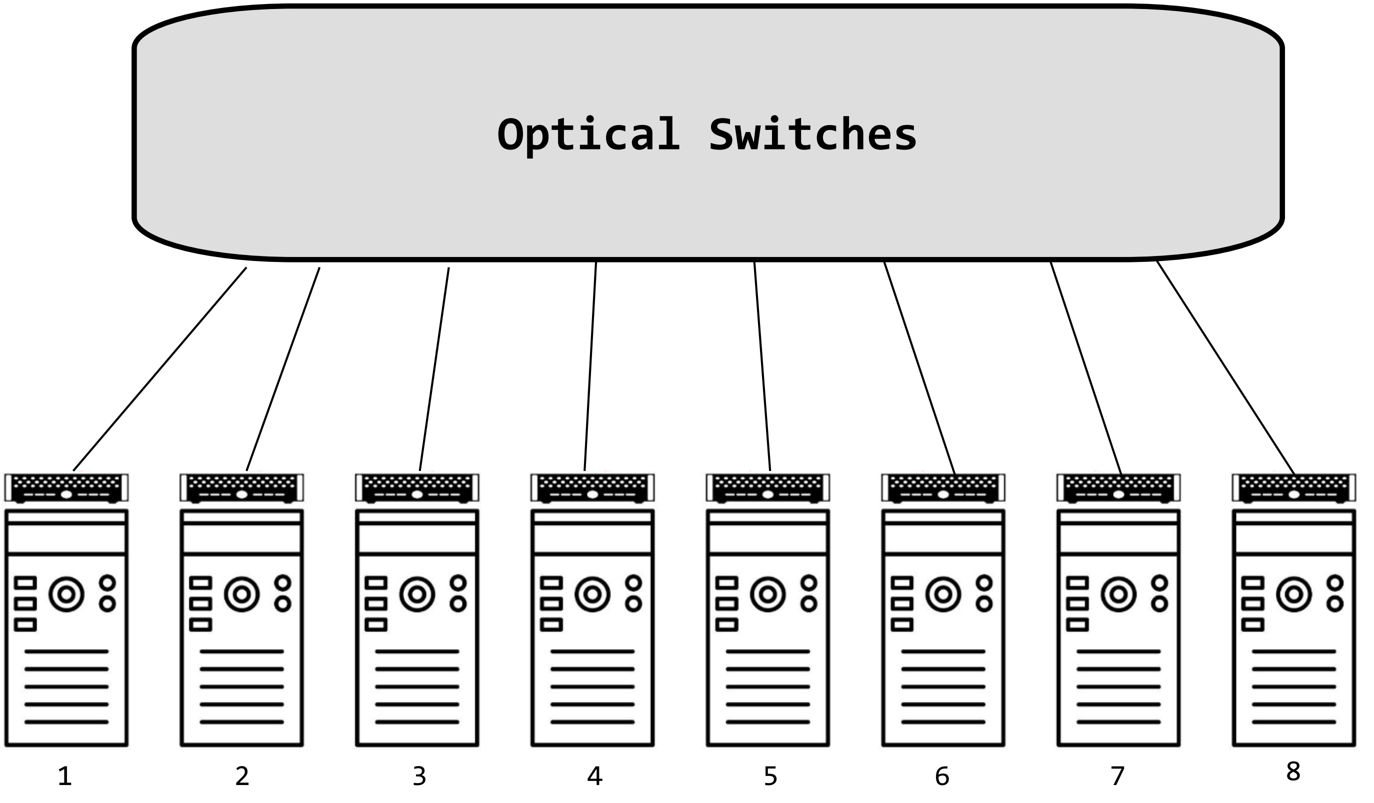
- **Observation 1:** Different apps have different flow size distributions.
- **Observation 2:** The transmission time of a flow depends on its size.
- **Observation 3:** For small flows, flow completion time suffers if network needs to be reconfigured first.
- **Observation 4:** For large flows, reconfiguration time may amortize.

Flow Size Matters

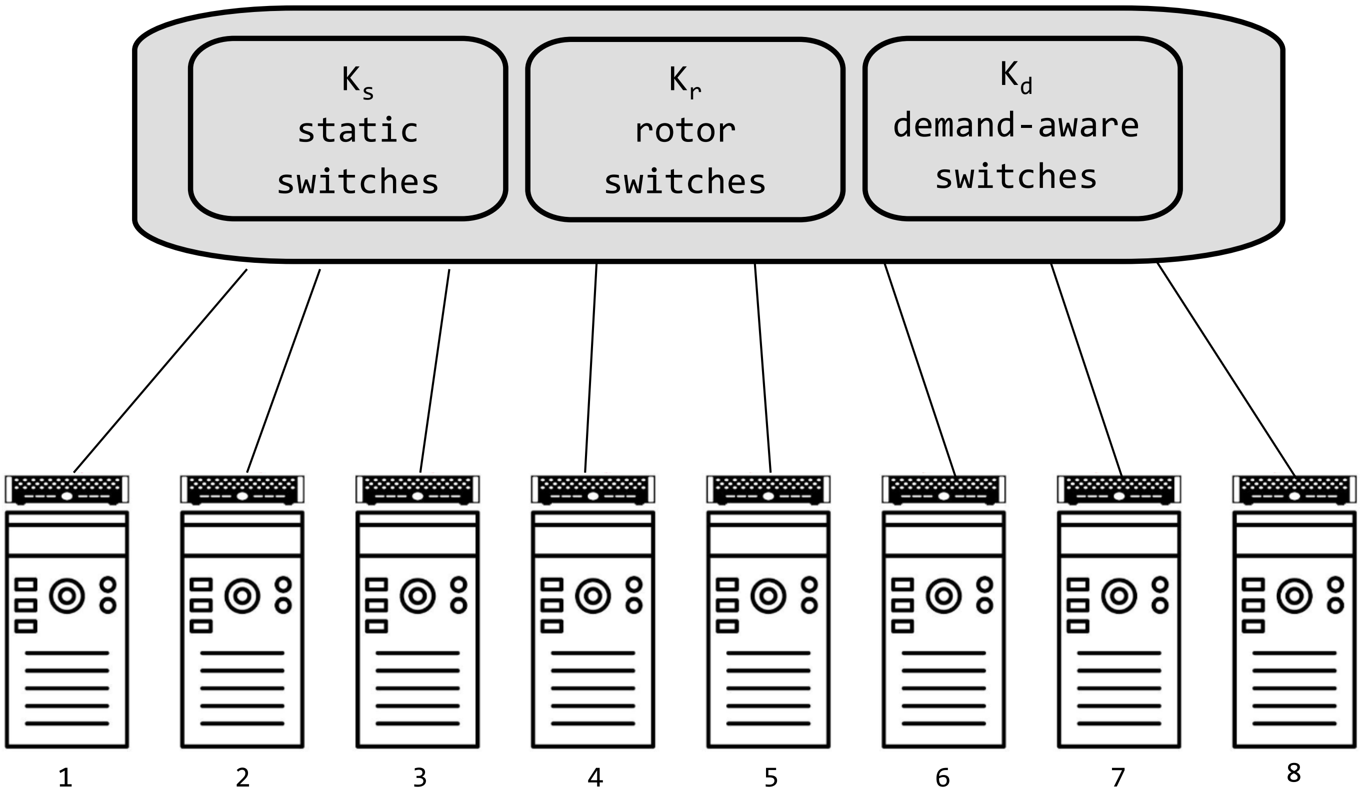


- **Observation 1:** Different apps have different flow size distributions.
- **Observation 2:** The transmission time of a flow depends on its size.
- **Observation 3:** For small flows, flow completion time suffers if network needs to be reconfigured first.
- **Observation 4:** For large flows, reconfiguration time may amortize.

Cerberus

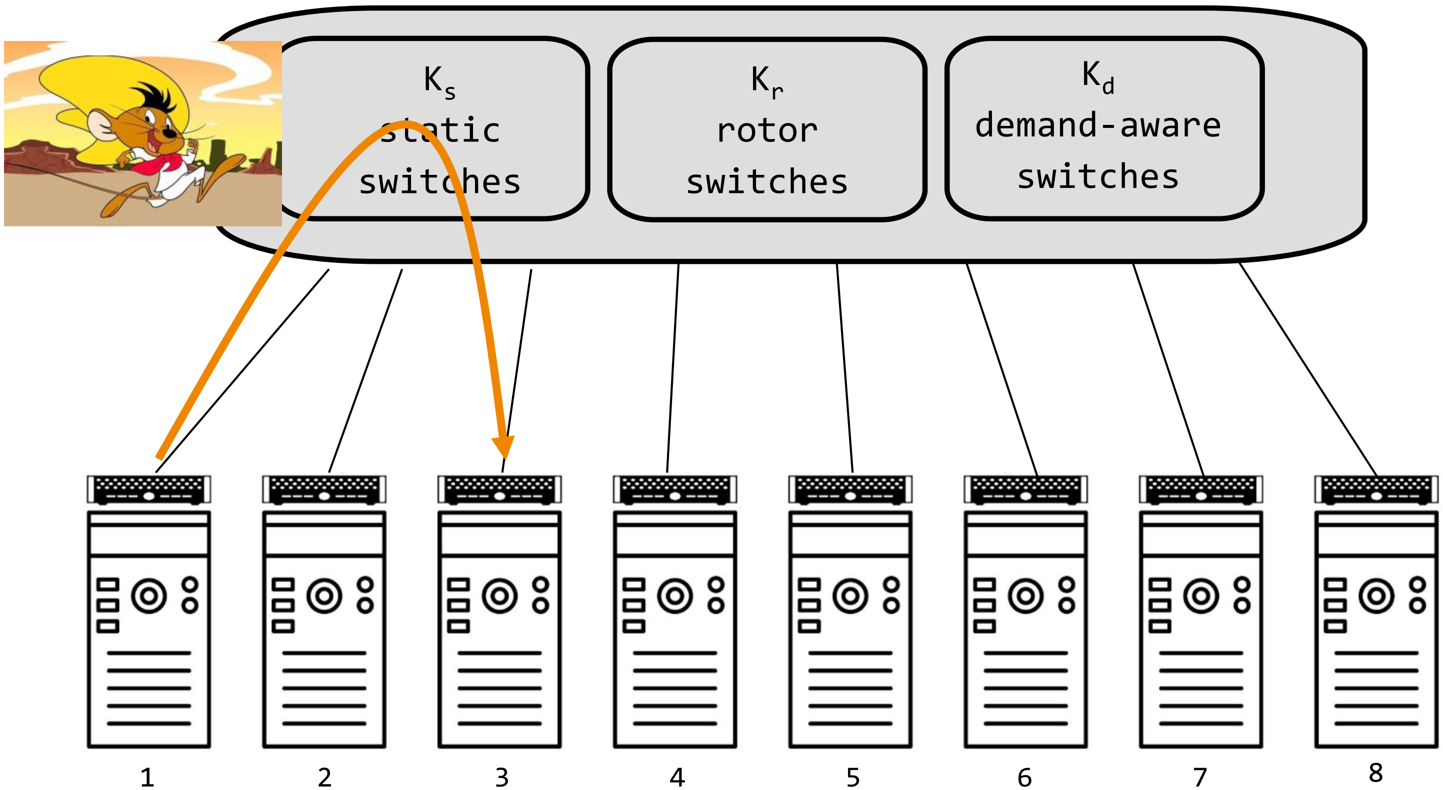


Cerberus



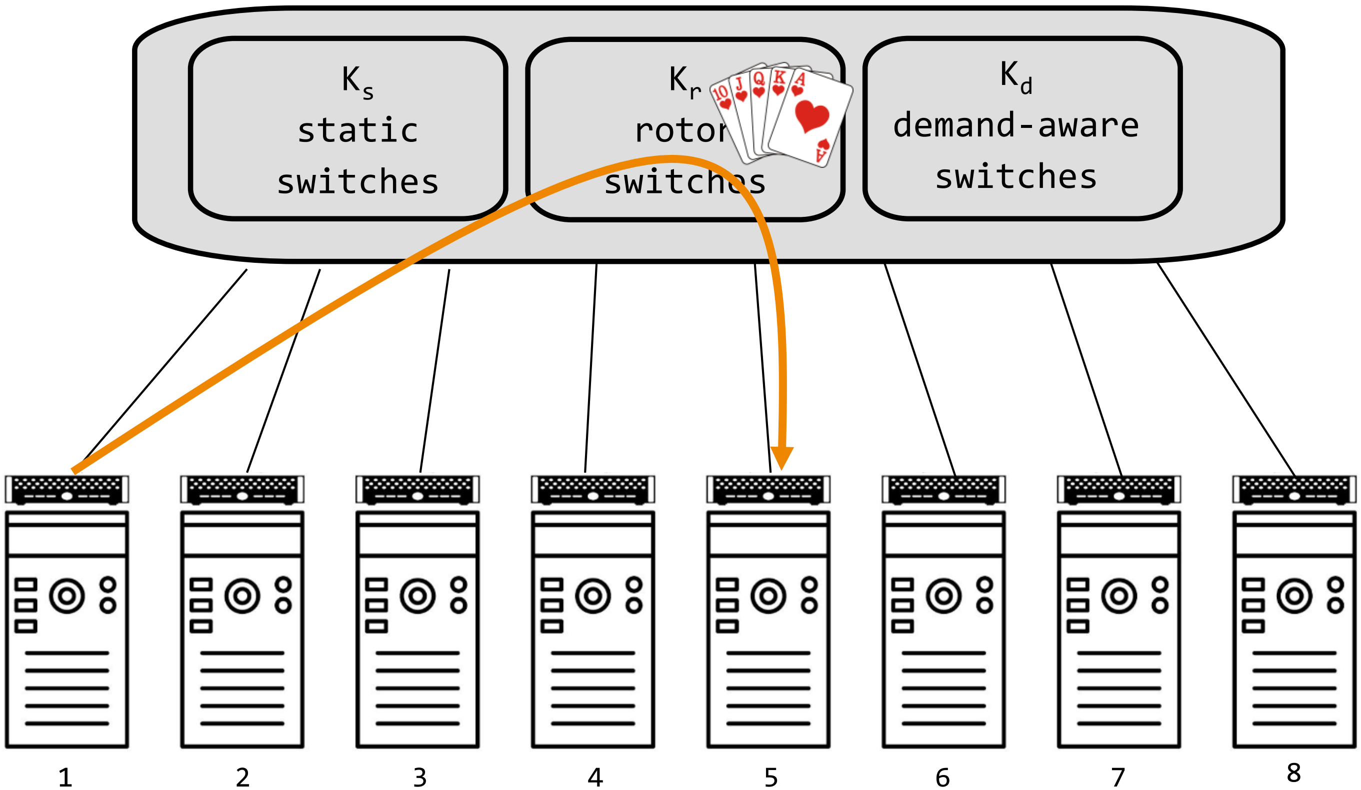
K_s
static
switches
K_r
rotor
switches
K_d
demand-aware
switches
1
2
3
4
5
6
7
8

Cerberus



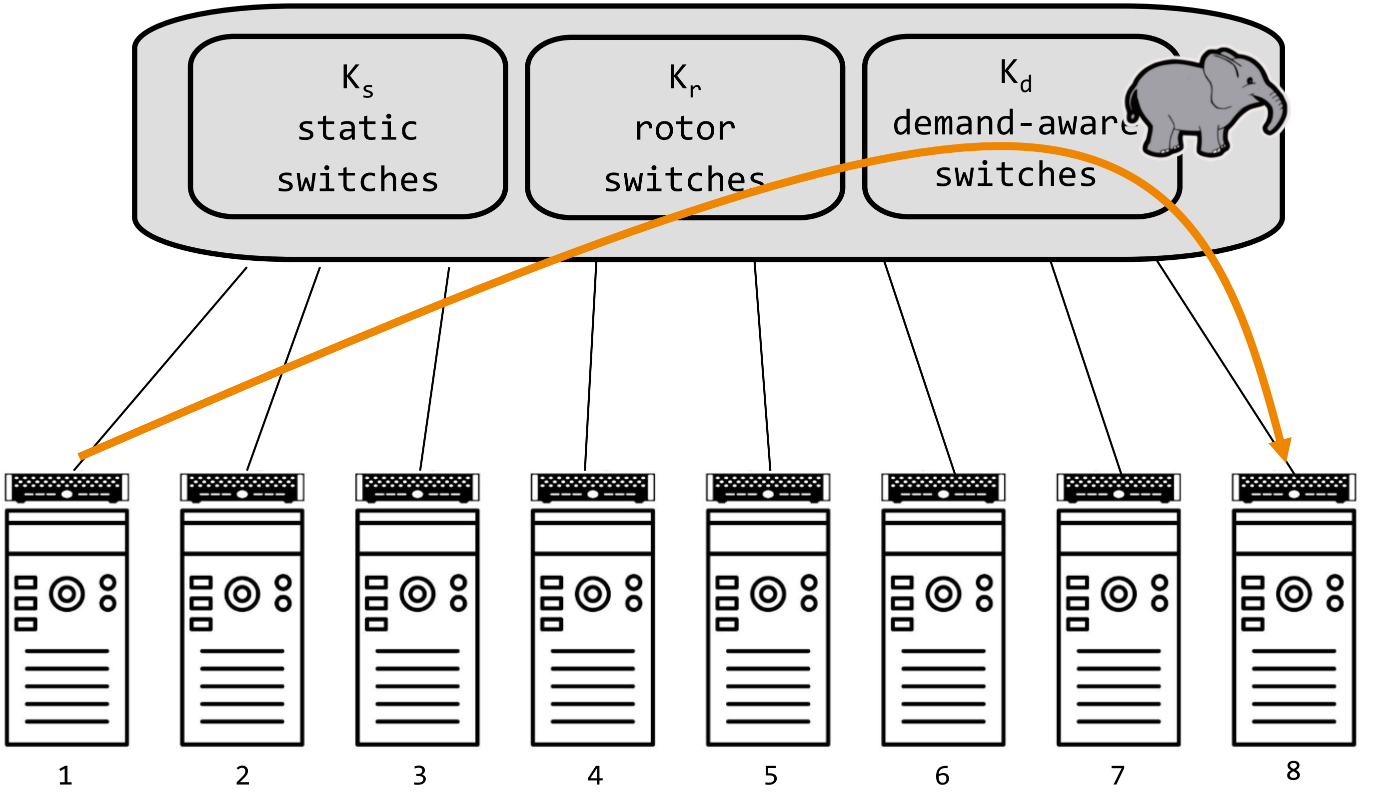
Scheduling: Small flows go via static switches…

Cerberus



Scheduling: … medium flows via rotor switches…

Cerberus



Scheduling: … and large flows via demand-aware switches
(if one available, otherwise via rotor).

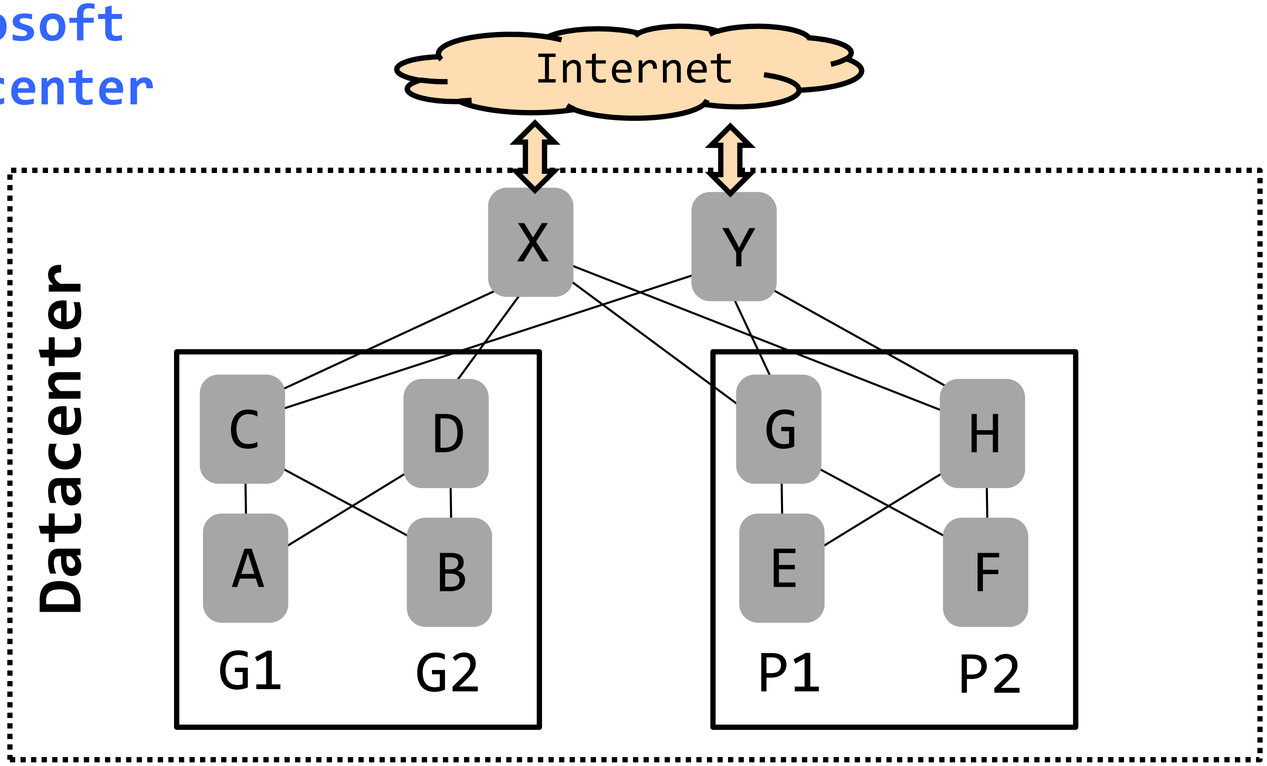
Roadmap

- Performance: Self-adjusting datacenter networks
- Modelling: How to model workloads, such as ML workloads?
- Dependability: Self-correcting MPLS networks
- More Use cases for self-driving networks

Challenge: Complexity

Especially Under Failures (Policy Compliance)

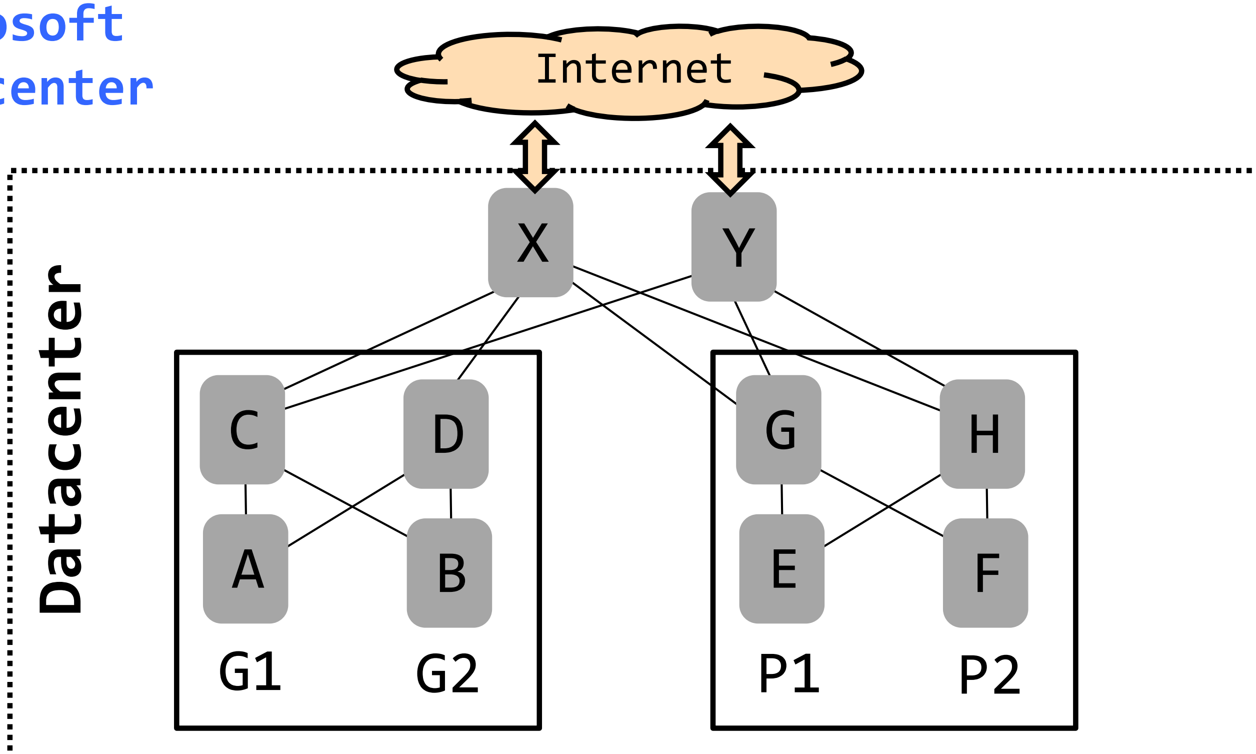
Example: BGP in **Microsoft datacenter**



Challenge: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in **Microsoft datacenter**



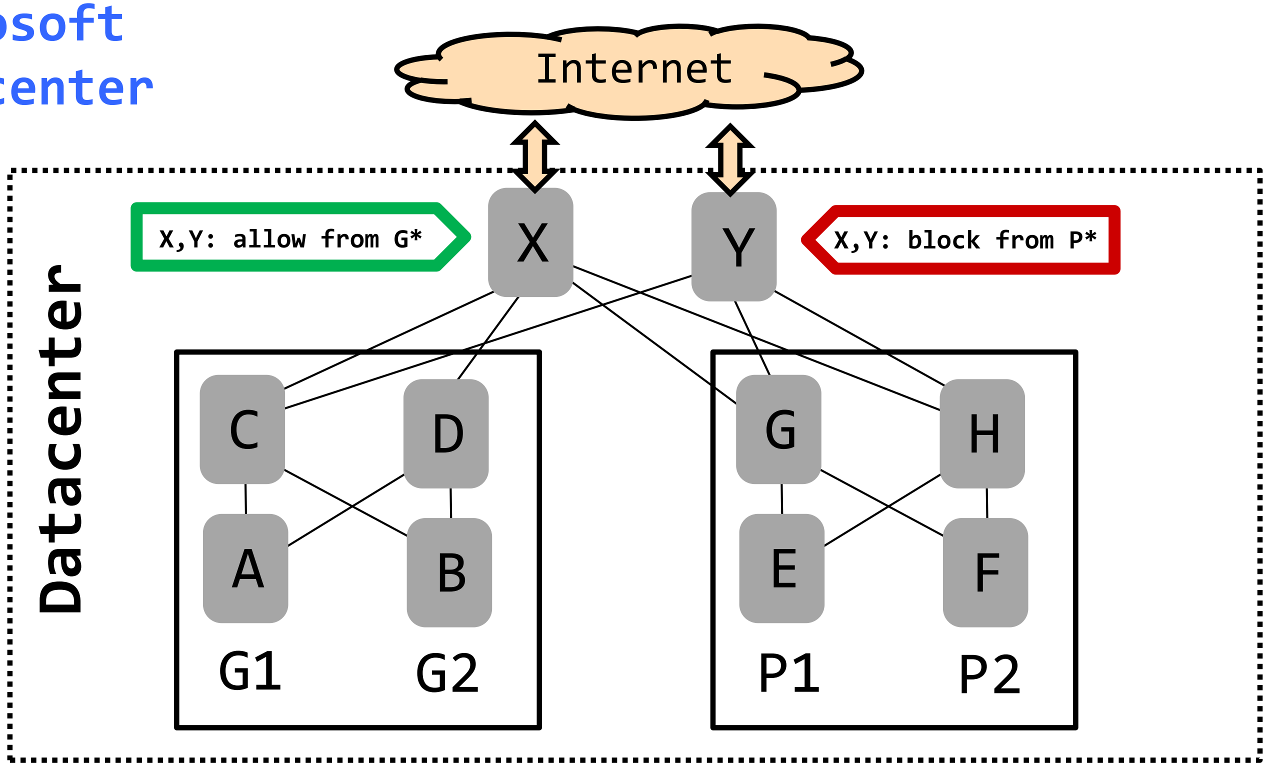
Cluster with globally reachable services

Cluster with internally accessible services

Challenge: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in **Microsoft datacenter**



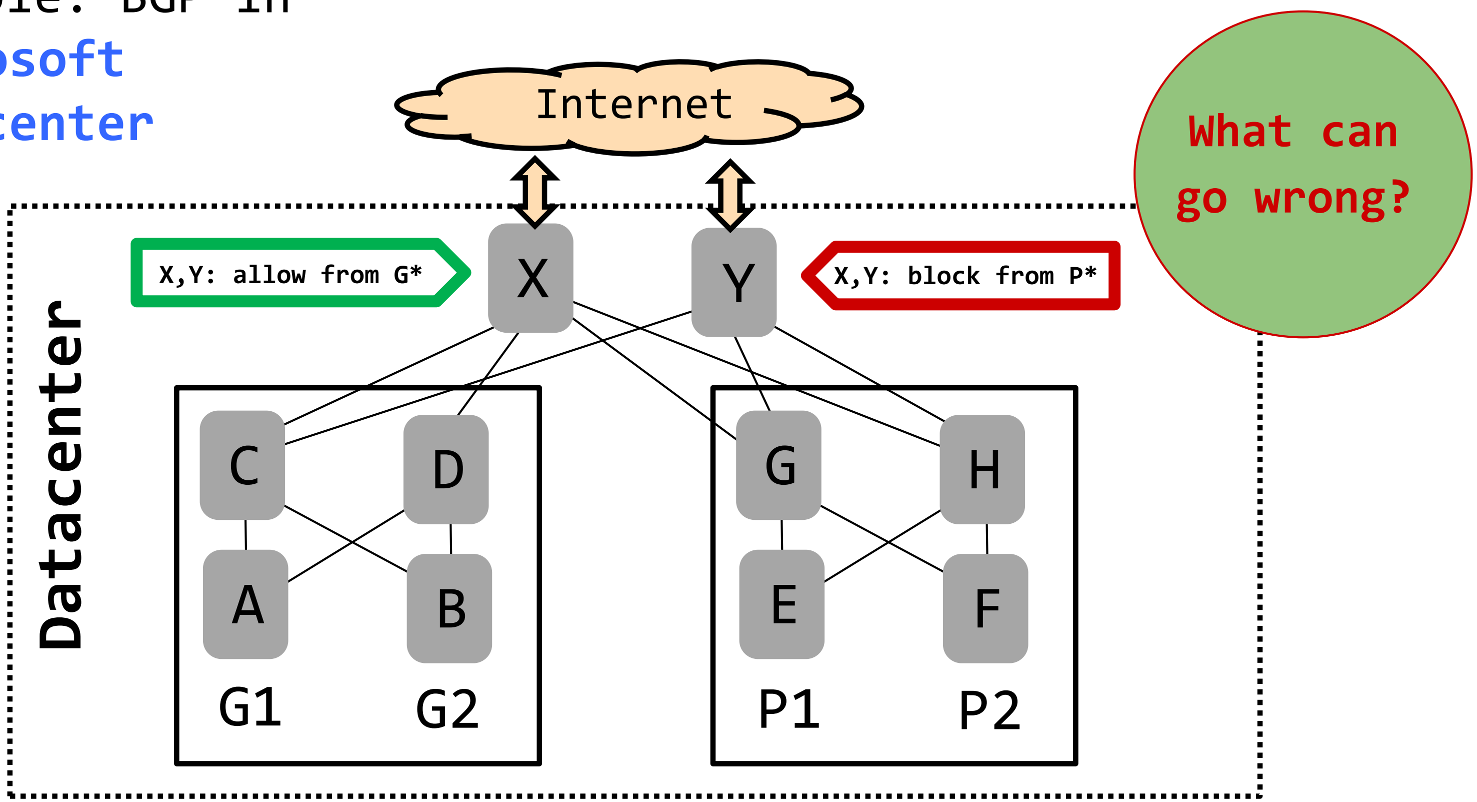
Cluster with globally reachable services

Cluster with internally accessible services

Challenge: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in **Microsoft datacenter**



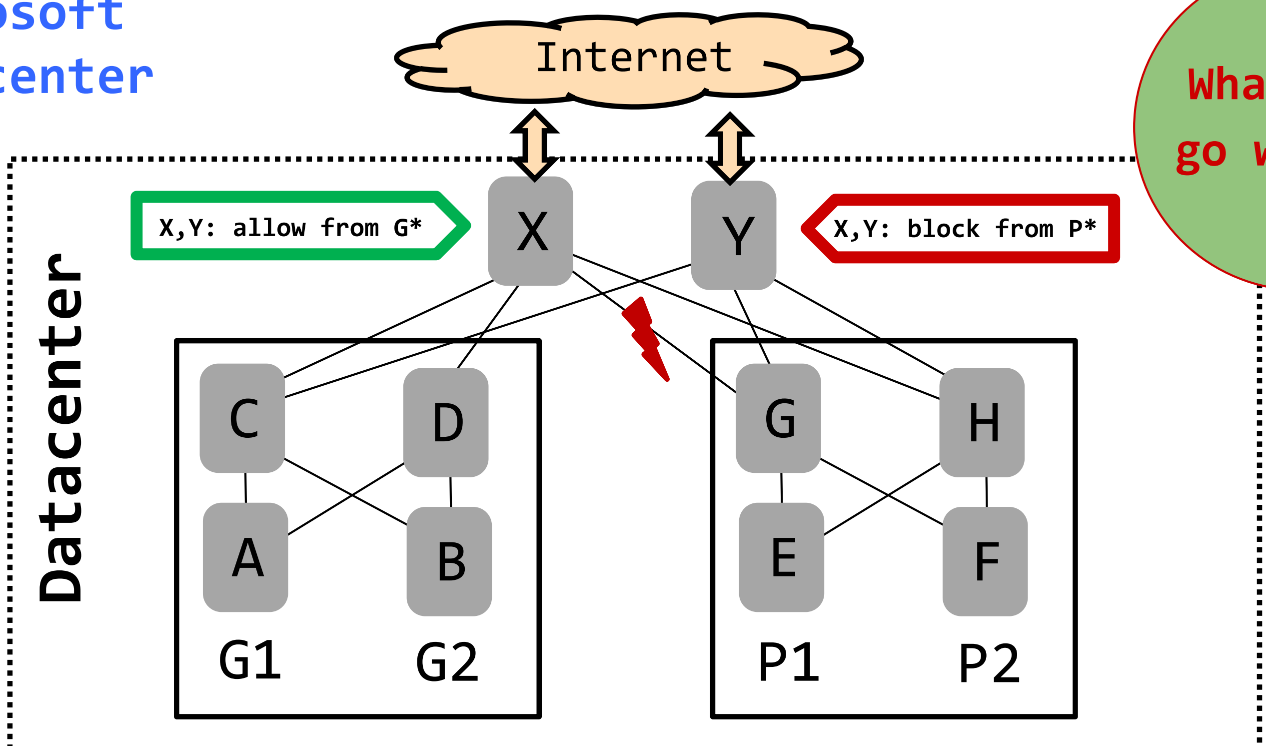
Cluster with globally reachable services

Cluster with internally accessible services

Challenge: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in **Microsoft datacenter**

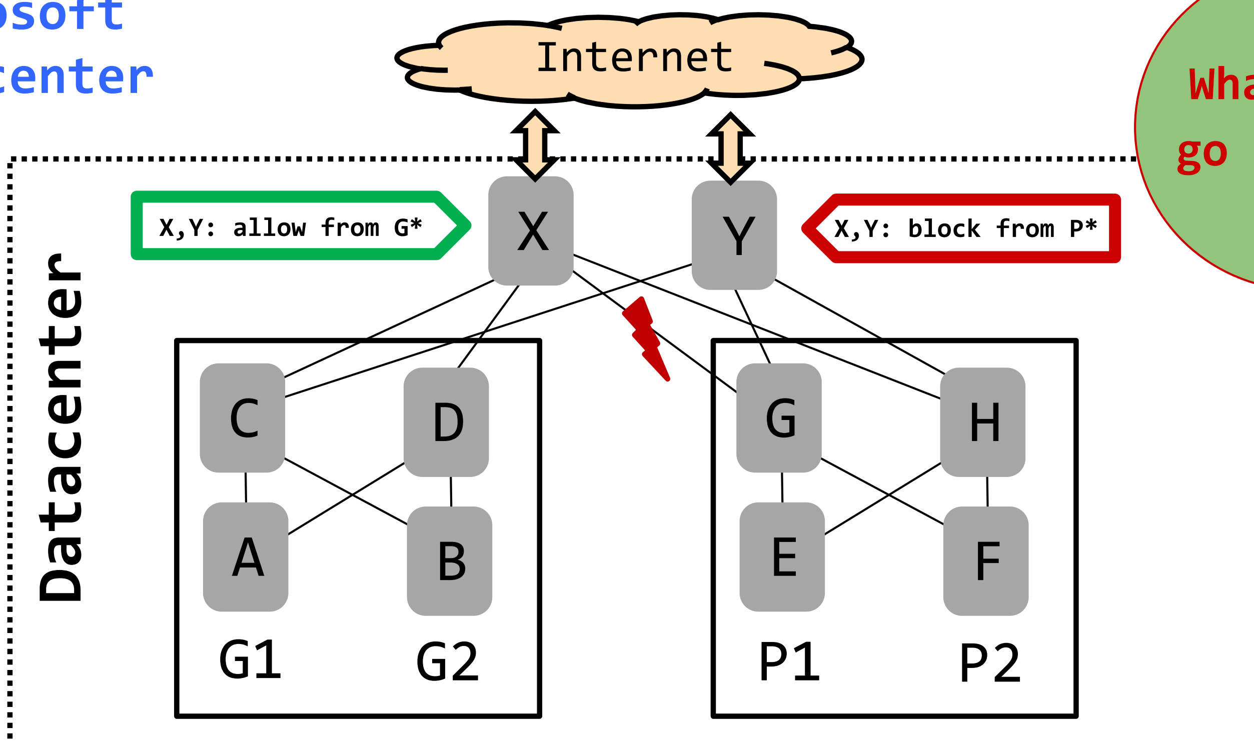


What can go wrong?

Challenge: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in **Microsoft datacenter**



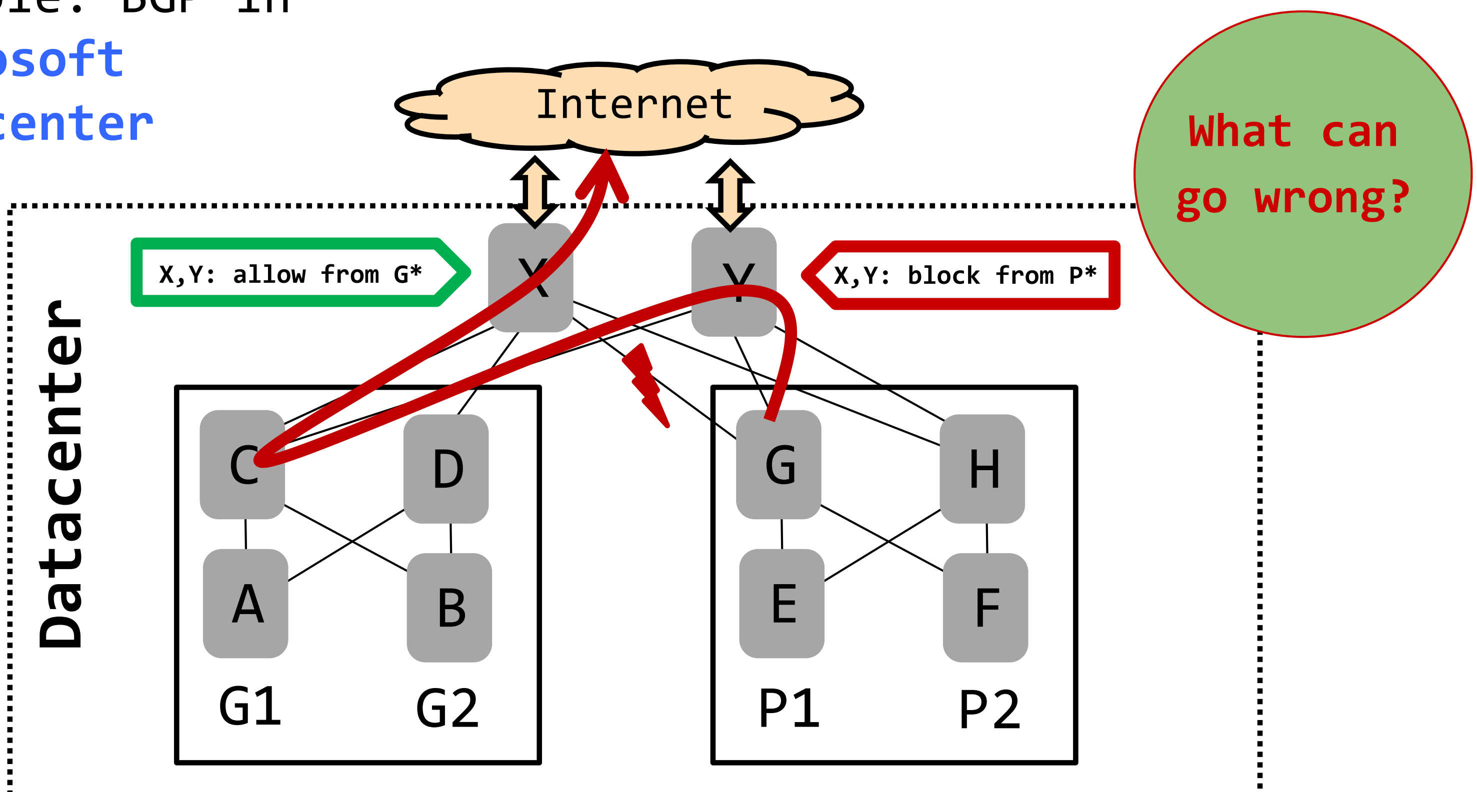
What can go wrong?

If link (G,X) fails and traffic from G is rerouted via Y and C to X:
X announces (does not block) G and H as it comes from C. (Note: BGP.)

Challenge: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in **Microsoft datacenter**

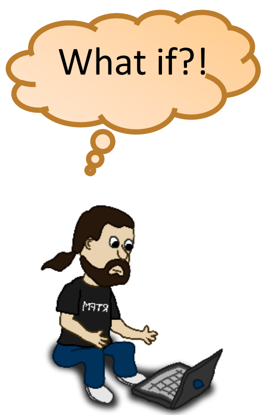


If link (G,X) fails and traffic from G is rerouted via Y and C to X:
X announces (does not block) G and H as it comes from C. (Note: BGP.)

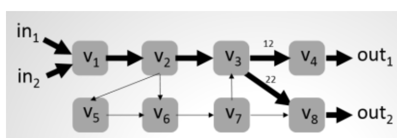
Dependable Networks with

Automated Whatif Analysis

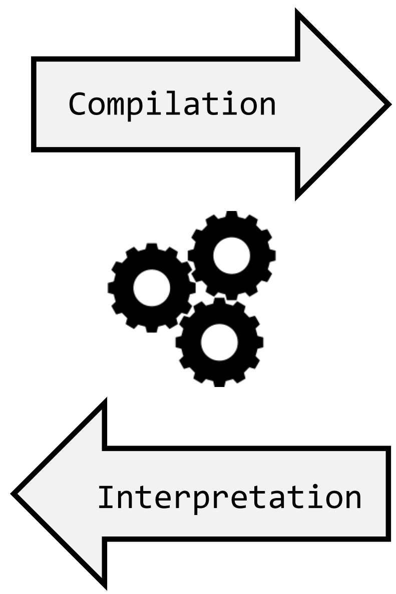
⇢ Formal methods good for verifying networks! E.g., P-Rex for MPLS (Jensen et al. CoNEXT’19)



FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\bot$	(v_1, v_2)	$push(10)$
	in_2	$\bot$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$
	(v_5, v_6)	71	(v_6, v_7)	$swap(72)$
τ_{v_7}	(v_6, v_7)	31	(v_7, v_3)	pop
	(v_6, v_7)	62	(v_7, v_3)	$swap(11)$
	(v_6, v_7)	72	(v_7, v_8)	$swap(22)$
τ_{v_8}	(v_3, v_8)	22	out_2	pop
	(v_7, v_8)	22	out_2	pop



local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$
global FFT	**Out-I**	**In-Label**	**Out-I**	**op**
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$



$$pX \Rightarrow qXX$$
$$pX \Rightarrow qYX$$
$$qY \Rightarrow rYY$$
$$rY \Rightarrow r$$
$$rX \Rightarrow pX$$

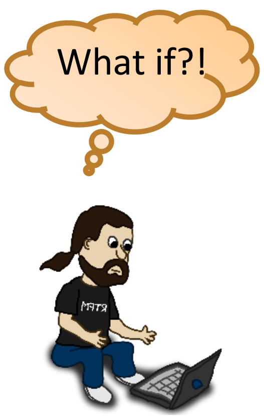
Router **configurations**
(Cisco, Juniper, etc.)

Formal language
which supports
automated analysis

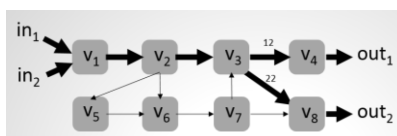
Dependable Networks with

Automated Whatif Analysis

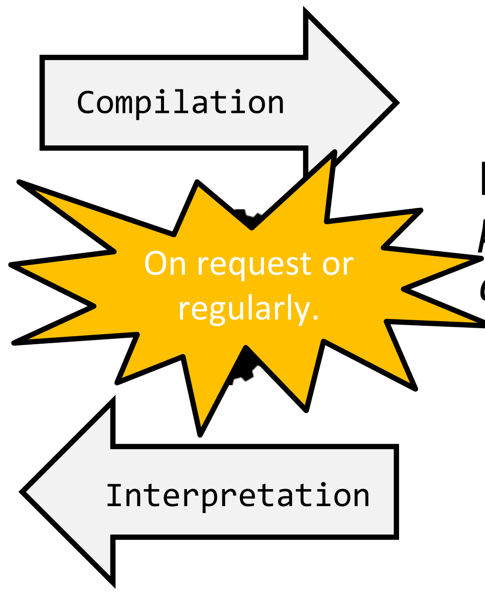
⇢ Formal methods good for verifying networks! E.g., P-Rex for MPLS (Jensen et al. CoNEXT’19)



FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\bot$	(v_1, v_2)	$push(10)$
	in_2	$\bot$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$
	(v_5, v_6)	71	(v_6, v_7)	$swap(72)$
τ_{v_7}	(v_6, v_7)	31	(v_7, v_3)	pop
	(v_6, v_7)	62	(v_7, v_3)	$swap(11)$
	(v_6, v_7)	72	(v_7, v_8)	$swap(22)$
τ_{v_8}	(v_3, v_8)	22	out_2	pop
	(v_7, v_8)	22	out_2	pop



local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$
global FFT	**Out-I**	**In-Label**	**Out-I**	**op**
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$



$pX \Rightarrow qXX$

$pX \Rightarrow qYX$

$qY \Rightarrow rYY$

$rY \Rightarrow r$

$rX \Rightarrow pX$

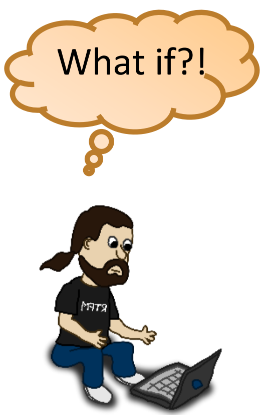
Router **configurations** (Cisco, Juniper, etc.)

Formal language which supports ***automated analysis***

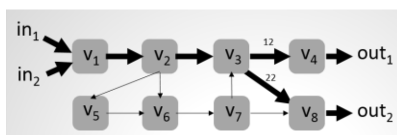
Dependable Networks with

Automated Whatif Analysis

⇢ Formal methods good for verifying networks! E.g., P-Rex for MPLS (Jensen et al. CoNEXT’19)



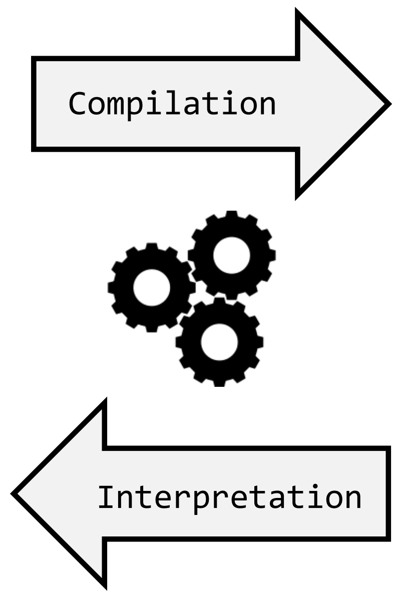
FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\bot$	(v_1, v_2)	$push(10)$
	in_2	$\bot$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$
	(v_5, v_6)	71	(v_6, v_7)	$swap(72)$
τ_{v_7}	(v_6, v_7)	31	(v_7, v_3)	pop
	(v_6, v_7)	62	(v_7, v_3)	$swap(11)$
	(v_6, v_7)	72	(v_7, v_8)	$swap(22)$
τ_{v_8}	(v_3, v_8)	22	out_2	pop
	(v_7, v_8)	22	out_2	pop



local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$

global FFT	Out-I	In-Label	Out-I	op
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$

Router configurations
(Cisco, Juniper, etc.)

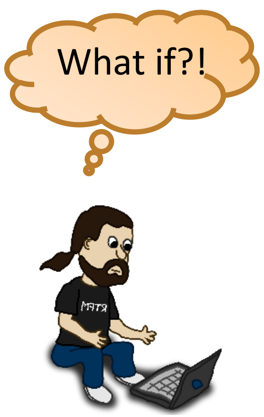


Many alternatives: ***automata*** theory, binary decision diagrams (***BDDs***), ***games*** (e.g., Stackelberg, Petri nets), ***SMTs***, ***ILPs*** …

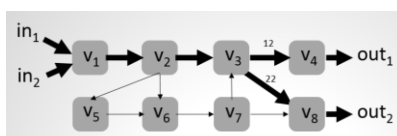
Even more automation:

Synthesis

⇢ Formal methods good for verifying networks! E.g., P-Rex for MPLS (Jensen et al. CoNEXT’19)



FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\bot$	(v_1, v_2)	$push(10)$
	in_2	$\bot$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$
	(v_5, v_6)	71	(v_6, v_7)	$swap(72)$
τ_{v_7}	(v_6, v_7)	31	(v_7, v_3)	pop
	(v_6, v_7)	62	(v_7, v_3)	$swap(11)$
	(v_6, v_7)	72	(v_7, v_8)	$swap(22)$
τ_{v_8}	(v_3, v_8)	22	out_2	pop
	(v_7, v_8)	22	out_2	pop



local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$

global FFT	Out-I	In-Label	Out-I	op
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$

Router configurations
(Cisco, Juniper, etc.)

Compilation

Where configuration not compliant?

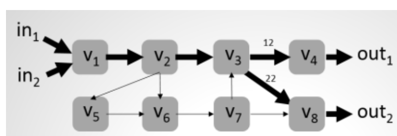
Synthesis!

Even more automation:

Synthesis

- Formal methods good for verifying networks! E.g., P-Rex for MPLS (Jensen et al. CoNEXT'19)

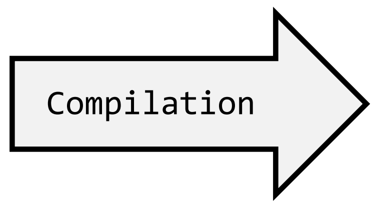
FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\bot$	(v_1, v_2)	$push(10)$
	in_2	$\bot$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$
	(v_5, v_6)	71	(v_6, v_7)	$swap(72)$
τ_{v_7}	(v_6, v_7)	31	(v_7, v_3)	pop
	(v_6, v_7)	62	(v_7, v_3)	$swap(11)$
	(v_6, v_7)	72	(v_7, v_8)	$swap(22)$
τ_{v_8}	(v_3, v_8)	22	out_2	pop
	(v_7, v_8)	22	out_2	pop

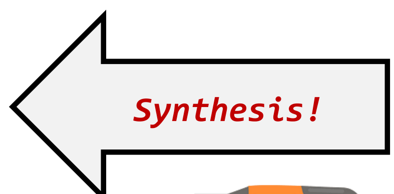


local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$

global FFT	Out-I	In-Label	Out-I	op
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$

Router configurations
(Cisco, Juniper, etc.)





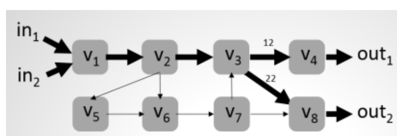
Where configuration not compliant?

Even more automation:

Synthesis

⇢ Formal methods good for verifying networks! E.g., P-Rex for MPLS (Jensen et al. CoNEXT’19)

FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\bot$	(v_1, v_2)	$push(10)$
	in_2	$\bot$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$
	(v_5, v_6)	71	(v_6, v_7)	$swap(72)$
τ_{v_7}	(v_6, v_7)	31	(v_7, v_3)	pop
	(v_6, v_7)	62	(v_7, v_3)	$swap(11)$
	(v_6, v_7)	72	(v_7, v_8)	$swap(22)$
τ_{v_8}	(v_3, v_8)	22	out_2	pop
	(v_7, v_8)	22	out_2	pop



local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$

global FFT	Out-I	In-Label	Out-I	op
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$

Compilation ⇒

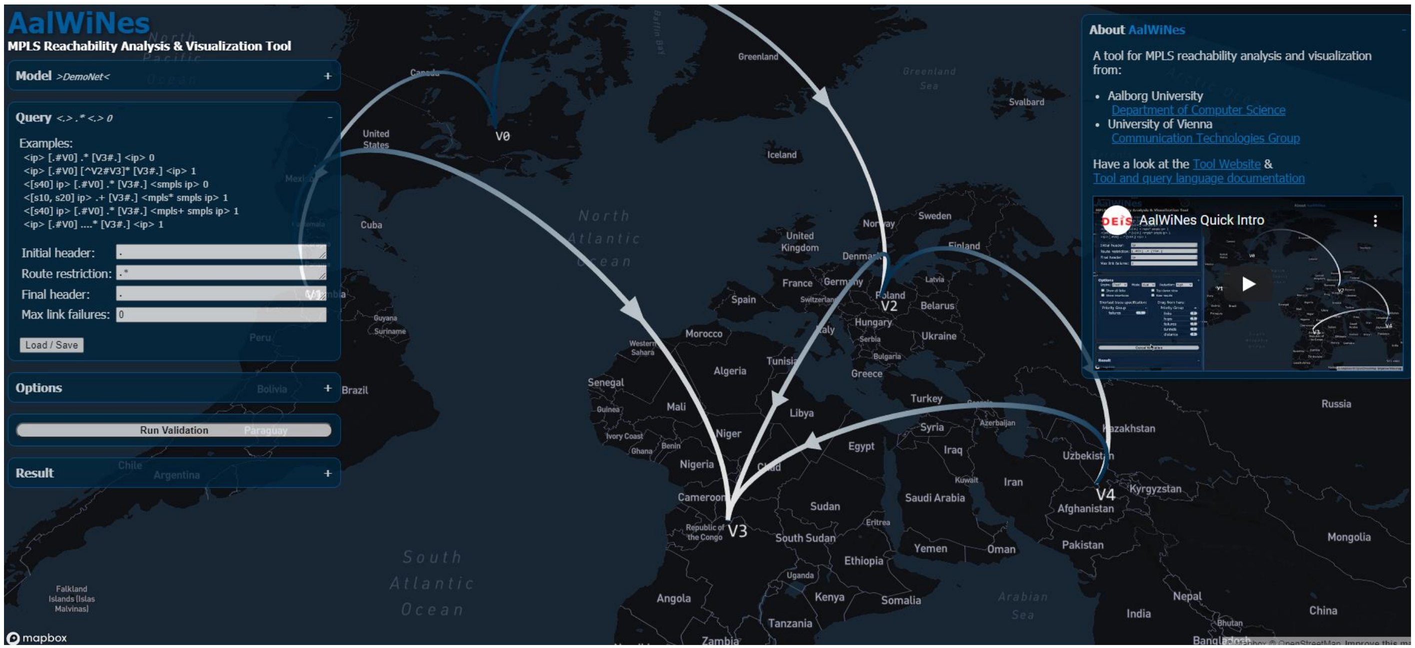
Where configuration not compliant?

⇐ *Synthesis!*

Router configurations
(Cisco, Juniper, etc.)

Literature: P-Rex: Fast Verification of MPLS Networks with Multiple Link Failures. Jensen et al. ACM CoNEXT, 2018.

P-Rex / AalWiNes Tool



Tool: https://demo.aalwines.cs.aau.dk/
Youtube: https://www.youtube.com/watch?v=mvXAn9i7_Q0

Efficient Synthesis?

ML+FM!

- Formal ***synthesis slower*** than verification
- An opportunity for using ML!
- ***Ideally ML+FM***: guarantees from formal methods, performance from ML
- For example: synthesize with ML then verify with formal methods
- Examples: DeepMPLS, DeepBGP, …

On a related note:

Human in the Loop?

⇢ When and how to keep the human in the loop?

⇢ Critical: can system realize when ***help*** is needed?

⇢ But AI tools (e.g. LLM) may also influence the human: can the operator become ***too confident*** with such tools?

⇢ Challenges to be discussed!

Roadmap

- Performance: Self-adjusting datacenter networks
- Modelling: How to model workloads, such as ML workloads?
- Dependability: Self-correcting MPLS networks
- More use cases for self-driving networks

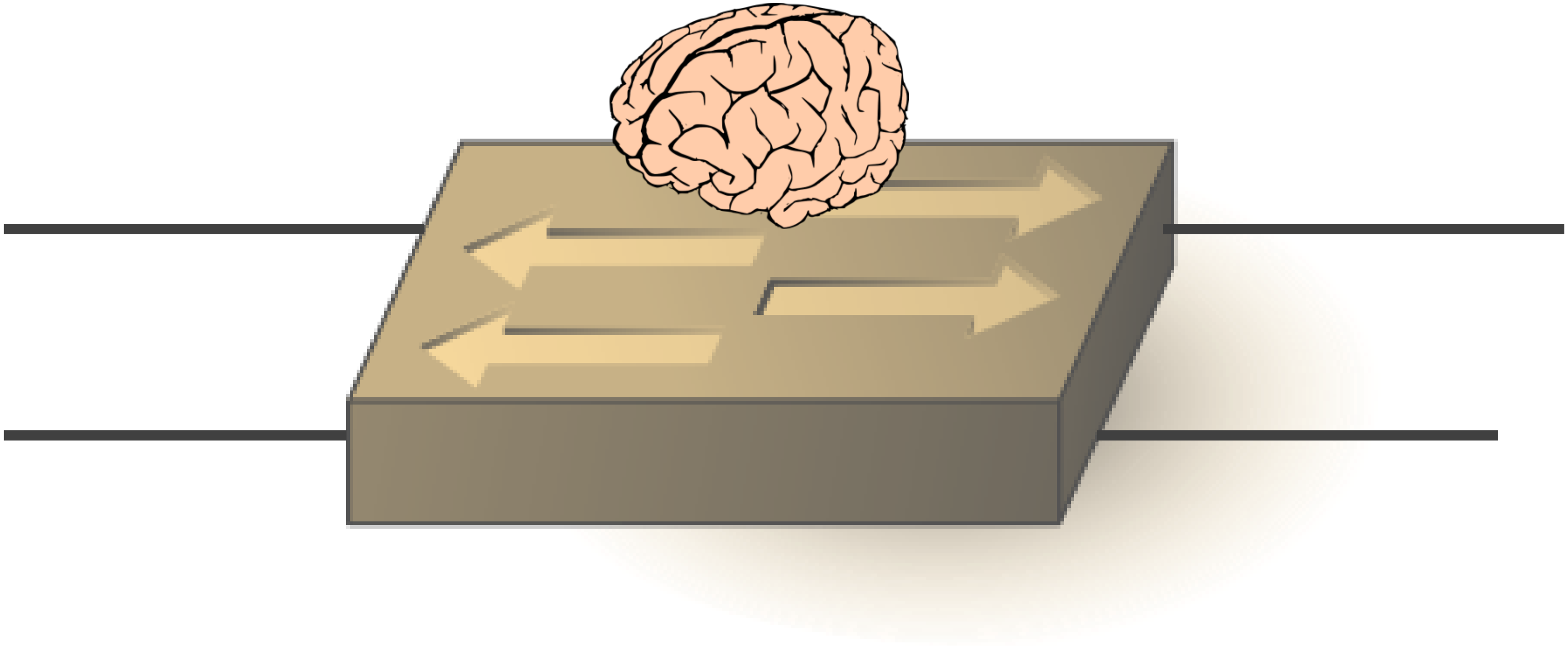
Addanki et al. (NSDI 2024)

Smart Switches

Addanki et al. (NSDI 2024)

Smart Switches

⇢ What if switches become smart?



Scenario 1

- What if switches become smart? Assume: shared memory size 3.

Scenario 1

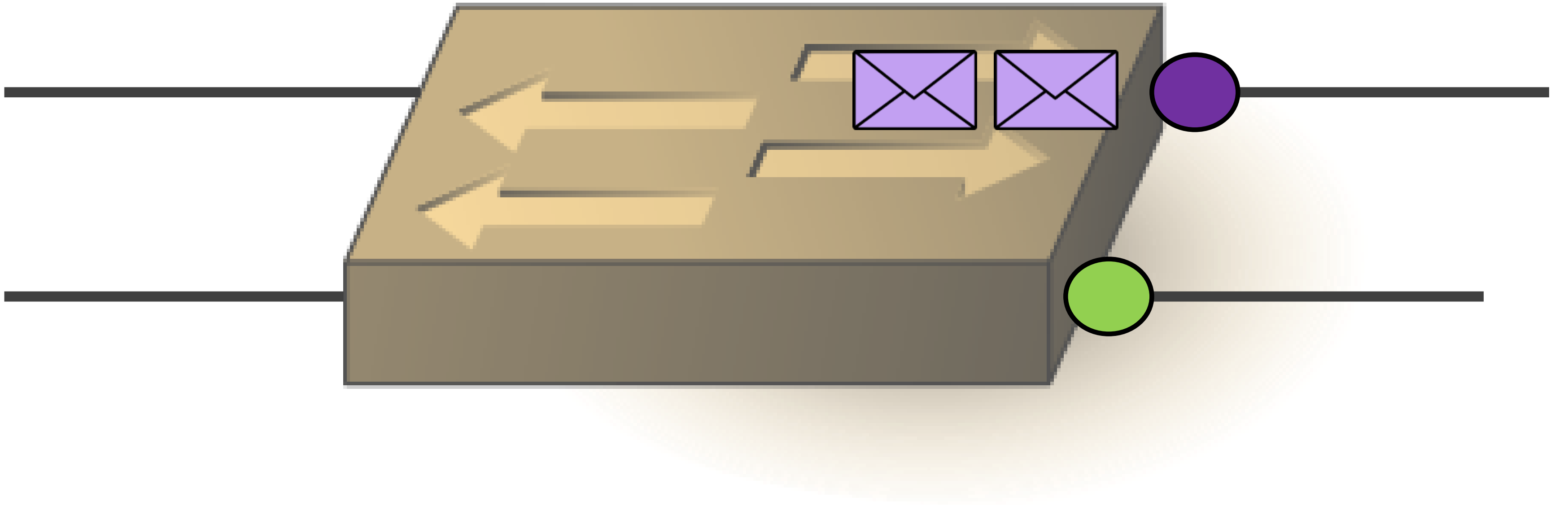
⇢ What if switches become smart? Assume: shared memory size 3.

Scenario 1

⇢ What if switches become smart? Assume: shared memory size 3.

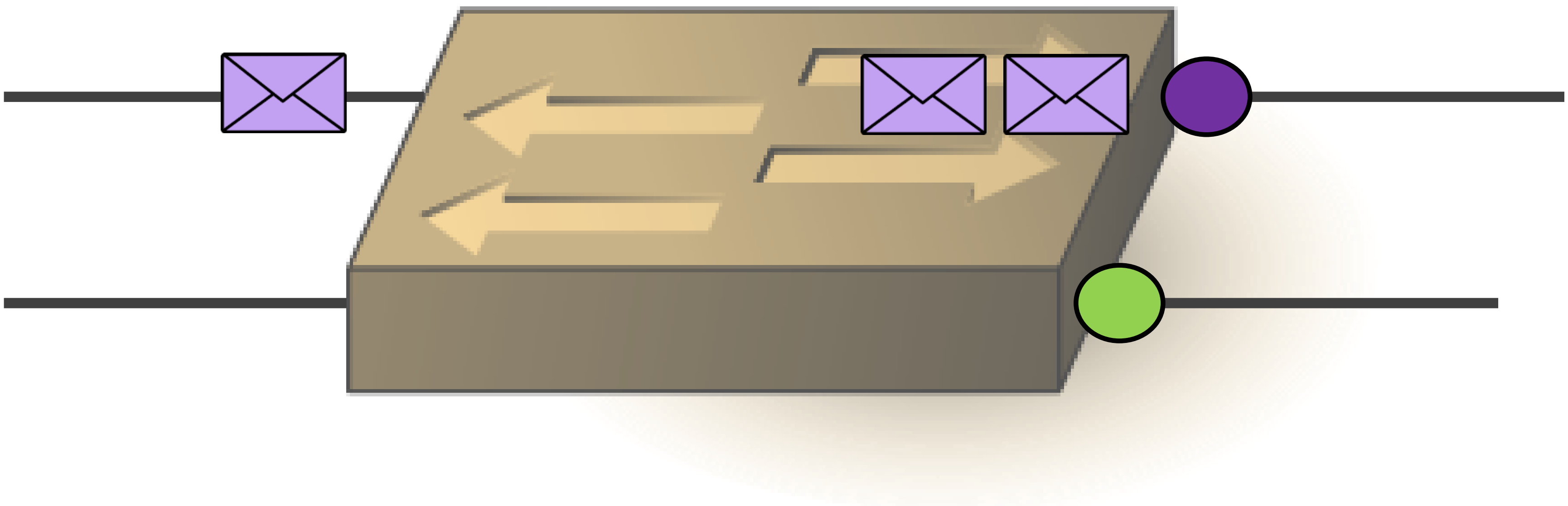
Scenario 1

⇢ What if switches become smart? Assume: shared memory size 3.



Scenario 1

⇢ What if switches become smart? Assume: shared memory size 3.



Scenario 1

- What if switches become smart? Assume: shared memory size 3.

Scenario 1

- What if switches become smart? Assume: shared memory size 3.

Scenario 1

⇢ What if switches become smart? Assume: shared memory size 3.

⇢ Suboptimal: green packets could be transmitted in parallel, but there is no more space! (Output rate 1 vs 2!)

Scenario 2

- What if switches become smart? Assume: shared memory size 3.

Scenario 2

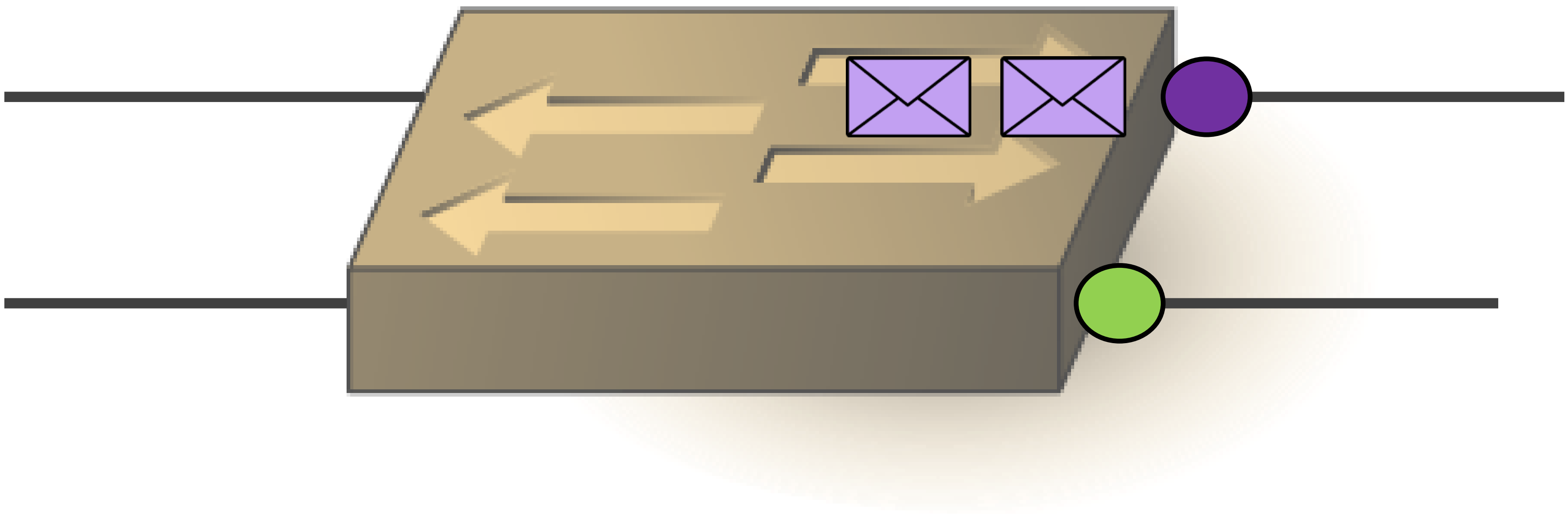
⇢ What if switches become smart? Assume: shared memory size 3.

Scenario 2

⇢ What if switches become smart? Assume: shared memory size 3.

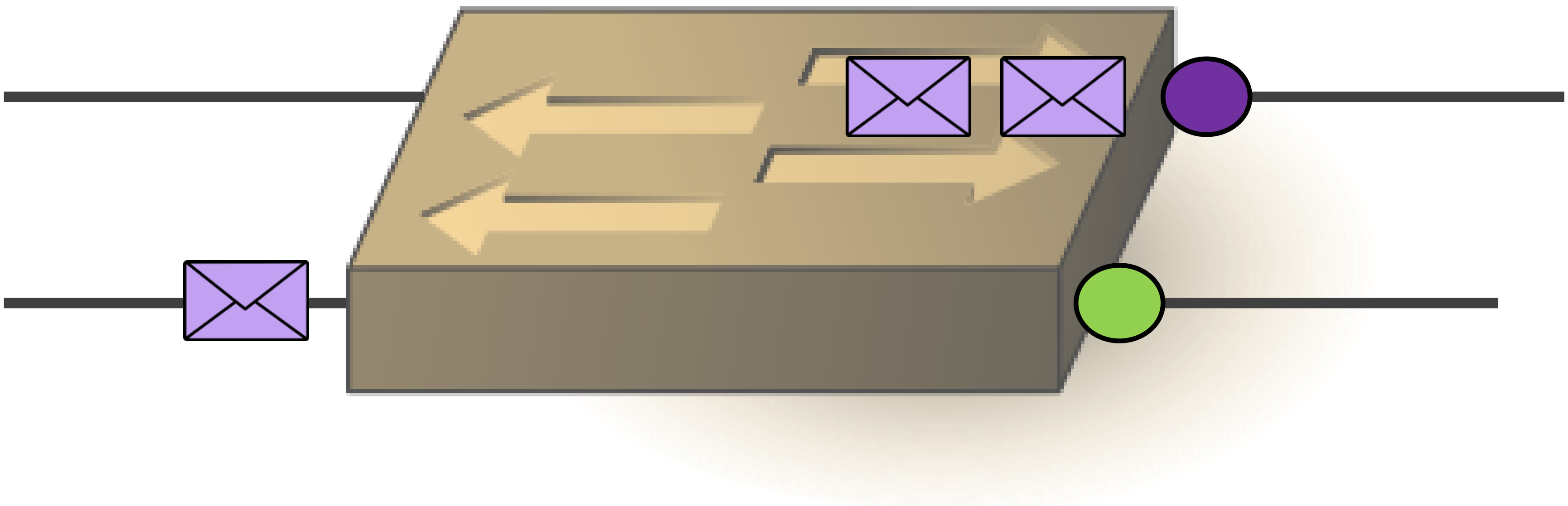
Scenario 2

- What if switches become smart? Assume: shared memory size 3.



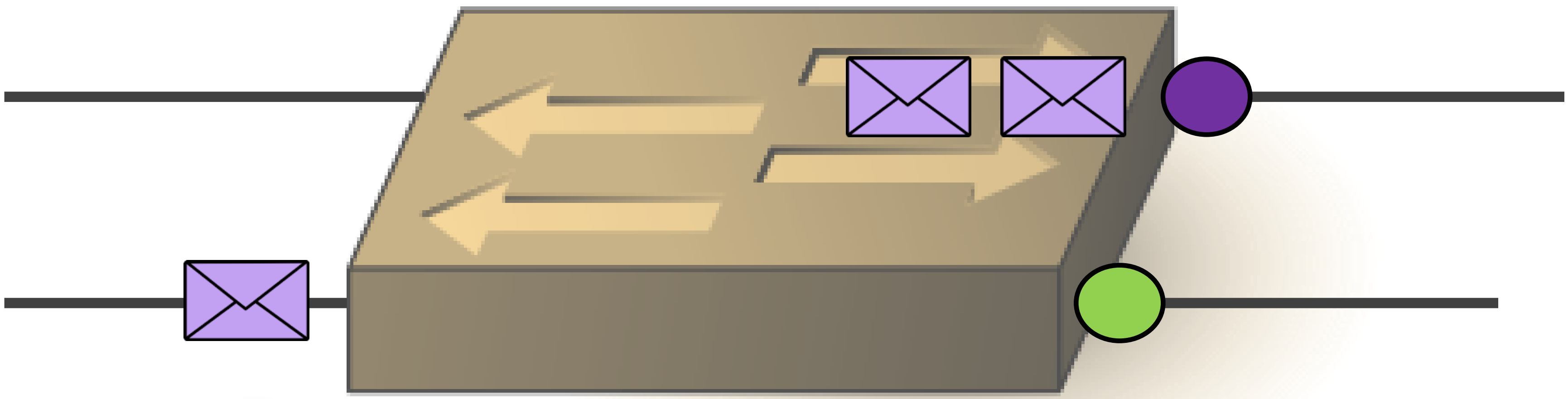
Scenario 2

⇢ What if switches become smart? Assume: shared memory size 3.



Scenario 2

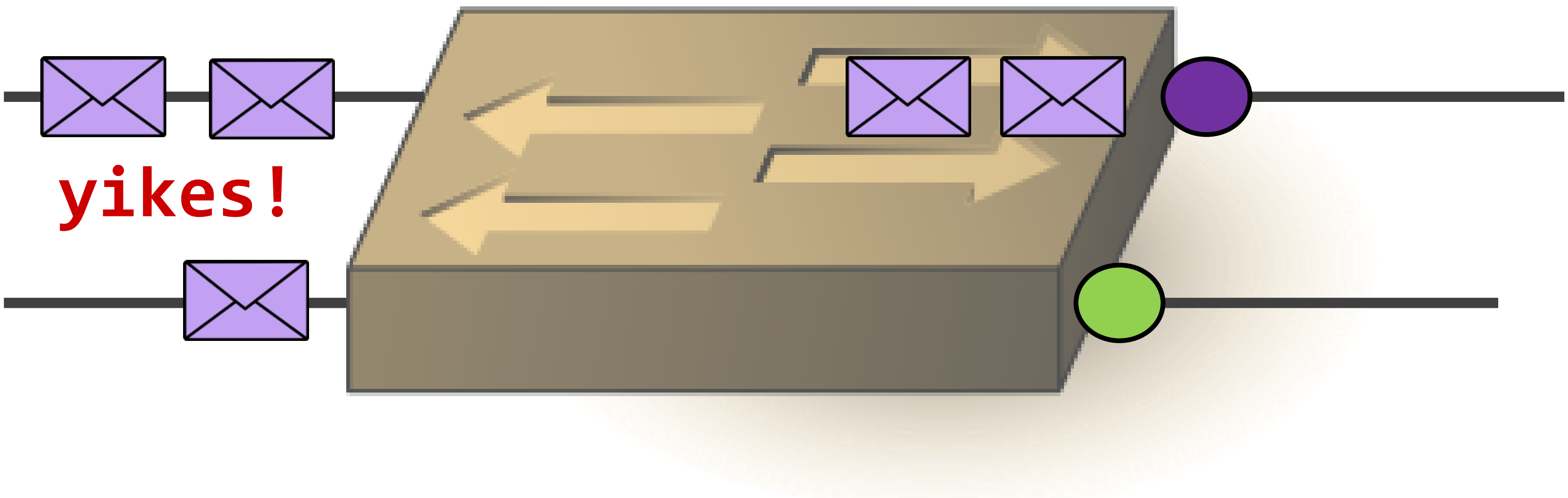
⇢ What if switches become smart? Assume: shared memory size 3.



Idea: keep space for green!

Scenario 2

⇢ What if switches become smart? Assume: shared memory size 3.



⇢ Suboptimal: drop to leave space but no space needed!

Addanki et al. (NSDI 2024)

Credence

- ⇢ Traffic at switch can be ***predicted*** fairly well
- ⇢ AI/ML could significantly ***improve buffer management***…
- ⇢ … and hence ***admission control and throughput***!

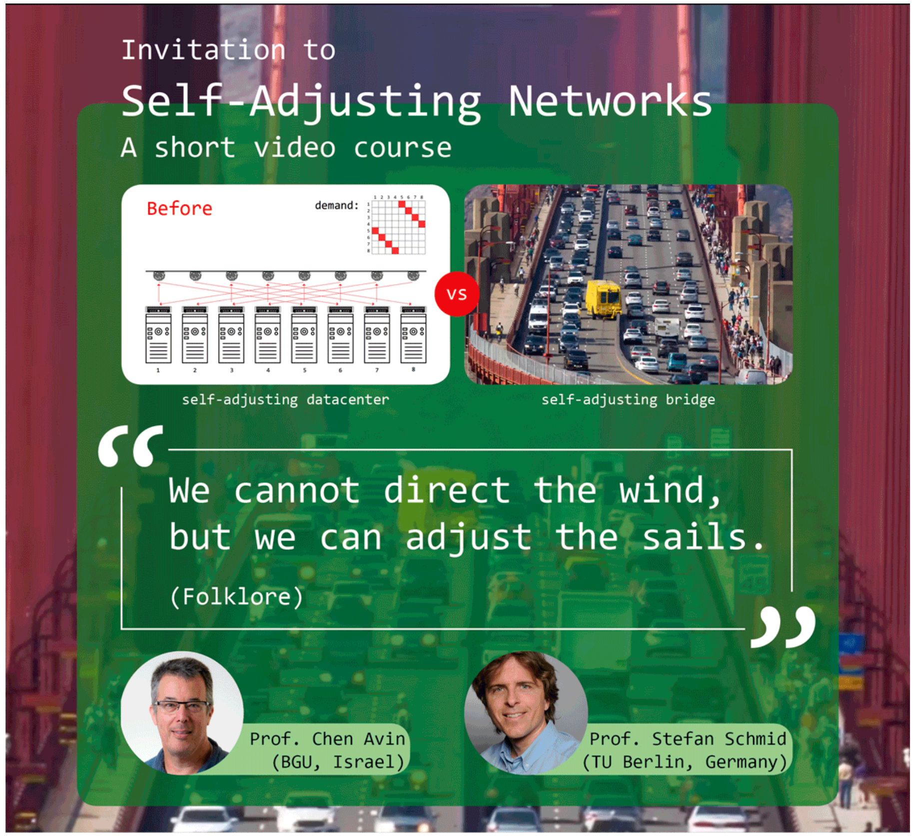
Further reading:

Credence: Augmenting Datacenter Switch Buffer Sharing with ML Predictions
Vamsi Addanki, Maciej Pacut, and Stefan Schmid.
21st USENIX Symposium on Networked Systems Design and Implementation (**NSDI**), 2024.

Conclusion

- Opportunity: *structure* in demand and *reconfigurable* networks
- Enables self-driving networks
- Just the tip of the iceberg!
  - Optimal *collaboration* of ML, FM, and “human in the loop”?
  - Impact of self-driving layer on *other layers*?
  - *Scalable control* plane?
  - *Application-specific* self-adjusting networks, e.g., for LLMs?

Online Video Course



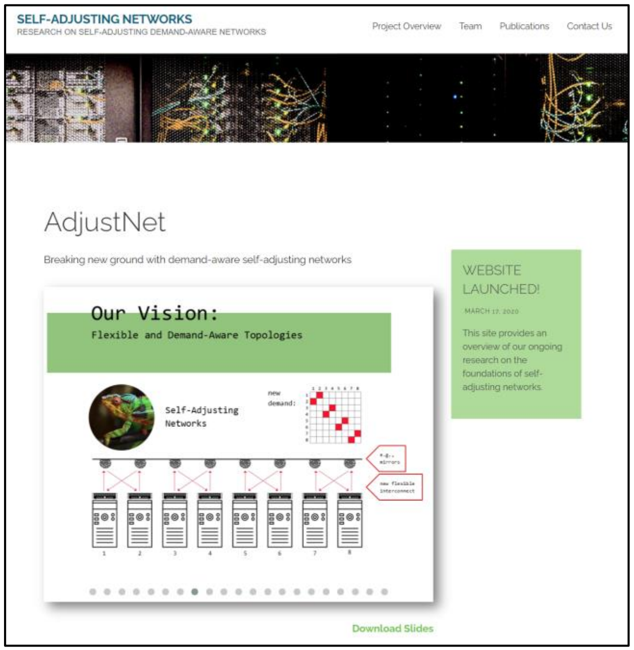
https://self-adjusting.net/course

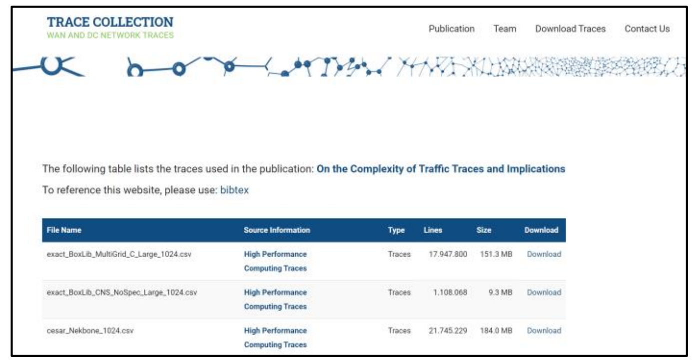
YouTube Interview & CACM

Check out our **YouTube interviews**
on Reconfigurable Datacenter Networks:

[Revolutionizing Datacenter Networks via Reconfigurable Topologies](https://www.youtube.com/@self-adjusting-networks-course)
Chen Avin and Stefan Schmid.
Communications of the ACM (**CACM**), 2025.
Watch here: https://www.youtube.com/@self-adjusting-networks-course

Websites





https://trace-collection.net/
Trace collection website

http://self-adjusting.net/
Project website

Upcoming CACM Article

Revolutionizing Datacenter Networks via Reconfigurable Topologies

CHEN AVIN, is a Professor at Ben-Gurion University of the Negev, Beersheva, Israel
STEFAN SCHMID, is a Professor at TU Berlin, Berlin, Germany

With the popularity of cloud computing and data-intensive applications such as machine learning, datacenter networks have become a critical infrastructure for our digital society. Given the explosive growth of datacenter traffic and the slowdown of Moore's law, significant efforts have been made to improve datacenter network performance over the last decade. A particularly innovative solution is reconfigurable datacenter networks (RDCNs): datacenter networks whose topologies dynamically change over time, in either a demand-oblivious or a demand-aware manner. Such dynamic topologies are enabled by recent optical switching technologies and stand in stark contrast to state-of-the-art datacenter network topologies, which are fixed and oblivious to the actual traffic demand. In particular, reconfigurable demand-aware and "self-adjusting" datacenter networks are motivated empirically by the significant spatial and temporal structures observed in datacenter communication traffic. This paper presents an overview of reconfigurable datacenter networks. In particular, we discuss the motivation for such reconfigurable architectures, review the technological enablers, and present a taxonomy that classifies the design space into two dimensions: static vs. dynamic and demand-oblivious vs. demand-aware. We further present a formal model and discuss related research challenges. Our article comes with complementary video interviews in which three leading experts, Manya Ghobadi, Amin Vahdat, and George Papen, share with us their perspectives on reconfigurable datacenter networks.

KEY INSIGHTS

- **Datacenter networks have become a critical infrastructure** for our digital society, serving explosively growing communication traffic.
- **Reconfigurable datacenter networks (RDCNs)** which can adapt their topology dynamically, based on innovative **optical switching technologies**, bear the potential to improve datacenter network performance, and to simplify datacenter planning and operations.
- Demand-aware dynamic topologies are particularly interesting because of the **significant spatial and temporal structures** observed in real-world traffic, e.g., related to distributed machine learning.
- The study of RDCNs and self-adjusting networks raises many **novel technological and research challenges** related to their design, control, and performance.

References (1)

On the Complexity of Traffic Traces and Implications
Chen Avin, Manya Ghobadi, Chen Griner, and Stefan Schmid.
ACM **SIGMETRICS** and ACM Performance Evaluation Review (**PER**), Boston, Massachusetts, USA, June 2020.

Toward Demand-Aware Networking: A Theory for Self-Adjusting Networks (Editorial)
Chen Avin and Stefan Schmid.
ACM SIGCOMM Computer Communication Review (**CCR**), October 2018.

Revolutionizing Datacenter Networks via Reconfigurable Topologies
Chen Avin and Stefan Schmid.
Communications of the ACM (**CACM**), 2025.

Cerberus: The Power of Choices in Datacenter Topology Design (A Throughput Perspective)
Chen Griner, Johannes Zerwas, Andreas Blenk, Manya Ghobadi, Stefan Schmid, and Chen Avin.
ACM **SIGMETRICS** and ACM Performance Evaluation Review (**PER**), Mumbai, India, June 2022.

AalWiNes: A Fast and Quantitative What-If Analysis Tool for MPLS Networks
Peter Gjøl Jensen, Morten Konggaard, Dan Kristiansen, Stefan Schmid, Bernhard Clemens Schrenk, and Jiri Srba.
16th ACM International Conference on emerging Networking EXperiments and Technologies (**CoNEXT**), Barcelona, Spain, December 2020.

Latte: Improving the Latency of Transiently Consistent Network Update Schedules
Mark Glavind, Niels Christensen, Jiri Srba, and Stefan Schmid.
38th International Symposium on Computer Performance, Modeling, Measurements and Evaluation (**PERFORMANCE**) and ACM Performance Evaluation Review (**PER**), Milan, Italy, November 2020.

Model-Based Insights on the Performance, Fairness, and Stability of BBR (IRTF Applied Networking Research Prize)
Simon Scherrer, Markus Legner, Adrian Perrig, and Stefan Schmid.
ACM Internet Measurement Conference (**IMC**), Nice, France, October 2022.

Credence: Augmenting Datacenter Switch Buffer Sharing with ML Predictions
Vamsi Addanki, Maciej Pacut, and Stefan Schmid.
21st USENIX Symposium on Networked Systems Design and Implementation (**NSDI**), Santa Clara, California, USA, April 2024.

References (2)

Mars: Near-Optimal Throughput with Shallow Buffers in Reconfigurable Datacenter Networks
Vamsi Addanki, Chen Avin, and Stefan Schmid.
ACM **SIGMETRICS** and ACM Performance Evaluation Review (**PER**), Orlando, Florida, USA, June 2023.

Duo: A High-Throughput Reconfigurable Datacenter Network Using Local Routing and Control
Johannes Zerwas, Csaba Györgyi, Andreas Blenk, Stefan Schmid, and Chen Avin.
ACM **SIGMETRICS** and ACM Performance Evaluation Review (**PER**), Orlando, Florida, USA, June 2023.

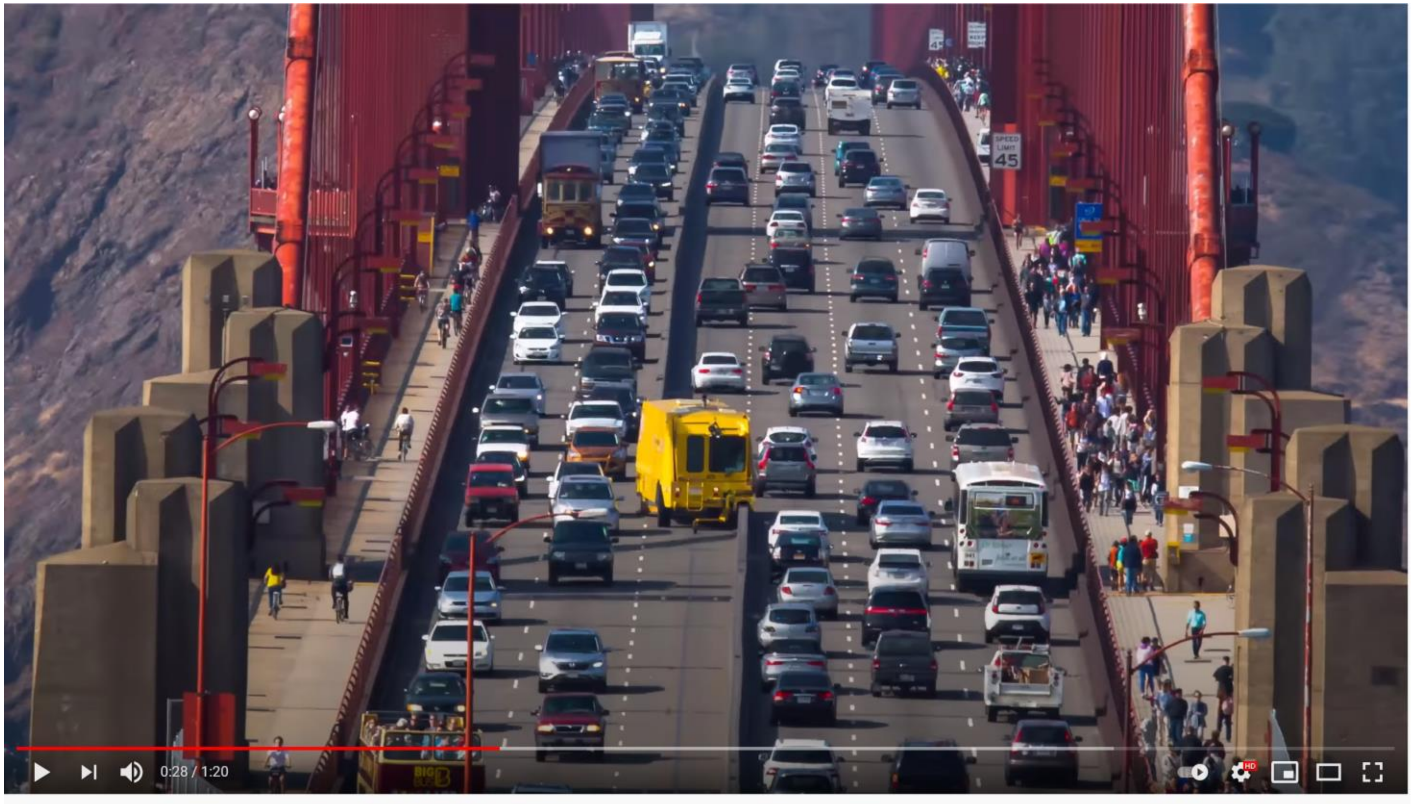
SyPer: Synthesis of Perfectly Resilient Local Fast Rerouting Rules for Highly Dependable Networks
Csaba Györgyi, Kim G. Larsen, Stefan Schmid, and Jiri Srba.
IEEE Conference on Computer Communications (**INFOCOM**), Vancouver, Canada, May 2024.

Demand-Aware Network Design with Minimal Congestion and Route Lengths
Chen Avin, Kaushik Mondal, and Stefan Schmid.
IEEE/ACM Transactions on Networking (**TON**), 2022.

A Survey of Reconfigurable Optical Networks
Matthew Nance Hall, Klaus-Tycho Foerster, Stefan Schmid, and Ramakrishnan Durairajan.
Optical Switching and Networking (**OSN**), Elsevier, 2021.

SplayNet: Towards Locally Self-Adjusting Networks
Stefan Schmid, Chen Avin, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler, and Zvi Lotker.
IEEE/ACM Transactions on Networking (**TON**), Volume 24, Issue 3, 2016.
.

Questions?



Slides
available
here: