Self-Adjusting Networks

The Power of Choices in Datacenter Topology Design

Stefan Schmid
(kudos to Chen Avin)

> “We cannot direct the wind,
> but we can adjust the sails.”

(Folklore)

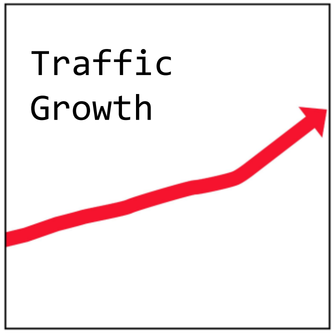
Acknowledgements:

Trend

Data-Centric Applications

Datacenters (“hyper-scale”)

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



Source: Facebook

The Problem

Huge Infrastructure, Inefficient Use

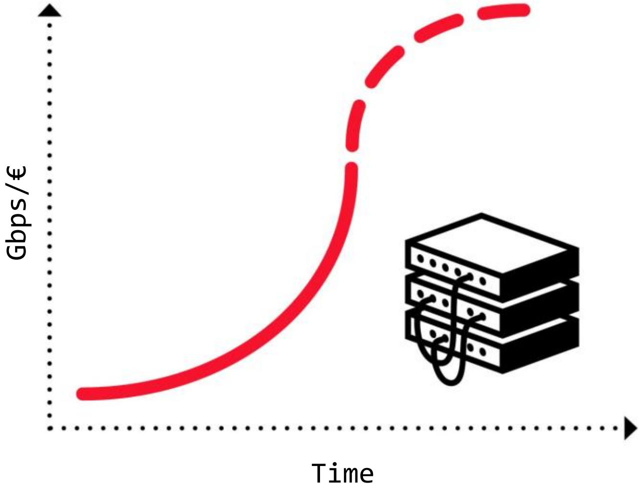
⇢ Network equipment reaching capacity limits

⇀ Transistor density rates stalling

⇀ “End of **Moore‘s Law** in networking” [1]

⇢ Hence: more equipment, larger networks

⇢ Resource intensive and: **inefficient**



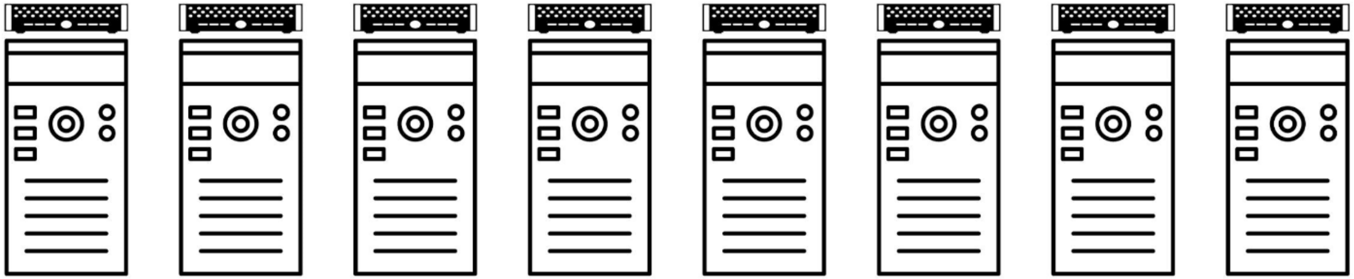
[1] Source: Microsoft, 2019

Annoying for companies, **opportunity** for researchers

A Root Cause

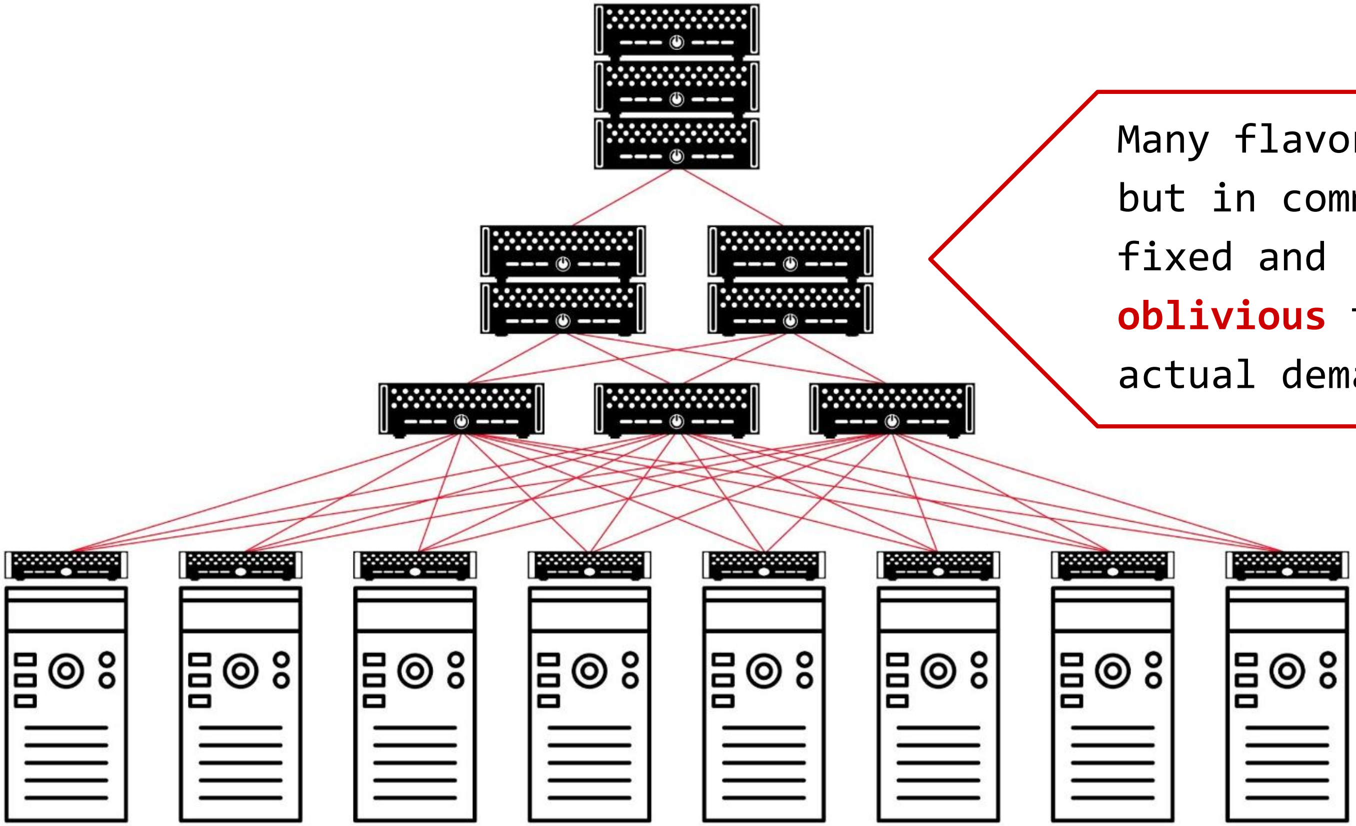
Demand-Oblivious Topology

How to interconnect?



A Root Cause

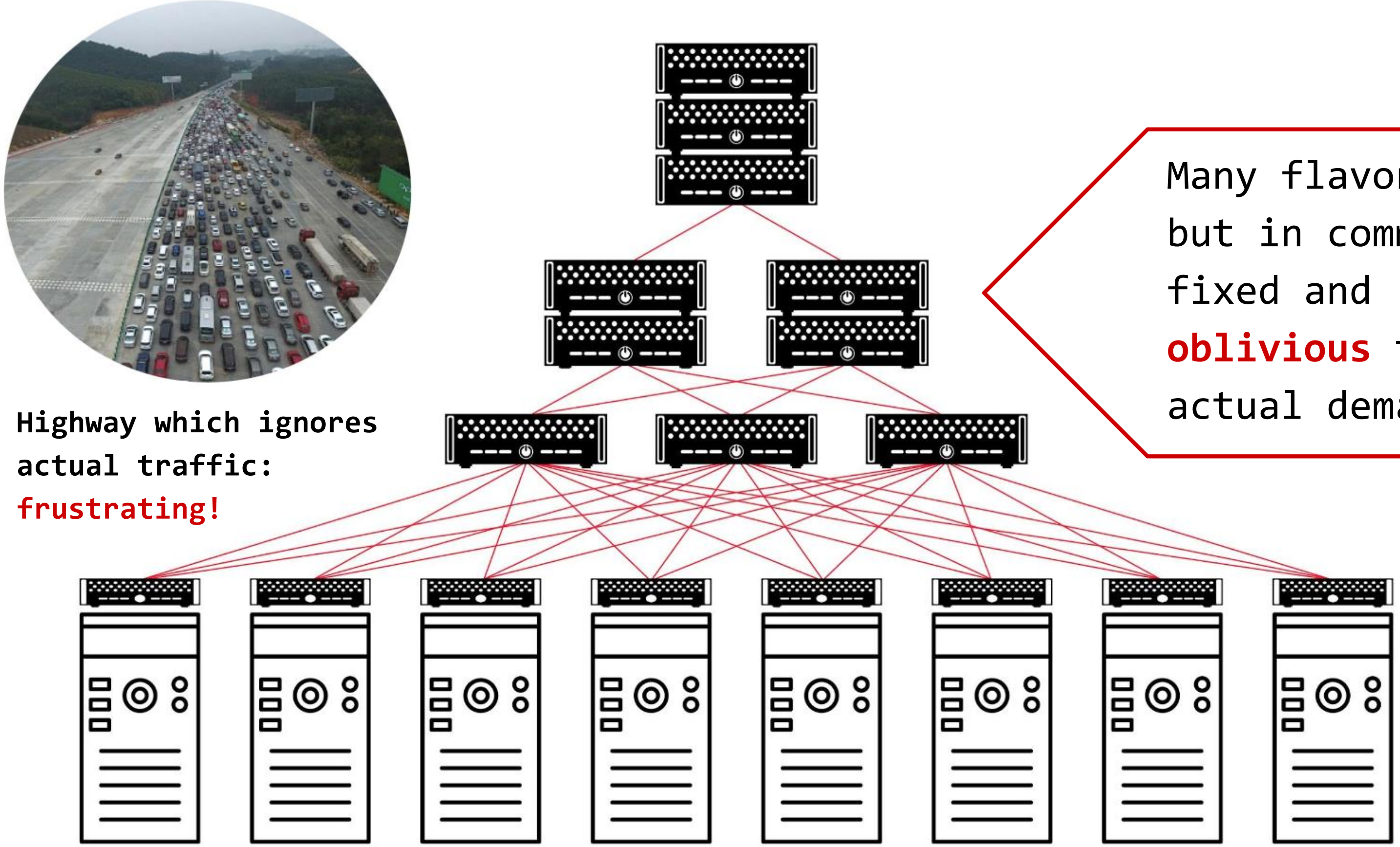
Demand-Oblivious Topology



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

A Root Cause

Demand-Oblivious Topology



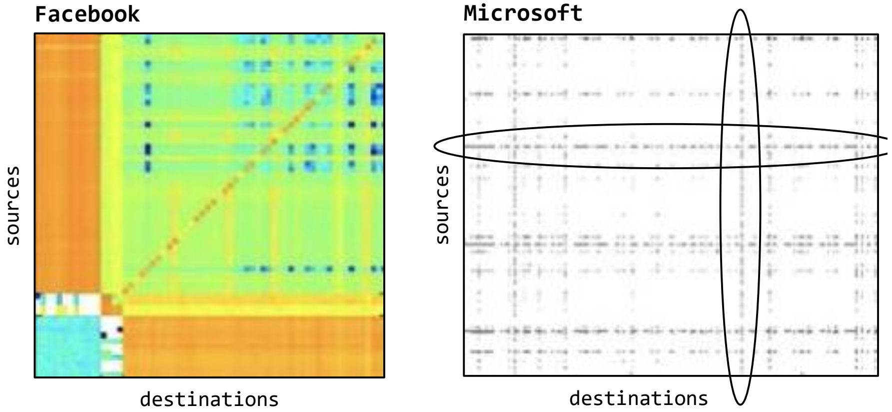
Highway which ignores actual traffic: frustrating!

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

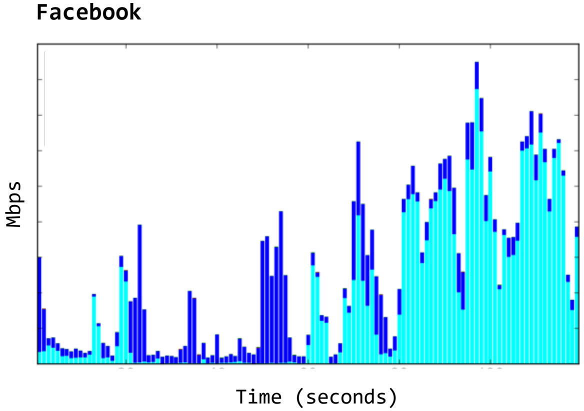
Empirical Motivation

Traffic does not only grow but also has much structure:

traffic matrices sparse and skewed

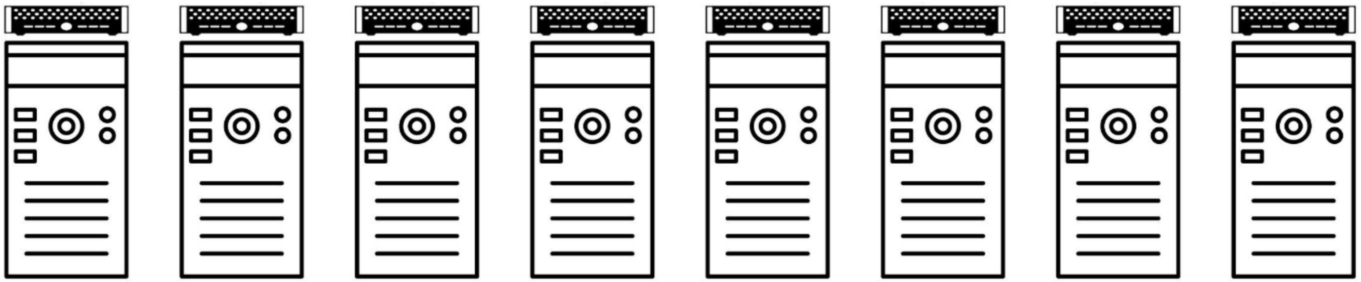


traffic bursty over time

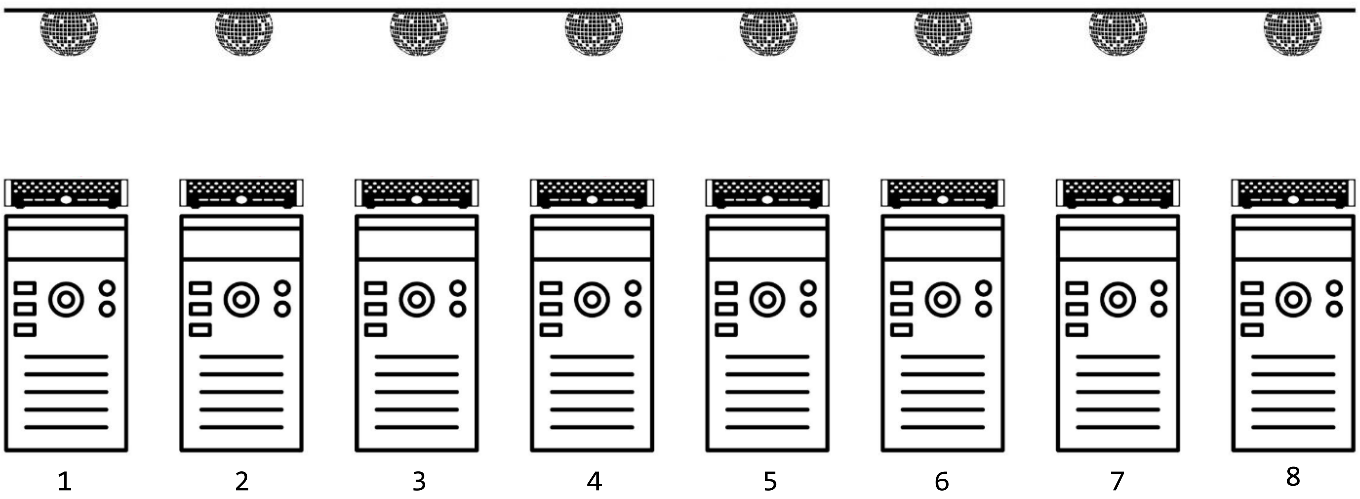


My hypothesis: can be exploited.

The Vision of Demand-Aware Topologies



The Vision of Demand-Aware Topologies



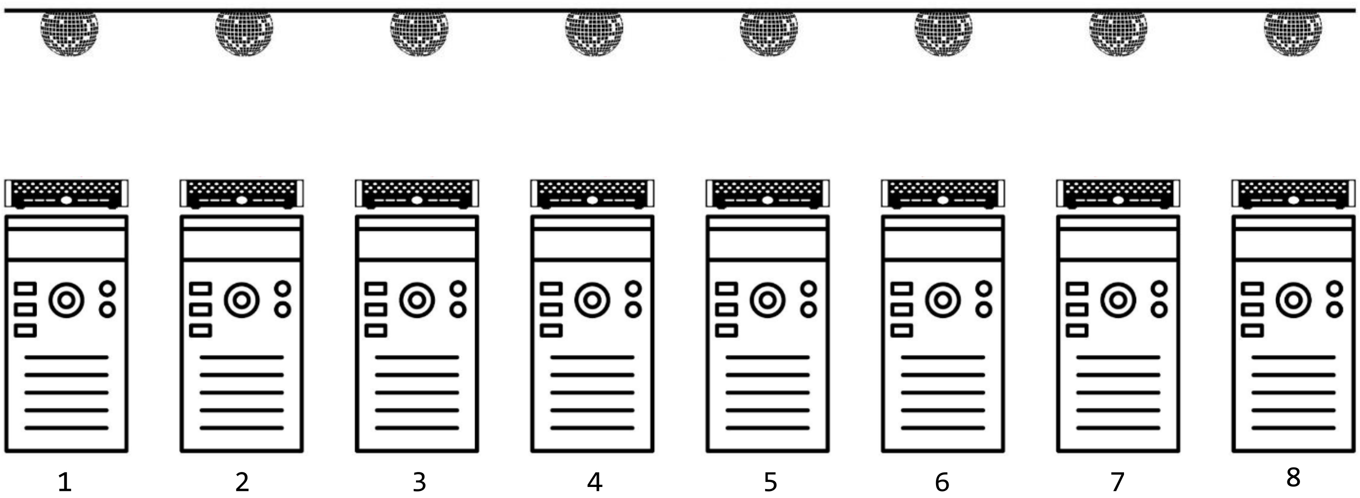
e.g., mirrors

new flexible interconnect

The Vision of Demand-Aware Topologies

demand matrix:

	1	2	3	4	5	6	7	8
1					🟥			
2						🟥		
3							🟥	
4								🟥
5	🟥							
6		🟥						
7			🟥					
8				🟥				



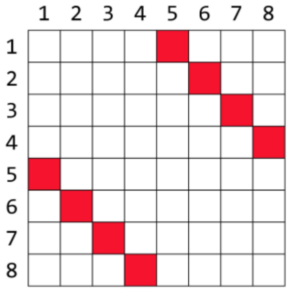
e.g., mirrors

new flexible interconnect

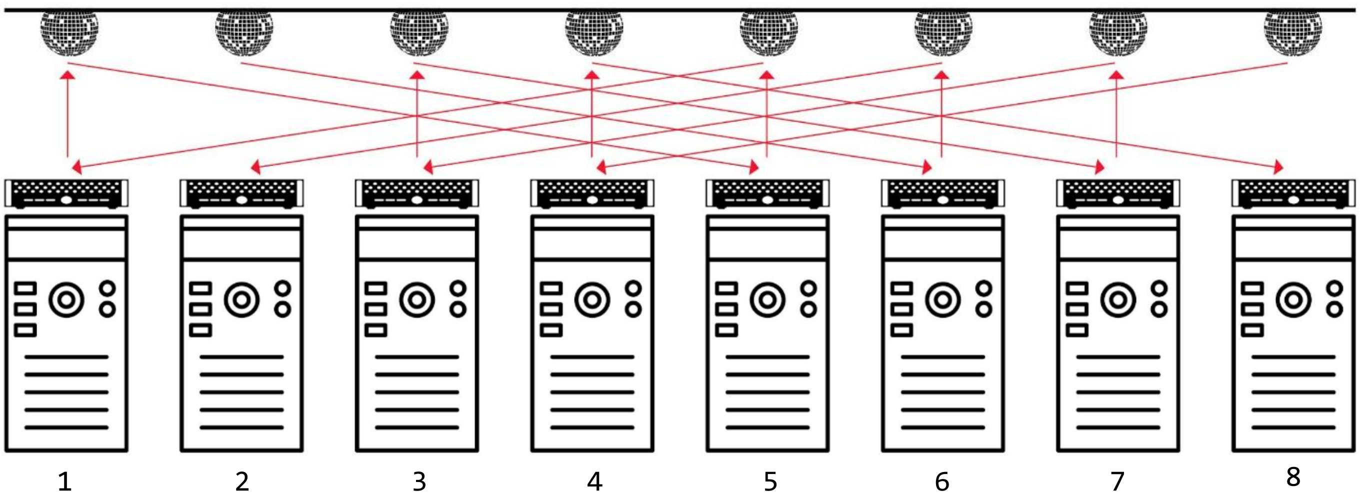
The Vision of Demand-Aware Topologies

Matches demand

demand matrix:



	1	2	3	4	5	6	7	8
1					🟥			
2						🟥		
3							🟥	
4								🟥
5	🟥							
6		🟥						
7			🟥					
8				🟥				

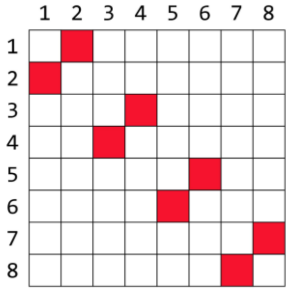


e.g., mirrors

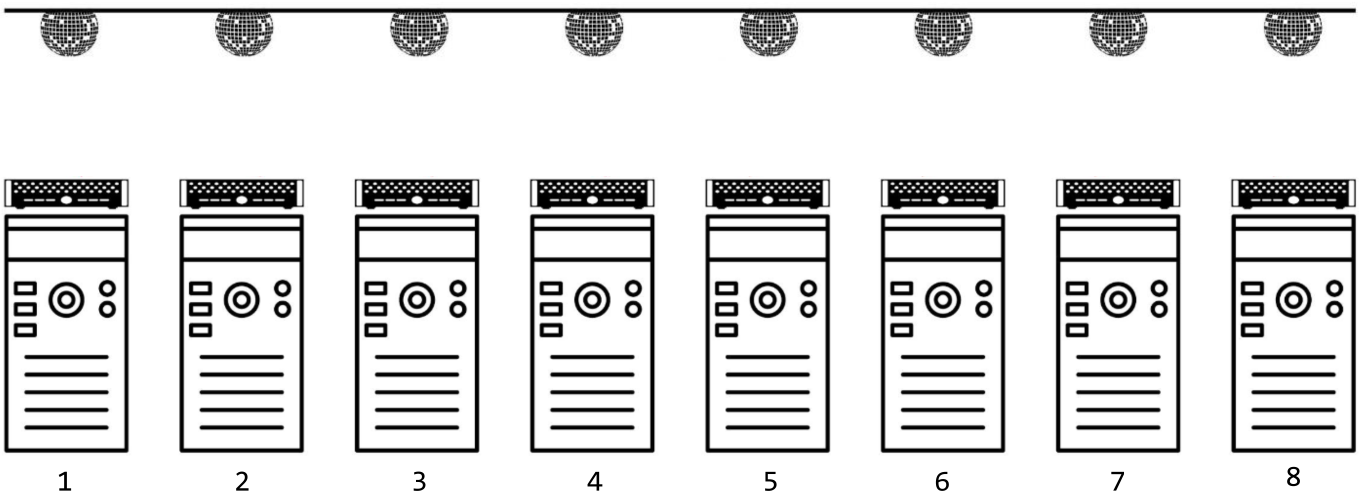
new flexible interconnect

The Vision of Demand-Aware Topologies

new demand:



	1	2	3	4	5	6	7	8
1		🟥						
2	🟥							
3				🟥				
4			🟥					
5						🟥		
6					🟥			
7								🟥
8							🟥	



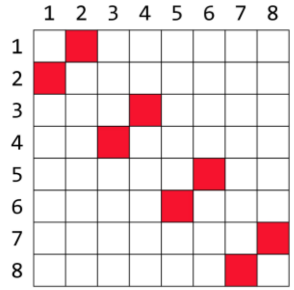
e.g., mirrors

new flexible interconnect

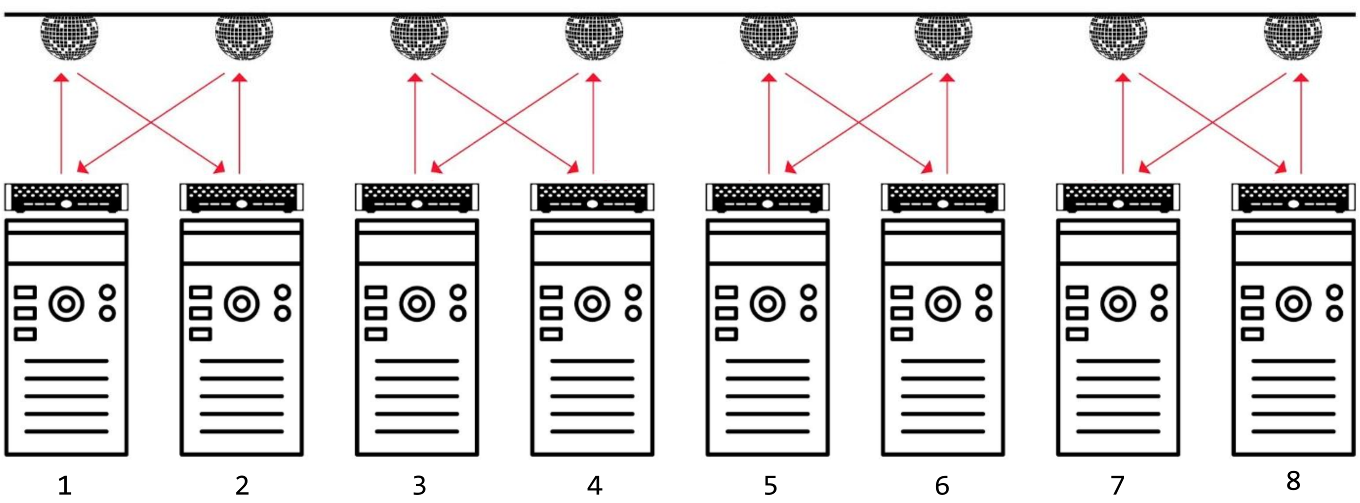
The Vision of Demand-Aware Topologies

Matches demand

new demand:



	1	2	3	4	5	6	7	8
1		🟥						
2	🟥							
3				🟥				
4			🟥					
5						🟥		
6					🟥			
7								🟥
8							🟥	



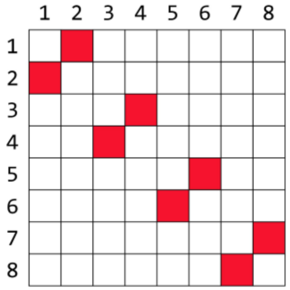
e.g., mirrors

new flexible interconnect

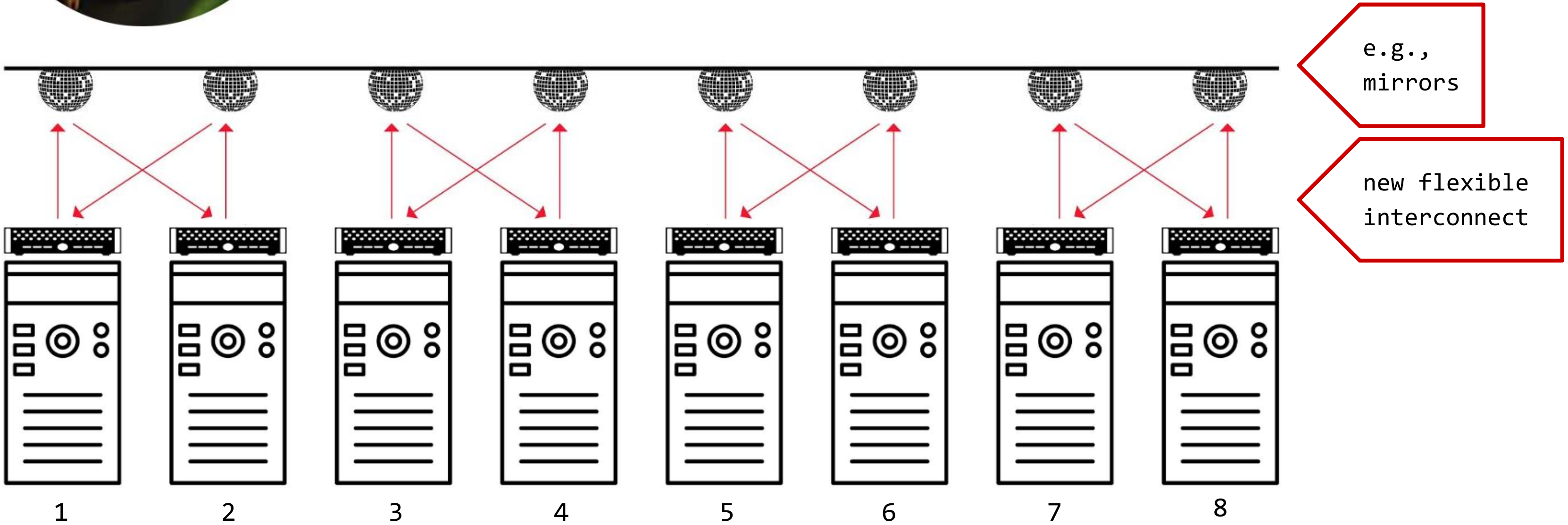
The Vision of Demand-Aware Topologies

Self-Adjusting Networks

new demand:



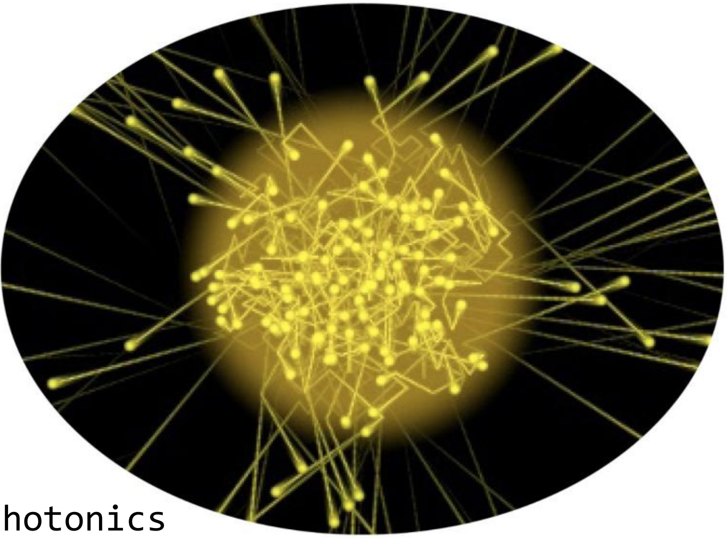
	1	2	3	4	5	6	7	8
1		🟥						
2	🟥							
3				🟥				
4			🟥					
5						🟥		
6					🟥			
7								🟥
8							🟥	



e.g., mirrors

new flexible interconnect

Sounds Crazy? Emerging Enabling Technology.


Photonics

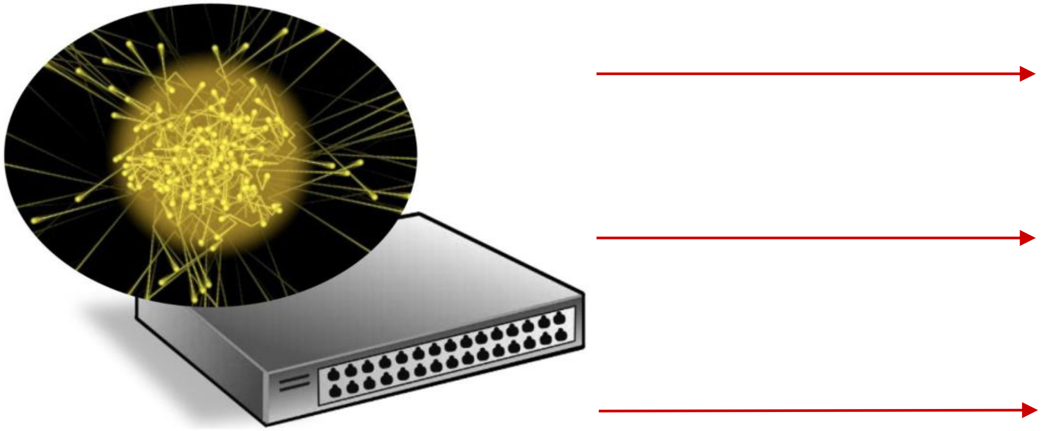
H2020:
“Photonics one of only five key enabling technologies for future prosperity.”

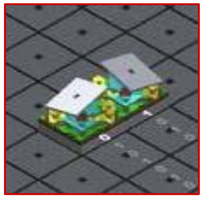
US National Research Council:
“Photons are the new Electrons.”

Enabler

Novel Reconfigurable Optical Switches

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





Prototype 1

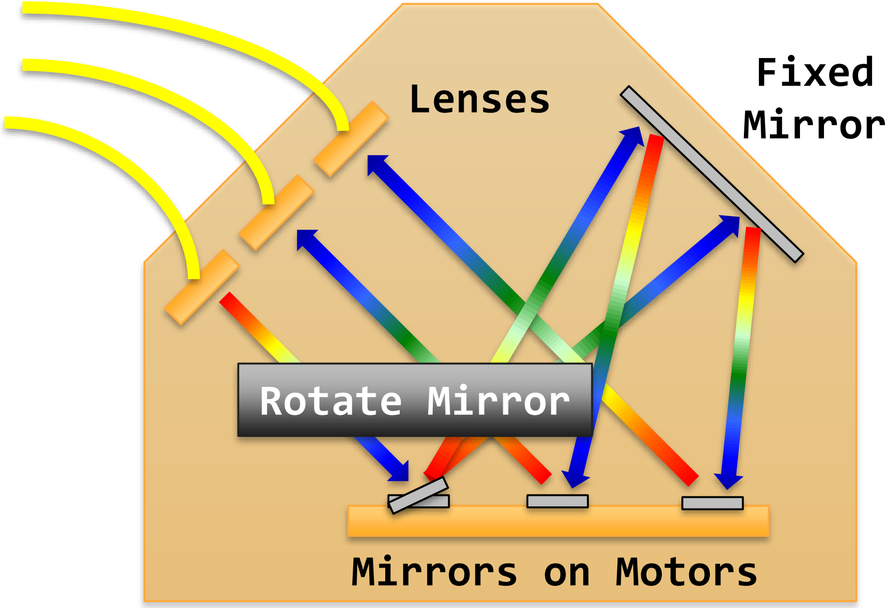
Prototype 2

Prototype 3

Example

Optical Circuit Switch

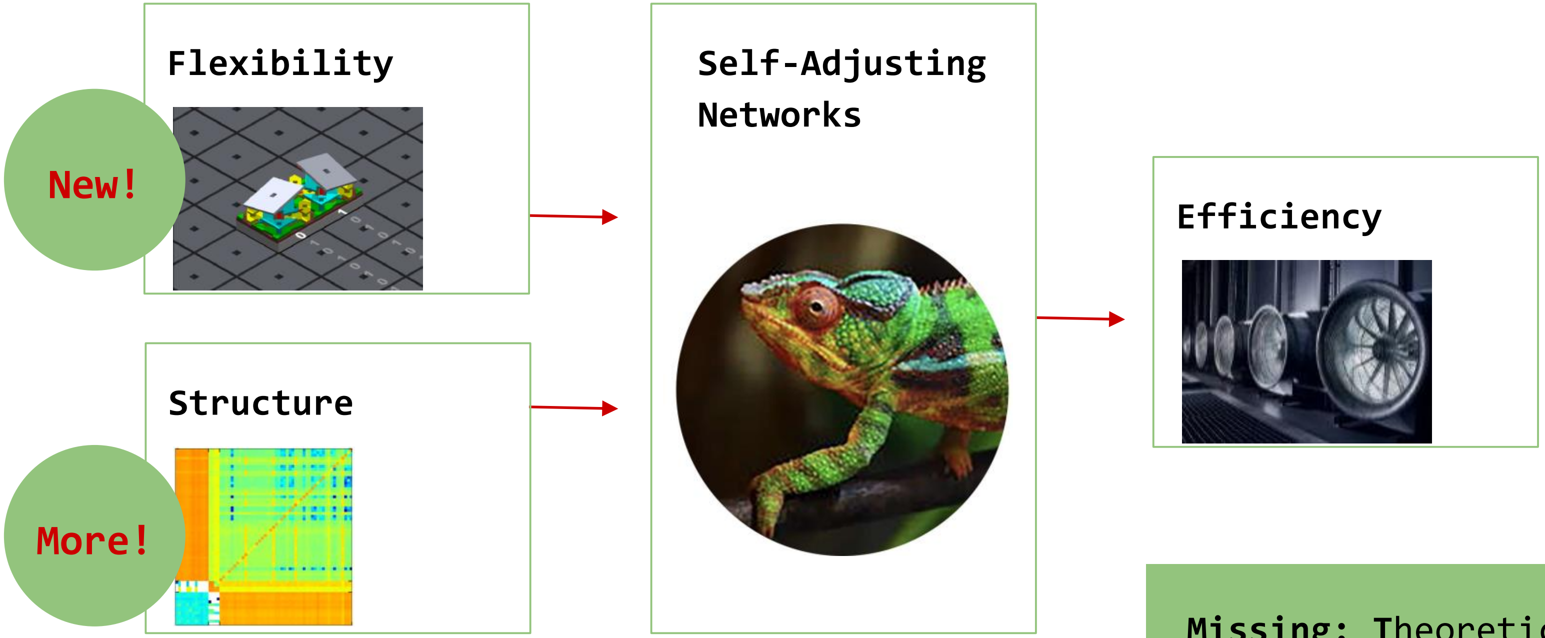
- ⇢ Optical Circuit Switch rapid adaption of physical layer
  - ⇀ Based on rotating mirrors



Optical Circuit Switch

By Nathan Farrington, SIGCOMM 2010

The Big Picture



Flexibility

New!

Structure

More!

Self-Adjusting Networks

Efficiency

Now is the time!

Missing: Theoretical and practical **foundations** of demand-aware, self-adjusting networks.

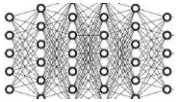
Unique Position

Demand-Aware, Self-Adjusting Systems

Everywhere, but mainly in software

Algorithmic trading

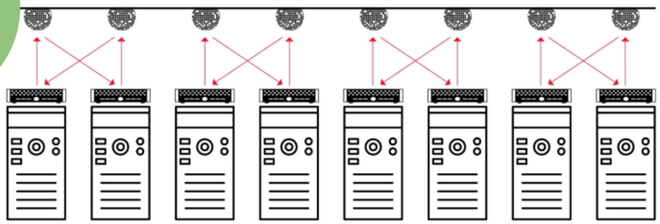
Recommender systems



Neural networks

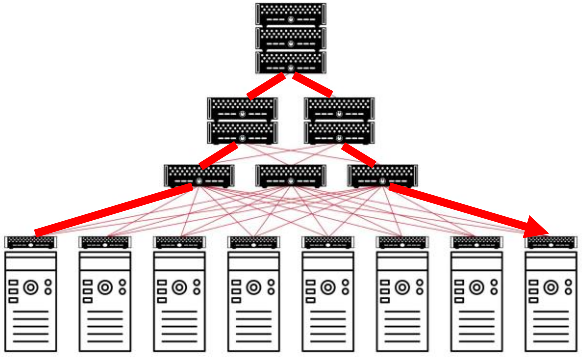
VS

Our focus: in hardware

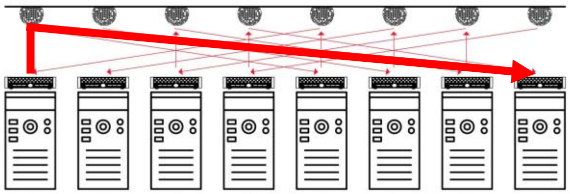


It is more complicated…

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



vs



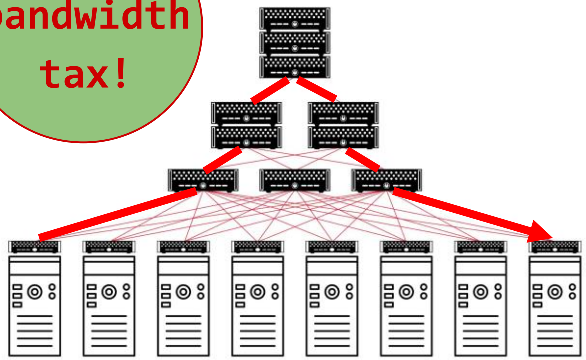
6 hops

1 hop

It is more complicated…

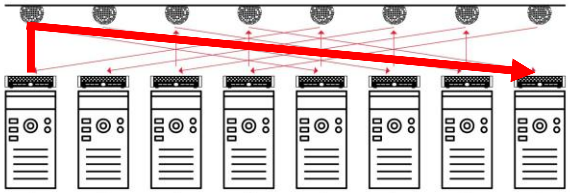
→ Self-adjusting networks may be really useful to serve large flows (elephant flows): avoiding multi-hop routing

bandwidth tax!



6 hops

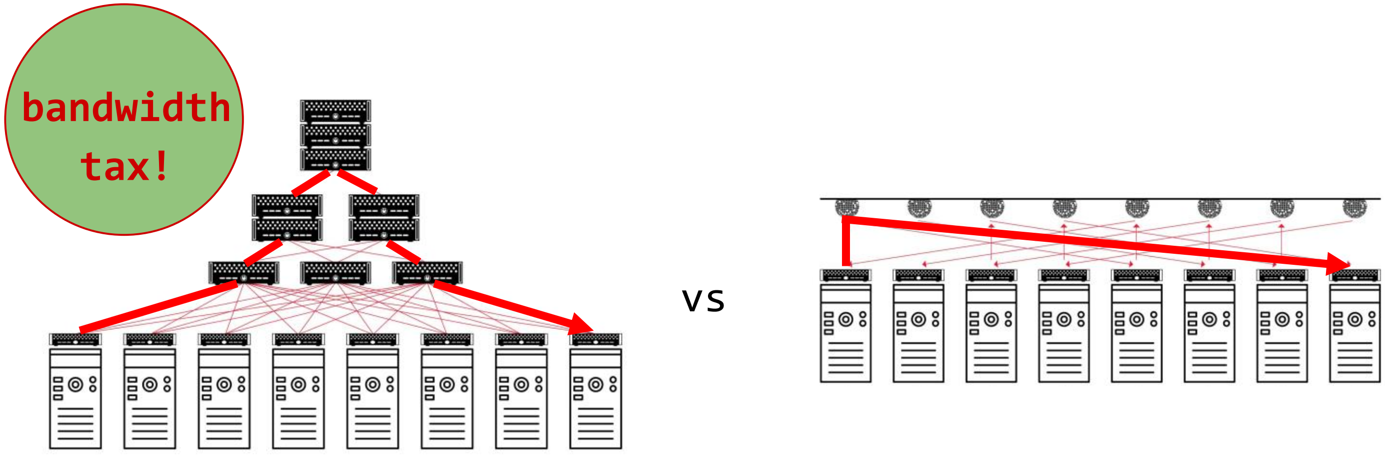
vs



1 hop

It is more complicated…

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

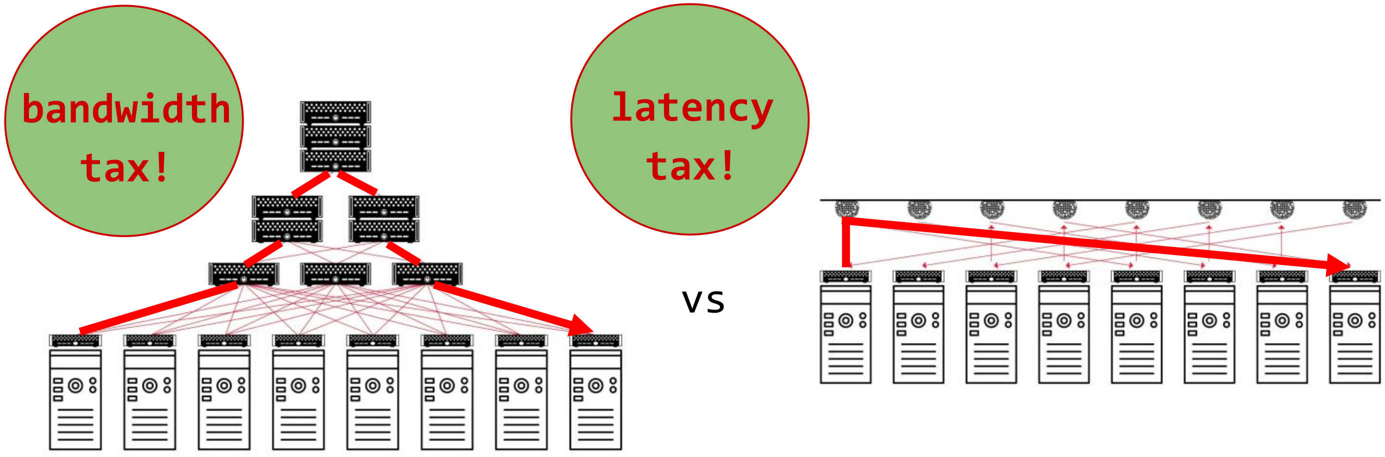


6 hops **1 hop**

⇀ However, requires optimization and adaption, which takes time

It is 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

Shuffling
All-to-All

ML
Large flows

Delay sensitive

Telemetry / control

Opportunity: Tech Diversity

Diverse topology components:

⇀ demand-oblivious and
demand-aware

Demand-
oblivious

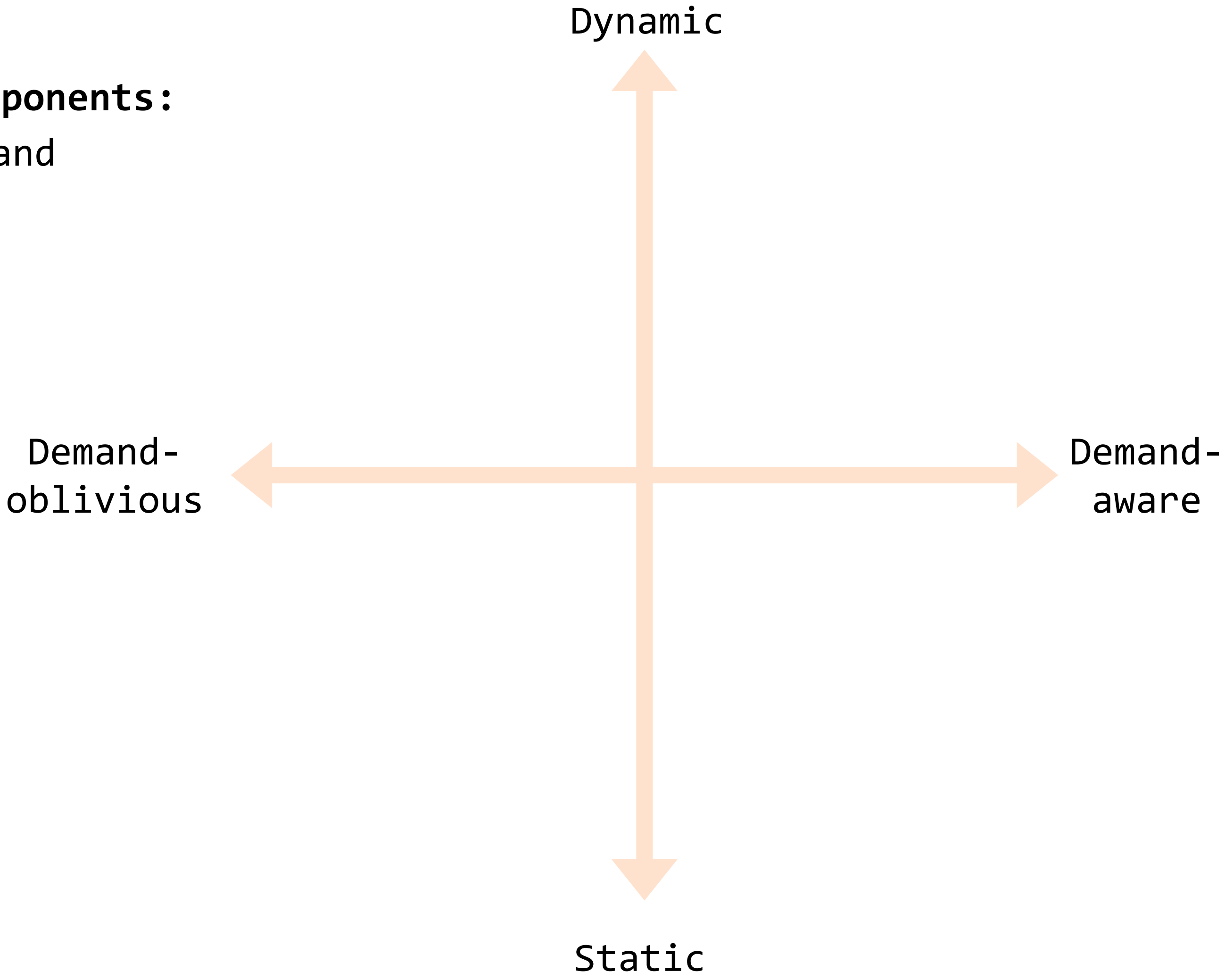
Demand-
aware

Opportunity: Tech Diversity

Diverse topology components:

⇀ demand-oblivious and
   demand-aware

⇀ static vs dynamic

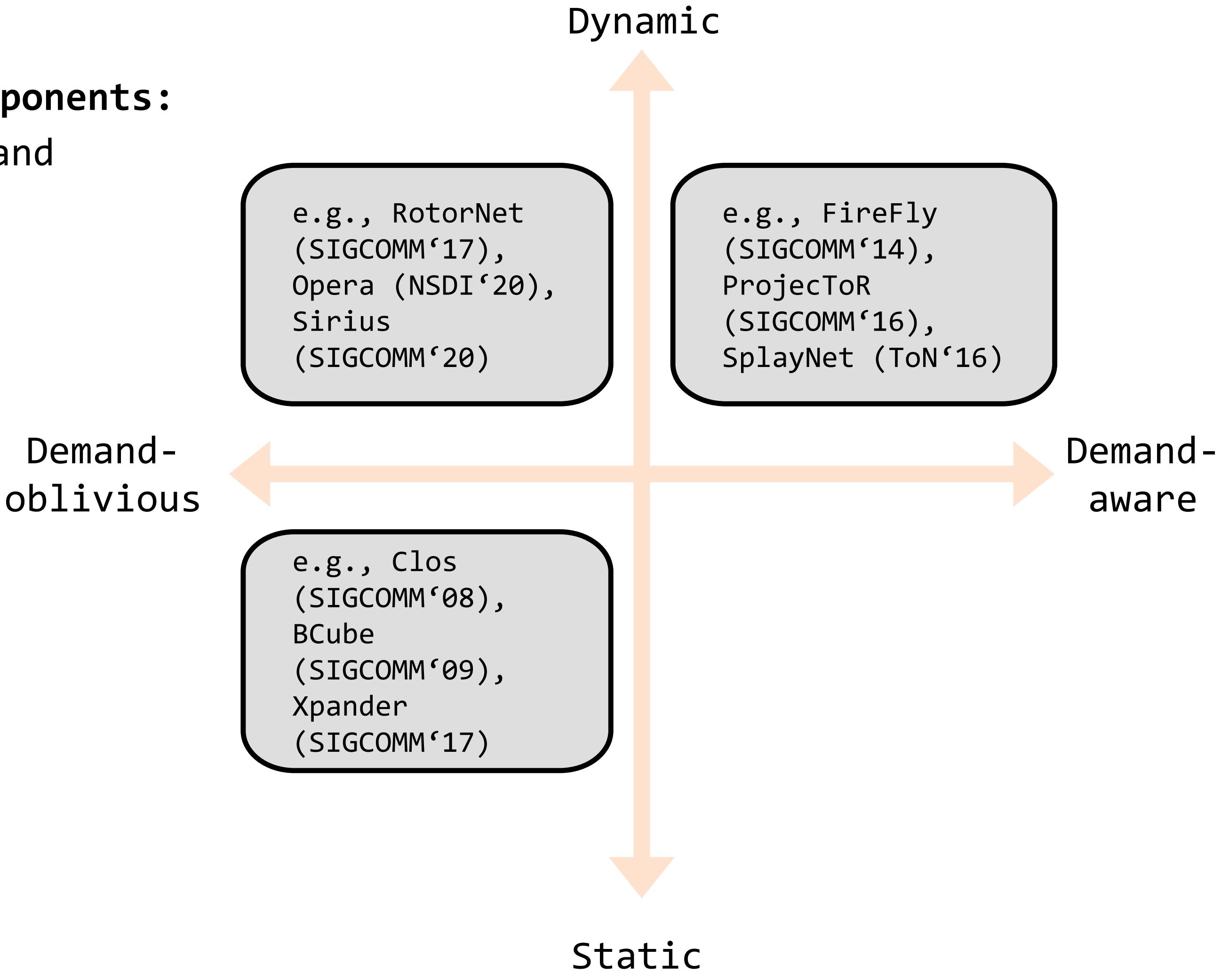


Opportunity: Tech Diversity

Diverse topology components:

⇀ demand-oblivious and
demand-aware

⇀ static vs dynamic



Dynamic

e.g., RotorNet (SIGCOMM‘17), Opera (NSDI‘20), Sirius (SIGCOMM‘20)

e.g., FireFly (SIGCOMM‘14), ProjecToR (SIGCOMM‘16), SplayNet (ToN‘16)

Demand-oblivious

Demand-aware

e.g., Clos (SIGCOMM‘08), BCube (SIGCOMM‘09), Xpander (SIGCOMM‘17)

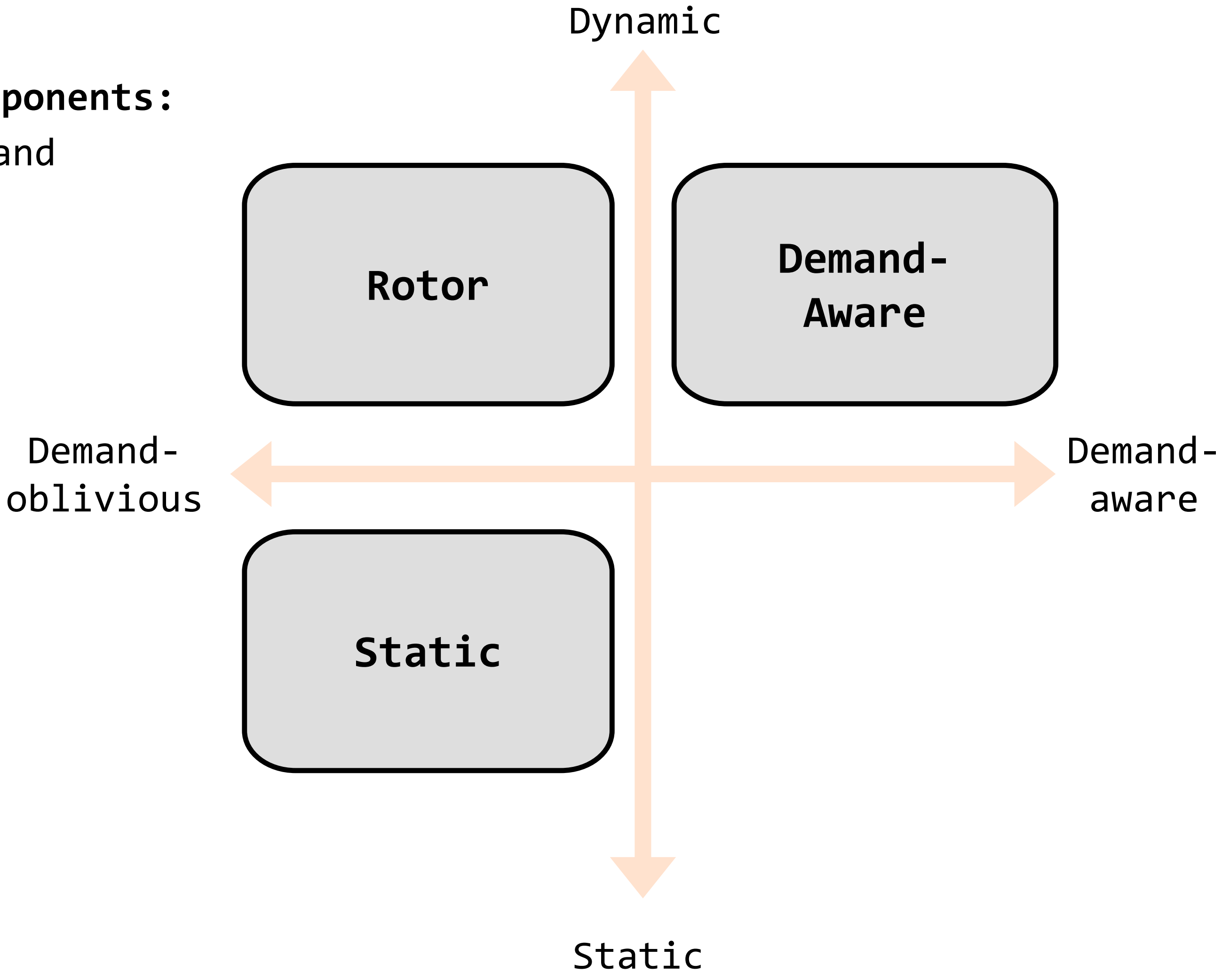
Static

Opportunity: Tech Diversity

Diverse topology components:

⇀ demand-oblivious and
demand-aware

⇀ static vs dynamic

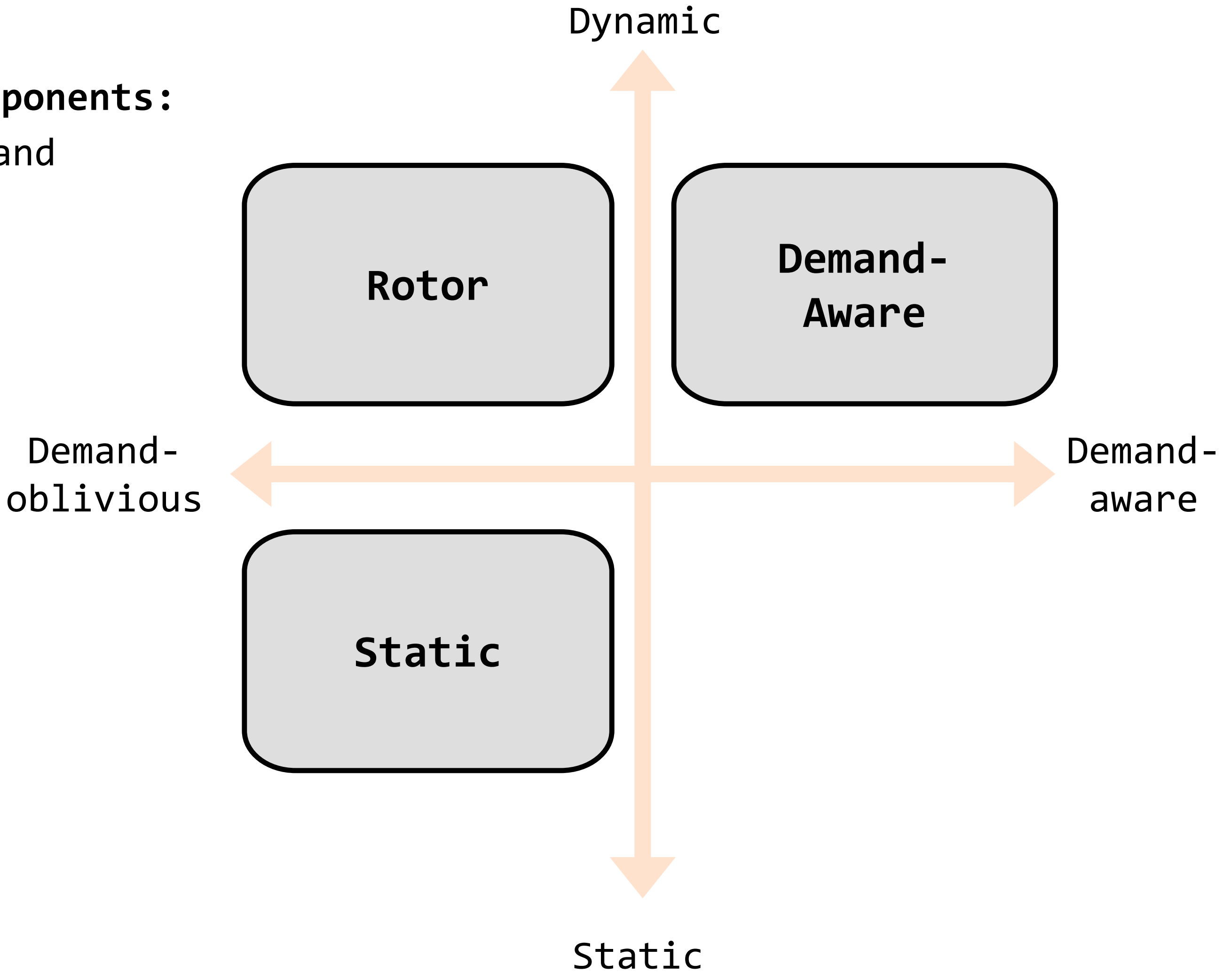


Opportunity: Tech Diversity

Diverse topology components:

⇀ demand-oblivious and
demand-aware

⇀ static vs dynamic



Dynamic

Rotor

Demand-Aware

Demand-oblivious

Demand-aware

Static

Static

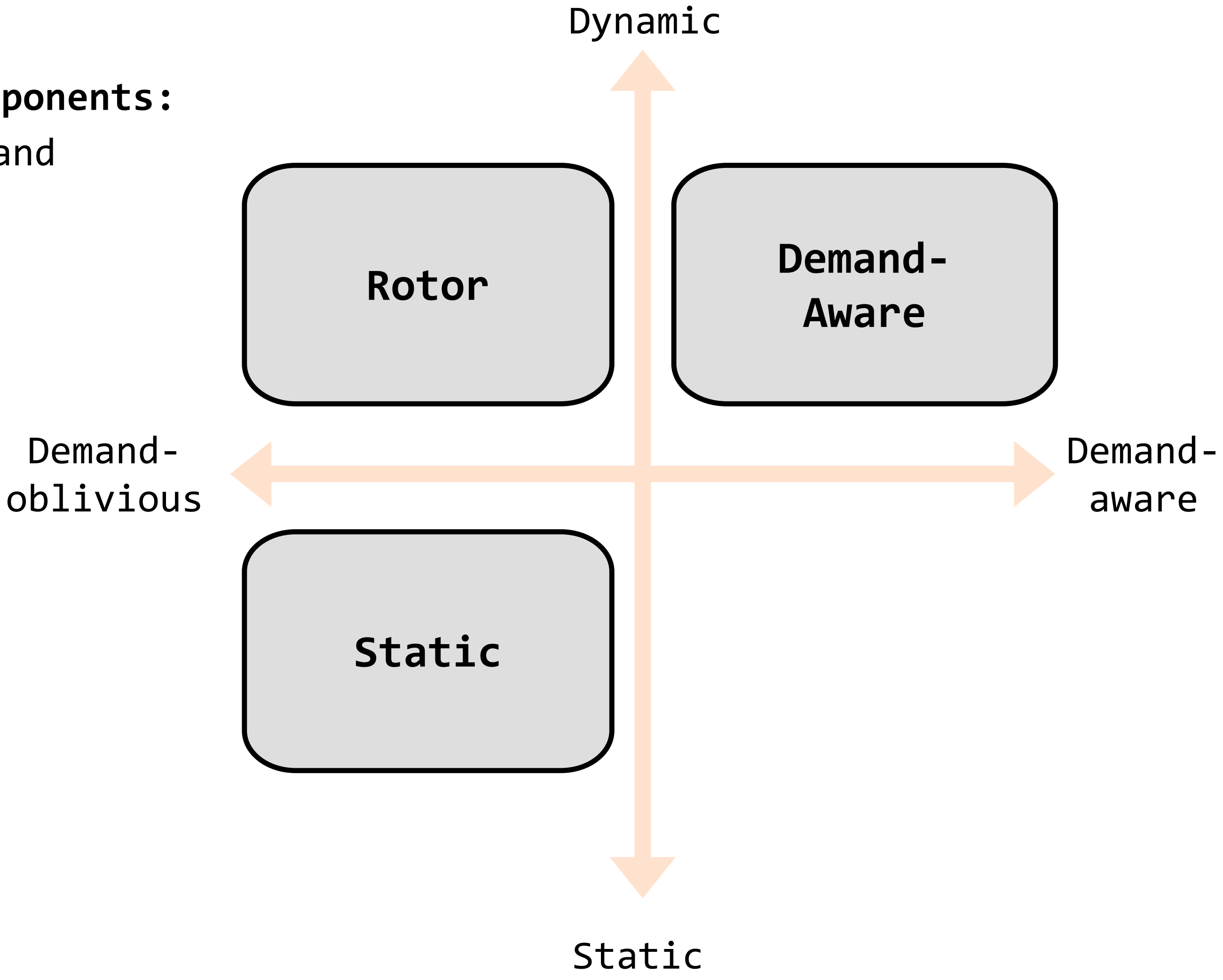
Which approach is best?

Opportunity: Tech Diversity

Diverse topology components:

⇀ demand-oblivious and demand-aware

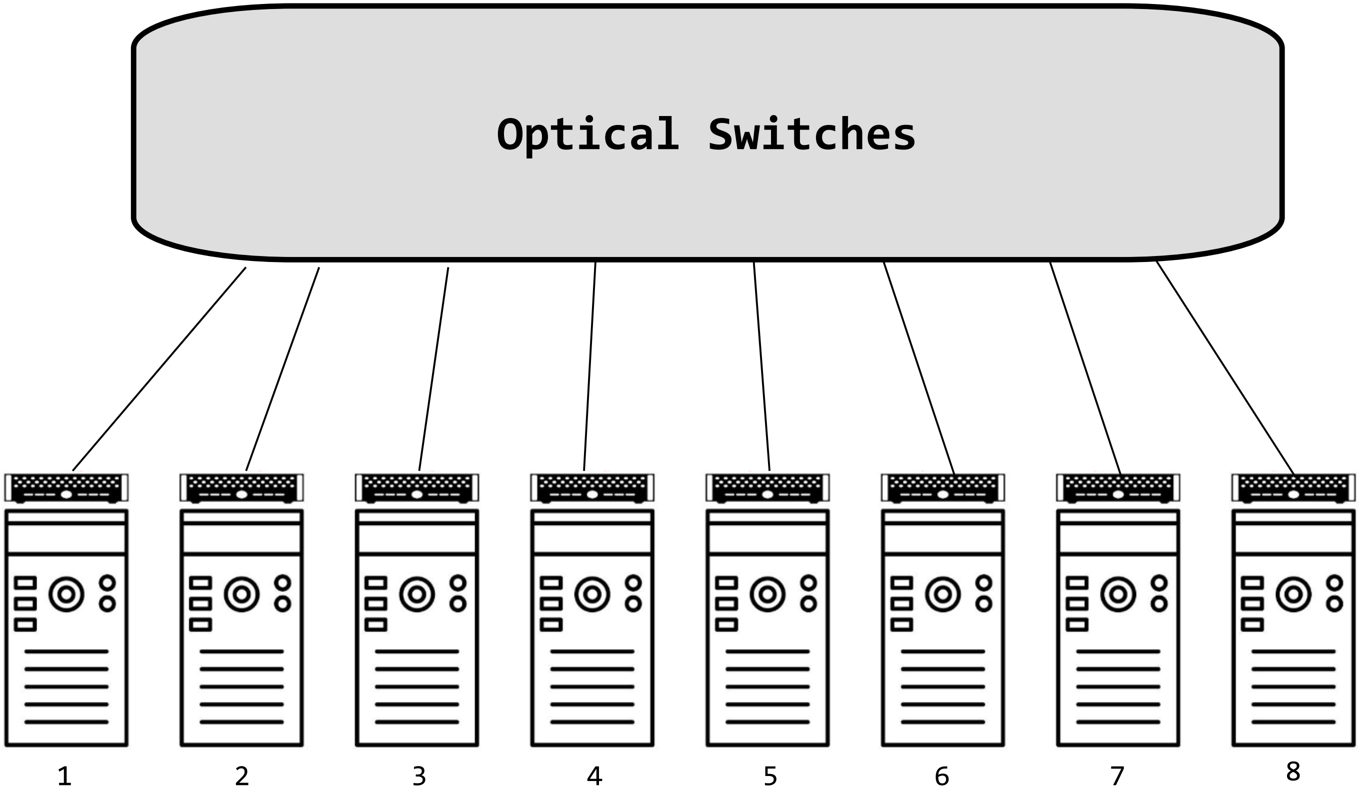
⇀ static vs dynamic



Which approach is best?

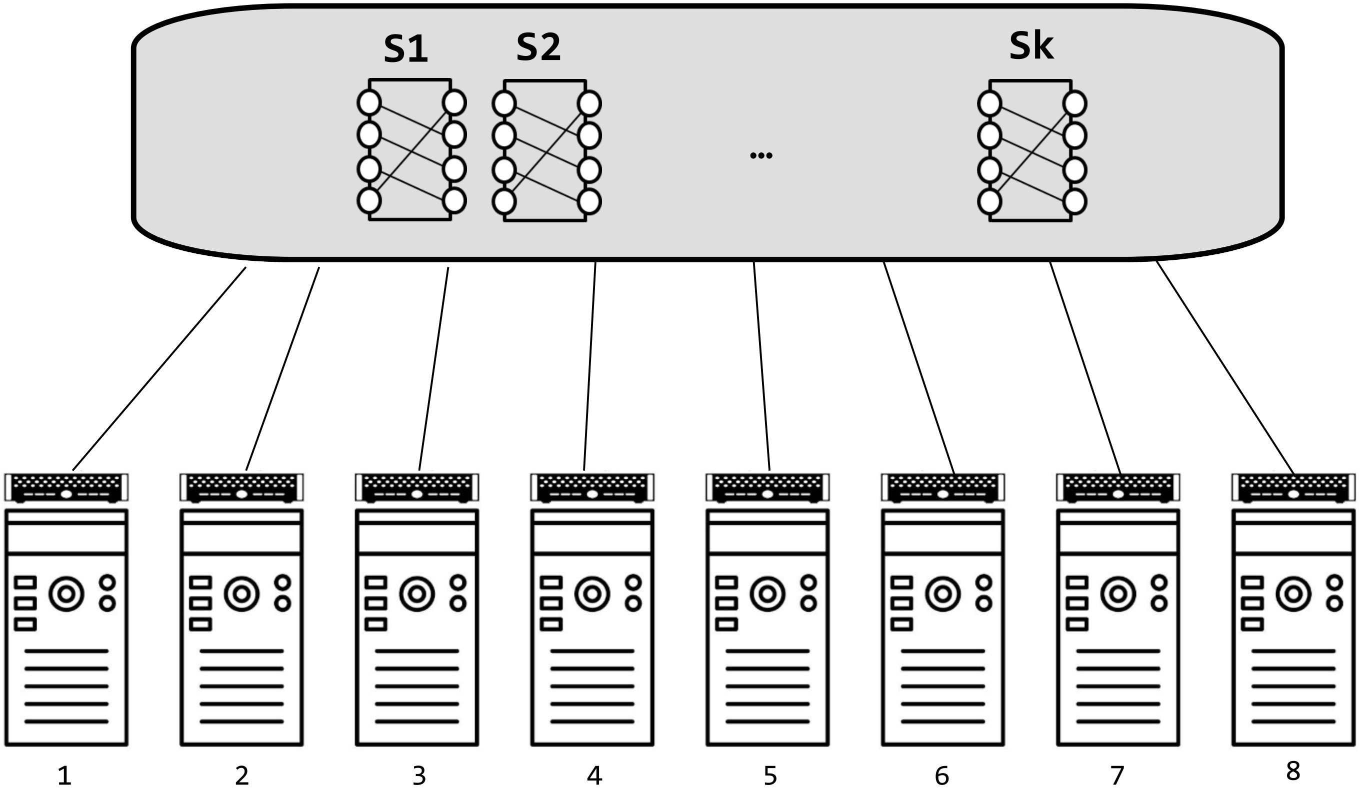
As always in CS:
It depends…

Rack Interconnect



Typical rack internconnect: ToR-Matching-ToR (TMT) model

Rack Interconnect

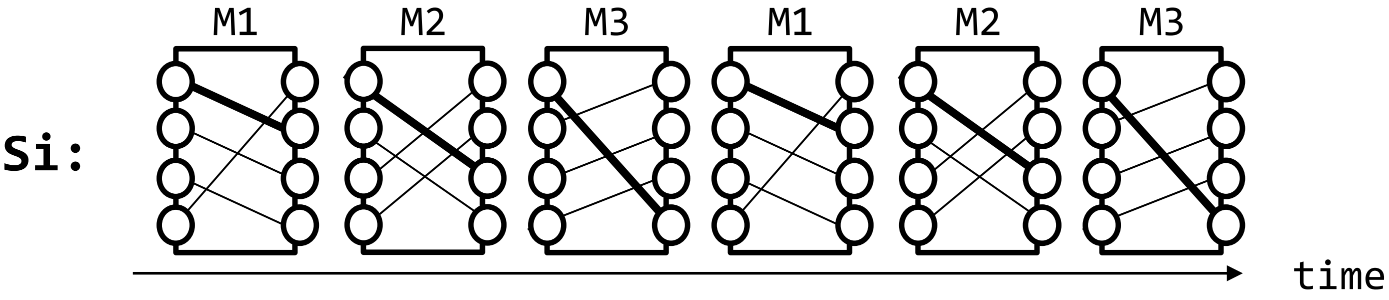


Typical rack internconnect: ToR-Matching-ToR (TMT) model

Details: Switch Types

Periodic Switch (aka Rotor Switch)

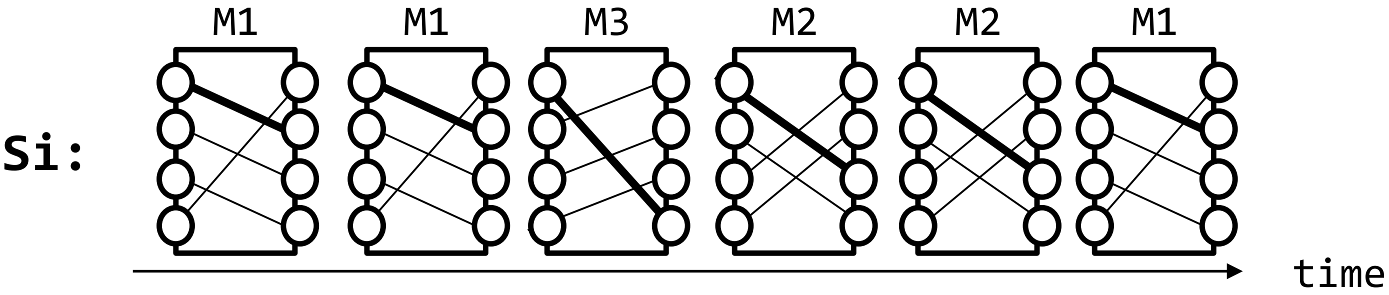
Rotor switch: periodic matchings (demand-oblivious)



Details: Switch Types

Demand-Aware Switch

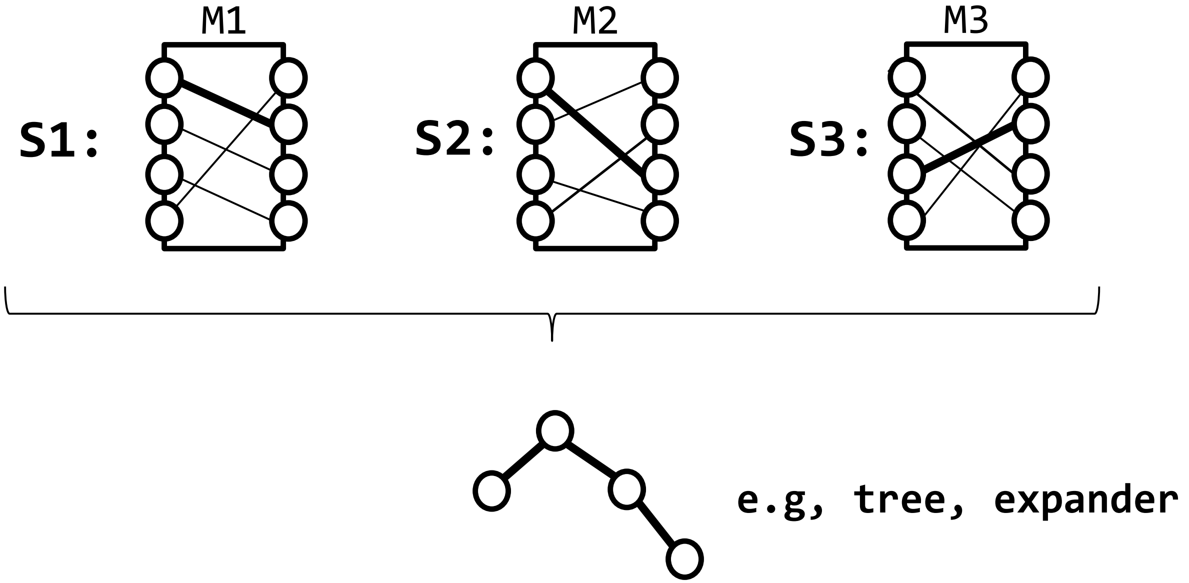
Demand-aware switch: optimized matchings



Details: Switch Types

Static Switch

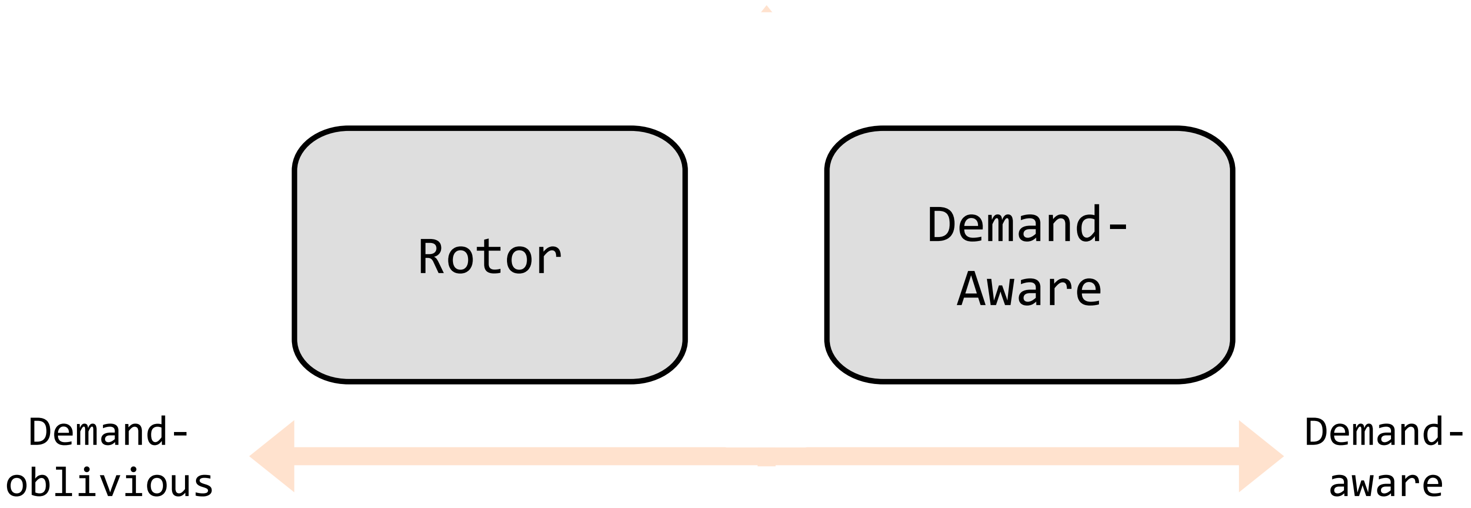
Static switches: combine for optimized static topology



e.g, tree, expander

Design Tradeoffs (1)

The “Awareness-Dimension”



Rotor

Demand-Aware

Demand-oblivious

Demand-aware

Good for all-to-all traffic!

⇀ oblivious: very fast periodic direct connectivity

⇀ no control plane overhead

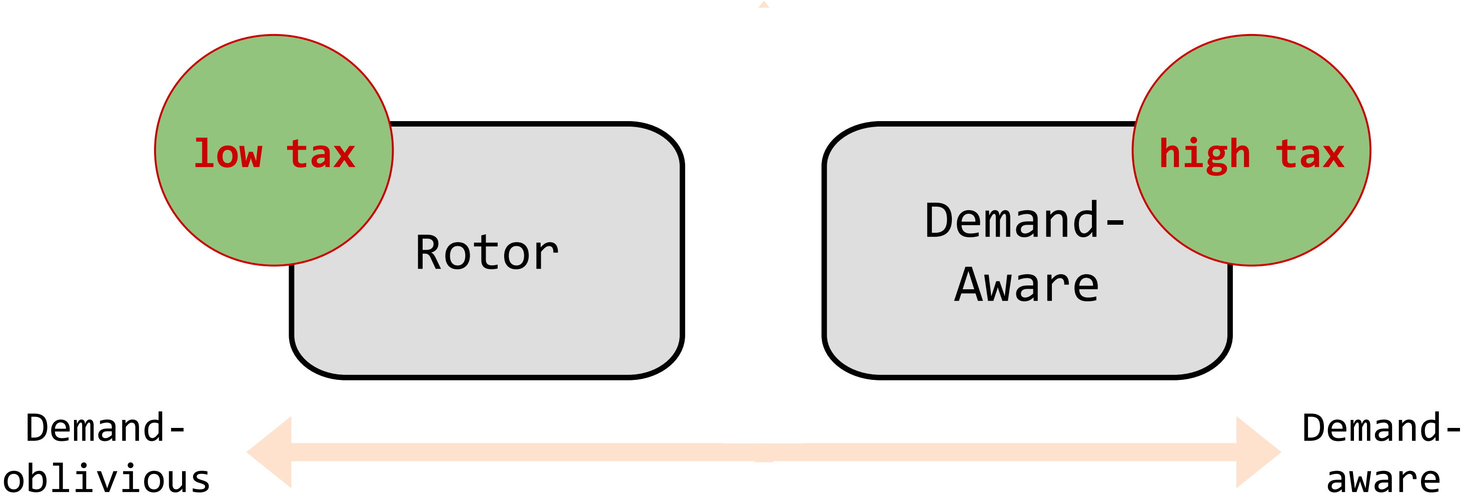
Good for elephant flows!

⇀ optimizable toward traffic

⇀ but slower

Design Tradeoffs (1)

The “Awareness-Dimension”



Good for all-to-all traffic!

⇀ oblivious: very fast
periodic direct connectivity

⇀ no control plane overhead

Good for elephant flows!

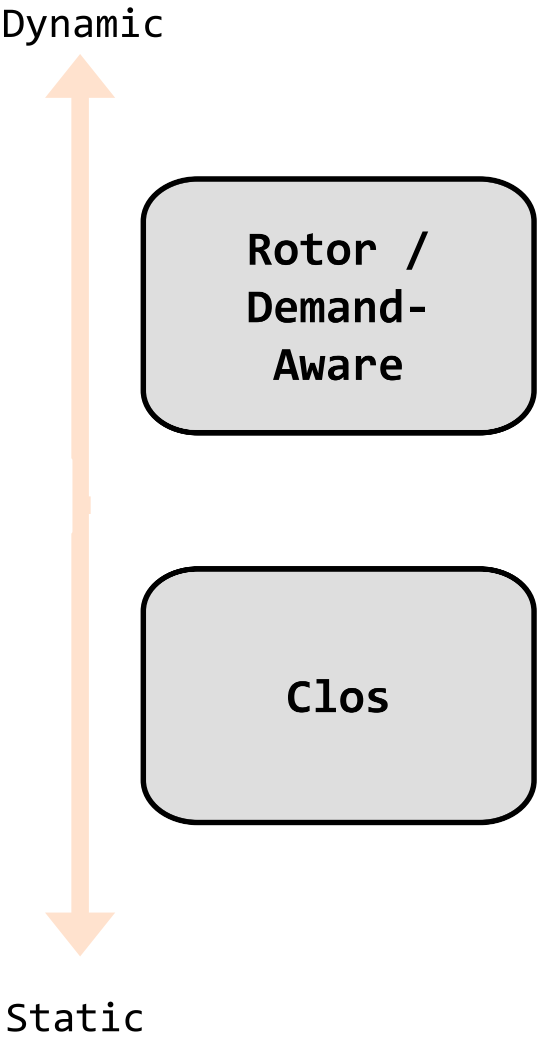
⇀ optimizable toward traffic

⇀ but slower

Compared to static networks: latency tax!

Design Tradeoffs (2)

The “Flexibility-Dimension”



Good for high throughput!
→ direct connectivity saves bandwidth along links

Good for low latency!
→ no need to wait for reconfigurable links
→ **compared to dynamic: bandwidth tax (multi-hop)**

Design Tradeoffs (2)

The “Flexibility-Dimension”

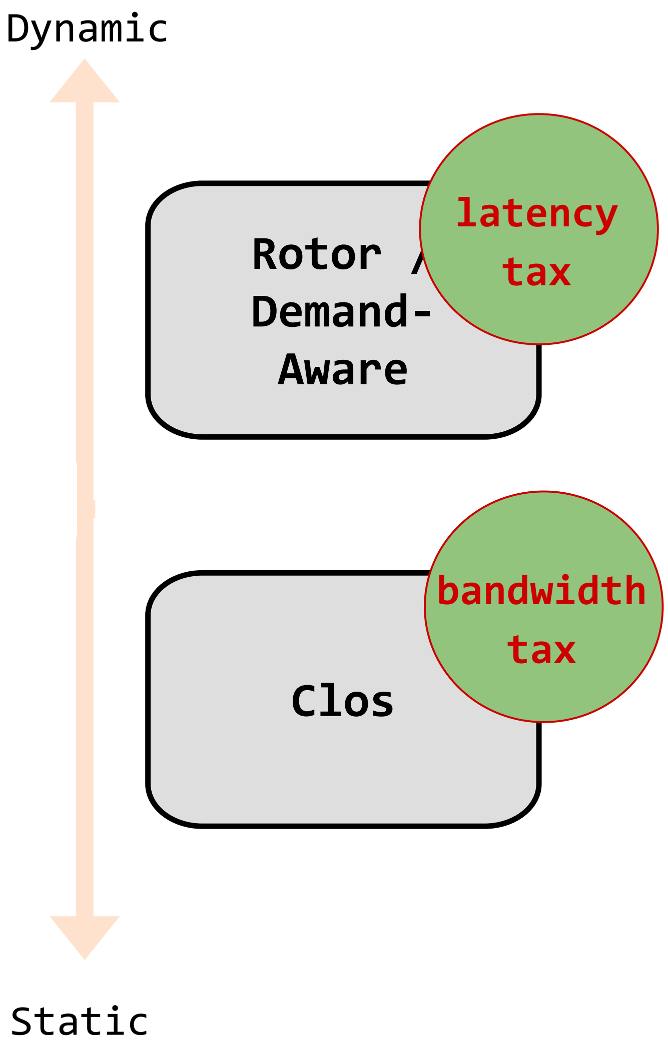
Good for high throughput!

⇀ direct connectivity saves bandwidth along links

Good for low latency!

⇀ no need to wait for reconfigurable links

⇀ **compared to dynamic: bandwidth tax (multi-hop)**



Dynamic

Rotor / Demand-Aware

latency tax

Clos

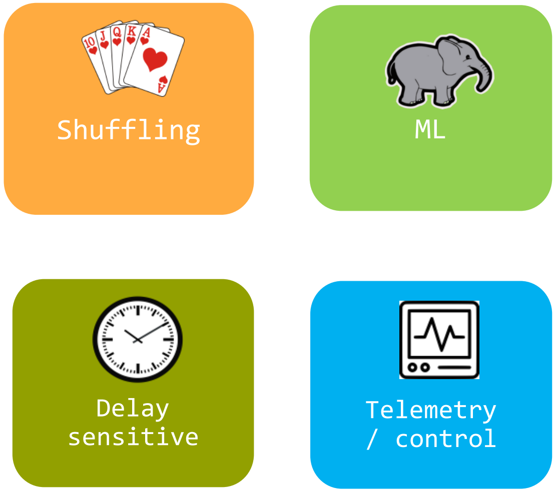
bandwidth tax

Static

First Observations

- **Observation 1:** Different topologies provide different tradeoffs.
- **Observation 2:** Different traffic requires different topology types.
- **Observation 3:** A **mismatch of demand** and topology can increase **flow completion times**.

Examples:
Match or Mismatch?



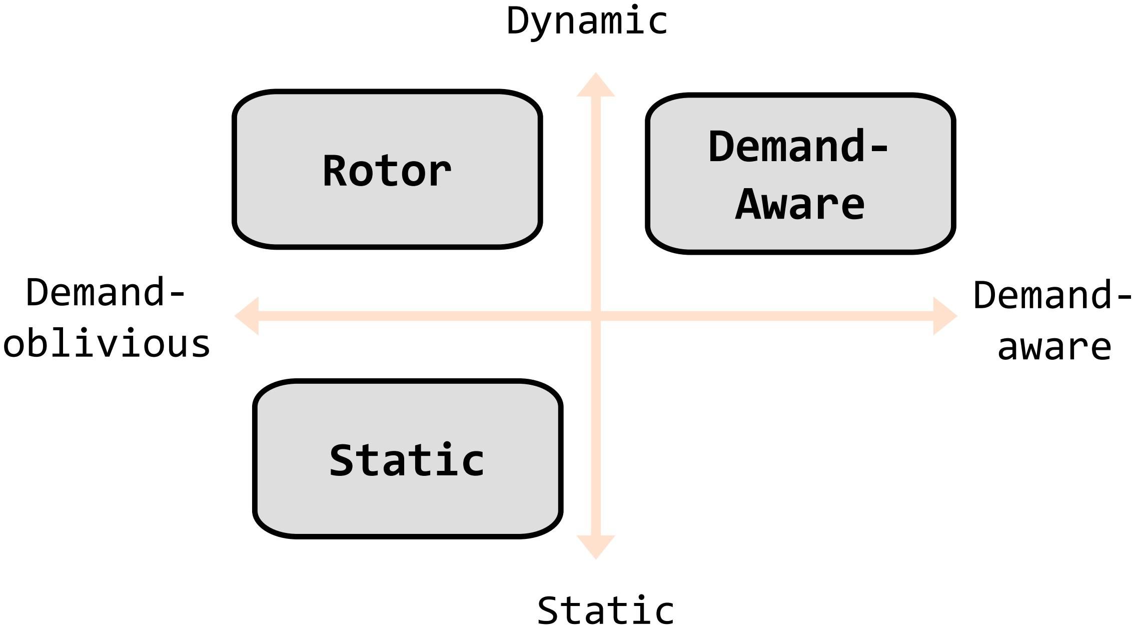
Shuffling

ML

Delay sensitive

Telemetry / control

Demand



Dynamic

Demand-oblivious

Rotor

Demand-Aware

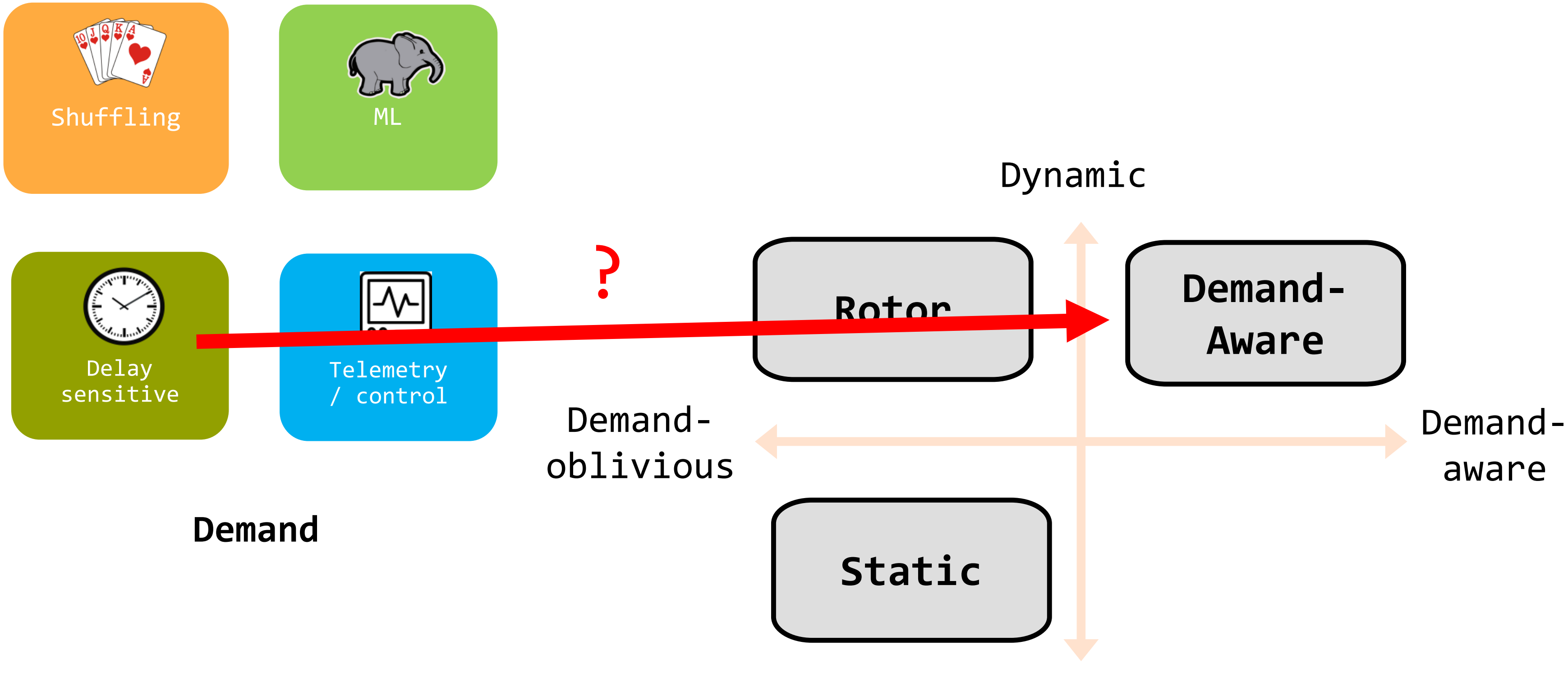
Static

Demand-aware

Static

Topology

Examples:
Match or Mismatch?



Shuffling

ML

Delay sensitive

Telemetry / control

?

Dynamic

Rotor

Demand-Aware

Demand-oblivious

Demand-aware

Static

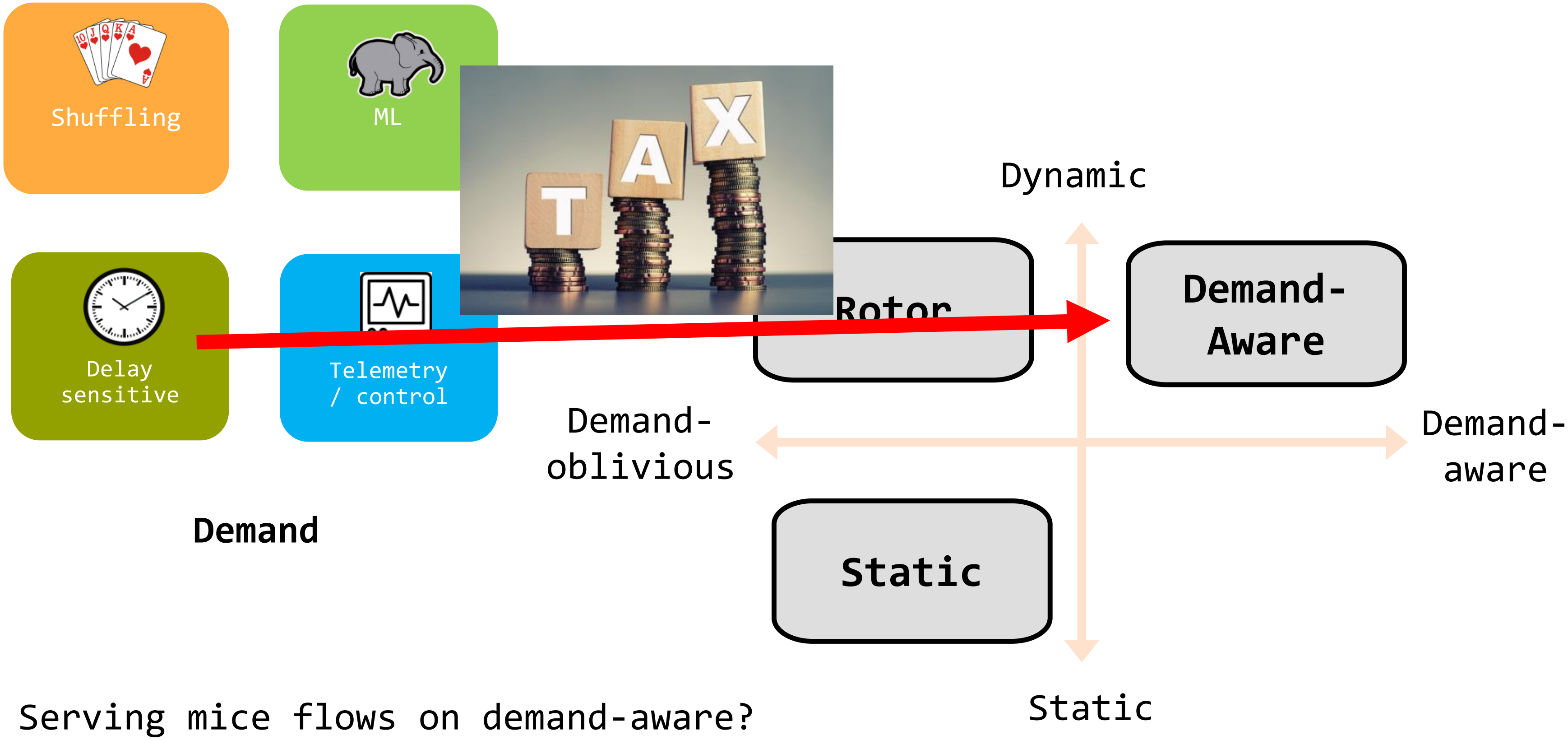
Demand

Serving mice flows on demand-aware?

Static

Topology

Examples:
Match or Mismatch?



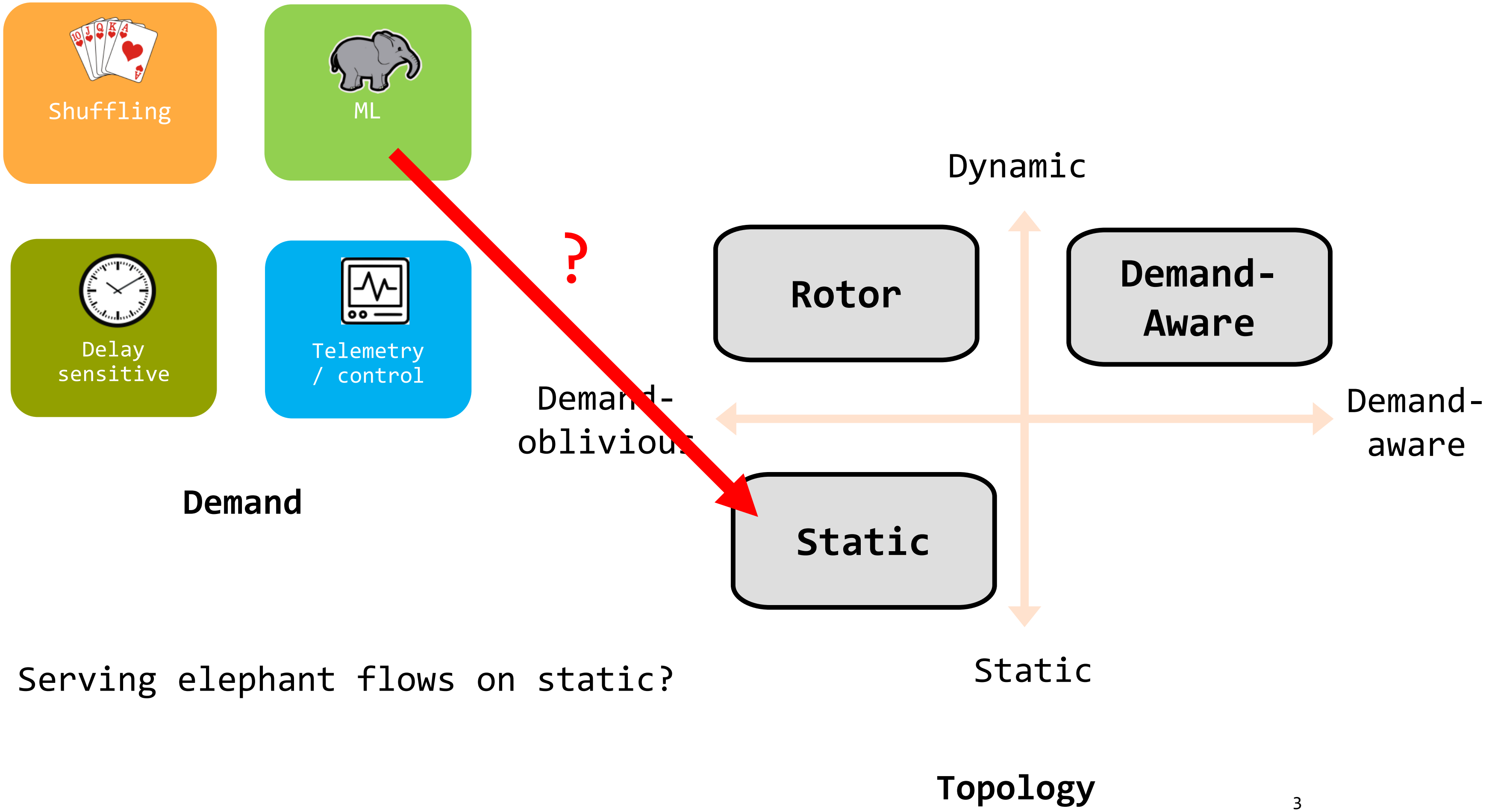
Demand

Serving mice flows on demand-aware?

Bad idea! Latency tax.

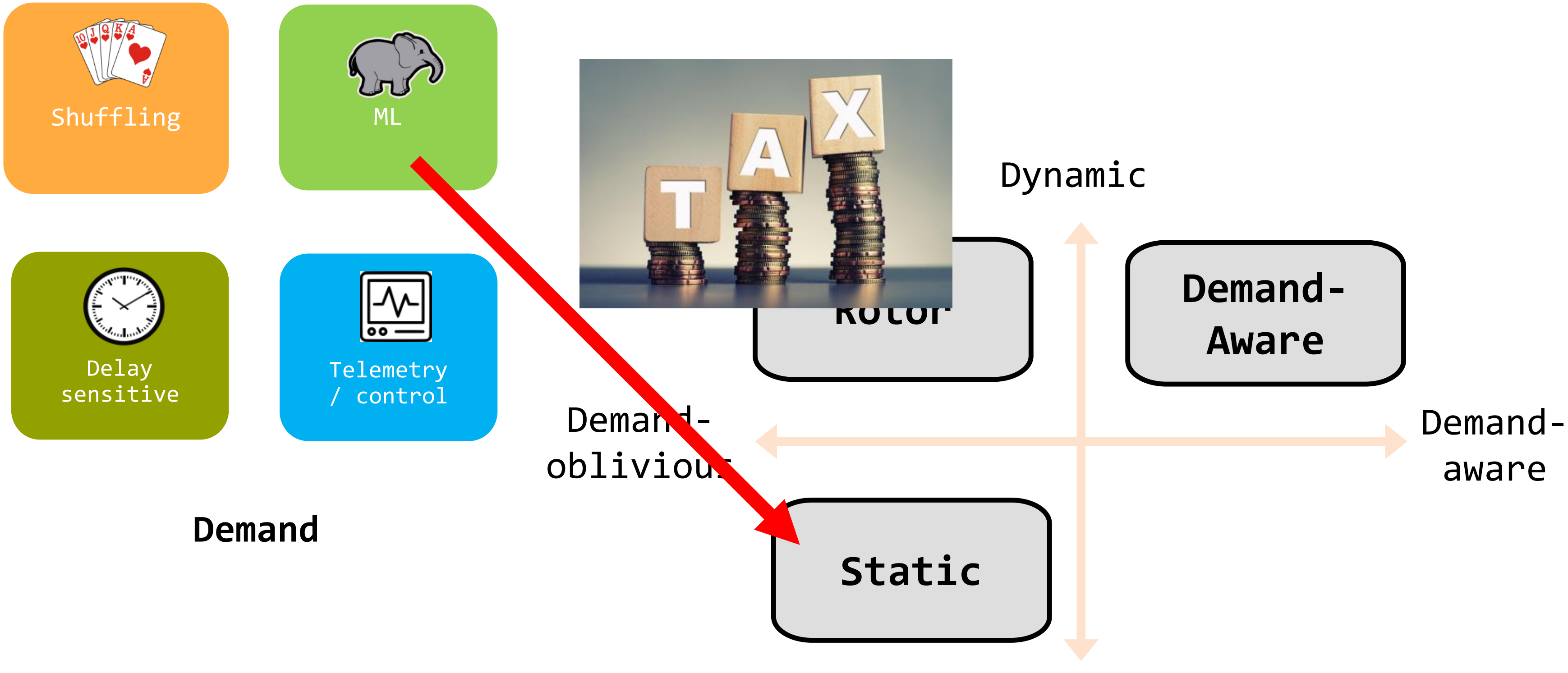
Topology

Examples:
Match or Mismatch?



Serving elephant flows on static?

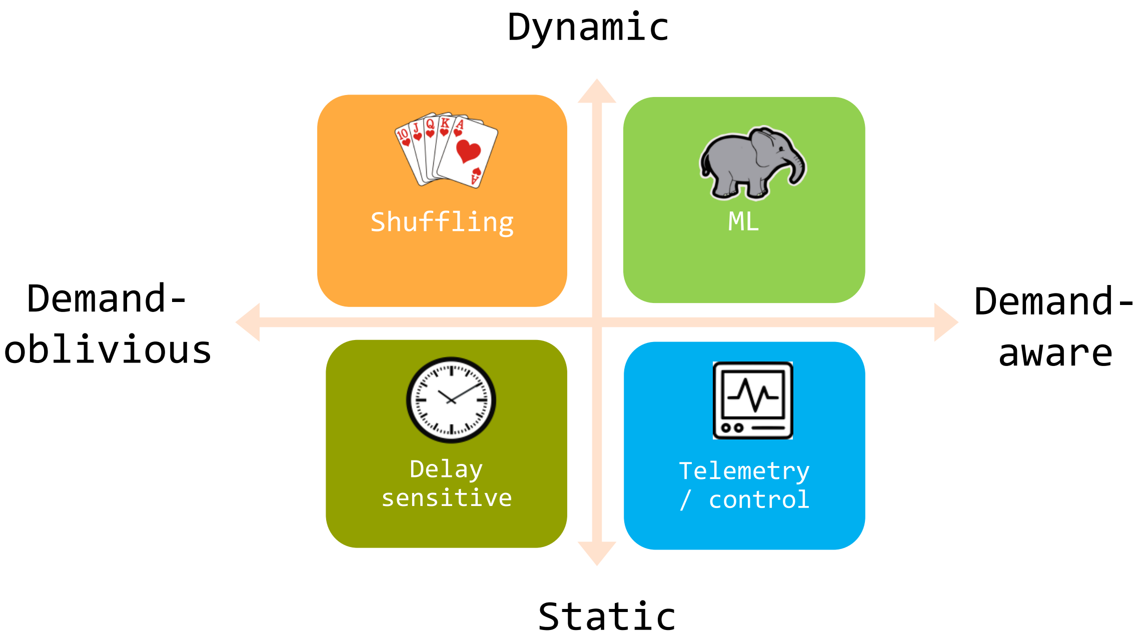
Examples:
Match or Mismatch?



Serving elephant flows on static?

Bad idea! Bandwidth tax.

Cerberus: It's a Match!



Dynamic

Shuffling

ML

Demand-oblivious

Demand-aware

Delay sensitive

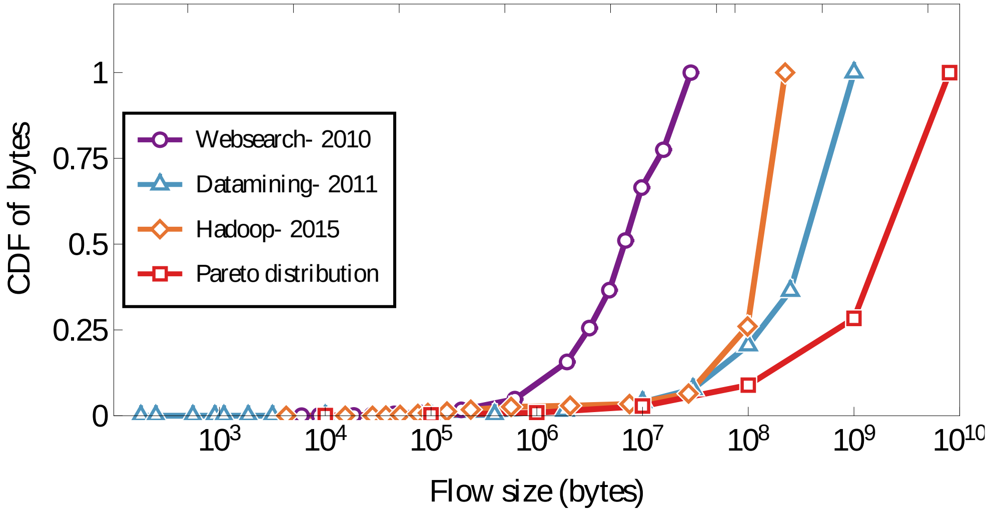
Telemetry / control

Static

Our system Cerberus* serves traffic on the “best topology”!

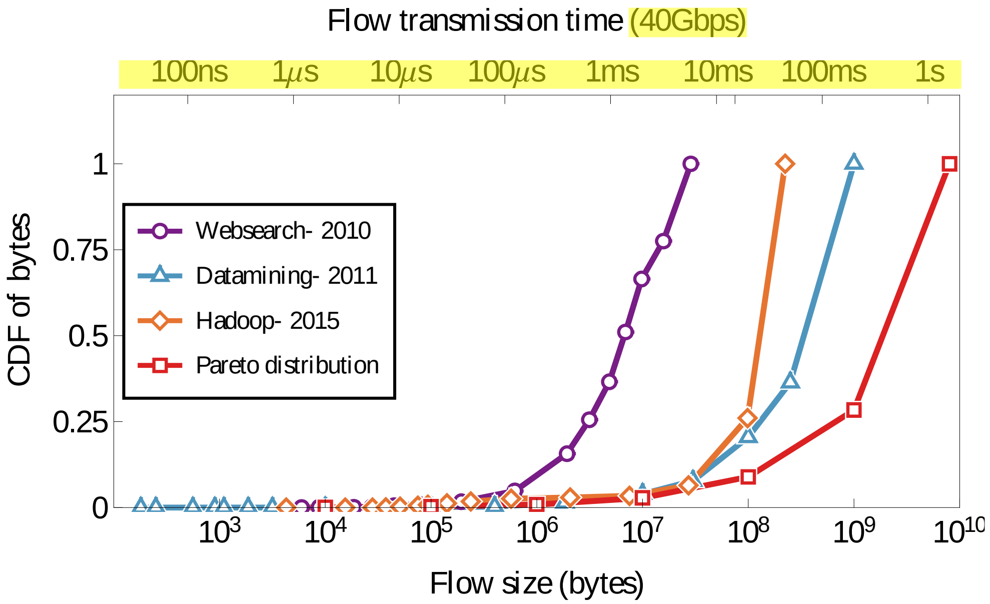
* Griner et al., ACM SIGMETRICS 2022

Flow Size Matters



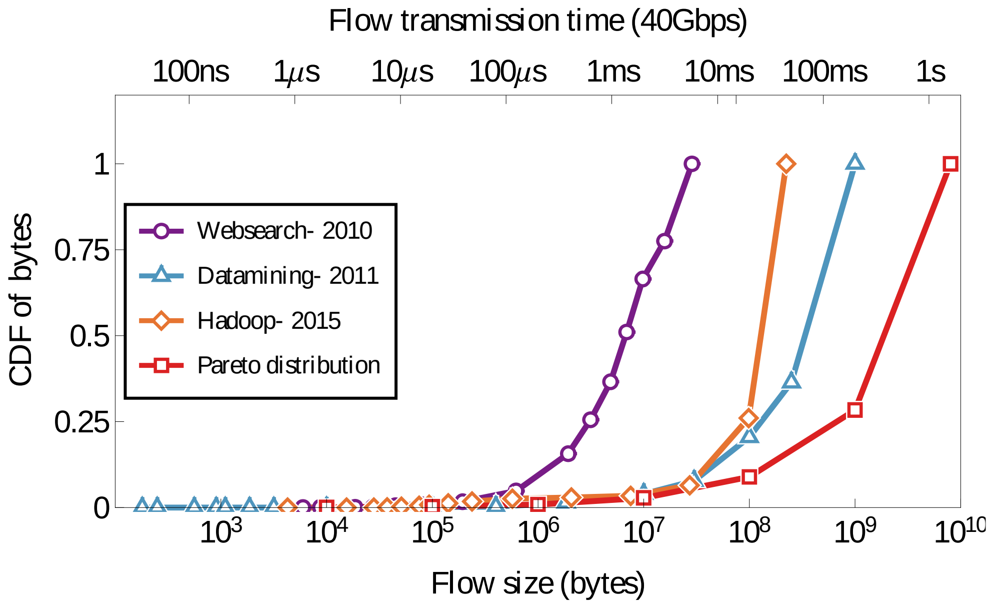
⇢ **Observation 1:** Different apps have different flow size distributions.

Flow Size Matters



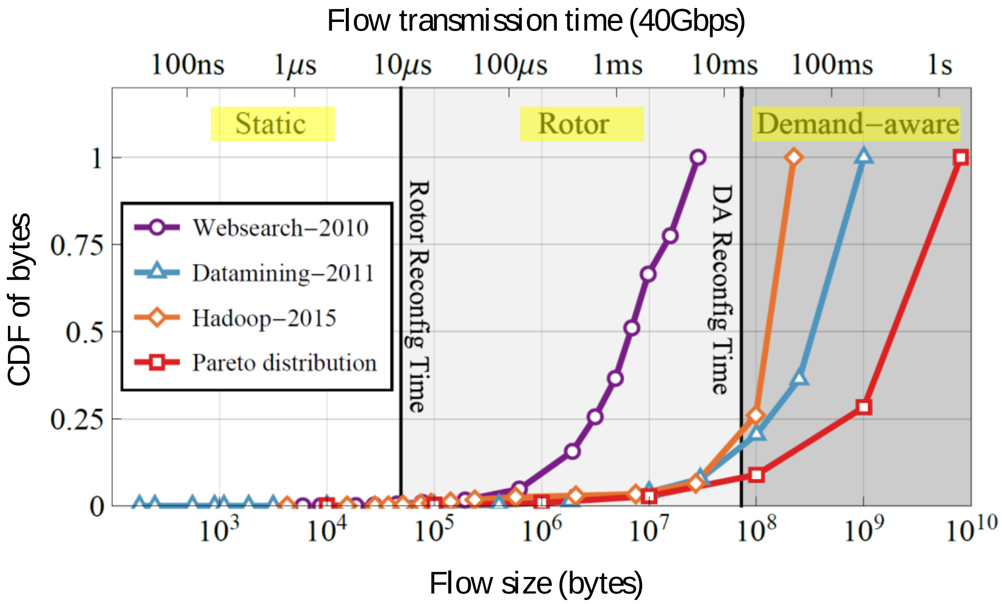
- **Observation 1:** Different apps have different flow size distributions.
- **Observation 2:** The transmission time of a flow depends on its size.

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.

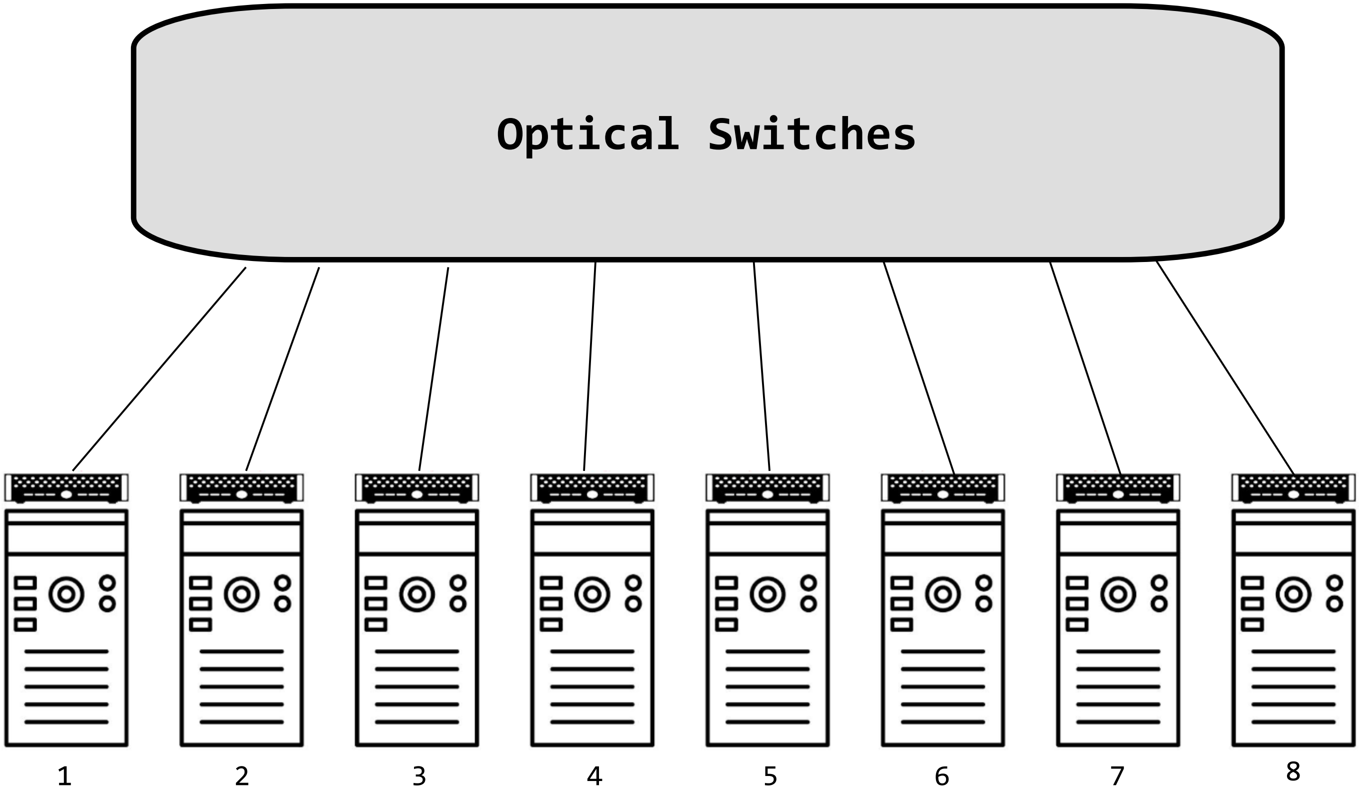
Flow Size Matters



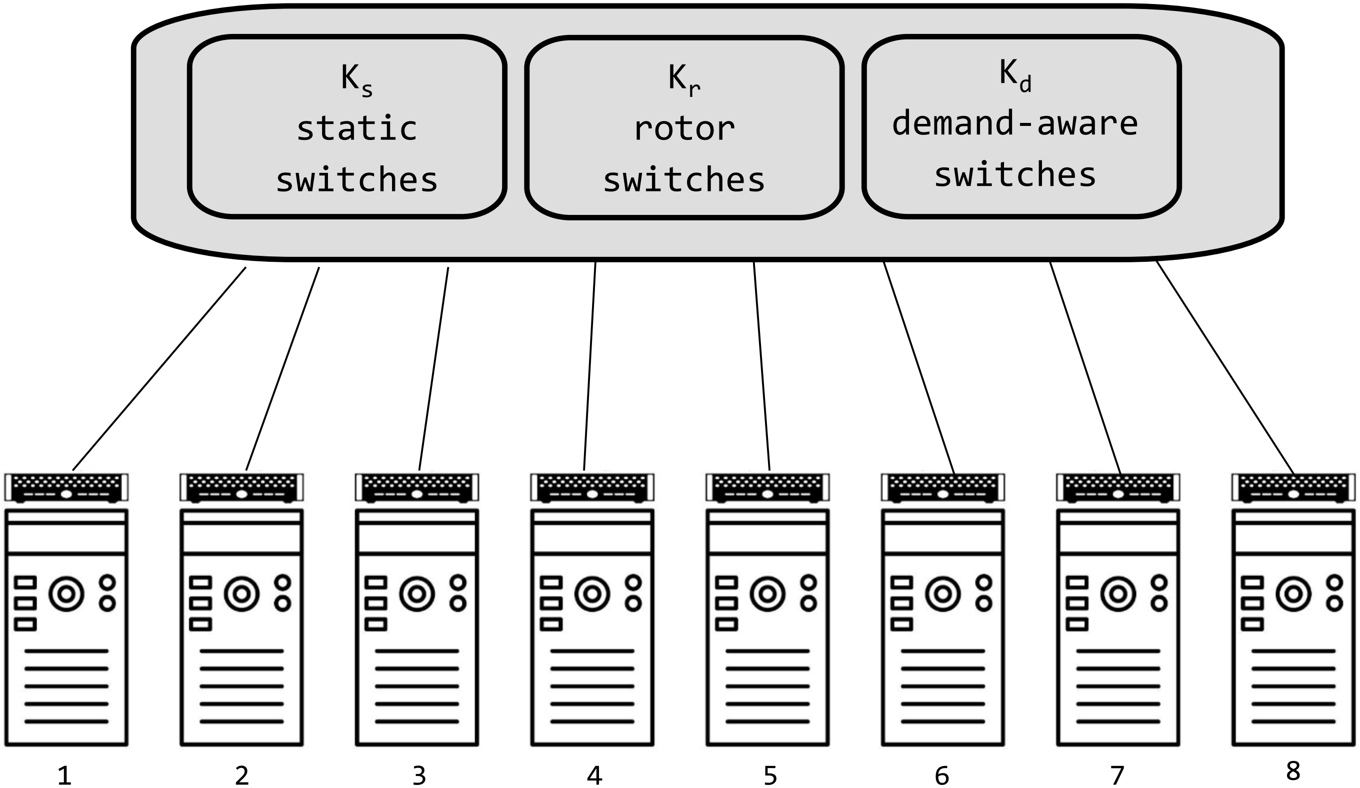
It's a Match!

- ⇢ **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

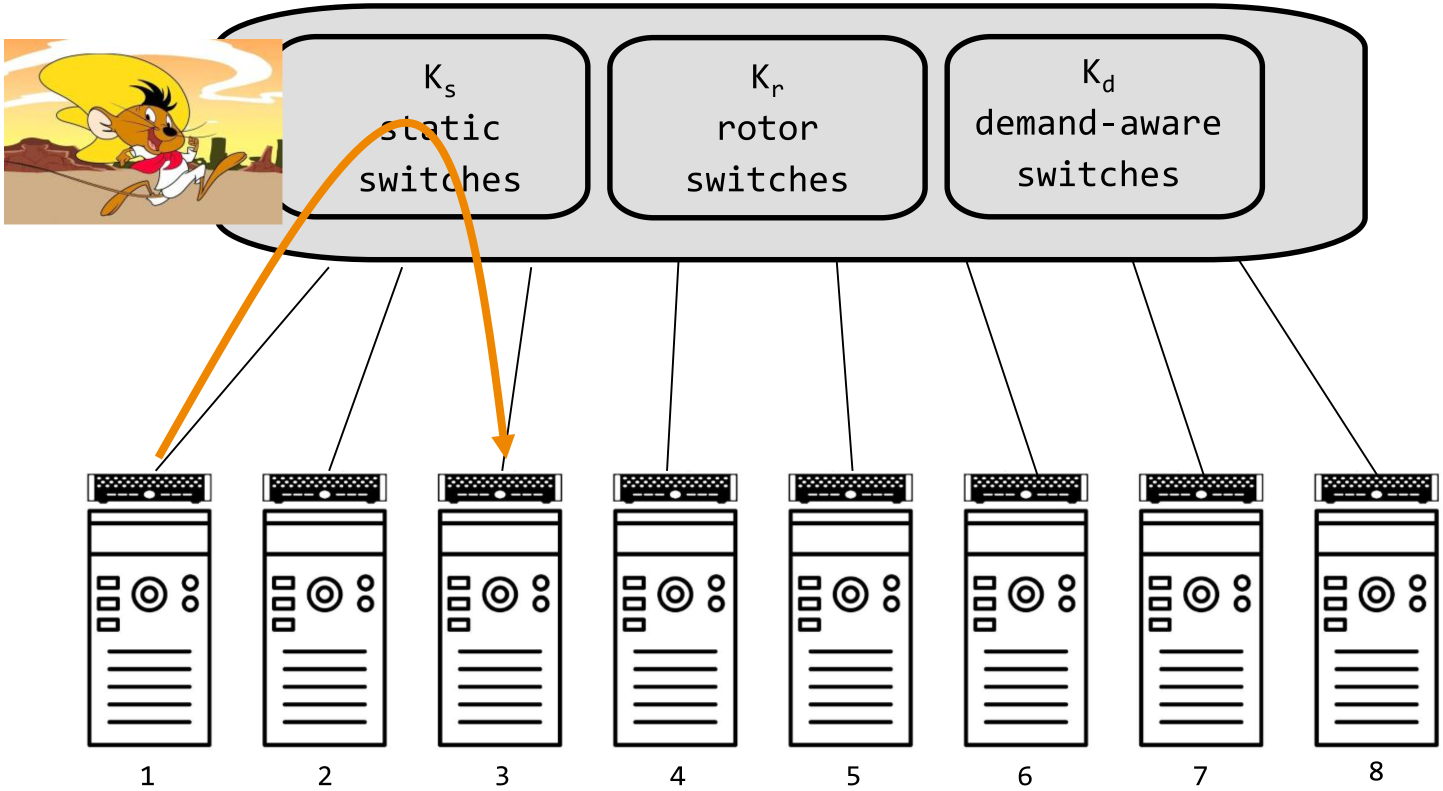


K_s
static
switches

K_r
rotor
switches

K_d
demand-aware
switches

Cerberus



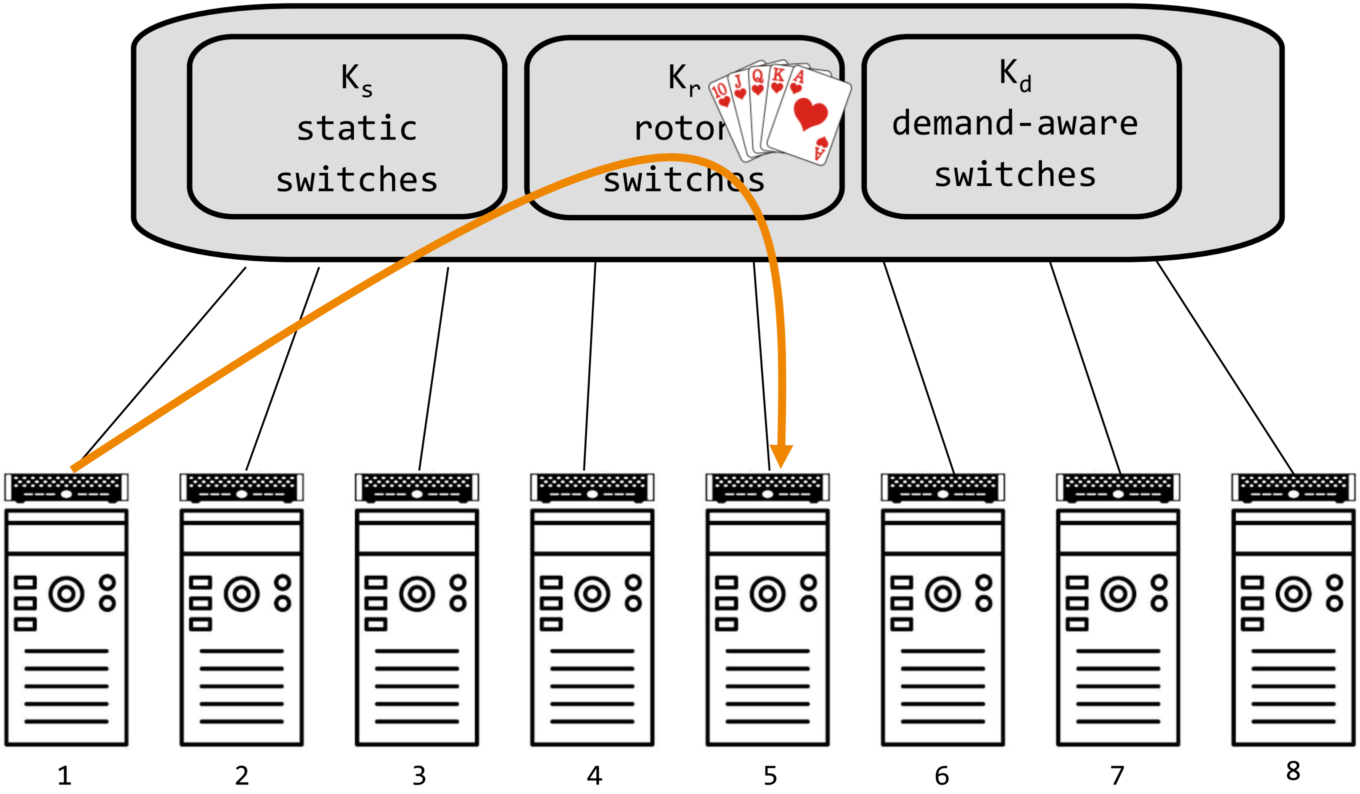
K_s
static
switches

K_r
rotor
switches

K_d
demand-aware
switches

Scheduling: Small flows go via static switches…

Cerberus



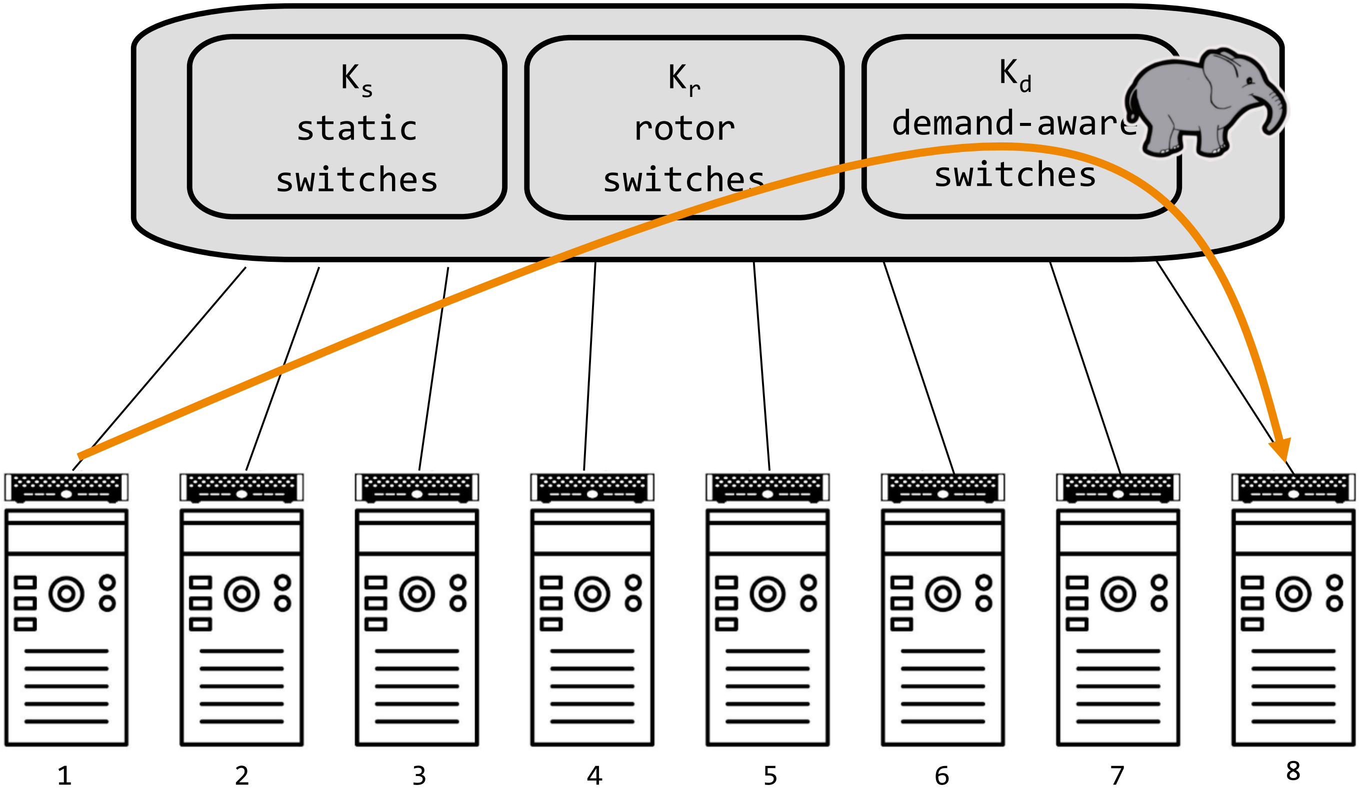
K_s
static
switches

K_r
rotor
switches

K_d
demand-aware
switches

Scheduling: … medium flows via rotor switches…

Cerberus



K_s
static
switches

K_r
rotor
switches

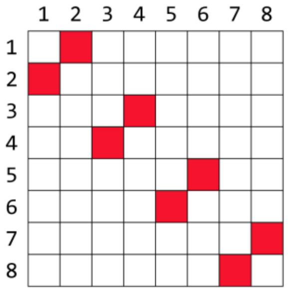
K_d
demand-aware
switches

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

Throughput Analysis

Demand Matrix

T



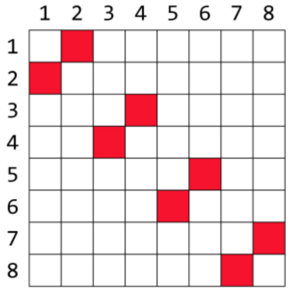
	1	2	3	4	5	6	7	8
1		🟥						
2	🟥							
3				🟥				
4			🟥					
5						🟥		
6					🟥			
7								🟥
8							🟥	

Metric: throughput
of a demand matrix…

Throughput Analysis

Demand Matrix

T



	1	2	3	4	5	6	7	8
1		🟥						
2	🟥							
3				🟥				
4			🟥					
5						🟥		
6					🟥			
7								🟥
8							🟥	

$\times \ \theta(T)$

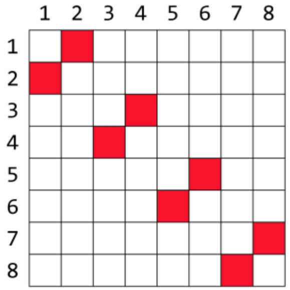
Metric: throughput of a demand matrix…

… is the maximal scale down factor by which traffic is feasible.

Throughput Analysis

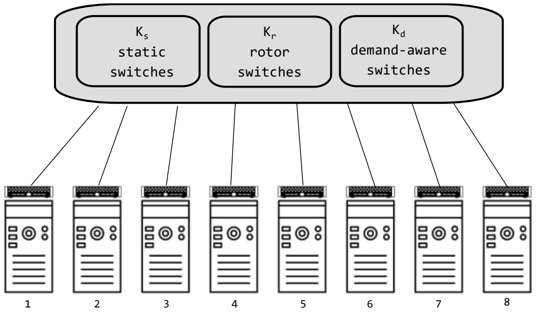
Demand Matrix

T



	1	2	3	4	5	6	7	8
1		🟥						
2	🟥							
3				🟥				
4			🟥					
5						🟥		
6					🟥			
7								🟥
8							🟥	

$$\times \;\; \theta(T) \qquad \Rightarrow$$



Metric: throughput of a demand matrix…

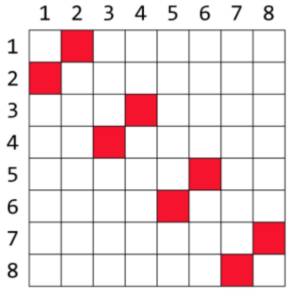
… is the maximal scale down factor by which traffic is feasible.

Throughput of network θ^*: worst case T

Throughput Analysis

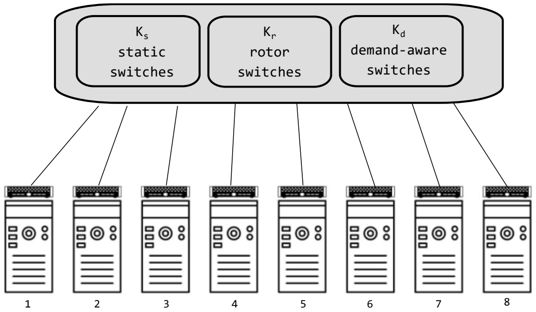
Demand Matrix

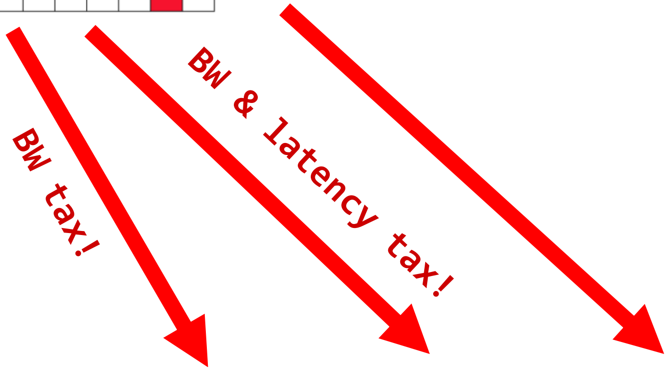
T



	1	2	3	4	5	6	7	8
1		🟥						
2	🟥							
3				🟥				
4			🟥					
5						🟥		
6					🟥			
7								🟥
8							🟥	

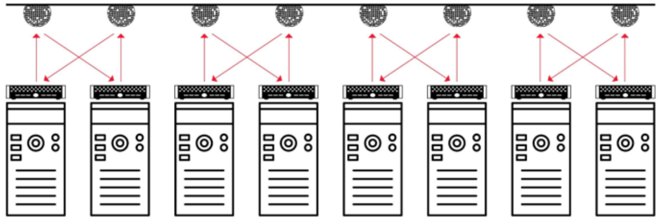
$\times \ \theta(T) \quad \Rightarrow$





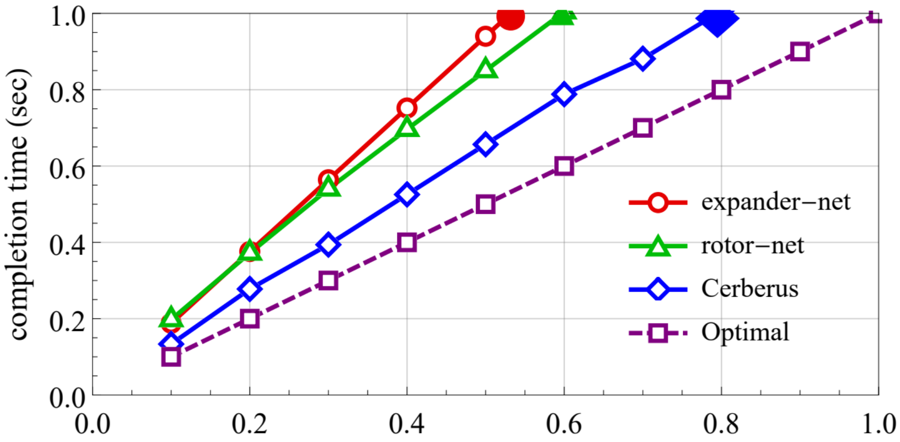
	expander-net	*rotor-net*	CERBERUS
BW-Tax	✓	✓	✗
LT-Tax	✗	✓	✓
$\theta(T)$	Thm 2	Thm 3	Thm 5
θ^*	0.53	0.45	Open
Datamining	0.53	0.6	0.8 (+33%)
Permutation	0.53	0.45	≈ 1 (+88%)
Case Study	0.53	0.66	0.9 (+36%)

Worst demand matrix for static and rotor: permutation. Best case for demand-aware!



Completion Time

- Demand completion time: How long does it take to serve a demand matrix?



Data mining workload

- Also useful in analysis: throughput can be computed more easily via demand completion time.

Conclusion

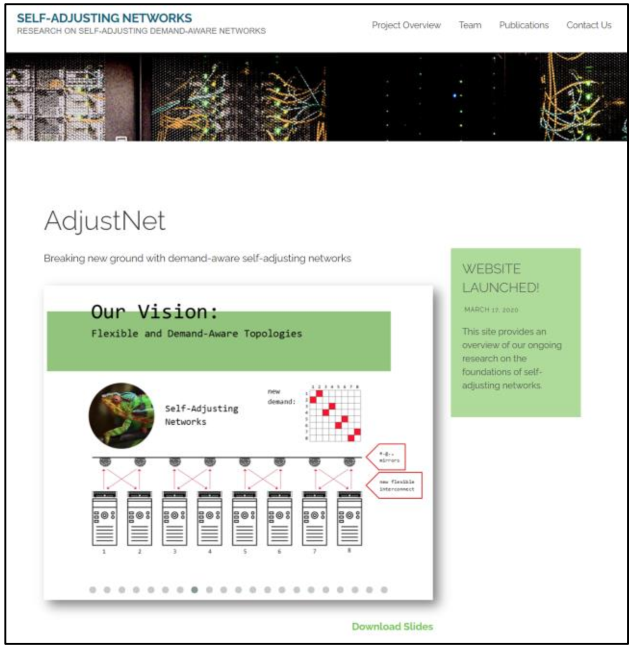
⇢ Diverse traffic requires diverse technologies

⇢ Cerberus aims to assign traffic to its best topology
  - ⇀ Depending on flow size

⇢ Many challenges
  - ⇀ Impact on routing and congestion control
  - ⇀ Sensitivity analysis
  - ⇀ Prototyping

Thank you!

Websites



SELF-ADJUSTING NETWORKS
RESEARCH ON SELF-ADJUSTING DEMAND-AWARE NETWORKS

AdjustNet

Breaking new ground with demand-aware self-adjusting networks

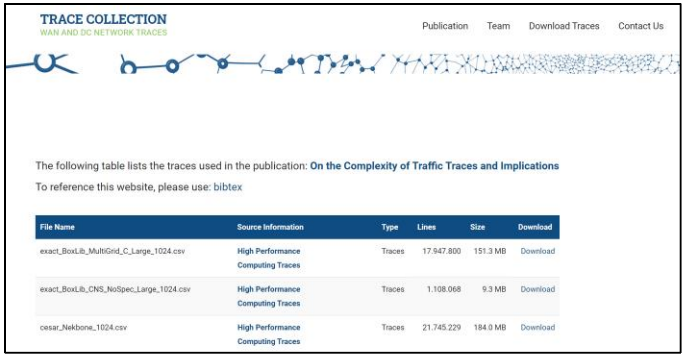
Our Vision:
Flexible and Demand-Aware Topologies

WEBSITE LAUNCHED!

MARCH 17, 2020

This site provides an overview of our ongoing research on the foundations of self-adjusting networks.

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



TRACE COLLECTION
WAN AND DC NETWORK TRACES

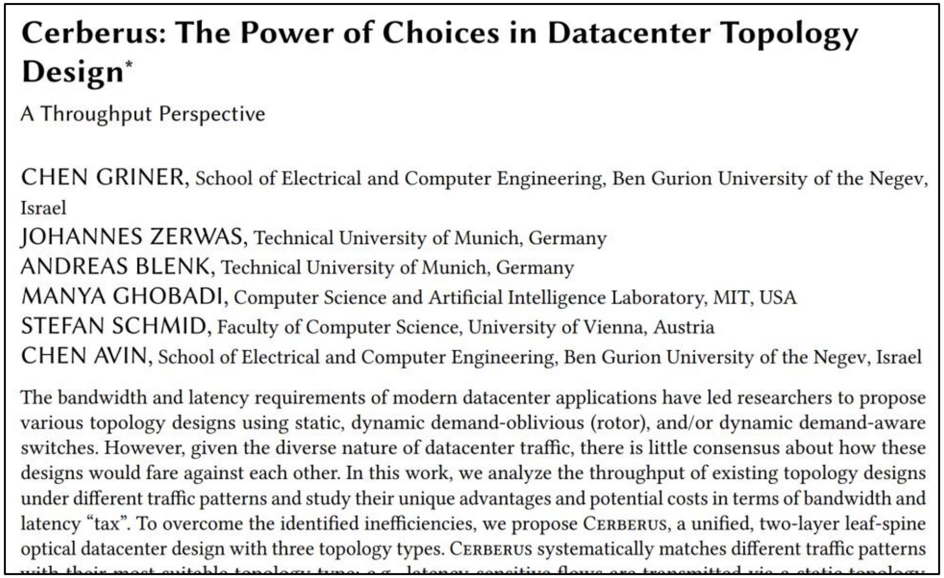
The following table lists the traces used in the publication: **On the Complexity of Traffic Traces and Implications**

To reference this website, please use: bibtex

File Name	Source Information	Type	Lines	Size	Download
exact_BoxLib_MultiGrid_C_Large_1024.csv	High Performance Computing Traces	Traces	17.947.800	151.3 MB	Download
exact_BoxLib_CNS_NoSpec_Large_1024.csv	High Performance Computing Traces	Traces	1.108.068	9.3 MB	Download
cesar_Nekbone_1024.csv	High Performance Computing Traces	Traces	21.745.229	184.0 MB	Download

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

Further Reading

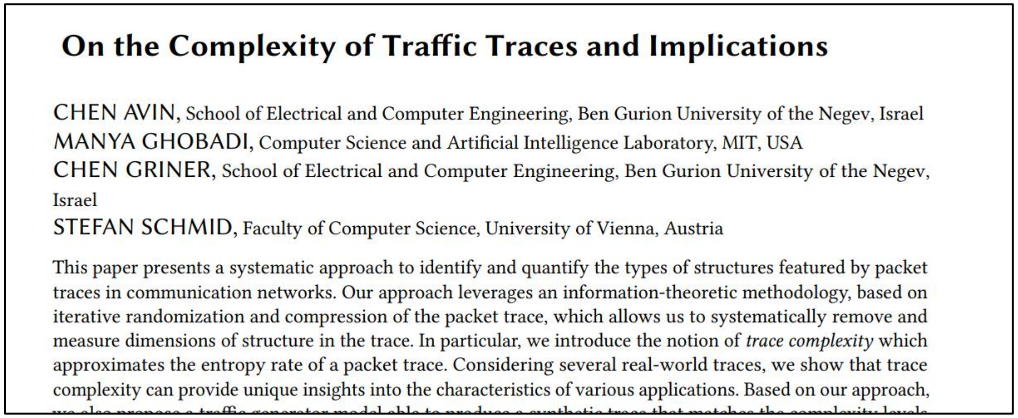


Cerberus: The Power of Choices in Datacenter Topology Design*

A Throughput Perspective

CHEN GRINER, School of Electrical and Computer Engineering, Ben Gurion University of the Negev, Israel
JOHANNES ZERWAS, Technical University of Munich, Germany
ANDREAS BLENK, Technical University of Munich, Germany
MANYA GHOBADI, Computer Science and Artificial Intelligence Laboratory, MIT, USA
STEFAN SCHMID, Faculty of Computer Science, University of Vienna, Austria
CHEN AVIN, School of Electrical and Computer Engineering, Ben Gurion University of the Negev, Israel

The bandwidth and latency requirements of modern datacenter applications have led researchers to propose various topology designs using static, dynamic demand-oblivious (rotor), and/or dynamic demand-aware switches. However, given the diverse nature of datacenter traffic, there is little consensus about how these designs would fare against each other. In this work, we analyze the throughput of existing topology designs under different traffic patterns and study their unique advantages and potential costs in terms of bandwidth and latency "tax". To overcome the identified inefficiencies, we propose CERBERUS, a unified, two-layer leaf-spine optical datacenter design with three topology types. CERBERUS systematically matches different traffic patterns



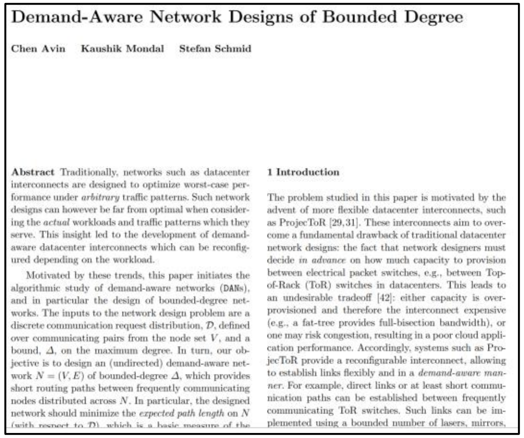
On the Complexity of Traffic Traces and Implications

CHEN AVIN, School of Electrical and Computer Engineering, Ben Gurion University of the Negev, Israel
MANYA GHOBADI, Computer Science and Artificial Intelligence Laboratory, MIT, USA
CHEN GRINER, School of Electrical and Computer Engineering, Ben Gurion University of the Negev, Israel
STEFAN SCHMID, Faculty of Computer Science, University of Vienna, Austria

This paper presents a systematic approach to identify and quantify the types of structures featured by packet traces in communication networks. Our approach leverages an information-theoretic methodology, based on iterative randomization and compression of the packet trace, which allows us to systematically remove and measure dimensions of structure in the trace. In particular, we introduce the notion of *trace complexity* which approximates the entropy rate of a packet trace. Considering several real-world traces, we show that trace complexity can provide unique insights into the characteristics of various applications. Based on our approach,

Further Reading

Static DAN



Demand-Aware Network Designs of Bounded Degree

Chen Avin **Kaushik Mondal** **Stefan Schmid**

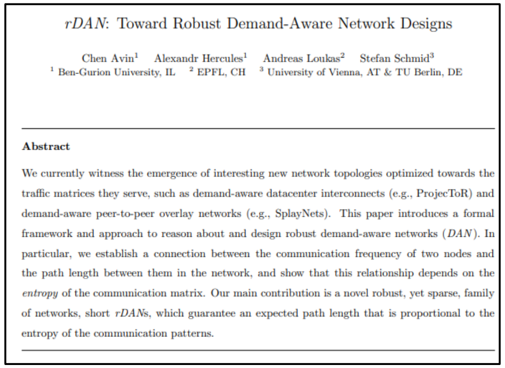
Abstract Traditionally, networks such as datacenter interconnects are designed to optimize worst-case performance under *arbitrary* traffic patterns. Such network designs can however be far from optimal when considering the *actual* workloads and traffic patterns which they serve. This insight led to the development of demand-aware datacenter interconnects which can be reconfigured depending on the workload.

Motivated by these trends, this paper initiates the algorithmic study of demand-aware networks (`DANs`), and in particular the design of bounded-degree networks. The inputs to the network design problem are a discrete communication request distribution, $\mathcal{D}$, defined over communicating pairs from the node set V, and a bound, Δ, on the maximum degree. In turn, our objective is to design an (undirected) demand-aware network $N = (V, E)$ of bounded-degree Δ, which provides short routing paths between frequently communicating nodes distributed across N. In particular, the designed network should minimize the *expected path length* on N (with respect to $\mathcal{D}$), which is a basic measure of the

1 Introduction

The problem studied in this paper is motivated by the advent of more flexible datacenter interconnects, such as ProjecToR [29,31]. These interconnects aim to overcome a fundamental drawback of traditional datacenter network designs: the fact that network designers must decide *in advance* on how much capacity to provision between electrical packet switches, e.g., between Top-of-Rack (ToR) switches in datacenters. This leads to an undesirable tradeoff [42]: either capacity is over-provisioned and therefore the interconnect expensive (e.g., a fat-tree provides full-bisection bandwidth), or one may risk congestion, resulting in a poor cloud application performance. Accordingly, systems such as ProjecToR provide a reconfigurable interconnect, allowing to establish links flexibly and in a *demand-aware manner*. For example, direct links or at least short communication paths can be established between frequently communicating ToR switches. Such links can be implemented using a bounded number of lasers, mirrors,

Robust DAN



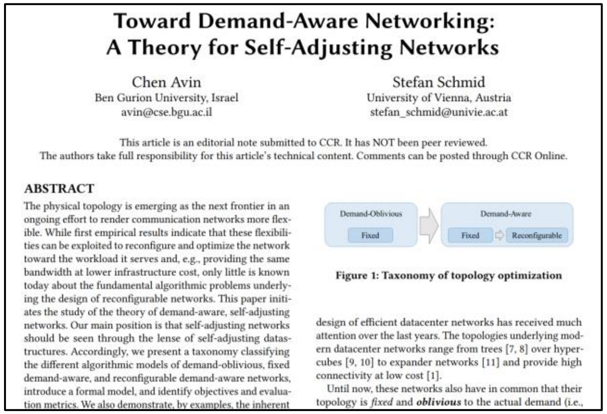
rDAN: Toward Robust Demand-Aware Network Designs

Chen Avin[1] Alexandr Hercules[1] Andreas Loukas[2] Stefan Schmid[3]
[1] Ben-Gurion University, IL [2] EPFL, CH [3] University of Vienna, AT & TU Berlin, DE

Abstract

We currently witness the emergence of interesting new network topologies optimized towards the traffic matrices they serve, such as demand-aware datacenter interconnects (e.g., ProjecToR) and demand-aware peer-to-peer overlay networks (e.g., SplayNets). This paper introduces a formal framework and approach to reason about and design robust demand-aware networks (*DAN*). In particular, we establish a connection between the communication frequency of two nodes and the path length between them in the network, and show that this relationship depends on the *entropy* of the communication matrix. Our main contribution is a novel robust, yet sparse, family of networks, short *rDAN*s, which guarantee an expected path length that is proportional to the entropy of the communication patterns.

Overview: Models



Toward Demand-Aware Networking: A Theory for Self-Adjusting Networks

Chen Avin
Ben Gurion University, Israel
avin@cse.bgu.ac.il

Stefan Schmid
University of Vienna, Austria
stefan_schmid@univie.ac.at

This article is an editorial note submitted to CCR. It has NOT been peer reviewed.
The authors take full responsibility for this article's technical content. Comments can be posted through CCR Online.

ABSTRACT

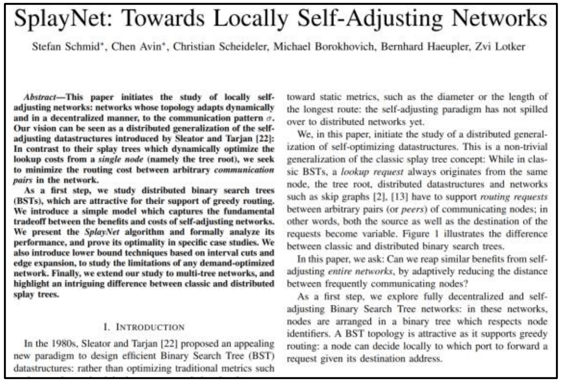
The physical topology is emerging as the next frontier in an ongoing effort to render communication networks more flexible. While first empirical results indicate that these flexibilities can be exploited to reconfigure and optimize the network toward the workload it serves and, e.g., providing the same bandwidth at lower infrastructure cost, only little is known today about the fundamental algorithmic problems underlying the design of reconfigurable networks. This paper initiates the study of the theory of demand-aware, self-adjusting networks. Our main position is that self-adjusting networks should be seen through the lense of self-adjusting datastructures. Accordingly, we present a taxonomy classifying the different algorithmic models of demand-oblivious, fixed demand-aware, and reconfigurable demand-aware networks, introduce a formal model, and identify objectives and evaluation metrics. We also demonstrate, by examples, the inherent

Figure 1: Taxonomy of topology optimization

design of efficient datacenter networks has received much attention over the last years. The topologies underlying modern datacenter networks range from trees [7, 8] over hypercubes [9, 10] to expander networks [11] and provide high connectivity at low cost [1].

Until now, these networks also have in common that their topology is *fixed* and ***oblivious*** to the actual demand (i.e.,

Dynamic DAN



SplayNet: Towards Locally Self-Adjusting Networks

Stefan Schmid*, Chen Avin*, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler, Zvi Lotker

***Abstract*—This paper initiates the study of locally self-adjusting networks: networks whose topology adapts dynamically and in a decentralized manner, to the communication pattern σ. Our vision can be seen as a distributed generalization of the self-adjusting datastructures introduced by Sleator and Tarjan [22]: In contrast to their splay trees which dynamically optimize the lookup costs from a *single node* (namely the tree root), we seek to minimize the routing cost between arbitrary *communication pairs* in the network.**

As a first step, we study distributed binary search trees (BSTs), which are attractive for their support of greedy routing. We introduce a simple model which captures the fundamental tradeoff between the benefits and costs of self-adjusting networks. We present the *SplayNet* algorithm and formally analyze its performance, and prove its optimality in specific case studies. We also introduce lower bound techniques based on interval cuts and edge expansion, to study the limitations of any demand-optimized network. Finally, we extend our study to multi-tree networks, and highlight an intriguing difference between classic and distributed splay trees.

I. INTRODUCTION

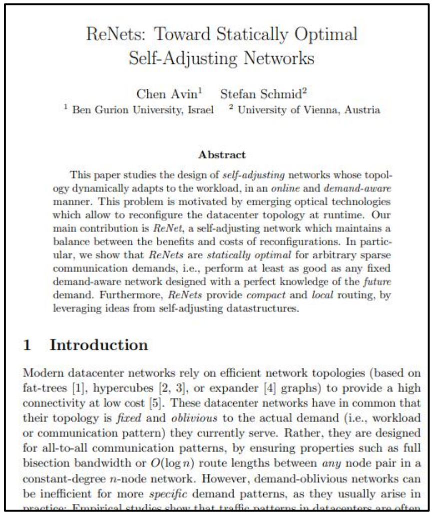
In the 1980s, Sleator and Tarjan [22] proposed an appealing new paradigm to design efficient Binary Search Tree (BST) datastructures: rather than optimizing traditional metrics such toward static metrics, such as the diameter or the length of the longest route: the self-adjusting paradigm has not spilled over to distributed networks yet.

We, in this paper, initiate the study of a distributed generalization of self-optimizing datastructures. This is a non-trivial generalization of the classic splay tree concept: While in classic BSTs, a *lookup request* always originates from the same node, the tree root, distributed datastructures and networks such as skip graphs [2], [13] have to support *routing requests* between arbitrary pairs (or *peers*) of communicating nodes; in other words, both the source as well as the destination of the requests become variable. Figure 1 illustrates the difference between classic and distributed binary search trees.

In this paper, we ask: Can we reap similar benefits from self-adjusting *entire networks*, by adaptively reducing the distance between frequently communicating nodes?

As a first step, we explore fully decentralized and self-adjusting Binary Search Tree networks: in these networks, nodes are arranged in a binary tree which respects node identifiers. A BST topology is attractive as it supports greedy routing: a node can decide locally to which port to forward a request given its destination address.

Static Optimality



ReNets: Toward Statically Optimal Self-Adjusting Networks

Chen Avin[1] Stefan Schmid[2]
[1] Ben Gurion University, Israel [2] University of Vienna, Austria

Abstract

This paper studies the design of *self-adjusting* networks whose topology dynamically adapts to the workload, in an *online* and *demand-aware* manner. This problem is motivated by emerging optical technologies which allow to reconfigure the datacenter topology at runtime. Our main contribution is *ReNet*, a self-adjusting network which maintains a balance between the benefits and costs of reconfigurations. In particular, we show that *ReNets* are *statically optimal* for arbitrary sparse communication demands, i.e., perform at least as good as any fixed demand-aware network designed with a perfect knowledge of the *future* demand. Furthermore, *ReNets* provide *compact* and *local* routing, by leveraging ideas from self-adjusting datastructures.

1 Introduction

Modern datacenter networks rely on efficient network topologies (based on fat-trees [1], hypercubes [2, 3], or expander [4] graphs) to provide a high connectivity at low cost [5]. These datacenter networks have in common that their topology is *fixed* and *oblivious* to the actual demand (i.e., workload or communication pattern) they currently serve. Rather, they are designed for all-to-all communication patterns, by ensuring properties such as full bisection bandwidth or $O(\log n)$ route lengths between *any* node pair in a constant-degree n-node network. However, demand-oblivious networks can be inefficient for more *specific* demand patterns, as they usually arise in practice: Empirical studies show that traffic patterns in datacenters are often

Concurrent DANs

CBNet: Minimizing Adjustments in Concurrent Demand-Aware Tree Networks

Otavio Augusto de Oliveira Souza[1] Olga Goussevskaia[1] Stefan Schmid[2]
[1] Universidade Federal de Minas Gerais, Brazil [2] University of Vienna, Austria

***Abstract*—This paper studies the design of demand-aware network topologies: networks that dynamically adapt themselves toward the demand they currently serve, in an online manner. While demand-aware networks may be significantly more efficient than demand-oblivious networks, frequent adjustments are still costly. Furthermore, a centralized controller of such networks may become a bottleneck.**

CBNet is based on concepts from self-adjusting data struc
tures, and in particular, CBTrees [12]. CBNet gradually adapt
the network topology toward the communication pattern in a
online manner, i.e., without previous knowledge of the deman
distribution. At the same time, *bidirectional semi-splaying* an
counters are used to maintain state, minimize reconfiguratio

Selected References

On the Complexity of Traffic Traces and Implications
Chen Avin, Manya Ghobadi, Chen Griner, and Stefan Schmid.
ACM SIGMETRICS, Boston, Massachusetts, USA, June 2020.

Survey of Reconfigurable Data Center Networks: Enablers, Algorithms, Complexity
Klaus-Tycho Foerster and Stefan Schmid.
SIGACT News, June 2019.

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

Dynamically Optimal Self-Adjusting Single-Source Tree Networks
Chen Avin, Kaushik Mondal, and Stefan Schmid.
14th Latin American Theoretical Informatics Symposium (LATIN), University of Sao Paulo, Sao Paulo, Brazil, May 2020.

Demand-Aware Network Design with Minimal Congestion and Route Lengths
Chen Avin, Kaushik Mondal, and Stefan Schmid.
38th IEEE Conference on Computer Communications (**INFOCOM**), Paris, France, April 2019.

Distributed Self-Adjusting Tree Networks
Bruna Peres, Otavio Augusto de Oliveira Souza, Olga Goussevskaia, Chen Avin, and Stefan Schmid.
38th IEEE Conference on Computer Communications (**INFOCOM**), Paris, France, April 2019.

Efficient Non-Segregated Routing for Reconfigurable Demand-Aware Networks
Thomas Fenz, Klaus-Tycho Foerster, Stefan Schmid, and Anaïs Villedieu.
IFIP Networking, Warsaw, Poland, May 2019.

DaRTree: Deadline-Aware Multicast Transfers in Reconfigurable Wide-Area Networks
Long Luo, Klaus-Tycho Foerster, Stefan Schmid, and Hongfang Yu.
IEEE/ACM International Symposium on Quality of Service (**IWQoS**), Phoenix, Arizona, USA, June 2019.

Demand-Aware Network Designs of Bounded Degree
Chen Avin, Kaushik Mondal, and Stefan Schmid.
31st International Symposium on Distributed Computing (**DISC**), Vienna, Austria, October 2017.

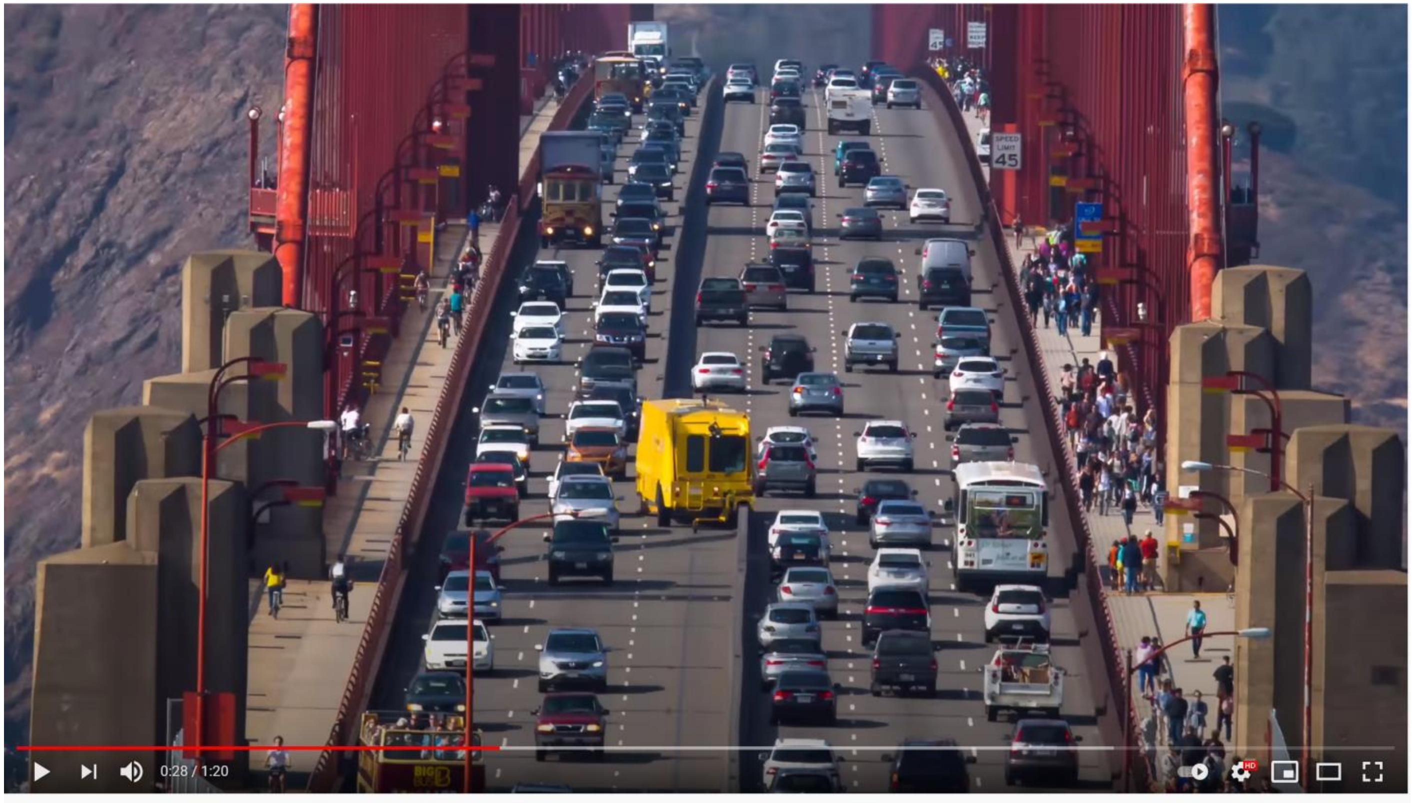
SplayNet: Towards Locally Self-Adjusting Networks
Stefan Schmid, Chen Avin, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler, and Zvi Lotker.
IEEE/ACM Transactions on Networking (**TON**), Volume 24, Issue 3, 2016. Early version: IEEE **IPDPS** 2013.

Characterizing the Algorithmic Complexity of Reconfigurable Data Center Architectures
Klaus-Tycho Foerster, Monia Ghobadi, and Stefan Schmid.
ACM/IEEE Symposium on Architectures for Networking and Communications Systems (**ANCS**), Ithaca, New York, USA, July 2018.

Bonus Material

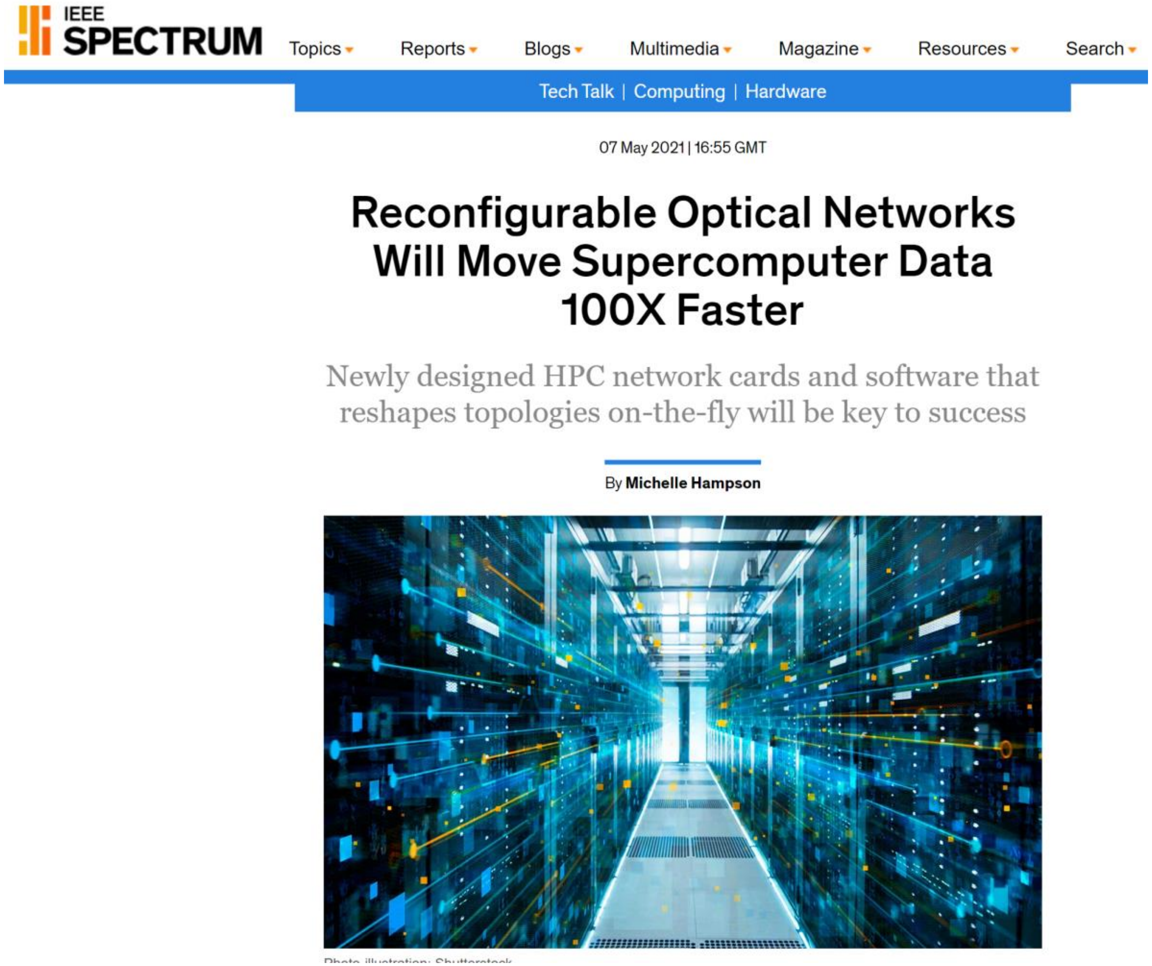
Hogwarts Stair

Bonus Material



Golden Gate Zipper

Bonus Material



Tech Talk | Computing | Hardware

07 May 2021 | 16:55 GMT

Reconfigurable Optical Networks Will Move Supercomputer Data 100X Faster

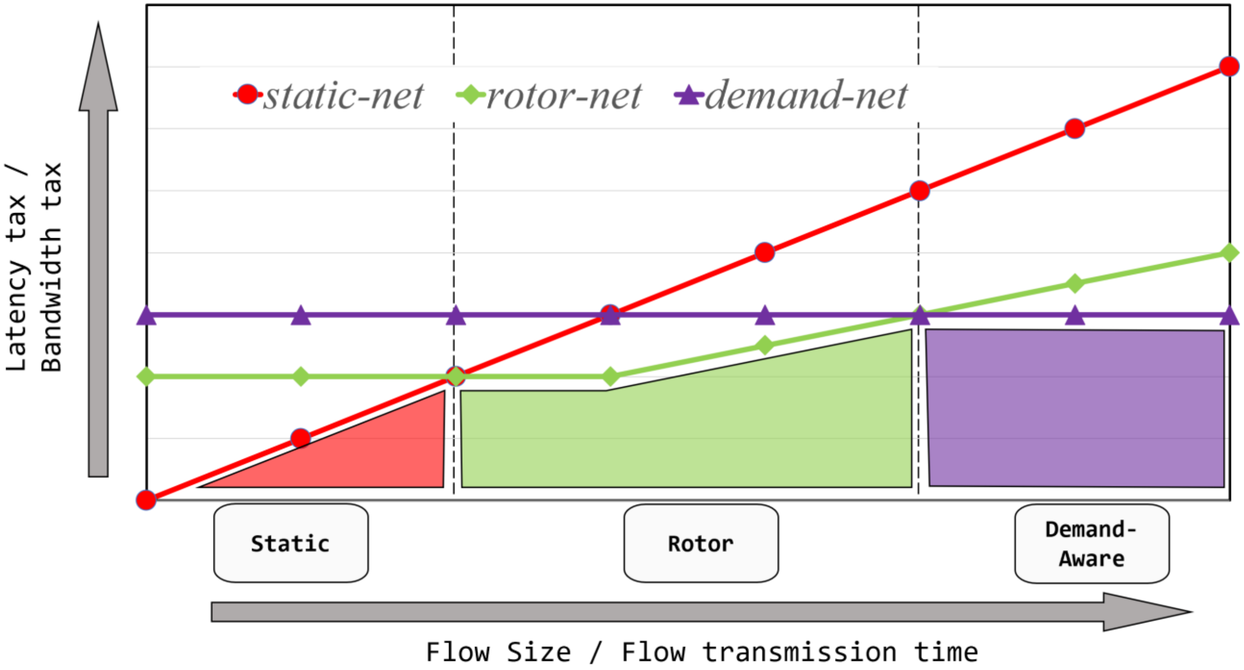
Newly designed HPC network cards and software that reshapes topologies on-the-fly will be key to success

By **Michelle Hampson**

In HPC

Matching Topologies

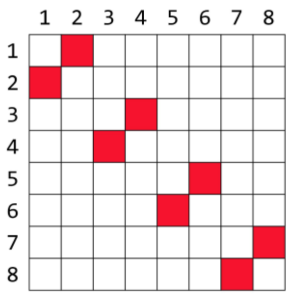
- **Static** is good for small flows, but then incurs latency tax
- **Rotor** is good for medium flows, but cannot provide low latency for small flows and cannot be optimized towards elephant flows
- **Demand-aware** topology can adapt toward really large flows



Throughput Analysis

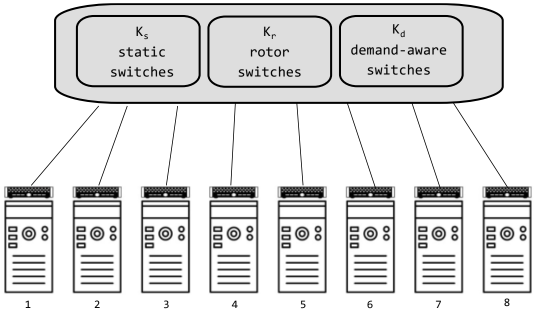
Demand Matrix

T

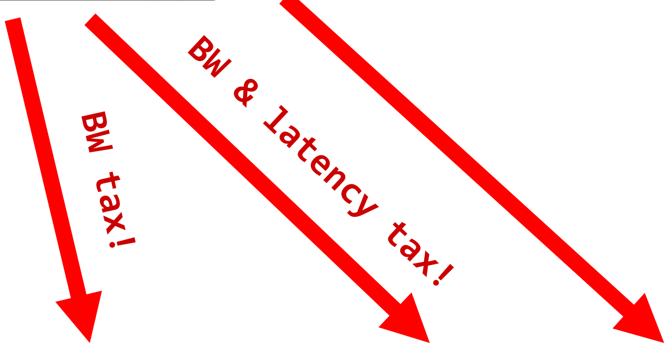


	1	2	3	4	5	6	7	8
1		🟥						
2	🟥							
3				🟥				
4			🟥					
5						🟥		
6					🟥			
7								🟥
8							🟥	

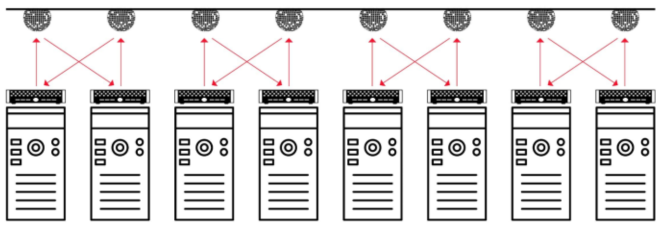
$\times\ \theta(T)$ ⇒



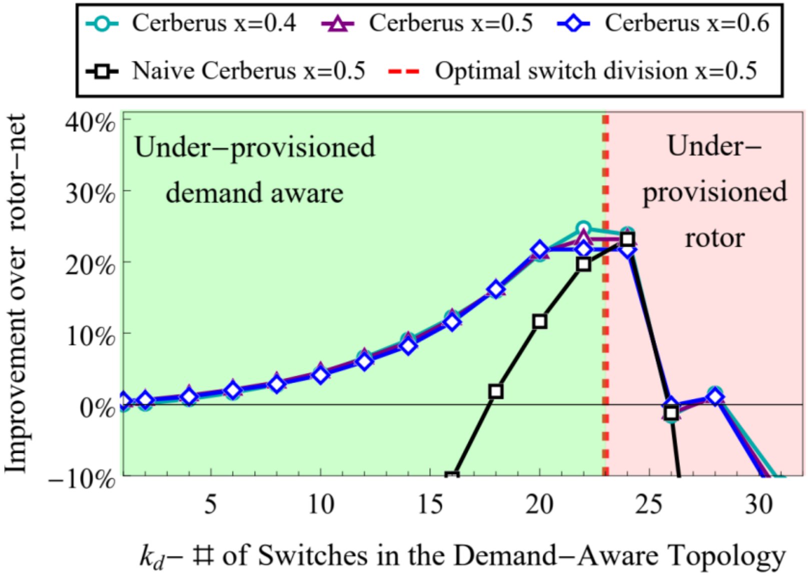
θ^* worst case T



	expander-net	*rotor-net*	CERBERUS
BW-Tax	✓	✓	✗
LT-Tax	✗	✓	✓
$\theta(T)$	Thm 2	Thm 3	Thm 5
θ^*	0.53	0.45	Open
Datamining	0.53	0.6	0.8 (+33%)
Permutation	0.53	0.45	≈ 1 (+88%)
Case Study	0.53	0.66	0.9 (+36%)



Sensitivity Analysis



Data mining workload

Question 1:

How to Quantify such “Structure” in the Demand?

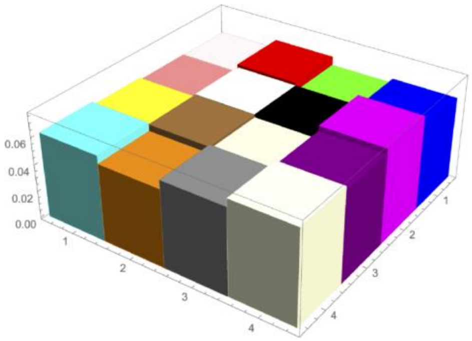
Intuition

Which demand has more structure?

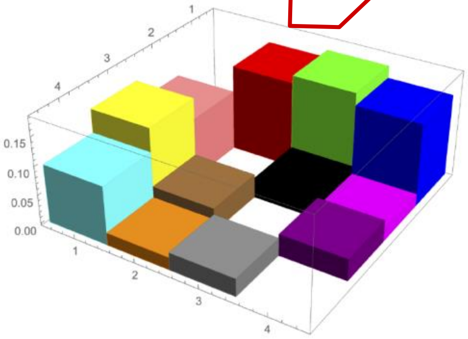
⇢ Traffic matrices of two different distributed ML applications

- → GPU-to-GPU

Color = communication pair



VS



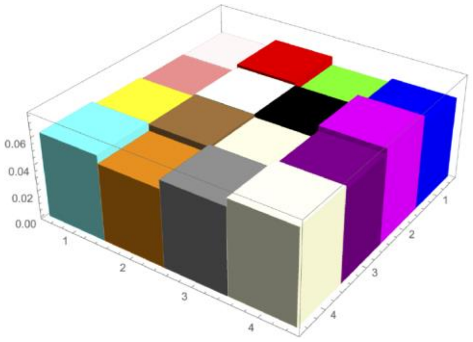
Intuition

Which demand has more structure?

⇢ Traffic matrices of two different distributed ML applications

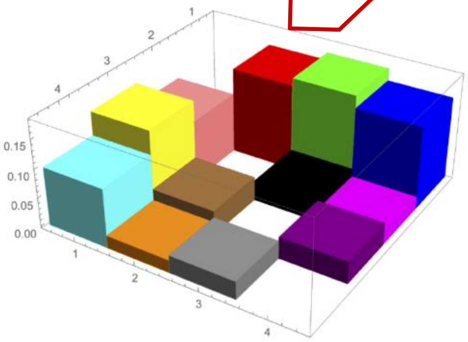
- → GPU-to-GPU

Color = communication pair



More uniform

VS



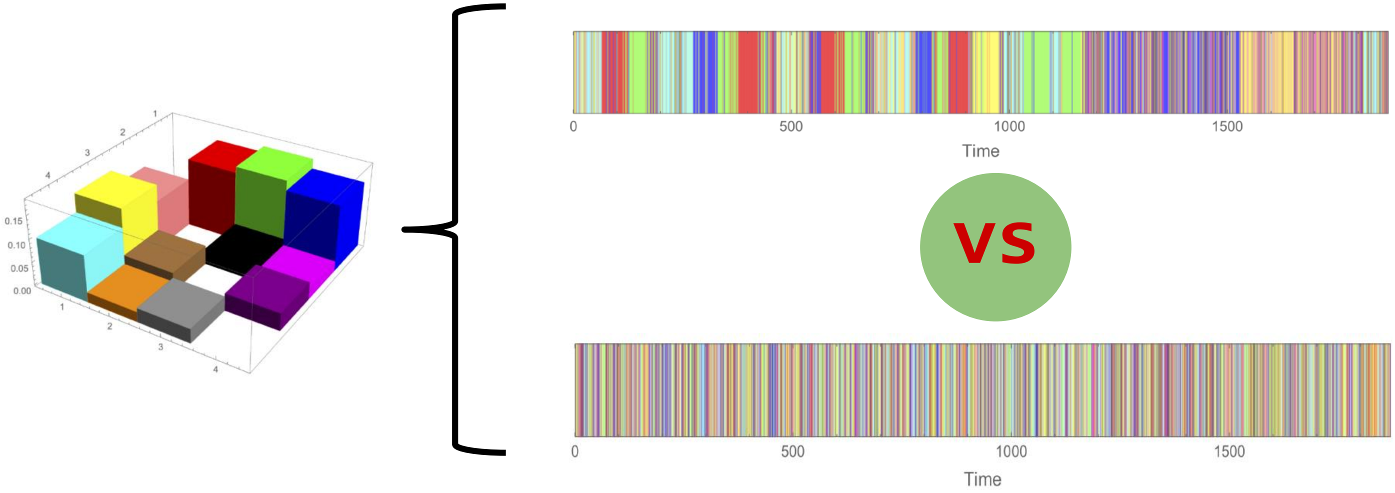
More structure

Intuition

Spatial vs temporal structure

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

- ⇢ Which one has more structure?

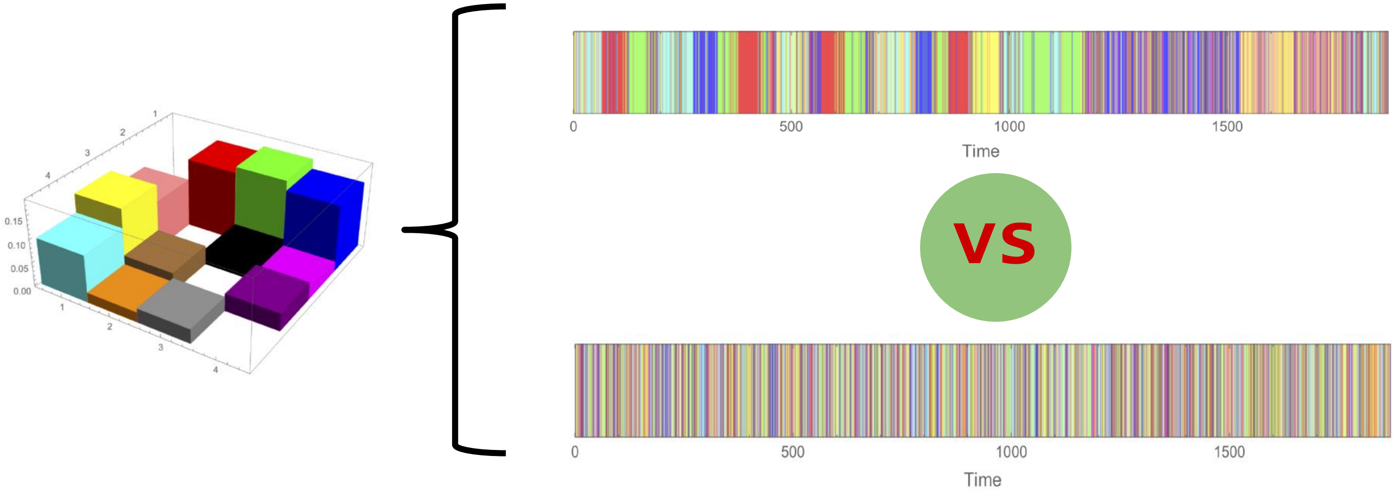


Intuition

Spatial vs temporal structure

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

- ⇢ Which one has more structure?

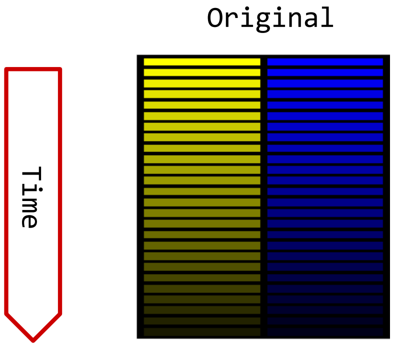


Systematically?

Trace Complexity

Information-Theoretic Approach

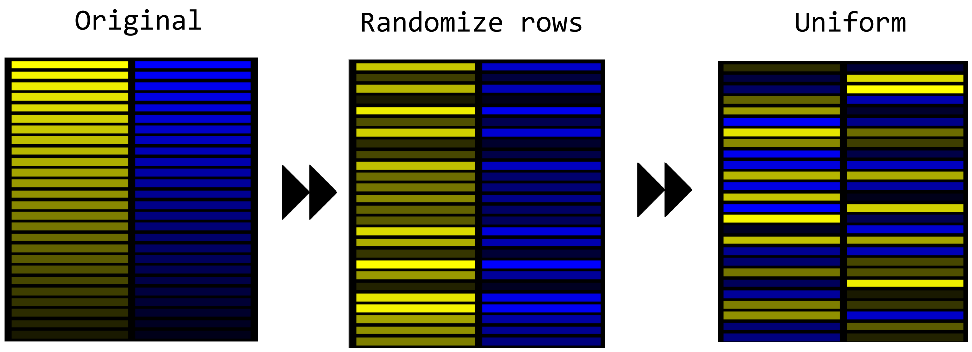
“Shuffle&Compress”



Trace Complexity

Information-Theoretic Approach

“Shuffle&Compress”



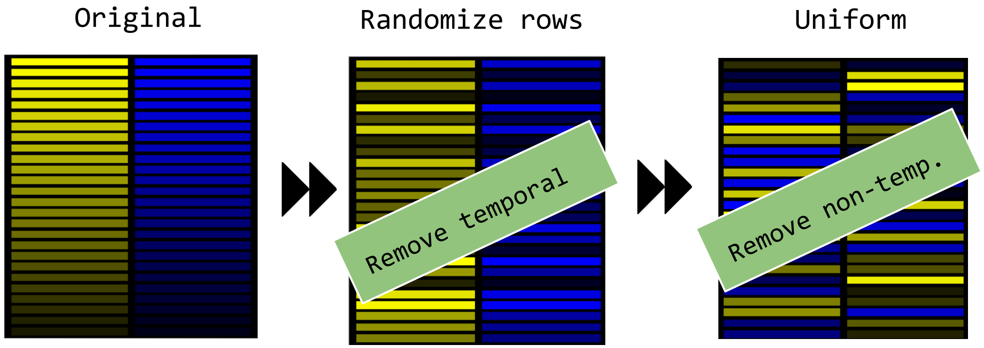
Increasing complexity (systematically randomized)

More structure (compresses better)

Trace Complexity

Information-Theoretic Approach

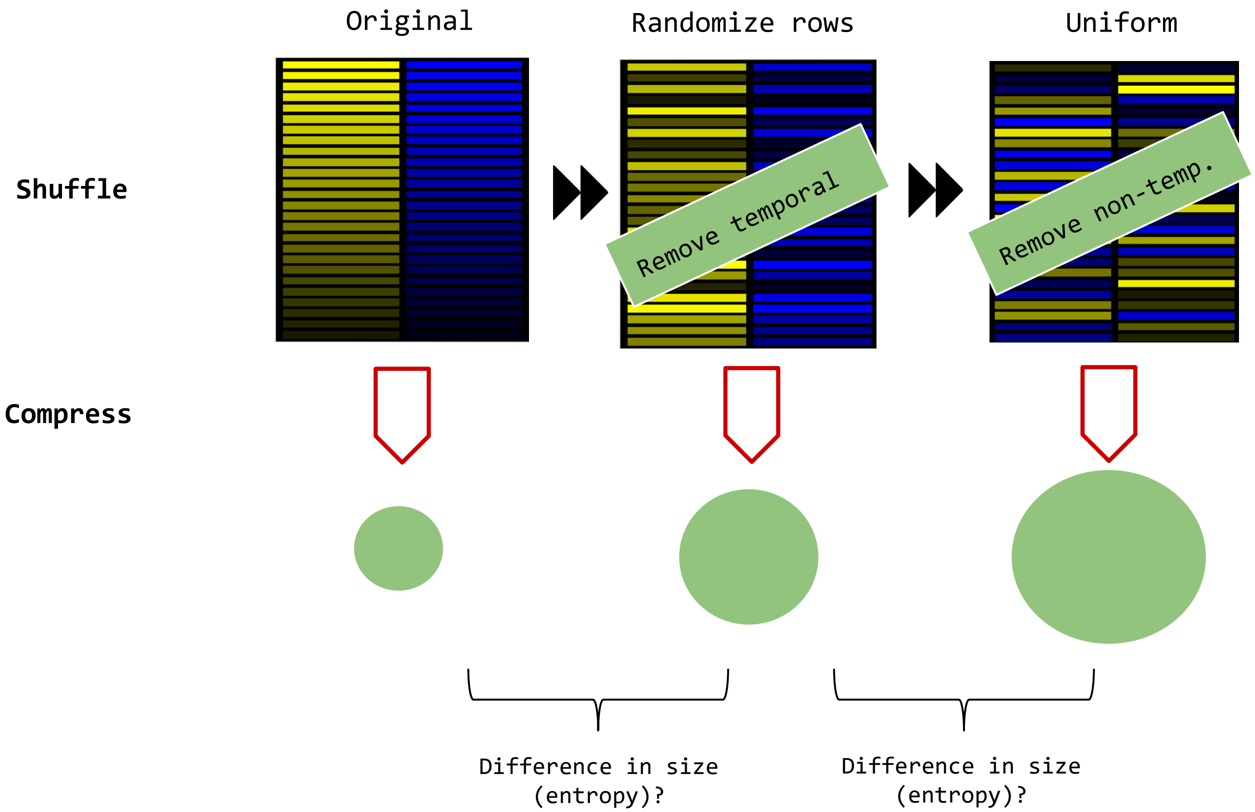
“Shuffle&Compress”



Trace Complexity

Information-Theoretic Approach

“Shuffle&Compress”



Original

Randomize rows

Uniform

Shuffle

Remove temporal

Remove non-temp.

Compress

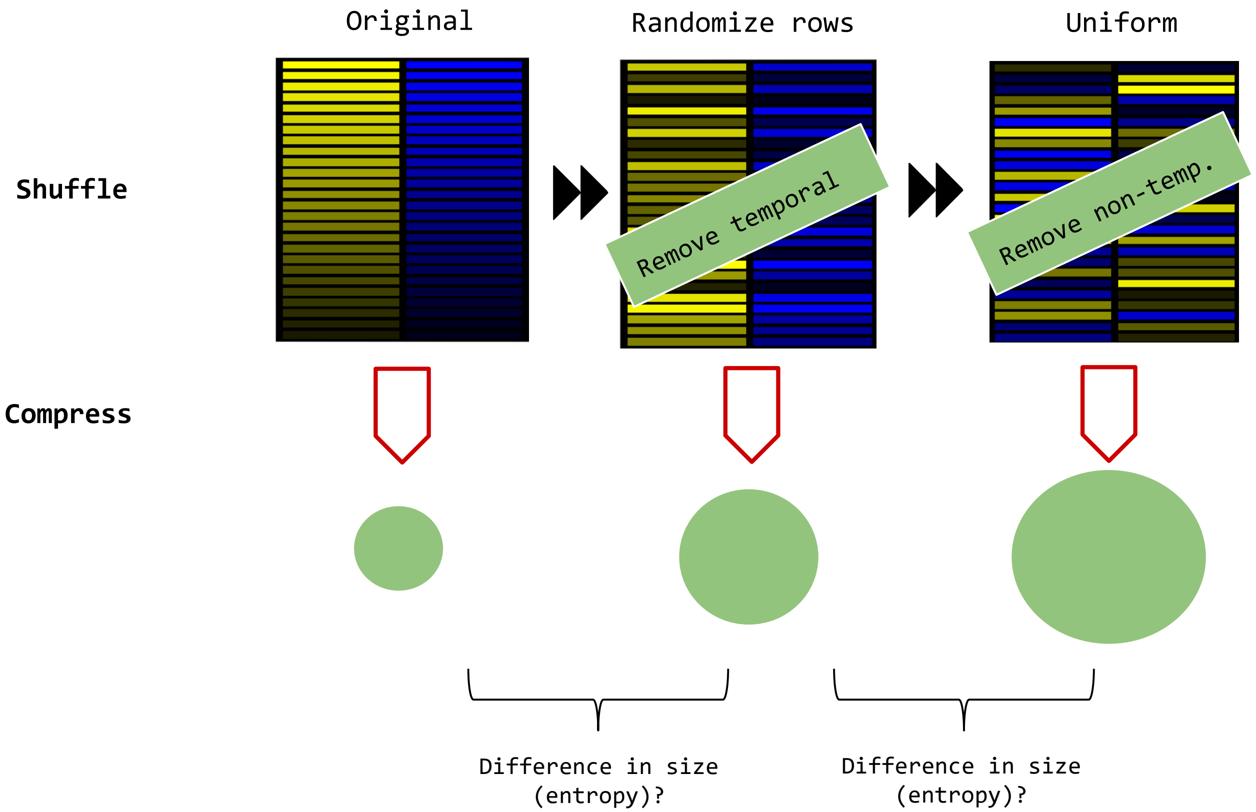
Difference in size (entropy)?

Difference in size (entropy)?

Trace Complexity

Information-Theoretic Approach

“Shuffle&Compress”



Original

Randomize rows

Uniform

Shuffle

Remove temporal

Remove non-temp.

Compress

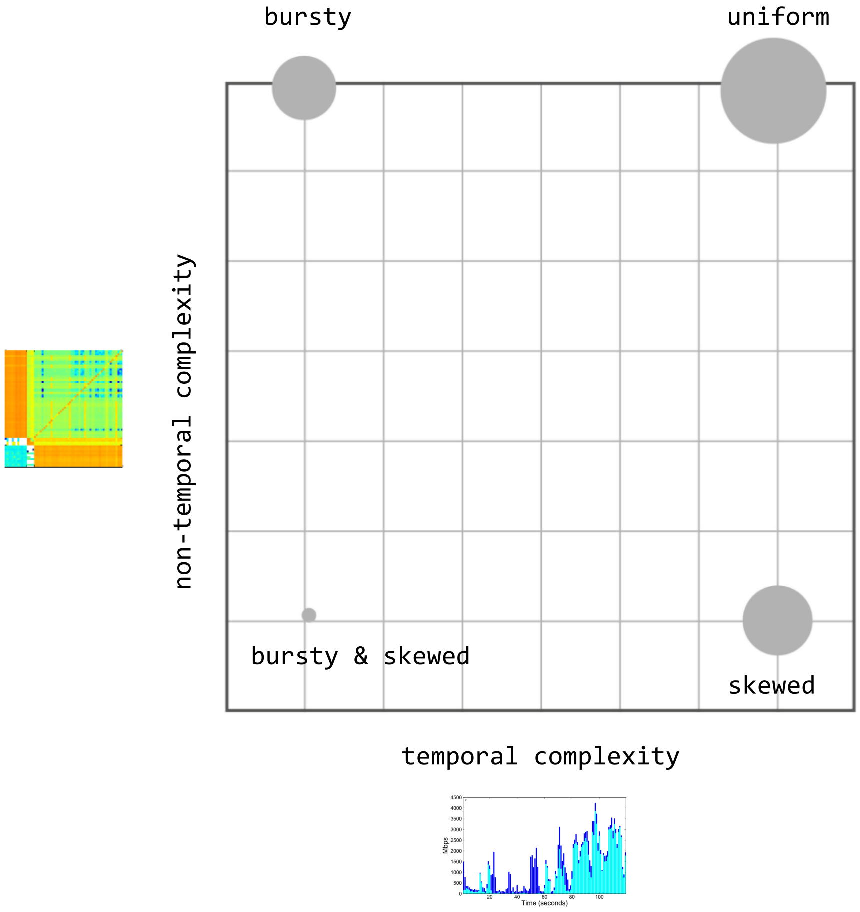
Difference in size
(entropy)?

Difference in size
(entropy)?

Can be used to define 2-dimensional complexity map!

Our Methodology

Complexity Map


bursty

uniform

bursty & skewed

skewed

non-temporal complexity

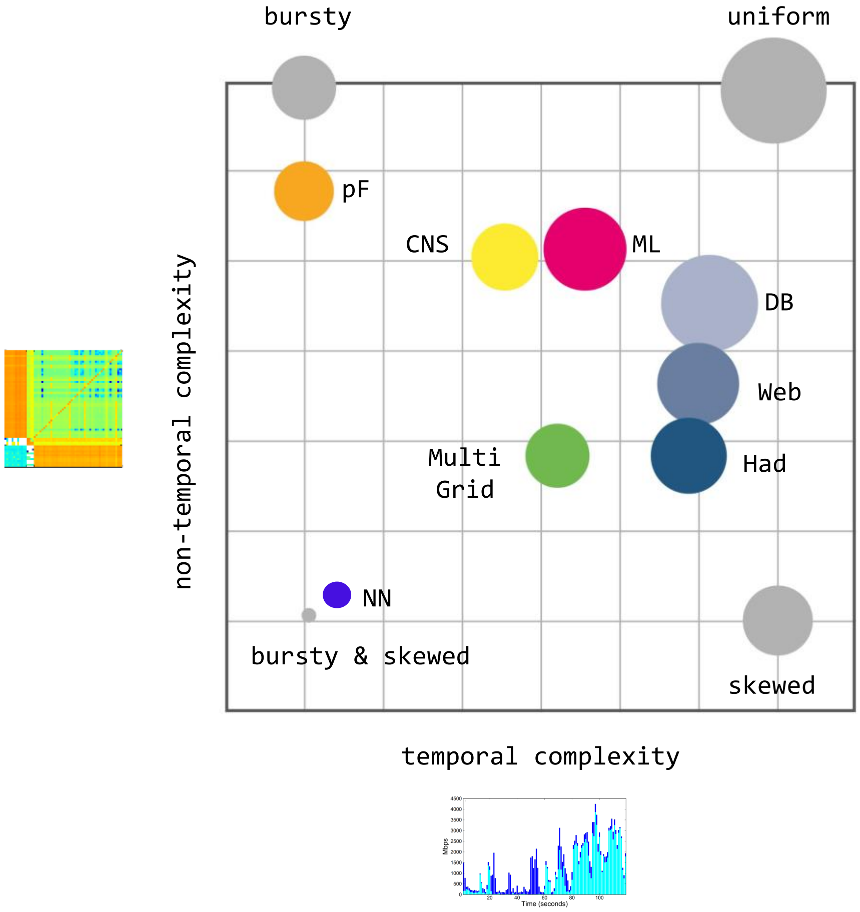
temporal complexity

No structure

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

Our Methodology

Complexity Map



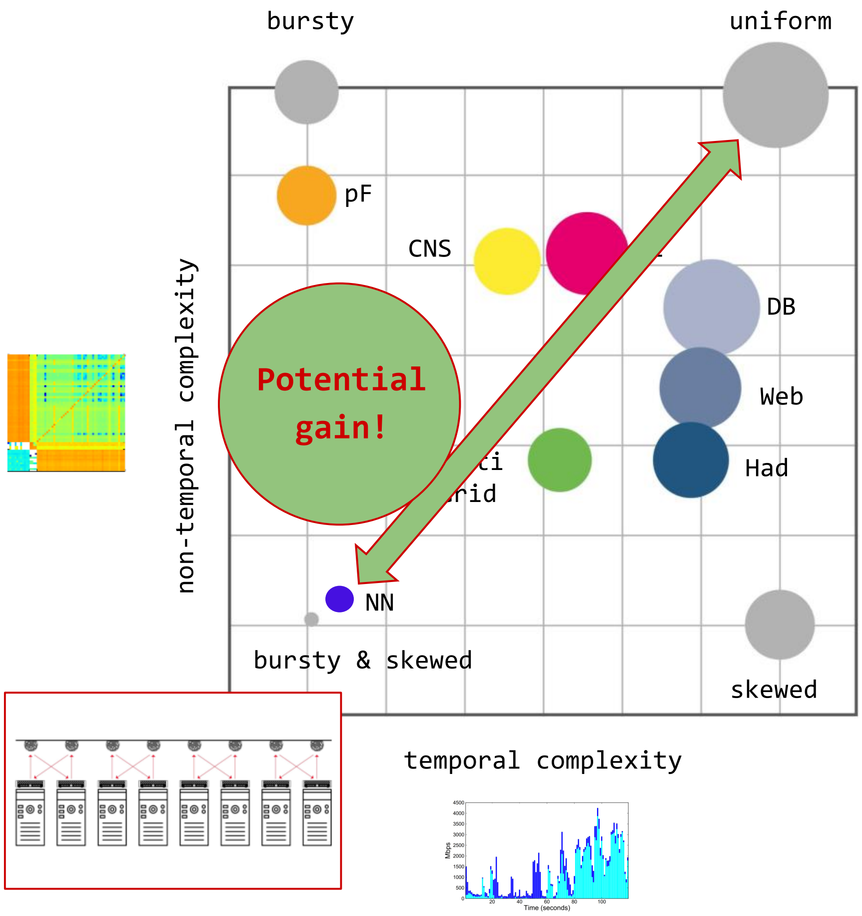
No structure

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

Different structures!

Our Methodology

Complexity Map



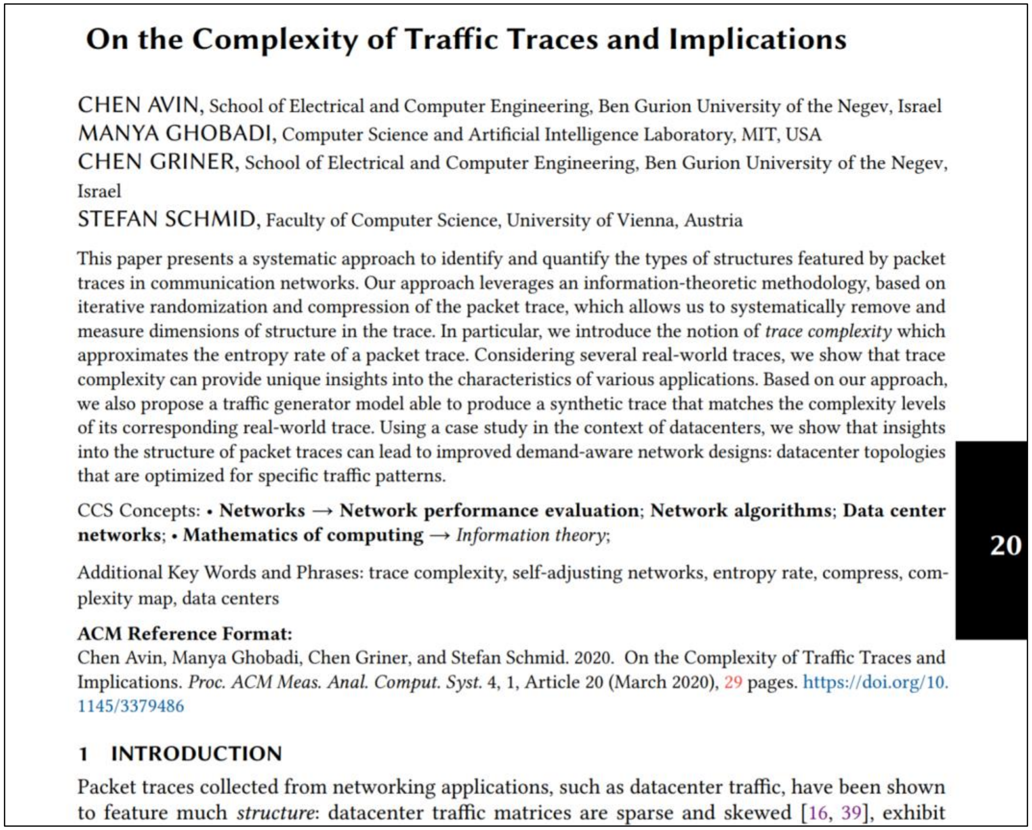
Potential gain!

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

Different structures!

Further Reading

ACM SIGMETRICS 2020



On the Complexity of Traffic Traces and Implications

CHEN AVIN, School of Electrical and Computer Engineering, Ben Gurion University of the Negev, Israel
MANYA GHOBADI, Computer Science and Artificial Intelligence Laboratory, MIT, USA
CHEN GRINER, School of Electrical and Computer Engineering, Ben Gurion University of the Negev, Israel
STEFAN SCHMID, Faculty of Computer Science, University of Vienna, Austria

This paper presents a systematic approach to identify and quantify the types of structures featured by packet traces in communication networks. Our approach leverages an information-theoretic methodology, based on iterative randomization and compression of the packet trace, which allows us to systematically remove and measure dimensions of structure in the trace. In particular, we introduce the notion of *trace complexity* which approximates the entropy rate of a packet trace. Considering several real-world traces, we show that trace complexity can provide unique insights into the characteristics of various applications. Based on our approach, we also propose a traffic generator model able to produce a synthetic trace that matches the complexity levels of its corresponding real-world trace. Using a case study in the context of datacenters, we show that insights into the structure of packet traces can lead to improved demand-aware network designs: datacenter topologies that are optimized for specific traffic patterns.

CCS Concepts: • **Networks** → **Network performance evaluation**; **Network algorithms**; **Data center networks**; • **Mathematics of computing** → *Information theory*;

Additional Key Words and Phrases: trace complexity, self-adjusting networks, entropy rate, compress, complexity map, data centers

ACM Reference Format:
Chen Avin, Manya Ghobadi, Chen Griner, and Stefan Schmid. 2020. On the Complexity of Traffic Traces and Implications. *Proc. ACM Meas. Anal. Comput. Syst.* 4, 1, Article 20 (March 2020), 29 pages. https://doi.org/10.1145/3379486

1 INTRODUCTION

Packet traces collected from networking applications, such as datacenter traffic, have been shown to feature much *structure*: datacenter traffic matrices are sparse and skewed [16, 39], exhibit

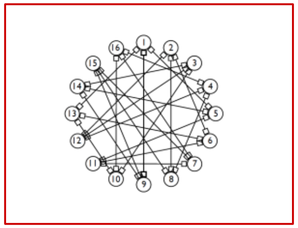
Question 2:

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

A first insight: entropy of the demand.

Models and Connection to Datastructures & Coding

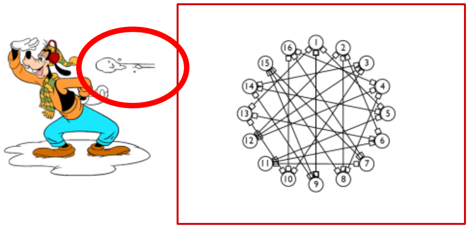
Oblivious networks
(worst-case traffic)



More structure: **lower routing cost**

Models and Connection to Datastructures & Coding

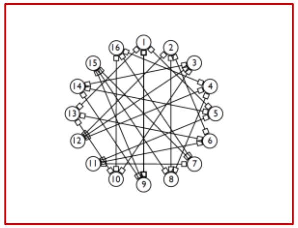
Oblivious networks
(worst-case traffic)



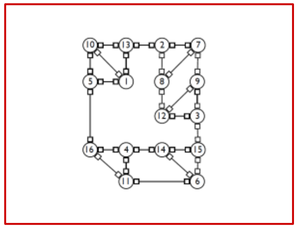
More structure: **lower routing cost**

Models and Connection to Datastructures & Coding

Oblivious networks
(worst-case traffic)



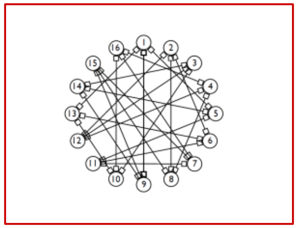
Demand-aware networks
(spatial structure)



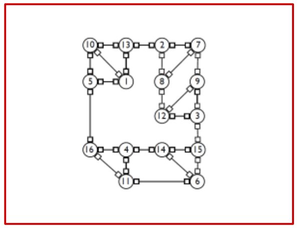
More structure: **lower routing cost**

Models and Connection to Datastructures & Coding

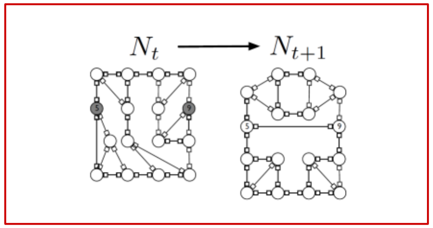
Oblivious networks
(worst-case traffic)



Demand-aware networks
(spatial structure)



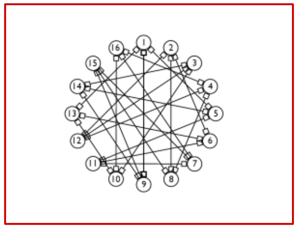
Self-adjusting networks
(temporal structure)



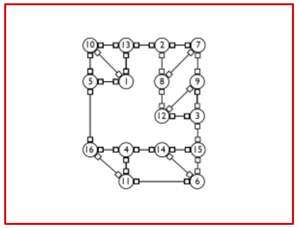
More structure: **lower routing cost**

Models and Connection to Datastructures & Coding

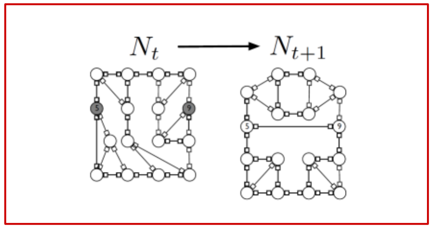
Oblivious networks
(worst-case traffic)



Demand-aware networks
(spatial structure)

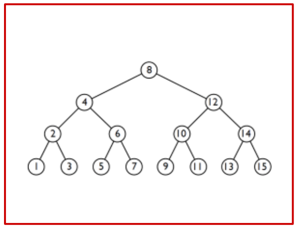


Self-adjusting networks
(temporal structure)

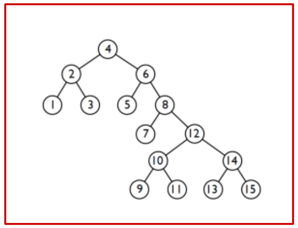


More structure: **lower routing cost**

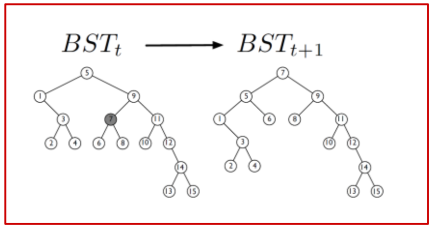
Traditional BST
(Worst-case coding)



Demand-aware BST
(Huffman coding)



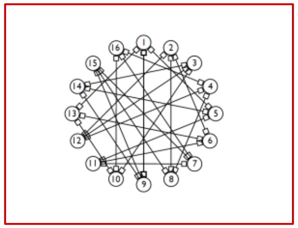
Self-adjusting BST
(Dynamic Huffman coding)



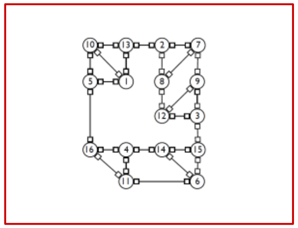
More structure: improved **access cost** / shorter **codes**

Models and Connection to Datastructures & Coding

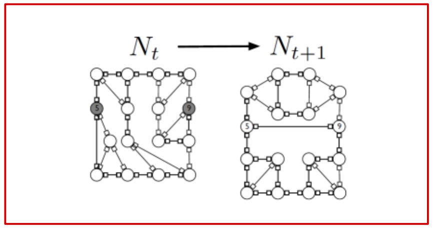
Oblivious networks
(worst-case traffic)



Demand-aware networks
(spatial structure)

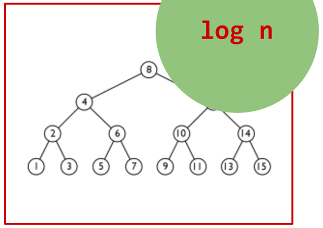


Self-adjusting networks
(temporal structure)



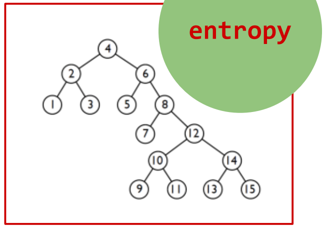
More structure: **lower routing cost**

Traditional BST
(Worst-case



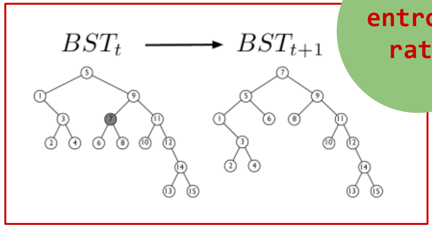
log n

Demand-aware BST
(Huffman c



entropy

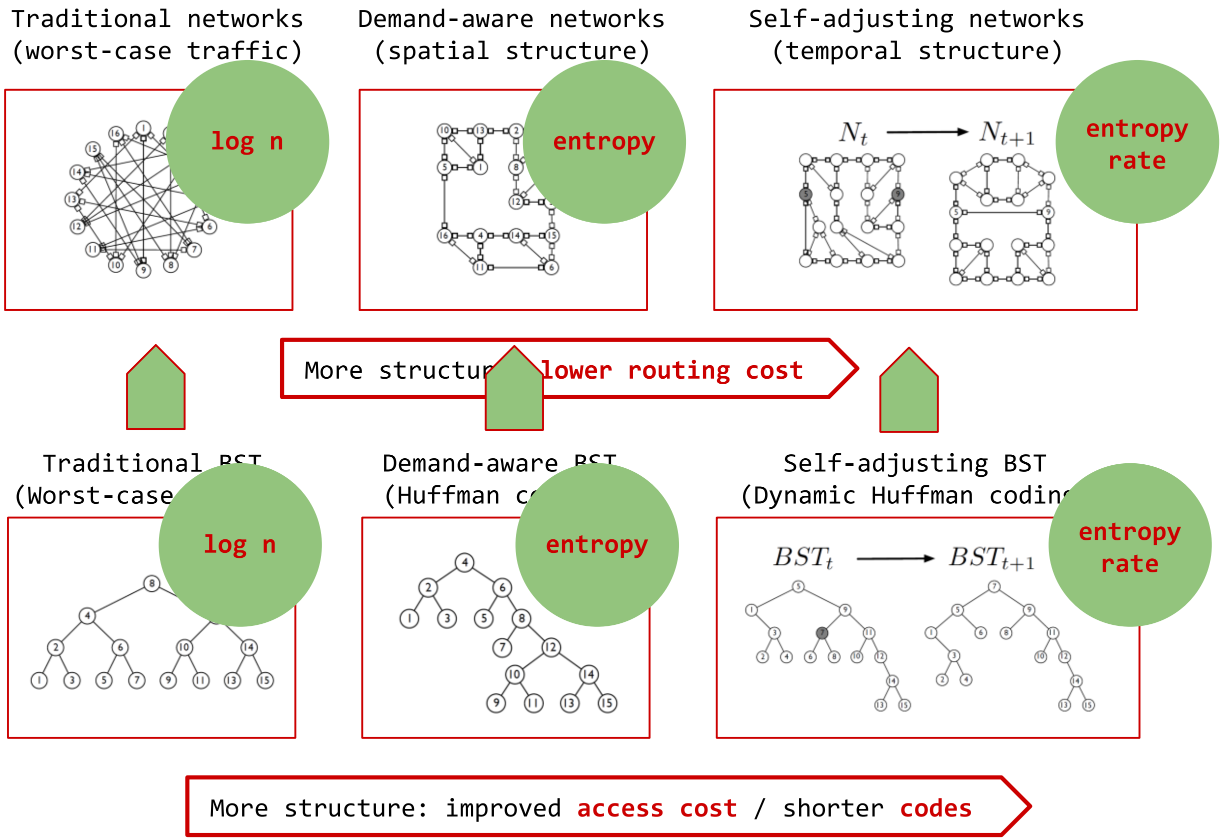
Self-adjusting BST
(Dynamic Huffman codin



entropy rate

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

Models and Connection to Datastructures & Coding



Traditional networks (worst-case traffic)

log n

Demand-aware networks (spatial structure)

entropy

Self-adjusting networks (temporal structure)

entropy rate

More structur **lower routing cost**

Traditional BST
(Worst-case

log n

Demand-aware BST
(Huffman c

entropy

Self-adjusting BST
(Dynamic Huffman codin

entropy rate

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

More than
an analogy!

Generalize methodology:

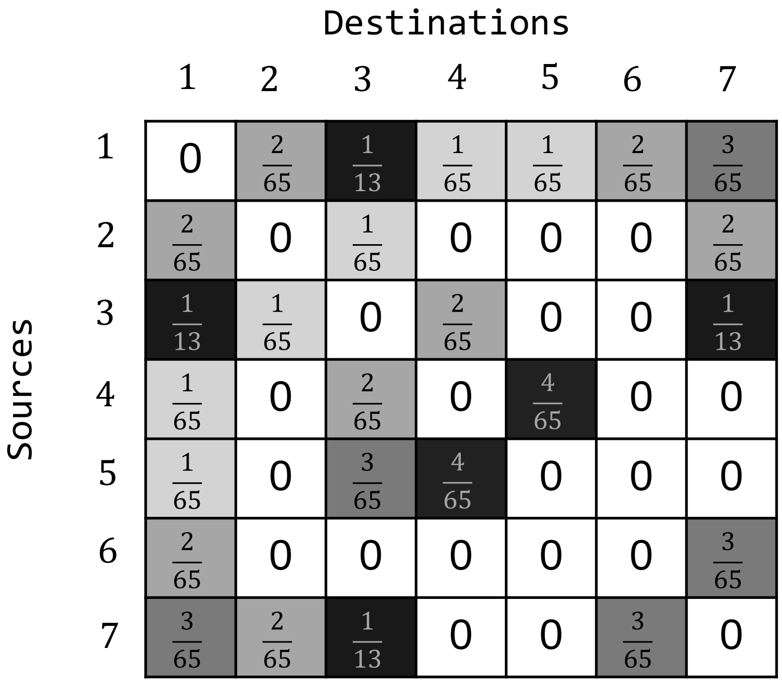
... and transfer entropy bounds and algorithms of data-structures to networks.

First result:

Demand-aware networks of asymptotically optimal route lengths.

Case Study “Route Lengths”

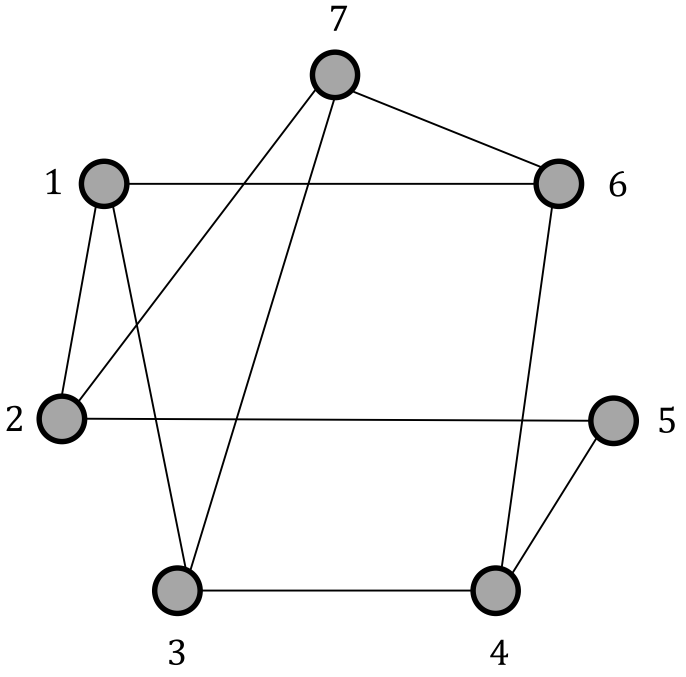
Constant-Degree Demand-Aware Network



Destinations

Sources

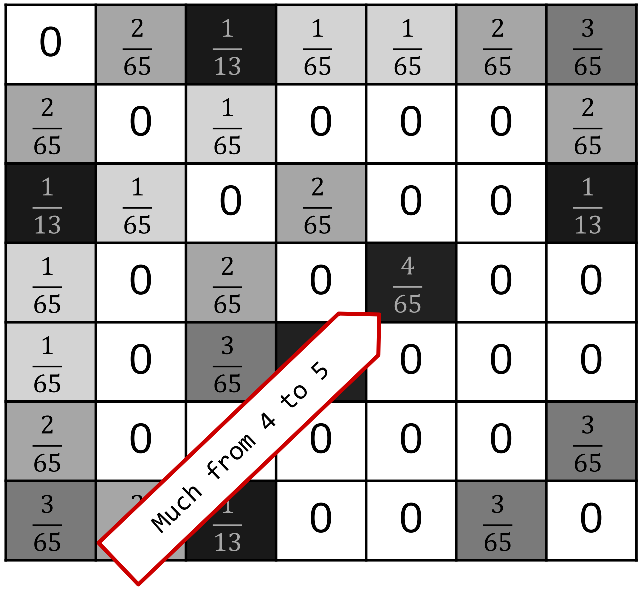
	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	$\frac{4}{65}$	0	0	0
6	$\frac{2}{65}$	0	0	0	0	0	$\frac{3}{65}$
7	$\frac{3}{65}$	$\frac{2}{65}$	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0



$$\mathrm{ERL}(\mathcal{D},\mathrm{N}) = \sum_{(u,v)\in\mathcal{D}} p(u,v)\cdot d_{N}(u,v)$$

Case Study “Route Lengths”

Constant-Degree Demand-Aware Network

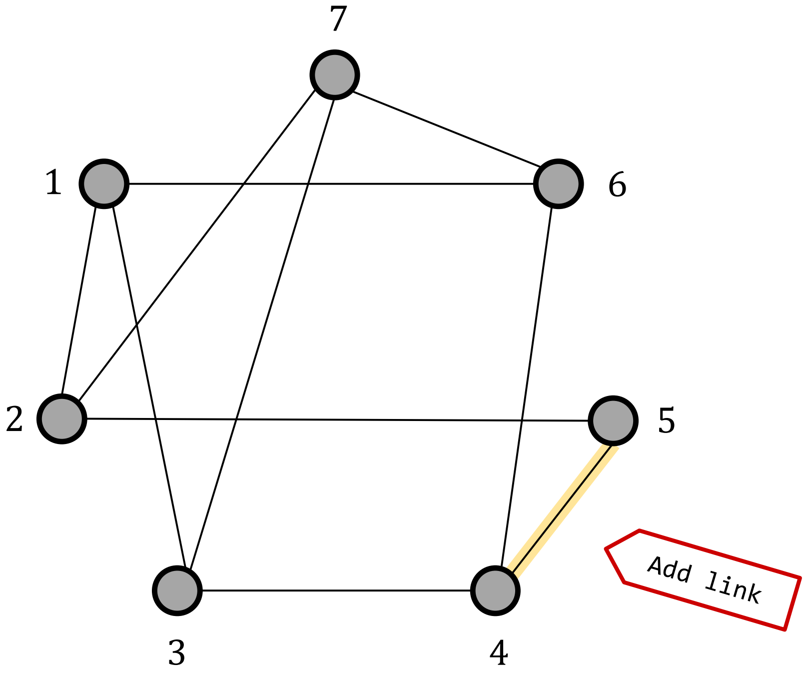


Destinations

Sources

	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$		0	0	0
6	$\frac{2}{65}$	0		0	0	0	$\frac{3}{65}$
7	$\frac{3}{65}$		$\frac{1}{13}$	0	0	$\frac{3}{65}$	0

Much from 4 to 5



Add link

$$\mathrm{ERL}(\mathcal{D},\mathrm{N}) = \sum_{(u,v)\in\mathcal{D}} p(u,v) \cdot d_N(u,v)$$

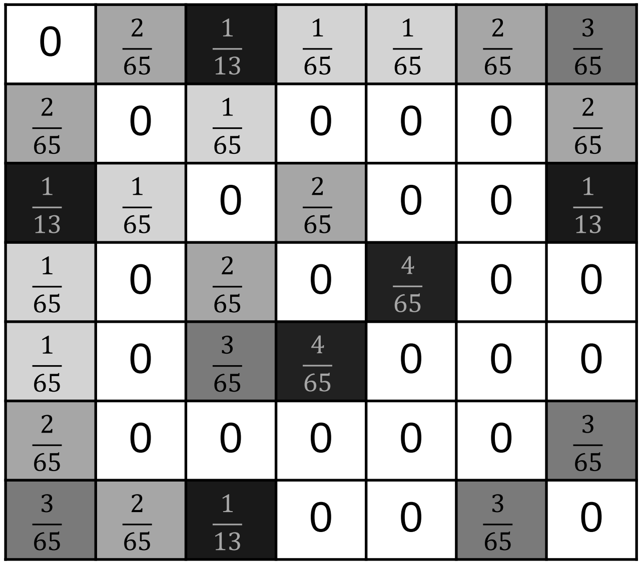
Case Study “Route Lengths”

Constant-Degree Demand-Aware Network

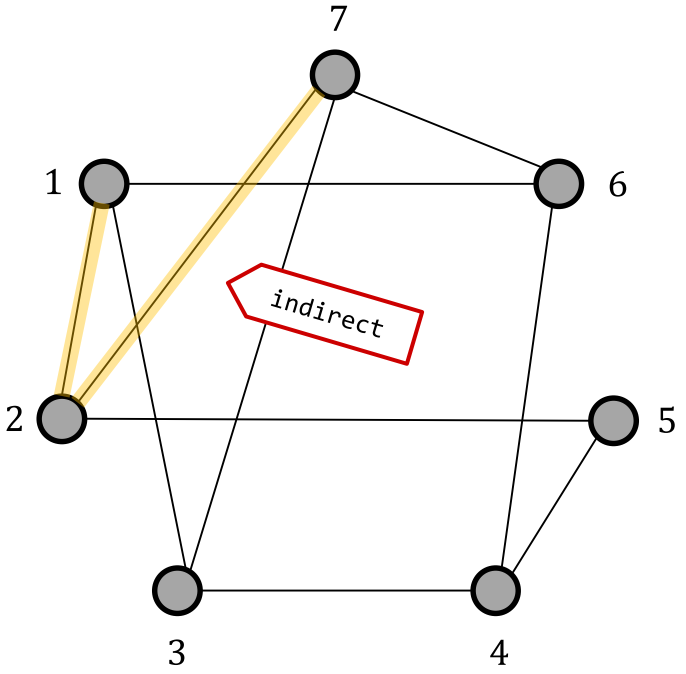
Communicated with many

Destinations

Sources



	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	$\frac{4}{65}$	0	0	0
6	$\frac{2}{65}$	0	0	0	0	0	$\frac{3}{65}$
7	$\frac{3}{65}$	$\frac{2}{65}$	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0

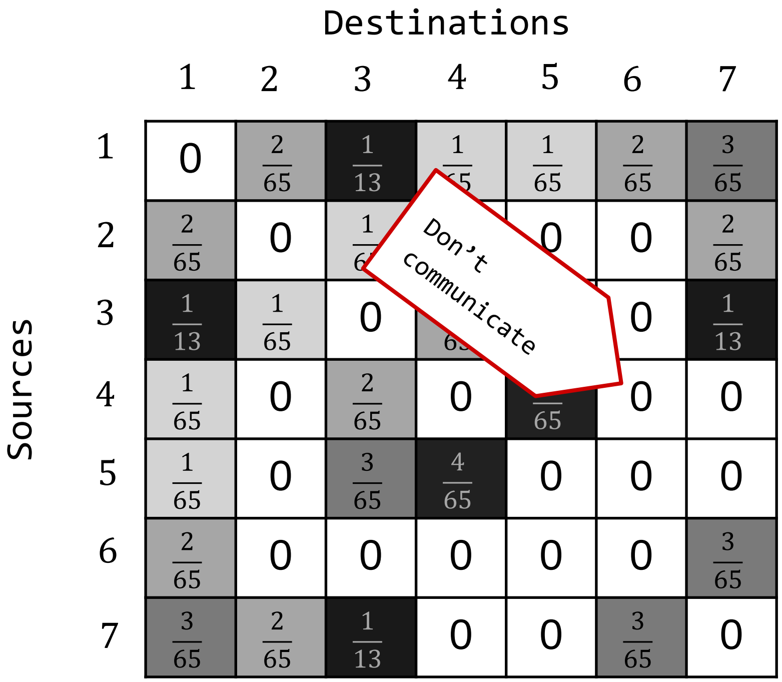


indirect

$$\mathrm{ERL}(\mathcal{D},N) = \sum_{(u,v)\in\mathcal{D}} p(u,v)\cdot d_N(u,v)$$

Case Study “Route Lengths”

Constant-Degree Demand-Aware Network

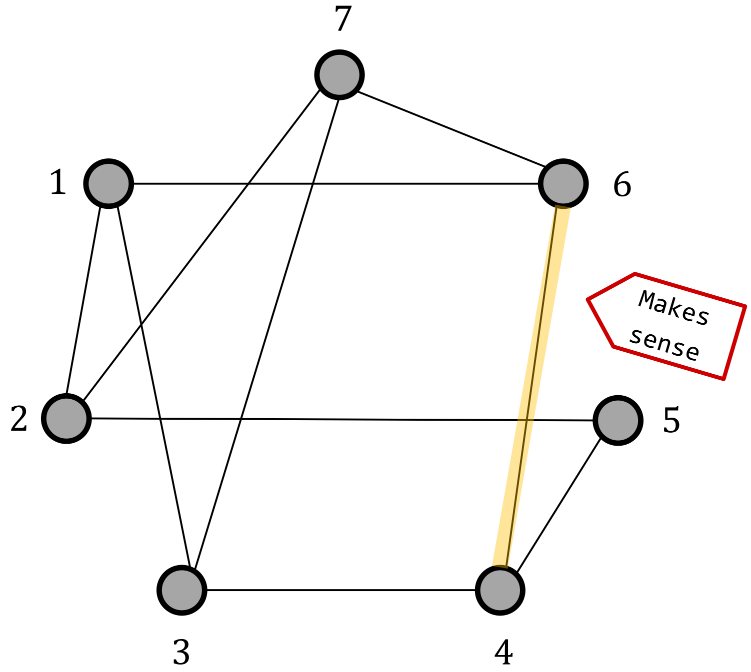


Destinations

Sources

	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{6}$		0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	6		0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	65	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	$\frac{4}{65}$	0	0	0
6	$\frac{2}{65}$	0	0	0	0	0	$\frac{3}{65}$
7	$\frac{3}{65}$	$\frac{2}{65}$	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0

Don’t communicate

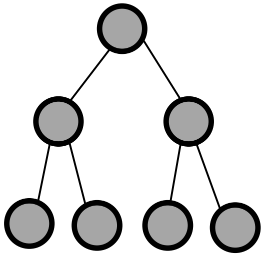


Makes sense

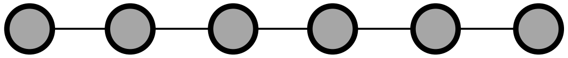
$$\mathrm{ERL}(\mathcal{D},\mathrm{N}) = \sum_{(\mathrm{u},\mathrm{v})\in\mathcal{D}} \mathrm{p}(\mathrm{u},\mathrm{v}) \cdot \mathrm{d}_{\mathrm{N}}(\mathrm{u},\mathrm{v})$$

Examples

- DAN for △=3
  - E.g., complete **binary tree** would be **log n**
  - Can we do better?

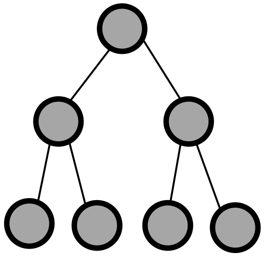


- DAN for △=2
  - Set of **lines** and **cycles**

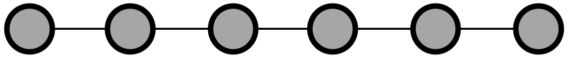


Examples

- DAN for △=3
  - E.g., complete **binary tree** would be **log n**
  - Can we do better?



- DAN for △=2
  - Set of **lines** and **cycles**



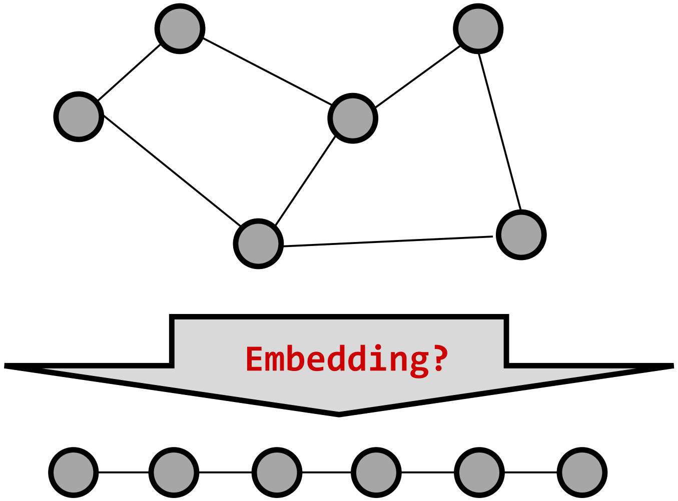
How hard?

Related Problem

Virtual Network Embedding Problem (VNEP)

Example △=2: A Minium Linear Arrangement (**MLA**) Problem

→ Minimizes sum of virtual edges

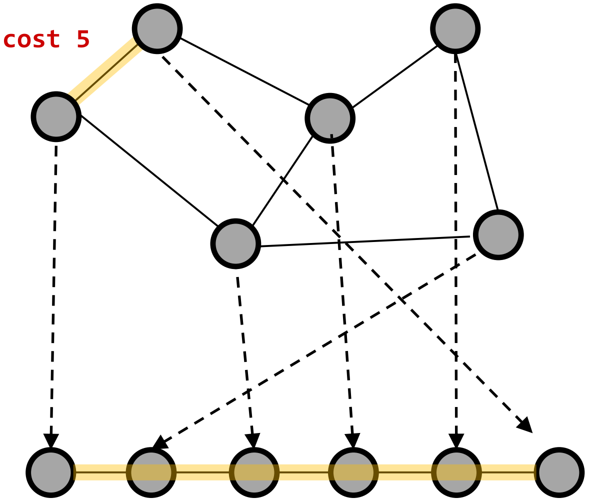


Related Problem

Virtual Network Embedding Problem (VNEP)

Example △=2: A Minium Linear Arrangement (**MLA**) Problem

→ Minimizes sum of virtual edges



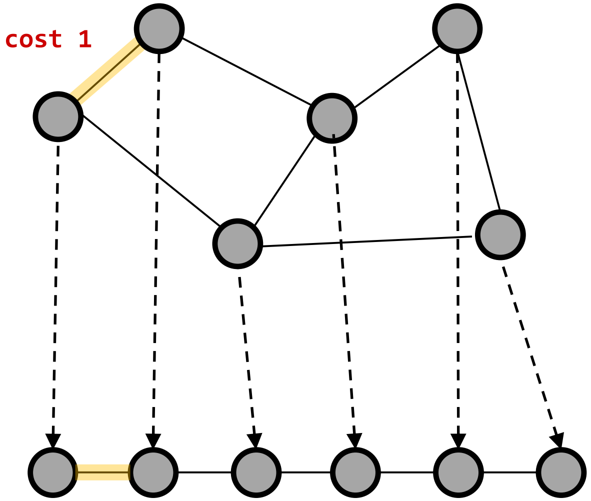
Bad!

Related Problem

Virtual Network Embedding Problem (VNEP)

Example △=2: A Minium Linear Arrangement (**MLA**) Problem

→ Minimizes sum of virtual edges



Good!

Related Problem

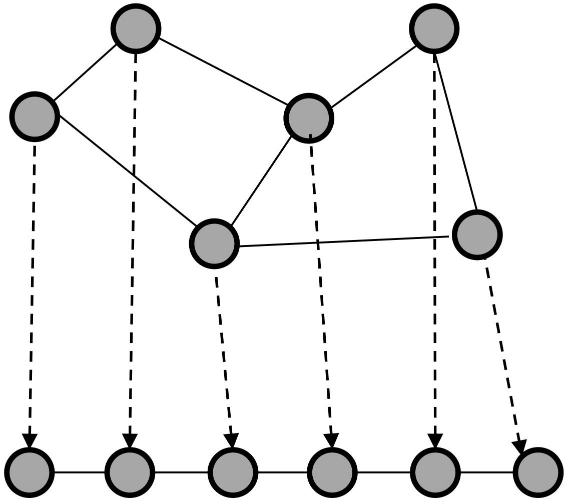
Virtual Network Embedding Problem (VNEP)

Example △=2: A Minium Linear Arrangement (**MLA**) Problem

⇀ Minimizes sum of virtual edges

MLA is **NP-hard**

⇀ … and so is our problem!



Related Problem

Virtual Network Embedding Problem (VNEP)

Example △=2: A Minium Linear Arrangement (**MLA**) Problem

⇀ Minimizes sum of virtual edges

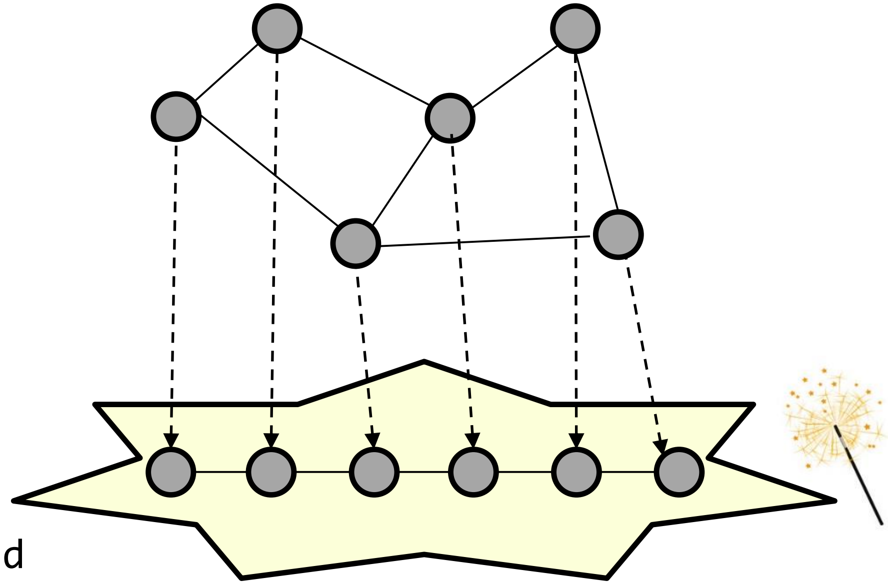
MLA is **NP-hard**

⇀ … and so is our problem!

But what about **△>2**?

⇀ Embedding problem still hard

⇀ But we have a new **degree of freedom**!



Related Problem

Virtual Network Embedding Problem (VNEP)

Example △=2: A Minium Linear Arrangement (**MLA**) Problem

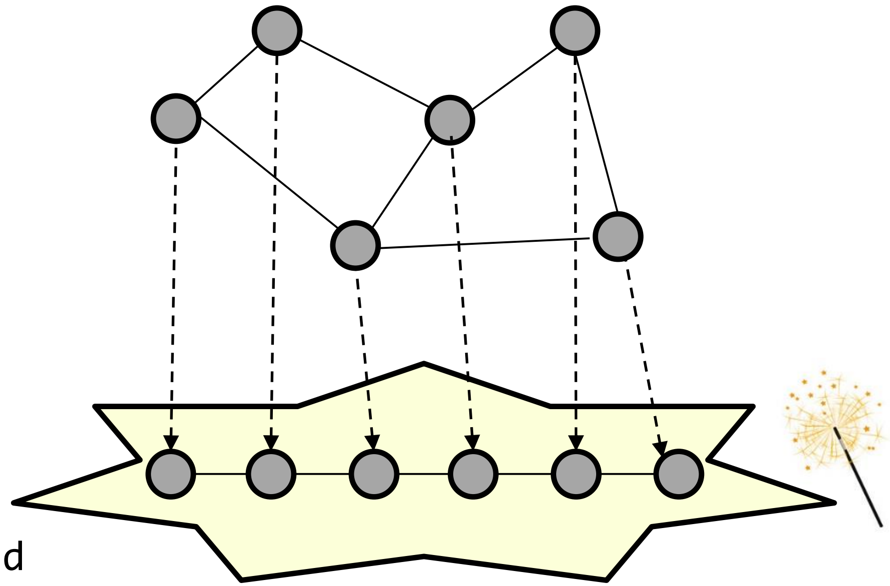
- ⇀ Minimizes sum of virtual edges

MLA is **NP-hard**

- ⇀ … and so is our problem!

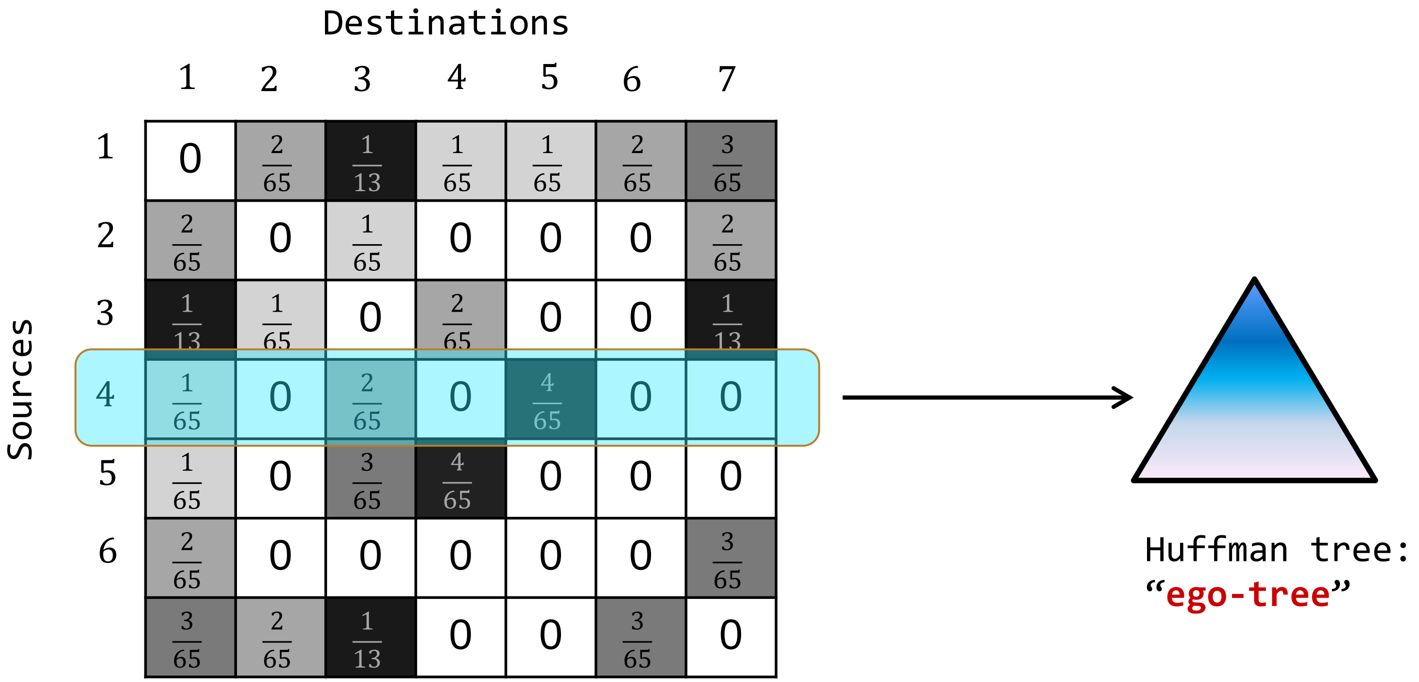
But what about **△>2**?

- ⇀ Embedding problem still hard
- ⇀ But we have a new **degree of freedom**!



Simplifies problem?!

Entropy Lower Bound



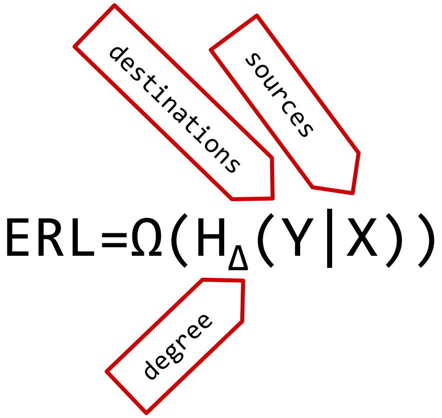
Destinations

Sources

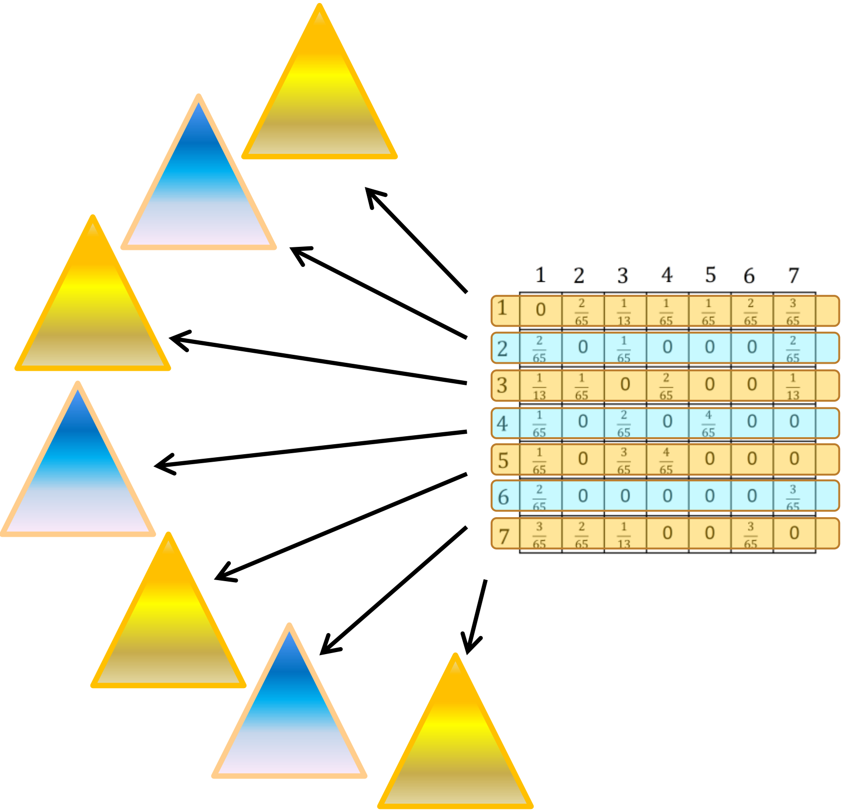
	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	$\frac{4}{65}$	0	0	0
6	$\frac{2}{65}$	0	0	0	0	0	$\frac{3}{65}$
	$\frac{3}{65}$	$\frac{2}{65}$	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0

Huffman tree:
“**ego-tree**”

Entropy Lower Bound



$$\mathrm{ERL}=\Omega(H_{\Delta}(Y|X))$$

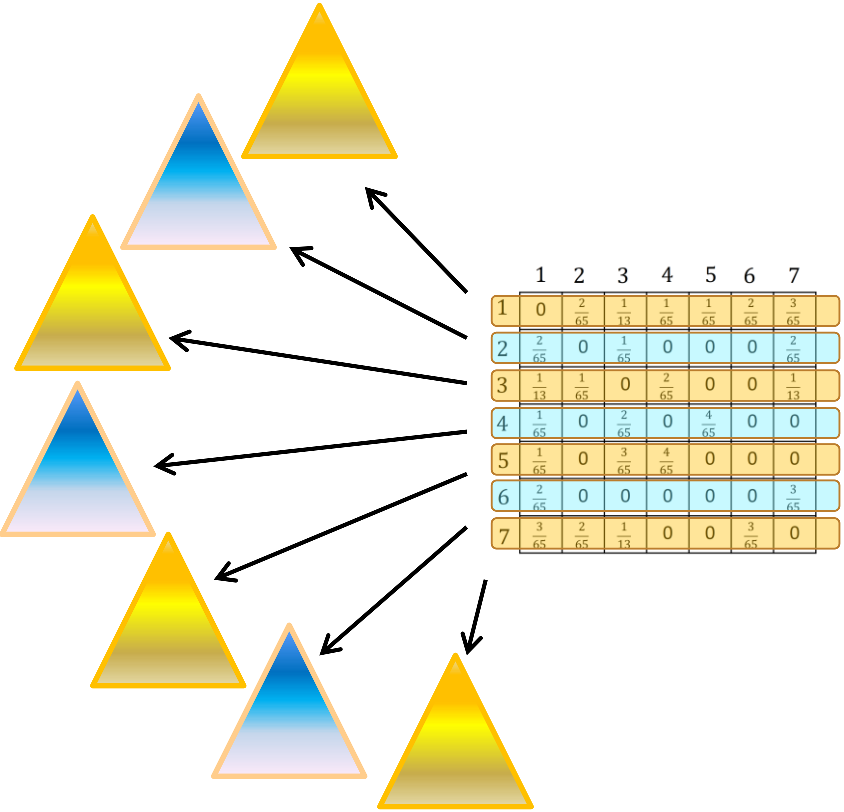


	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	$\frac{4}{65}$	0	0	0
6	$\frac{2}{65}$	0	0	0	0	0	$\frac{3}{65}$
7	$\frac{3}{65}$	$\frac{2}{65}$	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0

Entropy Upper Bound

- Idea for algorithm:
  - union of trees
  - reduce degree
  - but keep distances

Static

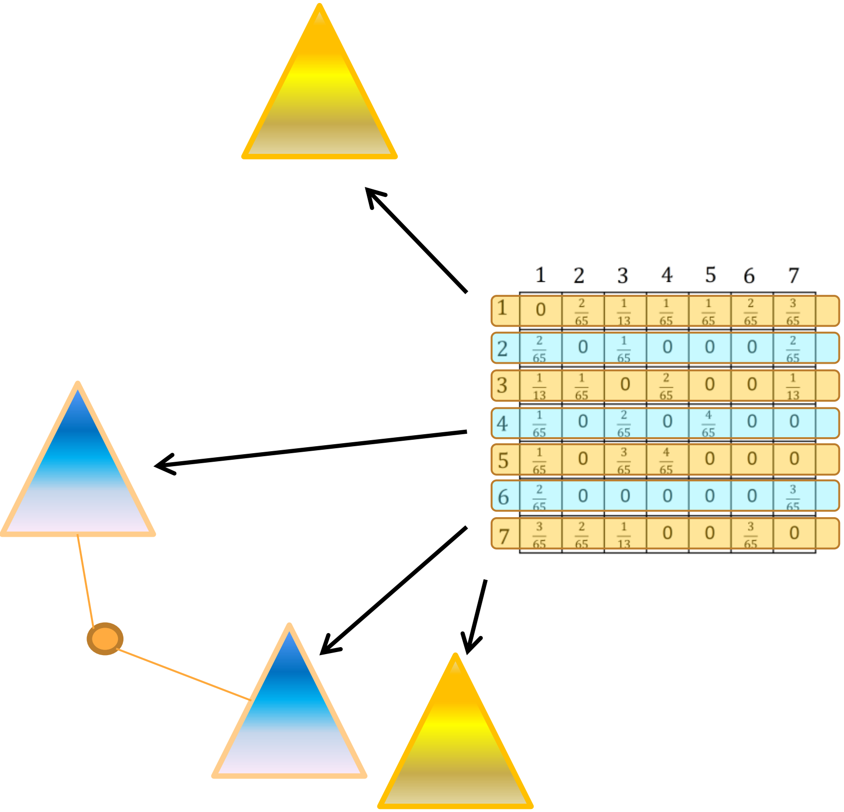


	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	$\frac{4}{65}$	0	0	0
6	$\frac{2}{65}$	0	0	0	0	0	$\frac{3}{65}$
7	$\frac{3}{65}$	$\frac{2}{65}$	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0

Entropy Upper Bound

- Idea for algorithm:
    - union of trees
    - reduce degree
    - but keep distances
- Ok for sparse demands
    - not everyone gets tree
    - helper nodes

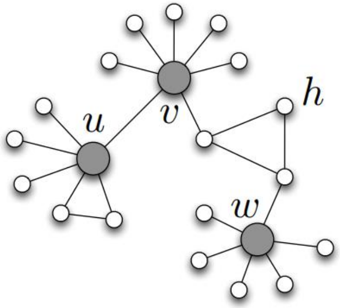
Static



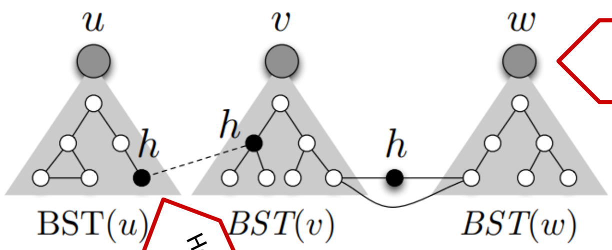
	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	$\frac{4}{65}$	0	0	0
6	$\frac{2}{65}$	0	0	0	0	0	$\frac{3}{65}$
7	$\frac{3}{65}$	$\frac{2}{65}$	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0

Intuition of Algorithm

Demand graph:



Demand-aware network:

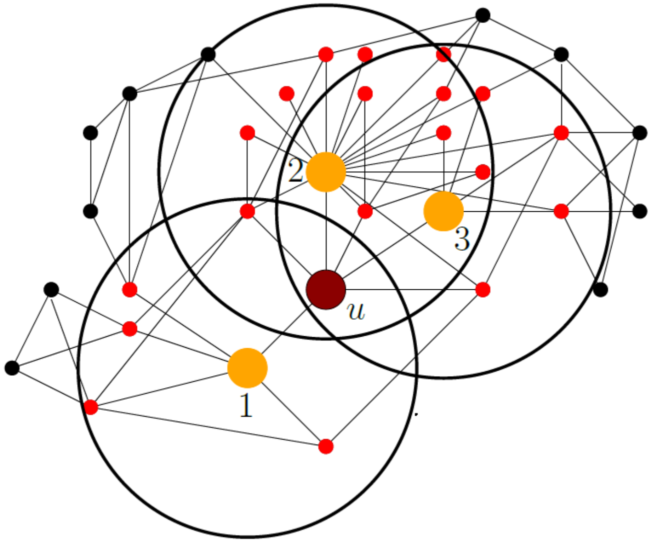


Ego-trees for large nodes

Helper node

More Optimal Graphs

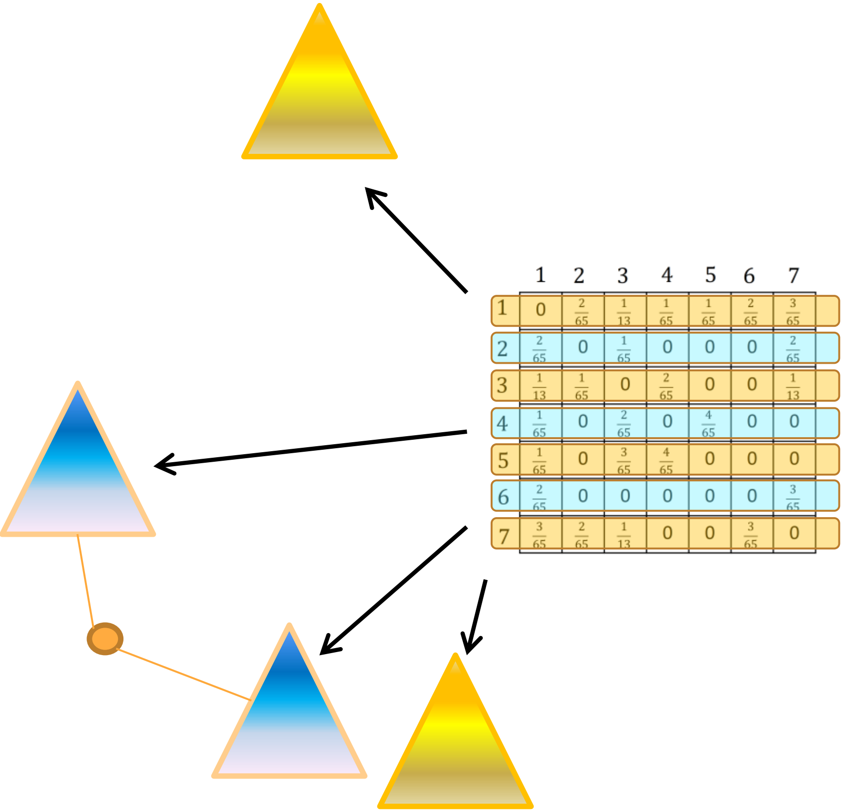
- For regular and uniform demands which admit constant distortion linear spanner
- Graphs of bounded doubling dimension



Accounting for Load

- Still use ego-trees
- But balance for load

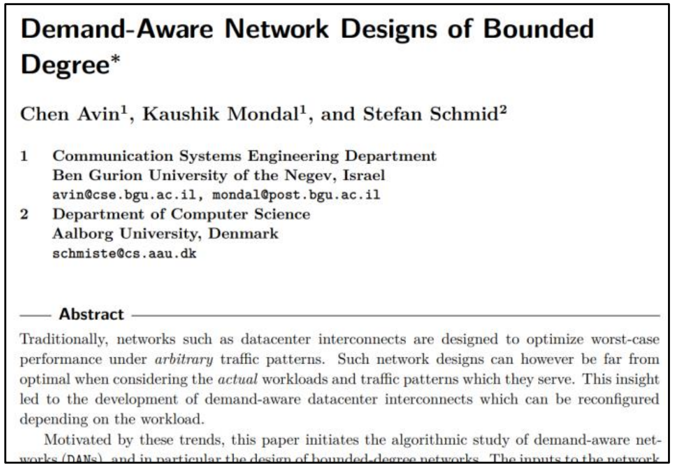
Load



	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	$\frac{4}{65}$	0	0	0
6	$\frac{2}{65}$	0	0	0	0	0	$\frac{3}{65}$
7	$\frac{3}{65}$	$\frac{2}{65}$	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0

Further Reading

TON 2016, DISC 2017, CCR 2019, INFOCOM 2019



Demand-Aware Network Designs of Bounded Degree*

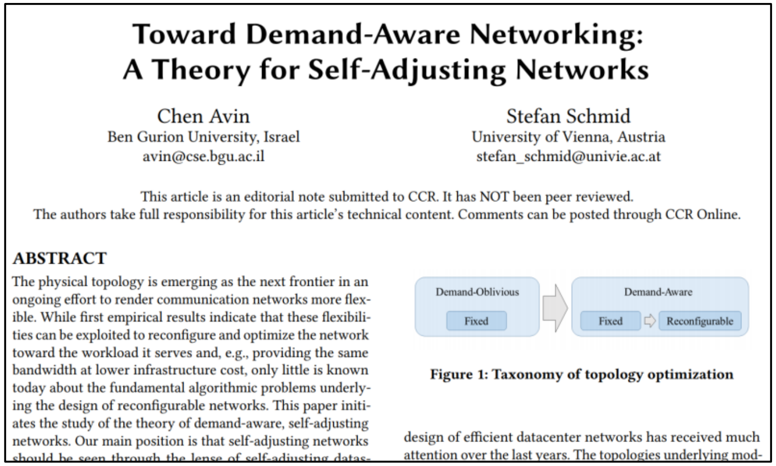
Chen Avin[1], Kaushik Mondal[1], and Stefan Schmid[2]

1 Communication Systems Engineering Department
Ben Gurion University of the Negev, Israel
avin@cse.bgu.ac.il, mondal@post.bgu.ac.il
2 Department of Computer Science
Aalborg University, Denmark
schmiste@cs.aau.dk

Abstract

Traditionally, networks such as datacenter interconnects are designed to optimize worst-case performance under *arbitrary* traffic patterns. Such network designs can however be far from optimal when considering the *actual* workloads and traffic patterns which they serve. This insight led to the development of demand-aware datacenter interconnects which can be reconfigured depending on the workload.

Motivated by these trends, this paper initiates the algorithmic study of demand-aware networks (DANs), and in particular the design of bounded-degree networks. The inputs to the network



Toward Demand-Aware Networking: A Theory for Self-Adjusting Networks

Chen Avin
Ben Gurion University, Israel
avin@cse.bgu.ac.il

Stefan Schmid
University of Vienna, Austria
stefan_schmid@univie.ac.at

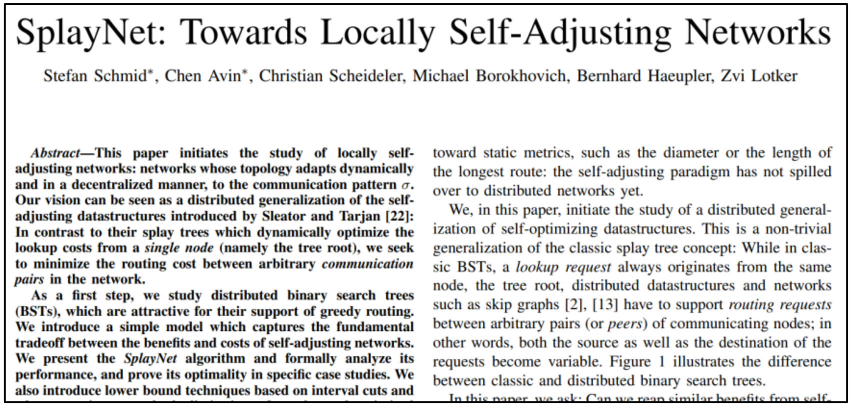
This article is an editorial note submitted to CCR. It has NOT been peer reviewed.
The authors take full responsibility for this article's technical content. Comments can be posted through CCR Online.

ABSTRACT

The physical topology is emerging as the next frontier in an ongoing effort to render communication networks more flexible. While first empirical results indicate that these flexibilities can be exploited to reconfigure and optimize the network toward the workload it serves and, e.g., providing the same bandwidth at lower infrastructure cost, only little is known today about the fundamental algorithmic problems underlying the design of reconfigurable networks. This paper initiates the study of the theory of demand-aware, self-adjusting networks. Our main position is that self-adjusting networks should be seen through the lense of self-adjusting datas-

Figure 1: Taxonomy of topology optimization

design of efficient datacenter networks has received much attention over the last years. The topologies underlying mod-



SplayNet: Towards Locally Self-Adjusting Networks

Stefan Schmid*, Chen Avin*, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler, Zvi Lotker

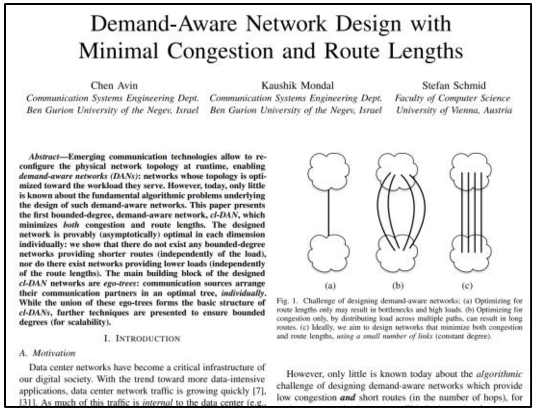
***Abstract*—This paper initiates the study of locally self-adjusting networks: networks whose topology adapts dynamically and in a decentralized manner, to the communication pattern σ. Our vision can be seen as a distributed generalization of the self-adjusting datastructures introduced by Sleator and Tarjan [22]: In contrast to their splay trees which dynamically optimize the lookup costs from a *single node* (namely the tree root), we seek to minimize the routing cost between arbitrary *communication pairs* in the network.**

As a first step, we study distributed binary search trees (BSTs), which are attractive for their support of greedy routing. We introduce a simple model which captures the fundamental tradeoff between the benefits and costs of self-adjusting networks. We present the *SplayNet* algorithm and formally analyze its performance, and prove its optimality in specific case studies. We also introduce lower bound techniques based on interval cuts and

toward static metrics, such as the diameter or the length of the longest route: the self-adjusting paradigm has not spilled over to distributed networks yet.

We, in this paper, initiate the study of a distributed generalization of self-optimizing datastructures. This is a non-trivial generalization of the classic splay tree concept: While in classic BSTs, a *lookup request* always originates from the same node, the tree root, distributed datastructures and networks such as skip graphs [2], [13] have to support *routing requests* between arbitrary pairs (or *peers*) of communicating nodes; in other words, both the source as well as the destination of the requests become variable. Figure 1 illustrates the difference between classic and distributed binary search trees.

In this paper, we ask: Can we reap similar benefits from self-



Demand-Aware Network Design with Minimal Congestion and Route Lengths

Chen Avin
Communication Systems Engineering Dept.
Ben Gurion University of the Negev, Israel

Kaushik Mondal
Communication Systems Engineering Dept.
Ben Gurion University of the Negev, Israel

Stefan Schmid
Faculty of Computer Science
University of Vienna, Austria

***Abstract*—Emerging communication technologies allow to reconfigure the physical network topology at runtime, enabling *demand-aware networks (DANs)*: networks whose topology is optimized toward the workload they serve. However, today, only little is known about the fundamental algorithmic problems underlying the design of such demand-aware networks. This paper presents the first bounded-degree, demand-aware network, *cl-DAN*, which minimizes *both* congestion and route lengths. The designed network is provably (asymptotically) optimal in each dimension individually: we show that there do not exist any bounded-degree networks providing shorter routes (independently of the load), nor do there exist networks providing lower loads (independently of the route lengths). The main building block of the designed *cl-DAN* networks are *ego-trees*: communication sources arrange their communication partners in an optimal tree, *individually*. While the union of these ego-trees forms the basic structure of *cl-DANs*, further techniques are presented to ensure bounded degrees (for scalability).**

I. INTRODUCTION

A. Motivation

Data center networks have become a critical infrastructure of our digital society. With the trend toward more data-intensive applications, data center network traffic is growing quickly [7], [31]. As much of this traffic is *internal* to the data center (e.g.,

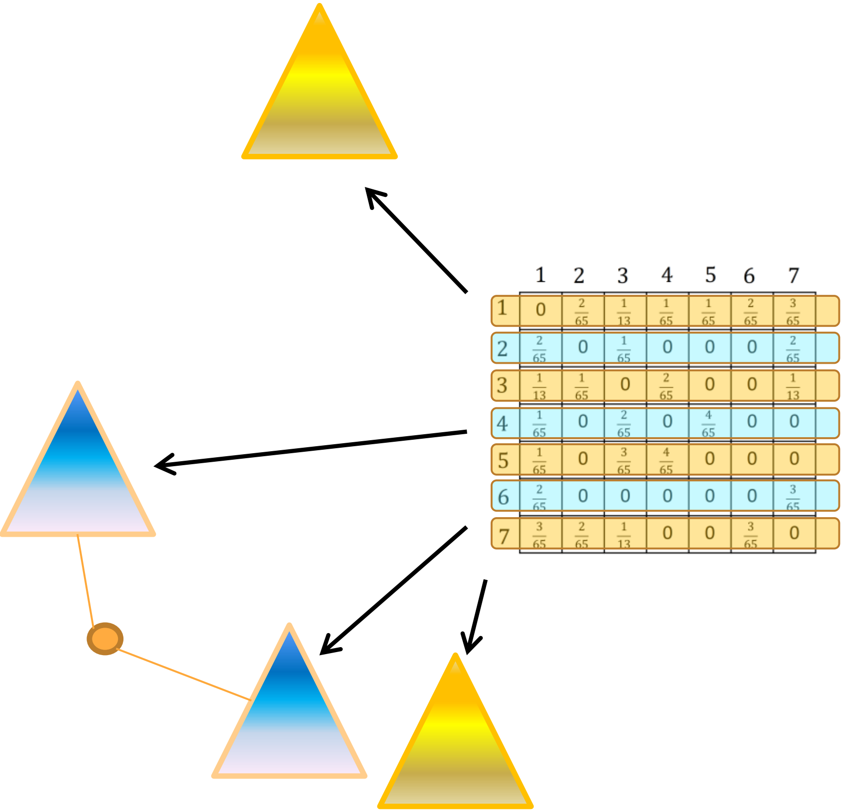
Fig. 1. Challenge of designing demand-aware networks: (a) Optimizing for route lengths only may result in bottlenecks and high loads. (b) Optimizing for congestion only, by distributing load across multiple paths, can result in long routes. (c) Ideally, we aim to design networks that minimize both congestion and route lengths, *using a small number of links* (constant degree).

However, only little is known today about the *algorithmic* challenge of designing demand-aware networks which provide low congestion ***and*** short routes (in the number of hops), for

Dynamic Setting

- ⇢ Dynamic the same:
  - → union of **dynamic ego-trees**
- ⇢ E.g., SplayNets
- ⇢ Online algorithms

Dynamic



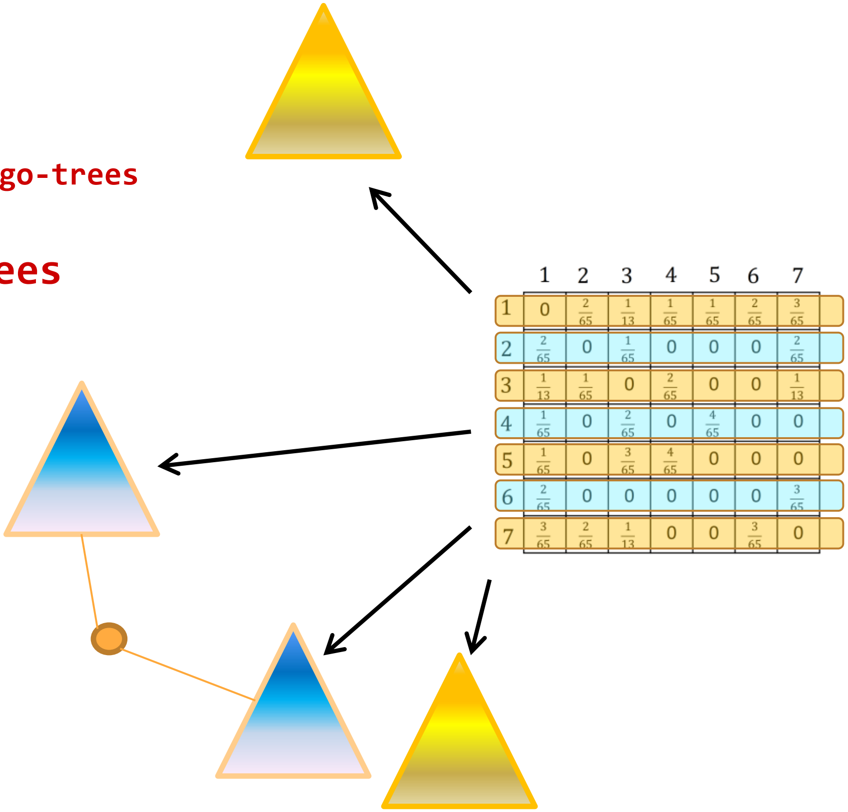
$$\begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 1 & 0 & \frac{2}{65} & \frac{1}{13} & \frac{1}{65} & \frac{1}{65} & \frac{2}{65} & \frac{3}{65} \\ 2 & \frac{2}{65} & 0 & \frac{1}{65} & 0 & 0 & 0 & \frac{2}{65} \\ 3 & \frac{1}{13} & \frac{1}{65} & 0 & \frac{2}{65} & 0 & 0 & \frac{1}{13} \\ 4 & \frac{1}{65} & 0 & \frac{2}{65} & 0 & \frac{4}{65} & 0 & 0 \\ 5 & \frac{1}{65} & 0 & \frac{3}{65} & \frac{4}{65} & 0 & 0 & 0 \\ 6 & \frac{2}{65} & 0 & 0 & 0 & 0 & 0 & \frac{3}{65} \\ 7 & \frac{3}{65} & \frac{2}{65} & \frac{1}{13} & 0 & 0 & \frac{3}{65} & 0 \end{array}$$

Dynamic Setting

& distributed

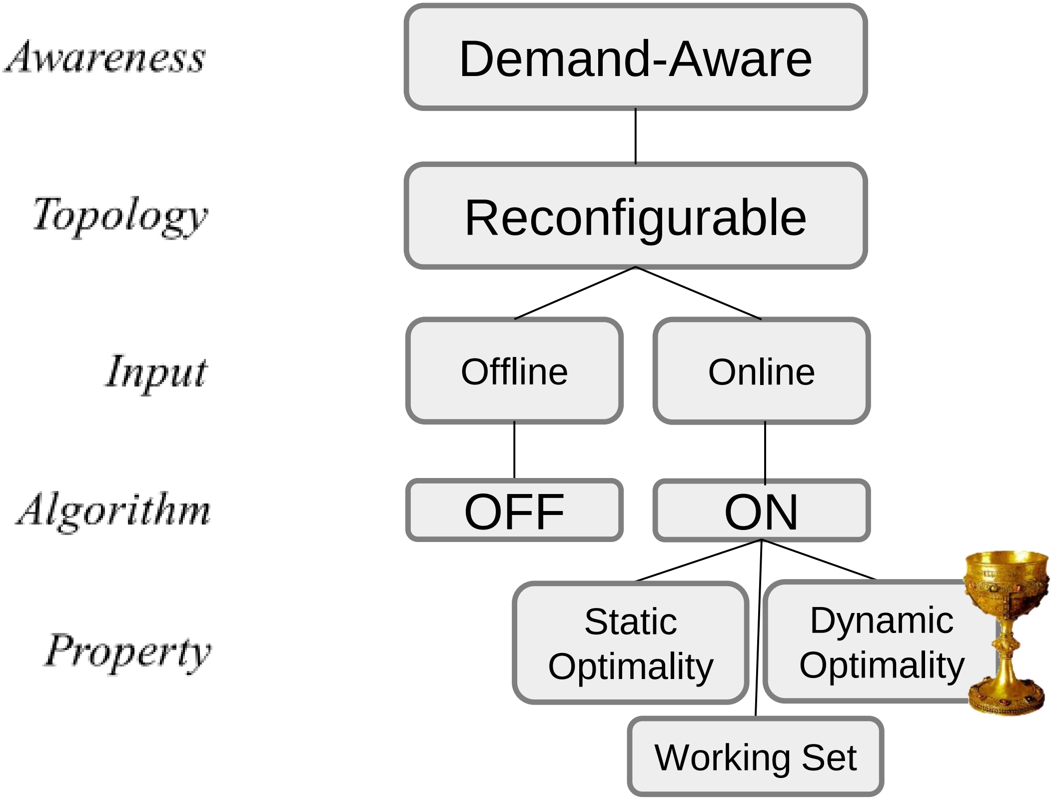
- Dynamic the same:
  - union of **dynamic&distributed ego-trees**
- E.g., SplayNets or **CB trees**
- Online algorithms

Dynamic



$$\begin{array}{c|ccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
1 & 0 & \frac{2}{65} & \frac{1}{13} & \frac{1}{65} & \frac{1}{65} & \frac{2}{65} & \frac{3}{65} \\
2 & \frac{2}{65} & 0 & \frac{1}{65} & 0 & 0 & 0 & \frac{2}{65} \\
3 & \frac{1}{13} & \frac{1}{65} & 0 & \frac{2}{65} & 0 & 0 & \frac{1}{13} \\
4 & \frac{1}{65} & 0 & \frac{2}{65} & 0 & \frac{4}{65} & 0 & 0 \\
5 & \frac{1}{65} & 0 & \frac{3}{65} & \frac{4}{65} & 0 & 0 & 0 \\
6 & \frac{2}{65} & 0 & 0 & 0 & 0 & 0 & \frac{3}{65} \\
7 & \frac{3}{65} & \frac{2}{65} & \frac{1}{13} & 0 & 0 & \frac{3}{65} & 0
\end{array}$$

Dynamic Objectives



Awareness Demand-Aware

Topology Reconfigurable

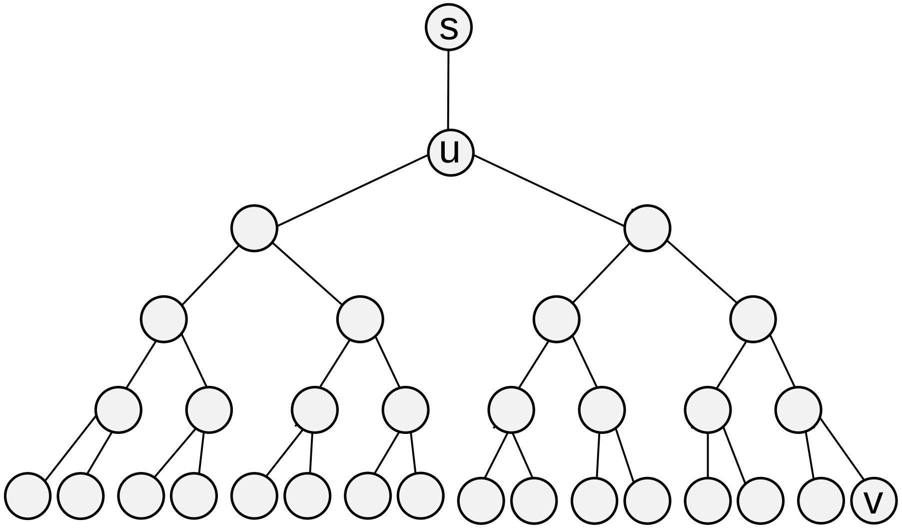
Input Offline Online

Algorithm OFF ON

Property Static Optimality Dynamic Optimality Working Set

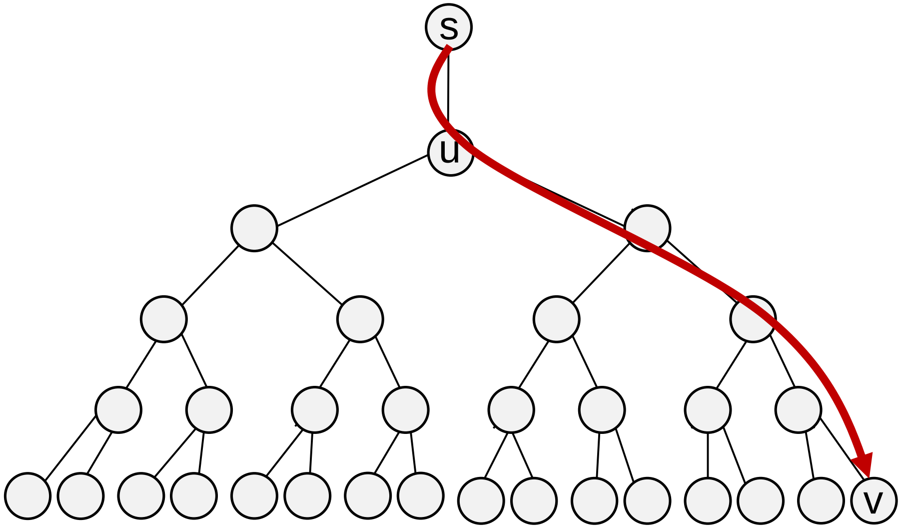
Dynamic Optimality: Push-Down Trees

- For unordered search trees, dynamic optimality is possible: **Push-Down Trees**
- Useful property: **most recently used (MRU)**



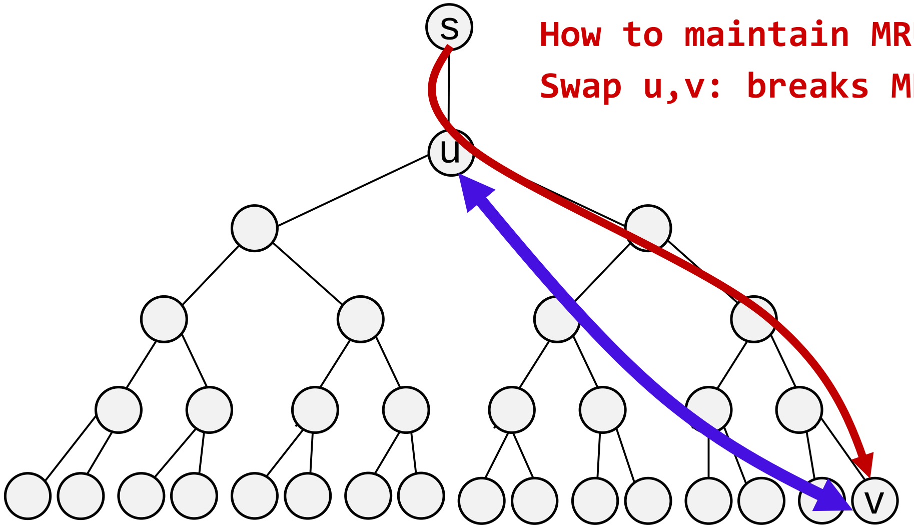
Dynamic Optimality: Push-Down Trees

- For unordered search trees, dynamic optimality is possible: **Push-Down Trees**
- Useful property: **most recently used (MRU)**



Dynamic Optimality: Push-Down Trees

- For unordered search trees, dynamic optimality is possible: **Push-Down Trees**
- Useful property: **most recently used (MRU)**

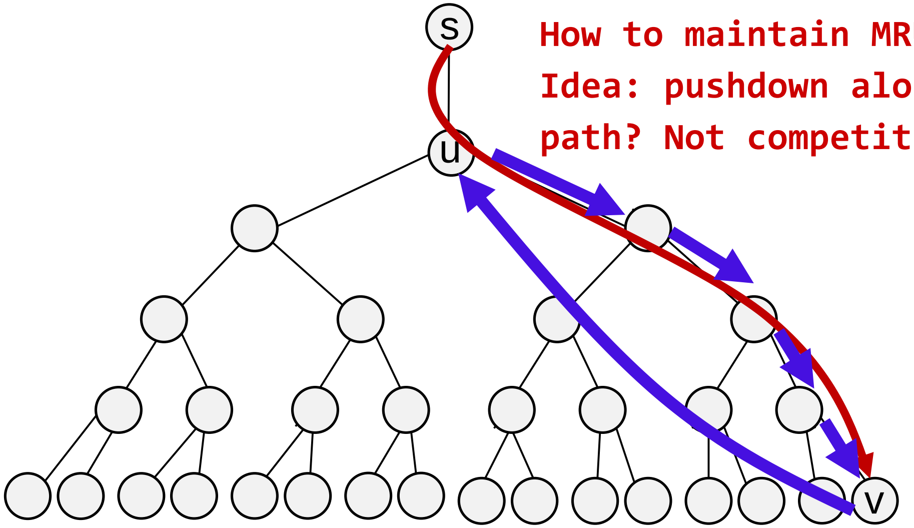


How to maintain MRU?

Swap u,v: breaks MRU!

Dynamic Optimality: Push-Down Trees

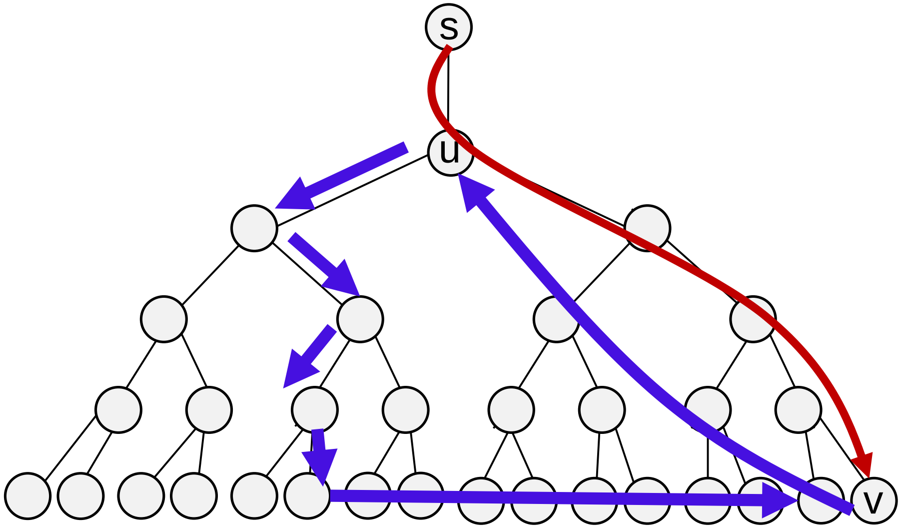
- For unordered search trees, dynamic optimality is possible: **Push-Down Trees**
- Useful property: **most recently used (MRU)**



How to maintain MRU? Idea: pushdown along path? Not competitive!

Dynamic Optimality: Push-Down Trees

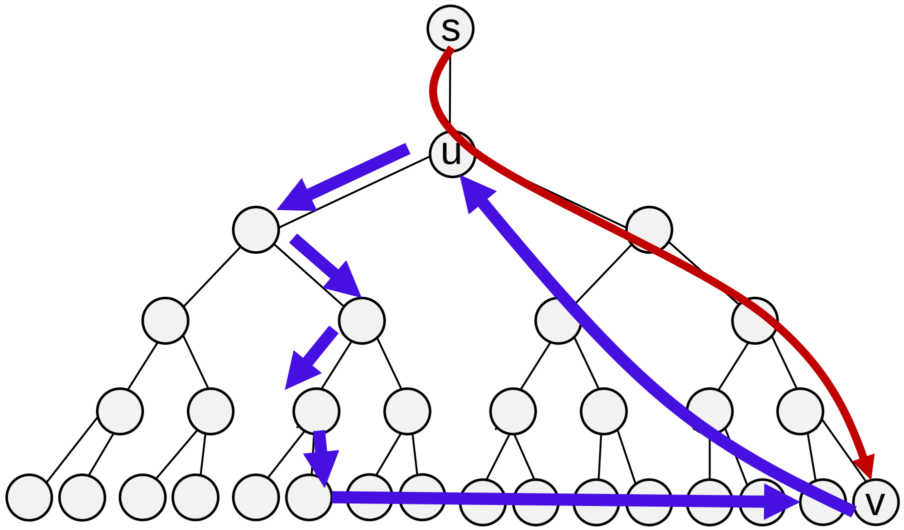
- For unordered search trees, dynamic optimality is possible: **Push-Down Trees**
- Useful property: **most recently used (MRU)**
- Idea: **balanced** pushdown (random vs deterministic?)



Dynamic Optimality: Push-Down Trees

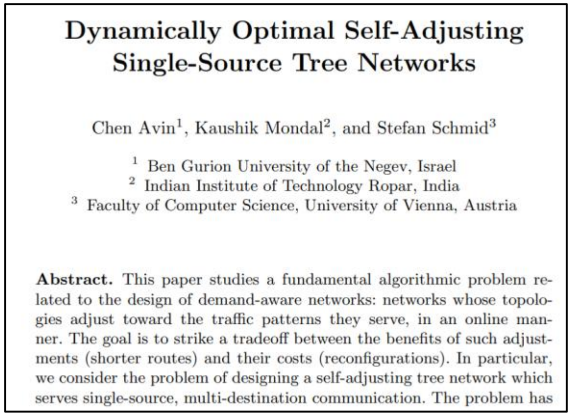
- For unordered search trees, dynamic optimality is possible: **Push-Down Trees**
- Useful property: **most recently used (MRU)**
- Idea: **balanced** pushdown (random vs deterministic?)

Random walk preservers MRU!
Constant competitive.
Deterministic does not.
Still constant competitive?



Further Reading

LATIN 2020, IPDPS 2021

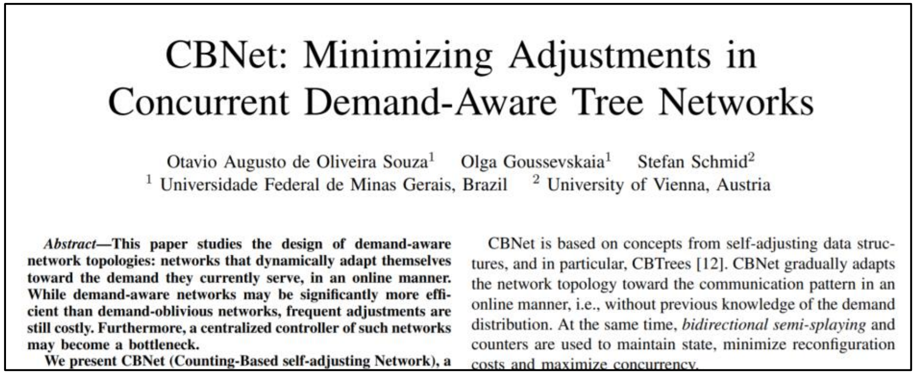


Dynamically Optimal Self-Adjusting Single-Source Tree Networks

Chen Avin[1], Kaushik Mondal[2], and Stefan Schmid[3]

[1] Ben Gurion University of the Negev, Israel
[2] Indian Institute of Technology Ropar, India
[3] Faculty of Computer Science, University of Vienna, Austria

Abstract. This paper studies a fundamental algorithmic problem related to the design of demand-aware networks: networks whose topologies adjust toward the traffic patterns they serve, in an online manner. The goal is to strike a tradeoff between the benefits of such adjustments (shorter routes) and their costs (reconfigurations). In particular, we consider the problem of designing a self-adjusting tree network which serves single-source, multi-destination communication. The problem has



CBNet: Minimizing Adjustments in Concurrent Demand-Aware Tree Networks

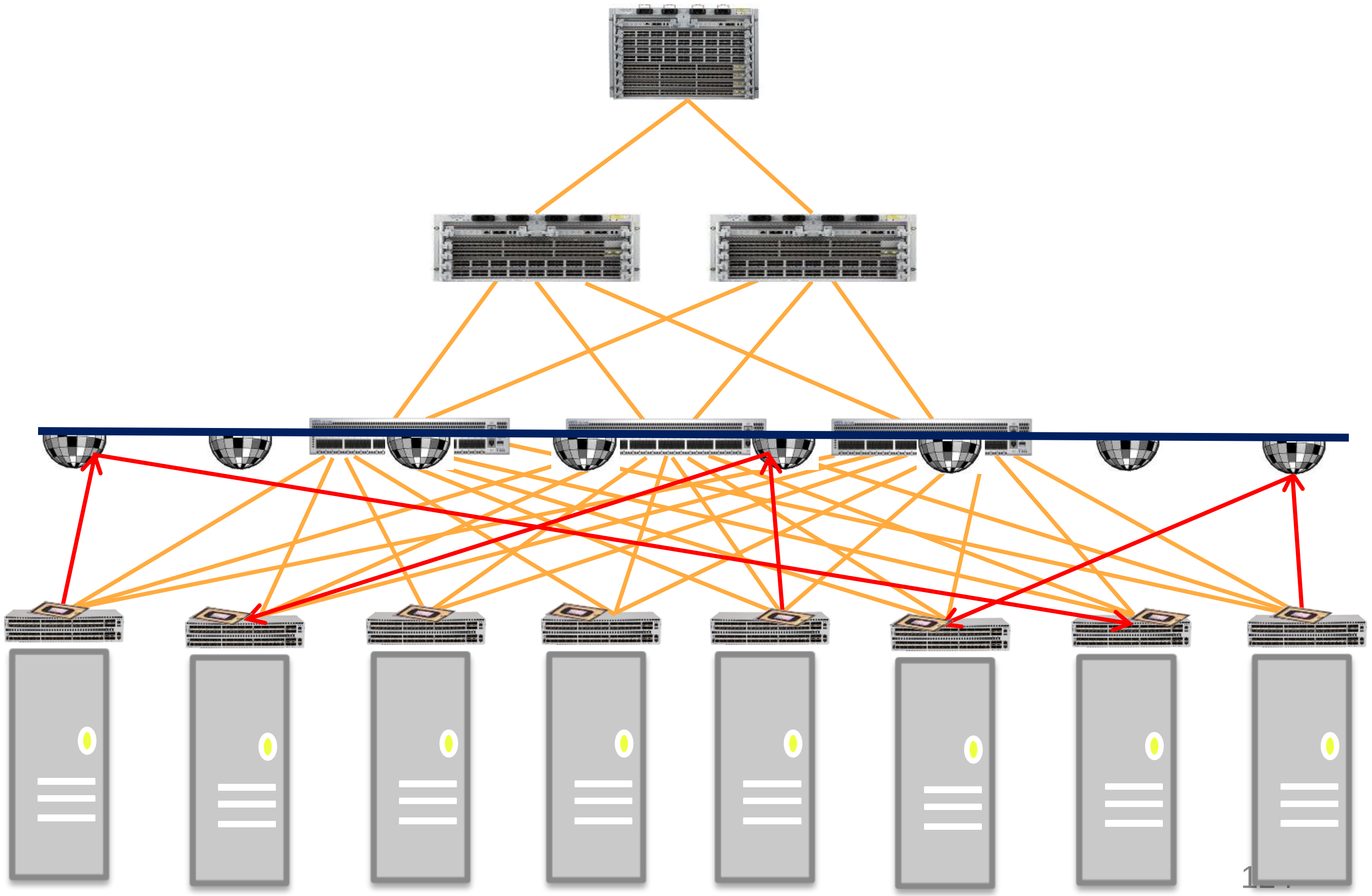
Otavio Augusto de Oliveira Souza[1] Olga Goussevskaia[1] Stefan Schmid[2]
[1] Universidade Federal de Minas Gerais, Brazil [2] University of Vienna, Austria

***Abstract*—This paper studies the design of demand-aware network topologies: networks that dynamically adapt themselves toward the demand they currently serve, in an online manner. While demand-aware networks may be significantly more efficient than demand-oblivious networks, frequent adjustments are still costly. Furthermore, a centralized controller of such networks may become a bottleneck.**

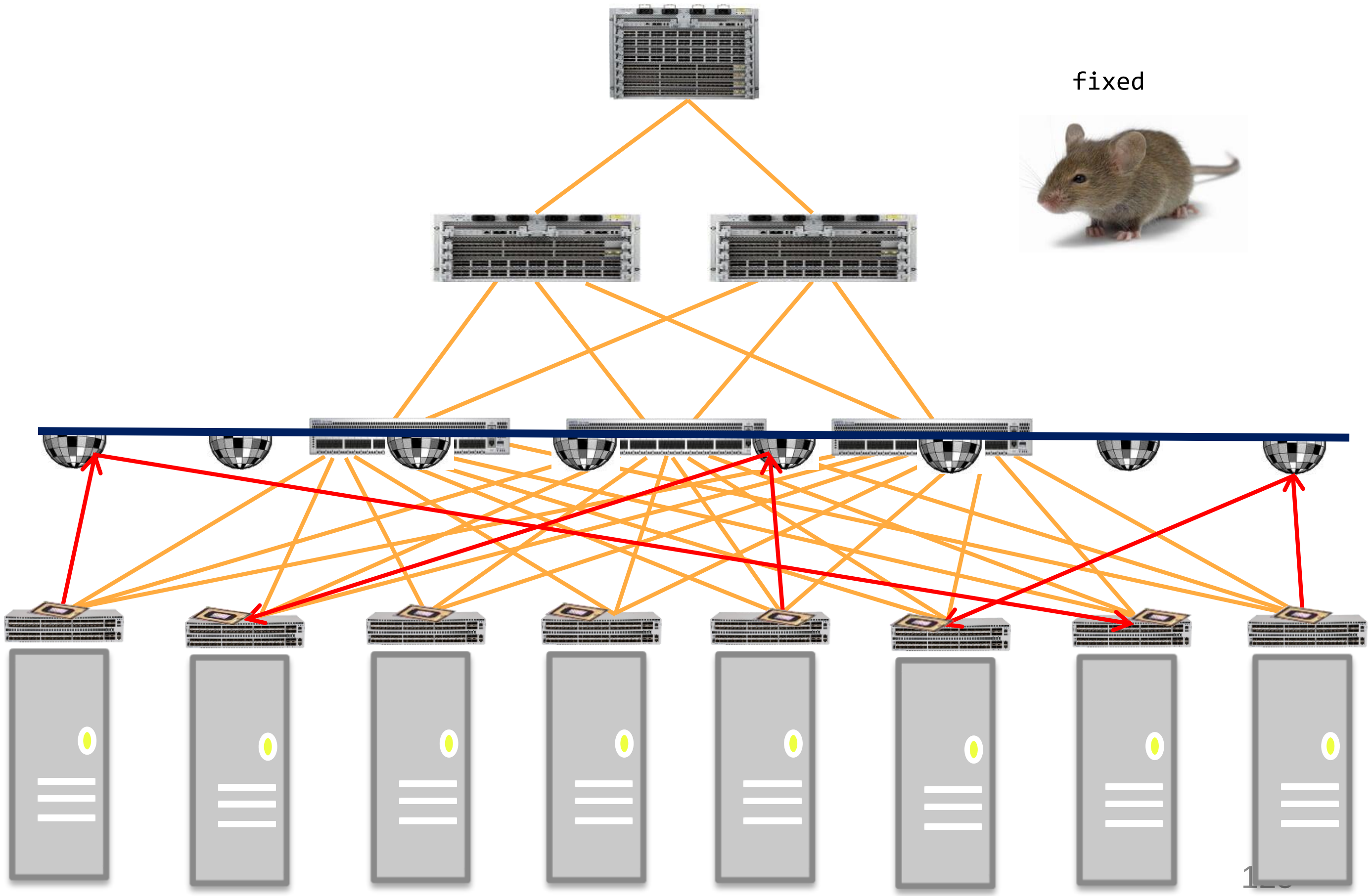
We present CBNet (Counting-Based self-adjusting Network), a

CBNet is based on concepts from self-adjusting data structures, and in particular, CBTrees [12]. CBNet gradually adapts the network topology toward the communication pattern in an online manner, i.e., without previous knowledge of the demand distribution. At the same time, *bidirectional semi-splaying* and counters are used to maintain state, minimize reconfiguration costs and maximize concurrency.

Hybrid Networks



Hybrid Networks



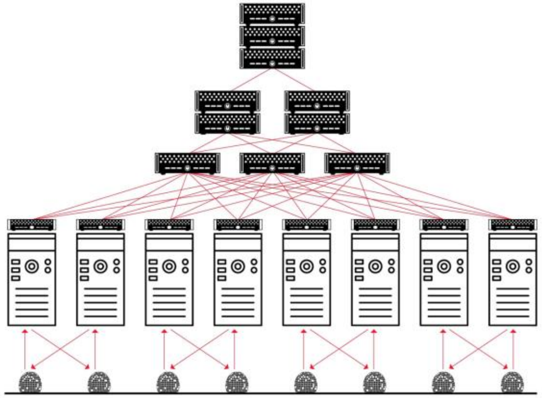
fixed

reconfigurable

ReNet

A Statically Optimal Demand-Aware Network

- Model: **hybrid architecture**
  - Fixed network of diameter log n plus reconfigurable network (**constant** number of direct links)
  - **Segregated** routing
  - **Online** sequence of requests: σ = (σ1, σ2, σ3, ...)
  - Global controller



fixed

reconfigurable

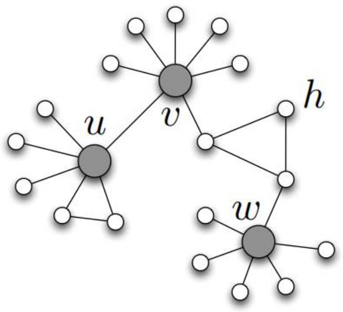
- **Objective:** Minimize route length plus reconfigurations
  - More specifically: be **statically optimal**
  - Compared to a fixed algorithm which knows σ ahead of time

- Compact routing (constant tables)
- Local routing (greedy)
- Arbitrary addressing

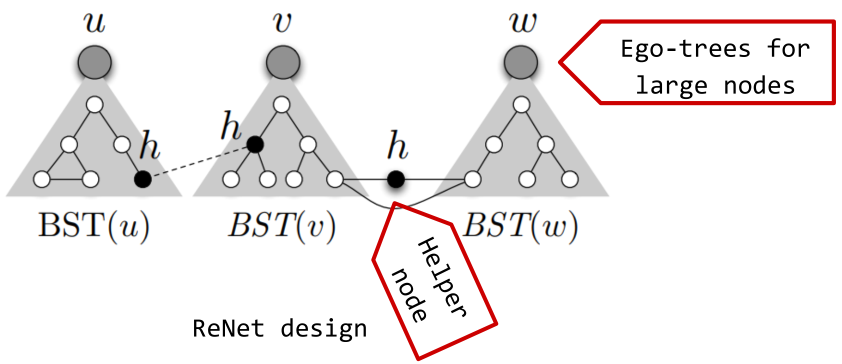
The ReNet Algorithm (1)

Algorithmic building blocks:

1. **Working Set** (WS)
   - ⇀ Nodes keep track of recent communication partners in σ.
2. Small/large nodes and **Ego-Tree**
   - ⇀ Nodes with small WS connect to WS directly, nodes with large WS via a self-adjusting binary search tree (e.g., a **splay tree**)
3. **Helper nodes** to reduce the degree
   - ⇀ Large nodes may appear in many ego-trees, so get help of small nodes



Demand graph

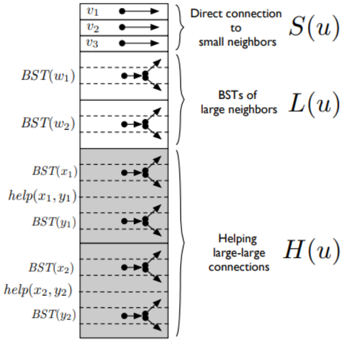


ReNet design

The ReNet Algorithm (2)

Continued:

4. **Self adjustments**
   - Keep track of WS; when too large: **flush-when-full**
5. Centralized coordination
   - Fairly **decentralized**: coordinator only needs to keep track of which nodes are large and which small
   - Nodes inform coordinator when adding node to working set
   - Coordinator then assigns helper node on demand



Analytical Results (1)

Theorem 1:

For any sparse communication sequence of a certain length, ReNets are statically optimal while ensuring a bounded degree.

⇢ Sparse: subsequences of only involve a linear number of nodes

⇢ Required to ensure availability of helper nodes (DISC 2017)

Analytical Results (2)

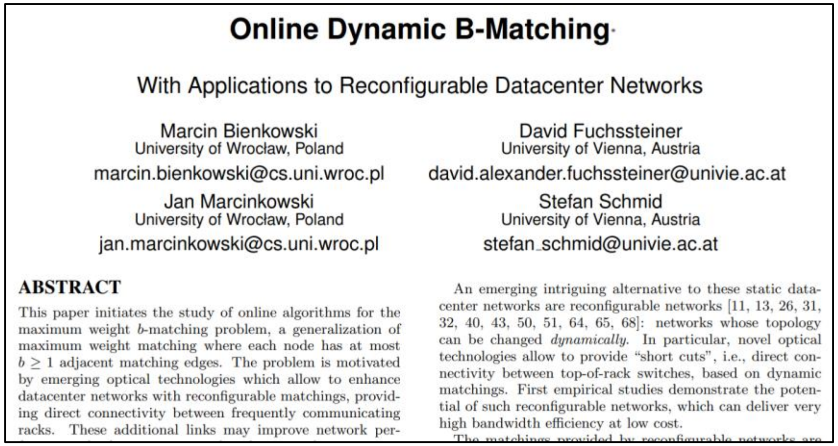
Theorem 2:

Under certain communication patterns, the amortized cost of ReNet can be significantly lower than the static optimum, i.e., Ω(log n).

- Example: consider sequence of $\sigma = (\sigma^{(1)}, \sigma^{(2)}, \sigma^{(3)}, \ldots)$ where each $\sigma^{(i)}$ is of length n log n, sparse and corresponds to different **2-dimensional grid**.
- In this example, the cost of ReNet is **constant** for each $\sigma^{(i)}$.
- Overall, the union of the grids form a uniform pattern, so the cost of the static algorithm is **log n** (for constant degree).

Further Reading

PERFORMANCE 2020, SPAA 2021, APOCS 2021



Online Dynamic B-Matching

With Applications to Reconfigurable Datacenter Networks

Marcin Bienkowski
University of Wrocław, Poland
marcin.bienkowski@cs.uni.wroc.pl

David Fuchssteiner
University of Vienna, Austria
david.alexander.fuchssteiner@univie.ac.at

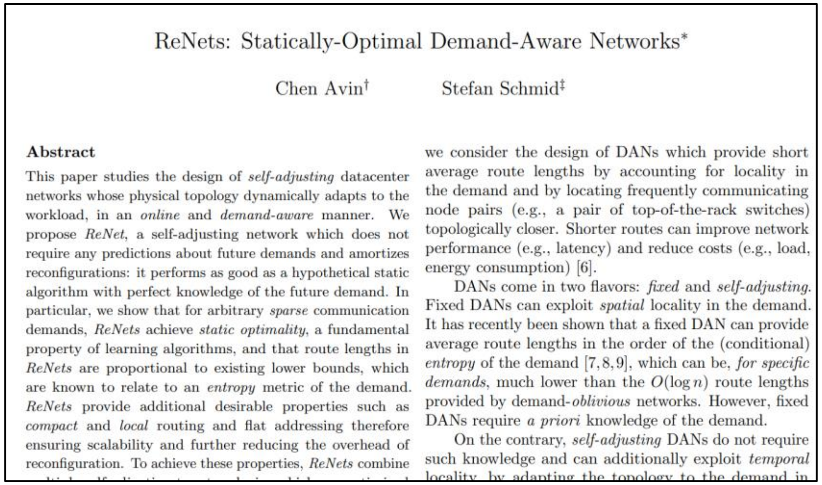
Jan Marcinkowski
University of Wrocław, Poland
jan.marcinkowski@cs.uni.wroc.pl

Stefan Schmid
University of Vienna, Austria
stefan_schmid@univie.ac.at

ABSTRACT

This paper initiates the study of online algorithms for the maximum weight b-matching problem, a generalization of maximum weight matching where each node has at most $b \geq 1$ adjacent matching edges. The problem is motivated by emerging optical technologies which allow to enhance datacenter networks with reconfigurable matchings, providing direct connectivity between frequently communicating racks. These additional links may improve network per-

An emerging intriguing alternative to these static datacenter networks are reconfigurable networks [11, 13, 26, 31, 32, 40, 43, 50, 51, 64, 65, 68]: networks whose topology can be changed *dynamically*. In particular, novel optical technologies allow to provide "short cuts", i.e., direct connectivity between top-of-rack switches, based on dynamic matchings. First empirical studies demonstrate the potential of such reconfigurable networks, which can deliver very high bandwidth efficiency at low cost.



ReNets: Statically-Optimal Demand-Aware Networks[*]

Chen Avin[†]   Stefan Schmid[‡]

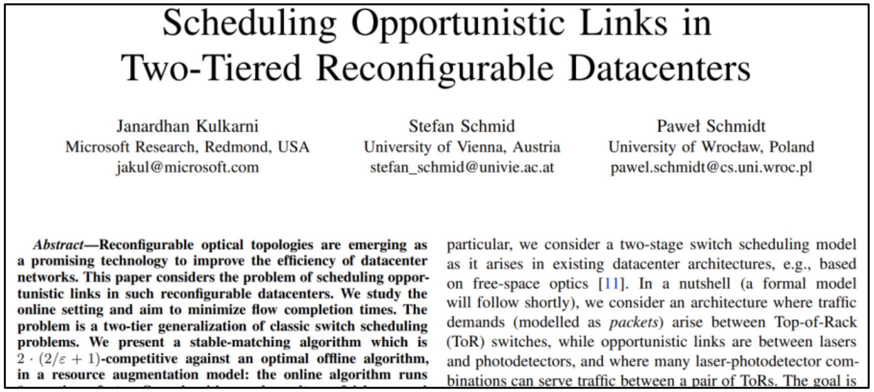
Abstract

This paper studies the design of *self-adjusting* datacenter networks whose physical topology dynamically adapts to the workload, in an *online* and *demand-aware* manner. We propose *ReNet*, a self-adjusting network which does not require any predictions about future demands and amortizes reconfigurations: it performs as good as a hypothetical static algorithm with perfect knowledge of the future demand. In particular, we show that for arbitrary *sparse* communication demands, *ReNets* achieve *static optimality*, a fundamental property of learning algorithms, and that route lengths in *ReNets* are proportional to existing lower bounds, which are known to relate to an *entropy* metric of the demand. *ReNets* provide additional desirable properties such as *compact* and *local* routing and flat addressing therefore ensuring scalability and further reducing the overhead of reconfiguration. To achieve these properties, *ReNets* combine

we consider the design of DANs which provide short average route lengths by accounting for locality in the demand and by locating frequently communicating node pairs (e.g., a pair of top-of-the-rack switches) topologically closer. Shorter routes can improve network performance (e.g., latency) and reduce costs (e.g., load, energy consumption) [6].

DANs come in two flavors: *fixed* and *self-adjusting*. Fixed DANs can exploit *spatial* locality in the demand. It has recently been shown that a fixed DAN can provide average route lengths in the order of the (conditional) *entropy* of the demand [7,8,9], which can be, *for specific demands*, much lower than the $O(\log n)$ route lengths provided by demand-*oblivious* networks. However, fixed DANs require *a priori* knowledge of the demand.

On the contrary, *self-adjusting* DANs do not require such knowledge and can additionally exploit *temporal*



Scheduling Opportunistic Links in Two-Tiered Reconfigurable Datacenters

Janardhan Kulkarni
Microsoft Research, Redmond, USA
jakul@microsoft.com

Stefan Schmid
University of Vienna, Austria
stefan_schmid@univie.ac.at

Paweł Schmidt
University of Wrocław, Poland
pawel.schmidt@cs.uni.wroc.pl

***Abstract*—Reconfigurable optical topologies are emerging as a promising technology to improve the efficiency of datacenter networks. This paper considers the problem of scheduling opportunistic links in such reconfigurable datacenters. We study the online setting and aim to minimize flow completion times. The problem is a two-tier generalization of classic switch scheduling problems. We present a stable-matching algorithm which is $2 \cdot (2/\varepsilon + 1)$-competitive against an optimal offline algorithm, in a resource augmentation model: the online algorithm runs**

particular, we consider a two-stage switch scheduling model as it arises in existing datacenter architectures, e.g., based on free-space optics [11]. In a nutshell (a formal model will follow shortly), we consider an architecture where traffic demands (modelled as *packets*) arise between Top-of-Rack (ToR) switches, while opportunistic links are between lasers and photodetectors, and where many laser-photodetector combinations can serve traffic between a pair of ToRs. The goal is