Self-Adjusting Trees Using Rotor Walks

Chen Avin, Marcin Bienkowski, Iosif Salem,
Robert Sama, ***Stefan Schmid,*** Paweł Schmidt

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

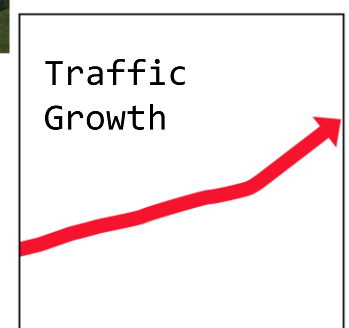
(Folklore)

Acknowledgements:

Trend

Data-Centric Applications

Datacenters ("hyper-scale")



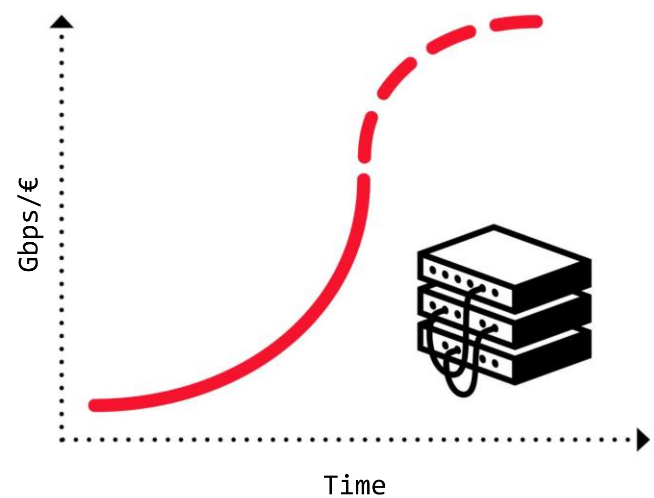
Source: Facebook

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

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**



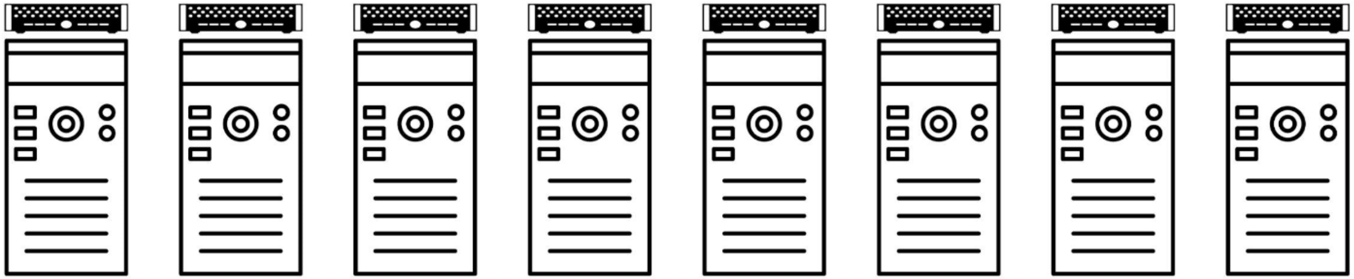
[1] Source: Microsoft, 2019

Annoying for companies, **opportunity** for researchers

Root Cause

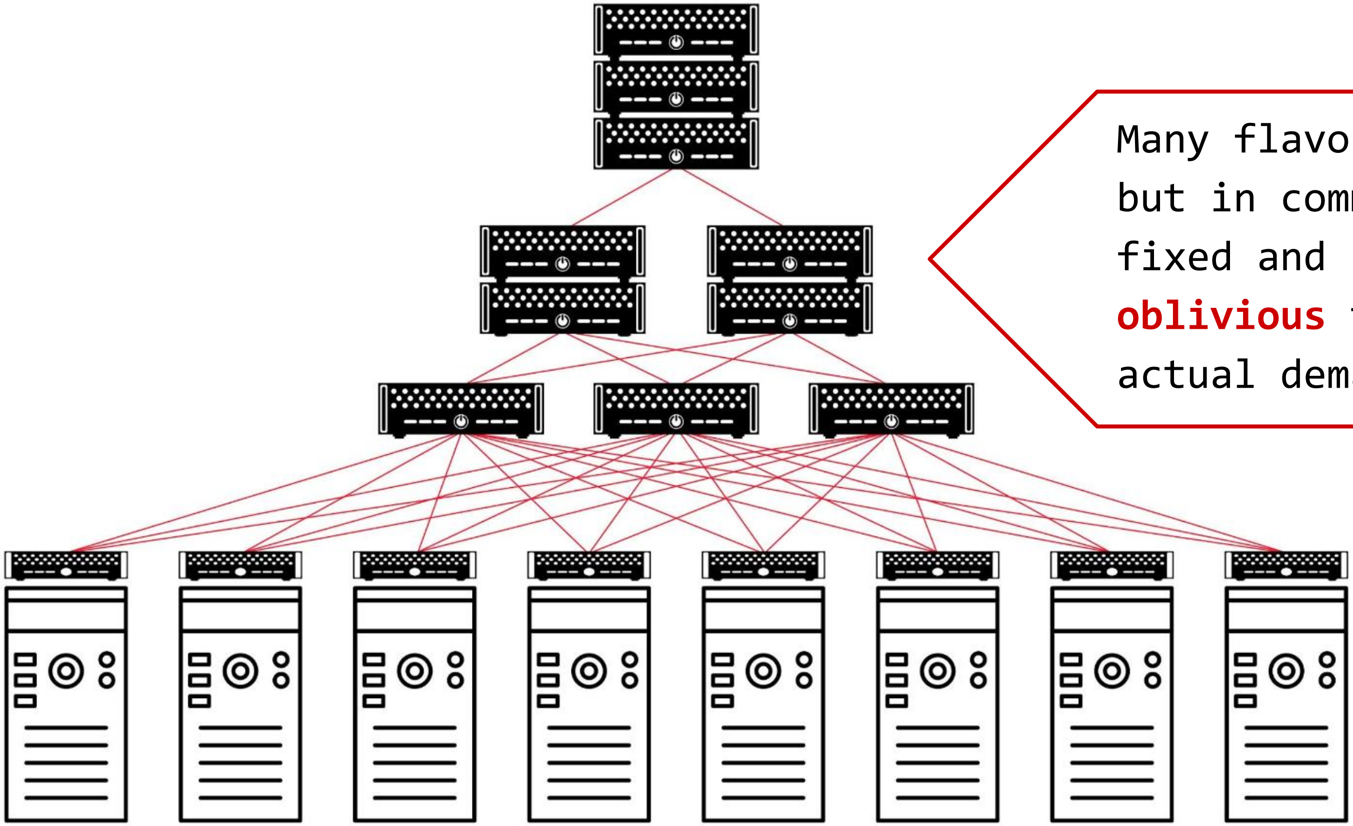
Fixed and Demand-Oblivious Topology

How to interconnect?



Root Cause

Fixed and Demand-Oblivious Topology

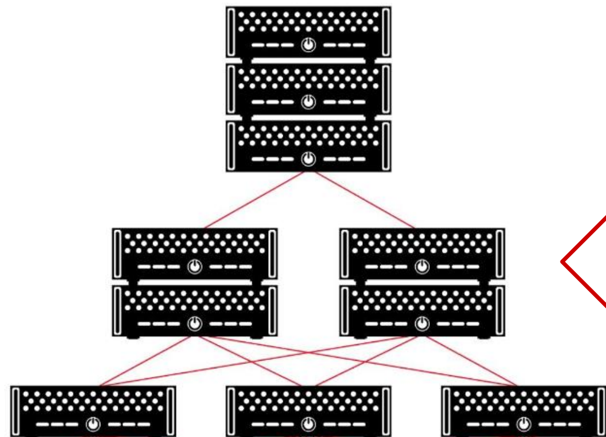
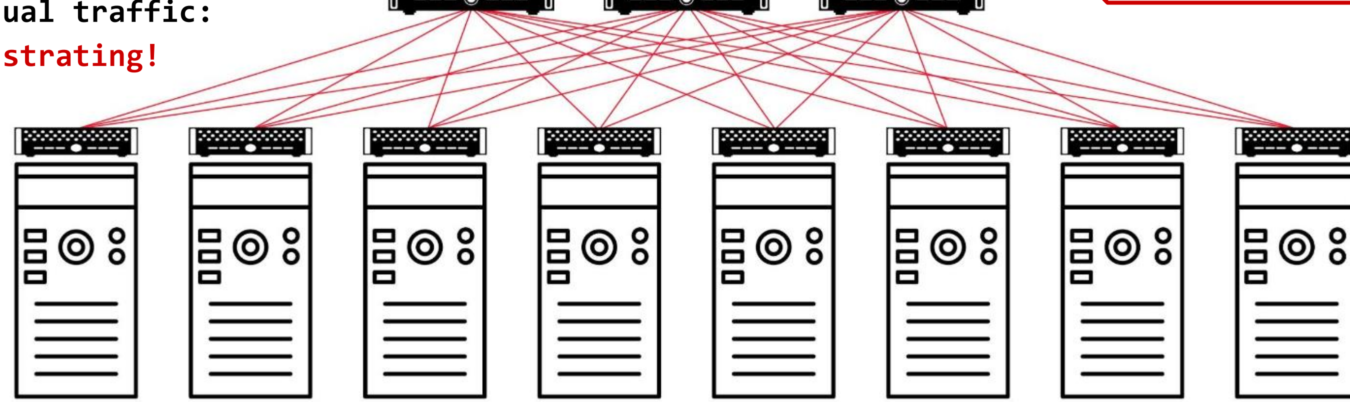


Many flavors,
but in common:
fixed and
oblivious to
actual demand.

Root Cause

Fixed and Demand-Oblivious Topology

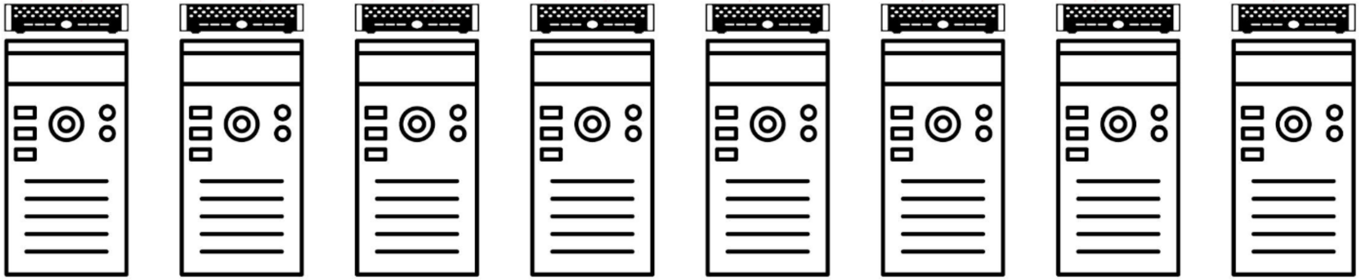
Highway which ignores actual traffic: **frustrating!**

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

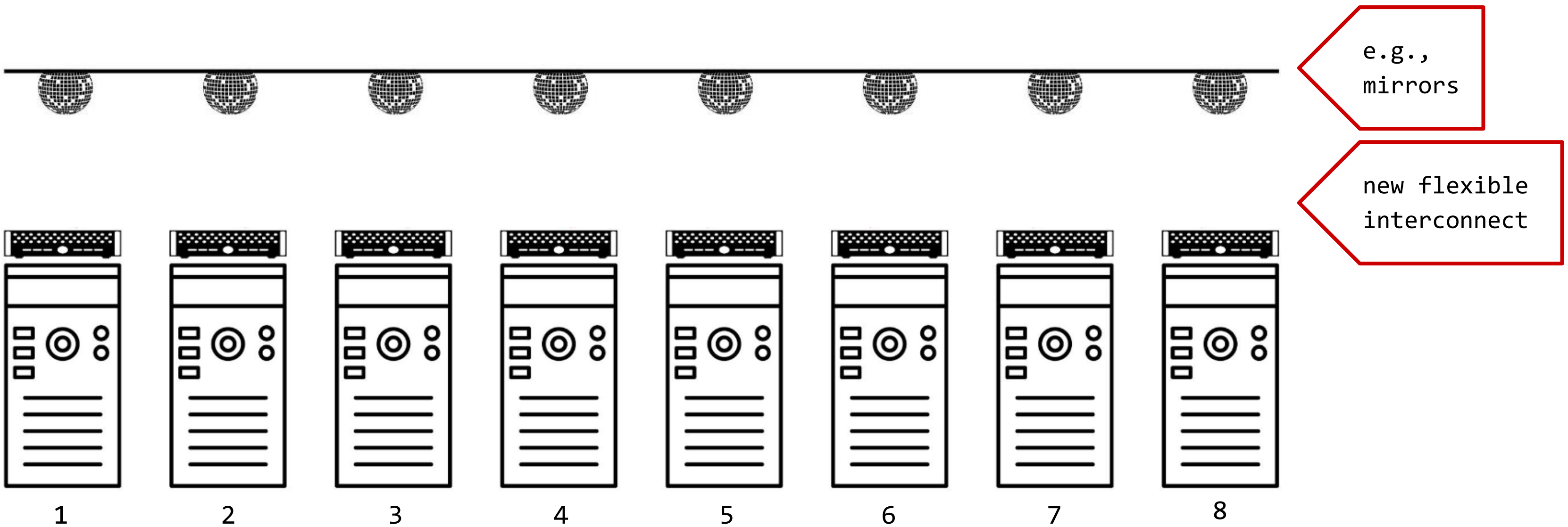
Our Vision

Flexible and Demand-Aware Topologies



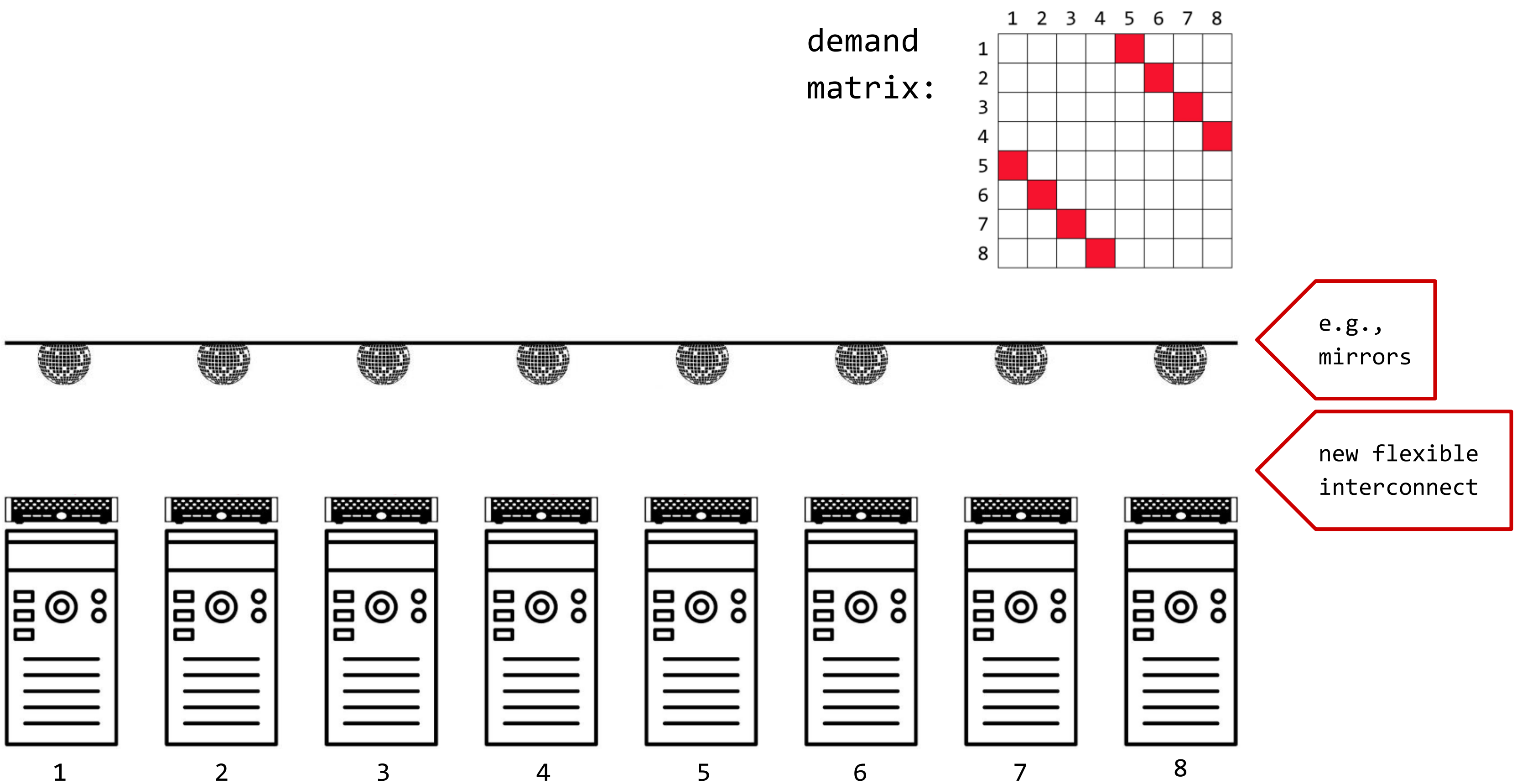
Our Vision

Flexible and Demand-Aware Topologies



Our Vision

Flexible and Demand-Aware Topologies

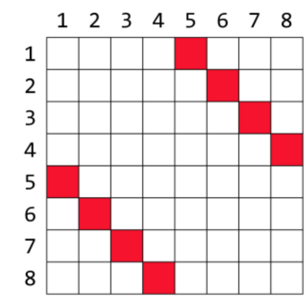


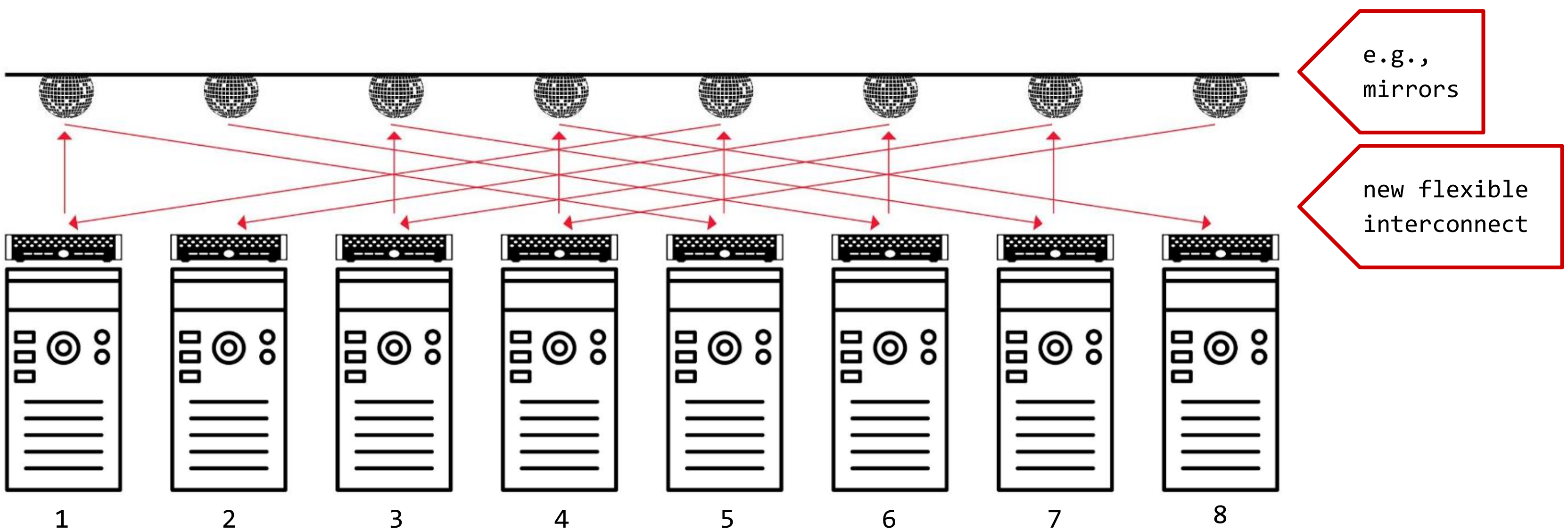
Our Vision

Flexible and Demand-Aware Topologies

Matches demand

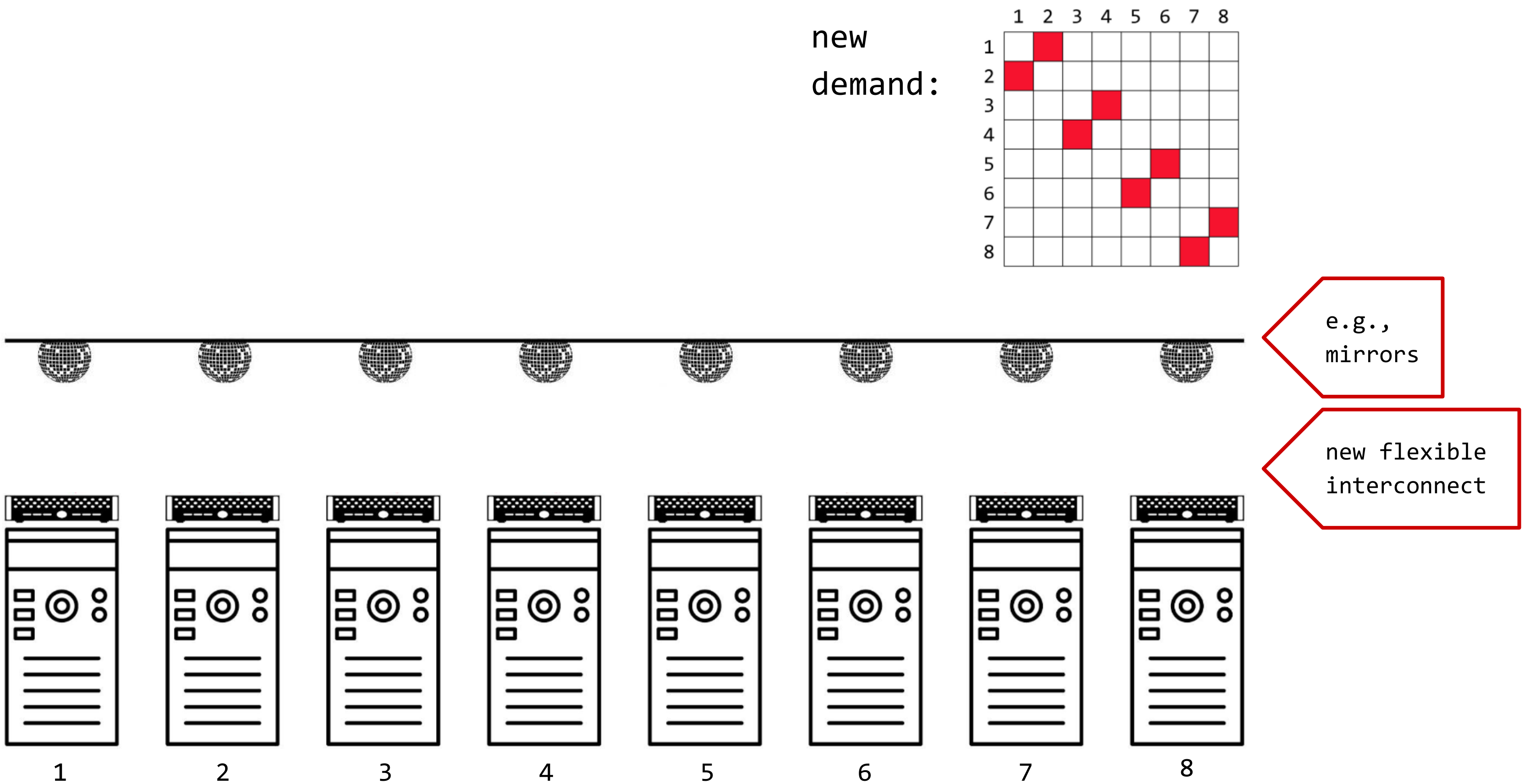
demand matrix:





Our Vision

Flexible and Demand-Aware Topologies

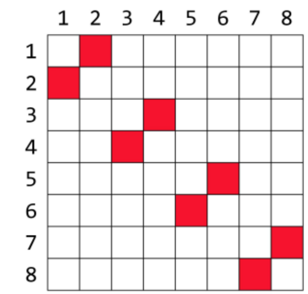


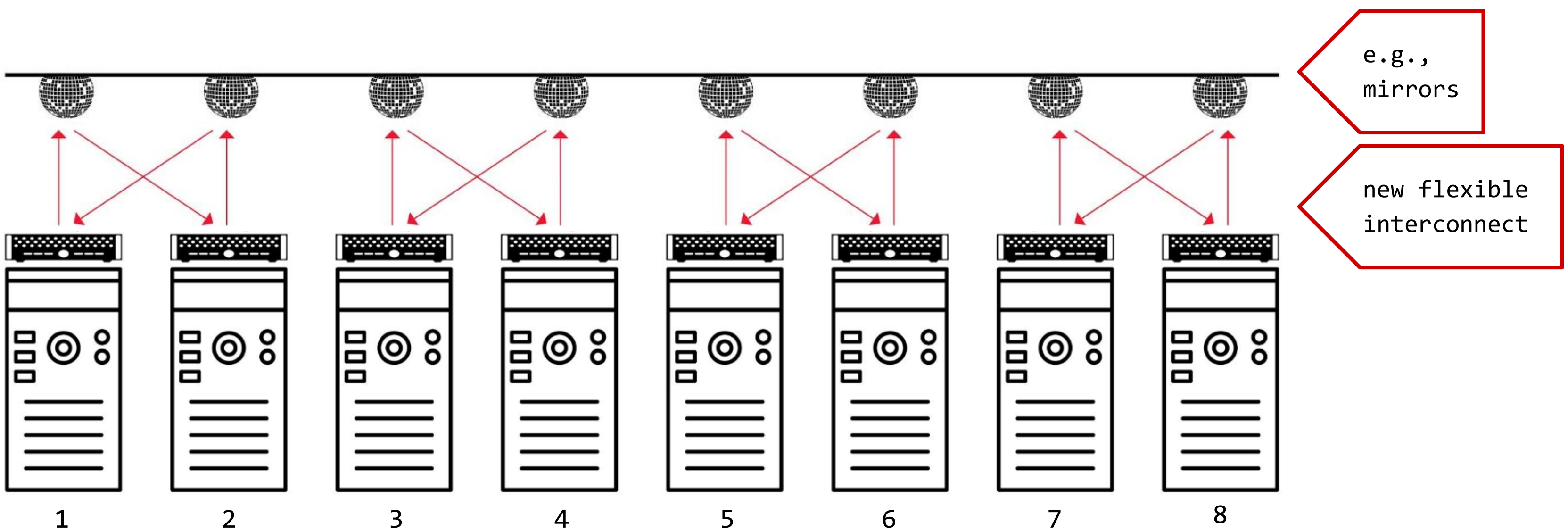
Our Vision

Flexible and Demand-Aware Topologies

Matches demand

new
demand:



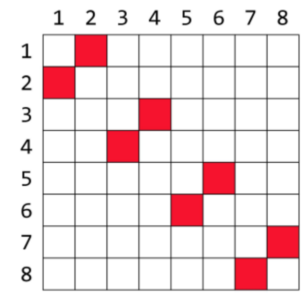


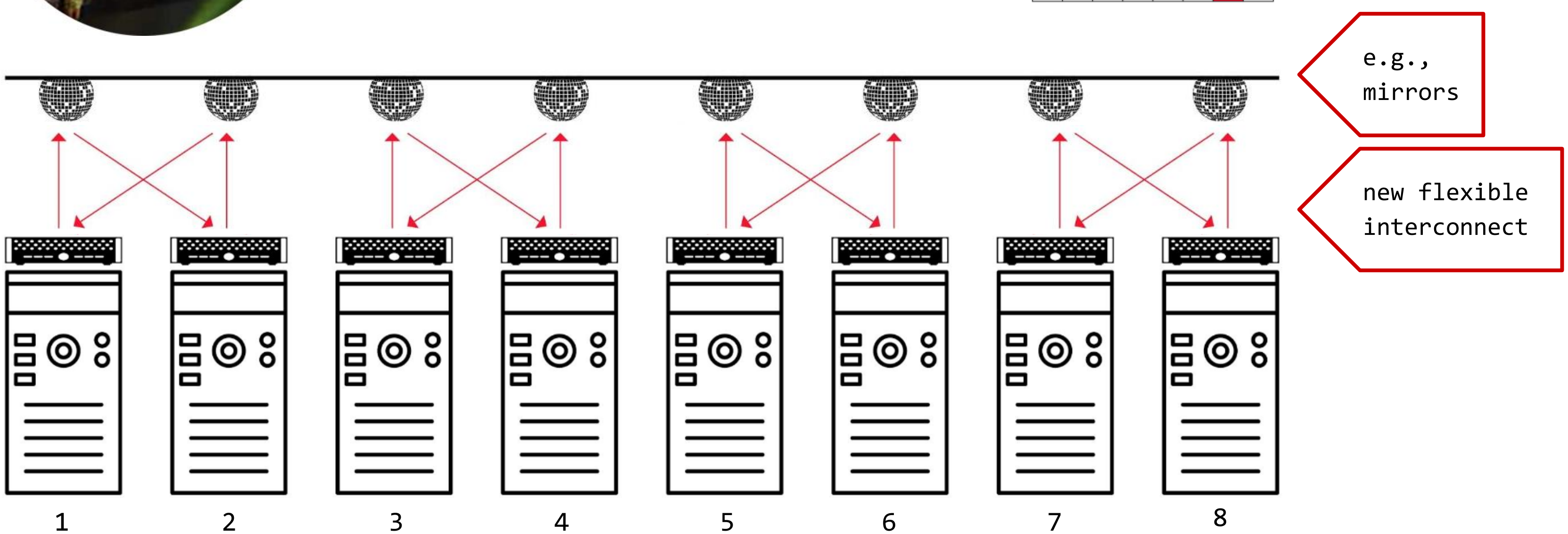
Our Vision

Flexible and Demand-Aware Topologies

Self-Adjusting Networks

new demand:





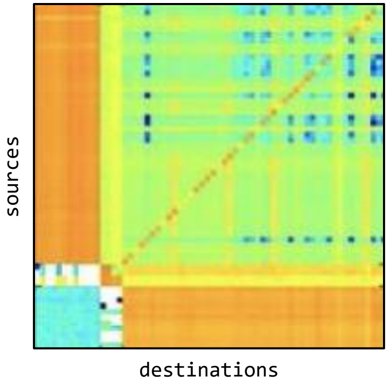
Our Motivation

Much Structure in the Demand

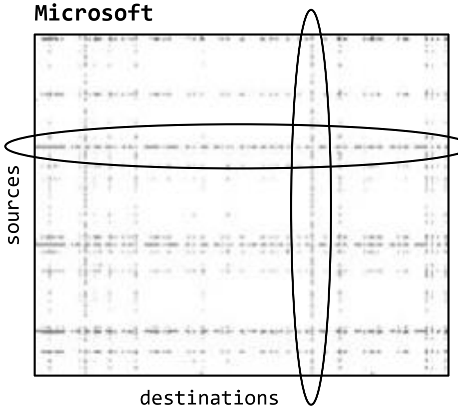
Empirical studies:

traffic matrices sparse and skewed

Facebook

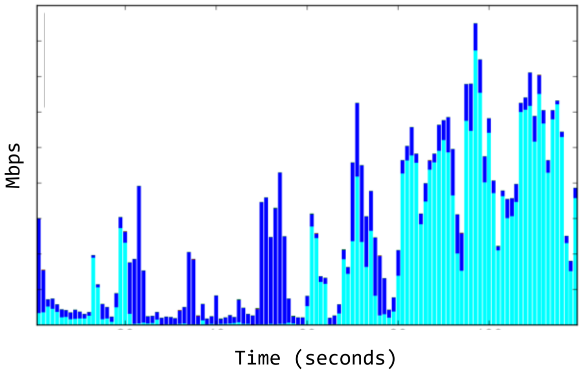


Microsoft



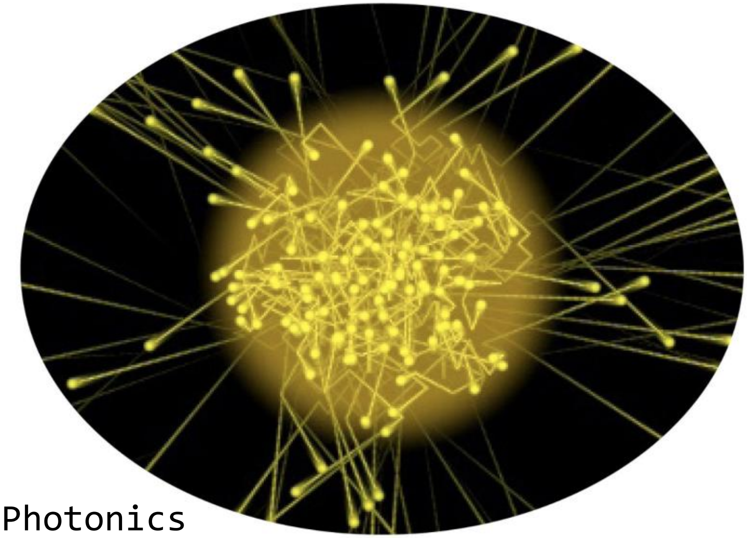
traffic bursty over time

Facebook



Our hypothesis: can be exploited.

Sounds Crazy?
Emerging Enabling Technology.



Photonics

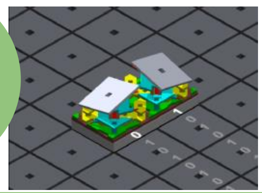
H2020:
“Photonics one of only five key enabling technologies for future prosperity.”

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

The Big Picture

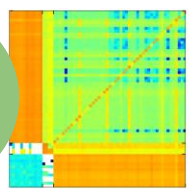
New!

Flexibility



More!

Structure



Self-Adjusting Networks

Now is the time!

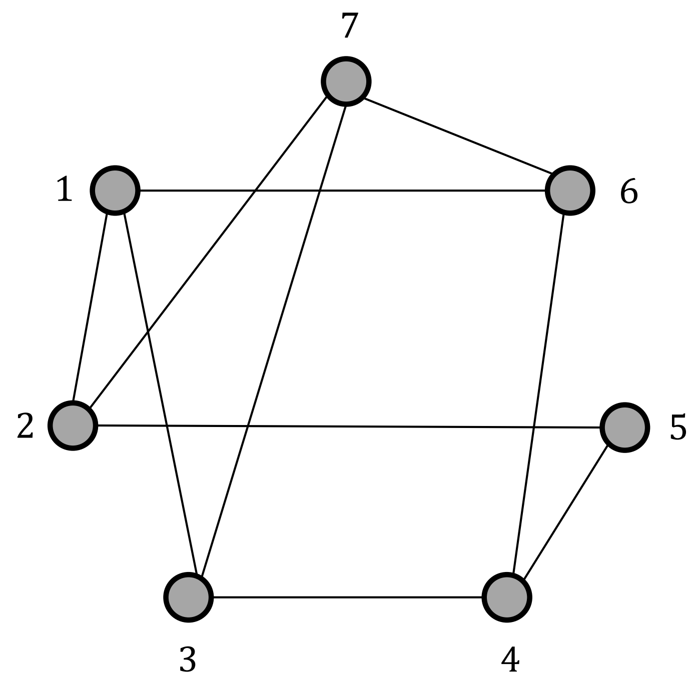
Efficiency

Static Problem

Demand-Aware Network of Bounded Degree

Destinations (columns), Sources (rows)

	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



$$\underbrace{\mathrm{WPL}(\mathcal{D},N)}_{\text{weighted path length}} = \sum_{(u,v)\in\mathcal{D}} p(u,v)\cdot d_N(u,v)$$

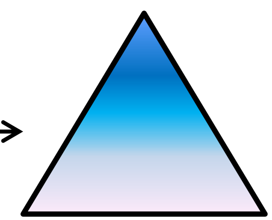
Static Algorithm

Reduction to Ego-Trees

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



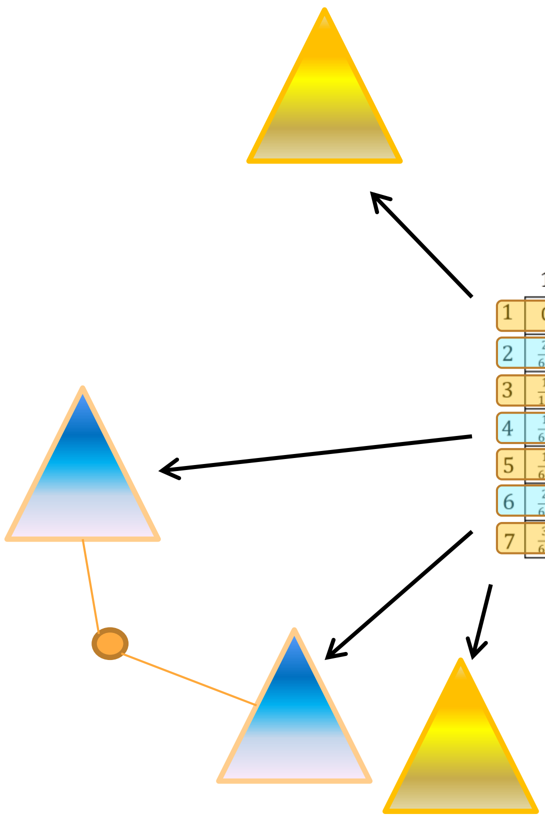
Huffman tree:
"**ego-tree**"

Static Algorithm

Reduction to Ego-Trees

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

Constant Approx

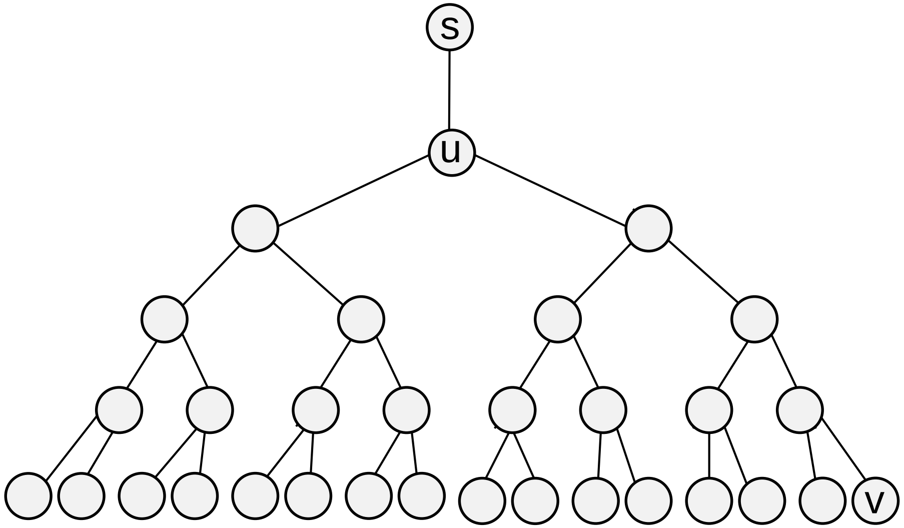


	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$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

This Paper: Dynamic

Maintaining Ego-Trees Dynamically

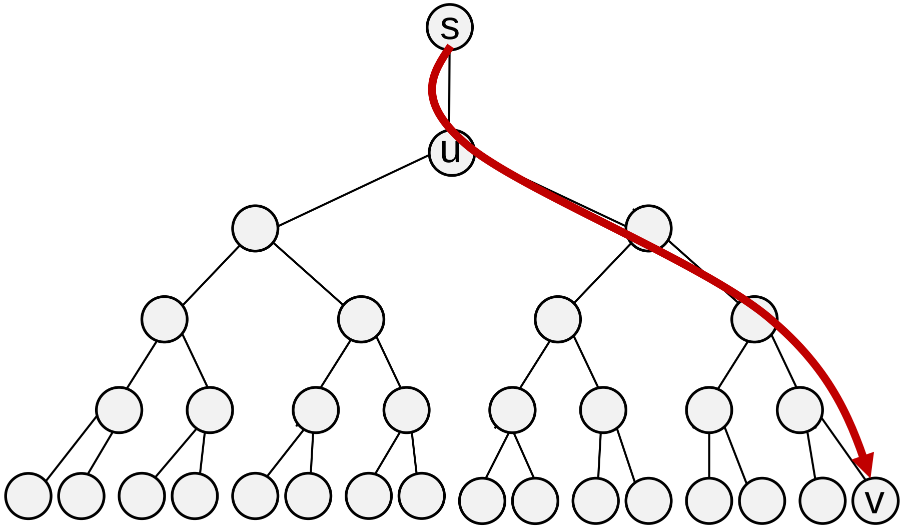
- Input: sequence of nodes $\sigma = (v_1, v_2, \ldots)$
- Cost: access cost + number of swaps
- Like splay trees, but **unordered** trees
- Goal: **online algorithm** which is **competitive** to offline
- Useful property: **most recently used (MRU)**



This Paper: Dynamic

Maintaining Ego-Trees Dynamically

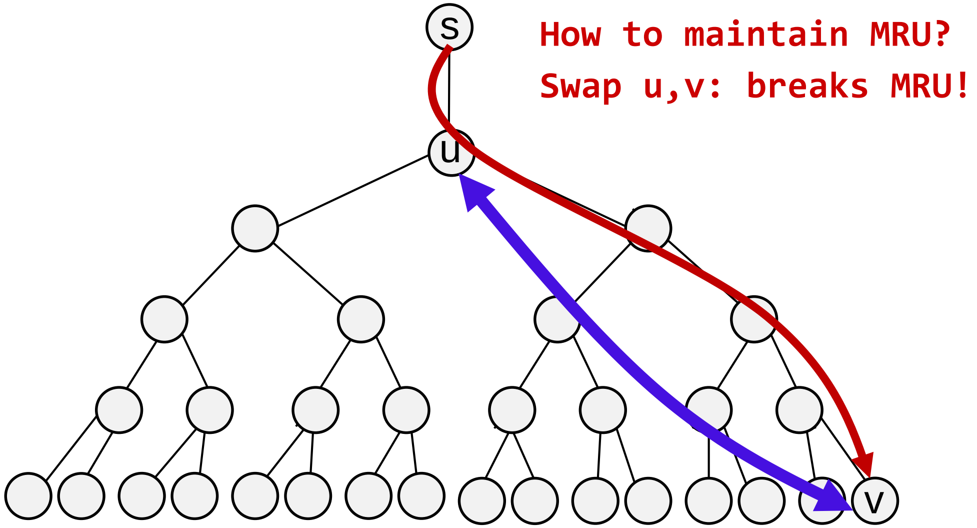
- Input: sequence of nodes $\sigma = (v_1, v_2, \ldots)$
- Cost: access cost + number of swaps
- Like splay trees, but **unordered** trees
- Goal: **online algorithm** which is **competitive** to offline
- Useful property: **most recently used (MRU)**



This Paper: Dynamic

Maintaining Ego-Trees Dynamically

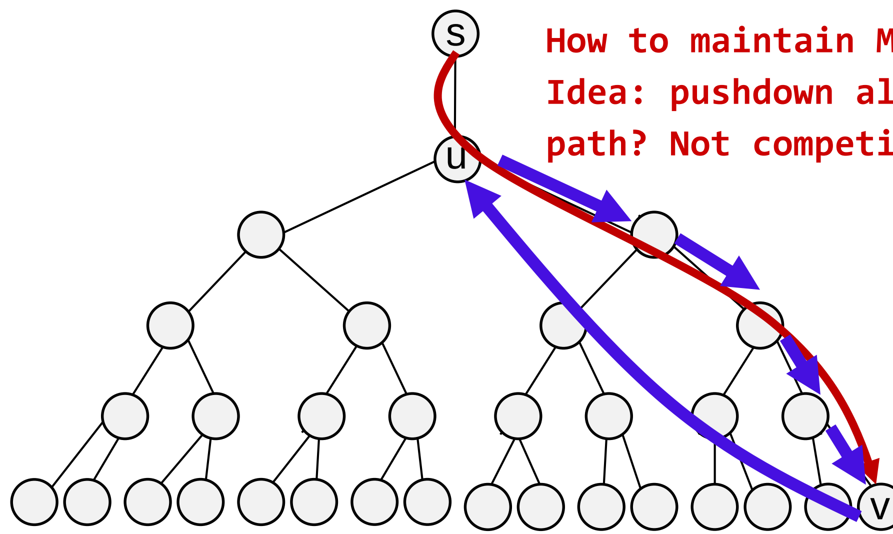
- Input: sequence of nodes $\sigma = (v_1, v_2, \ldots)$
- Cost: access cost + number of swaps
- Like splay trees, but **unordered** trees
- Goal: **online algorithm** which is **competitive** to offline
- Useful property: **most recently used (MRU)**



This Paper: Dynamic

Maintaining Ego-Trees Dynamically

- Input: sequence of nodes $\sigma = (v_1, v_2, \ldots)$
- Cost: access cost + number of swaps
- Like splay trees, but **unordered** trees
- Goal: **online algorithm** which is **competitive** to offline
- Useful property: **most recently used (MRU)**

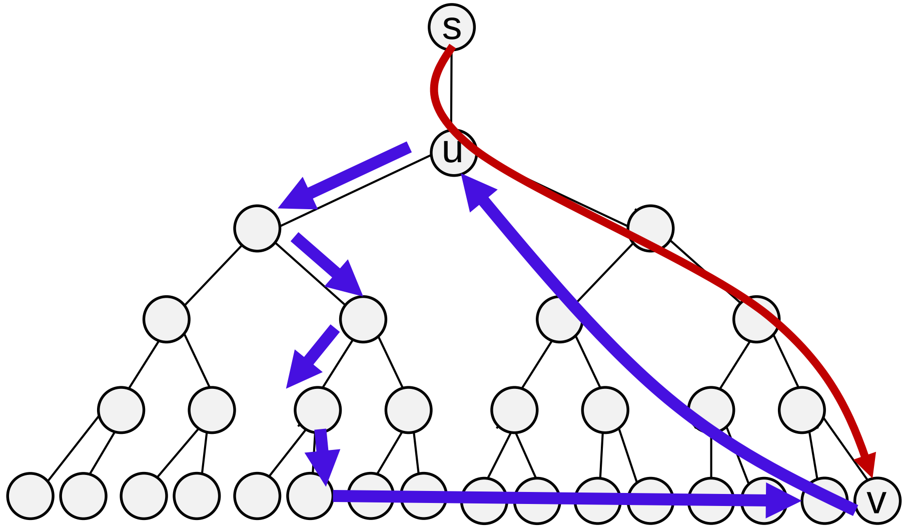


How to maintain MRU?
Idea: pushdown along path? Not competitive!

This Paper: Dynamic

Maintaining Ego-Trees Dynamically

- Input: sequence of nodes $\sigma = (v_1, v_2, \dots)$
- Cost: access cost + number of swaps
- Like splay trees, but **unordered** trees
- Goal: **online algorithm** which is **competitive** to offline
- Useful property: **most recently used (MRU)**

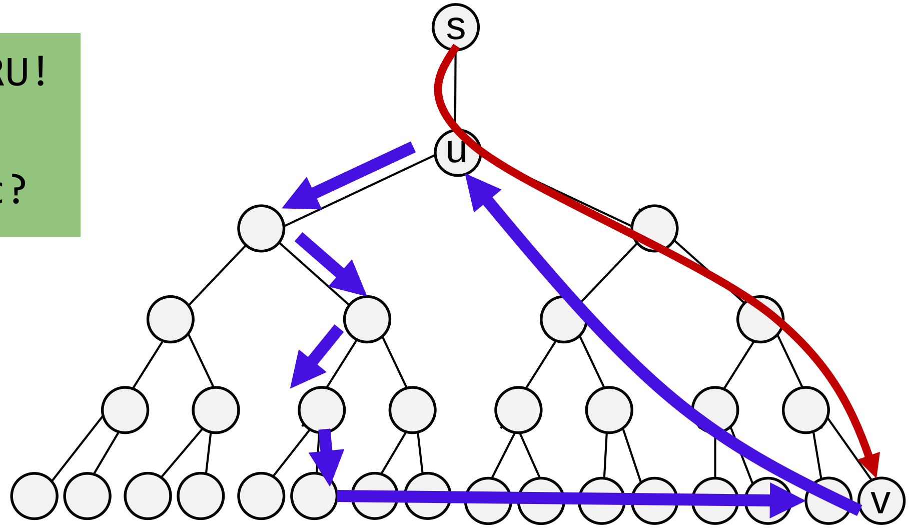


This Paper: Dynamic

Maintaining Ego-Trees Dynamically

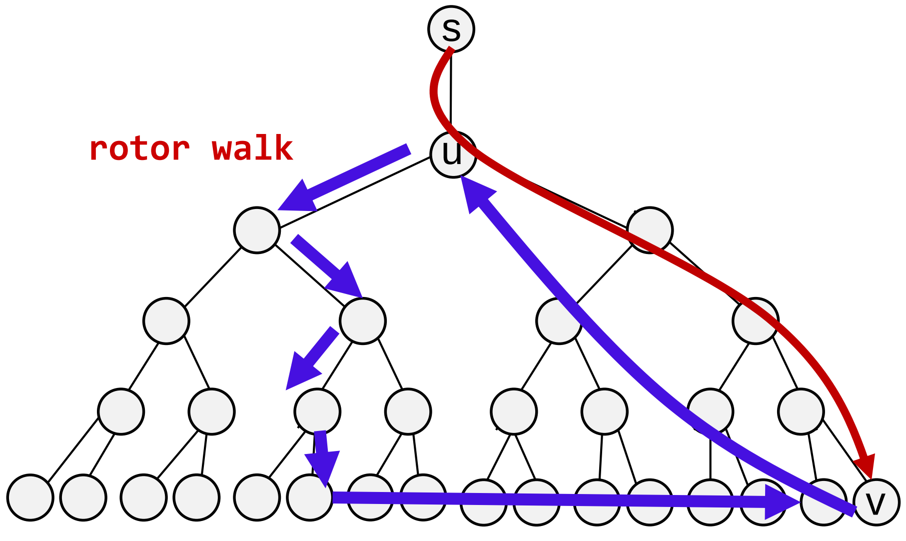
- Input: sequence of nodes $\sigma = (v_1, v_2, \ldots)$
- Cost: access cost + number of swaps
- Like splay trees, but **unordered** trees
- Goal: **online algorithm** which is **competitive** to offline
- Useful property: **most recently used (MRU)**

Random walk preservers MRU!
Constant competitive.
Competitive deterministic?



Our Contributions

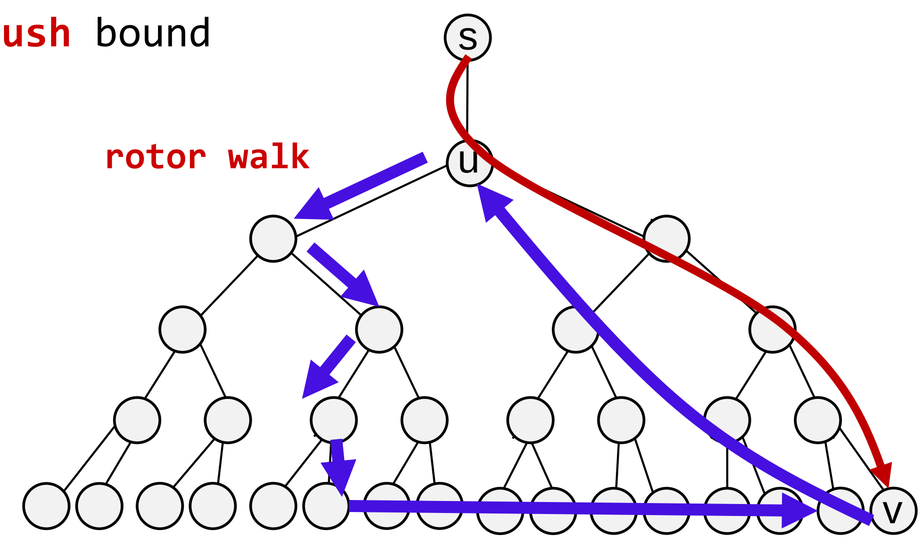
- Rotor-push: select push-down path by **rotor walk**
  - Each node has a toggle switch, left or right
  - Upon traversal, flip the switch
  - “Deterministic random walk”



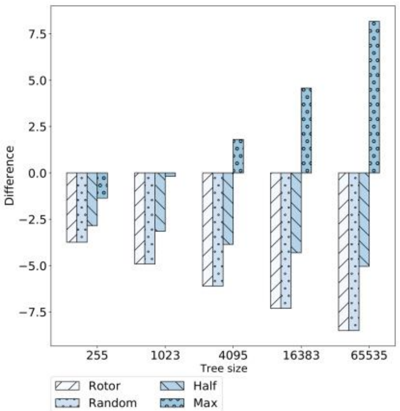
Our Contributions

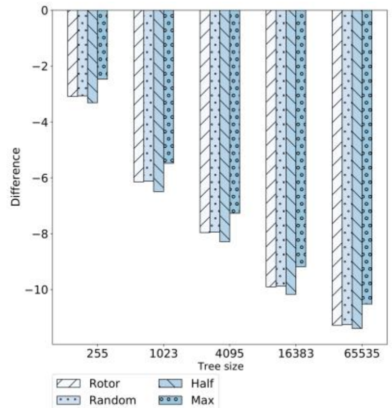
- Rotor-push: select push-down path by **rotor walk**
  - Each node has a toggle switch, left or right
  - Upon traversal, flip the switch
  - “Deterministic random walk”

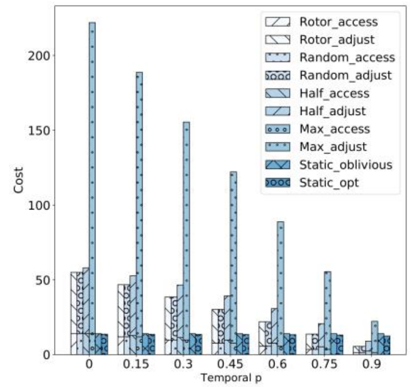
- Theorem: gives **12-competitive** tree
- We also improved **random push** bound from 60 to 16

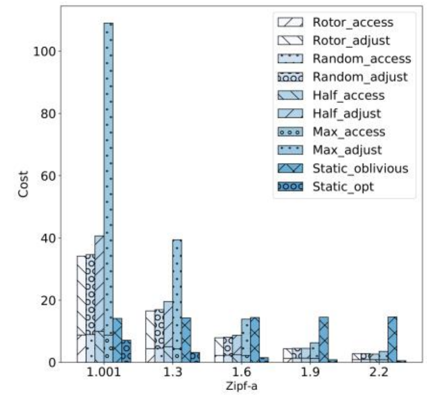


Empirical Results









Takeaway 1: The larger the network, the more beneficial self-adjustments compared to static

Takeaway 2: The more locality in the demand, the more beneficial as well.

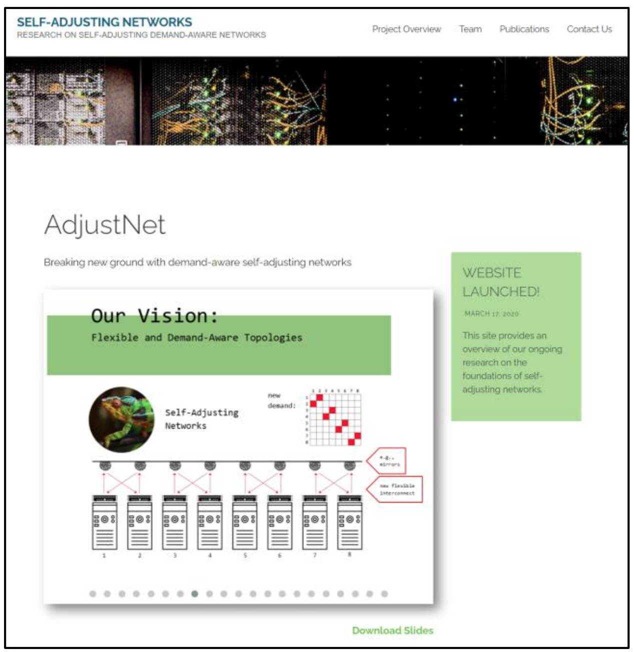
Takeaway 3: In practice, Rotor Push and Random Push have almost same cost.

Conclusion

- Self-adjusting tree: building block for self-adjusting general graphs (“datacenters”)
- Rotor walk: a constant-competitive online algorithm, finds optimal tradeoff between routing and adjustment costs
- Future work
  - Non-asymptotic analysis
  - Accounting also for load?

Thank you!

Websites



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

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

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,

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

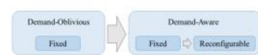


Figure 1: Taxonomy of topology optimization

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.,

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

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.

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.

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 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

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.

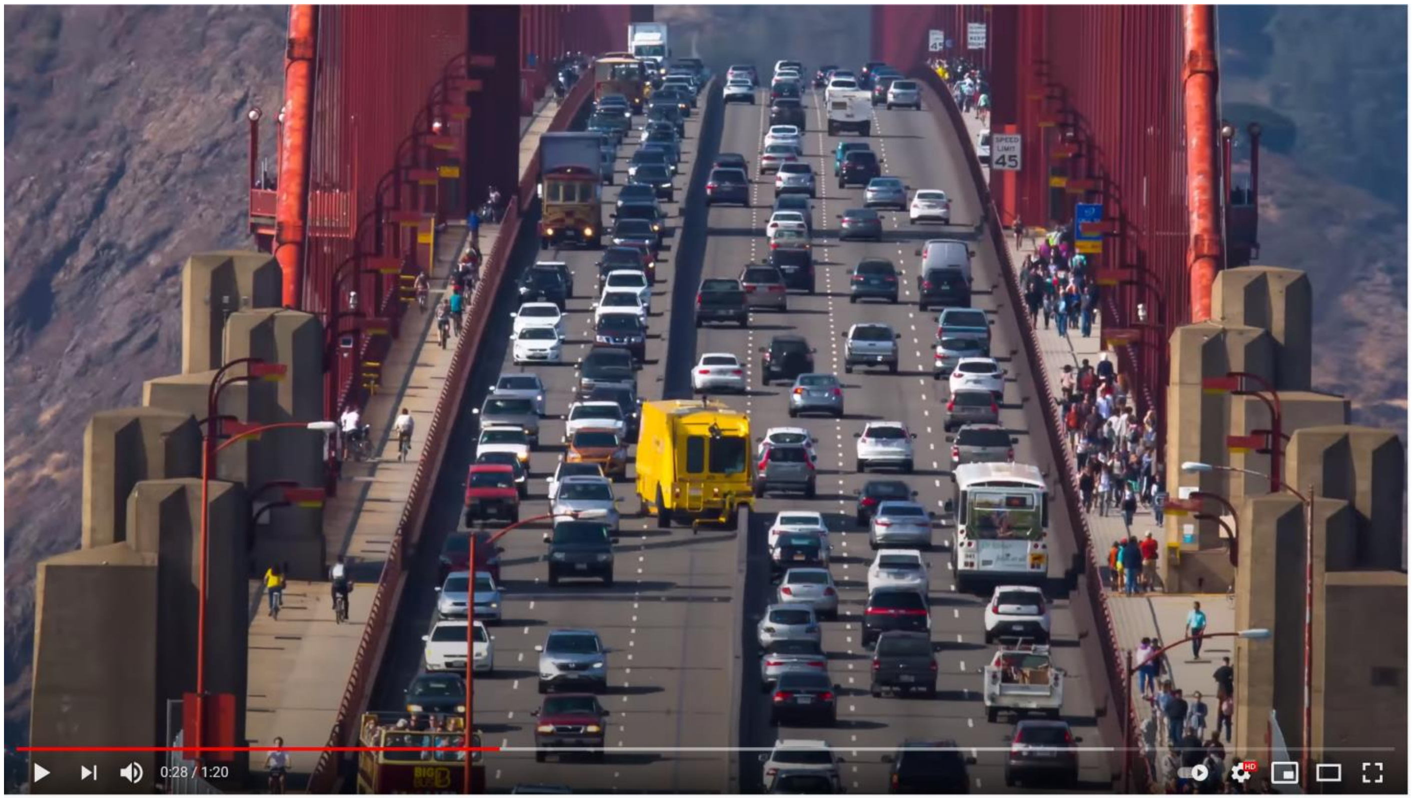
SplayNet: Towards Locally Self-Adjusting Networks
Stefan Schmid, Chen Avin, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler, and Zvi Lotker.
IEEE/ACM Transactions on Networking (**TON**), Volume 24, Issue 3, 2016. Early version: IEEE **IPDPS** 2013.

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

IEEE SPECTRUM

Topics Reports Blogs Multimedia Magazine Resources Search

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**

Photo illustration: Shutterstock

In HPC