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

(Folklore)
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

Annoying for companies, opportunity for researchers

[1] Source: Microsoft, 2019
Root Cause

Fixed and Demand-Oblivious Topology

How to interconnect?
Root Cause

Fixed and Demand-Oblivious Topology

Many flavors, but in common: fixed and oblivous 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

e.g., mirrors
new flexible interconnect
Our Vision
Flexible and Demand-Aware Topologies

e.g., mirrors
new flexible interconnect

demand matrix:
Our Vision
Flexible and Demand-Aware Topologies

Matches demand
Our Vision
Flexible and Demand-Aware Topologies

e.g., mirrors
new flexible interconnect

new demand:
Our Vision

Flexible and Demand-Aware Topologies

Matches demand

e.g., mirrors

new flexible interconnect

new demand:
Our Vision
Flexible and Demand-Aware Topologies

Self-Adjusting Networks

e.g., mirrors

new flexible interconnect

new demand:
Our Motivation
Much Structure in the Demand

Empirical studies:

traffic matrices **sparse** and **skewed**

traffic **bursty** over time

My **hypothesis**: can be exploited.
Sounds Crazy?
Emerging Enabling Technology.

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
Example
Optical Circuit Switch

→ Optical Circuit Switch rapid adaption of physical layer
  → Based on rotating mirrors
The Big Picture

Flexibility

Structure

Self-Adjusting Networks

Efficiency

Now is the time!
Our goal: Develop the theoretical 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

Our focus: in hardware
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
Intuition
Which demand has more structure?

→ Traffic matrices of two different distributed ML applications
  → GPU-to-GPU

More uniform

More structure
Intuition

Spatial vs temporal structure

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

→ Which one has more structure?
Intuition
Spatial vs temporal structure

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

→ Which one has more structure?

Systematically?
Trace Complexity
Information-Theoretic Approach
“Shuffle&Compress”
Trace Complexity

Information-Theoretic Approach
“Shuffle&Compress”

Increasing complexity (systematically randomized)

More structure (compresses better)
Trace Complexity

Information-Theoretic Approach

“Shuffle&Compress”
Trace Complexity

Information-Theoretic Approach
“Shuffle&Compress”

Difference in size (entropy)?
Difference in size (entropy)?
Trace Complexity
Information-Theoretic Approach
“Shuffle&Compress”

Can be used to define 2-dimensional complexity map!
Our Methodology

Complexity Map

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

Complexity Map

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

Different structures!
Our Methodology

Complexity Map

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:

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.
Oblivious networks
(worst-case traffic)

More structure: lower routing cost
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 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

Traditional 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

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

\[ \text{ERL}(\mathcal{D}, N) = \sum_{(u, v) \in \mathcal{D}} p(u, v) \cdot d_N(u, v) \]
### Case Study “Route Lengths”

#### Constant-Degree Demand-Aware Network

![Graph Image]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2/65</td>
<td>1/13</td>
<td>1/65</td>
<td>1/65</td>
<td>2/65</td>
<td>3/65</td>
</tr>
<tr>
<td>2</td>
<td>2/65</td>
<td>0</td>
<td>1/65</td>
<td>0</td>
<td>0</td>
<td>2/65</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1/13</td>
<td>1/65</td>
<td>0</td>
<td>2/65</td>
<td>0</td>
<td>0</td>
<td>1/13</td>
</tr>
<tr>
<td>4</td>
<td>1/65</td>
<td>0</td>
<td>2/65</td>
<td>0</td>
<td>4/65</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1/65</td>
<td>0</td>
<td>3/65</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>2/65</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3/65</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>3/65</td>
<td>2/13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3/65</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
ERL(\mathcal{D},N) = \sum_{(u,v) \in \mathcal{D}} p(u,v) \cdot d_N(u,v)
\]
Case Study “Route Lengths”

Constant-Degree Demand-Aware Network

\[
\text{ERL}(\mathcal{D},N) = \sum_{(u,v) \in \mathcal{D}} p(u,v) \cdot d_N(u,v)
\]
Case Study “Route Lengths”

Constant-Degree Demand-Aware Network

**ERL(Ω,N) = \sum_{(u,v) \in \Omega} p(u,v) \cdot d_N(u,v)**
Examples

→ DAN for $\triangle=3$
  → E.g., complete binary tree would be $\log n$
  → Can we do better?

→ DAN for $\triangle=2$
  → Set of lines and cycles
Examples

→ DAN for $\triangle=3$
  → E.g., complete binary tree would be $\log n$
  → Can we do better?

→ DAN for $\triangle=2$
  → Set of lines and cycles

How hard?
Related Problem

Virtual Network Embedding Problem (VNEP)

Example $\Delta=2$: A Minimum Linear Arrangement (MLA) Problem
→ Minimizes sum of virtual edges
Related Problem

Virtual Network Embedding Problem (VNEP)

Example $\triangle=2$: A Minimum Linear Arrangement (MLA) Problem
\[
\rightarrow \text{Minimizes sum of virtual edges}
\]

Bad!
Related Problem

Virtual Network Embedding Problem (VNEP)

Example $\triangle=2$: A Minimum Linear Arrangement (MLA) Problem
→ Minimizes sum of virtual edges
Related Problem

Virtual Network Embedding Problem (VNEP)

Example $\Delta=2$: A Minimum Linear Arrangement (MLA) Problem
  $\rightarrow$ Minimizes sum of virtual edges

MLA is NP-hard
  $\rightarrow$ ... and so is our problem!
Virtual Network Embedding Problem (VNEP)

Example $\Delta=2$: A Minimum Linear Arrangement (MLA) Problem

$\rightarrow$ Minimizes sum of virtual edges

MLA is \textbf{NP-hard}

$\rightarrow$ ... and so is our problem!

But what about $\Delta>2$?

$\rightarrow$ Embedding problem still hard

$\rightarrow$ But we have a new \textit{degree of freedom}!
Related Problem

Virtual Network Embedding Problem (VNEP)

Example $\Delta=2$: A Minimum Linear Arrangement (MLA) Problem

→ Minimizes sum of virtual edges

MLA is NP-hard

→ ... and so is our problem!

But what about $\Delta>2$?

→ Embedding problem still hard
→ But we have a new degree of freedom!

Simplifies problem?!
### Entropy Lower Bound

#### Huffman Tree: "ego-tree"

<table>
<thead>
<tr>
<th>Sources</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>\frac{2}{65}</td>
<td>\frac{1}{13}</td>
<td>\frac{1}{65}</td>
<td>\frac{1}{65}</td>
<td>\frac{2}{65}</td>
<td>\frac{3}{65}</td>
</tr>
<tr>
<td>2</td>
<td>\frac{2}{65}</td>
<td>0</td>
<td>\frac{1}{65}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>\frac{2}{65}</td>
</tr>
<tr>
<td>3</td>
<td>\frac{1}{13}</td>
<td>\frac{1}{65}</td>
<td>0</td>
<td>\frac{2}{65}</td>
<td>0</td>
<td>0</td>
<td>\frac{1}{13}</td>
</tr>
<tr>
<td>4</td>
<td>\frac{1}{65}</td>
<td>0</td>
<td>\frac{2}{65}</td>
<td>0</td>
<td>\frac{4}{65}</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>\frac{1}{65}</td>
<td>0</td>
<td>\frac{3}{65}</td>
<td>\frac{4}{65}</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>\frac{2}{65}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>\frac{3}{65}</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>\frac{3}{65}</td>
<td>\frac{2}{65}</td>
<td>\frac{1}{13}</td>
<td>0</td>
<td>0</td>
<td>\frac{3}{65}</td>
<td>0</td>
</tr>
</tbody>
</table>
Entropy Lower Bound

\[ \text{ERL} = \Omega(H_\Delta(Y|X)) \]
Idea for algorithm:

- union of trees
- reduce degree
- but keep distances
\textbf{Entropy Upper Bound}

\begin{itemize}
  \item Idea for algorithm:
    \begin{itemize}
      \item union of trees
      \item reduce degree
      \item but keep distances
    \end{itemize}

  \item Ok for sparse demands
    \begin{itemize}
      \item not everyone gets tree
      \item helper nodes
    \end{itemize}
\end{itemize}
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
Still use ego-trees

But balance for load
Further Reading

TON 2016, DISC 2017, CCR 2019, INFOCOM 2019

Demand-Aware Network Designs of Bounded Degree

Chen Avin1, Kaushik Mondal1, and Stefan Schmid2
1 Communication Systems Engineering Department
Ben Gurion University of the Negev, Israel
avin@cs.ece.bgu.ac.il, mondal1post.bgu.ac.il
2 Department of Computer Science
Aalborg University, Denmark
schmidte@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. In particular, we study the design of self-adjusting datacenter interconnects. The main conclusion of this work is that self-adjusting networks should be seen through the lens of self-adjusting datacenter interconnects.

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. Our vision is that a distributed generalization of the self-adjusting datastructures introduced by Sleator and Tarjan [22]: In contrast to their splay trees which dynamically optimize the search 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 [9, 11] have to support routing requests between arbitrary nodes or peers of communicating nodes; in other words, both the source as well as the destination of the request can vary. Figure 1 illustrates the difference between classic and distributed binary search trees.

In this paper, we ask: Can we gain similar benefits from self-adjusting networks, and how can we design efficient datacenter networks based on this abstraction? The answer is yes, but the design is non-trivial.

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

Chen Avin
Ben Gurion University, Israel
avin@cs.ece.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 the 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 lens of self-adjusting datacenter networks.

As a first step towards this understanding, we introduce the concepts of Demand-Aware and Demand-Oblivious systems. Figure 1 illustrates the taxonomy of topology optimization. In particular, Demand-Oblivious systems are oblivious to the topology and cannot make topology aware decisions, whereas Demand-Aware systems have access to topology information and can therefore make topology aware decisions.

In this paper, we ask: Can we gain similar benefits from self-adjusting networks, and how can we design efficient datacenter networks based on this abstraction? The answer is yes, but the design is non-trivial.

Demand-Aware Network Design with Minimal Congestion and Route Lengths

Chen Avin
Communication Systems Engineering Dept.
Ben Gurion University of the Negev, Israel
avin@cs.ece.bgu.ac.il

Kaushik Mondal
Communication Systems Engineering Dept.
Ben Gurion University of the Negev, Israel
mondal1post.bgu.ac.il

Stefan Schmid
Department of Computer Science
University of Vienna, Austria
stefan.schmid@univie.ac.at

ABSTRACT

Emerging communication technologies allow for networks to adapt dynamically and in a decentralized manner, to the communication pattern. Our vision is that a distributed generalization of the self-adjusting datastructures introduced by Sleator and Tarjan [24]: In contrast to their splay trees which dynamically optimize the search 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, 11] have to support routing requests between arbitrary nodes or peers of communicating nodes; in other words, both the source as well as the destination of the request can vary. Figure 1 illustrates the difference between classic and distributed binary search trees.

In this paper, we ask: Can we gain similar benefits from self-adjusting networks, and how can we design efficient datacenter networks based on this abstraction? The answer is yes, but the design is non-trivial.
Dynamic Setting

→ Dynamic the same:
  → union of dynamic ego-trees

→ E.g., SplayNets

→ Online algorithms
Dynamic Setting

& distributed

→ Dynamic the same:
→ union of dynamic & distributed ego-trees

→ E.g., SplayNets or CB trees

→ Online algorithms
### Dynamic Objectives

**Awareness**
- Demand-Aware

**Topology**
- Reconfigurable

**Input**
- Offline
- Online

**Algorithm**
- OFF
- ON

**Property**
- Static Optimality
- Dynamic Optimality
- Working Set

---

3.3 Additional Properties

Besides the properties that are specific... to be changed in order to transform the network.
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)
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, but still constant competitive!
An Alternative: SplayNets

Idea: generalize splay trees to networks

Splay Tree vs BST is nice for networks: *Local (greedy) search!*

Splay Tree

SplayNet
**SplayNets: A Simple Idea**

@t: access x

Splay Tree

@t+1

SplayNet

@t: comm (x,y)

x

splay

y

LCA

@t+1

double-splay

x
Properties of SplayNets

→ Statically optimal if demand comes from a *product distribution*
  - Product distribution: entropy equals conditional entropy, i.e.,
    \[ H(X) + H(Y) = H(X|Y) + H(X|Y) \]

→ Converges to optimal static topology in
  - **Multicast scenario**: requests come from a binary tree as well
  - **Cluster scenario**: communication only within interval
  - **Laminated scenario**: communication is „non-crossing matching“
More Specifically

Cluster scenario: SplayNet will converge to state where paths between cluster nodes only includes cluster nodes.

Non-crossing matching scenario: SplayNet will converge to state where all communication pairs are adjacent.
Further Reading

TON 2016, LATIN 2020, IPDPS 2021

SplayNet: Towards Locally Self-Adjusting Networks

Stefan Schmid1, Chen Avin1, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler, Zvi Lotker

Abstract—This paper initiates the study of locally self-adjusting networks whose topology adapts dynamically and in a decentralized manner, to the communication pattern $r$. Our vision can be seen as a distributed generalization of the self-adjusting data structures 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 lowest-common-ancestor queries.

Dynamically Optimal Self-Adjusting Single-Source Tree Networks

Chen Avin1, Kaushik Monda2, and Stefan Schmid3
1 Ben-Gurion University of the Negev, Israel
2 Indian Institute of Technology Roorkee, 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 Souza1, Olga Goussevskaia2, Stefan Schmid2
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
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:
    $\sigma = (\sigma_1, \sigma_2, \sigma_3, \ldots)$
  → Global controller

→ **Objective**: Minimize route length
  → plus reconfigurations
    → More specifically:
      be **statically optimal**
    → Compared to a fixed algorithm
      which knows $\sigma$ ahead of time

**Bonus**
→ 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 $\sigma$.

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
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
Theorem 1:

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

\begin{itemize}
  \item Sparse: subsequences of only involve a linear number of nodes
  \item Required to ensure availability of helper nodes (DISC 2017)
\end{itemize}
Analytical Results (2)

Theorem 2:

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

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).
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 ≥ 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. Those additional links may improve network performance.

Scheduling Opportunistic Links in Two-Tiered Reconfigurable Datacenters
Janardhan Kulkarni
Microsoft Research, Redmond, USA
jakart@ microsoft.com
Stefan Schmid
University of Vienna, Austria
stefan.schmid@univie.ac.at
Pawel 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 · (2/c + 1)-competitive against an optimal offline algorithm, in a resource augmentation model: the online algorithm runs particularly, 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, where many laser-photodetector combinations can serve traffic between a pair of ToRs. The goal is

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 ReNets, 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 be related to an entropy metric of the demand. ReNets provide additional desirable properties such as compact and local routing and that addressing therefore ensuring scalability and further reducing the overhead of reconfiguration. To achieve these properties, ReNets combine techniques from previous work with new ideas.

Further Reading
PERFORMANCE 2020,
SPAA 2021, APOCS 2021
Notion of self-adjusting networks opens a large uncharted field with many questions:

- Metrics and algorithms: by how much can load be lowered, energy reduced, quality-of-service improved, etc. in demand-aware networks? Even for route length not clear!
- How to model reconfiguration costs?
- Impact on other layers?

Requires knowledge in networking, distributed systems, algorithms, performance evaluation.
Conclusion

→ Demand-aware networks
  → Much potential...
  → ... if demand has structure
  → Metrics? E.g., entropy

→ Avenues for future work
  → Dense communication
  → Dynamic optimality
  → Distributed control plane

Thank you!

A Self-Adjusting Search Tree
by Jorge Stolfi (1987)
Websites

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

http://self-adjusting.net/trace-collection.net/
Trace collection website
Further Reading

**Static DAN**

**Overview: Models**

**Static Optimality**

**Robust DAN**

**Dynamic DAN**

**Concurrent DANs**
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.

Toward Demand-Aware Networking: A Theory for Self-Adjusting Networks (Editorial)
Chen Avin and Stefan Schmid.

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.

Distributed Self-Adjusting Tree Networks
Bruna Peres, Otavio Augusto de Oliveira Souza, Olga Goussevskaia, Chen Avin, and Stefan Schmid.

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.

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.

Characterizing the Algorithmic Complexity of Reconfigurable Data Center Architectures
Klaus-Tycho Foerster, Manya Ghobadi, and Stefan Schmid.
Bonus Material

Golden Gate Zipper
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
Focus Topic: Analysis of ProjecToR

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

Pawel Schmidt  
University of Wroclaw, 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 \(2 \cdot (2/e + 1)\)-competitive against an optimal offline algorithm, in a resource augmentation model: the online algorithm runs in 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...
A 2-Tiered Architecture

Reconfigurable network of ProjecToR relies on a 2-tiered architecture:

- Traffic demands (modelled as packets) arise between ToR switches
- Opportunitistic links are between lasers and photodetectors

Many *laser-photodetector combinations* can serve traffic between a pair of ToRs

How to optimally transmit packets over reconfigurable links (a matching)?
The Model

Packets between ToRs arrive in an online fashion (adversarial).
Online matching schedule minimizing latency?

<table>
<thead>
<tr>
<th>Round</th>
<th>Packet</th>
<th>Path</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>S1 → D2</td>
<td>(T2,R3)</td>
<td>1</td>
</tr>
<tr>
<td>1st</td>
<td>S2 → D2</td>
<td>(T3,R3)</td>
<td>2</td>
</tr>
<tr>
<td>2nd</td>
<td>S2 → D2</td>
<td>(T3,R3)</td>
<td>2</td>
</tr>
<tr>
<td>2nd</td>
<td>S1 → D2</td>
<td>(T1,R2)</td>
<td>1</td>
</tr>
<tr>
<td>2nd</td>
<td>S1 → D1</td>
<td>(T1,R1)</td>
<td>2</td>
</tr>
</tbody>
</table>

Matchings:
1st round: (T2,R3)
2nd round: (T3,R3), (T1,R2)
3rd round: (T3,R3), (T1,R1)

ALG = 8
OPT = 6
use (T1,R2) and (T3,R3) in 1st round
The Model

Packets between ToRs arrive in an **online fashion** (adversarial).
Online matching schedule minimizing latency?

<table>
<thead>
<tr>
<th>Round</th>
<th>Packet</th>
<th>Path</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>S1 → D2</td>
<td>(T2,R3)</td>
<td>1</td>
</tr>
<tr>
<td>1st</td>
<td>S2 → D2</td>
<td>(T3,R3)</td>
<td>2</td>
</tr>
<tr>
<td>2nd</td>
<td>S2 → D2</td>
<td>(T3,R3)</td>
<td>2</td>
</tr>
<tr>
<td>2nd</td>
<td>S1 → D2</td>
<td>(T1,R2)</td>
<td>1</td>
</tr>
<tr>
<td>2nd</td>
<td>S1 → D1</td>
<td>(T1,R1)</td>
<td>2</td>
</tr>
</tbody>
</table>

Matchings:
1st round: (T2,R3)
2nd round: (T3,R3), (T1,R2)
3rd round: (T3,R3), (T1,R1)

ALG = 8
OPT = 6
use (T1,R2) and (T3,R3) in 1st round

Related to online switch scheduling but 2 tiers!
Algorithm

Scheduler ("transmit stable matchings"):

→ Based on a generalization of the stable-matching algorithm for two-tier networks
→ Each transmitter maintains a queue of packets that are not scheduled yet
→ In each time step, find a stable matching between transmitters and receivers

Dispatcher (greedy):

→ Incoming packet assigned to (transmitter, receiver) pair based on estimated worst case latency increase, taking into account set of queued packets in the system
What is the worst-case impact of S1 → D2 packet of weight 5?

Use (T1,R2): 5 + 5 + 2
Transmitted after BC, before A

Use (T2,R3): 5 + 4
Transmitted after E, before D
Result and Analysis

Alg is $O(\varepsilon^{-2})$-competitive in a resource augmentation model with speedup $(2+\varepsilon)$.

→ Our algorithm is competitive in the speed augmentation model: online algorithm can transmit the packets at twice the rate of the optimal offline algorithm

→ Dinitz and Moseley: otherwise no competitive online algorithm

→ Analysis via dual fitting fitting inspired by scheduling for unrelated machines: take LP and dual-LP, assuming entire input (ok as only for analysis)

→ The crux of the dual-fitting analysis: how to relate the cost of our algorithm to a feasible dual solution
High-Level Overview

→ Find **LP relaxation** (primal) of the resource-augmented problem (OPT has limited transmission speed)

→ Write its **dual** linear program

→ Construct a **solution D** to dual program

→ **Charge** ALG's cost to D

→ Use **weak duality** to relate cost of D and OPT
Primal Program

Variables:
\( X_{p \ e \ T} \) for packet \( p \), compatible edge \( e \), time \( T \)
(fraction of \( p \) sent through \( e \) at time \( T \))

Objective:
\[ \sum_p \sum_e \sum_T \text{weight}(p) \times (T - \text{release}(p)) \times X_{p \ e \ T} \]

Constraints:
\[ \sum_p \sum_e \sum_T X_{p \ e \ T} \geq 1 \]
\[ \sum_e \sum_p X_{p \ e \ T} \leq 1/(2+\varepsilon) \text{ matching for transmitters} \]
\[ \sum_e \sum_p X_{p \ e \ T} \leq 1/(2+\varepsilon) \text{ matching for receivers} \]
Dual Program

Variables:
- $A_p$ for packet $p$
- $B_{tT}, B_{rT}$ for time $T$, transmitter $t$ or receiver $r$

Objective:
$$\sum_p A_p - \frac{1}{2+\epsilon} (\sum_t \sum_T B_{tT} + \sum_t \sum_T B_{tT})$$

Constraints:
$$A_p - B_{tT} - B_{rT} \leq \text{weight}(p) \times (T - \text{release}(p))$$
For packet $p$, compatible edge $e = (t,r)$, time $t$
Dual Assignment

Variables:

\( A_p \) = worst-case impact of \( p \)

\( B_{tT} \) = weight of packets assigned to \( t \), pending at time \( T \)

\( B_{rT} \) = weight of packets assigned to \( t \), pending at time \( T \)

**ALG-to-DUAL** ratio:

\[ \sum p A_p = ALG \]

\[ \sum_t \sum_T B_{tT} = \sum_t \sum_T B_{rT} = ALG \]

\[ \text{DUAL} = \sum p A_p - \frac{1}{2+\varepsilon} \left( \sum_t \sum_T B_{tT} + \sum_t \sum_T B_{rT} \right) \]

\[ = ALG - ALG \times \frac{2}{2+\varepsilon} \]

\[ = ALG \times \frac{\varepsilon}{2+\varepsilon} \]
Dual Assignment

Variables:

\[ A_p = \text{worst-case impact of } p \]
\[ B_{tT} = \text{weight of packets assigned to } t, \text{ pending at time } T \]
\[ B_{rT} = \text{weight of packets assigned to } t, \text{ pending at time } T \]

**ALG-to-DUAL ratio:**

\[ \sum_p A_p = \text{ALG} \]
\[ \sum_t \sum_T B_{tT} = \sum_t \sum_T B_{tT} = \text{ALG} \]

\[ \text{DUAL} = \sum_p A_p - \frac{1}{2+\varepsilon} \left( \sum_t \sum_T B_{tT} + \sum_t \sum_T B_{tT} \right) \]
\[ = \text{ALG} - \text{ALG} \times \frac{2}{2+\varepsilon} \]
\[ = \text{ALG} \times \frac{\varepsilon}{2+\varepsilon} \]

It remains to bound **DUAL-to-OPT** ratio...
Analysis: DUAL-to-OPT

At time $T = \text{release}(p)$ constraint
$$A_p - B_{tT} - B_{rT} \leq \text{weight}(p) \times (T - \text{release}(p))$$
holds with equality.

In one time step
- LHS increases by $2 \times \text{weight}(p)$
- (each $B$ decreases by $\text{weight}(p)$)
- RHS increases by $\text{weight}(p)$

Halving each variable yields a feasible solution to dual program.
We have $\text{ALG} = \frac{2+\varepsilon}{\varepsilon} \times DUAL$

It remains to bound DUAL-to-OPT ratio

Halving each variable yields a feasible solution to dual program.

$\frac{DUAL}{2} \leq \text{OPT}$ by weak duality

We obtain:

$\text{ALG} \leq 2 \times \frac{2+\varepsilon}{\varepsilon} \times \text{OPT}$
Supported Extensions

- Different edge lengths in the network
- Packets sizes
- Hybrid fixed and reconfigurable networks
Focus Topic:
Relationship to Spanners

Demand-Aware Network Designs of Bounded Degree*

Chen Avin¹, Kaushik Mondal¹, and Stefan Schmid²

1 Communication Systems Engineering Department
Ben Gurion University of the Negev, Israel
avin@ee.bgu.ac.il, mondal@post.bgu.ac.il
2 Department of Computer Science
Aalborg University, Denmark
schmids@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 (DAWs) and in particular the design of bounded degree networks. To initiate this research...
Low-Distortion Spanners

Classic problem: find sparse, distance-preserving (low-distortion) spanner (the “DAN”) of a graph (the demand)

But:
- Spanners aim at low distortion among all pairs; in our case, we are only interested in the local distortion, 1-hop communication neighbors
- We allow auxiliary edges (not a subgraph): similar to geometric spanners
- We require constant degree
Still Exploitable!

→ Yet: Can sometimes leverage connection to spanners

**Theorem:** If request distribution $\mathcal{D}$ is regular and uniform, and if we can find a constant distortion, linear sized (i.e., constant sparse) spanner for this request graph: then we can design a constant degree DAN providing an optimal ERL (i.e., $O(H(X|Y) + H(Y|X))$).

- **$r$-regular and uniform demand:**
- **Sparse, irregular (constant) spanner:**
- **Constant degree optimal DAN (ERL at most $\log r$):**
Theorem: If request distribution $\mathcal{D}$ is regular and uniform, and if we can find a constant distortion, linear sized (i.e., constant sparse) spanner for this request graph: then we can design a constant degree DAN providing an optimal ERL (i.e., $O(H(X|Y)+H(Y|X))$).

$r$-regular and uniform demand: Sparse, irregular (constant) spanner: Constant degree optimal DAN (ERL at most $\log r$):

Optimality: $r$-regular graphs have entropy $\log r$. 
Corollaries

Optimal DAN designs for

- Hypercubes (with $n \log n$ edges)
- Chordal graphs
- Trivial: graphs with polynomial degree
  (dense graphs)
- Graphs of locally bounded doubling dimension
Definition: Demand graph has a Locally-bounded Doubling Dimension (LDD) iff all 2-hop neighbors are covered by 1-hop neighbors of just $\lambda$ nodes. Note: care only about 2-neighborhood. Challenge: can be of high degree.
Lemma: There exists a sparse 9-(subgraph)spanner for LDD. This implies optimal DAN: still focus on regular and uniform!

Def. (ε-net): A subset $V'$ of $V$ is a $\varepsilon$-net for a graph $G = (V,E)$ if

$\rightarrow$ $V'$ sufficiently "independent": for every $u, v \in V'$,
$\quad d_G(u, v) > \varepsilon$

$\rightarrow$ "dominating" $V$: for each $w \in V$, $\exists$ at least one
$\quad u \in V'$ such that, $d_G(u, w) \leq \varepsilon$
Simple algorithm:
1. Find a 2-net

Easy: Select nodes into 2-net one-by-one in decreasing (remaining) degrees, remove 2-neighborhood. Iterate.
Example

Simple algorithm:
1. Find a 2-net
2. Assign nodes to one of the closest 2-net nodes: tree

Assign: at most 2 hops.

Union of these shortest paths: a forest. Add to spanner.
Example

Simple algorithm:
1. Find a 2-net
2. Assign nodes to one of the closest 2-net nodes: tree
3. Join two clusters if there are edges in between

Connect forests (single „connecting edge“): add to spanner.
**Example**

**Simple algorithm:**
1. Find a 2-net
2. Assign nodes to one of the closest 2-net nodes: tree
3. Join two clusters if there are edges in between

**Sparse:** Spanner only includes forest (sparse) plus “connecting edges”: but since in a locally doubling dimension graph the number of cluster heads at distance 5 is bounded, only a small number of neighboring clusters will communicate.

**Distortion 9:** *Short detour via*
clusterheads: $u, ch(u), x, y, ch(v), v$