Theory of Demand-Aware Networks

Stefan Schmid (University of Vienna)

IEEE ComSoc Autumn School
on Network Slicing and Data Driven Communication

The Trend: Flexibilities

Flexibilities: Along 3 Dimensions

Somewhere in beautiful Germany…

Flexibilities: Along 3 Dimensions

Somewhere in beautiful Germany…

Flexibilities: Along 3 Dimensions

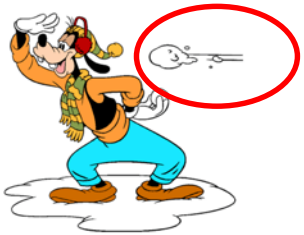
ful Germany…

Flexibilities Enable Demand-Aware Networks

But when are they useful?

A Simple Answer

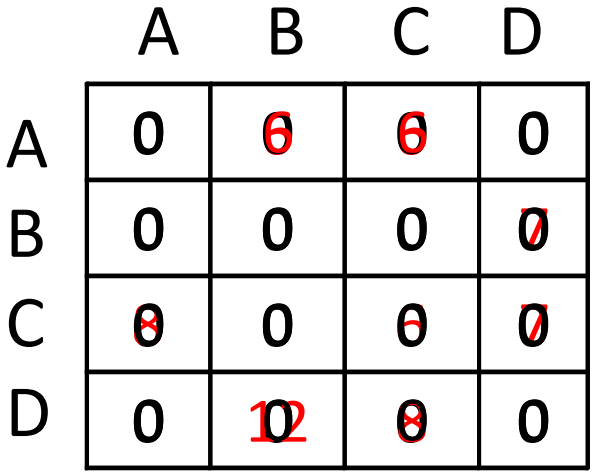
Demand-Oblivious Networks =



Seriously: We believe, often, in practice!

	A	B	C	D
A	0	3	3	3
B	3	0	3	3
C	3	3	0	3
D	3	3	3	0

In theory: traffic matrix uniform and static

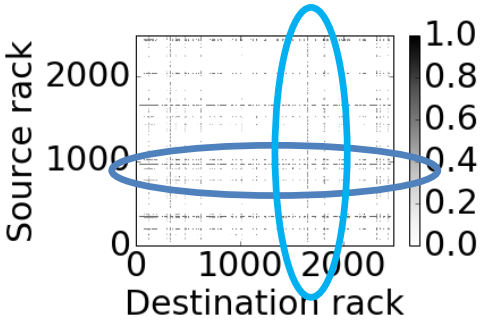


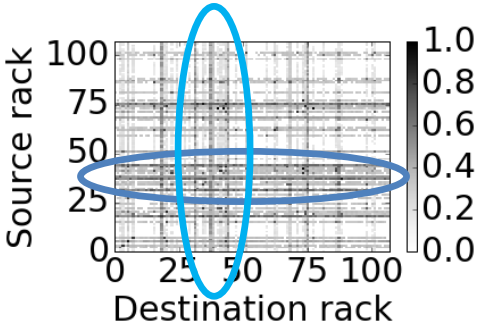
In practice: **skewed** and **dynamic**

Empirical Motivation

Observation 1:

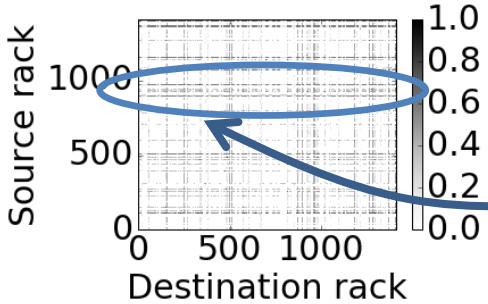
- Many rack pairs exchange ***little traffic***
- Only some ***hot rack pairs*** are active

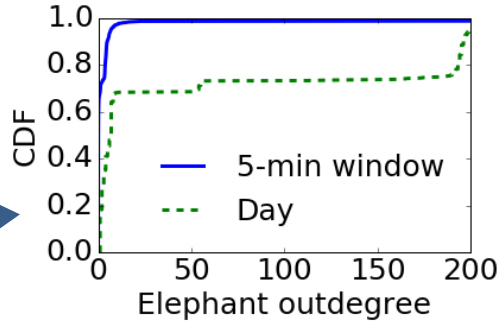




Observation 2:

- Some source racks send large amounts of traffic ***to many other racks***





Microsoft data: 200K servers across 4 production clusters, cluster sizes: 100 - 2500 racks.
Mix of workloads: MapReduce-type jobs, index builders, database and storage systems.

So: How *much* structure is there?

How to ***measure*** it?
And which ***types of structures***? E.g., **temporal** structure in addition to **non-temporal** structure?
Tricky!

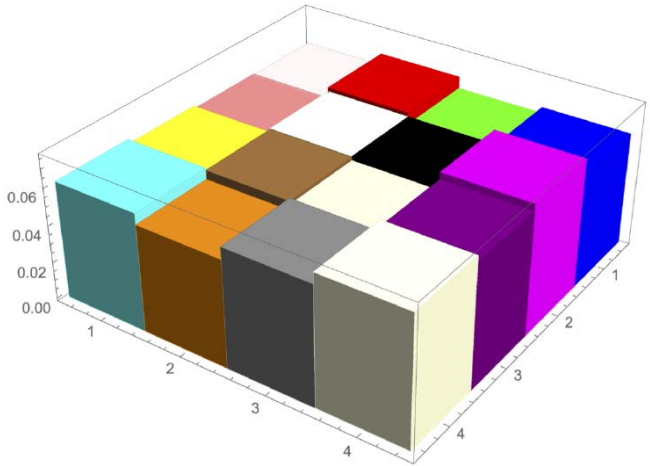
Often only intuitions in the literature…

"less than 1% of the rack pairs account for 80% of the total traffic"

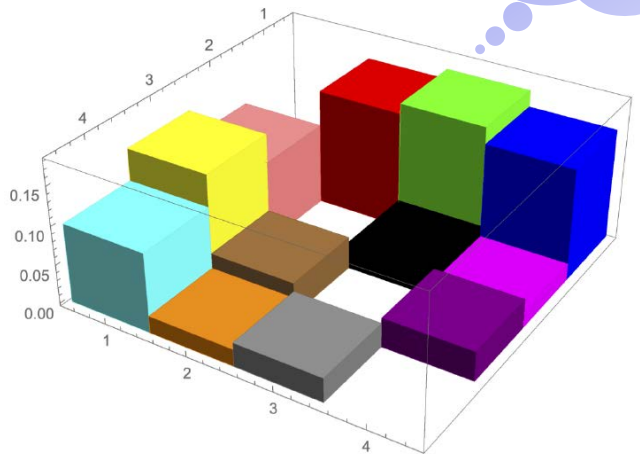
"only a few ToRs switches are hot and most of their traffic goes to a few other ToRs"

"over 90% bytes flow in elephant flows"

… and it *is* intuitive!
Non-temporal Structure



vs



Color = comm. pair

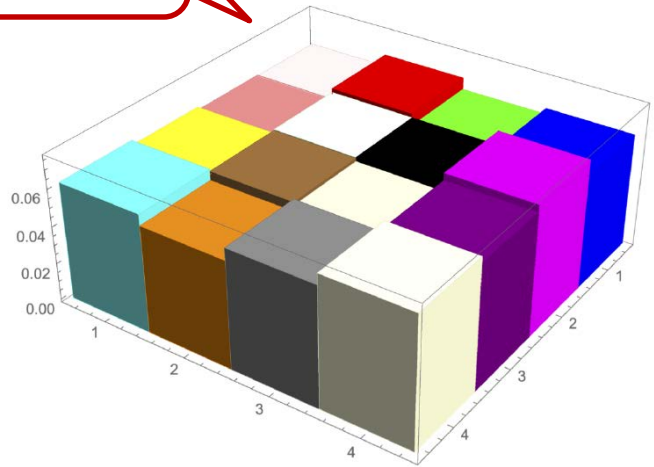
Traffic matrix of two different **distributed ML** applications (GPU-to-GPU):

Which one has *more structure*?

… and it *is* intuitive!

Non-temporal Structure

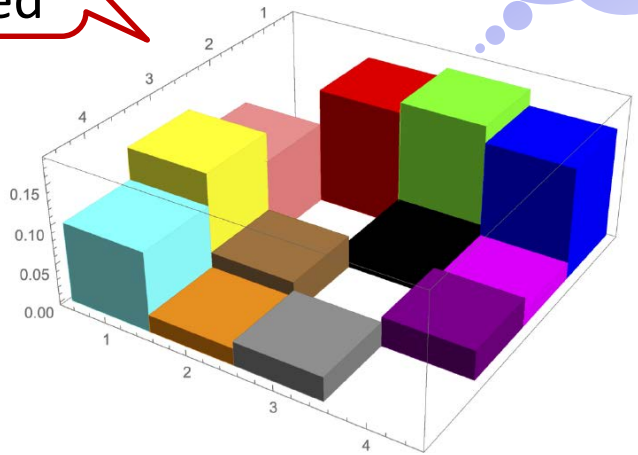
More uniform



vs

More skewed

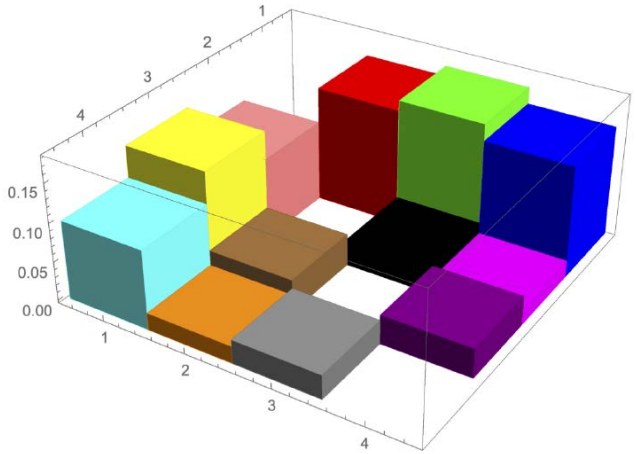
Color = comm. pair

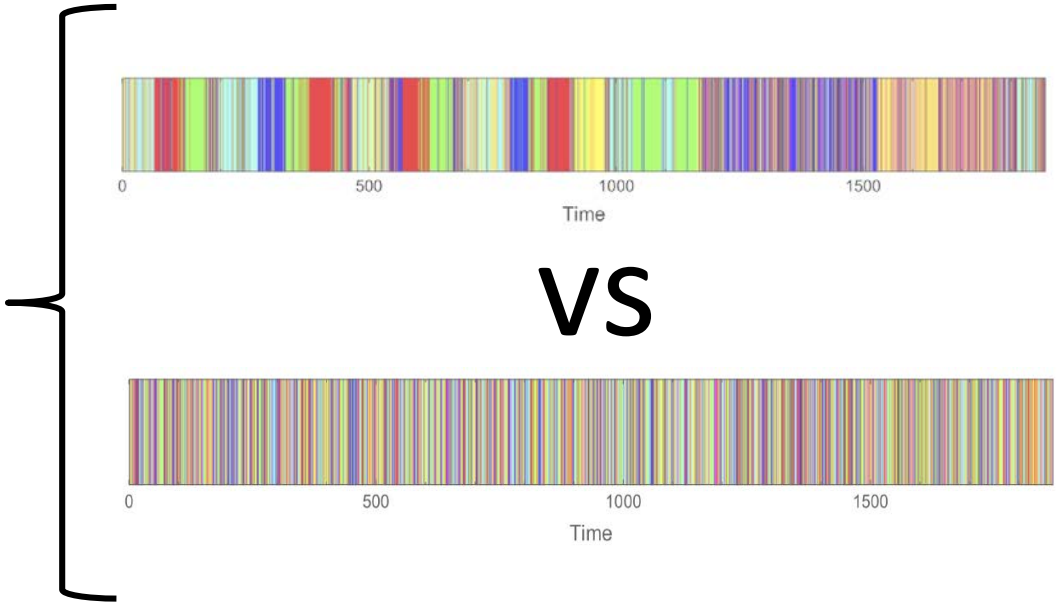


Traffic matrix of two different **distributed ML** applications (GPU-to-GPU):

Which one has *more structure*?

… and it *is* intuitive!
Temporal Structure

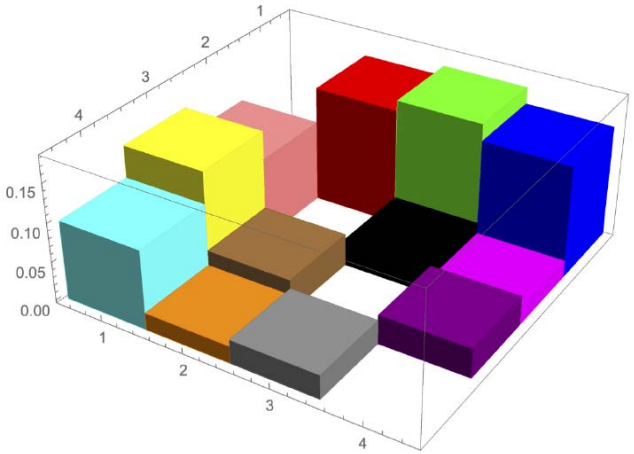




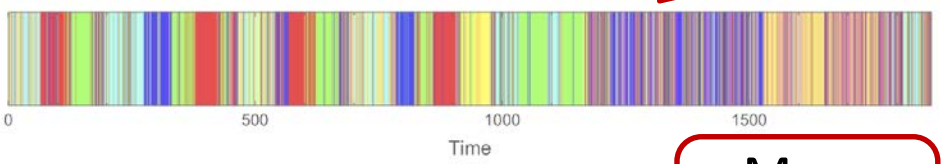
Two different ways to generate *same traffic matrix* (same non-temporal structure):

Which one has *more structure*?

… and it *is* intuitive!
Temporal Structure

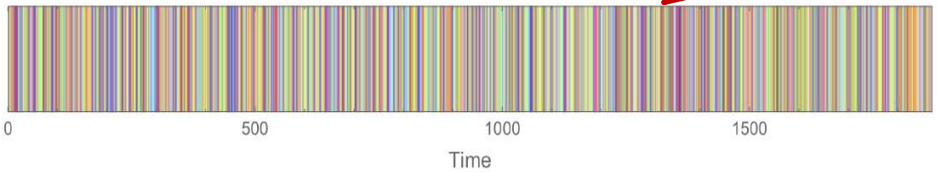


More bursty



VS

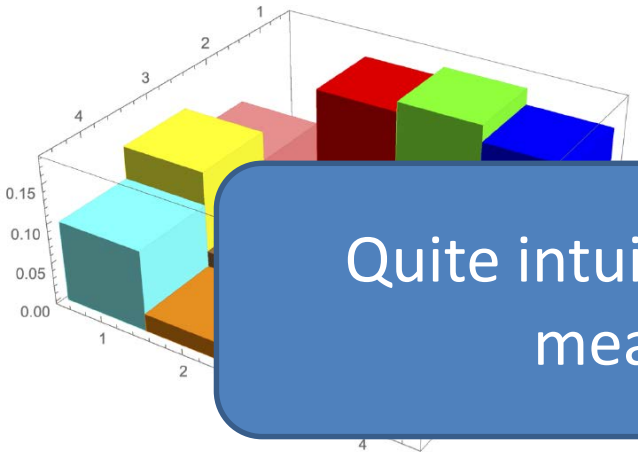
More random



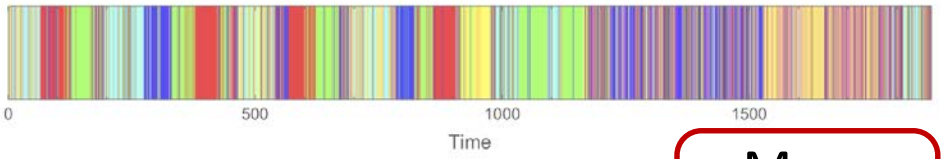
Two different ways to generate *same traffic matrix* (same non-temporal structure):

Which one has *more structure*?

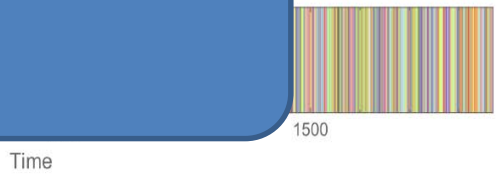
… and it *is* intuitive!
Temporal Structure



More bursty



More random



Quite intuitive: but how to define and measure systematically?

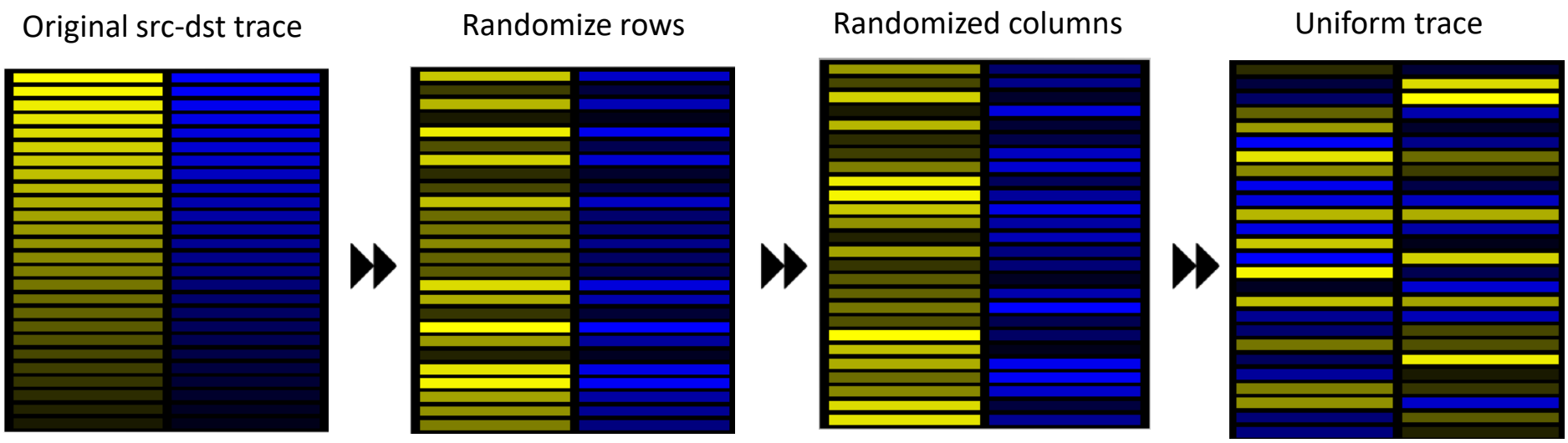
Two different ways to generate *same traffic matrix* (same non-temporal structure):

Which one has *more structure*?

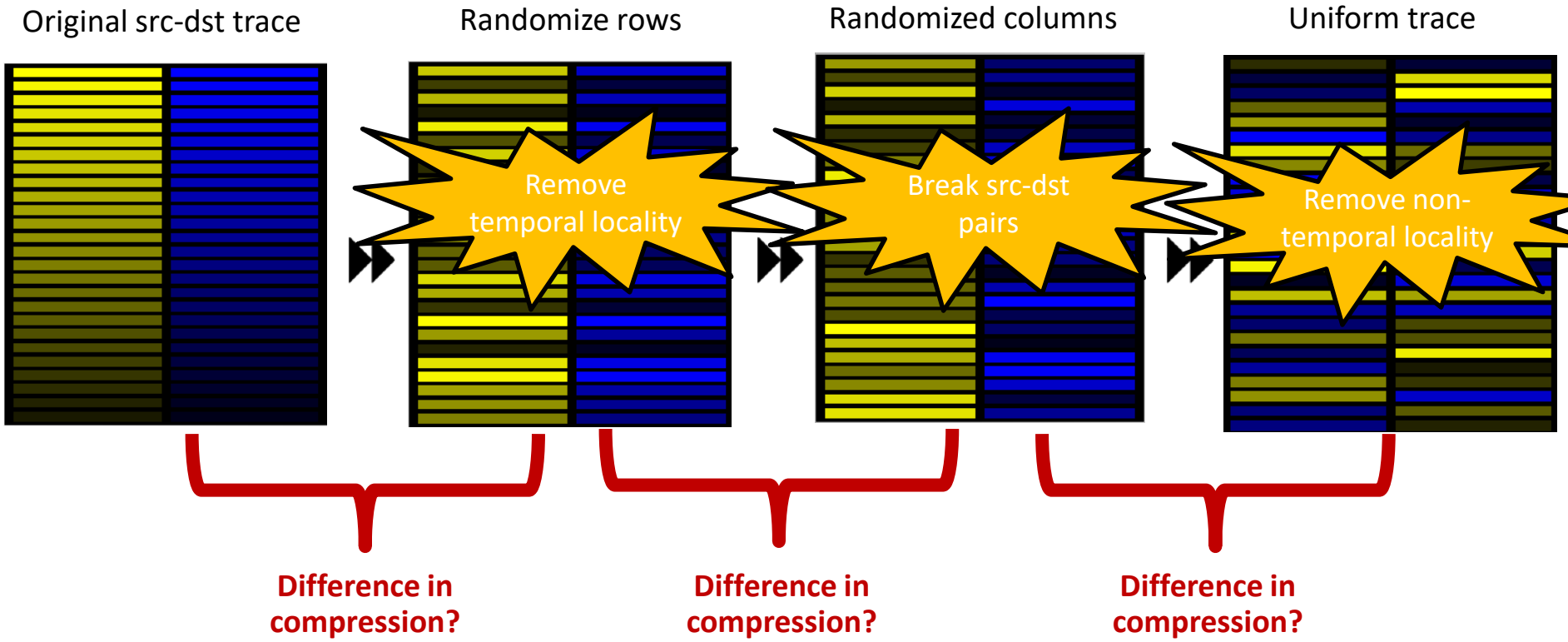
The Trace Complexity

- An **information-theoretic** approach: how can we ***measure the entropy*** (rate) of a traffic trace?
- Henceforth called the **trace complexity**
- Simple approximation: „**shuffle&compress**“
  - Remove structure by iterative *randomization*
  - Difference of compression *before and after* randomization: structure

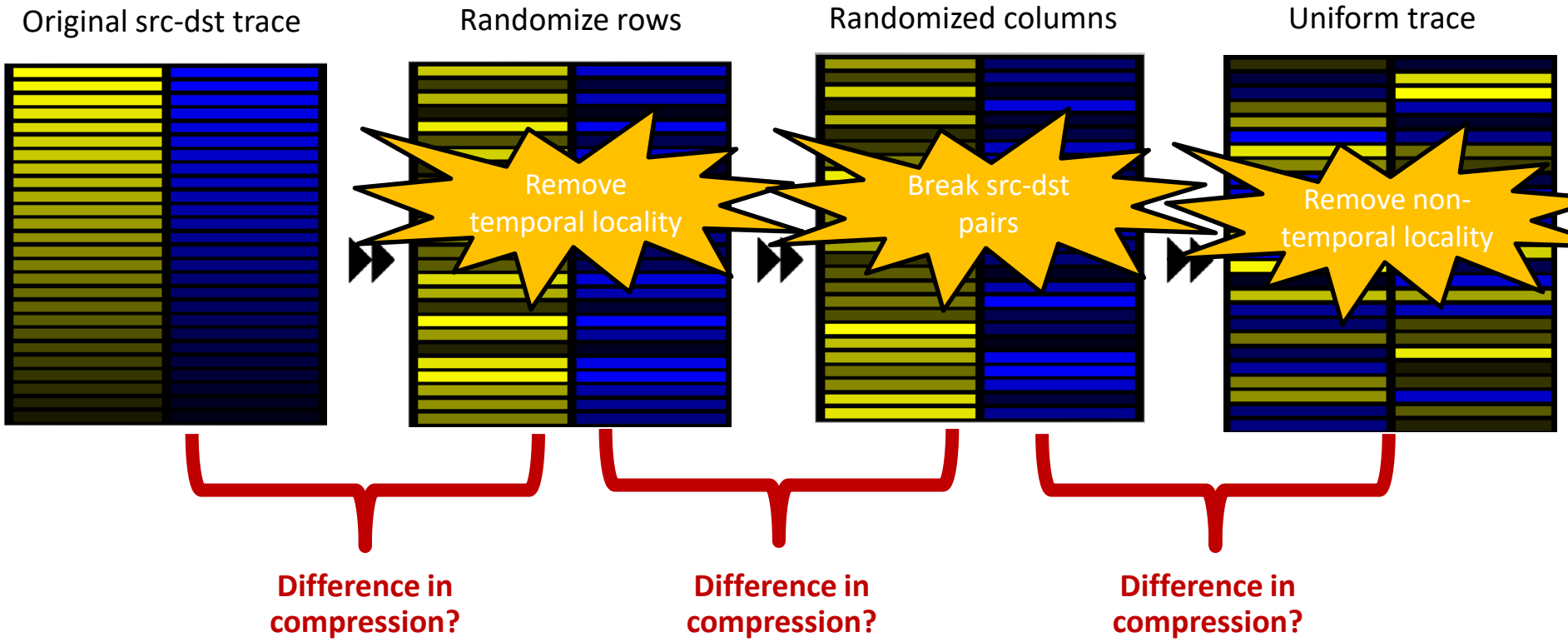
The Trace Complexity



The Trace Complexity

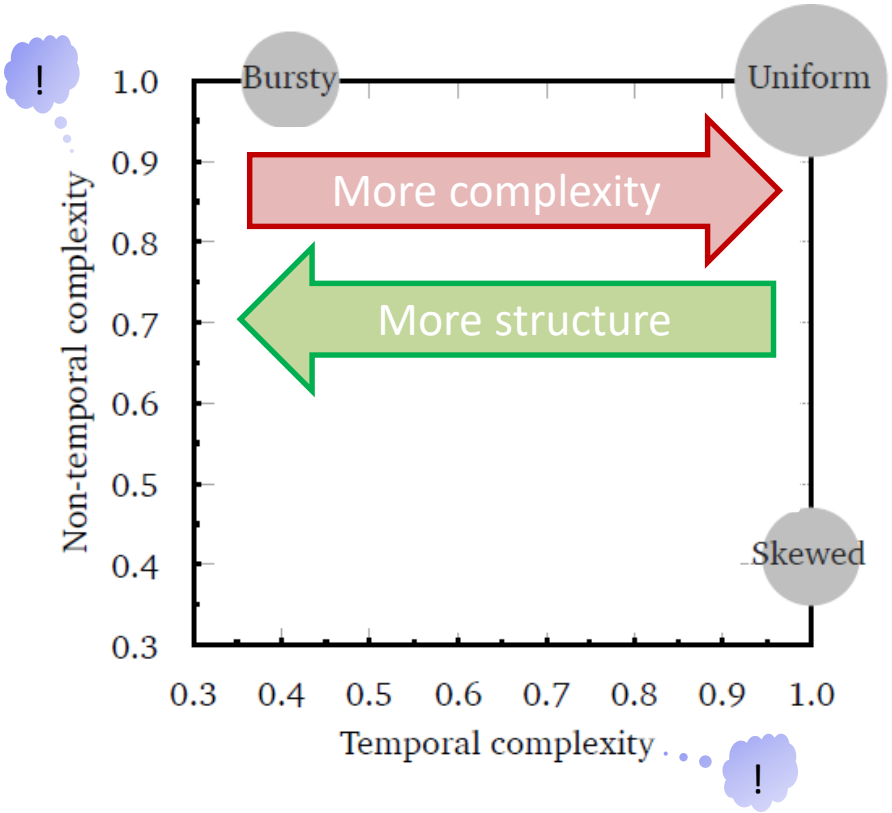


The Trace Complexity



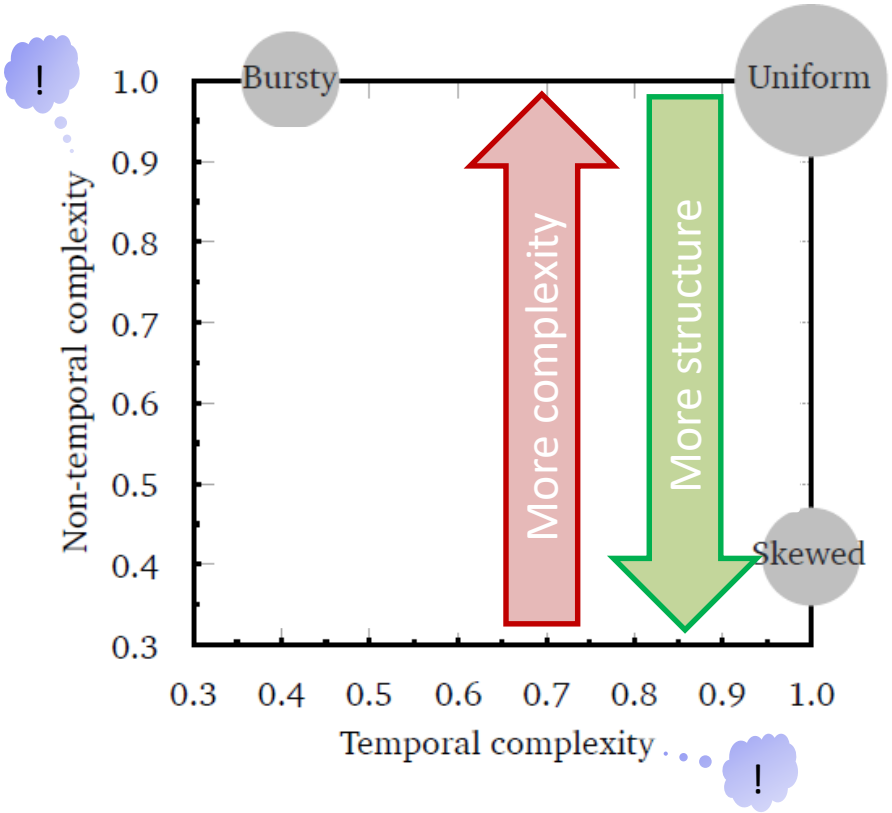
Can be used to define a „complexity map“!

The Complexity Map



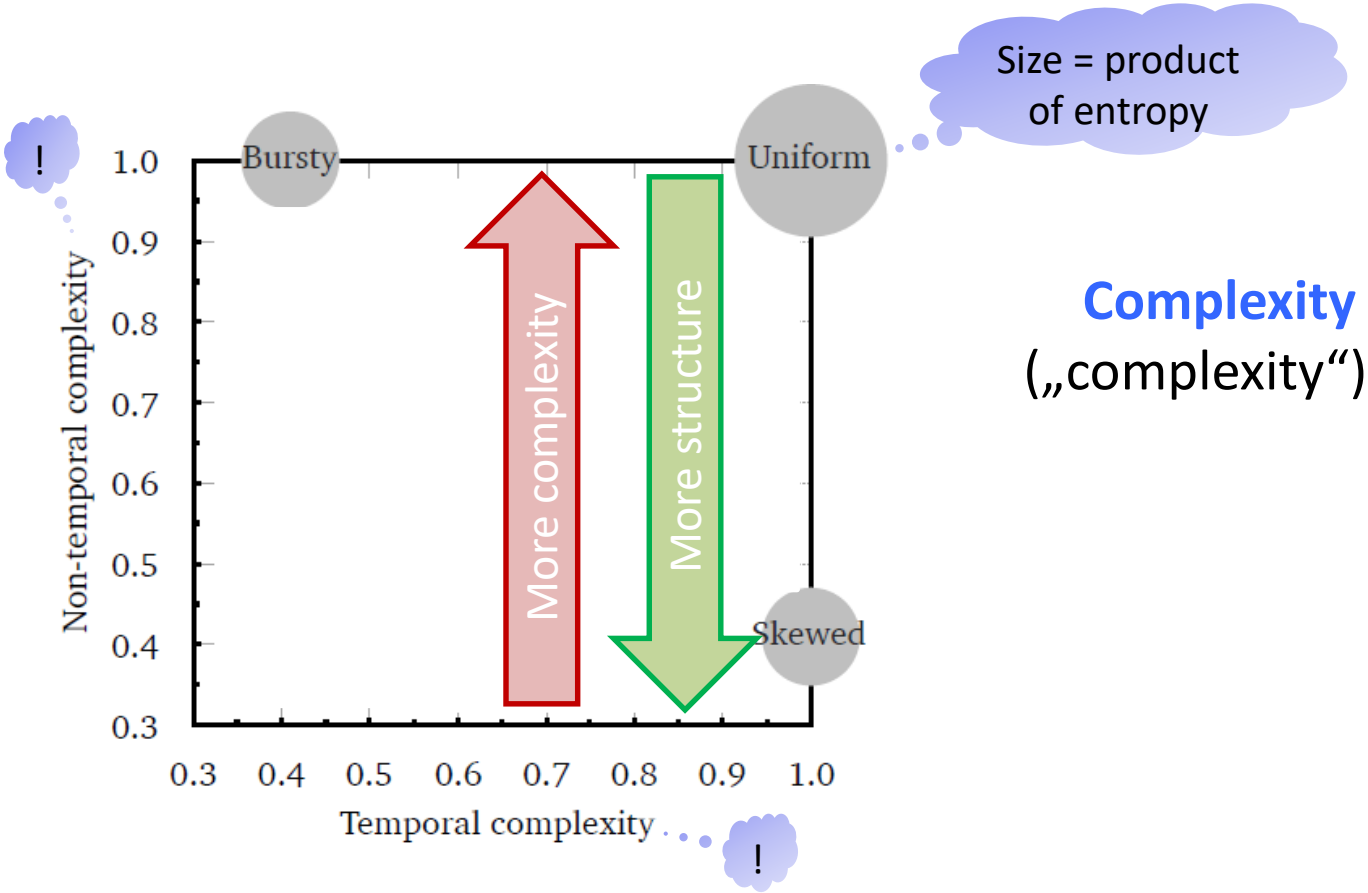
Complexity Map: Entropy („complexity“) of traffic traces.

The Complexity Map



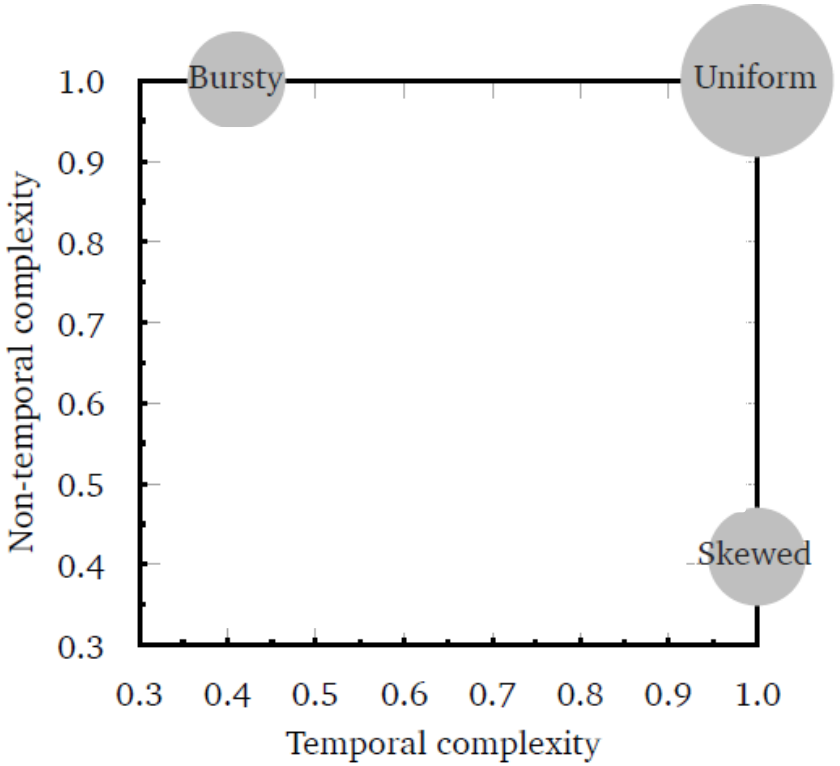
Complexity Map: Entropy („complexity“) of traffic traces.

The Complexity Map



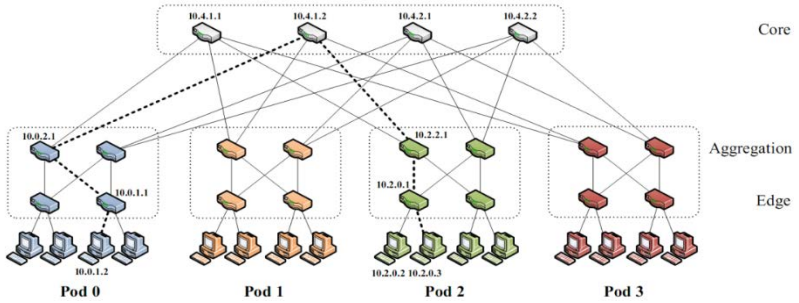
Complexity Map: Entropy („complexity“) of traffic traces.

The Complexity Map

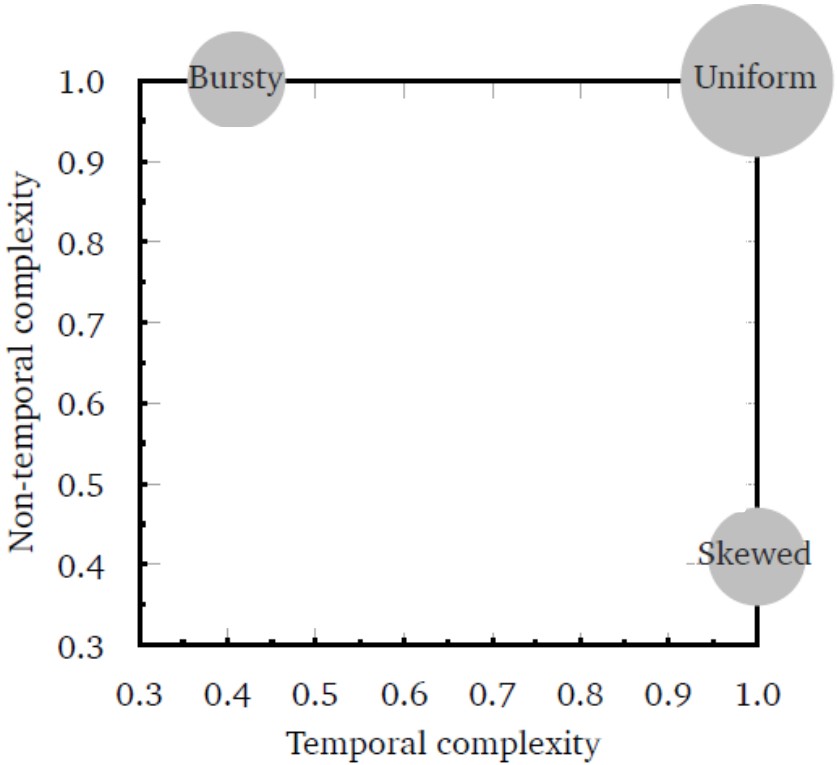


Uniform: Today's datacenters

- Traditional networks are optimized ***for the "worst-case"*** (**all-to-all** communication traffic)
- Example, fat-tree topologies: provide **full bisection bandwidth**

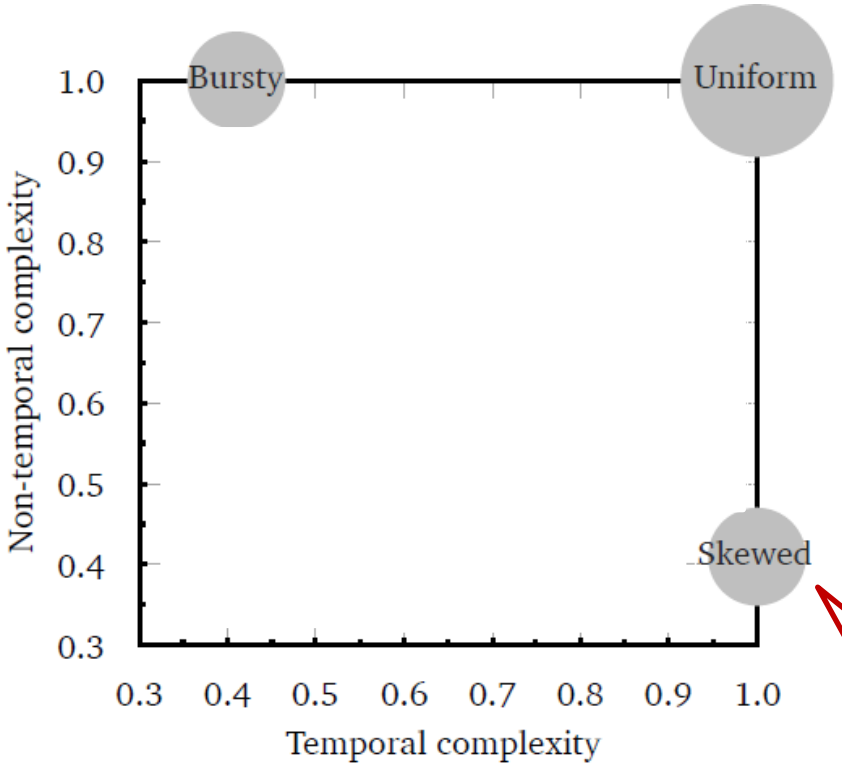


The Complexity Map



Good in the worst case ***but*****:**
cannot leverage different
temporal and **non-temporal**
structures of traffic traces!

The Complexity Map

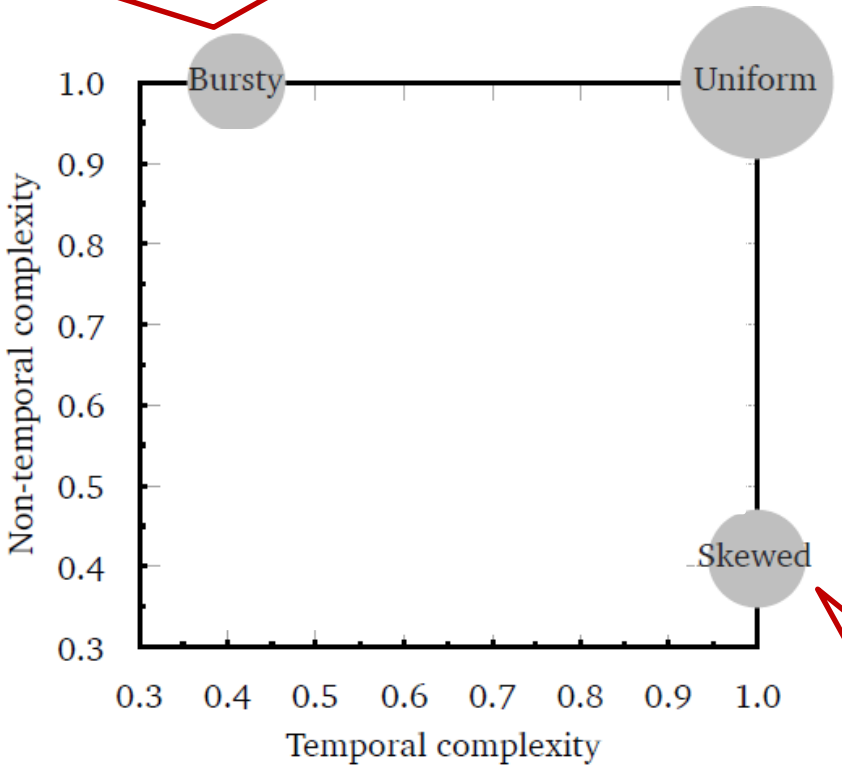


Good in the worst case ***but*****:** cannot leverage different **temporal** and **non-temporal** structures of traffic traces!

Non-temporal structure could be exploited already with ***static demand-aware networks***!

To exploit **temporal** structure, need ***adaptive demand-aware ("self-adjusting") networks***.

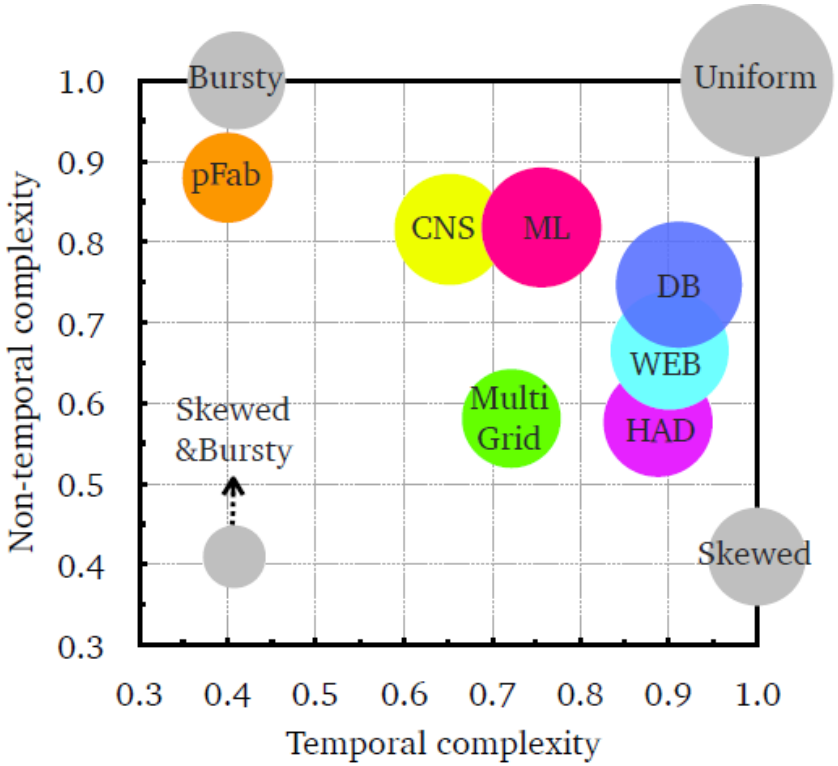
plexity Map



Good in the worst case ***but*****:** cannot leverage different **temporal** and **non-temporal** structures of traffic traces!

Non-temporal structure could be exploited already with ***static demand-aware networks***!

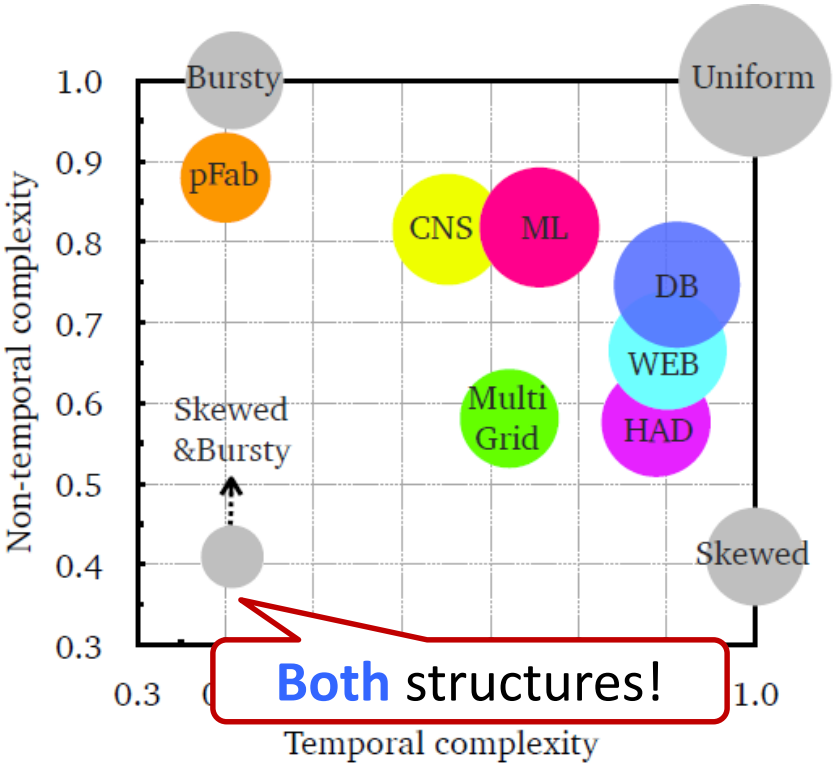
The Complexity Map



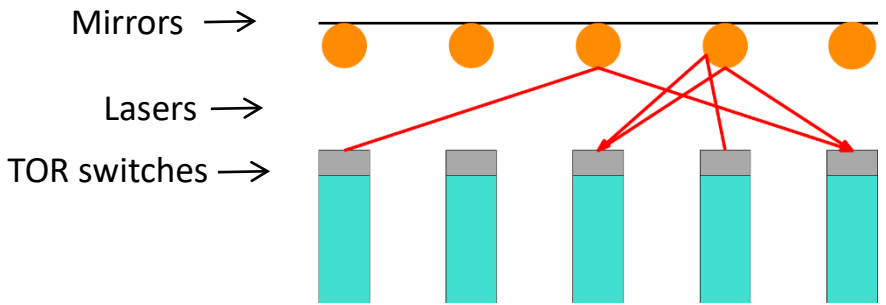
Observation: different applications feature quite significant (and different!) **temporal** and **non-temporal** structures.

- **Facebook** clusters: DB, WEB, HAD
- **HPC** workloads: CNS, Multigrid
- Distributed **Machine Learning** (ML)
- Synthetic traces like **pFabric**

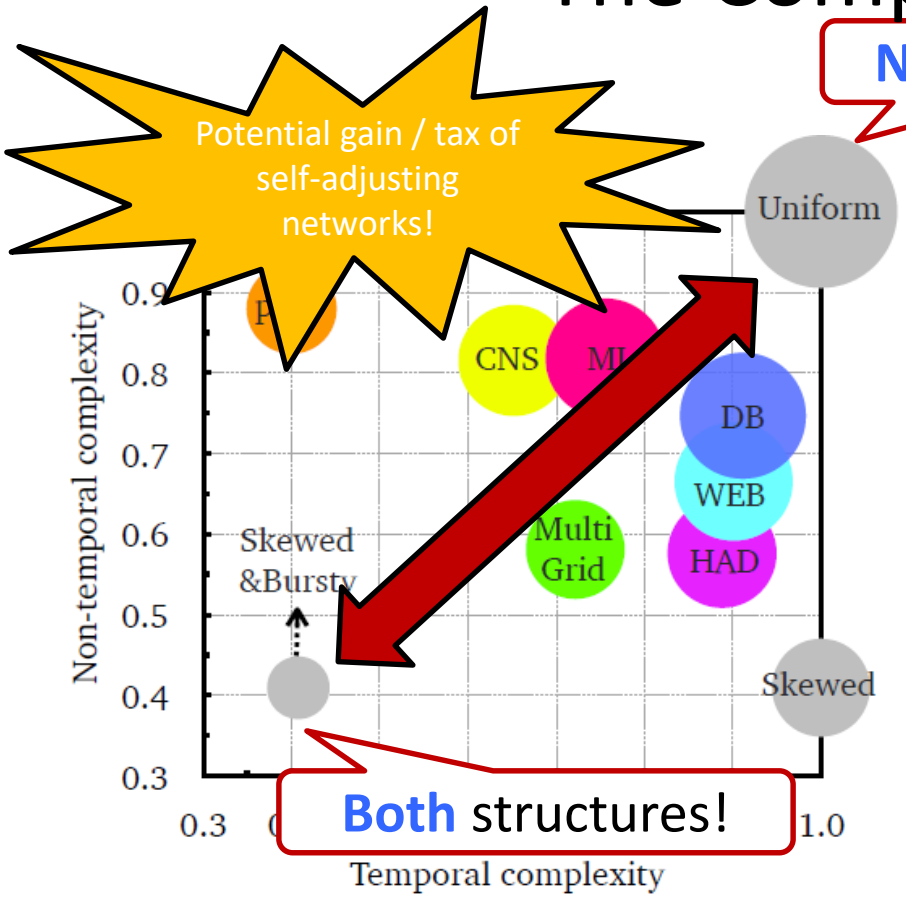
The Complexity Map



Goal: Design **self-adjusting networks** which leverage ***both*** dimensions of structure!



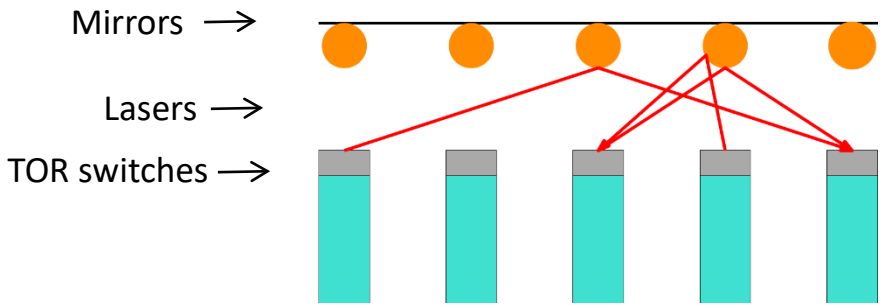
The Complexity Map



No structure!

Goal: Design **self-adjusting networks** which leverage ***both*** dimensions of structure!

Both structures!



Further Reading

Measuring the Complexity of Packet Traces. Avin, Ghobadi, Griner, Schmid. **ArXiv** 2019.

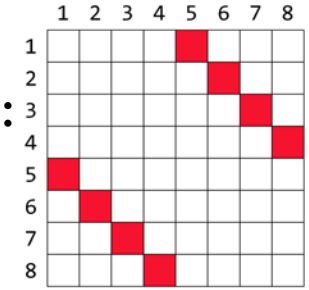
So: How to design networks which exploit this structure? How good can they be?

Metrics again!

A Simple Example: Network Design

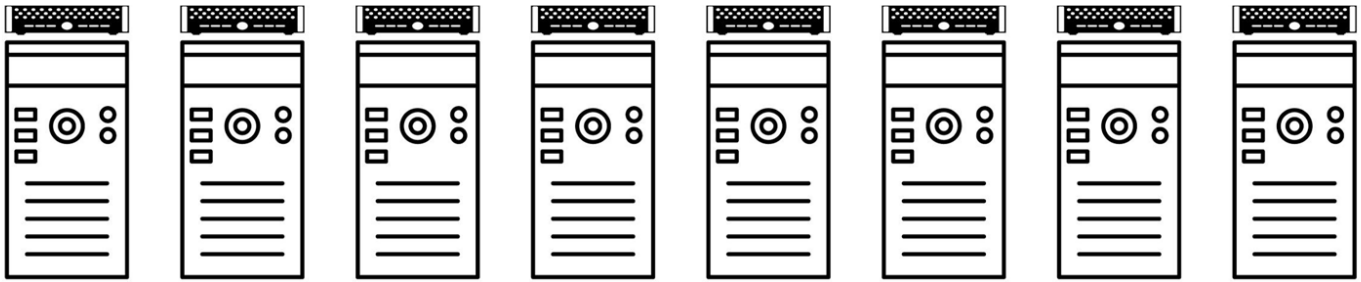
Demand-Aware Network Designs of Bounded Degree
Chen Avin, Kaushik Mondal, and Stefan Schmid.
31st International Symposium on Distributed Computing (**DISC**), Vienna, Austria, October 2017.

demand matrix:



e.g., mirrors

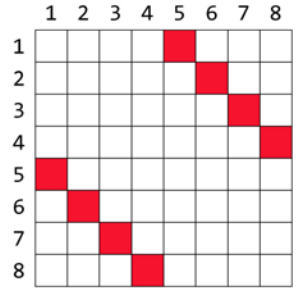
new flexible interconnect

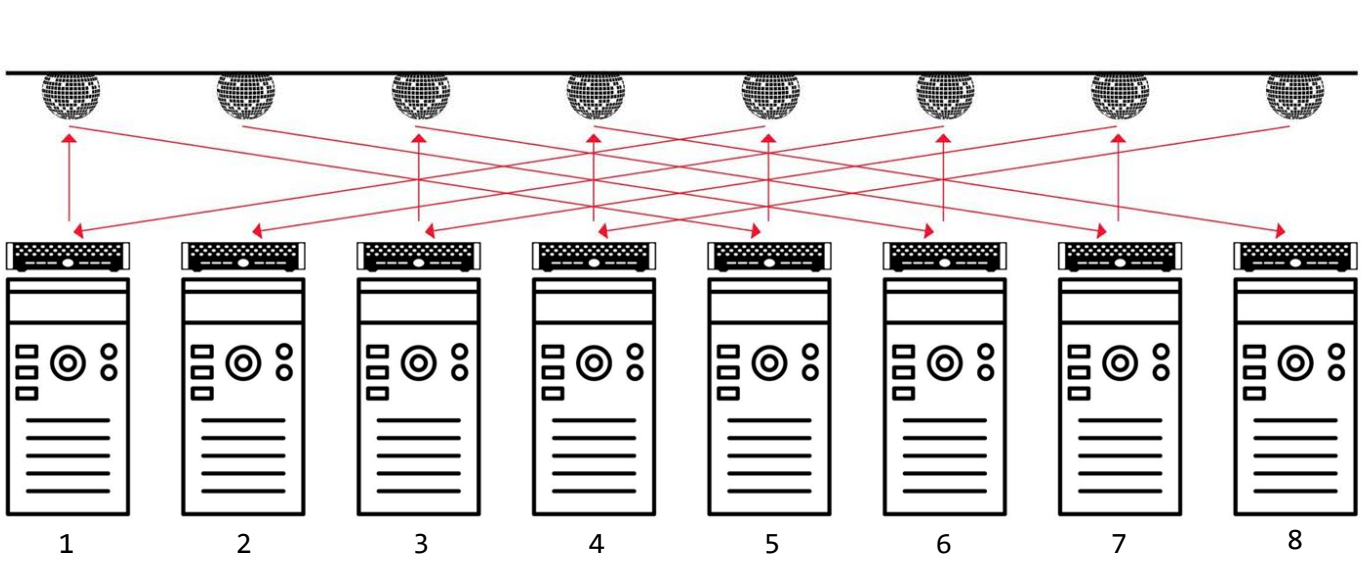


1 2 3 4 5 6 7 8

Matches demand

demand matrix:

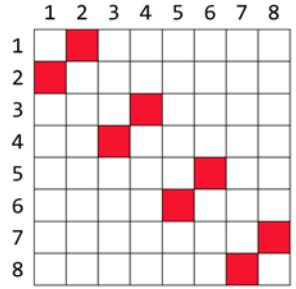




e.g., mirrors

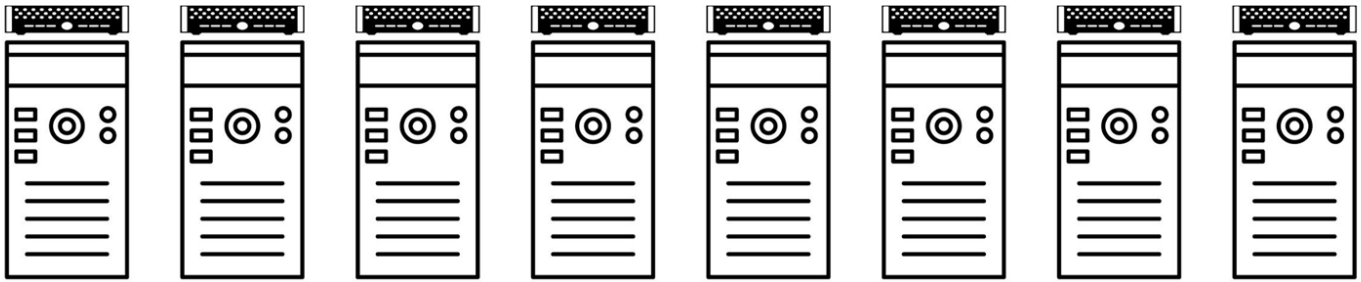
new flexible interconnect

new demand:



e.g., mirrors

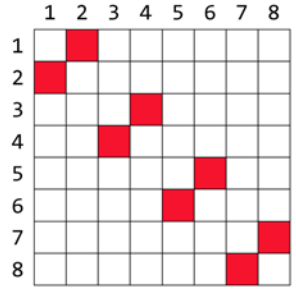
new flexible interconnect

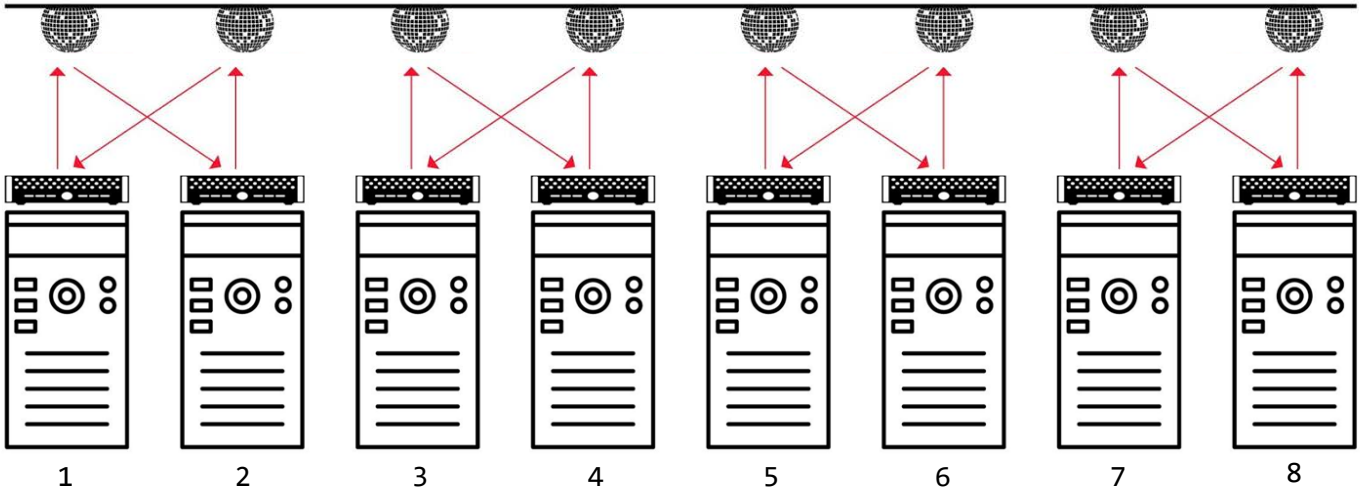


1 2 3 4 5 6 7 8

Matches demand

new demand:





e.g., mirrors

new flexible interconnect

More Formally

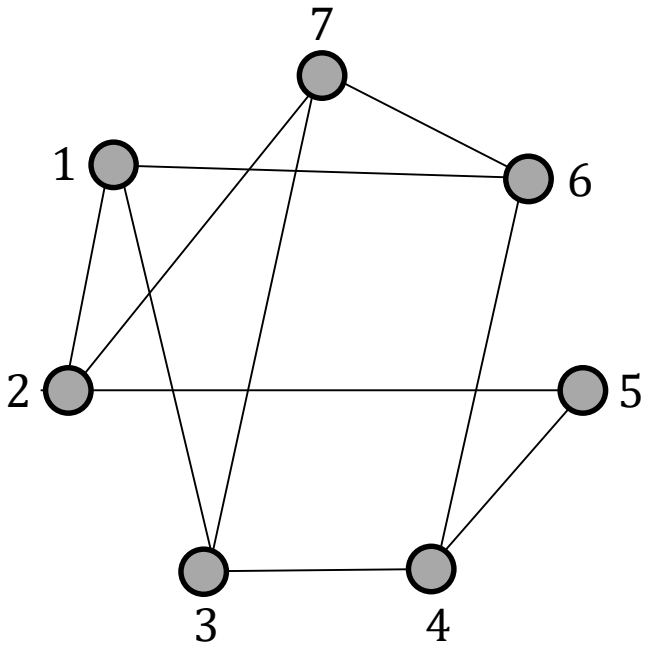
Input: Workload

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

D

Output: Constant-Degree DAN



N

Input: Workload

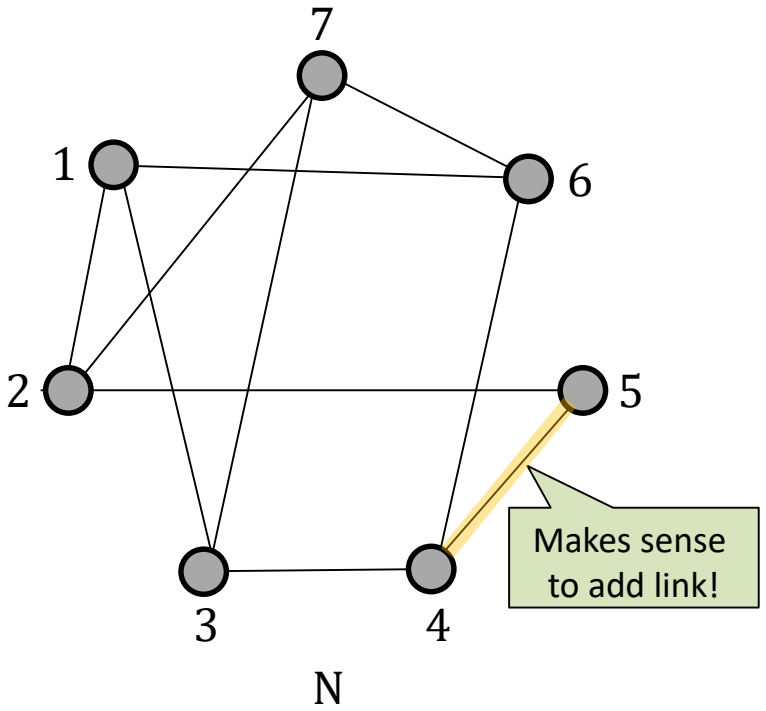
Destinations

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	$\frac{2}{65}$			
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

Much from 4 to 5.

D

Output: Constant-Degree DAN



Input: Workload

1 communicates to many.

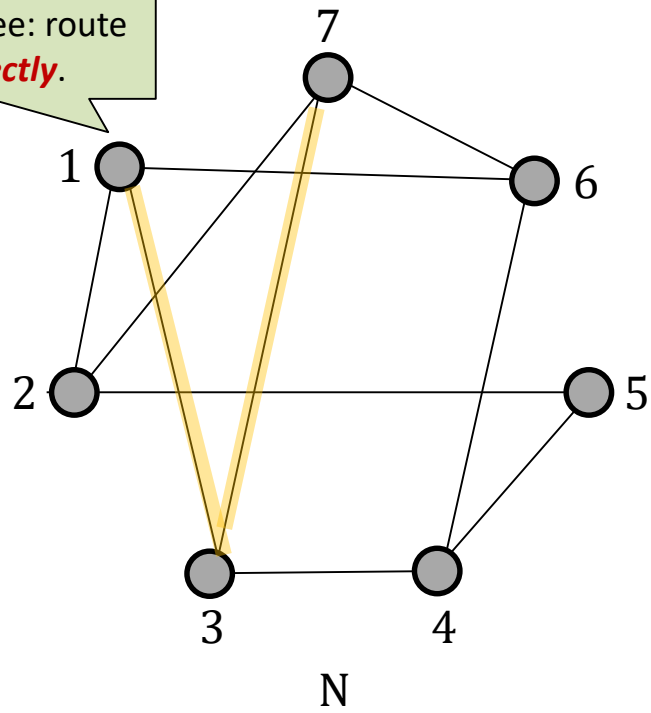
Destinations

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

D

Output: Constant-Degree DAN

Bounded degree: route to 7 ***indirectly***.



N

Input: Workload

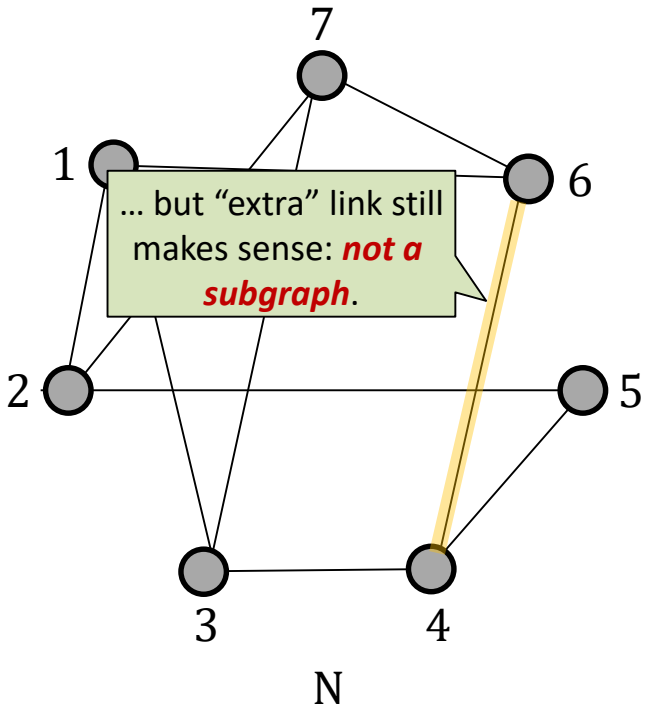
Destinations

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

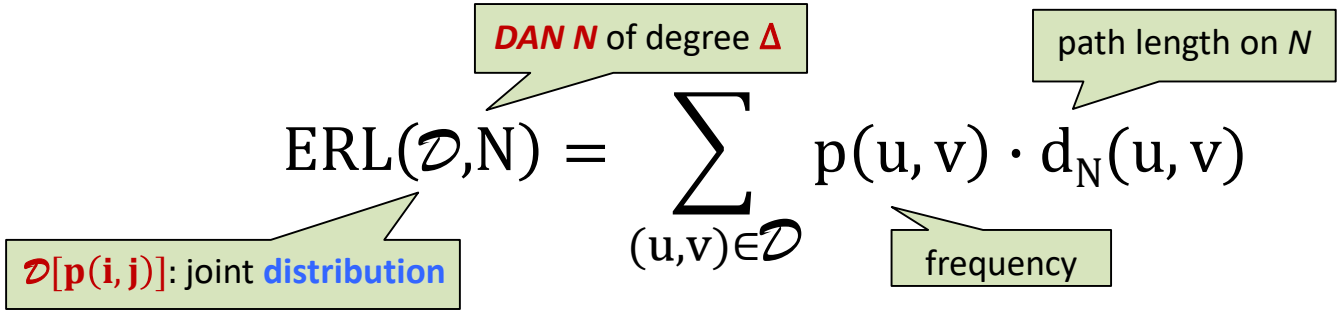
D

4 and 6 don't communicate…

Output: Constant-Degree DAN



Objective: Expected Route Length

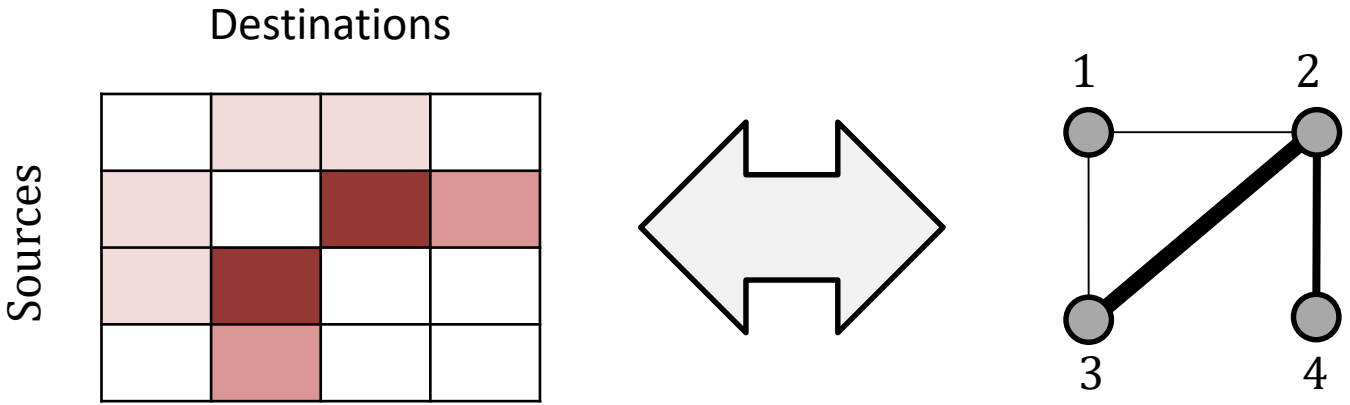


Remark

- Can represent demand matrix as a **demand graph**

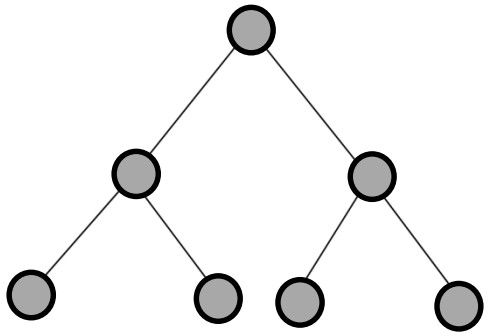
sparse distribution $\mathcal{D}$

sparse graph G($\mathcal{D}$)

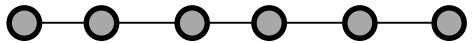


Some Examples

- DANs of $\Delta = 3$:
  - E.g., complete binary **tree**
  - $d_N(u,v) \leq 2 \log n$
  - Can we do ***better*** than ***log n***?



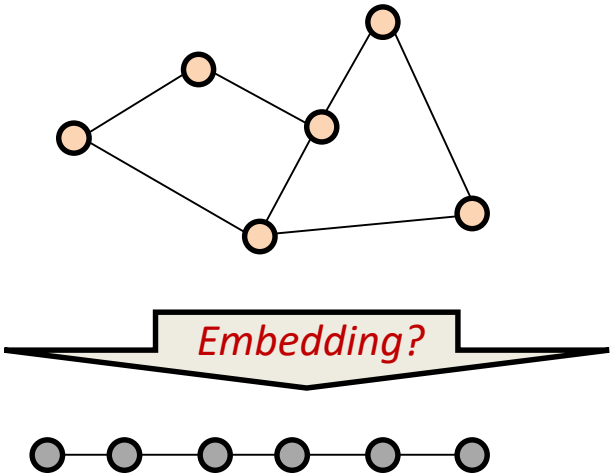
- DANs of $\Delta = 2$:
  - E.g., set of **lines** and **cycles**



Remark: Hardness Proof

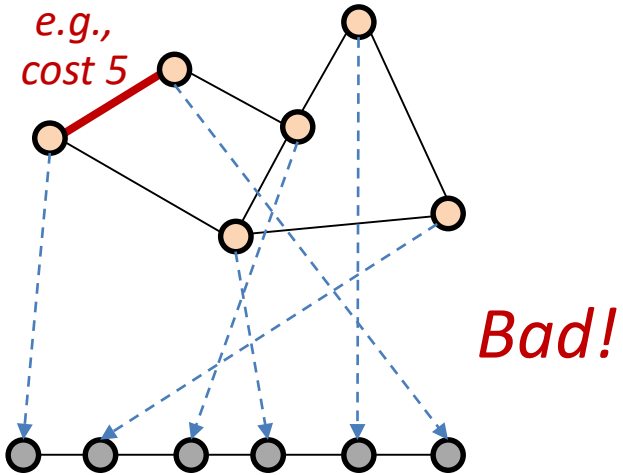
DAN design can be NP-hard

- **Example Δ = 2:** A Minimum Linear Arrangement (**MLA**) problem
  - A "Virtual Network Embedding Problem", VNEP
  - ***Minimize sum*** of lengths of virtual edges



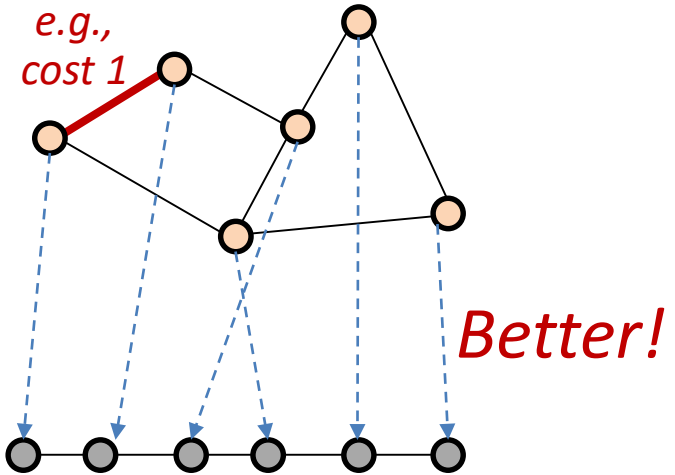
DAN design can be NP-hard

- **Example Δ = 2:** A Minimum Linear Arrangement (**MLA**) problem
  - A "Virtual Network Embedding Problem", VNEP
  - ***Minimize sum*** of lengths of virtual edges



DAN design can be NP-hard

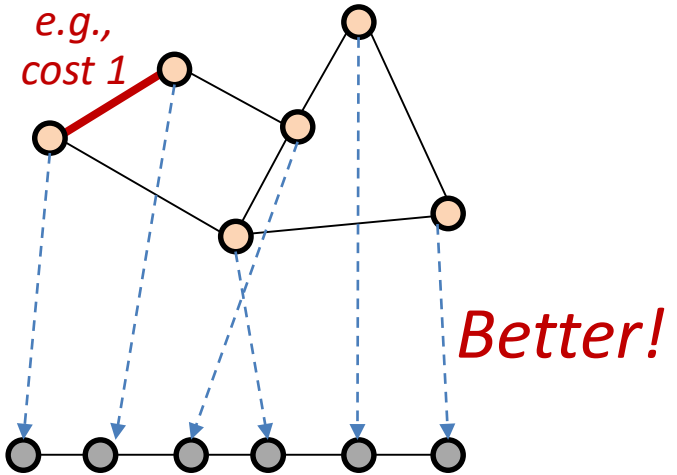
- **Example Δ = 2:** A Minimum Linear Arrangement (**MLA**) problem
  - A "Virtual Network Embedding Problem", VNEP
  - ***Minimize sum*** of lengths of virtual edges



DAN design can be NP-hard

- **Example Δ = 2:** A Mini… inear Arrangement (M… blem
  - A "Virtual N… bedding Problem", VNEP
  - *Minim…* engths of virtual edges

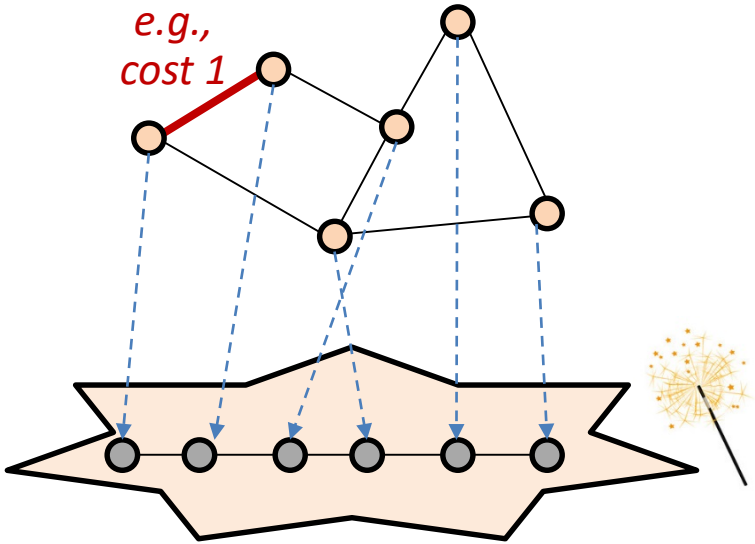
NP-hard, and so is DAN design.



DAN design can be NP-hard

- **Example Δ = 2:** A Mini… inear Arrangement (M… blem
  - A "Virtual N… bedding Problem", VNEP
  - *Minim…* engths of virtual edges

NP-hard, and so is DAN design.



- But what about > 2? ***Embedding*** problem still hard, but we have an additional **degree of freedom**:

Do topological flexibilities make problem easier or harder?!

A new knob for optimization!

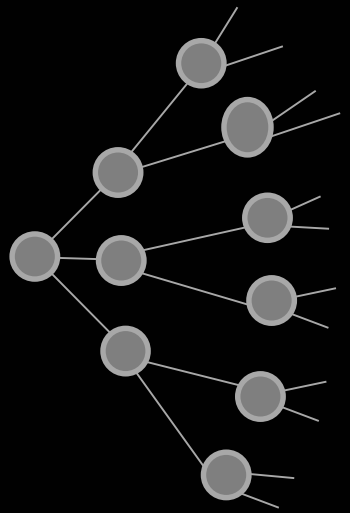
Rewinding the Clock: Degree-Diameter Tradeoff

Each network with n nodes and max degree $\Delta > 2$ must have a diameter of at least $\log(n)/\log(\Delta-1)-1$.

Example: constant Δ, $\log(n)$ diameter

Kudos to: Pedro Casas

Proof Idea



In k steps, reach at most $1 + \sum \Delta(\Delta-1)^k$ nodes

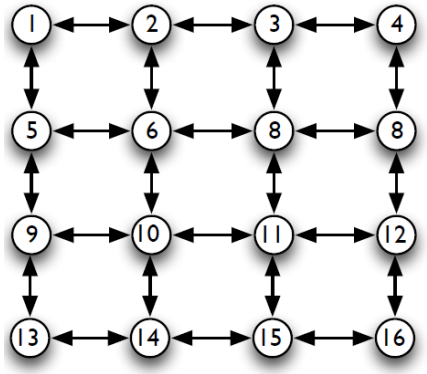
$1 \quad \Delta \quad \Delta(\Delta-1) \ldots$

Kudos to: Pedro Casas

Is there a better tradeoff in DANs?

Sometimes, DANs can be much better!

Example 1: low-degree demand

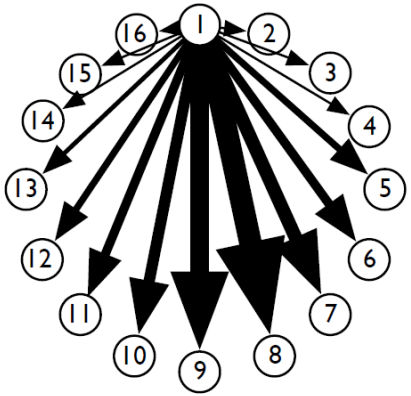


If **demand graph** is of degree Δ, it is trivial to design a **DAN** of degree Δ which achieves an *expected route length of 1*.

Just take DAN = demand graph!

Sometimes, DANs can be much better!

Example 2: skewed demand

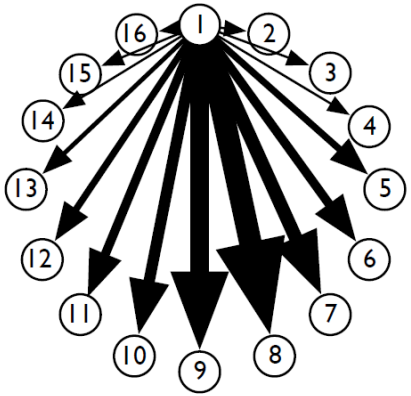


If **demand** is highly skewed, it is also possible to achieve an ***expected route length of O(1)*** in a constant-degree DAN.

?

Sometimes, DANs can be much better!

Example 2: skewed demand



If **demand** is highly skewed, it is also possible to achieve an ***expected route length of O(1)*** in a constant-degree DAN.

E.g., arrange neighbors of node 1 in a **Huffman** tree!

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

So on what does it depend?

So on what does it depend?

We argue (but still don‘t know!): on the **“entropy” of the demand**!

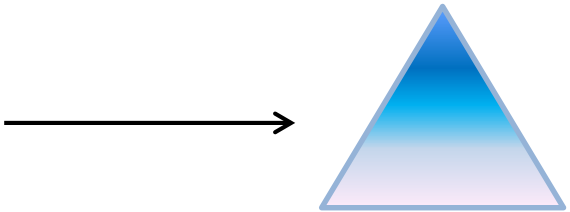
Intuition: Entropy Lower Bound

Lower Bound Idea: Leverage Coding or Datastructure

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

- DAN just for a ***single (source) node 3***



- How good can this tree be? Cannot do better than Δ-ary **Huffman tree** for its destinations
- **Entropy** lower bound on ERL known for binary trees, e.g. ***Mehlhorn*** 1975

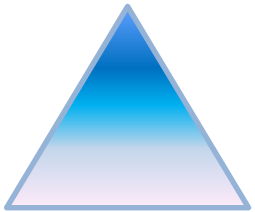
Lower Bound Idea: Leverage Coding or Datastructure

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

An optimal "**ego-tree**" for this source!

- DAN just for a ***single (source) node 3***



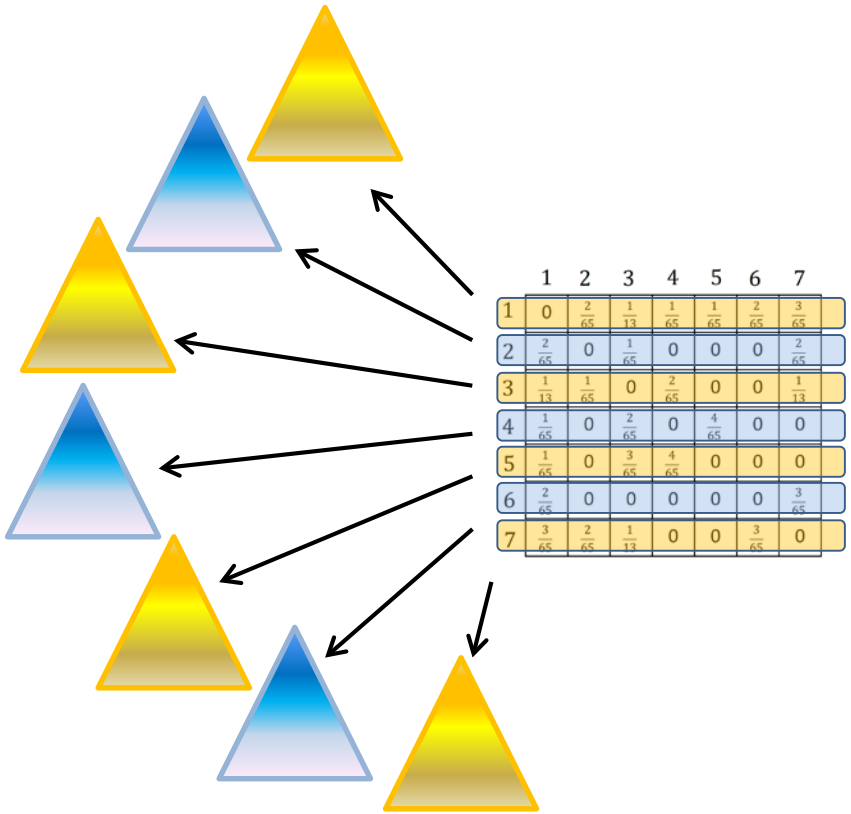
- How good can this tree be? Cannot do better than Δ-ary **Huffman tree** for its destinations
- **Entropy** lower bound on ERL known for binary trees, e.g. ***Mehlhorn*** 1975

So: Entropy of the *Entire* Demand

- **Proof idea** ($EPL=\Omega(H_{\Delta}(Y|X))$):

sources, destinations, entropy, degree

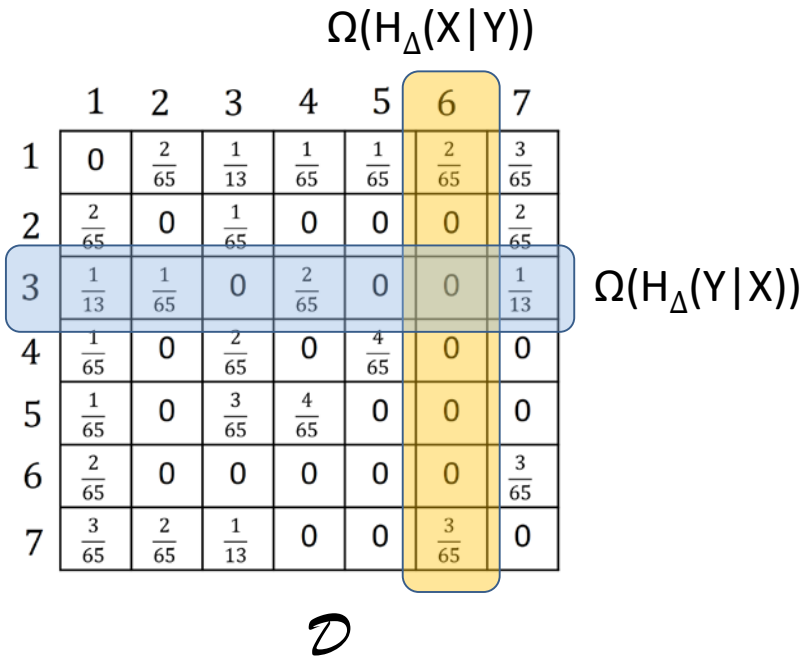
- Compute **ego-tree** for each source node
- Take ***union*** of all **ego-trees**
- Violates ***degree restriction*** but valid lower bound



Entropy of the *Entire* Demand: Sources *and* Destinations

Do this in **both dimensions**:

$EPL \geq \Omega(\max\{H_\Delta(Y|X), H_\Delta(X|Y)\})$



	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 of the *Entire* Demand: Sources *and* Destinations

Do this in **both dimensions**:

$EPL \geq \Omega(\max\{H_\Delta(Y|X), H_\Delta(X|Y)\})$

$\Omega(H_\Delta(X|Y))$

	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

$\Omega(H_\Delta(Y|X))$

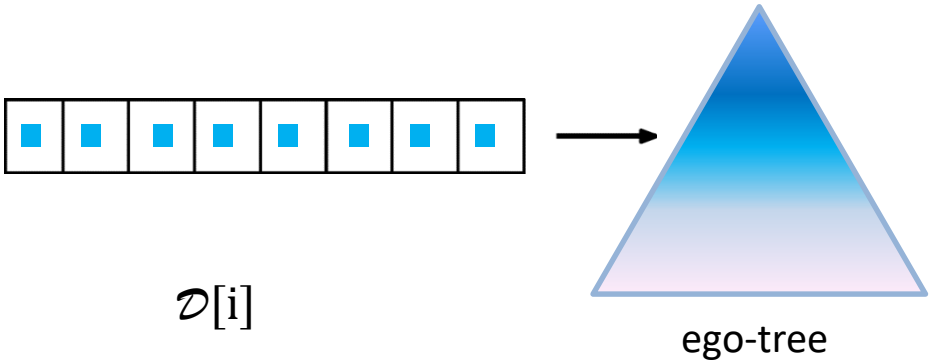
D

Demand-Aware Network Designs of Bounded Degree. Chen Avin, Kaushik Mondal, and Stefan Schmid. **DISC**, 2017.

Achieving Entropy Limit: Algorithms

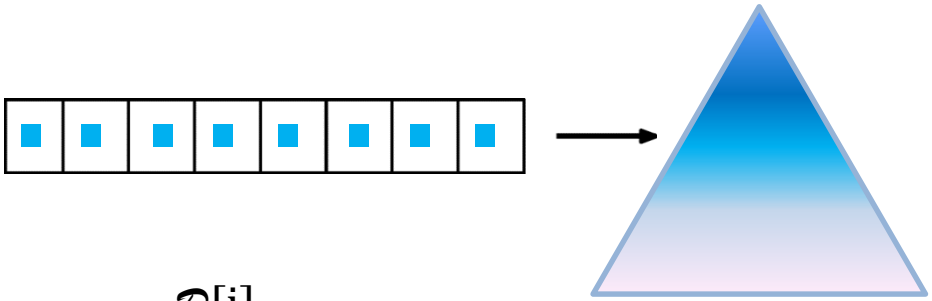
Ego-Trees Revisited

- ego-tree: optimal tree for a row (= given source)



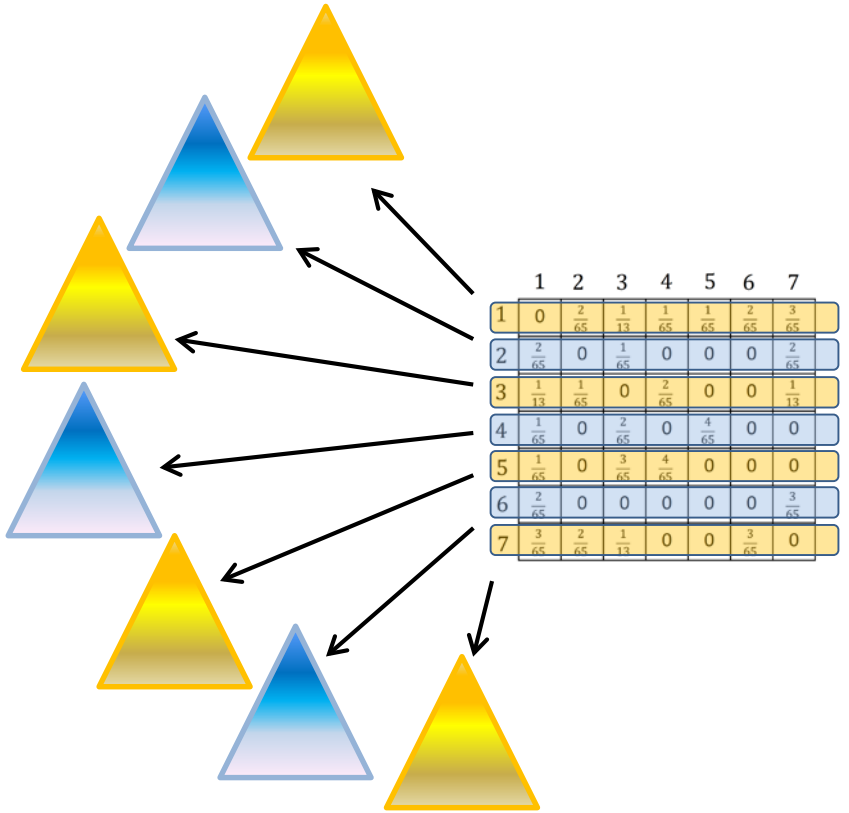
Ego-Trees Revisited

- ego-tree: optimal tree for a row (= given source)



$\mathcal{D}[i]$

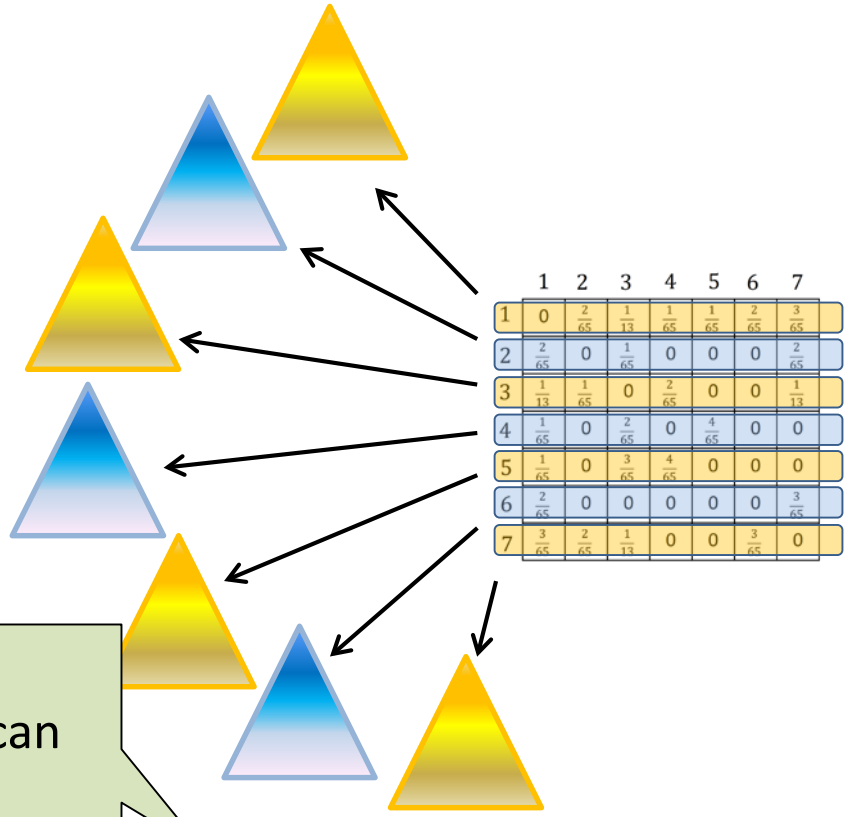
ego-tree



Can we merge the trees ***without distortion*** and ***keep degree low***?

Ego-Trees Revisited

- ego-tree: optimal tree for a row (= given source)



For **sparse demands** yes: enough ***low-degree nodes*** which can serve as "**helper nodes**"!

$\mathcal{D}[i]$

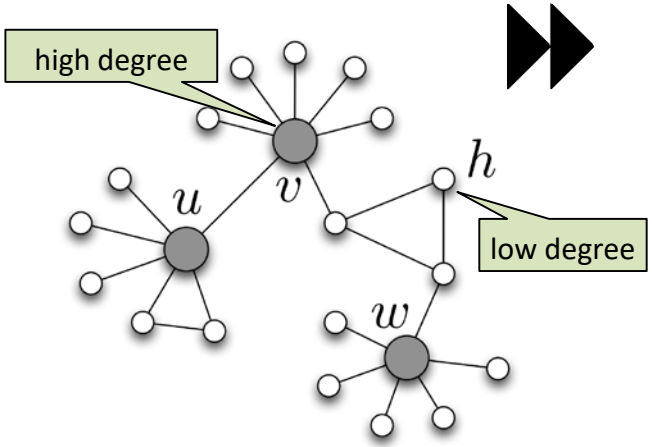
ego-tree

Can we merge the trees ***without distortion*** and ***keep degree low***?

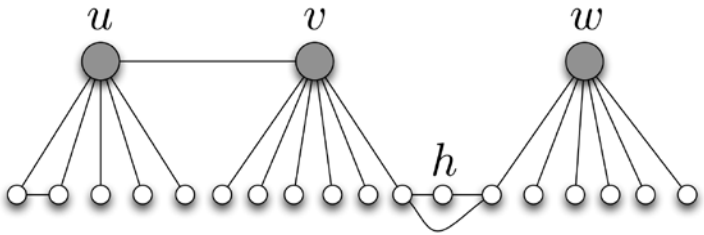
From Trees to Networks

Idea: Degree Reduction

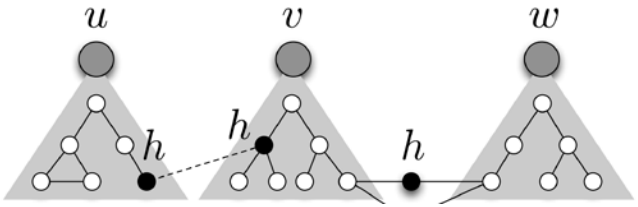
① **Demand graph**



② **Hierarchical representation**



③ **Add low-degree nodes as helpers**



Taking union of ego-trees results in ***high degree***:
u and *v* will appear in many ego-trees

Node *h* ***helps edge (u, v)*** by participating in *ego-tree(u)* as a relay node toward *v* and *in ego-tree(v)* as a relay toward *u*

Demand-Aware Network Designs of Bounded Degree. Chen Avin, Kaushik Mondal, and Stefan Schmid. **DISC**, 2017.

But: How to design DANs which also leverage *temporal structure*?

Inspiration from **self-adjusting datastructures** again!

An Analogy

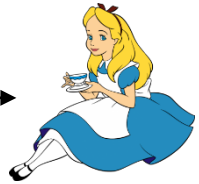
Static vs dynamic demand-aware networks!?

DANs vs **SANs**?

An Analogy to Coding

„Coming to Barcelona?“

00110101…



if demand ***arbitrary*** and ***unknown***

worst case network: Full BW
worst case coding: 00, 01, 10, 11

log diameter

log # bits / symbol

„Coming to Barcelona?“

An Analogy to Coding

01011…

if demand ***arbitrary*** and ***unknown***

worst case network: Full BW	*log diameter*
worst case coding: 00, 01, 10, 11	*log # bits / symbol*

DAN!

if demand ***known*** and ***fixed***

entropy?

entropy / symbol

static Demand-Aware Nets
static Huffman: 1, 01, 001, 000

An Analogy to Coding

„Coming to Barcelona?“

011…

if demand ***arbitrary*** and ***unknown***

worst case network: Full BW	*log diameter*
worst case coding: 00, 01, 10, 11	*log # bits / symbol*

Dynamic DANs: Aka. **Self-Adjusting Networks** (SANs)!

DAN!

SAN!

if demand ***known*** and ***fixed***

entropy?	static Demand-Aware Nets
entropy / symbol	static Huffman: 1, 01, 001, 000

if demand ***unknown*** but ***reconfigurable***

dynamic Demand-Aware Nets
dynamic Huffman codes

An Analogy to Coding

„Coming to Barcelona?“

011…

if demand ***arbitrary*** and ***unknown***

worst case network: Full BW	*log diameter*
worst case coding: 00, 01, 10, 11	*log # bits / symbol*

Dynamic DANs: Aka. **Self-Adjusting Networks** (SANs)!

DAN!

SAN!

if demand ***known*** and ***fixed***

static Demand-Aware Nets
static Huffman: 1, 01, 001, 000

Can exploit ***spatial locality***!

if demand ***unknown*** but ***reconfigurable***

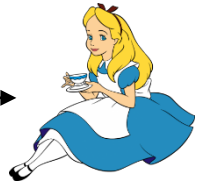
dynamic Demand-Aware Nets
dynamic Huffman codes

Additionally exploit ***temporal locality***!

An Analogy to Coding

„Coming to Barcelona?“

011…



if demand ***arbitrary*** and ***unknown***

worst case network: Full BW — *log diameter*

worst case coding: 00, 01, 10, 11 — *log # bits / symbol*

Dynamic DANs: Aka. **Self-Adjusting Networks** (SANs)!

DAN!

SAN!

if demand ***know***

Can exploit ***spati***

„Cheating“: need to know demand!

Aware Nets

static Huffman: 1, 01, 001, 000

if demand ***unknown*** but ***re***

dynamic Demand

Huffman codes

Need online algorithms!

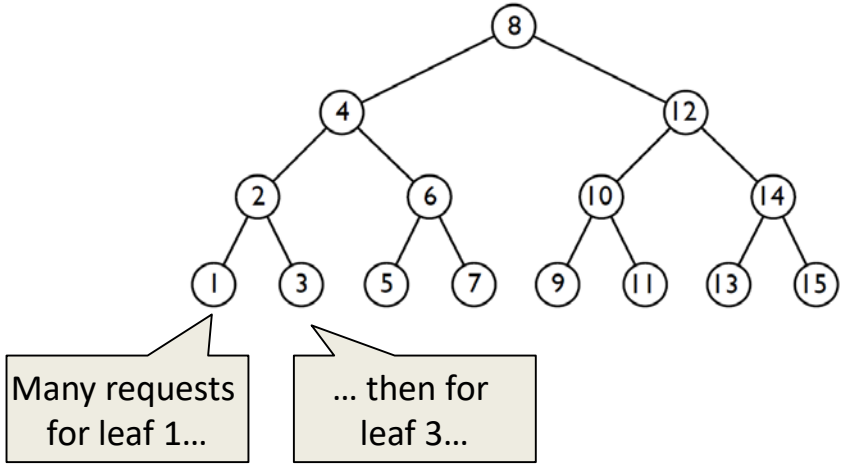
Additionally exploit ***temporal locality***!

Analogous to *Datastructures*: Oblivious…

- Traditional, **fixed** BSTs do not rely on any assumptions on the demand
- Optimize for the **worst-case**
- Example **demand**:

  1,…,1,3,…,3,5,…,5,7,…,7,…,log(n),…,log(n)

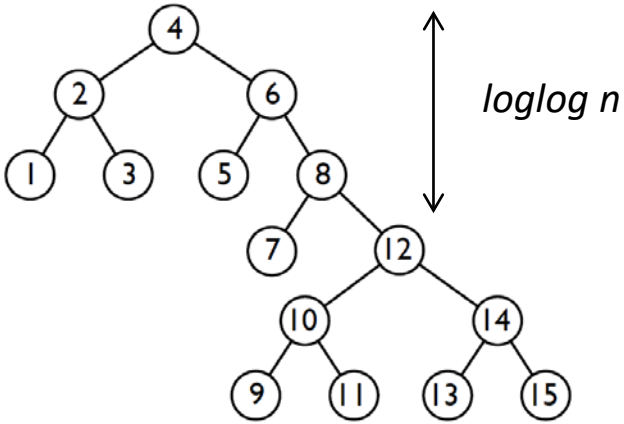
  many *many* *many* *many* *many*
- Items stored at ***O(log n)*** from the root, **uniformly** and **independently** of their frequency



Corresponds to ***max possible demand***!

… Demand-Aware …

- **Demand-aware fixed** BSTs can take advantage of ***spatial locality*** of the demand
- E.g.: place frequently accessed elements close to the root
- E.g., **Knuth/Mehlhorn/Tarjan** trees
- Recall example **demand**: 1,…,1,3,…,3,5,…,5,7,…,7,…,log(n),…,log(n)
  - Amortized cost ***O(loglog n)***

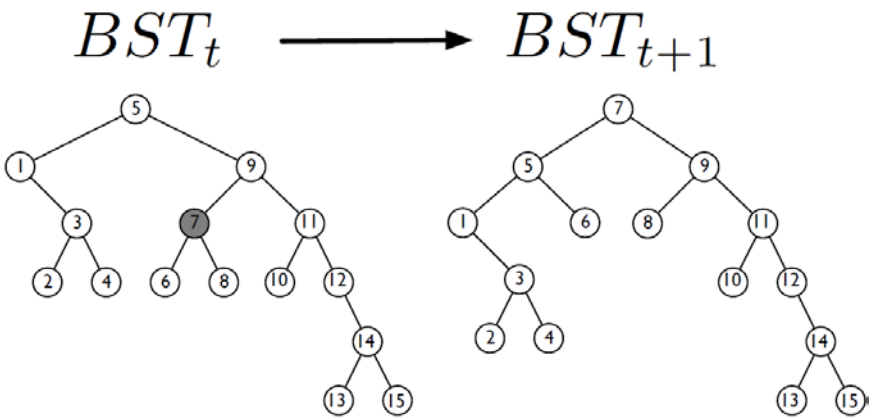


Amortized cost corresponds to ***empirical entropy of demand***!

… Self-Adjusting!

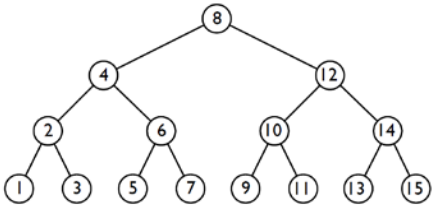
- **Demand-aware reconfigurable** BSTs can additionally take advantage of ***temporal locality***

- By moving accessed element to the root: amortized cost is ***constant***, i.e., O(1)
  - Recall example **demand**: 1,…,1,3,…,3,5,…,5,7,…,7,…,log(n),…,log(n)



Datastructures

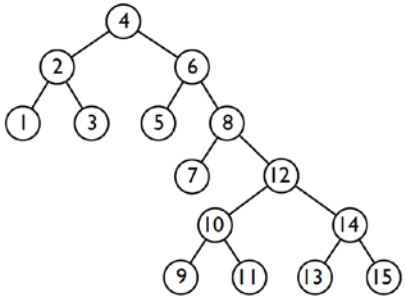
Oblivious



Lookup

O(log n)

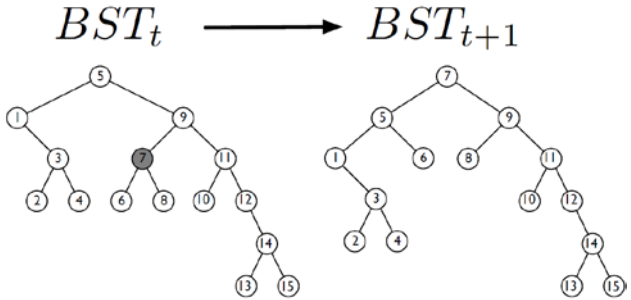
Demand-Aware



Exploit **spatial locality**:

empirical entropy O(loglog n)

Self-Adjusting

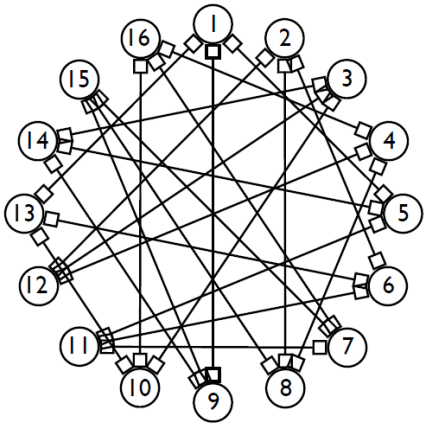


Exploit **temporal locality** as well:

O(1)

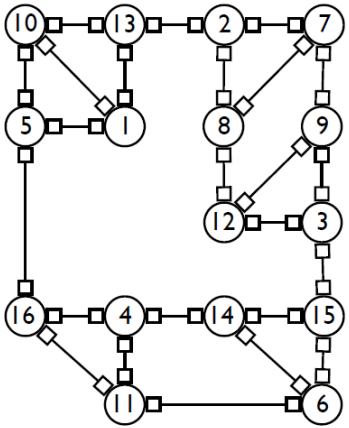
Analogously for Networks

Oblivious



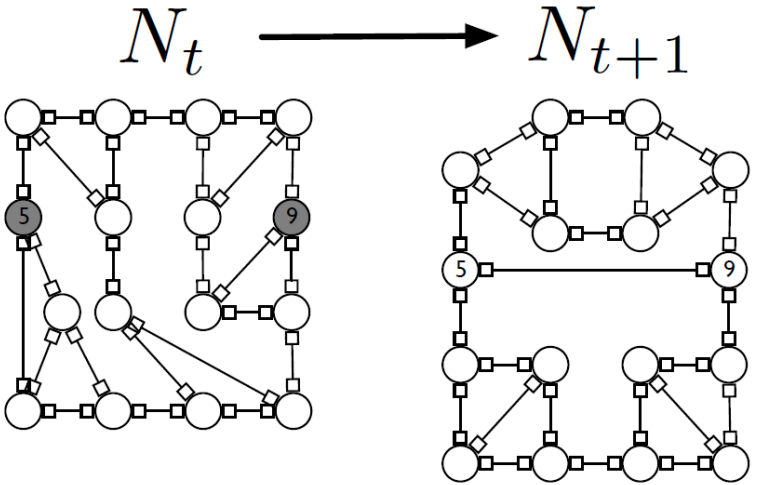
Const degree
(e.g., **expander**):
route lengths ***O(log n)***

DAN



Exploit **spatial locality**

SAN



Exploit **temporal locality** as well

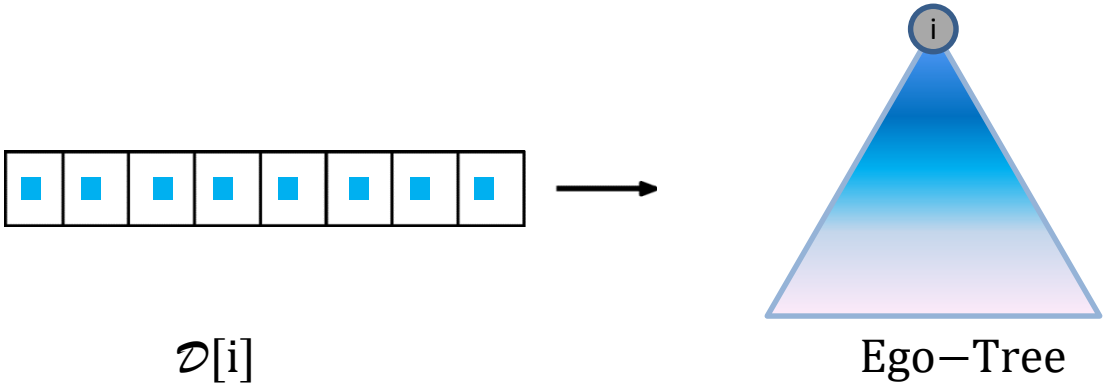
Avin, S.: Toward Demand-Aware Networking: A Theory for Self-Adjusting Networks. **SIGCOMM CCR** 2018.

Algorithms for Self-Adjusting Networks

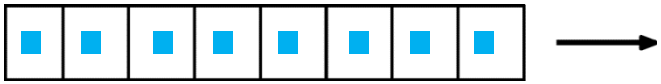
From trees to networks!

Ego-trees strike back!

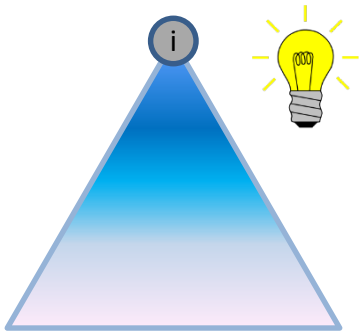
Ego-Trees!



Ego-Trees!



$\mathcal{D}[i]$



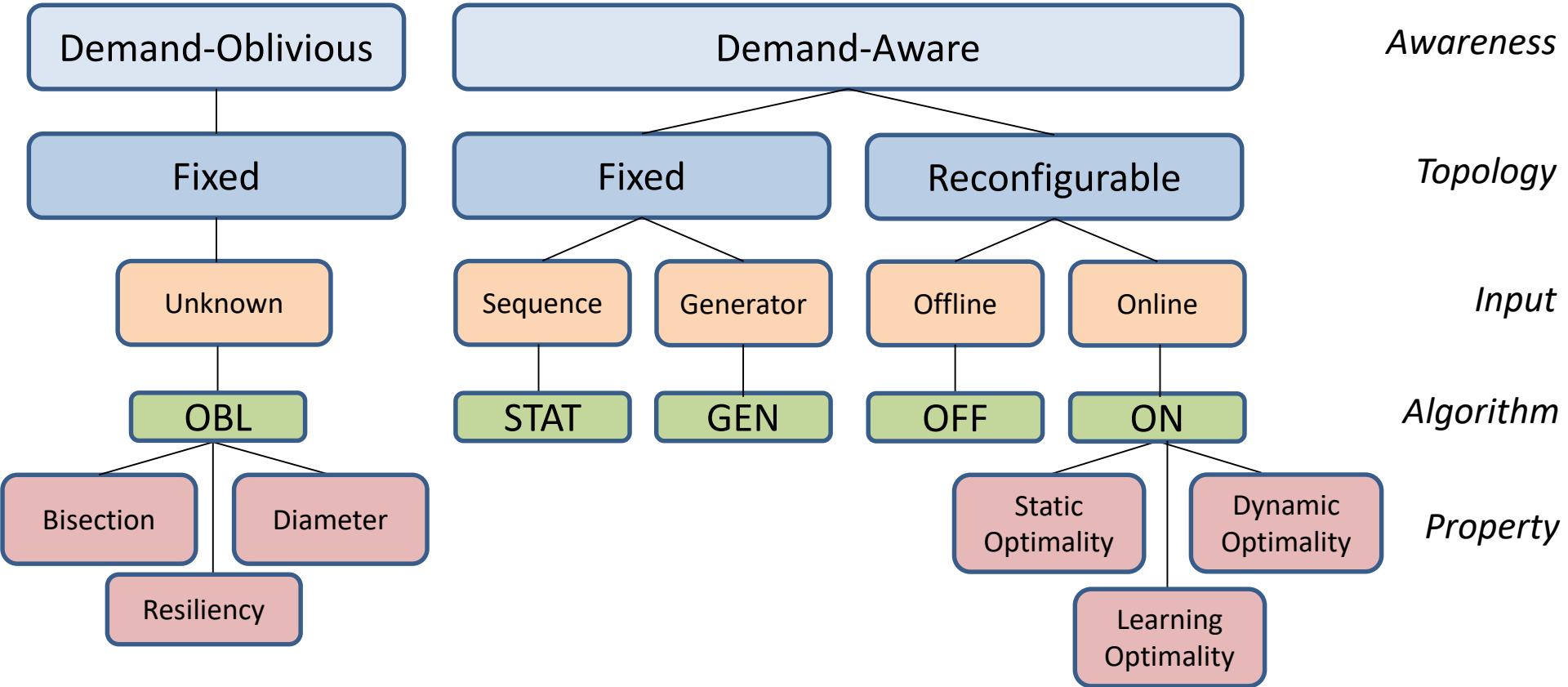
Ego−Tree

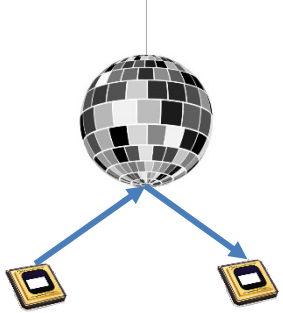
Idea: use our old approach but now let each node ***adjust its ego-tree***!

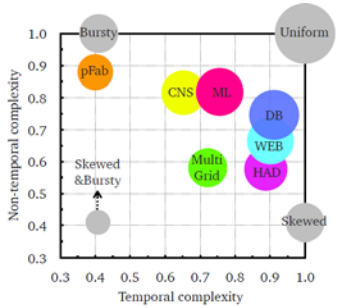
A Dynamic Ego-Tree: Splay Tree

Uncharted Landscape!

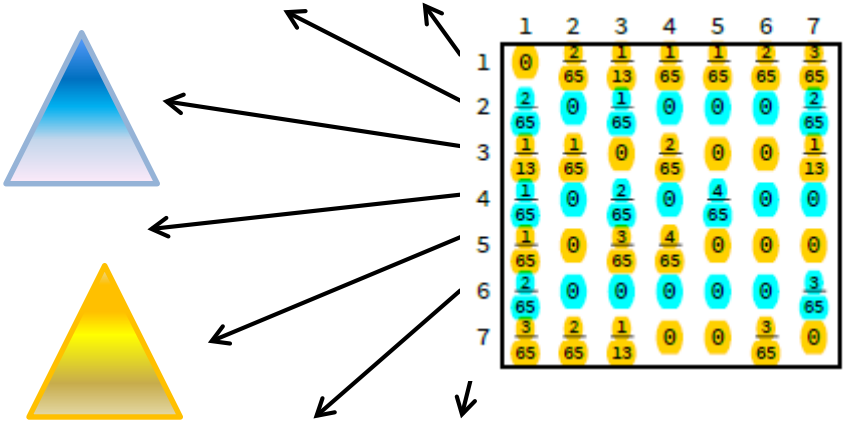
Toward Demand-Aware Networking: A Theory for Self-Adjusting Networks. **SIGCOMM CCR,** 2018.





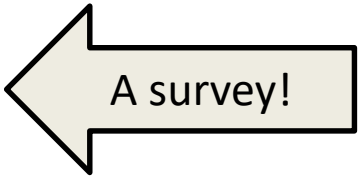


Thank you! Questions?



Further Reading

Demand-aware networks



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.

Measuring the Complexity of Network Traffic Traces
Chen Griner, Chen Avin, Manya Ghobadi, and Stefan Schmid.
arXiv, 2019.

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.

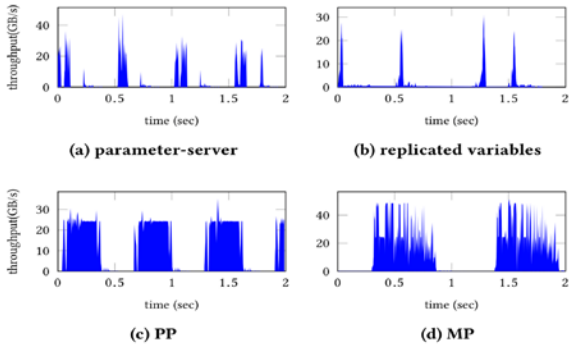
SplayNet: Towards Locally Self-Adjusting Networks
Stefan Schmid, Chen Avin, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler, and Zvi Lotker.
IEEE/ACM Transactions on Networking (**TON**), Volume 24, Issue 3, 2016. Early version: IEEE **IPDPS** 2013.

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.

How Predictable is Traffic?

Even if reconfiguration fast, control plane (e.g., data collection) can become a bottleneck. However, many good examples:

- Machine learning applications
- Trend to disaggregation (specialized racks)
- Datacenter communication dominated by elephant flows
- Etc.



ML workload (GPU to GPU):
deep convolutional neural network

Predictable from their dataflow graph