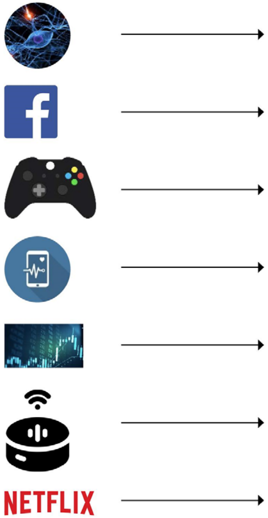
Online Dynamic B-Matching
with Applications to Reconfigurable Datacenter Networks

Marcin Bienkowski, David Fuchssteiner, Jan Marcinkowski, and Stefan Schmid

Uniwersytet Wrocławski
universität wien

Trend: Data-Centric Applications

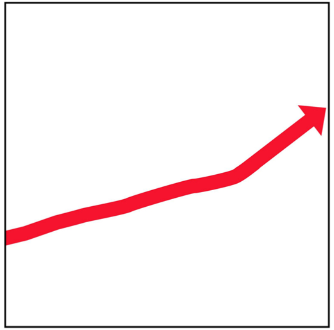


Datacenters ("hyper-scale")

Interconnecting networks:

a **critical infrastructure**

of our digital society.



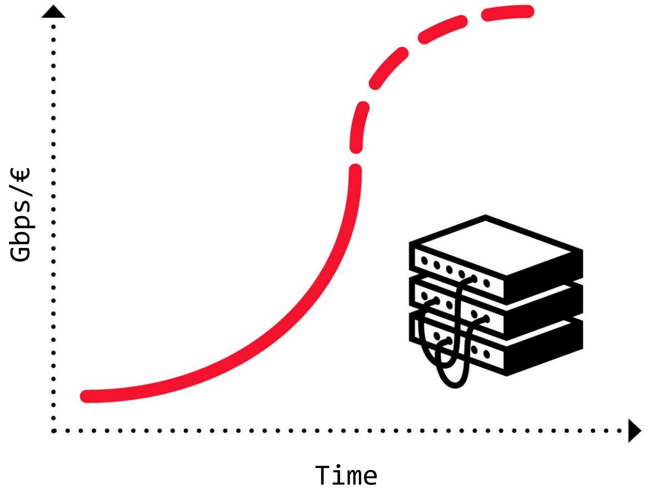
Source: Facebook

Problem: Huge Infrastructure, Inefficient Use

- Network equipment reaching capacity limits
  - Transistor density rates stalling
  - “End of **Moore‘s Law** in networking”

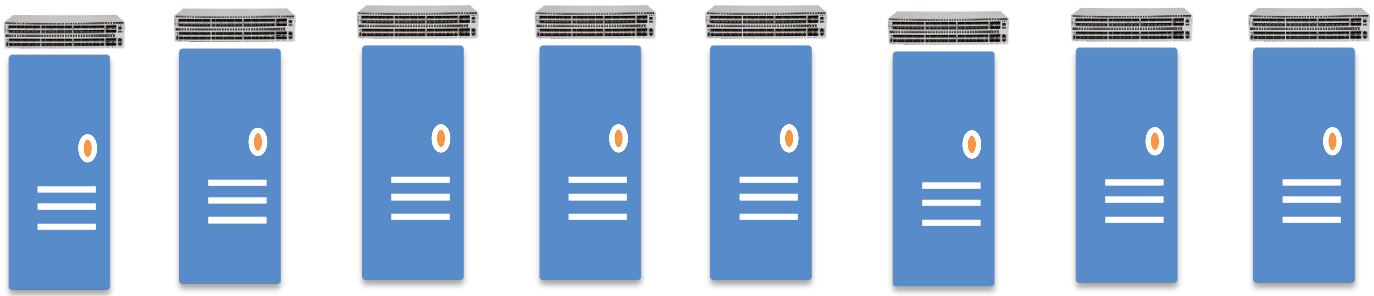
- Hence: more equipment, larger networks

- Resource intensive and: **inefficient**

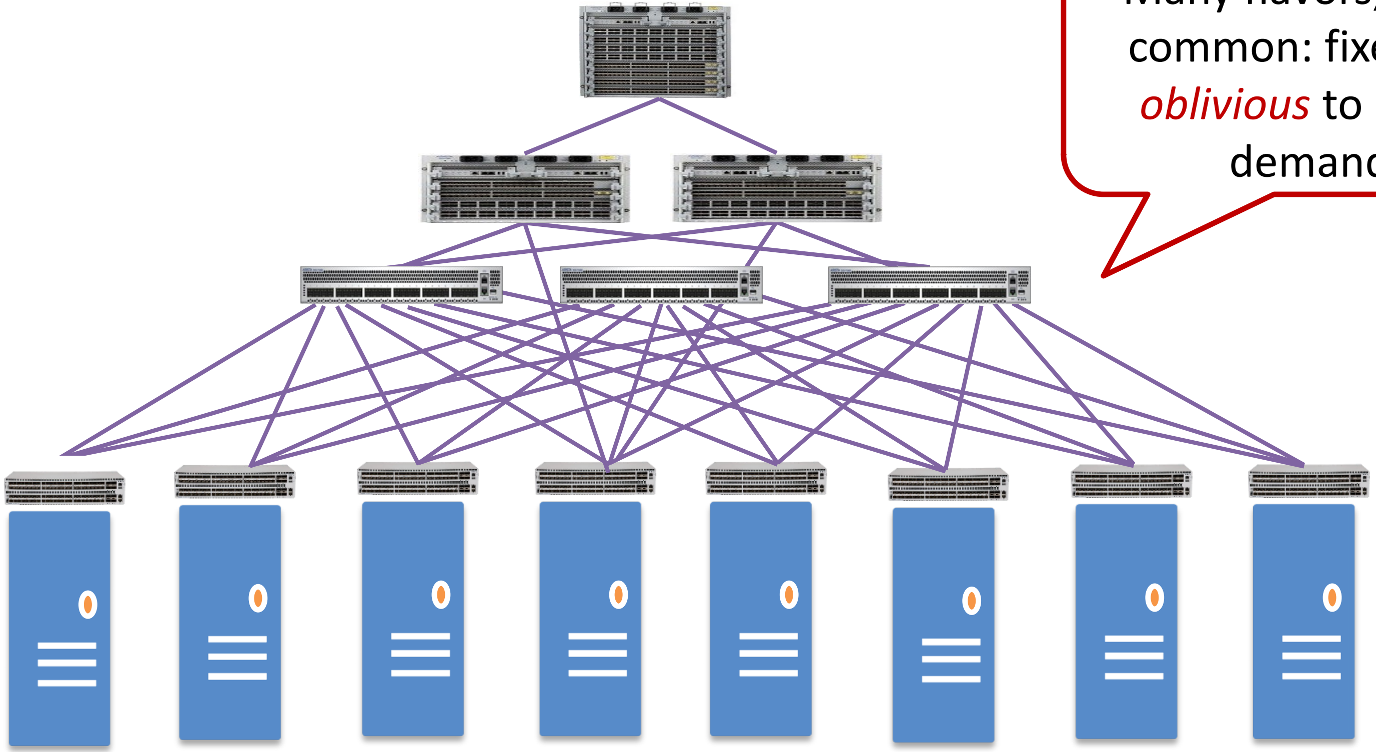


Root Cause: Fixed and Demand-Oblivious Topology

How to interconnect?



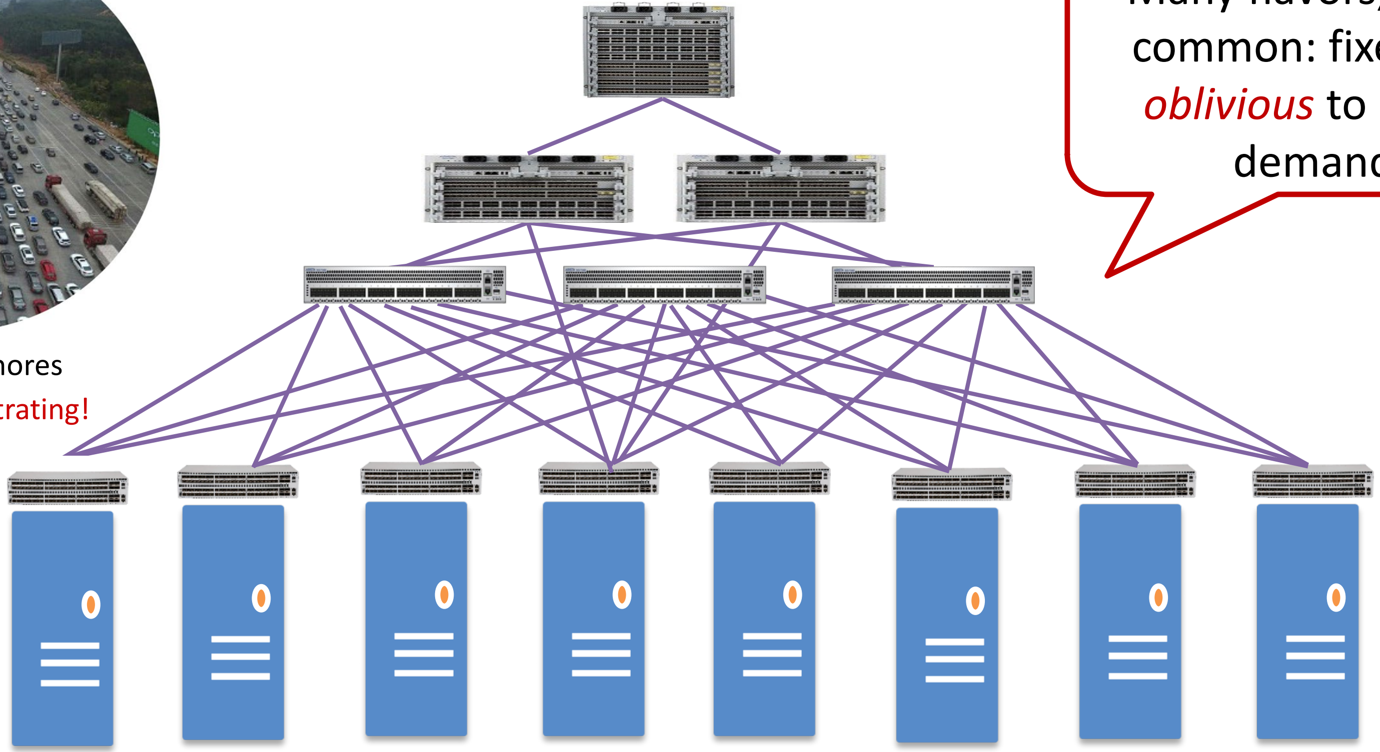
Root Cause: Fixed and Demand-Oblivious Topology



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

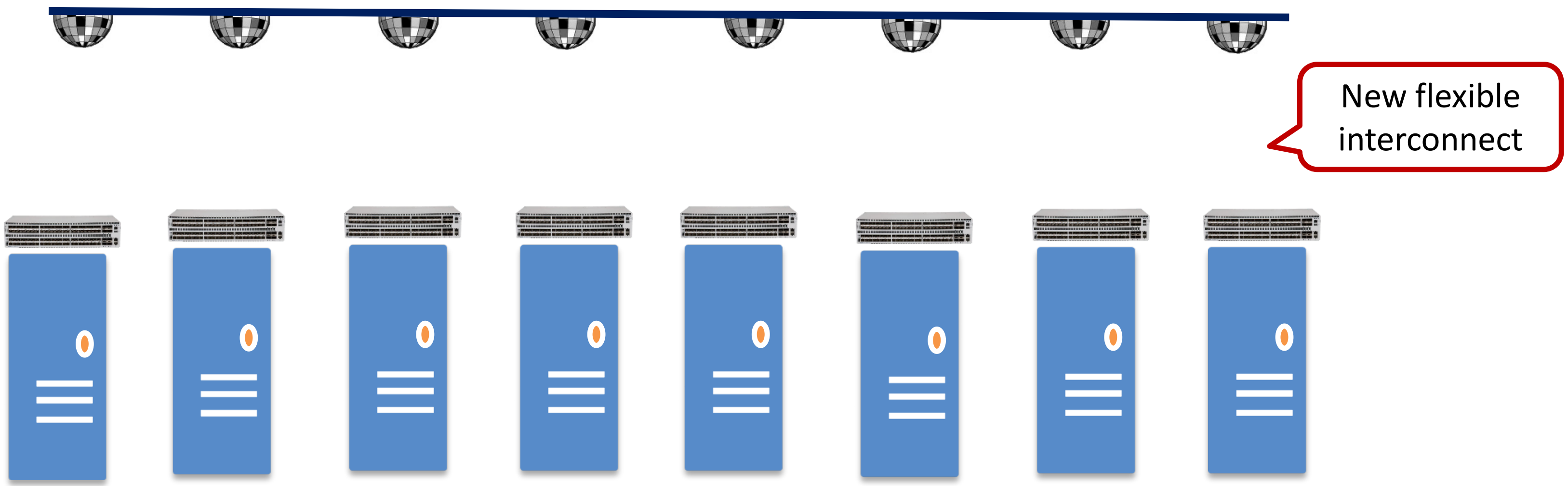
Root Cause: Fixed and Demand-Oblivious Topology

Highway which ignores actual traffic: frustrating!



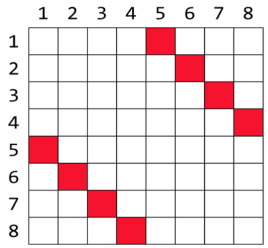
Many flavors, but in common: fixed and *oblivious* to actual demand.

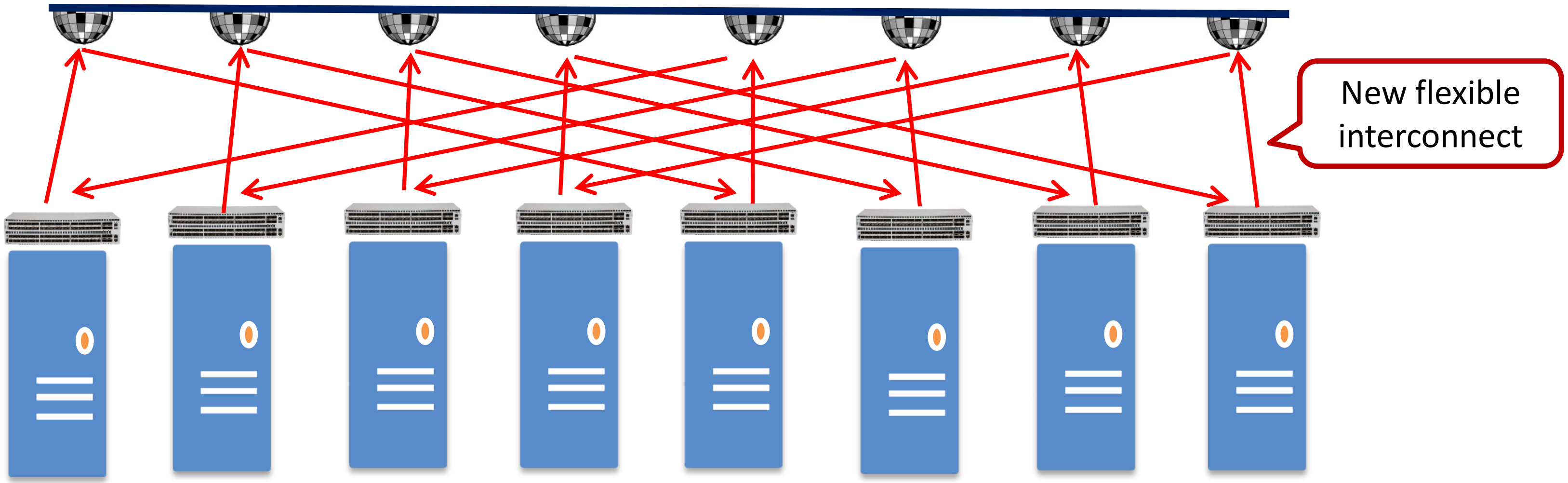
Our Vision: Self-Adjusting Networks



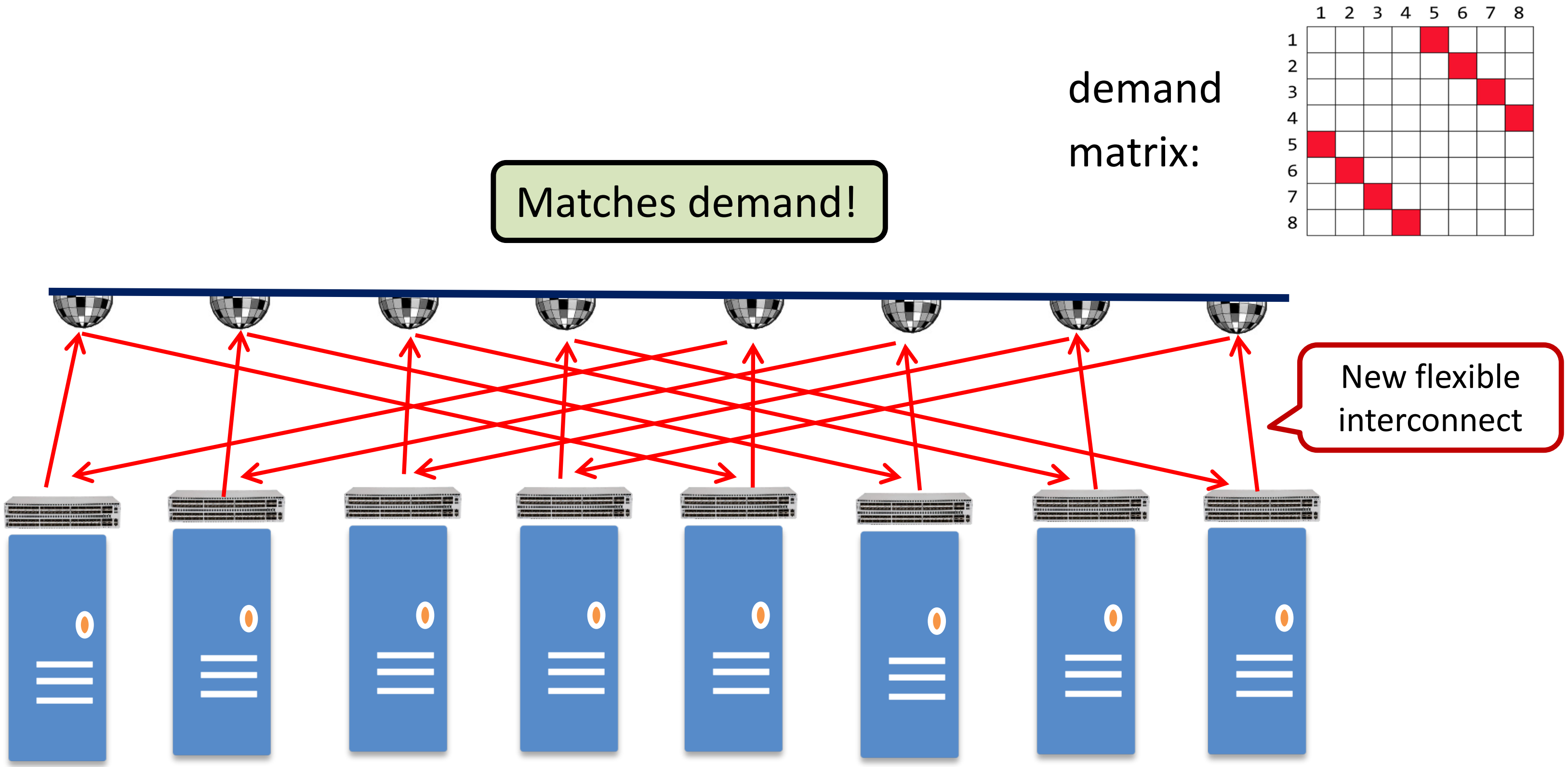
Our Vision: Self-Adjusting Networks

demand matrix:

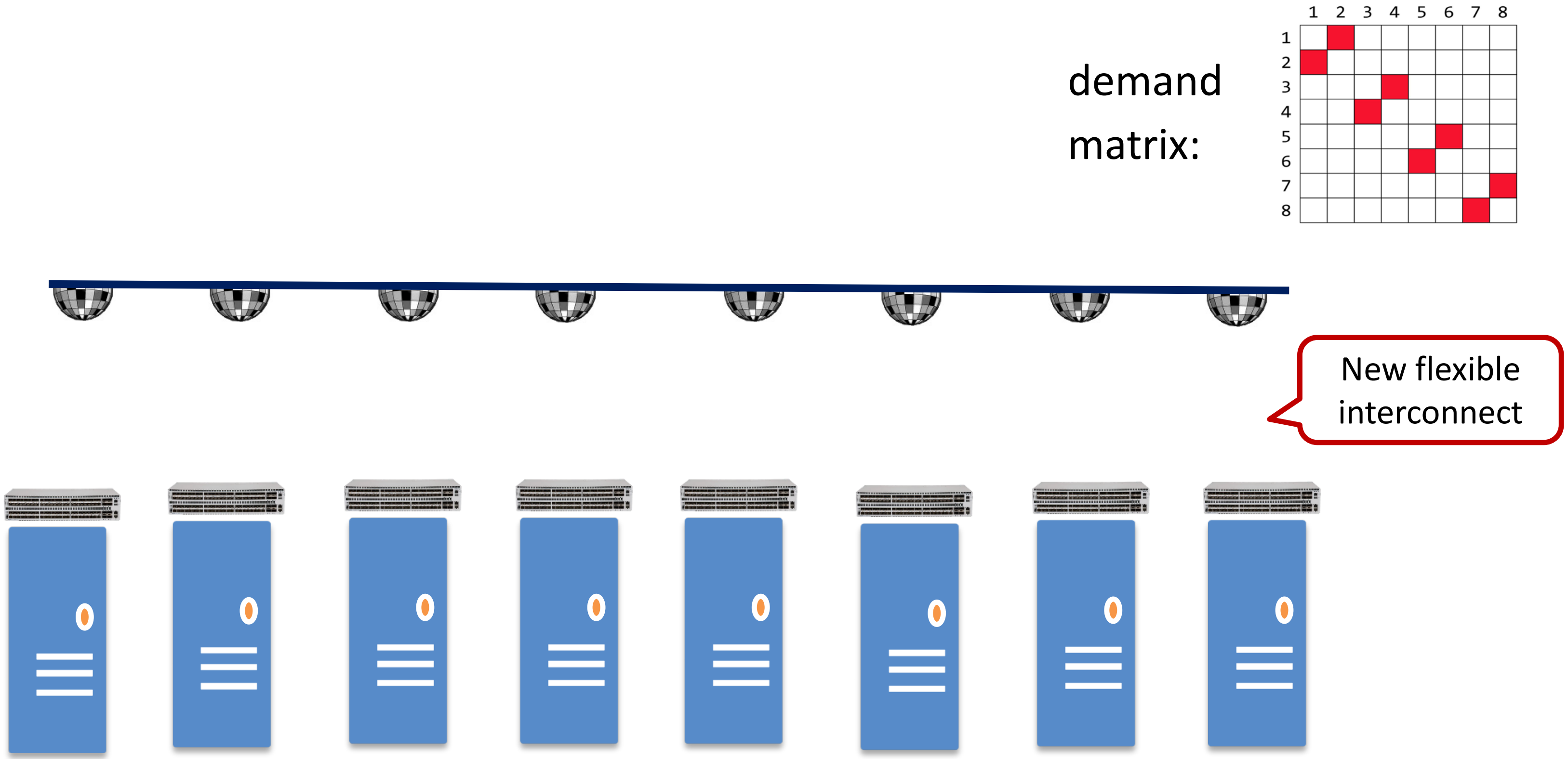




Our Vision: Self-Adjusting Networks

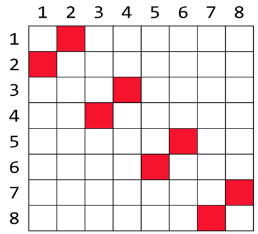


Our Vision: Self-Adjusting Networks

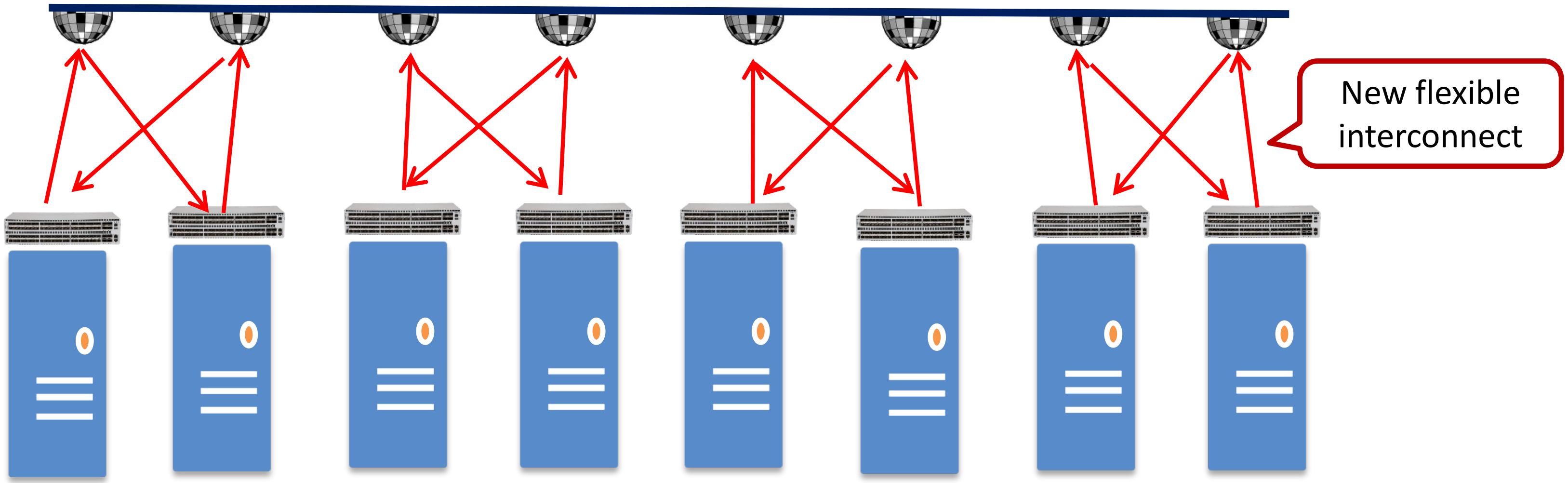


Our Vision: Self-Adjusting Networks

demand matrix:



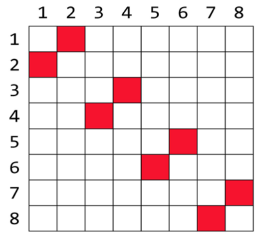
Matches demand!

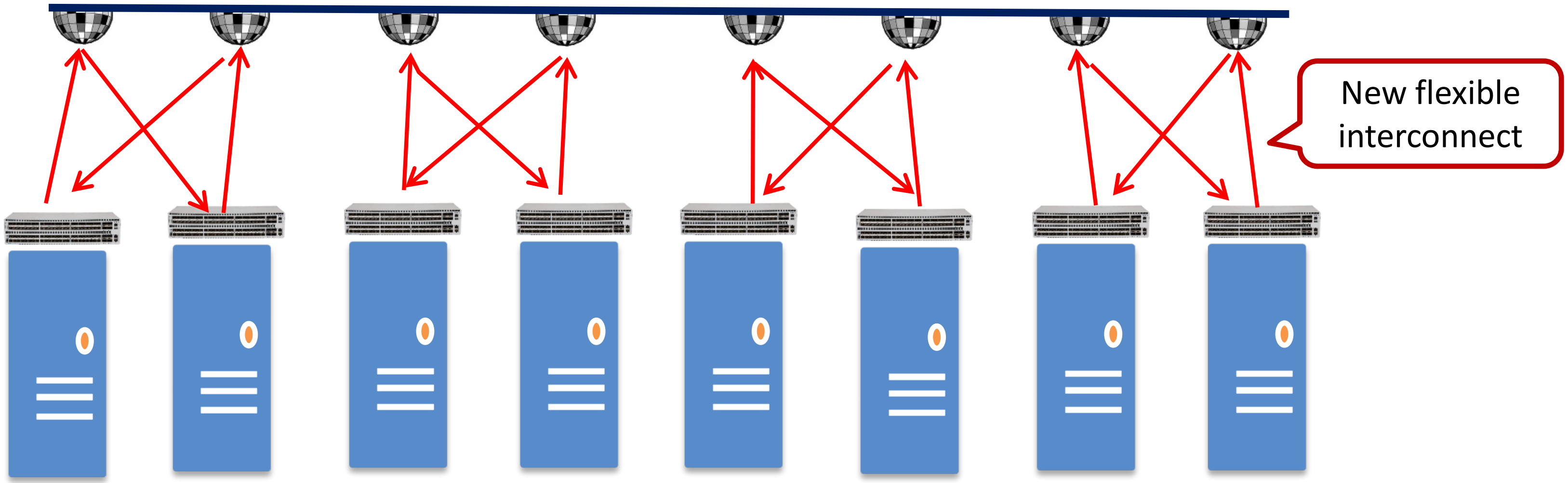


Our Vision: Self-Adjusting Networks

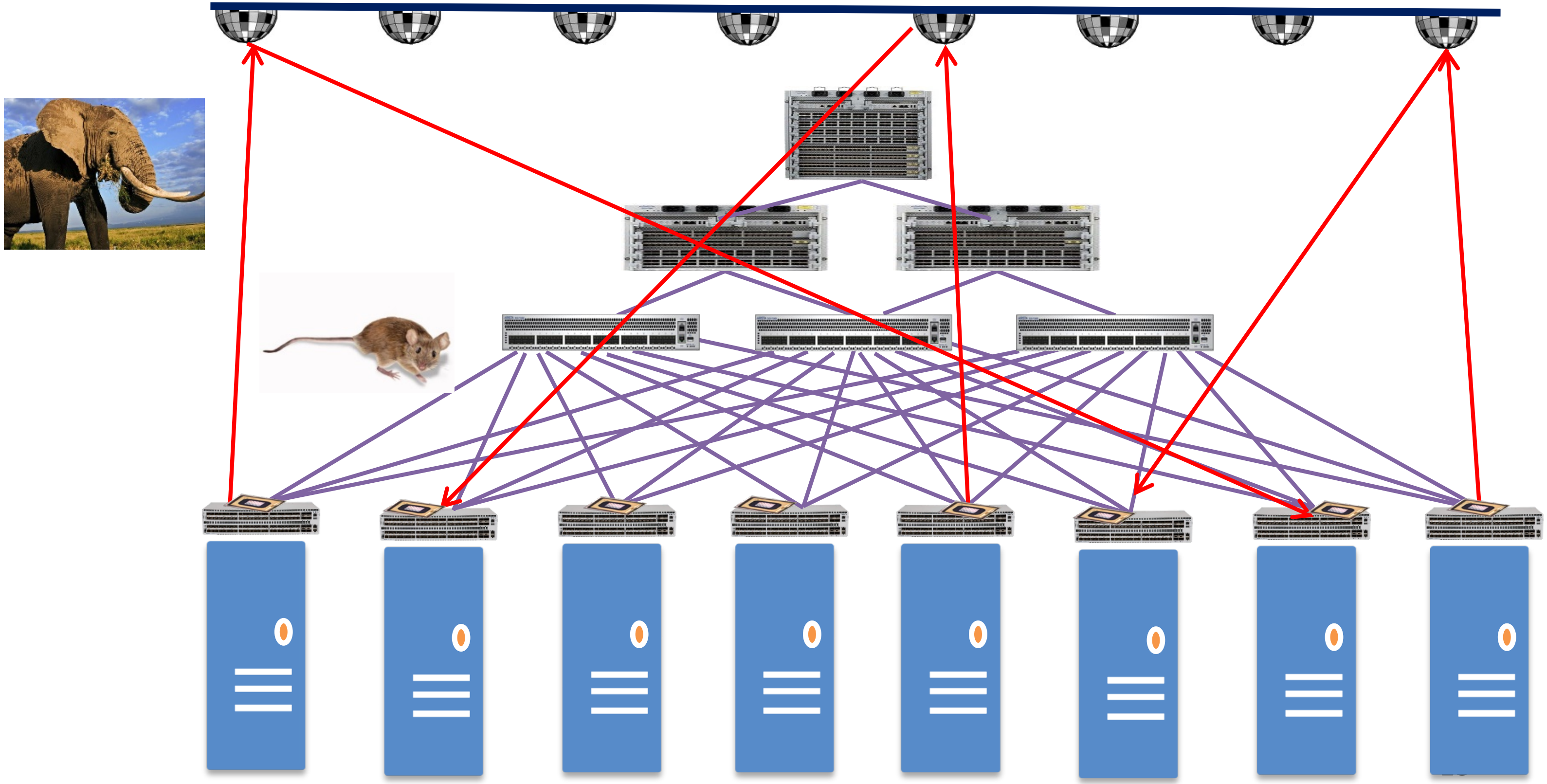
Self-adjusting networks!

demand matrix:

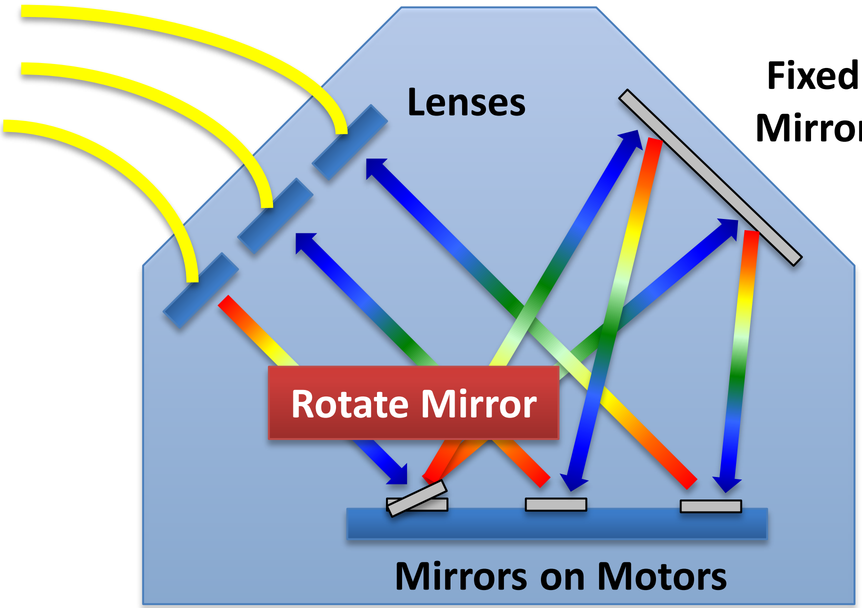




In Reality: Mostly Hybrid Architectures

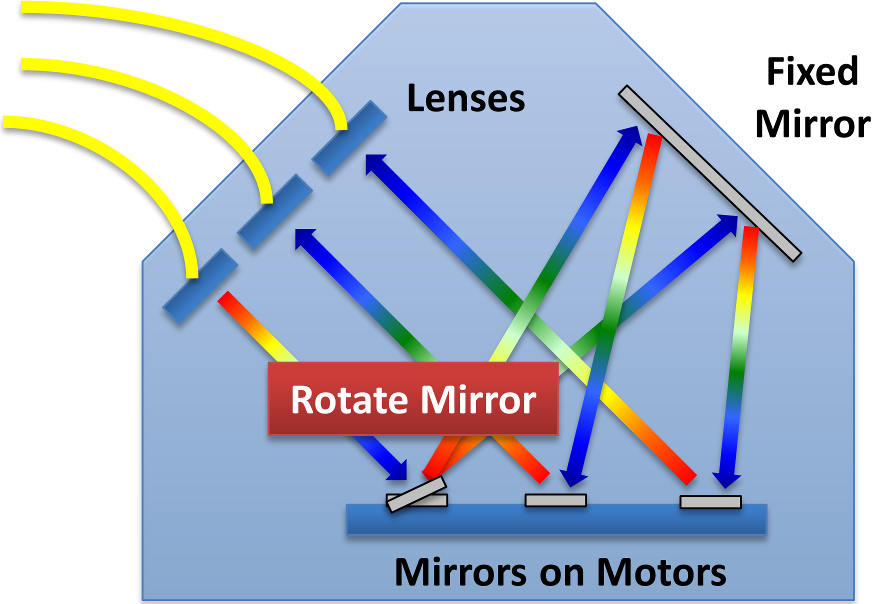


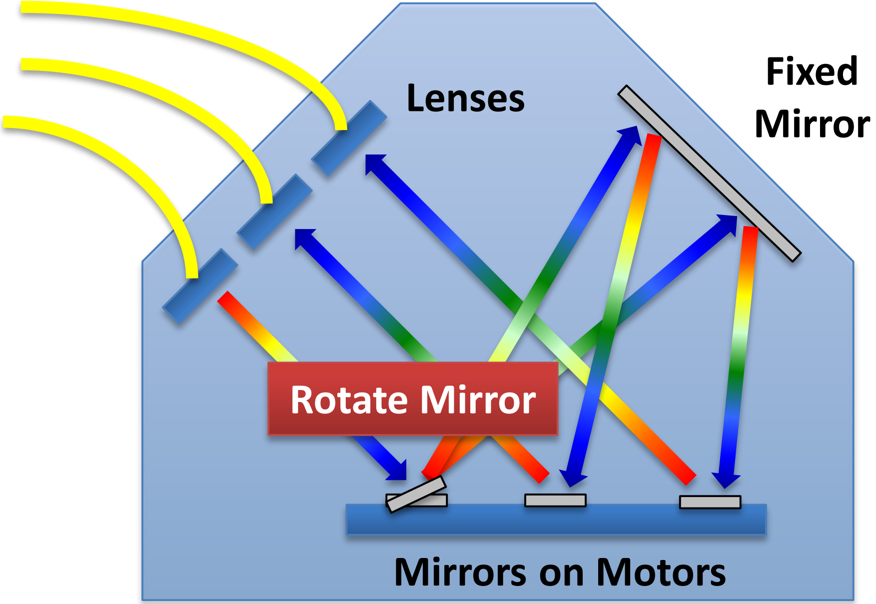
Enabling Technologies, e.g.: Optical Circuit Switch



Provides a **matching**!

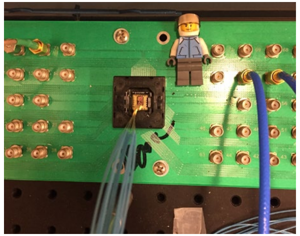
Enabling Technologies, e.g.: Optical Circuit Switch

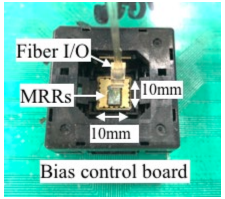


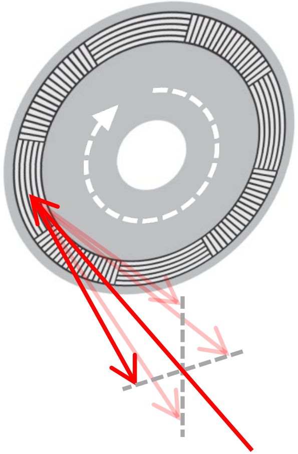


Provides a **B-matching**!

Other Prototypes







Based on silicon photonics

2-NEMS

Rotating disks

Further reading:
Wade et al., A Bandwidth-Dense, Low Power Electronic-Photonic Platform and Architecture for Multi-Tbps Optical I/O [OFC’18]
Porter et al., “Integrating Microsecond Circuit Switching into the Data Center”, Sigcomm’13

Focus of **this paper**: How to exploit these technologies algorithmically?

Roadmap

- The model: dynamic B-matching
- An online O(B)-competitive algorithm
- Simulation results

Roadmap

- **The model: dynamic B-matching**
- An online O(B)-competitive algorithm
- Simulation results

The Model in A Nutshell

Input:

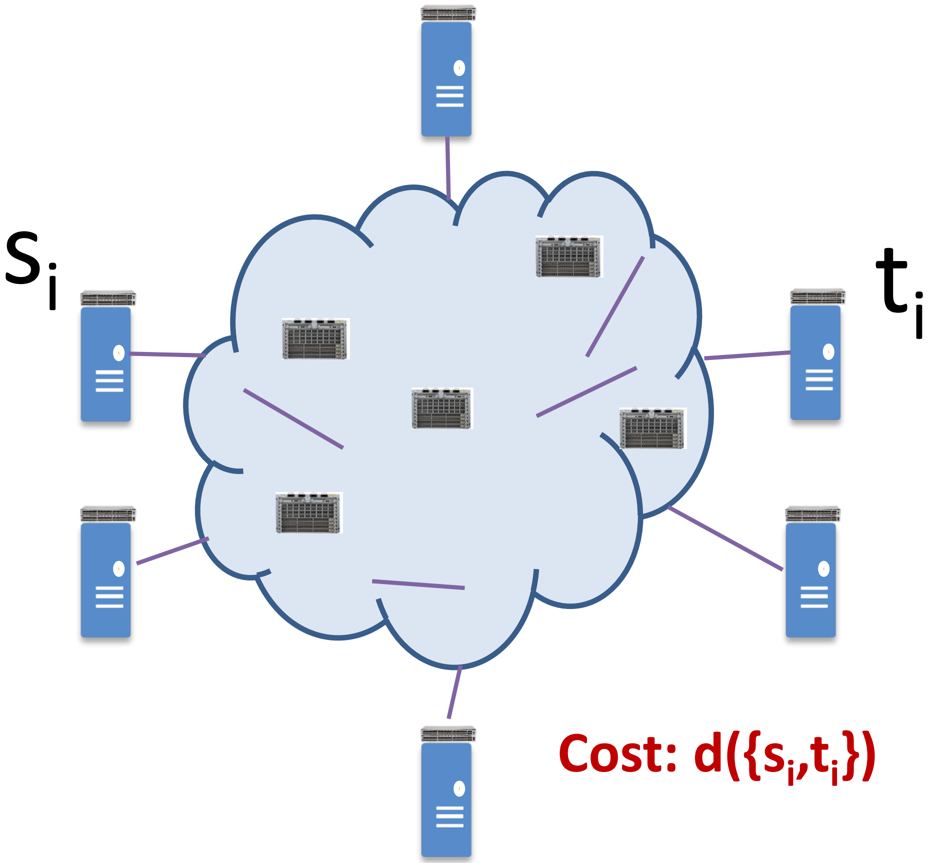
- Hybrid network
  - Arbitrary fixed network
  - B OCS (for B-matching)
- Communication requests
  - σ= {s_1,t_1}, {s_2,t_2}, {s_3,t_3}, ... arriving over time between servers

Output:

- Sequence of B-matchings

Cost:

- Adding/removing edge: α
- Routing:
  - Along fixed network: distance d({s_i,t_i})
  - Along optical edge: cost 0



The Model in A Nutshell

Input:

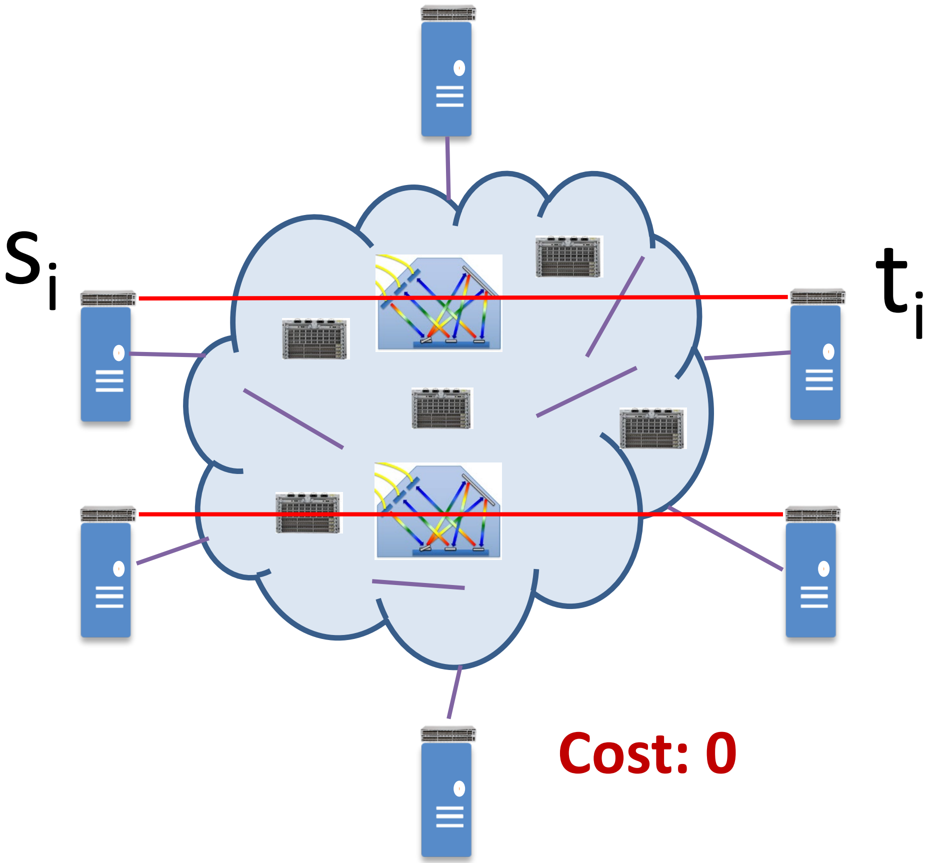
- Hybrid network
  - Arbitrary fixed network
  - B OCS (for B-matching)
- Communication requests
  - σ= {s_1,t_1}, {s_2,t_2}, {s_3,t_3}, ... arriving over time between servers

Output:

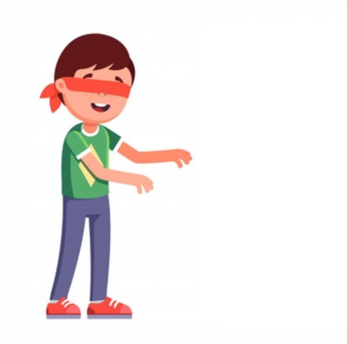
- Sequence of B-matchings

Cost:

- Adding/removing edge: α
- Routing:
  - Along fixed network: distance d({s_i,t_i})
  - Along optical edge: cost 0



Objective: Competitive Ratio



Roadmap

- The model: dynamic B-matching
- **An online O(B)-competitive algorithm**
- Simulation results

Connection to Online Paging...

online algorithm keeps at most B edges incident to any node w

$\Leftrightarrow$

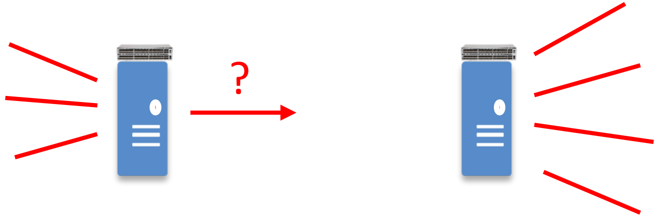
each node maintains a cache of at most B items

If request in cache, cost is 0; otherwise d. Fetching to cache costs α.
Known as online paging with bypassing!

... But With a Catch!

Coherence challenge:

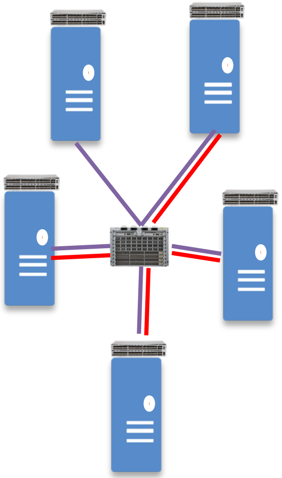
- We may simply run independent paging algorithms at all nodes
- But then decisions whether to evict or keep may conflict



Lower Bound $\Omega(B)$

Idea: still by a reduction to caching

- Assume graph is a star of B+2 nodes
  - Edge length 1
- Initial matching: connect center to B leaves
- Adversary chooses „missing node“ α times (a „chunk“)
  - DET pays at least α per chunk
- OPT could choose a different matching
  - Pays $\alpha k/B$ for k chunks



The BMA Algorithm: O(B) Competitive

Algorithm 1 Algorithm BMA

```
Initialization:                                   ▹ Matching is empty and counters are zero
    M ← ∅
    for each edge e do
        h_e ← 0

Request τ = {u, v} arrives:
    if τ ∉ M then
        h_τ ← h_τ + 1
        if h_τ = T_τ then                         ▹ If τ becomes saturated,
            Execute FixSaturation(u, τ)
            Execute FixSaturation(v, τ)
            if h_τ = T_τ then                     ▹ and if no desaturation event occured,
                Execute FixMatching(u)
                Execute FixMatching(v)
                M ← M ∪ {τ}                       ▹ add τ to the matching.

Routine FixSaturation(w, τ):
    E'_w = E_w \ {τ}
    if |E'_w ∩ {e : h_e = T_e}| ≥ b then          ▹ If the number of saturated node pairs from E'_w is at least b,
        for each edge e ∈ E_w do                  ▹ reset counters of all node pairs from E_w (desaturation event at w).
            h_e ← 0

Routine FixMatching(w):
    if |M ∩ E_w| = b then                         ▹ If there are already b incident matching edges,
        Pick any e* ∈ M ∩ E_w such that h_e* < T_e*   ▹ remove any unsaturated edge e* from the matching.
        M ← M \ {e*}
```

- Keep counter h_e for each edge e
  - (Usually) the number of times e was requested since last eviction
- If $h_e = \alpha$, edge becomes saturated
  - If an edge is saturated, it is in the matching

The BMA Algorithm: O(B) Competitive

Algorithm 1 Algorithm BMA

```
Initialization:                                   ▷ Matching is empty and counters are zero
    M ← ∅
    for each edge e do
        h_e ← 0

Request τ = {u, v} arrives:
    if τ ∉ M then
        h_τ ← h_τ + 1
        if h_τ = T_τ then                         ▷ If τ becomes saturated,
            Execute FixSaturation(u, τ)
            Execute FixSaturation(v, τ)
            if h_τ = T_τ then                     ▷ and if no desaturation event occured,
                Execute FixMatching(u)
                Execute FixMatching(v)
                M ← M ∪ {τ}                       ▷ add τ to the matching.

Routine FixSaturation(w, τ):
    E'_w = E_w \ {τ}
    if |E'_w ∩ {e : h_e = T_e}| ≥ b then          ▷ If the number of saturated node pairs from E'_w is at least b,
        for each edge e ∈ E_w do                  ▷ reset counters of all node pairs from E_w (desaturation event at w).
            h_e ← 0

Routine FixMatching(w):
    if |M ∩ E_w| = b then                         ▷ If there are already b incident matching edges,
        Pick any e* ∈ M ∩ E_w such that h_{e*} < T_{e*}   ▷ remove any unsaturated edge e* from the matching.
        M ← M \ {e*}
```

- Keep counter h_e for each edge e
  - (Usually) the number of times e was requested since last eviction
- If $h_e = \alpha$, edge becomes saturated
  - If an edge is saturated, it i the matching

Actually, that's more complicated! When the counter for edge e = (u,v) gets equal to α, u and v run an agreement scheme.

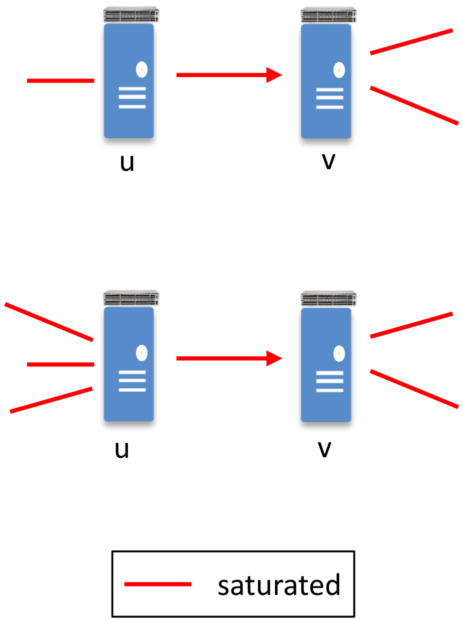
Agreement Scheme (Example B=3)

Case 1: “Easy case”

- After making (u,v) saturated, the number of incident saturated edges is at most B

Case 2:

- In this case, (u,v) cannot become saturated as u would have too many incident saturated edges
- We reset all counters for edges incident to u to zero.
  - This will become problematic in the analysis



Approach and Analysis

- Matching lazily follows saturation scheme:
  - If saturated, then in the matching
  - When an edge stops being saturated (it is reset to zero), it is not removed from the matching, but is a candidate for future removal

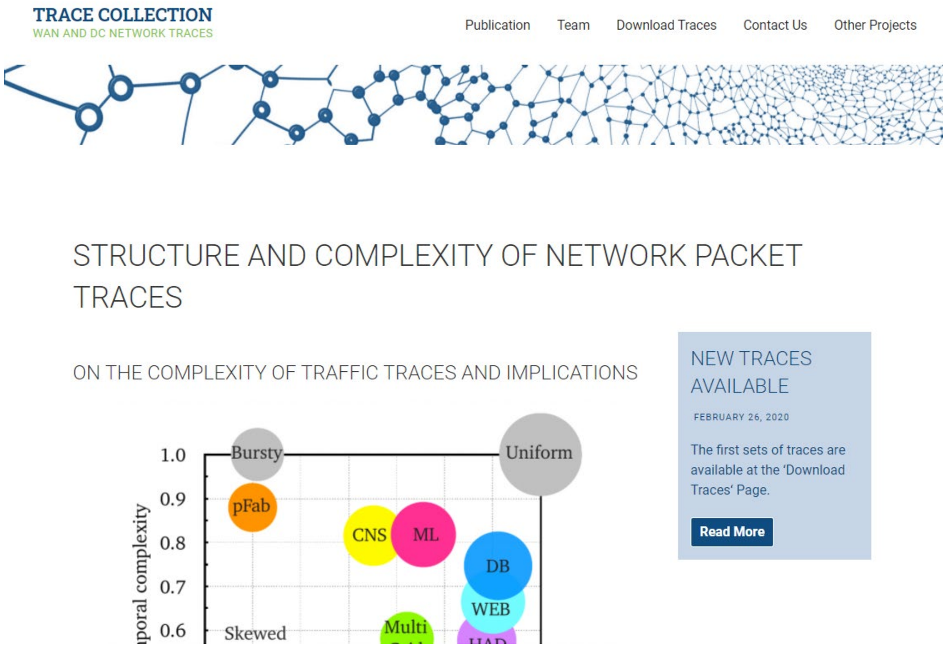
- Analysis ideas
  - When (B+1)-th edge incident to node w becomes saturated, this is a witness that OPT has to pay α for the requests that correspond to saturated edges.
  - Hard part of the proof: you cannot make this argument for a single node w, as incident edges can be reset multiple times and hence ALG's cost associated with w can be much larger than $(B + 1)\ \alpha$.

Roadmap

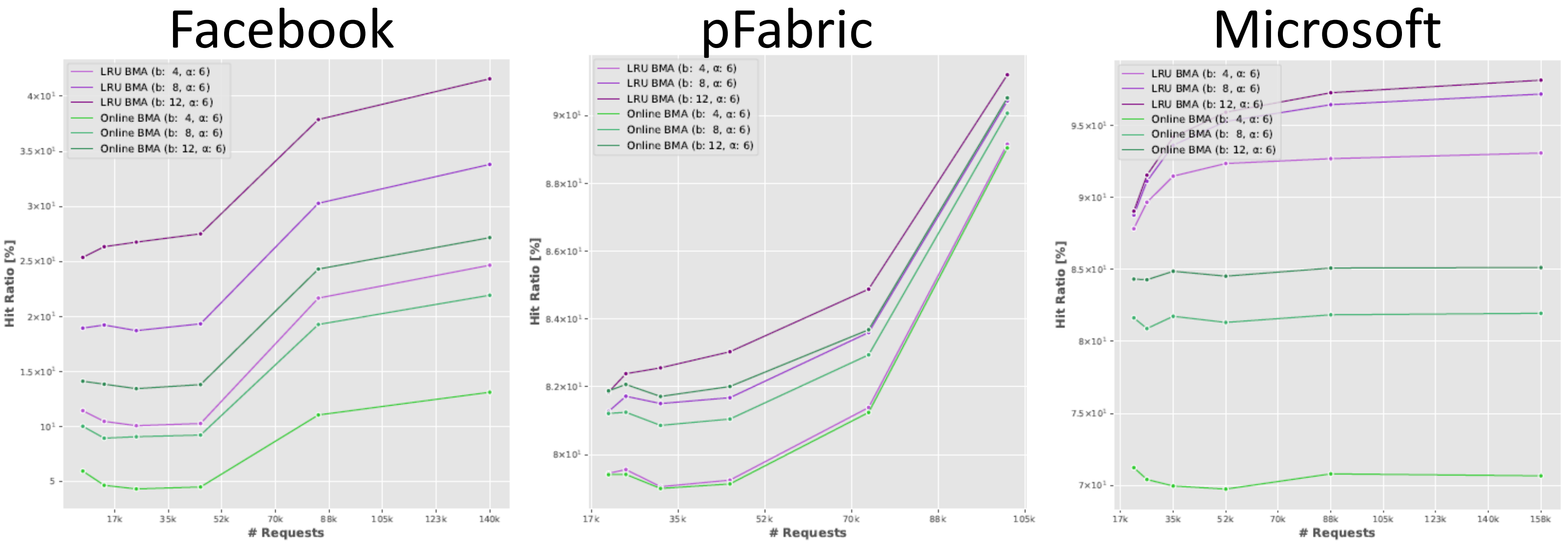
- The model: dynamic B-matching
- An online O(B)-competitive algorithm
- **Simulation results**

Traces

- Different datacenter traces
  - Real: Facebook, Microsoft
  - Synthetic: pFabric
- Available online
  - E.g., trace-collection.net

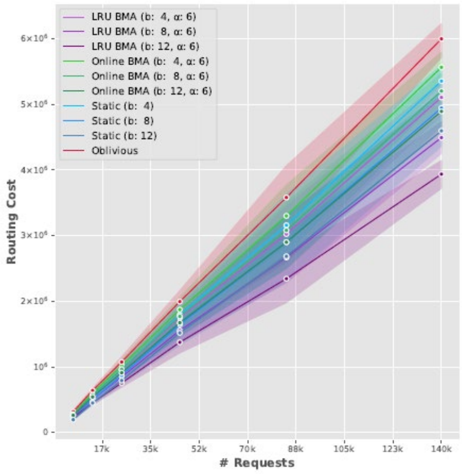


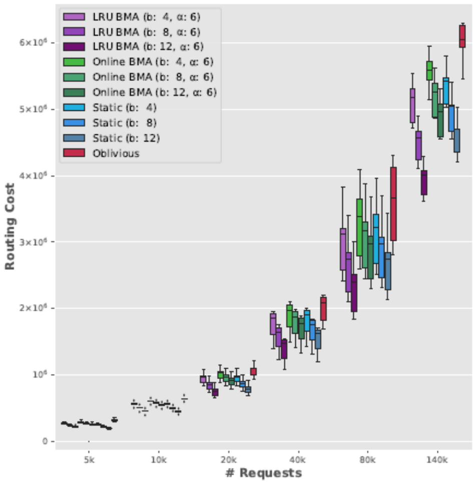
Empirical Results: Cache Hit Ratio



- High hit ratio especially for pFabric and Microsoft
- Expected from empirical studies on trace complexity (at SIGMETRICS'20)

Empirical Results: Routing Costs (Facebook)



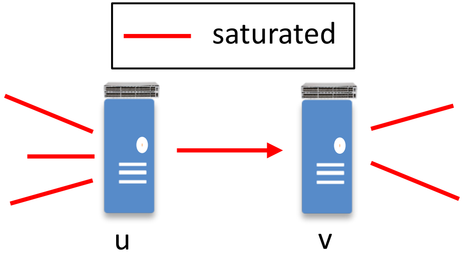


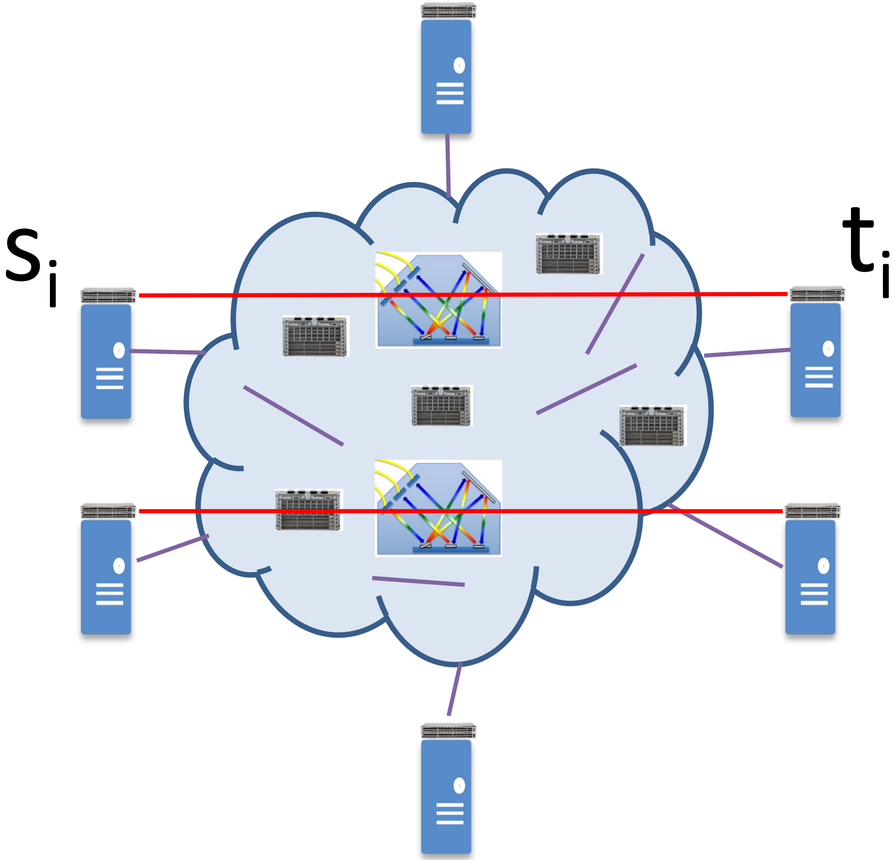
- Oblivious always performs worse than Static, Online BMA and BMA with LRU
- Online BMA comes close to Static, which knows the demands ahead of time
- We expect that under longer request sequences, when larger shifts in the communication patterns are likely to appear, the online approach will outperform the static offline algorithm

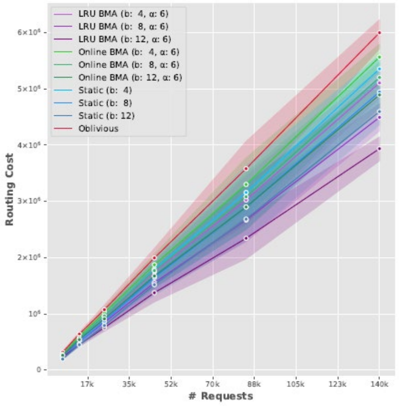
Conclusions

- Asymptotically optimal online B-matching
  - Practically attractive: decentralized caching algorithm
  - Problem relevant beyond reconfigurable datacenters
- Future work: randomized algorithms?

Thank you! Questions?







Further reading:

[On the Complexity of Traffic Traces and Implications](#)
Chen Avin, Manya Ghobadi, Chen Griner, and Stefan Schmid.
ACM **SIGMETRICS** and ACM Performance Evaluation Review (**PER**).