

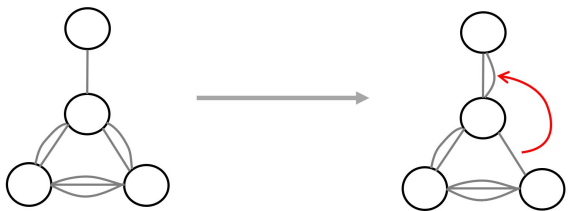
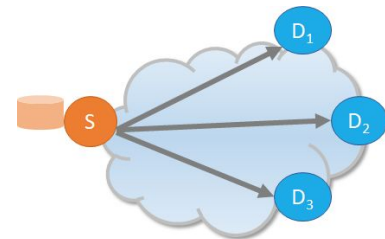
Reconfigurable Networks: Enablers, Algorithms, Complexity

Ramakrishnan Durairajan, Klaus-Tycho Forster, Stefan Schmid

Tutorial @ ACM Sigmetrics 2019
Phoenix, Arizona, USA

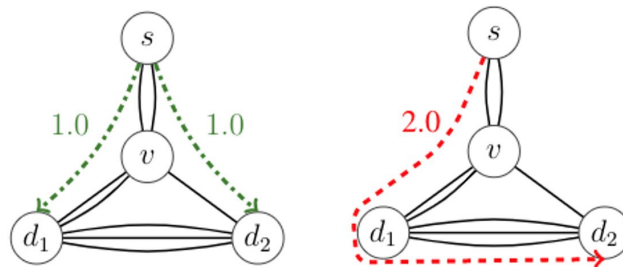
DarTree: Reconfigurable WAN Multicast

- **Multicast transfers:** common in WAN
 - E.g., 90% multicast traffic of inter-DC traffic from **Baidu**



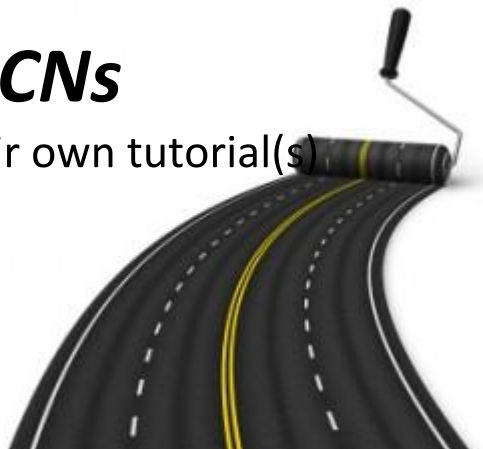
- How to support in reconfigurable WANs?
 - Can change **wavelengths** and **routing**!

Large potential and interesting optimization problem: Tmrw @ IWQoS



Roadmap

- Last part: WAN, now: Data Center Networks (DCNs)
 - This part: Mostly 1) estimate demand, 2) build topologies, 3) repeat
 - Next part: Dynamic settings, demand structures, data structure connections
- Focus on ***Algorithmic Challenges*** in ***DCNs***
 - Technology, system developments etc. would require their own tutorial(s)
- ***Paper-based*** approach
 - Selection of papers from 2009 to 2019



Timeline

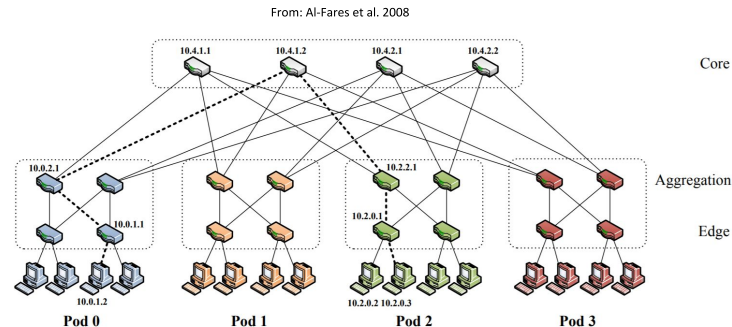
Reconfiguration time: from milliseconds *to microseconds* (and decentralized).

Survey of Reconfigurable Data Center Networks. Foerster and Schmid. SIGACT News, 2019.

- 2009 • – *Flyways* [51]: Steerable antennas (narrow beamwidth at 60 GHz [78]) to serve hotspots
- 2010 • – *Helios* [33]/*c-Through* [98, 99]: Hybrid switch architecture, maximum matching (Edmond’s algorithm [30]), single-hop reconfigurable connections ($O(10)ms$ reconfiguration time).
 - *Proteus* [21, 89]: k reconfigurable connections per ToR, multi-hop path stitching, multi-hop reconfigurable connections (weighted b -matching [69], edge-exchanges for connectivity [72], wavelength assignment via edge-coloring [67] on multigraphs)
- 2011 • – Extension of *Flyways* [51] to better handle practical concerns such as stability and interference for 60GHz links, along with greedy heuristics for dynamic link placement [45]
- 2012 • – *Mirror Mirror on the ceiling* [106]: 3D-beamforming (60 Ghz wireless), signals bounce off the ceiling
- 2013 • – *Mordia* [31, 32, 77]: Traffic matrix scheduling, matrix decomposition (Birkhoff-von-Neumann (BvN) [18, 97]), fiber ring structure with wavelengths ($O(10)\mu s$ reconfiguration time)
 - *SplayNets* [6, 76, 82]: Fine-grained and online reconfigurations in the spirit of self-adjusting datastructures (all links are reconfigurable), aiming to strike a balance between short route lengths and reconfiguration costs
- 2014 • – *REACToR* [56]: Buffer burst of packets at end-hosts until circuit provisioned, employs [77]
 - *Firefly* [14] Combination of Free Space Optics and Galvo/switchable mirrors (small fan-out)
- 2015 • – *Solstice* [57]: Greedy perfect matching based hybrid scheduling heuristic that outperforms BvN [77]
 - Designs for optical switches with a reconfiguration latency of $O(10)ns$ [3]
- 2016 • – *ProjecToR* [39]: Distributed Free Space Optics with digital micromirrors (high fan-out) [38] (Stable Matching [26]), goal of (starvation-free) low latency
 - *Eclipse* [95, 96]: $(1 - 1/e^{(1-\epsilon)})$ -approximation for throughput in traffic matrix scheduling (single-hop reconfigurable connections, hybrid switch architecture), outperforms heuristics in [57]
- 2017 • – *DAN* [7, 8, 11, 12]: Demand-aware networks based on reconfigurable links only and optimized for a demand snapshot, to minimized average route length and/or minimize load
 - *MegaSwitch* [23]: Non-blocking circuits over multiple fiber rings (stacking rings in [77] doesn’t suffice)
 - *Rotornet* [63]: Oblivious cyclical reconfiguration w. selector switches [64] (Valiant load balancing [94])
 - *Tale of Two Topologies* [105]: Convert locally between Clos [24] topology and random graphs [87, 88]
- 2018 • – *DeepConf* [81]/*xWeaver* [102]: Machine learning approaches for topology reconfiguration
- 2019 • – Complexity classifications for weighted average path lengths in reconfigurable topologies [34, 35, 36]
 - *ReNet* [13] and *Push-Down-Trees* [9] providing statically and dynamically optimal reconfigurations
 - *DisplayNets* [75]: fully decentralized *SplayNets*
 - *Opera* [60]: Maintaining expander-based topologies under (oblivious) reconfiguration

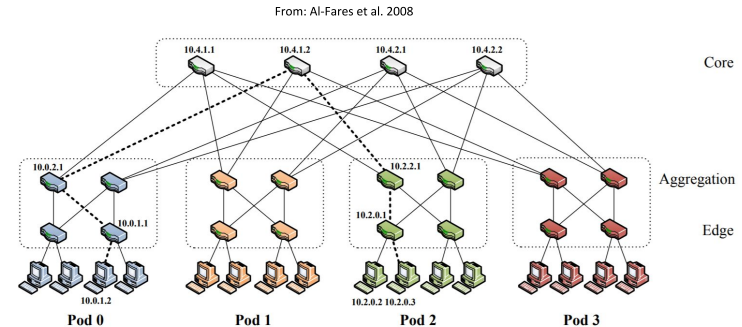
Today's Data Center Topologies

- Often *Clos*-based (e.g. *Fat-tree*)
 - Goal: optimize for all-to-all communication
 - Idea: Obtain good bisection bandwidth

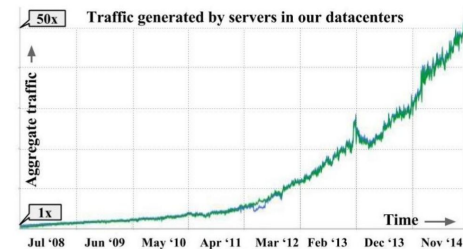


Today's Data Center Topologies

- Often *Clos*-based (e.g. *Fat-tree*)
 - Goal: optimize for all-to-all communication
 - Idea: Obtain good bisection bandwidth



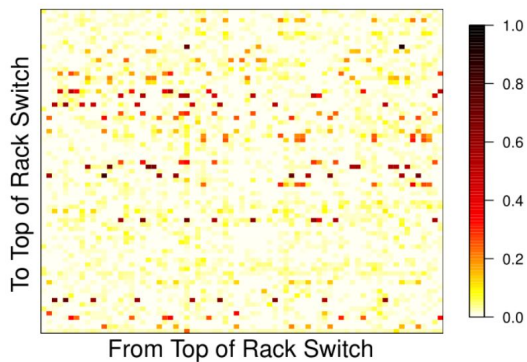
- However, traffic is growing at unprecedented rates
 - What can we do?
 - Exponentially bigger networks?



From Google's Datacenter Network. Singh et al., SIGCOMM'15

Data Center Traffic \neq Uniform

- However, DCN traffic is often **not** all-to-all

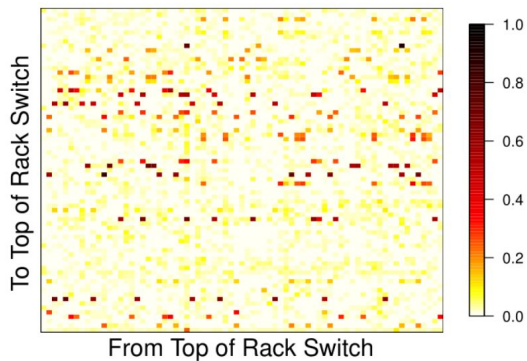


Traffic demands (normalized) between ToR switches.
Halperin et al., SIGCOMM'11

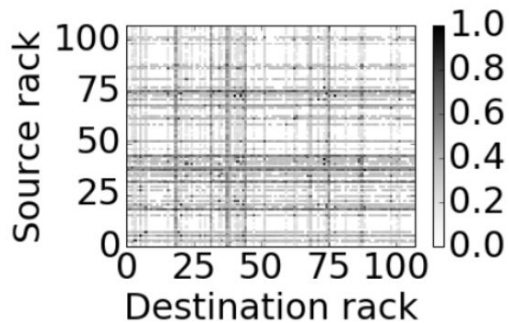
Data Center Traffic \neq Uniform

- However, DCN traffic is often **not** all-to-all

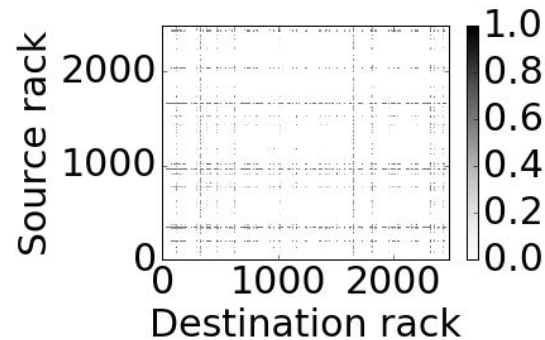
"Data reveal that 46-99% of the rack pairs exchange no traffic at all"



Traffic demands (normalized) between ToR switches.
Halperin et al., SIGCOMM'11

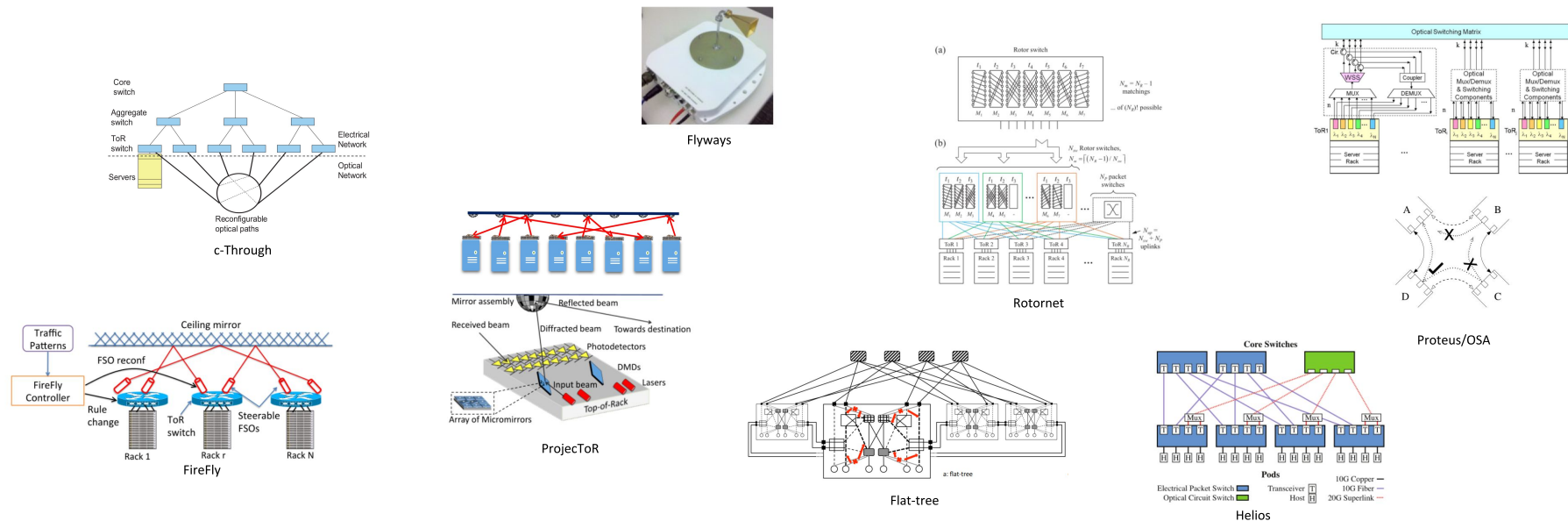


Heatmap of rack to rack traffic. Color intensity is log-scale and normalized.
Ghobadi et al., SIGCOMM'16



Enablers for Reconfigurable DCNs

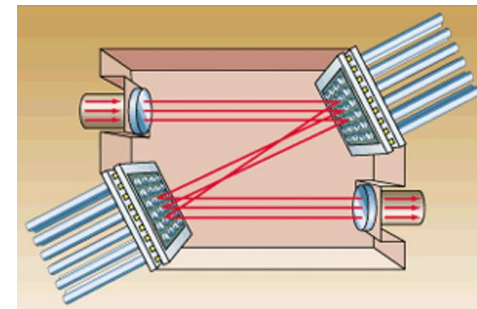
Programmable Physical Layer



Better power consumption, fan-out, “rate free”, at the cost of reconfiguration times & point-point connectivity

Overview of Technological Enablers

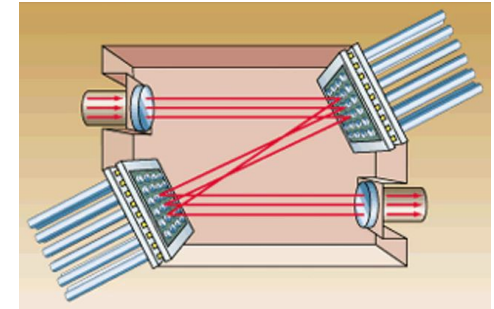
- Microelectromechanical systems (MEMS)
 - In 3D: $2N$ mirrors can connect N input to N output ports
 - In 2D: Less connectivity, but faster



<https://www.laserfocusworld.com/optics/article/16556781/many-approaches-taken-for-all-optical-switching> (Hecht, 2001)

Overview of Technological Enablers

- Microelectromechanical systems (MEMS)
 - In 3D: $2N$ mirrors can connect N input to N output ports
 - In 2D: Less connectivity, but faster



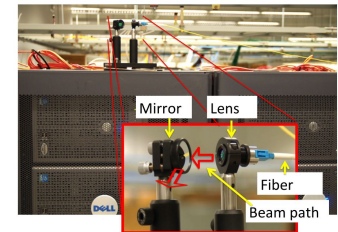
<https://www.laserfocusworld.com/optics/article/16556781/many-approaches-taken-for-alloptical-switching> (Hecht, 2001)

- (Beamformed) Wireless



Flyways

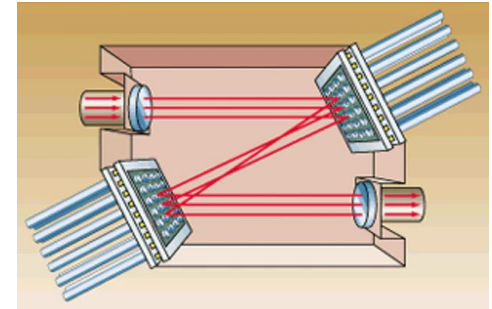
and Free-Space Optics



FireFly

Overview of Technological Enablers

- Microelectromechanical systems (MEMS)
 - In 3D: $2N$ mirrors can connect N input to N output ports
 - In 2D: Less connectivity, but faster



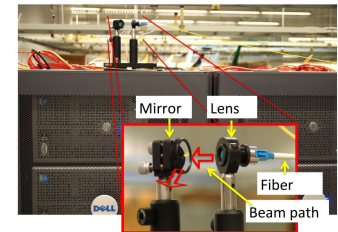
<https://www.laserfocusworld.com/optics/article/16556781/many-approaches-taken-for-alloptical-switching> (Hecht, 2001)

- (Beamformed) Wireless



Flyways

and Free-Space Optics



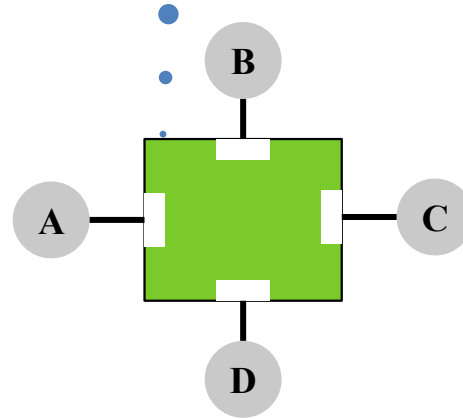
FireFly

- Connection medium can also often be shared, e.g., different wavelengths

It's a Match(ing)!

- Idea: Create “physical” connections

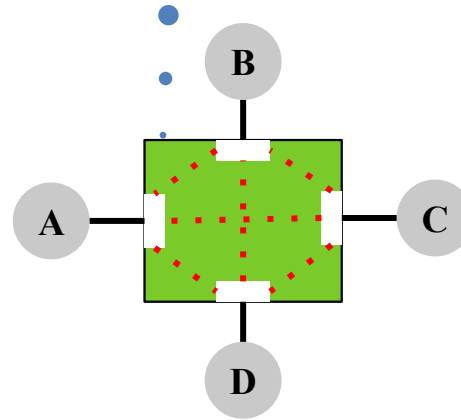
Reconfigurable Switch



It's a Match(ing)!

Reconfigurable Switch

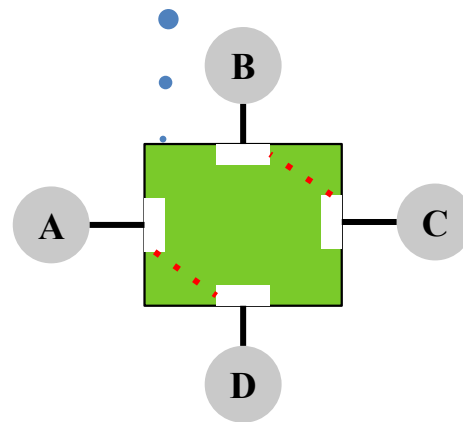
- Idea: Create “physical” connections
 - Difference: Not all-to-all switch
 - E.g. just 1 connection per node



It's a Match(ing)!

- Idea: Create “physical” connections
 - Difference: Not all-to-all switch
 - E.g. just 1 connection per node

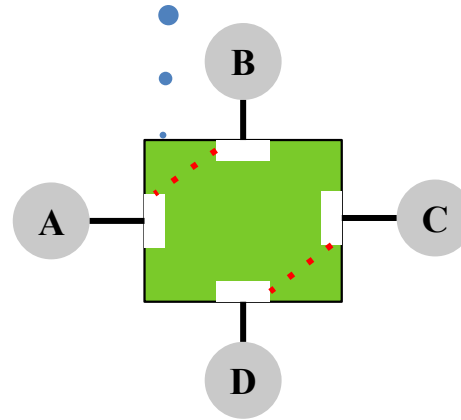
Reconfigurable Switch



It's a Match(ing)!

- Idea: Create “physical” connections
 - Difference: Not all-to-all switch
 - E.g. just 1 connection per node

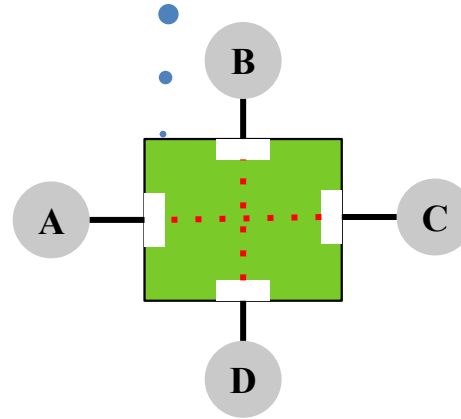
Reconfigurable Switch



It's a Match(ing)!

- Idea: Create “physical” connections
 - Difference: Not all-to-all switch
 - E.g. just 1 connection per node

Reconfigurable Switch



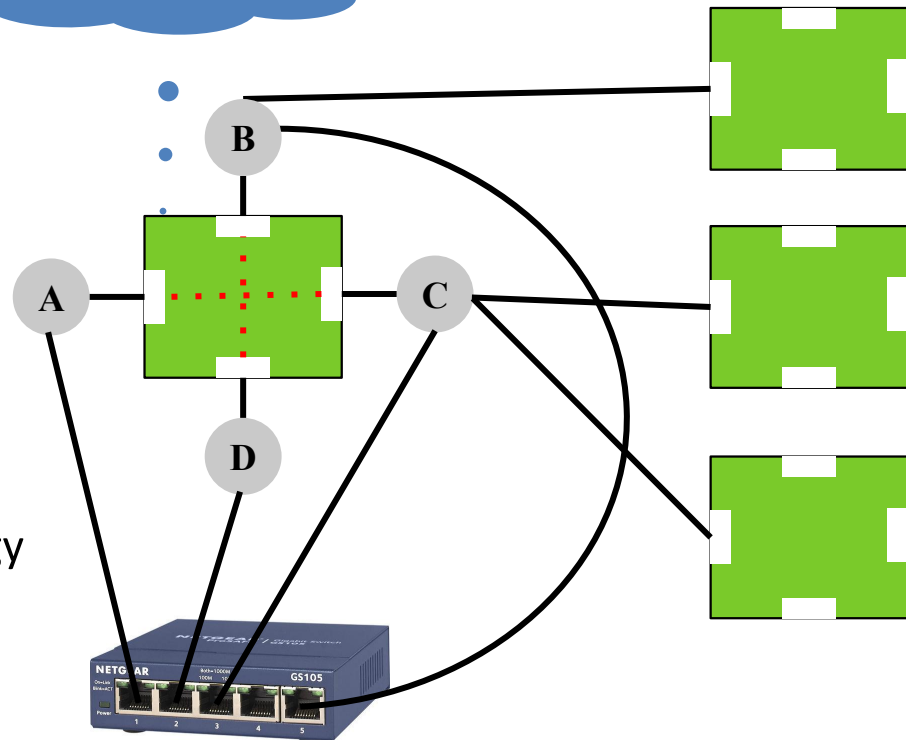
It's a Match(ing)!

- Idea: Create “physical” connections

- Difference: Not all-to-all switch
 - E.g. just 1 connection per node
 - Or many more than 1
 - Or separated sender/receiver

- Basic connectivity often by static topology
 - Hybrid: Static+Reconfigurable

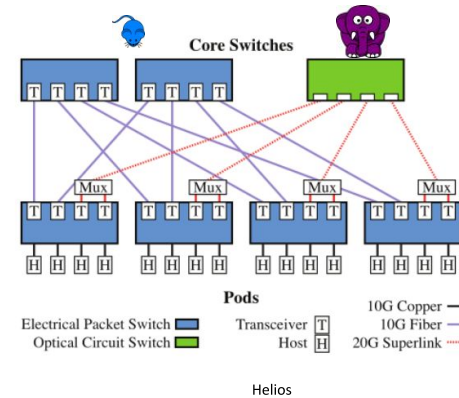
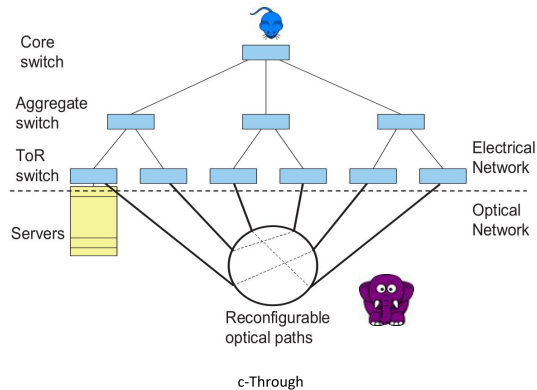
Reconfigurable Switch



Helios/c-Through

(Farrington et al. / Wang et al. @SIGCOMM 2010)

- Layout:
 - DCN-Topology + optical circuit switch (OCS, $\sim O(10)$ ms)



Helios/c-Through

(Farrington et al. / Wang et al. @SIGCOMM 2010)

- Algorithmic Idea:
 - Estimate demand between racks/pods
 - Pose as matching problem (throughput/volume per epoch as weights)
 - Unidirectional in Helios
 - Compute maximum matching (eg with Edmond's)



Helios/c-Through

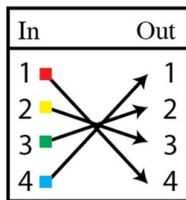
(Farrington et al. / Wang et al. @SIGCOMM 2010)

- Algorithmic Idea:
 - Estimate demand between racks/pods
 - Pose as matching problem (throughput/volume per epoch as weights)
 - Unidirectional in Helios
 - Compute maximum matching (eg with Edmond's)
- Example from Helios with 4 pods and link capacity of 4

Demand Matrix i

	1	2	3	4
1	0	1	1	3
2	7 ₍₄₎	0	3	1
3	1	3	0	9 ₍₄₎
4	3	2	1	0

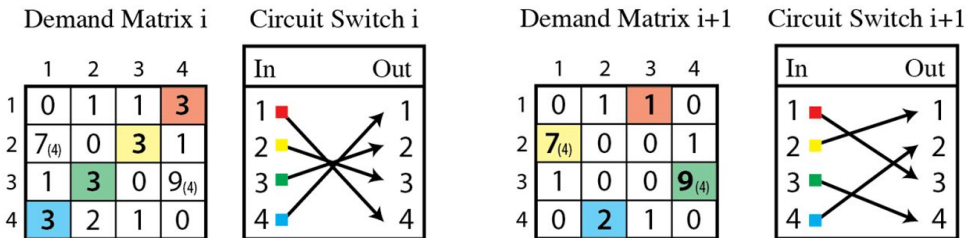
Circuit Switch i



Helios/c-Through

(Farrington et al. / Wang et al. @SIGCOMM 2010)

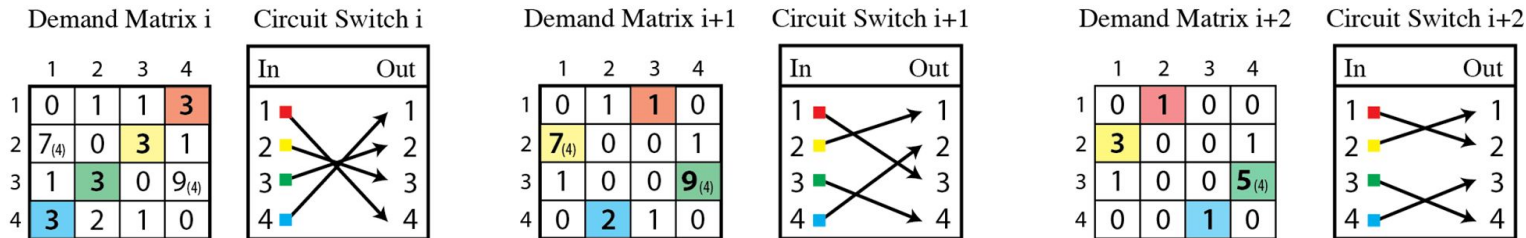
- Algorithmic Idea:
 - Estimate demand between racks/pods
 - Pose as matching problem (throughput/volume per epoch as weights)
 - Unidirectional in Helios
 - Compute maximum matching (eg with Edmond's)
- Example from Helios with 4 pods and link capacity of 4



Helios/c-Through

(Farrington et al. / Wang et al. @SIGCOMM 2010)

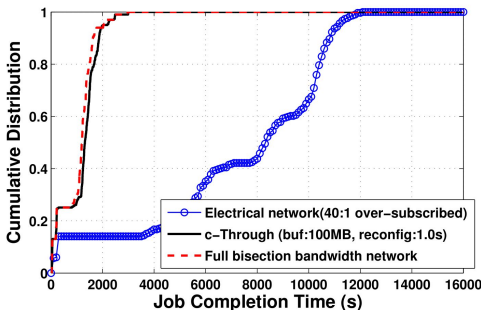
- Algorithmic Idea:
 - Estimate demand between racks/pods
 - Pose as matching problem (throughput/volume per epoch as weights)
 - Unidirectional in Helios
 - Compute maximum matching (eg with Edmond's)
- Example from Helios with 4 pods and link capacity of 4



Helios/c-Through

(Farrington et al. / Wang et al. @SIGCOMM 2010)

- Algorithmic Idea:
 - Estimate demand between racks/pods
 - Pose as matching problem (throughput/volume per epoch as weights)
 - Unidirectional in Helios
 - Compute maximum matching (eg with Edmond's)
- Performance example for data-intensive tasks:



c-Through: Hadoop Gridmix tasks

Routing Policy Restrictions

- So far: Routing options restricted to single-hop



Routing Policy Restrictions

- So far: Routing options restricted to single-hop



Routing Policy Restrictions

- So far: Routing options restricted to single-hop



Routing Policy Restrictions

- So far: Routing options restricted to single-hop



Routing Policy Restrictions

- So far: Routing options restricted to single-hop



Routing Policy Restrictions

- So far: Routing options restricted to single-hop



Routing Policy Restrictions

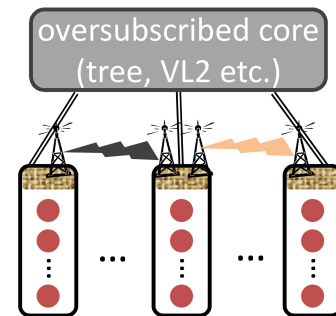
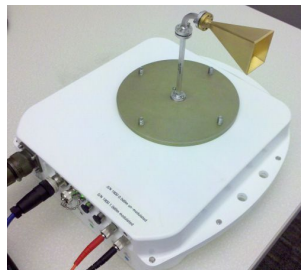
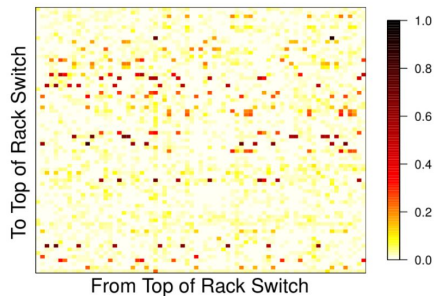
- So far: Routing options restricted to single-hop



Flyways

(Kandula et al., HotNets 2009/SIGCOMM 2011)

- Idea: Tackle hotspots by adding so-called flyways
 - Directional wireless (60GHz) [or also 802.11n/new static links]
 - $\sim O(10)$ ms (also propose the use of phased arrays, delay in microseconds)



Flyways

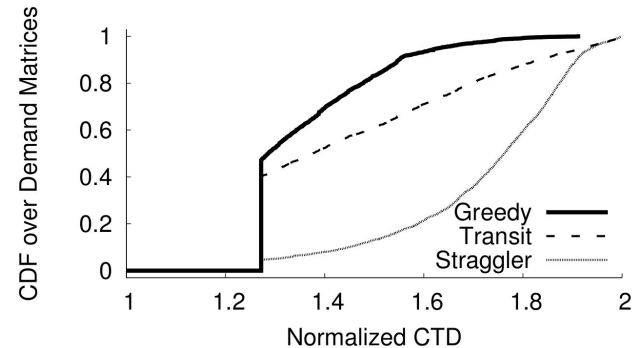
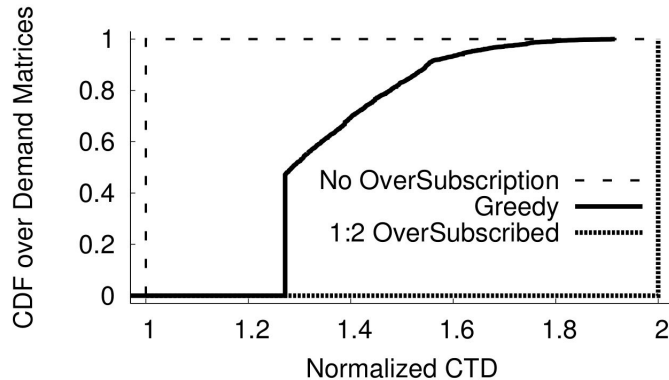
(Kandula et al., HotNets 2009/SIGCOMM 2011)

- Idea: Tackle hotspots by adding so-called flyways
 - Directional wireless (60GHz) [or also 802.11n/new static links]
- How to choose new links?
 - Three preliminary different strategies proposed w.r.t. completion time:
 - Optimization program
 - downside: intractable
 - Stragglers: help biggest demands on bottleneck links
 - downside: might not help much per bottleneck, fan-out/in of demands
 - Allow transit on straggler links
 - helps, but not optimized for it

Flyways

(Kandula et al., HotNets 2009/SIGCOMM 2011)

- Final strategy: **Greedy** with transit
 - Consider bottleneck link connected to ToR
 - Take flyway link that helps most w.r.t.
 - deviating traffic from bottleneck link + allowing transit



Beats other tractable strategies with one device per ToR

Images taken from the respective papers

Mirror Mirror on the Ceiling

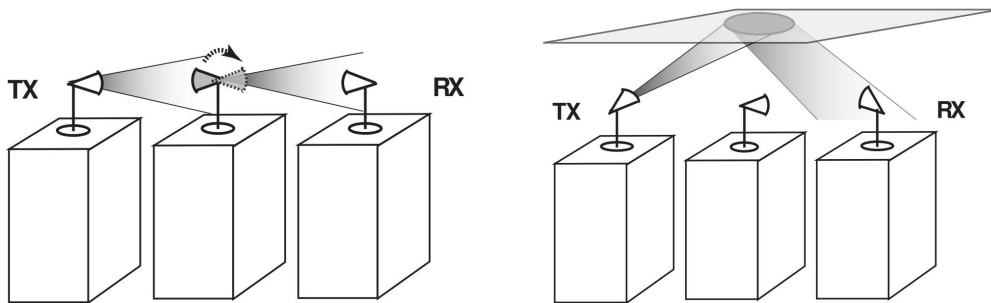
(Zhou et al., SIGCOMM 2012)

- Problem in Flyways: Limited any-to-any connections
 - Transmission might be blocked by obstacles (already 2.5mm is bad)
 - Radio interference between links
 - In combination limits many possible flyway configurations

Mirror Mirror on the Ceiling

(Zhou et al., SIGCOMM 2012)

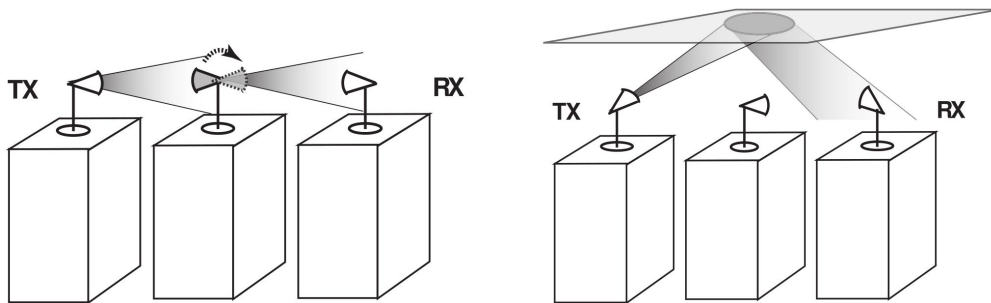
- Problem in Flyways: Limited any-to-any connections
 - Transmission might be blocked by obstacles (already 2.5mm is bad)
 - Radio interference between links
 - In combination limits many possible flyway configurations
- Approach from Zhou et al.: Go from 2D to 3D



Mirror Mirror on the Ceiling

(Zhou et al., SIGCOMM 2012)

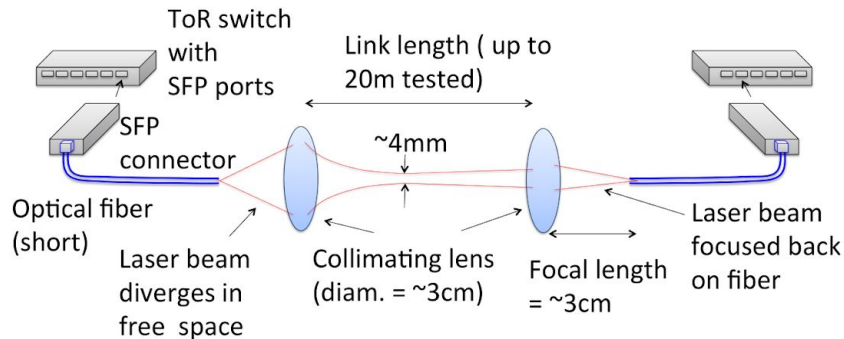
- Algorithmic/technical idea:
 - Leverage multiple radios per rack and multiple 60GHz frequency channels
 - Keep Signal-to-Interference-Noise-Ratio (SINR) in mind for conflicts
 - Greedily schedule requests ordered by conflict degrees
 - No preemption, no multi-hop



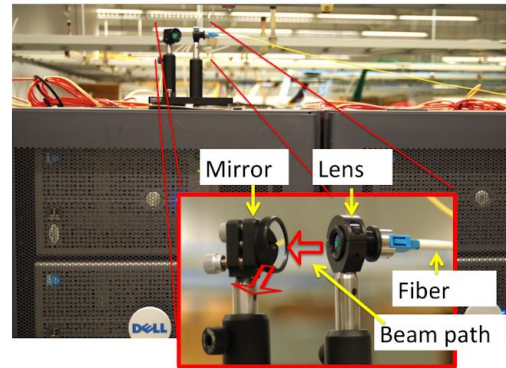
FireFly

(Hamedazimi et al., SIGCOMM 2012)

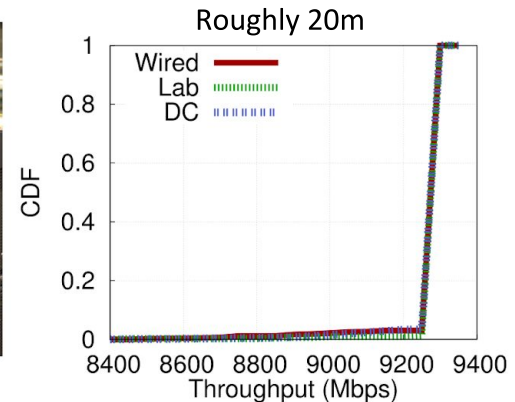
- Even beamformed wireless still suffers from interference/range
- Idea: Use Free-Space Optics (FSO)
 - $O(10)$ ms steering, $O(10)$ fan-out



(a) FSO link design



(b) DC set up

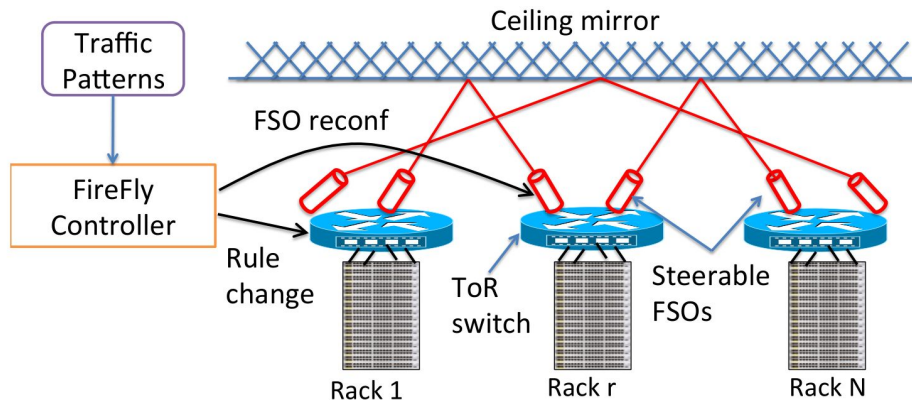


(c) TCP throughput

FireFly

(Hamedazimi et al., SIGCOMM 2012)

- Bring concept from *Mirror, Mirror* to FSO
- Core algorithmic idea:
 - Periodically recompute topology for throughput (optim. formulation)
 - Greedily augment current matching to shorten routes for eg new 🐜
 - (don't disconnect)



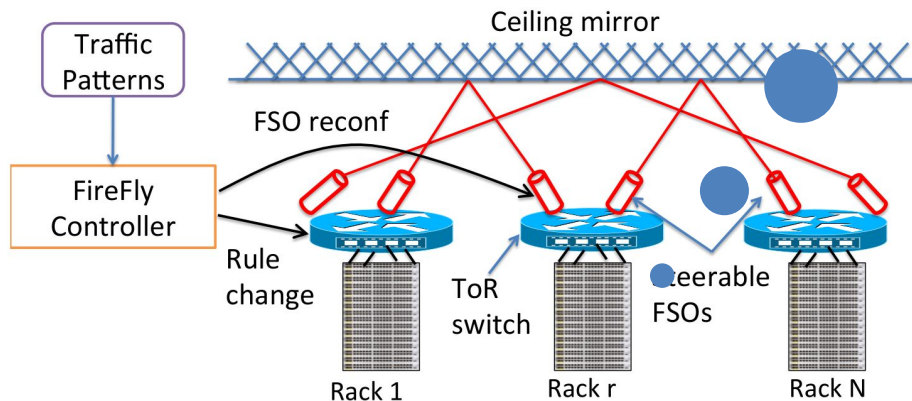
FireFly

(Hamedazimi et al., SIGCOMM 2012)

- Bring concept from *Mirror, Mirror* to
- Core algorithmic idea:
 - Periodically recompute topology for thro
 - Greedily augment current matching to sh
 - (don't disconnect)

More on FSO later in ProjectToR

- Distributed algorithm
- Microsecond reconfigurations
 - Programmable mirrors
- $O(1000)$ fan-out



Proteus/OSA

(Singla et al., HotNets 2010/NSDI 2012/ ToN 2014)

- More thoughts on all-reconfigurable:
 - What if traffic is extremely dynamic?
 - Prior studies show: Usually somewhat stable for many DCN applications
 - Recall:
 - Need multi-hop connections

Proteus/OSA

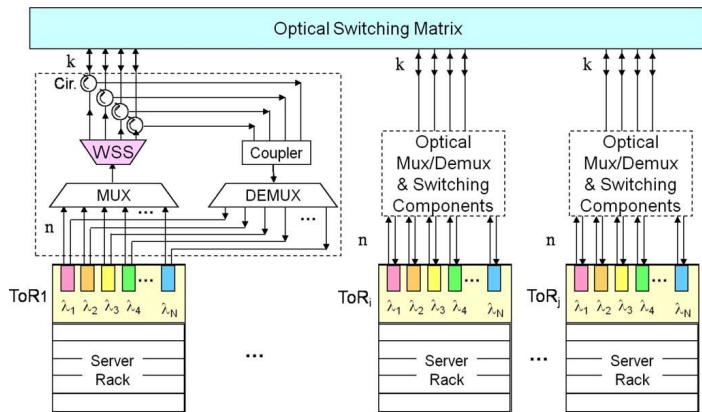
(Singla et al., HotNets 2010/NSDI 2012/ ToN 2014)

- More thoughts on all-reconfigurable:
 - What if traffic is extremely dynamic?
 - Prior studies show: Usually somewhat stable for many DCN applications
 - Recall:
 - Need multi-hop connections
- Multi-hop and single connection to OCS?
 - Not so good in all-reconfigurable setting
 - No connectivity (even with 2: requires Hamiltonian Cycle)
 - No scaling of link capacities

Proteus/OSA

(Singla et al., HotNets 2010/NSDI 2012/ ToN 2014)

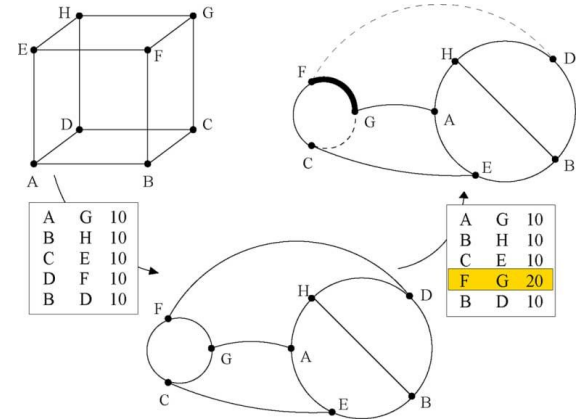
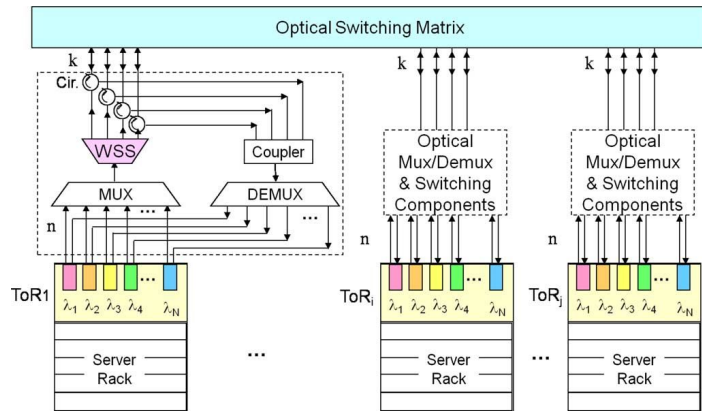
- Creating connectivity:
 - Multiple connections per rack



Proteus/OSA

(Singla et al., HotNets 2010/NSDI 2012/ ToN 2014)

- Creating connectivity:
 - Multiple connections per rack
- Scaling link capacities
 - Allow “parallel” links



Proteus/OSA

(Singla et al., HotNets 2010/NSDI 2012/ ToN 2014)

- Algorithmic idea:
 - Optimize for 🐙, make 🌐 feasible

Proteus/OSA

(Singla et al., HotNets 2010/NSDI 2012/ ToN 2014)

- Algorithmic idea:

- Optimize for 🧟, make 🌐 feasible

1. 🧟 For b connections per rack, leverage b -matching algorithms
 - a. Estimated demands as weights (Poly-time solvable, efficient heuristics exist)

Proteus/OSA

(Singla et al., HotNets 2010/NSDI 2012/ ToN 2014)

- Algorithmic idea:
 - Optimize for 🧑‍🔬, make 🌐 feasible
- 1. 🧑‍🔬 For b connections per rack, leverage b -matching algorithms
 - a. Estimated demands as weights (Poly-time solvable, efficient heuristics exist)
- 2. 🌐 Use link-exchanges to make topology connected
 - a. Heuristic: Connect 2 components by removing low weight links

Proteus/OSA

(Singla et al., HotNets 2010/NSDI 2012/ ToN 2014)

- Algorithmic idea:
 - Optimize for 🦹, make 🌐 feasible
- 1. 🦹 For b connections per rack, leverage b -matching algorithms
 - a. Estimated demands as weights (Poly-time solvable, efficient heuristics exist)
- 2. 🌐 Use link-exchanges to make topology connected
 - a. Heuristic: Connect 2 components by removing low weight links
- 3. Deploy routing (eg shortest paths)

Proteus/OSA

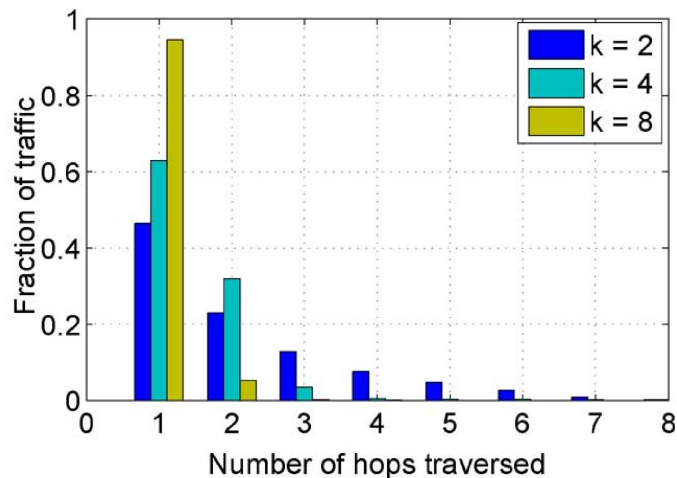
(Singla et al., HotNets 2010/NSDI 2012/ ToN 2014)

- Algorithmic idea:
 - Optimize for 🧟, make 🌐 feasible
- 1. 🧟 For b connections per rack, leverage b -matching algorithms
 - a. Estimated demands as weights (Poly-time solvable, efficient heuristics exist)
- 2. 🌐 Use link-exchanges to make topology connected
 - a. Heuristic: Connect 2 components by removing low weight links
- 3. Deploy routing (eg shortest paths)
- 4. 🧟 Assign wavelengths along links to distribute capacity
 - a. Each link at least one wavelength color, no color connected twice to a node
 - b. Solved by link-coloring on multigraphs (efficient heuristics exist)

Proteus/OSA

(Singla et al., HotNets 2010/NSDI 2012/ ToN 2014)

- Impact of degree on hop-counts (OSA uses $k=4$ in most evaluations)
- Using Mapreduce-demands with 80 ToRs



Average Path Length

(Foerster et al., ANCS 2018, SIGCOMM CCR 2019, Networking 2019)

- So far: Optimizing for throughput
 - Recall last slide: short paths are desired
 - Also used in FireFly for new elephants
 - Respectively: “**bandwidth tax**” (Mellette et al., 2019)

Average Path Length

(Foerster et al., ANCS 2018, SIGCOMM CCR 2019, Networking 2019)

- So far: Optimizing for throughput
 - Recall last slide: short paths are desired
 - Also used in FireFly for new elephants
 - Respectively: “**bandwidth tax**” (Mellette et al., 2019)
- Different objective:
 - **Minimize** (weighted) **average path length**
 - Popular in many fields, e.g., OSPF

Average Path Length

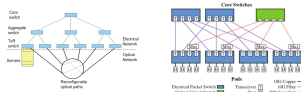
(Foerster et al., ANCS 2018, SIGCOMM CCR 2019, Networking 2019)

- So far: Optimizing for throughput
 - Recall last slide: short paths are desired
 - Also used in FireFly for new elephants
 - Respectively: “**bandwidth tax**” (Mellette et al., 2019)
- Different objective:
 - **Minimize** (weighted) **average path length**
 - Popular in many fields, e.g., OSPF
- How difficult from an algorithmic perspective?

Average Path Length

(Foerster et al., ANCS 2018, SIGCOMM CCR 2019, Networking 2019)

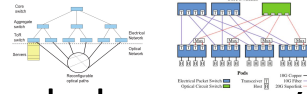
- In model from Helios/c-Through?
 - Recall: packet switched network XOR single-hop reconfigurable (eg OCS)



Average Path Length

(Foerster et al., ANCS 2018, SIGCOMM CCR 2019, Networking 2019)

- In model from Helios/c-Through?
 - Recall: packet switched network XOR single-hop reconfigurable (eg OCS)

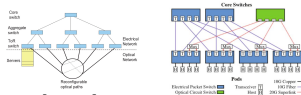


- Good algorithmic news:
 - Efficiently polynomial-time solvable (weighted matching algorithms)
 - Also for many connections per node to the OCS

Average Path Length

(Foerster et al., ANCS 2018, SIGCOMM CCR 2019, Networking 2019)

- In model from Helios/c-Through?
 - Recall: packet switched network XOR single-hop reconfigurable (eg OCS)



- Good algorithmic news:
 - Efficiently polynomial-time solvable (weighted matching algorithms)
 - Also for many connections per node to the OCS



- Bad algorithmic news:
 - Such a restriction is self-imposed and hurts performance

Average Path Length


(Foerster et al., ANCS 2018, SIGCOMM CCR 2019, Networking 2019)




- How hard to be optimal after lifting restrictions?
 - Already “simple” settings are NP-hard
 - E.g., each route may use at *most one reconfigurable link*

Average Path Length

(Foerster et al., ANCS 2018, SIGCOMM CCR 2019, Networking 2019)

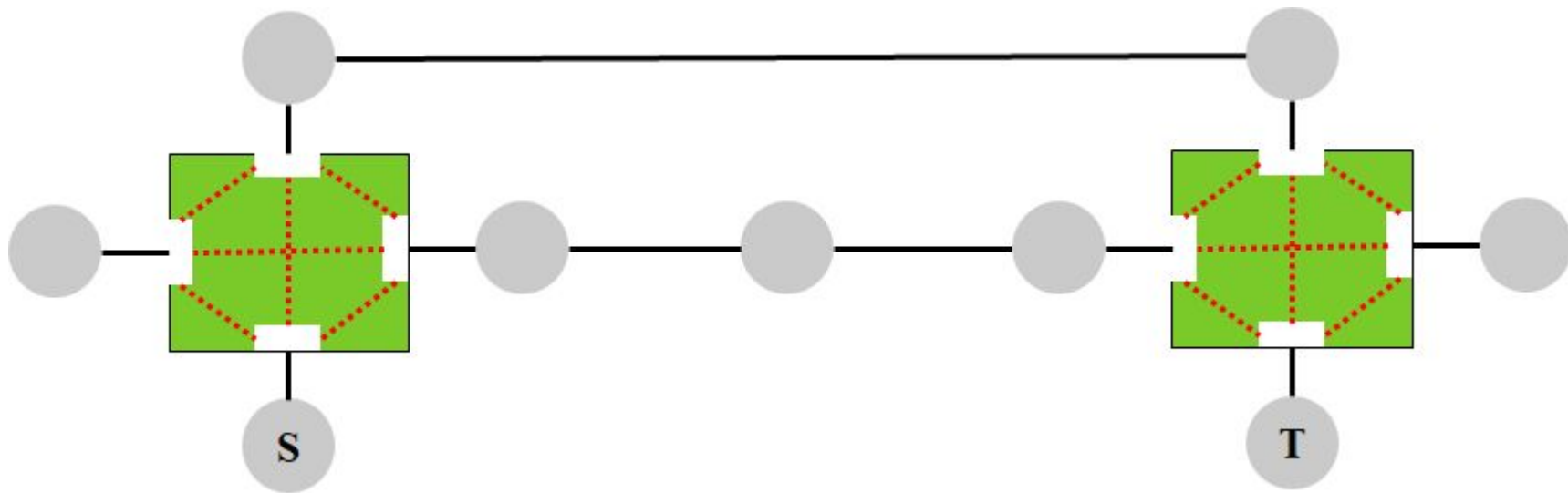
-  How hard to be optimal after lifting restrictions?
 - Already “simple” settings are NP-hard
 - E.g., each route may use at *most one reconfigurable link*

-  But: We can lift Dijkstra’s algorithm into this setting*
 - Each reconfigurable switch traversed at most once per flow



Average Path Length

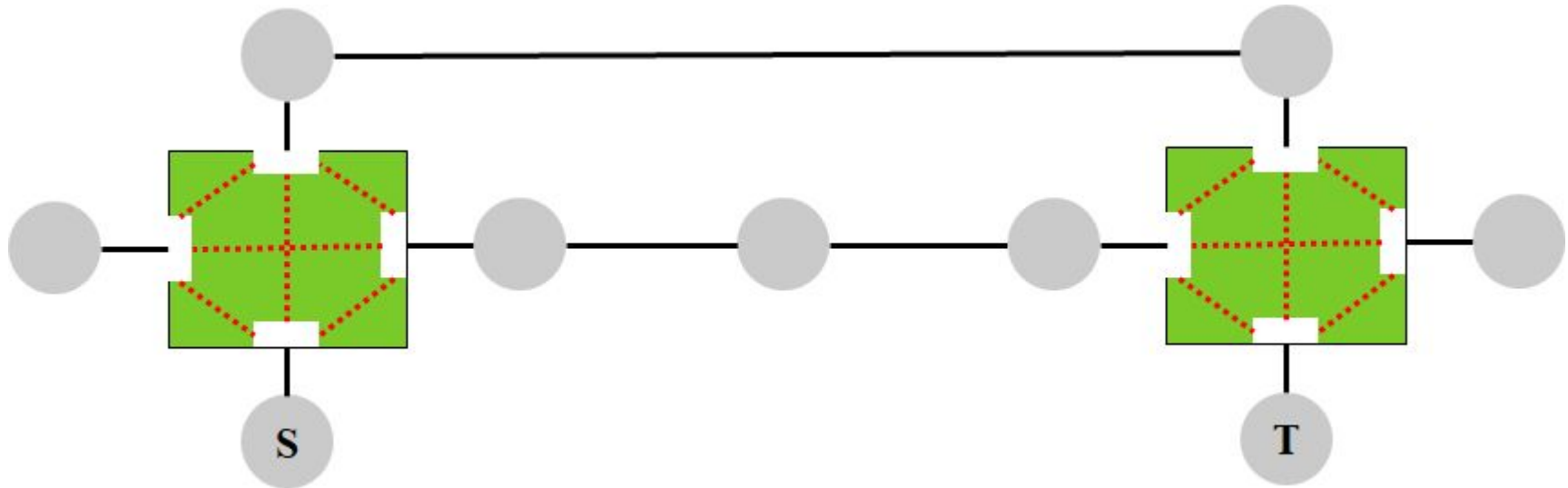
(Foerster et al., ANCS 2018, SIGCOMM CCR 2019, Networking 2019)



Average Path Length

(Foerster et al., ANCS 2018, SIGCOMM CCR 2019, Networking 2019)

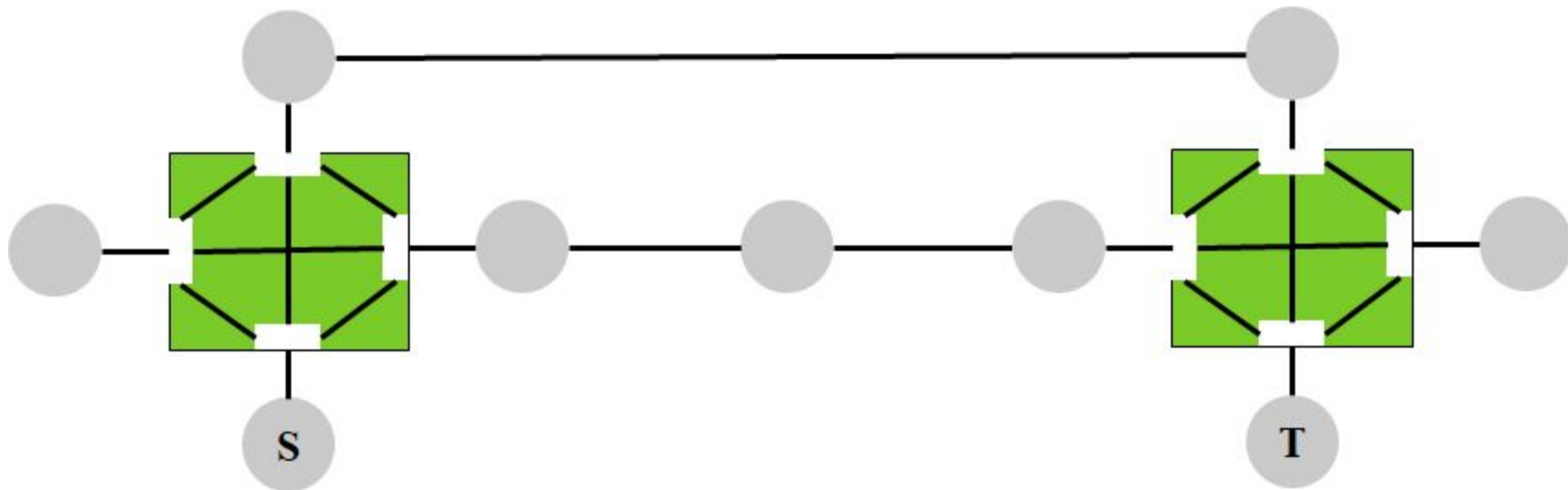
1. Add all still possible reconfigurable links as static links



Average Path Length

(Foerster et al., ANCS 2018, SIGCOMM CCR 2019, Networking 2019)

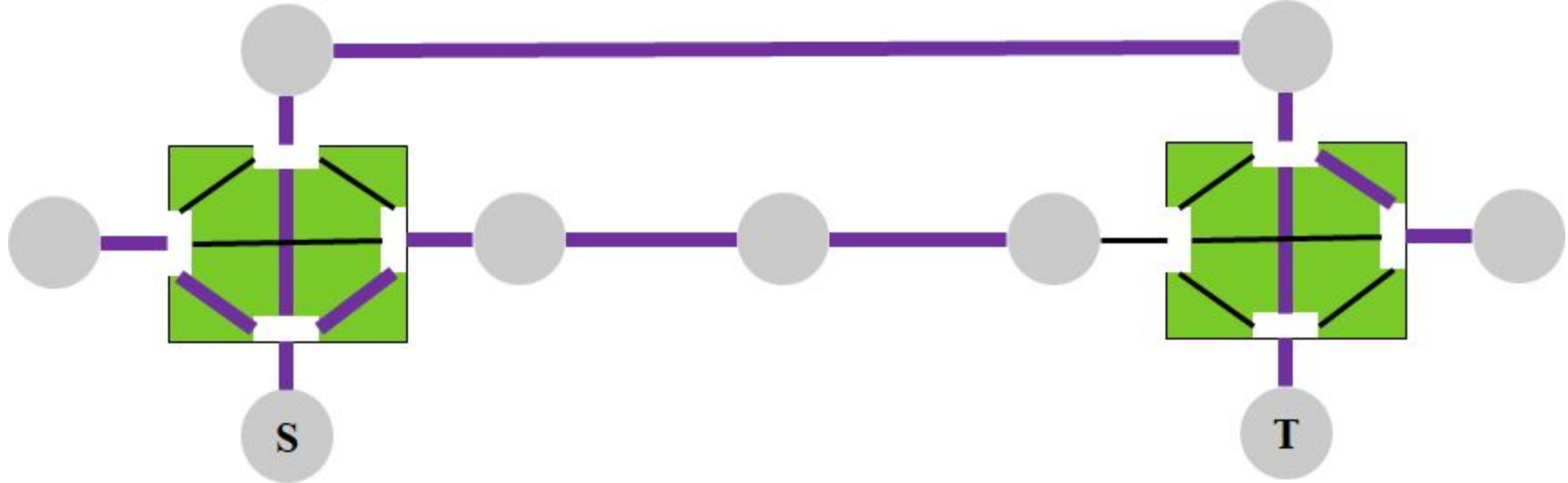
1. Add all still possible reconfigurable links as static links



Average Path Length

(Foerster et al., ANCS 2018, SIGCOMM CCR 2019, Networking 2019)

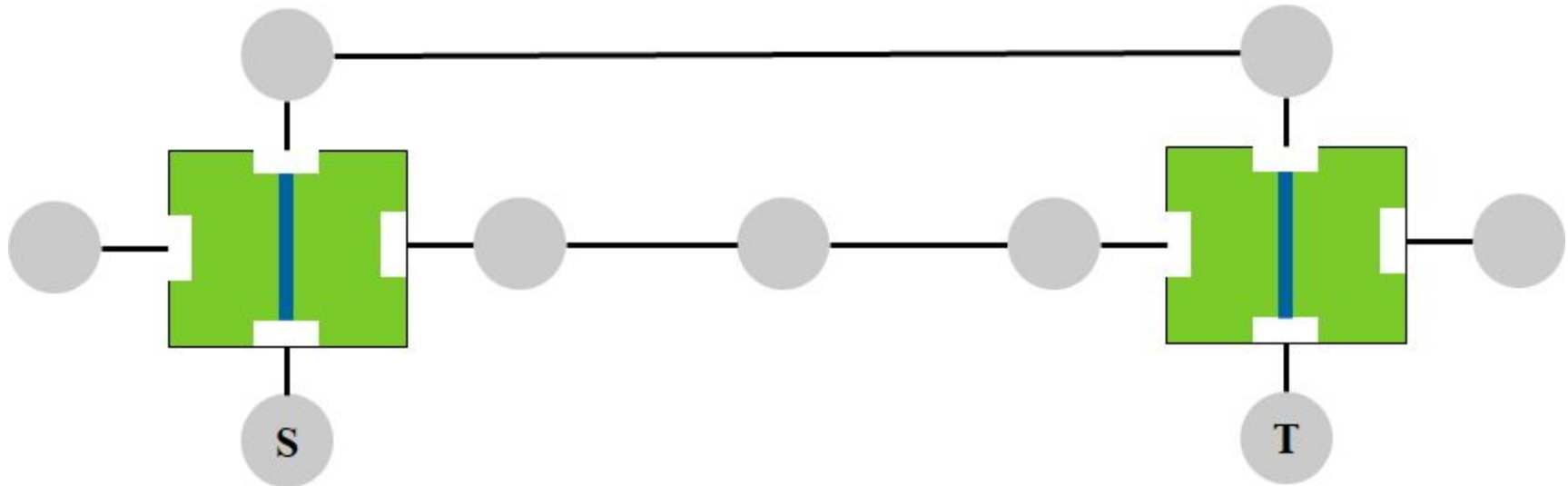
1. Add all still possible reconfigurable links as static links
2. **Run standard Dijkstra from source S**



Average Path Length

(Foerster et al., ANCS 2018, SIGCOMM CCR 2019, Networking 2019)

1. Add all still possible reconfigurable links as static links
2. Run standard Dijkstra from source S
3. **Add newly used links on shortest path to T to the matchings**

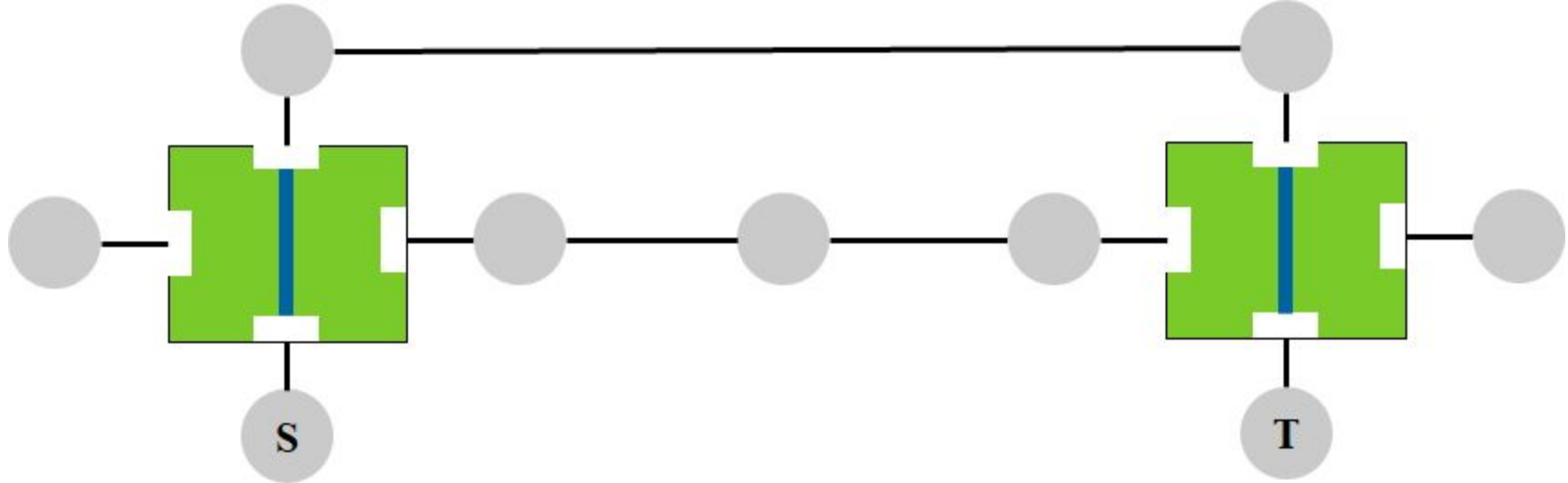


Average Path Length

(Foerster et al., ANCS 2018, SIGCOMM CCR 2019, Networking 2019)

1. Add all still possible reconfigurable links as static links
2. Run standard Dijkstra from source S
3. Add newly used links on shortest path to T to the matchings

Also works if some
matching links already exist

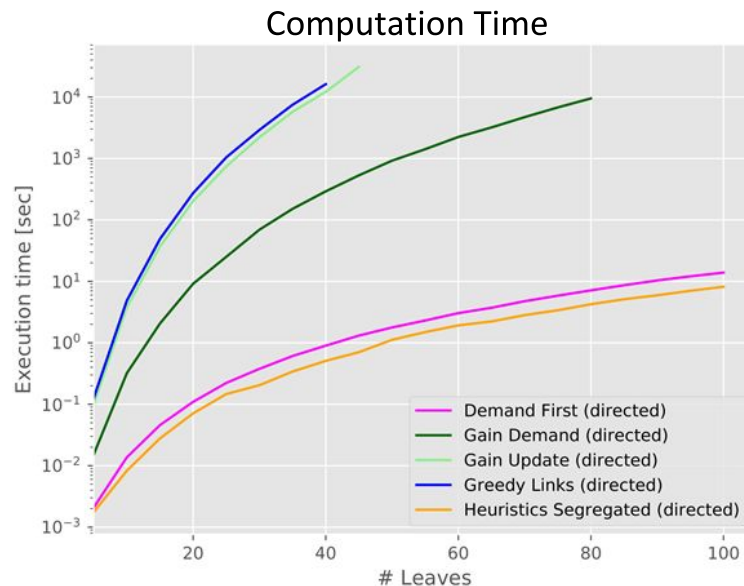
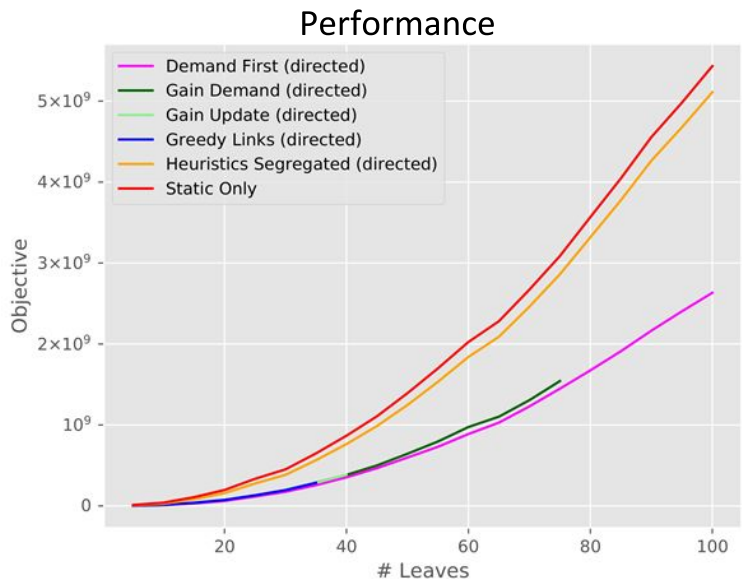


Average Path Length

(Foerster et al., ANCS 2018, SIGCOMM CCR 2019, Networking 2019)

- Leverage for greedy heuristics (eg greedily run Dijkstra:)

 single hop matching baseline (optimal w.r.t. restricted segregated routing)



Traffic from recent facebook dataset, in fat-tree + OCS setting

Mordia

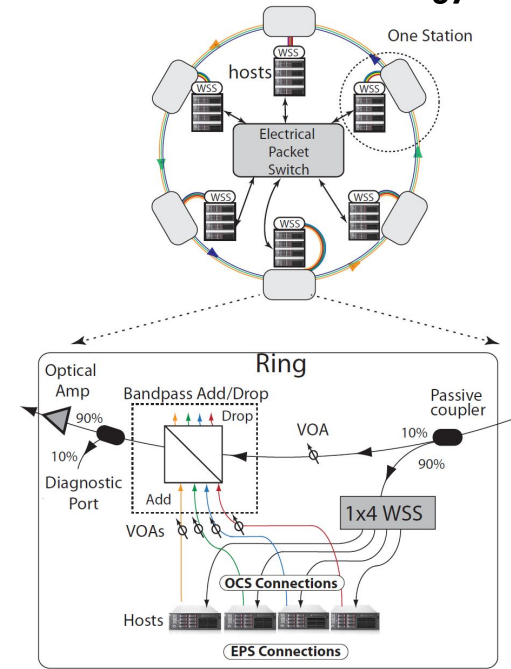
(Porter et al., SIGCOMM 2014)

- Back to the throughput objective

Mordia

(Porter et al., SIGCOMM 2014)

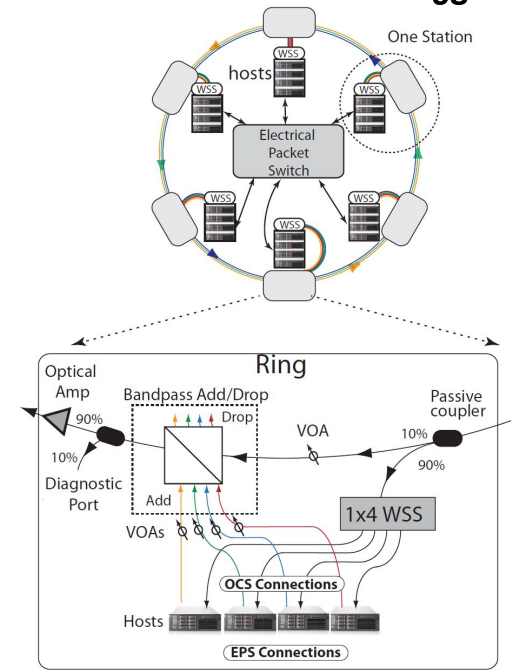
- Back to the throughput objective
- What if reconfiguration time goes
 - from milliseconds
 - to microseconds?



Mordia

(Porter et al., SIGCOMM 2014)

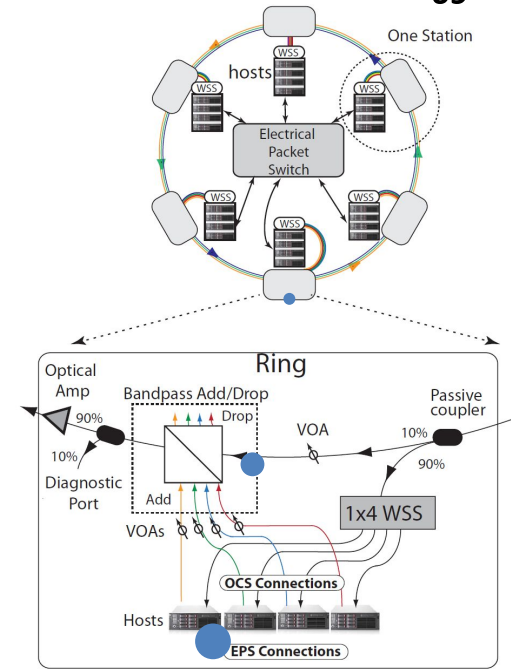
- Back to the throughput objective
- What if reconfiguration time goes
 - from milliseconds
 - to microseconds?
- Can't really compute well at microsecond scale?



Mordia

(Porter et al., SIGCOMM 2014)

- Back to the throughput objective
- What if reconfiguration time goes
 - from milliseconds
 - to microseconds?
- Can't really compute well at microsecond scale?



Extended to multi-ring in
Megaswitch (Chen et al., NSDI 2017)

Mordia

(Porter et al., SIGCOMM 2014)

- Key idea:
 - Instead of a single reconfiguration...
 - ... compute a traffic matrix schedule (TMS)!

Schedule

Detailed schedules are to be updated.

Go to schedule for: [Monday](#), [Tuesday](#), [Wednesday](#), [Thursday](#), [Friday](#)

Download the [conference program in PDF format \(TBA\)](#).

Monday, June 24th, 2019 (Tutorials)

Time	Tutorials		
08:15	Continental Breakfast		
09:00	Reconfigurable Networks: Enablers, Algorithms, Complexity (ReNets)		
09:30	<table border="0"> <tr> <td>Ramakrishnan Durairajan, Klaus-T. Foerster, and Stefan Schmid (Room 104A)</td> <td>The Power of SOAP Scheduling Mor Harchol-Balter and Ziv Scully (Room 104B)</td> </tr> </table>	Ramakrishnan Durairajan, Klaus-T. Foerster, and Stefan Schmid (Room 104A)	The Power of SOAP Scheduling Mor Harchol-Balter and Ziv Scully (Room 104B)
Ramakrishnan Durairajan, Klaus-T. Foerster, and Stefan Schmid (Room 104A)	The Power of SOAP Scheduling Mor Harchol-Balter and Ziv Scully (Room 104B)		
10:00			
10:30			
11:00	Break		
11:20	FCRC Keynote:		
12:00	James E. Smith (Abstract)		
12:30			
13:00	Lunch		
13:30			
14:00	Two-Sided Marketplaces: An Algorithmic Viewpoint		
14:30	Sid Bannerjee and Yang Cai		

Mordia

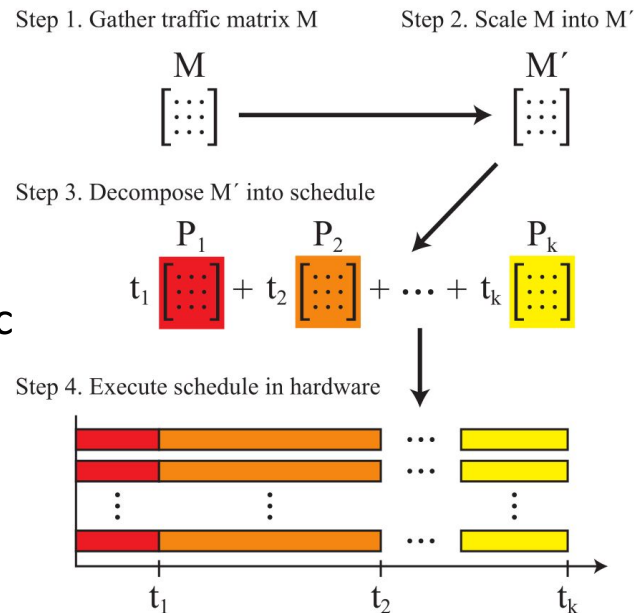
(Porter et al., SIGCOMM 2014)

- Traffic Matrix Scheduling
 - Look at all possible matchings
 - How much time to spend in each?

Mordia

(Porter et al., SIGCOMM 2014)

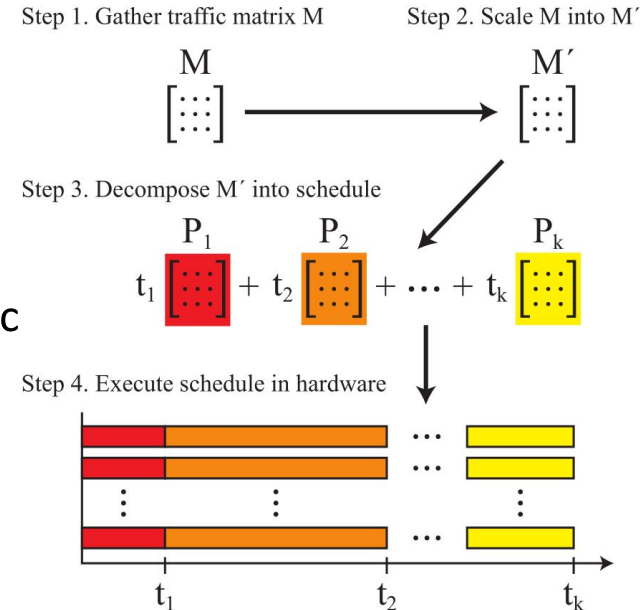
- Traffic Matrix Scheduling
 - Look at all possible matchings
 - How much time to spend in each?
- Idea: Single-hop, ideally fully utilize all links
 - Scale matrix s.t. is admissible & doubly-stochastic
 - Use Sinkhorn's algorithm, then
 - decompose with Birkhoff von Neumann decomposition ($\sim O(n^2)$ runtime)



Mordia

(Porter et al., SIGCOMM 2014)

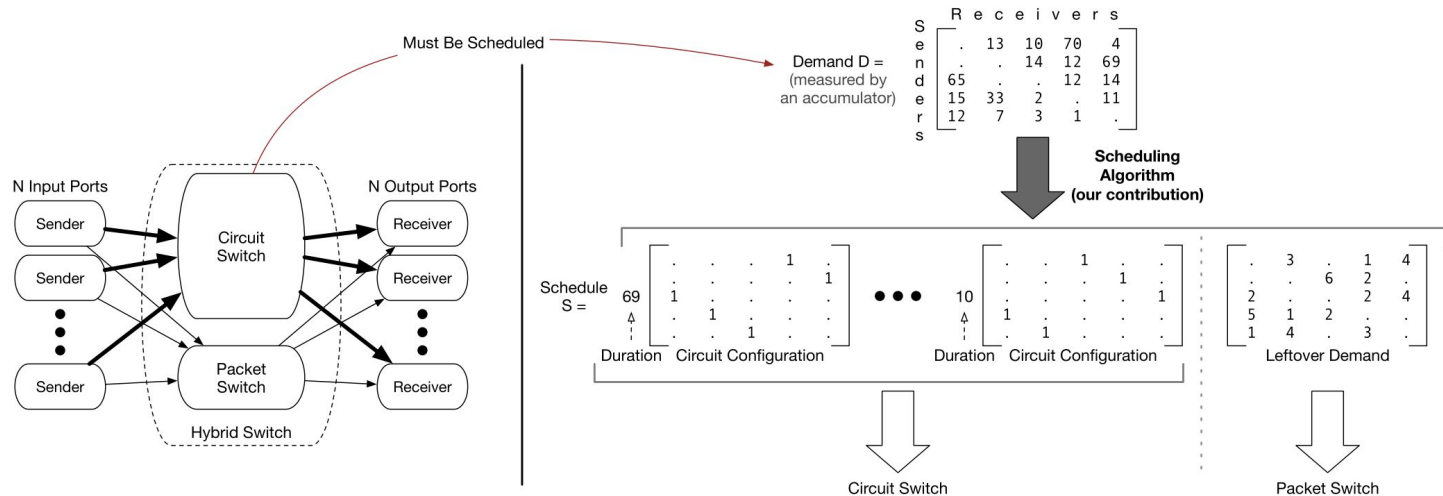
- Traffic Matrix Scheduling
 - Look at all possible matchings
 - How much time to spend in each?
- Idea: Single-hop, ideally fully utilize all links
 - Scale matrix s.t. is admissible & doubly-stochastic
 - Use Sinkhorn's algorithm, then
 - decompose with Birkhoff von Neumann decomposition ($\sim O(n^2)$ runtime)
- Problem: Some slots extremely brief, up to $O(n^2)$ many
 - Approximate by longest first, recompute after cut-off, tail into static network



Solstice

(Liu et al., CoNEXT 2015)

- So far: TMS does not take static network into account
 - Also not the reconfiguration time



Network Model

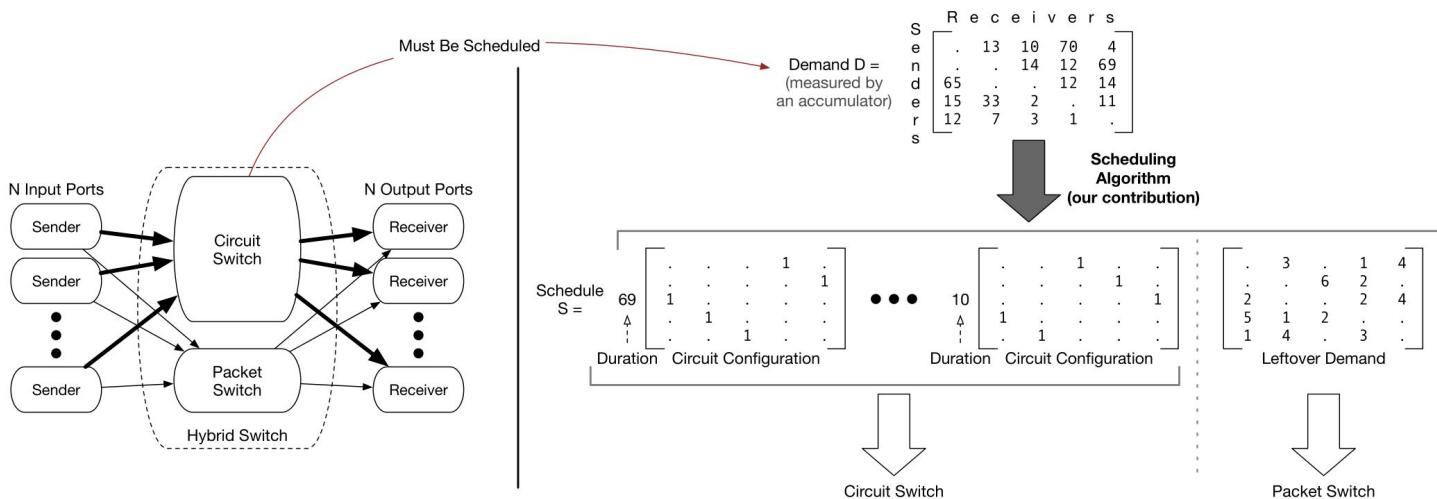
Images taken from the respective papers

Scheduling Overview

Solstice

(Liu et al., CoNEXT 2015)

- Rough heuristic idea: Greedily schedule perfect matchings
 - Maximize minimum element in matching, repeat (outperforms BvN up to 2.9x)
 - About 14% away from optimal utilization, but runtime in $\sim O(n^3)$



Network Model

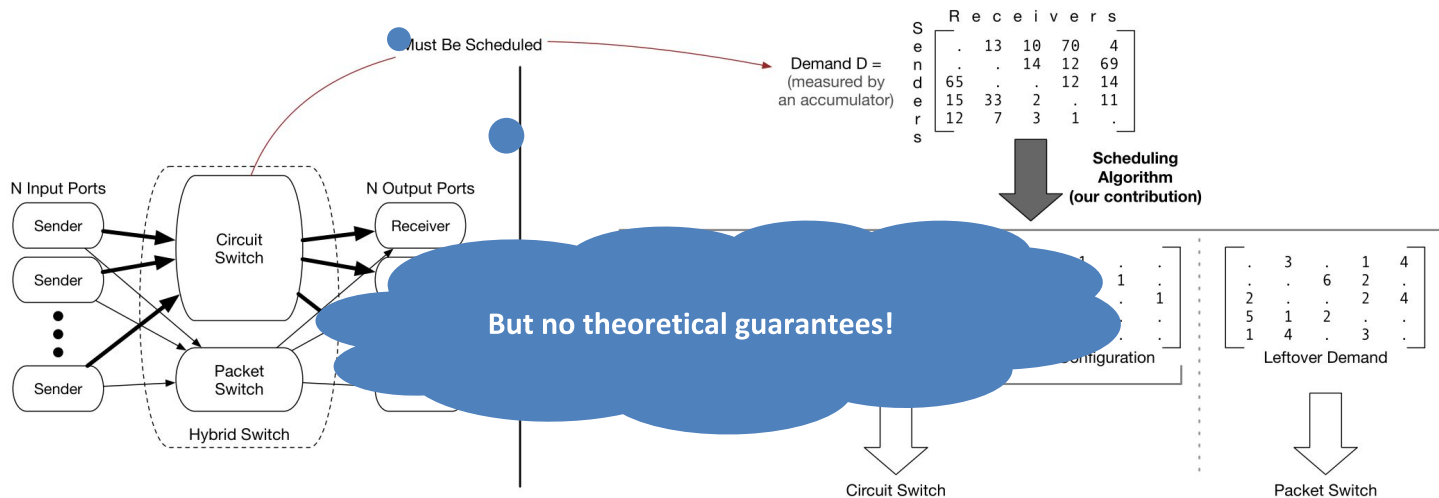
Images taken from the respective papers

Scheduling Overview

Solstice

(Liu et al., CoNEXT 2015)

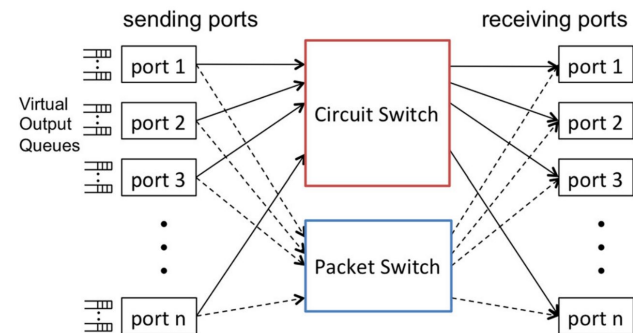
- Rough heuristic idea: Greedily schedule perfect matchings
 - Maximize minimum element in matching, repeat (outperforms BvN up to 2.9x)
 - About 14% away from optimal utilization, but runtime in $\sim O(n^3)$



Eclipse

(Venkatakrishnan et al., SIGMETRICS 2016 / Queuing Syst. 2018)

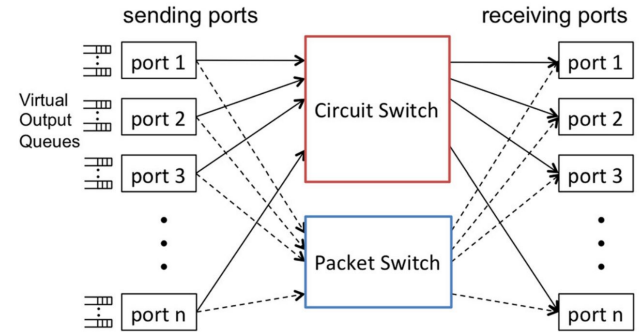
- Provides theoretical guarantees for TMS
 - hybrid switch model, reconfig. delay δ



Eclipse

(Venkatakrishnan et al., SIGMETRICS 2016 / Queuing Syst. 2018)

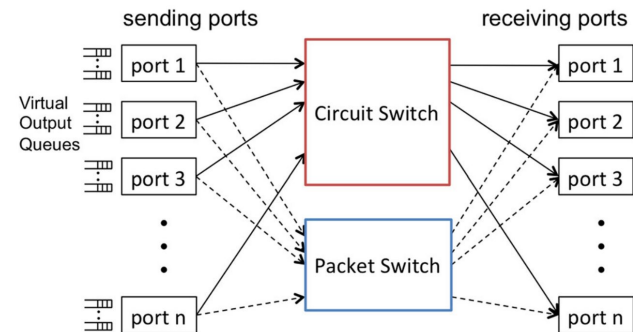
- Provides theoretical guarantees for TMS
 - hybrid switch model, reconfig. delay δ
- Problem has submodular structure
 - Allows for $(1-1/e) \approx 0.63$ approximation via thresholding/max. weight matchings



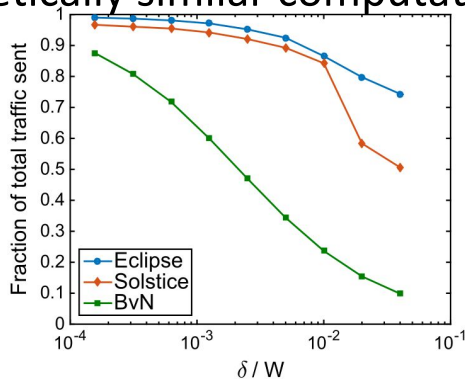
Eclipse

(Venkatakrisnan et al., SIGMETRICS 2016 / Queuing Syst. 2018)

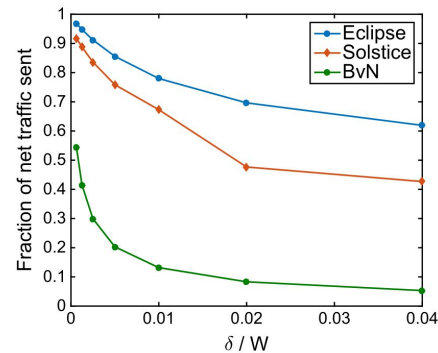
- Provides theoretical guarantees for TMS
 - hybrid switch model, reconfig. delay δ
- Problem has submodular structure
 - Allows for $(1-1/e) \approx 0.63$ approximation via thresholding/max. weight matchings
- Theoretically similar computation time to Solstice, performance for window W :



Sparse
Skewed
100 ports



Different sparsity and skew
in submatrices
200 ports

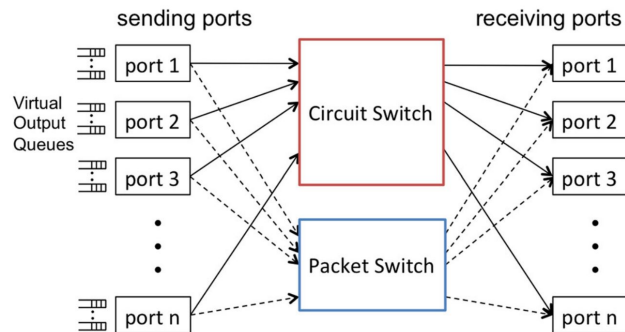


But no theoretical guarantees for multi-hop routing!

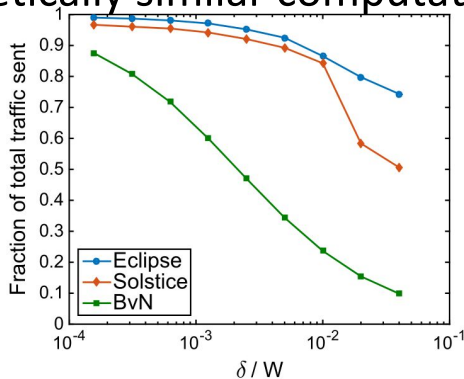
Eclipse

(Chen et al., SIGMETRICS 2016 / Queuing Syst. 2018)

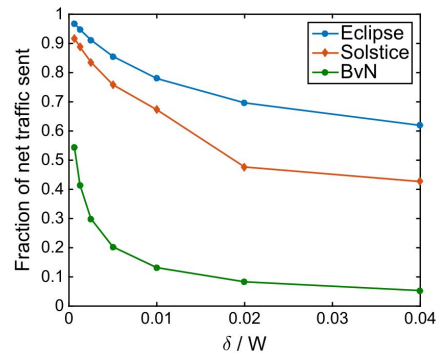
- Provides theoretical guarantees for TMS
 - hybrid switch model, reconfig. delay δ
- Problem has submodular structure.
 - Allows for $(1-1/e) \approx 0.63$ approximation via thresholding/max. weight matchings
- Theoretically similar computation time to Solstice, performance for window W :



Sparse
Skewed
100 ports




Different sparsity and skew
in submatrices
200 ports




Computation Times

- Trade-off: Computation time and efficiency
 - How to scale to larger networks?
 - Especially with micro-/nano-second switching times?

Computation Times



- Trade-off: Computation time and efficiency
 - How to scale to larger networks?
 - Especially with micro-/nano-second switching times?
- Algorithmic idea #1: **Distributed Control Plane**
 - E.g. ProjectToR (Ghobadi et al., 2016) 
 - Next part of the tutorial

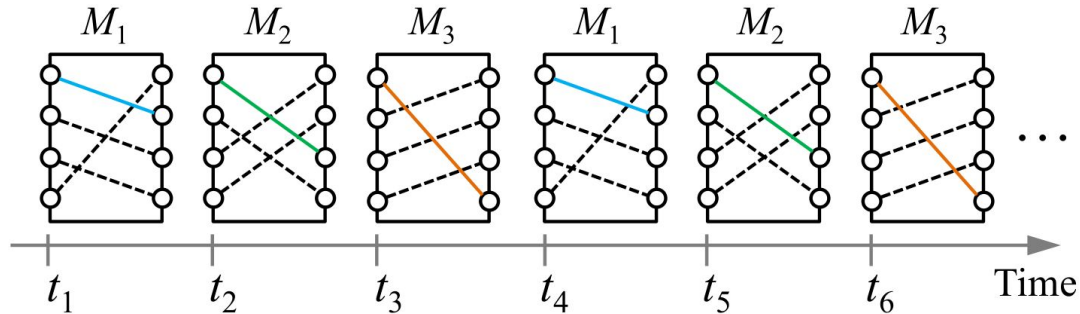
Computation Times

- Trade-off: Computation time and efficiency
 - How to scale to larger networks?
 - Especially with micro-/nano-second switching times?
- Algorithmic idea #1: **Distributed Control Plane**
 - E.g. ProjectoR (Ghobadi et al., 2016) 
 - Next part of the tutorial
- Algorithmic idea #2: **Oblivious Reconfiguration**
 - Combine topology design and cyclic reconfiguration schedule

Rotonet

(Mellette et al., SIGCOMM 2017)

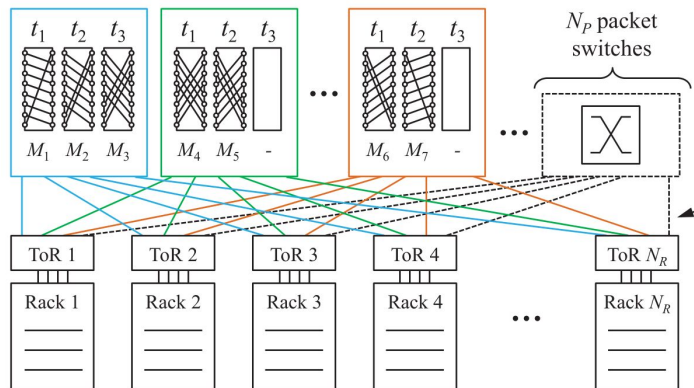
- Cycle through exponentially many matchings?
 - Observe: $O(n)$ matchings suffice for connectivity
 - Also allows for faster  and cheaper  hardware
- Provision 10%-20% as packet switching for ultra low latency traffic



Rotornet

(Mellette et al., SIGCOMM 2017)

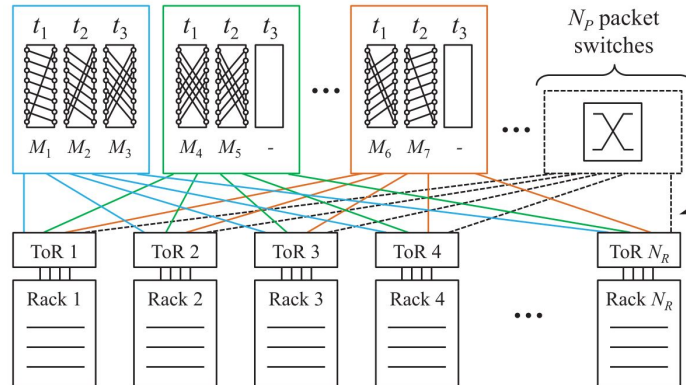
- Parallel less flexible switches to be even faster
 - Distribute matchings over rotor switches
 - E.g. 2048 racks: 16 different matchings with 128 switches
 - Reconfig in microseconds, serve traffic in $O(1)$ ms
 - Also: No single point of failure



Rotornet

(Mellette et al., SIGCOMM 2017)

- For uniform traffic: great behaviour with single-hop
- For skewed traffic: leverage Valiant's routing
 - Buffer indirect traffic on per-rack basis
 - Don't hinder direct traffic: distributed offer-accept protocol



Even More Oblivious: Static Networks

- Observation:
 - Random graphs are great for throughput via multi-hop
 - Build data centers randomly (Singla et al., Jellyfish, NSDI 2012)

Even More Oblivious: Static Networks

- Observation:
 - Random graphs are great for throughput via multi-hop
 - Build data centers randomly (Singla et al., Jellyfish, NSDI 2012)

- Go deterministic with Expanders: **Beyond fat-trees without antennae, mirrors, and disco-balls**

Algorithmica (2017) 78:1225–1245
DOI 10.1007/s00453-016-0269-x

Simon Kassing
ETH Zürich
simon.kassing@inf.ethz.ch

Asaf Valadarsky
Hebrew University of Jerusalem
asaf.valadarsky@mail.huji.ac.il

Gal Shahaf
Hebrew University of Jerusalem
gal.shahaf@mail.huji.ac.il

Michael Schapira
Hebrew University of Jerusalem
schapiram@huji.ac.il

Ankit Singla
ETH Zürich
ankit.singla@inf.ethz.ch

SIGCOMM'17

Explicit Expanding Expanders

Michael Dinitz¹ · Michael Schapira² ·
Asaf Valadarsky²

Xpander: Towards Optimal-Performance Datacenters

Asaf Valadarsky*
asaf.valadarsky@mail.huji.ac.il

Gal Shahaf†
gal.shahaf@mail.huji.ac.il

Michael Dinitz‡
mdinitz@cs.jhu.edu

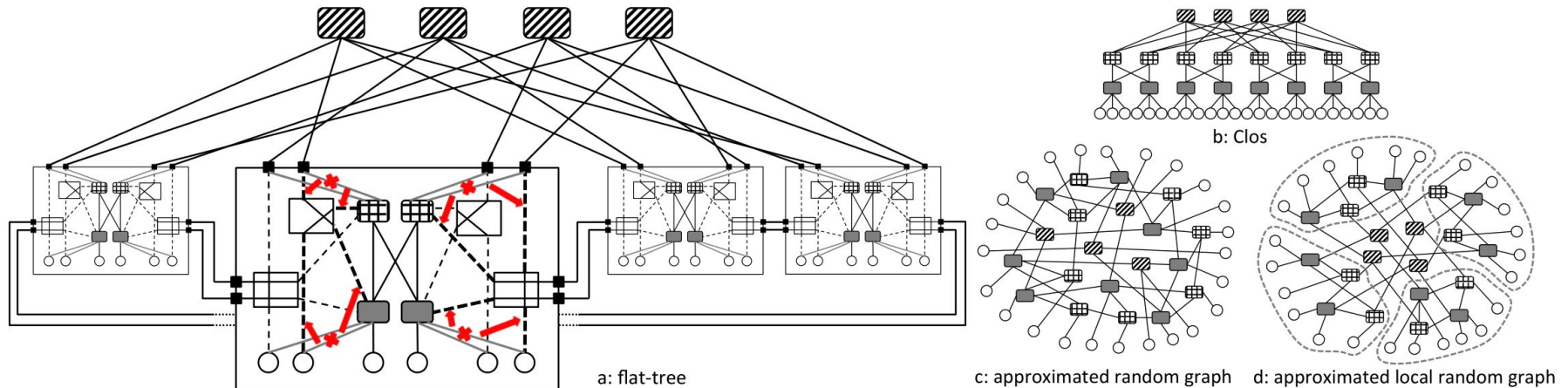
Michael Schapira*
schapiram@huji.ac.il

CoNEXT '16

Flat-Tree

(Xia et al, SIGCOMM 2017)

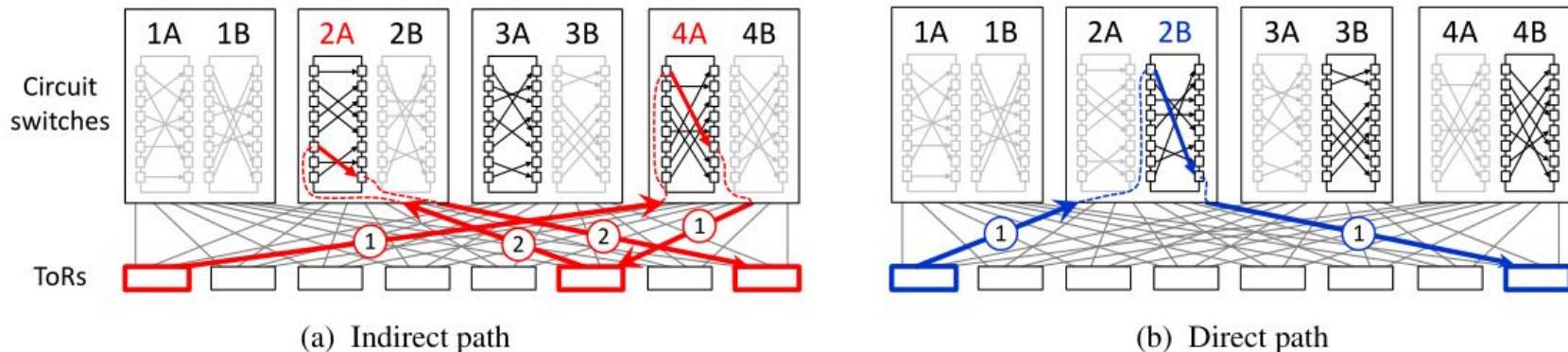
- Flat-Tree (note: not oblivious)
 - Locally convert between random graphs and Clos topologies
 - Use extremely cheap 4/6-port converter switches



Opera

(Mellette et al, arXiv 2019)

- Extend Rotornet to cycle through Expanders
 - Delay-tolerant traffic can wait for direct connections
 - Other traffic can use the expander networks
 - Low “bandwidth tax”
 - (Valiant 2-hop routing also possible)



Future (Algorithmic) Work Directions

- Single-hop reconfigurable “XOR” routing is sort of well-understood
 - But mixing with static network parts in general?
- Leveraging multi-hop connections?
 - Efficient heuristics exist, general theoretical framework?
- Speeding up the control plane
 - Oblivious is clearly very fast :-)
 - More distributed approaches

References

- Long Luo, Klaus-Tycho Foerster, Stefan Schmid, Hongfang Yu: DaRTree: deadline-aware multicast transfers in reconfigurable wide-area networks. IWQoS 2019
- Survey of Reconfigurable Data Center Networks. Foerster and Schmid. SIGACT News, 2019.
- Mohammad Al-Fares, Alexander Loukissas, Amin Vahdat: A scalable, commodity data center network architecture. SIGCOMM 2008
- Arjun Singh et al.: Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. SIGCOMM 2015
- Srikanth Kandula, Jitendra Padhye, Paramvir Bahl: Flyways To De-Congest Data Center Networks. HotNets 2009
- Manya Ghobadi et al.: ProjecToR: Agile Reconfigurable Data Center Interconnect. SIGCOMM 2016
- Daniel Halperin, Srikanth Kandula, Jitendra Padhye, Paramvir Bahl, David Wetherall: Augmenting data center networks with multi-gigabit wireless links. SIGCOMM 2011
- Nathan Farrington et al.: Helios: a hybrid electrical/optical switch architecture for modular data centers. SIGCOMM 2010
- Guohui Wang et al.: c-Through: part-time optics in data centers. SIGCOMM 2010
- Ankit Singla, Atul Singh, Kishore Ramachandran, Lei Xu, Yueping Zhang: Proteus: a topology malleable data center network. HotNets 2010
- Kai Chen et al.: OSA: An Optical Switching Architecture for Data Center Networks With Unprecedented Flexibility. IEEE/ACM Trans. Netw. 22(2): 498-511 (2014)
- Ankit Singla, Atul Singh, Yan Chen: OSA: An Optical Switching Architecture for Data Center Networks with Unprecedented Flexibility. NSDI 2012
- Navid Hamed Azimi et al.: FireFly: a reconfigurable wireless data center fabric using free-space optics. SIGCOMM 2014
- Yiting Xia et al.: A Tale of Two Topologies: Exploring Convertible Data Center Network Architectures with Flat-tree. SIGCOMM 2017
- Xia Zhou et al.: Mirror mirror on the ceiling: flexible wireless links for data centers. SIGCOMM 2012
- Klaus-Tycho Foerster, Manya Ghobadi, Stefan Schmid: Characterizing the algorithmic complexity of reconfigurable data center architectures. ANCS 2018
- T. Fenz, K.-T. Foerster, S. Schmid, A. Villedieu: Efficient Non-Segregated Routing for Reconfigurable Demand-Aware Networks. IFIP Networking 2019
- K.-T. Foerster et al.: On the Complexity of Non-Segregated Routing in Reconfigurable Data Center Architectures. ACM SIGCOMM CCR 49(2): 3-8 (2019)
- George Porter et al: Integrating microsecond circuit switching into the data center. SIGCOMM 2013
- Li Chen et al.: Enabling Wide-Spread Communications on Optical Fabric with MegaSwitch. NSDI 2017
- William M. Mellette et al.: RotorNet: A Scalable, Low-complexity, Optical Datacenter Network. SIGCOMM 2017
- He Liu et al.: Scheduling techniques for hybrid circuit/packet networks. CoNEXT 2015
- Shaileshh Bojja Venkatakrishnan et al.: Costly circuits, submodular schedules and approximate Carathéodory Theorems. SIGMETRICS 2016
- Shaileshh Bojja Venkatakrishnan et al.: Costly circuits, submodular schedules and approximate Carathéodory Theorems. Queueing Syst. 88(3-4): 311-347 (2018)
- William M. Mellette et al.: Expanding across time to deliver bandwidth efficiency and low latency. arXiv:1903.12307

Reconfigurable Networks: Enablers, Algorithms, Complexity

Ramakrishnan Durairajan, Klaus-Tycho Forster, Stefan Schmid

Tutorial @ ACM Sigmetrics 2019
Phoenix, Arizona, USA