ElasticTree: Saving Energy in Data Center Networks

Brandon Heller (Stanford)
Srini Seetharaman (Deutsche Telekom R&D, Los Altos)
Priya Mahadevan (Hewlett-Packard Labs, Palo Alto)
Yiannis Yiakoumis (Stanford)
Puneet Sharma (Hewlett-Packard Labs, Palo Alto)
Sujata Banerjee (Hewlett-Packard Labs, Palo Alto)
Nick McKeown (Stanford)

Network Power Consumption: 6B kWh in 2006!

~267K average size homes

$50M a month

a ginormous amount of CO_2

2x increase projected for 2011

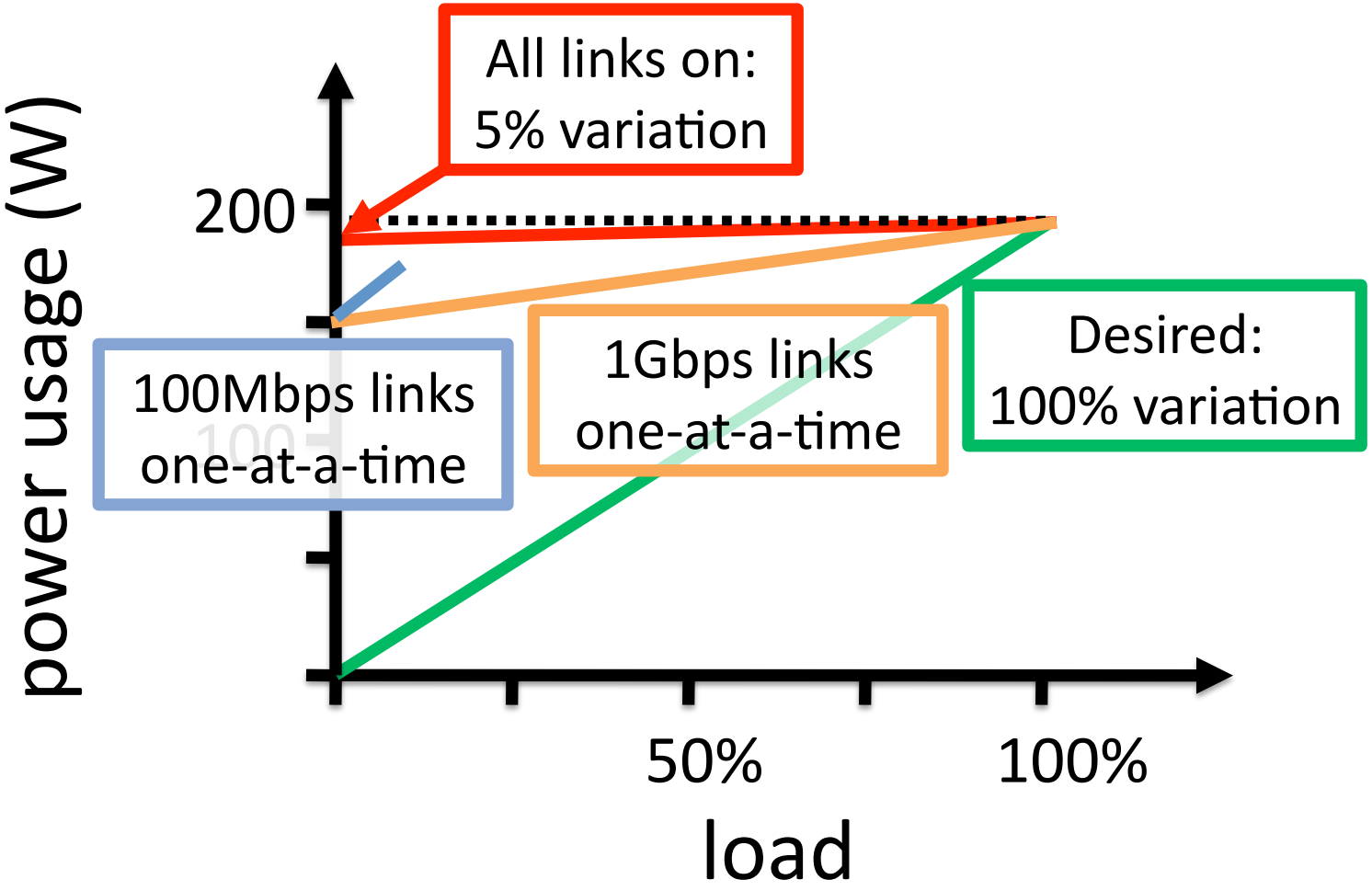
48-port Switch

~150W nothing connected

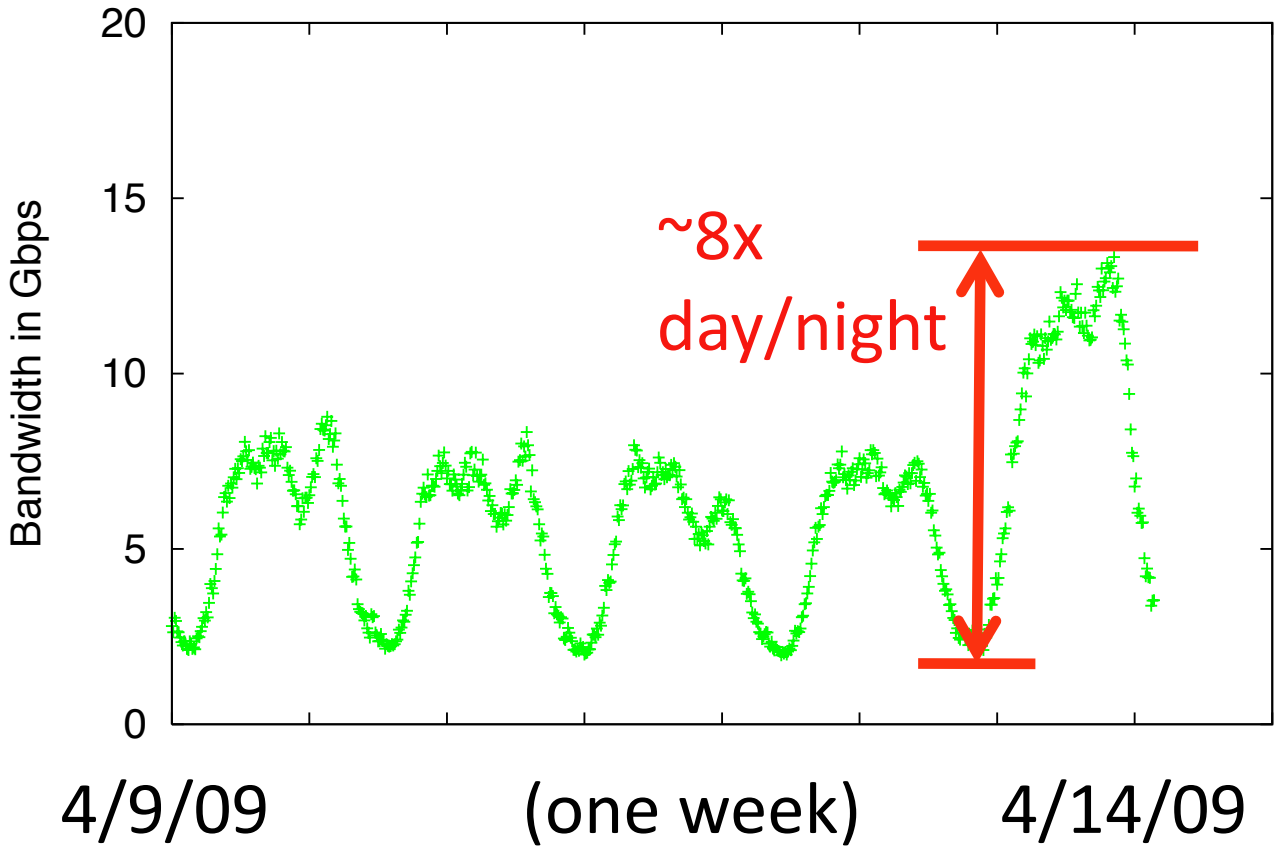
~185W all 48 1G links on

Power Meter

Top-of-Rack Switch

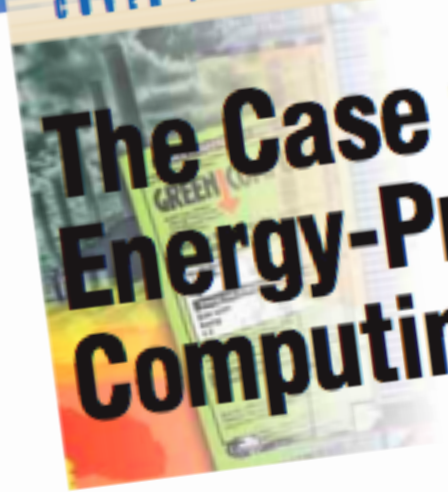


E-commerce website, 300 servers



GREEN COMPUTING

COVER FEATURE



The Case for Energy-Proportional Computing

Luiz André Barroso and Urs Hölzle

Google

Energy-proportional designs would enable large energy savings in servers, potentially doubling their efficiency in real-life use. Achieving energy proportionality will require significant improvements in the energy usage profile of every system component, particularly the memory and disk subsystems.

Energy efficiency, a new focus for general-purpose computing, has been a major technology driver in the mobile and embedded areas for some time. Earlier work emphasized extending battery life, but it has since expanded to include peak power reduction because thermal constraints began to limit further CPU performance improvements.

Energy management has now become a key issue for operations, focusing on the

trends could make energy a dominant factor in the total cost of ownership.[3] Besides the server electricity bill, TCO includes other energy-dependent components such as the cost of energy for the cooling infrastructure and provisioning costs, specifically the data center infrastructure's cost. To a first-order approximation, both cooling and provisioning costs are proportional to the average energy that servers consume, therefore energy efficiency improvements should benefit all energy-dependent TCO components.

such as the Climate Savers Computing Initiative

Provisioning for peak

+

Time-varying traffic demands

+

Low efficiency at low loads

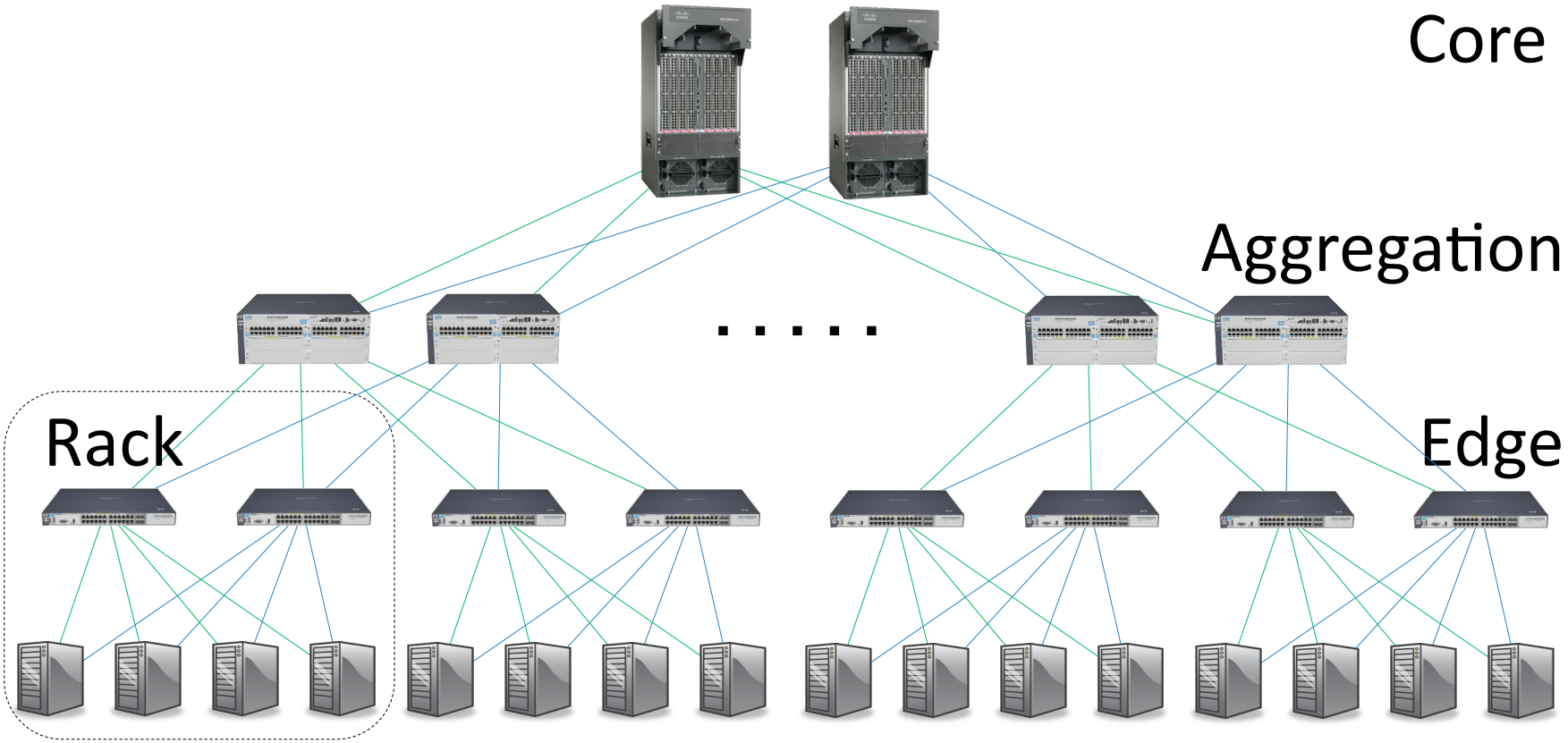
=

Lots of wasted power

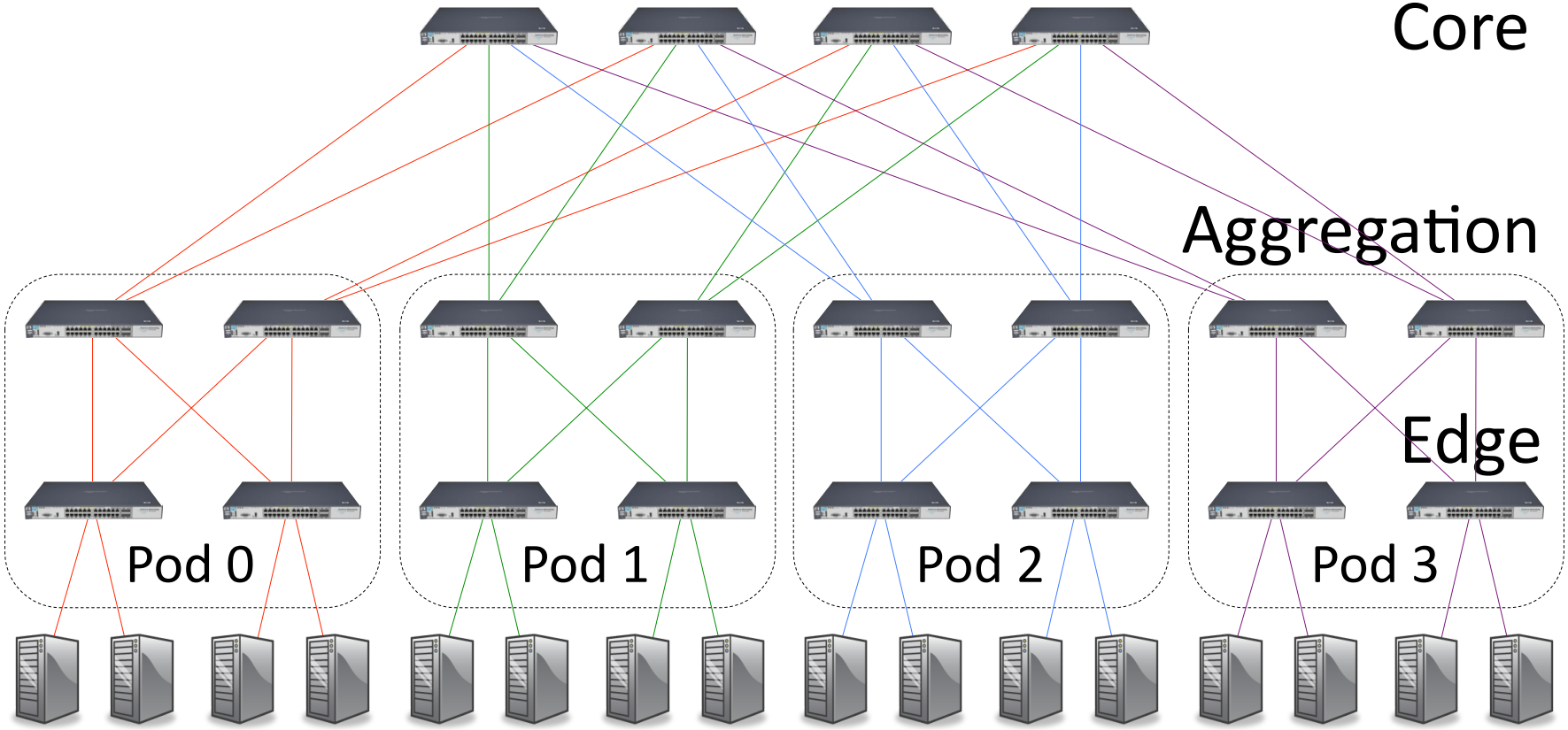
Can’t do much for 1 switch.

What if we have 1000 switches?

Scale-up Data Center: 2N Tree

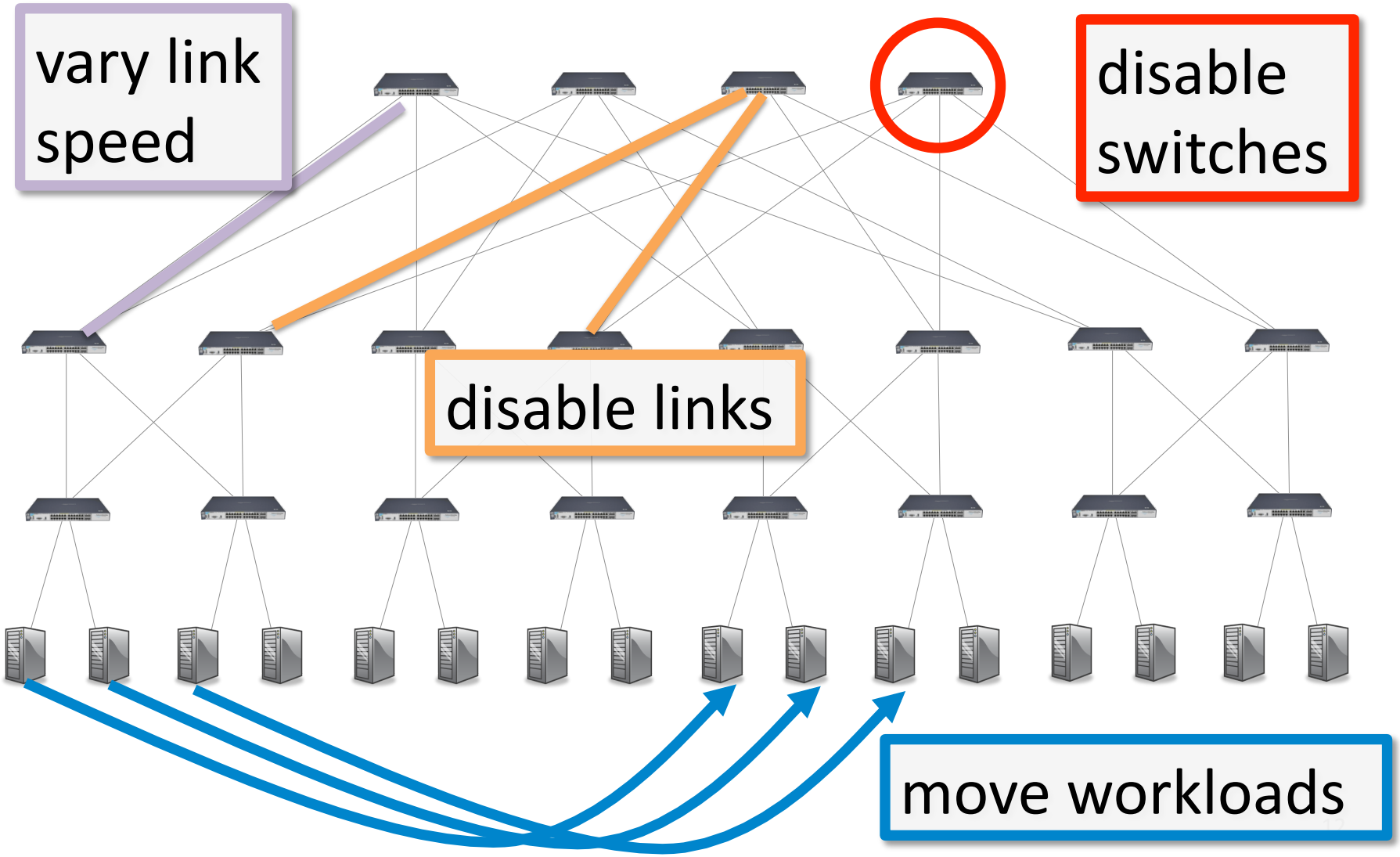


Scale-out Data Center: Fat Tree



from Scalable Commodity Data Center, SIGCOMM 2008, Al Fares et al.

Today's Network Power Knobs



Our approach:

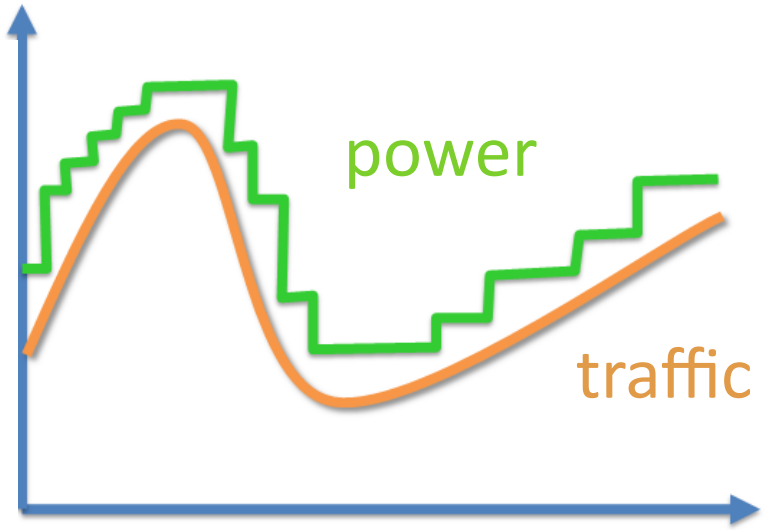
Turn off unneeded links and switches.

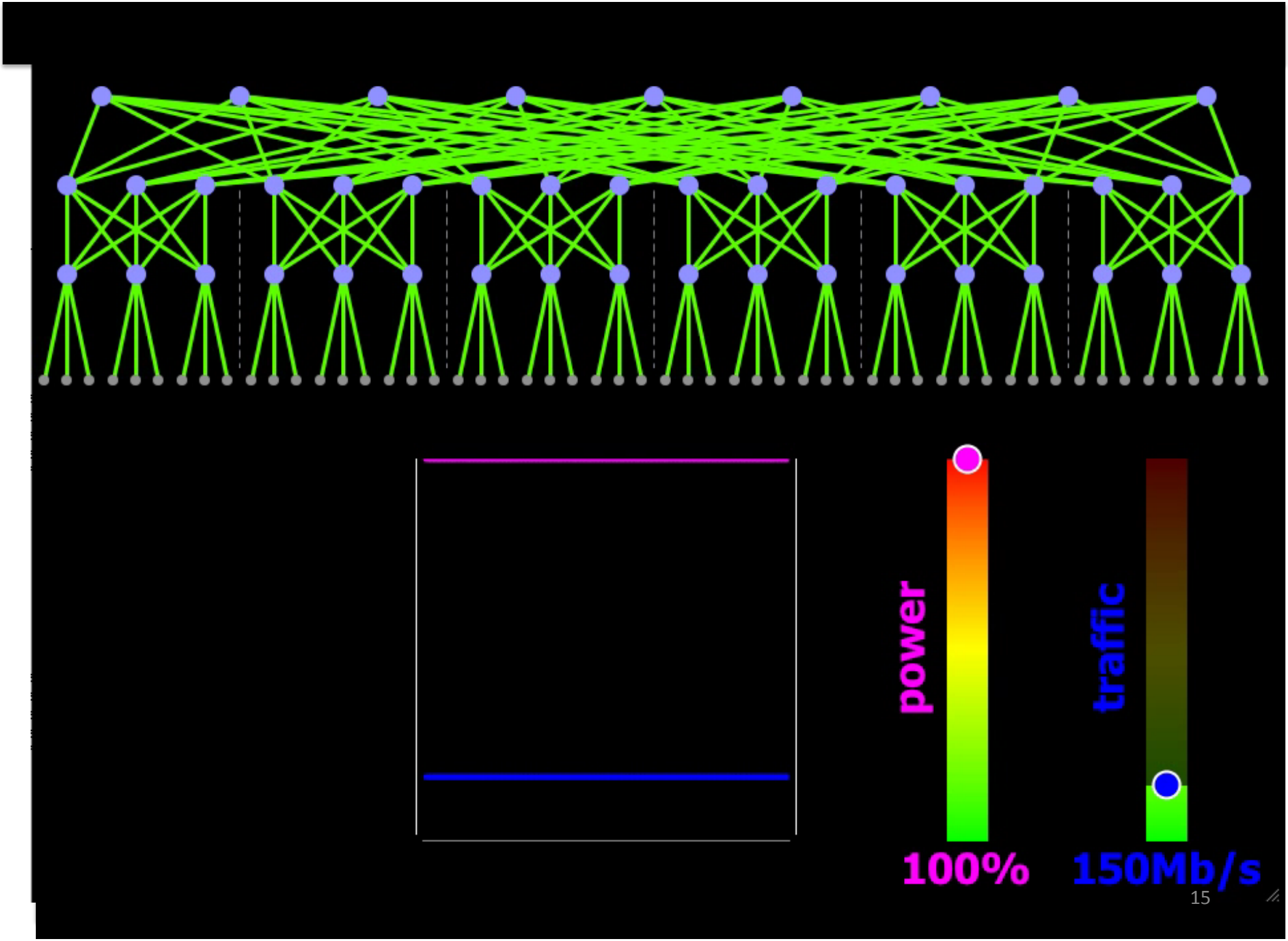
[Carefully]

[At Scale]

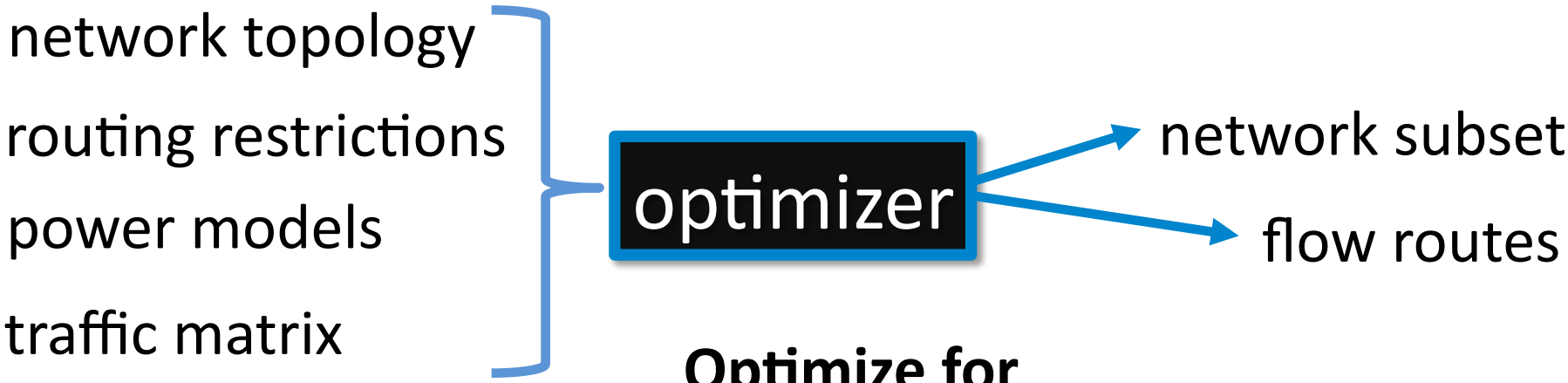
End goal:

Create an
energy-proportional data center **network**
from non-proportional components.




power
traffic
100%
150Mb/s

ElasticTree



Optimize for

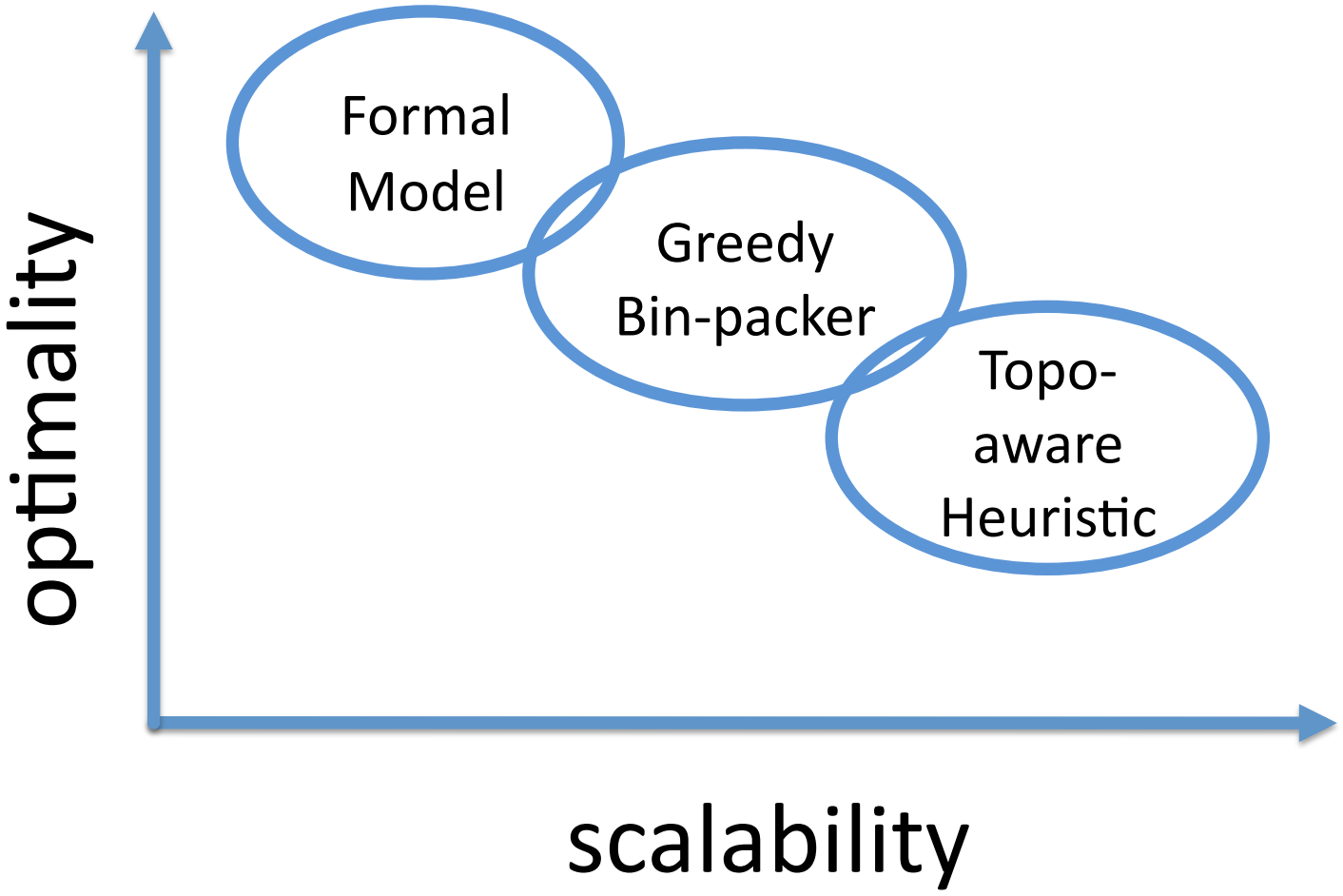
Power Efficiency

Later, balance:

+ Fault Tolerance

+ Utilization Bounds

3 Optimizers



Formal Model

Variables

Type	Description
Real	Amount of each flow along each link
Boolean	Switch power state
Boolean	Link power state

Optimization Goal

minimize Σ (link + switch power)

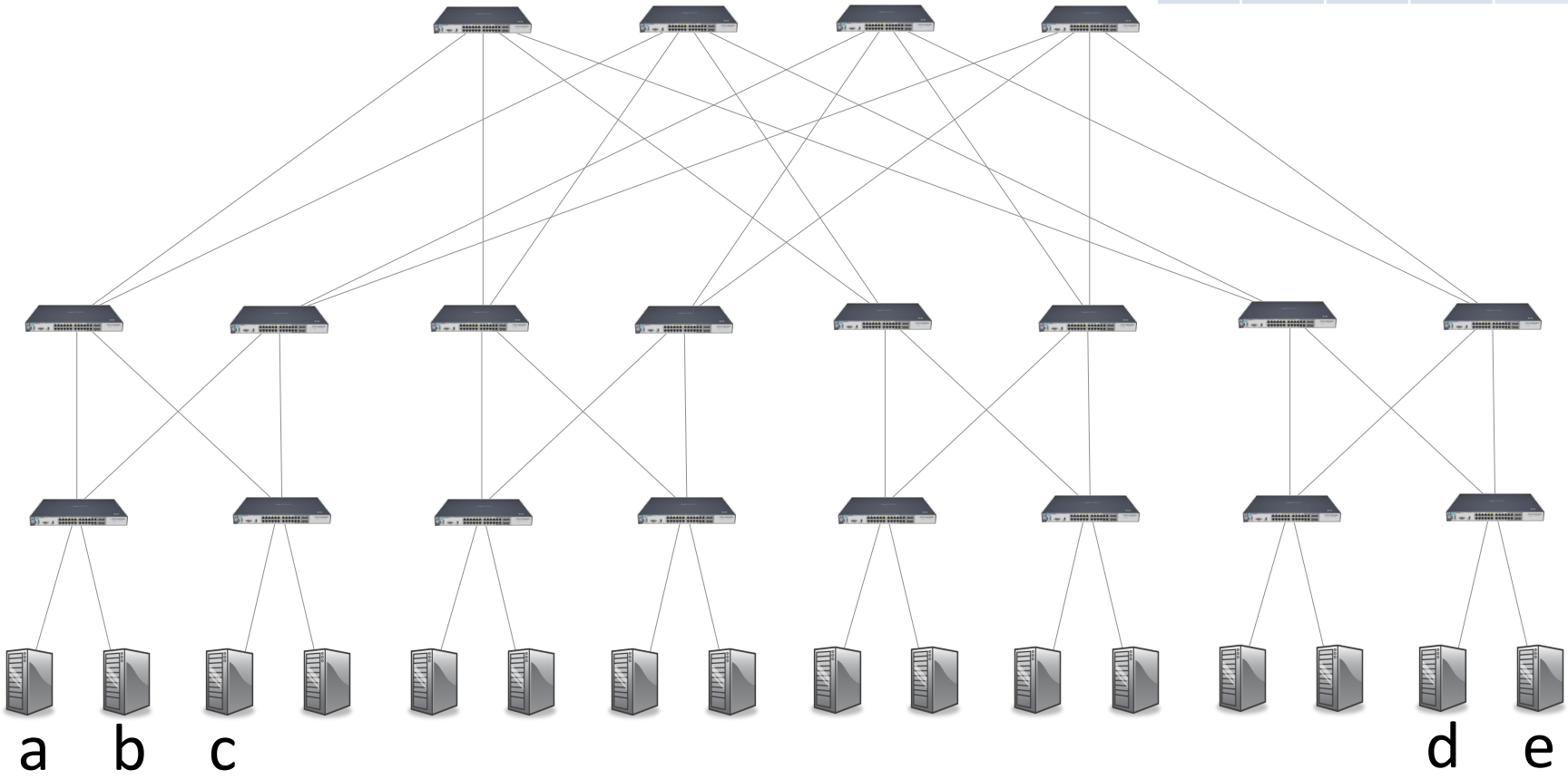
Constraints

Type	Constraint	Description
Multi-Commodity Flow	Capacity	traffic <= link rate?
	Flow Conservation	packets in = packets out?
	Demand Satisfaction	bandwidth >= demand?
Our Additions	Flow on active links only	link off ←→ no flow
	Connect switches and links	switch off ←→links off

Greedy bin-packing

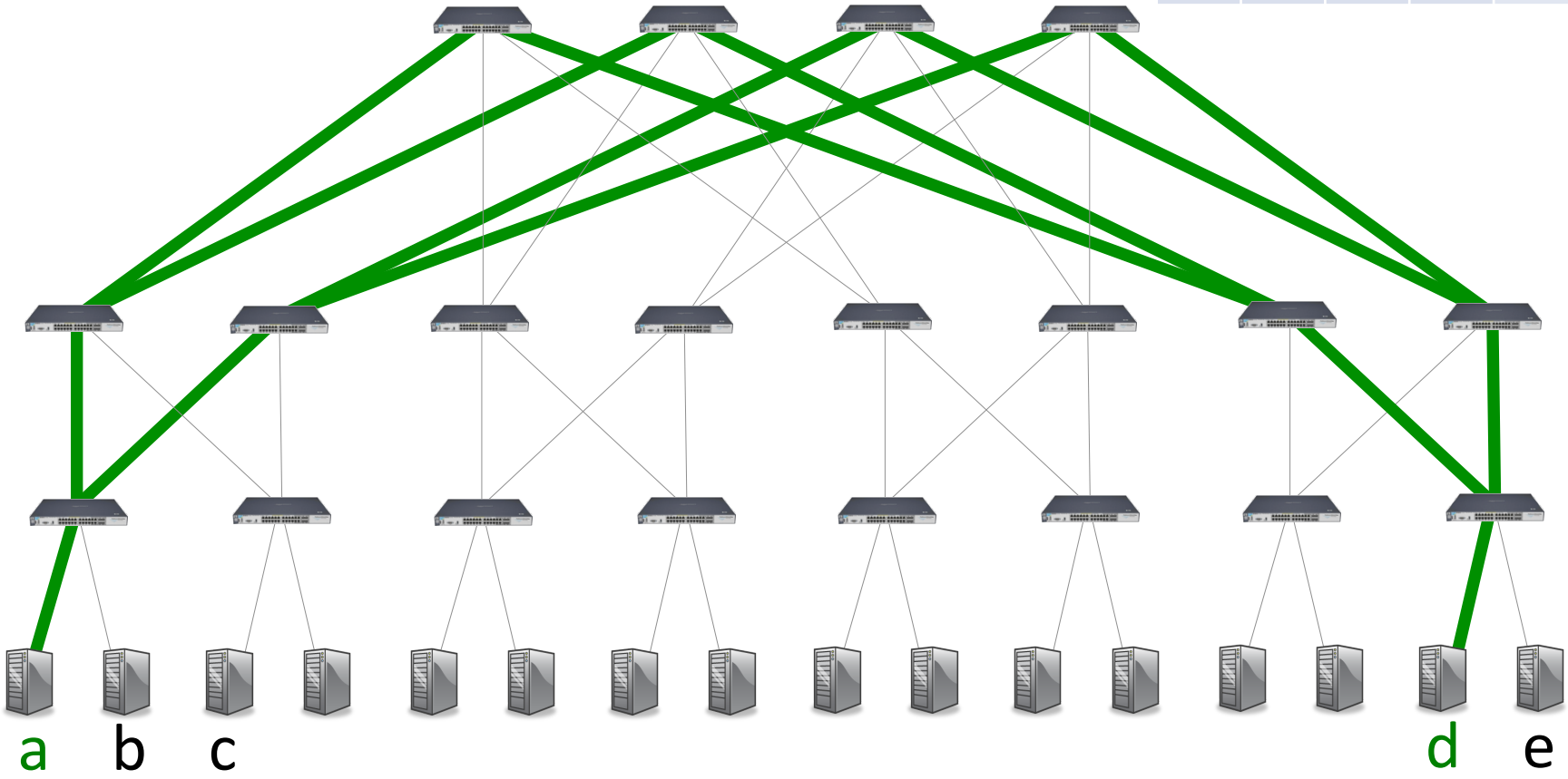
3 flows, each 0.3Gbps

	a	b	c	d	e
a				0.3	
b				0.3	
c					0.3
d					
e					



place flow [a,d]

	a	b	c	d	e
a				0.3	
b				0.3	
c					0.3
d					
e					



place flow [a,d]

	a	b	c	d	e
a				0.3	
b				0.3	
c					0.3
d					
e					

place flow [b,d]

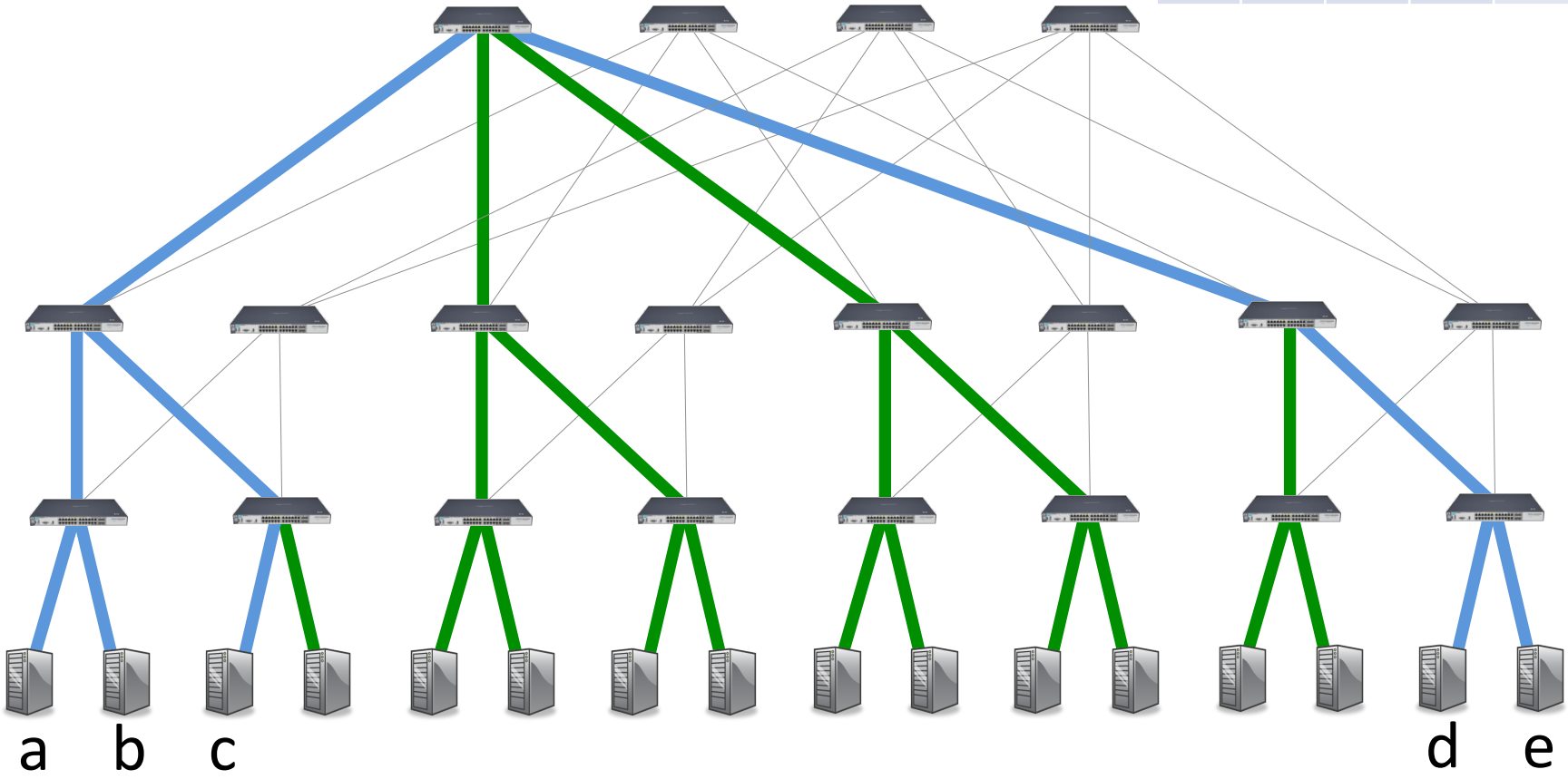
	a	b	c	d	e
a				0.3	
b				0.3	
c					0.3
d					
e					

place flow [c,e]

	a	b	c	d	e
a				0.3	
b				0.3	
c					0.3
d					
e					

ensure connectivity

	a	b	c	d	e
a				0.3	
b				0.3	
c					0.3
d					
e					

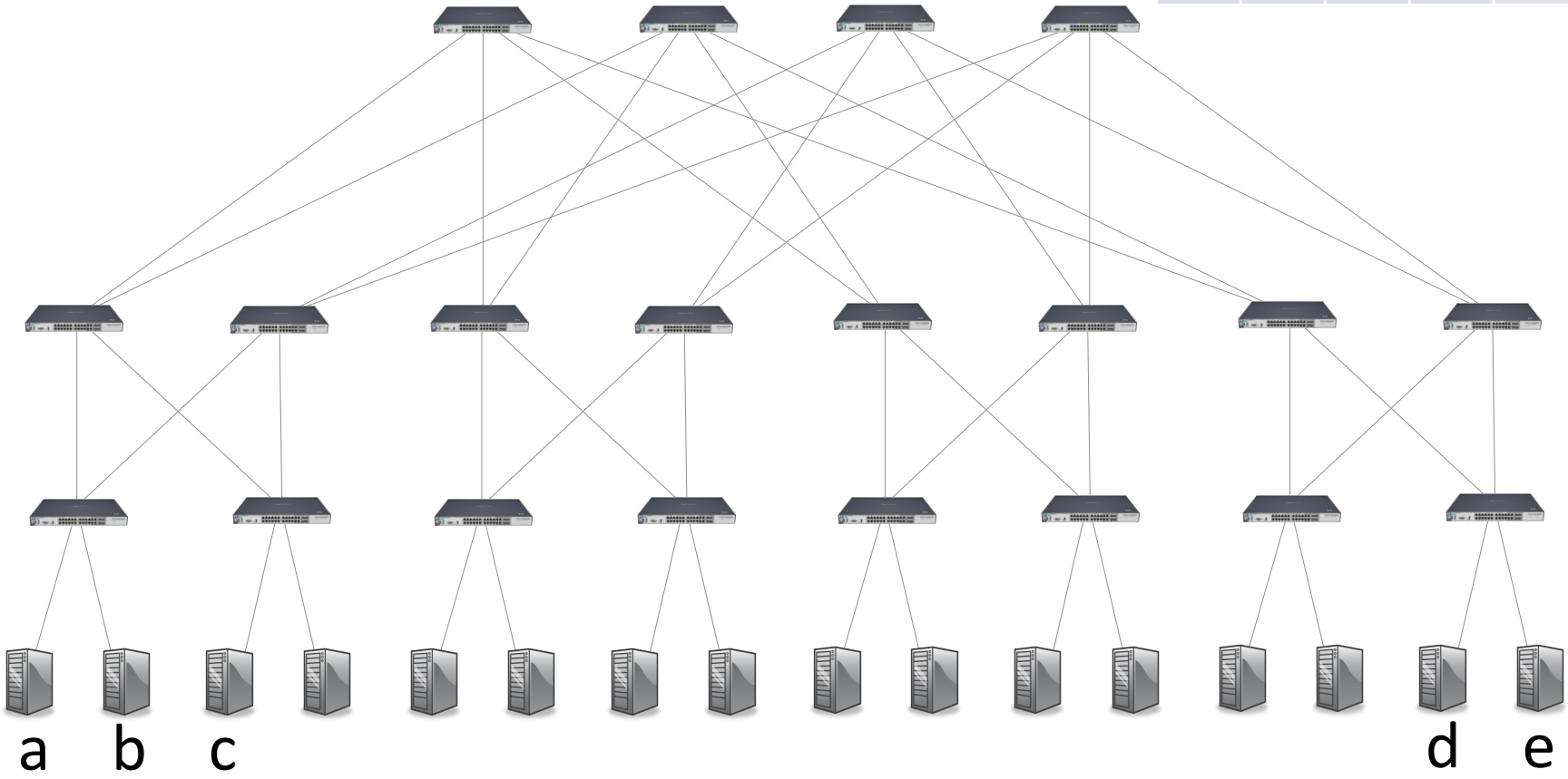


complete solution

	a	b	c	d	e
a				0.3	
b				0.3	
c					0.3
d					
e					

3 flows, each 0.4Gbps

	a	b	c	d	e
a				0.4	
b				0.4	
c					0.4
d					
e					



place flow [a,d]

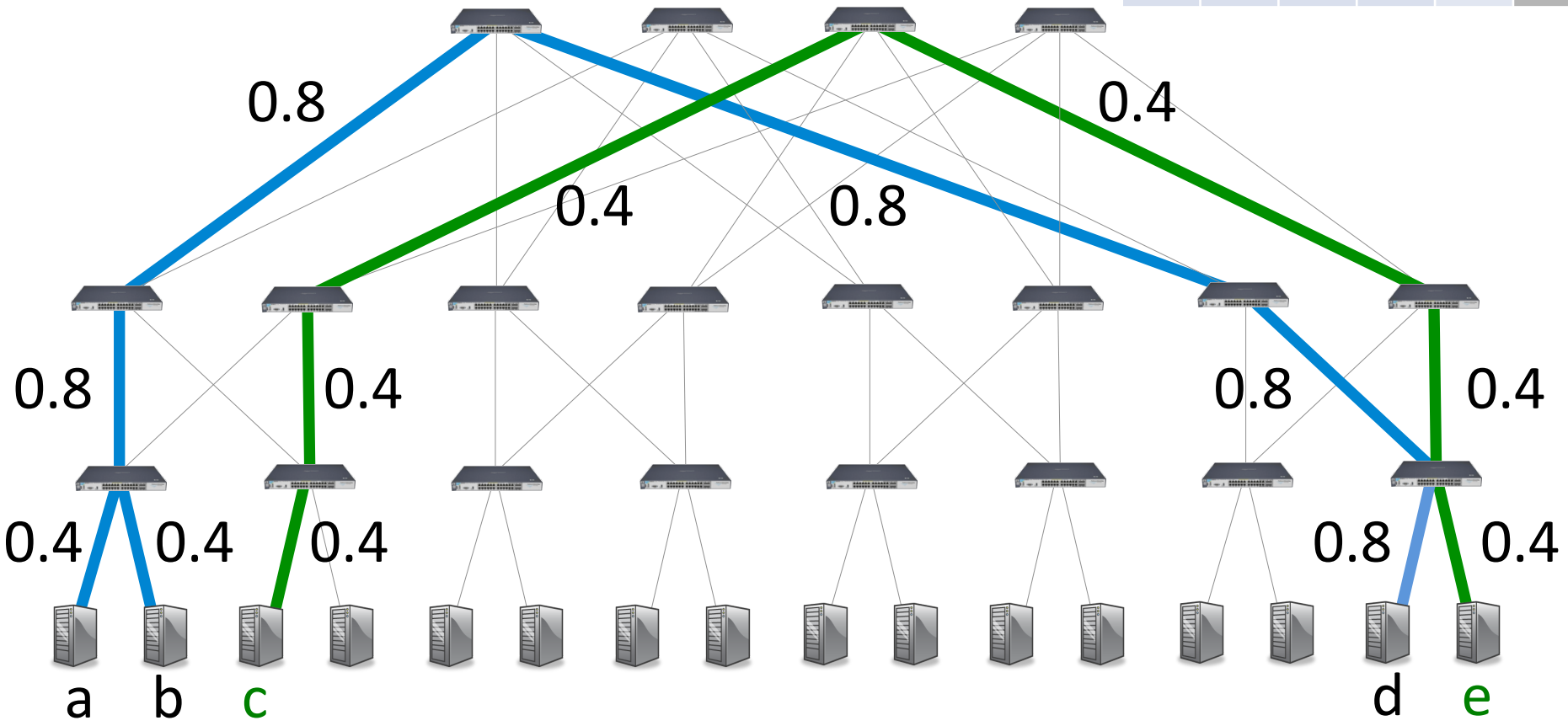
	a	b	c	d	e
a				0.4	
b				0.4	
c					0.4
d					
e					

place flow [b,d]

	a	b	c	d	e
a				0.4	
b				0.4	
c					0.4
d					
e					

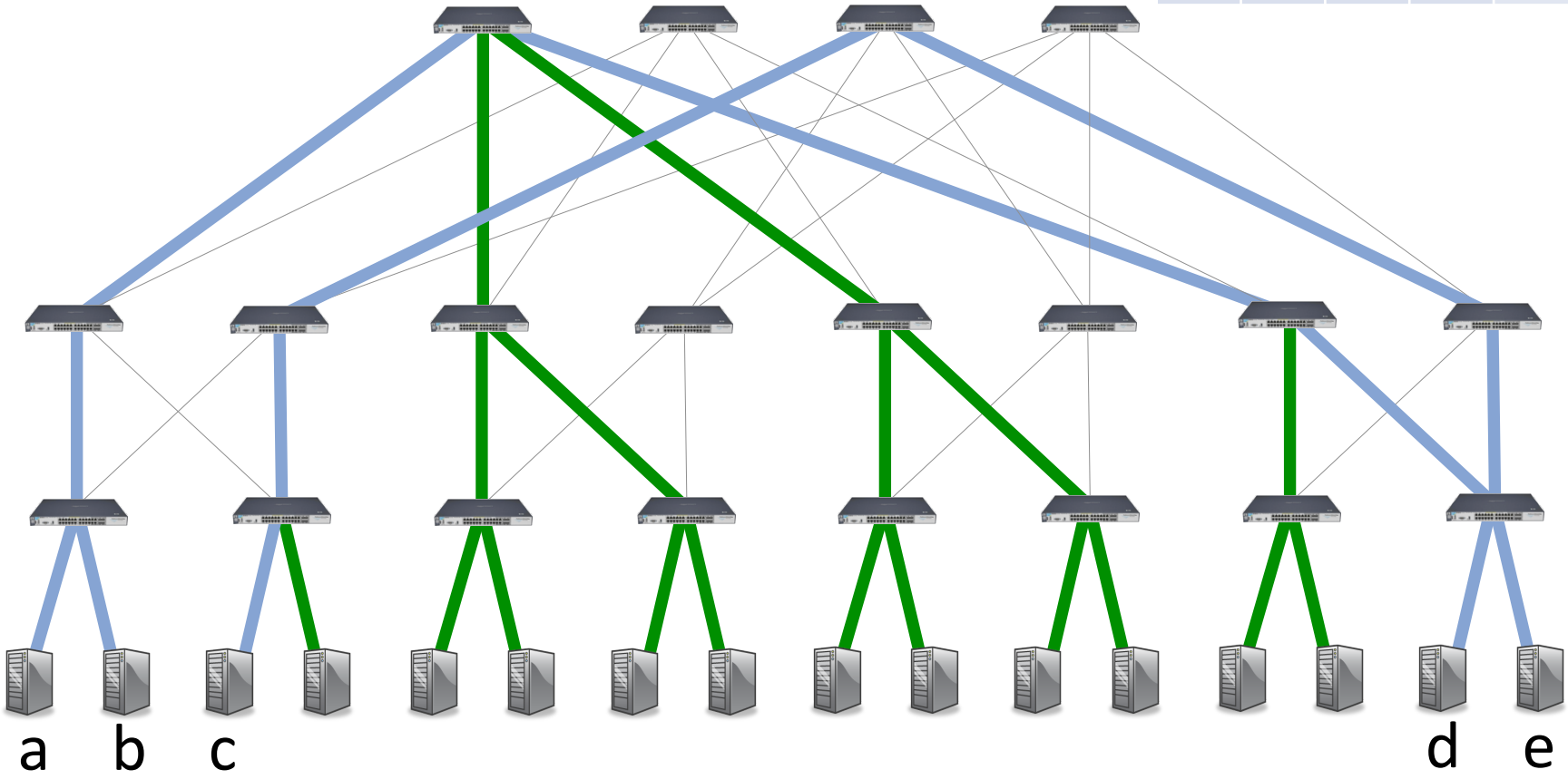
place flow [c,e]

	a	b	c	d	e
a				0.4	
b				0.4	
c					0.4
d					
e					



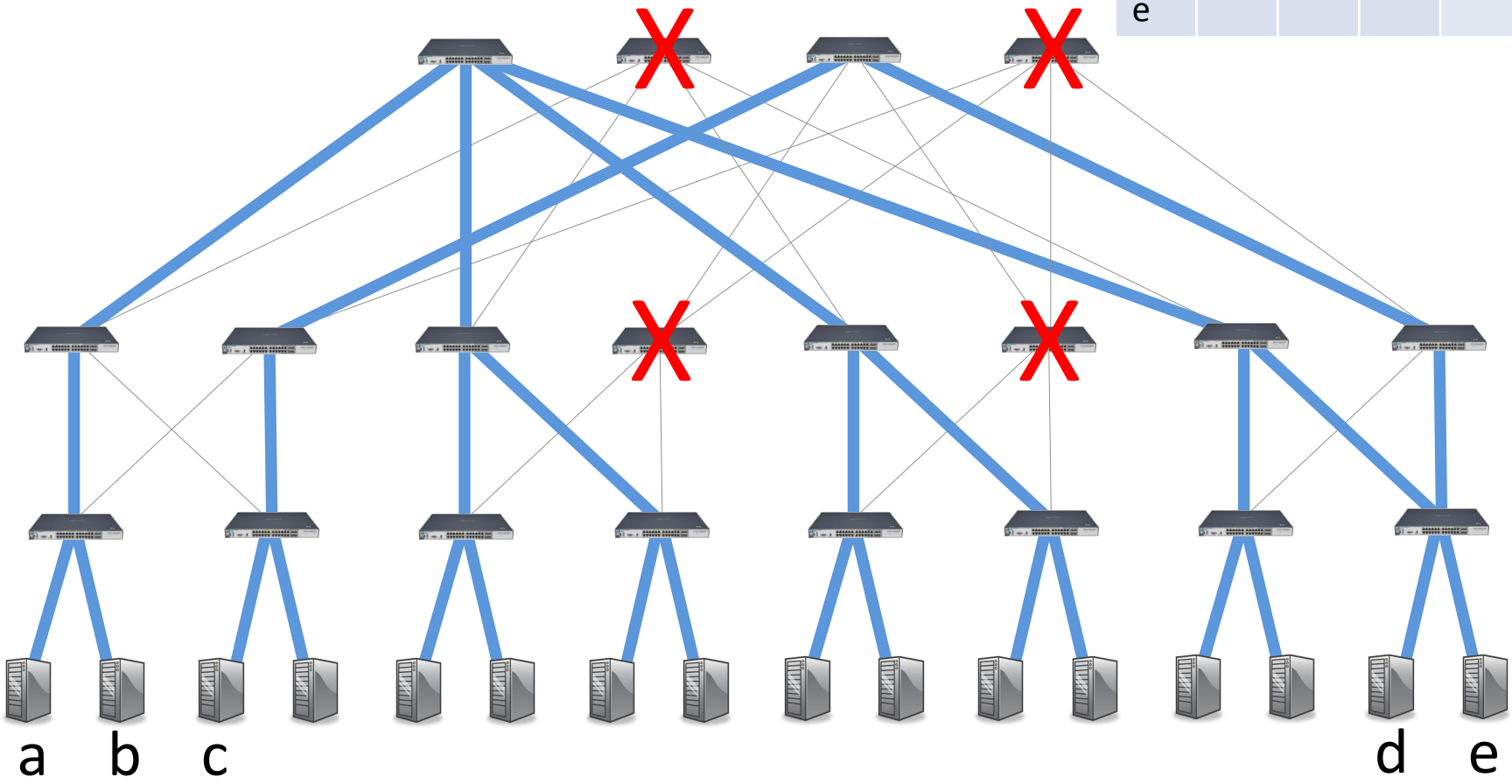
ensure connectivity

	a	b	c	d	e
a				0.4	
b				0.4	
c					0.4
d					
e					

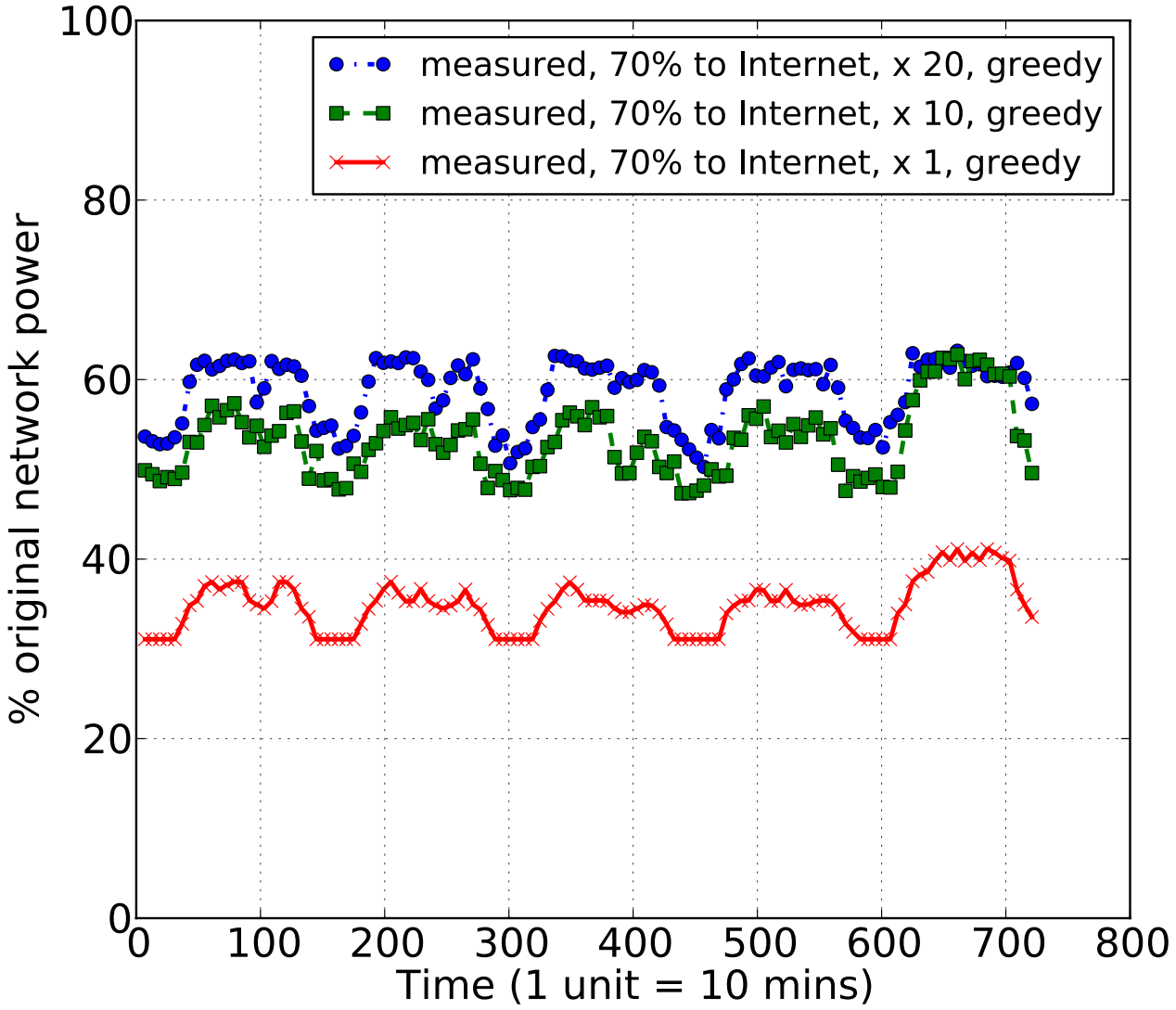


complete solution

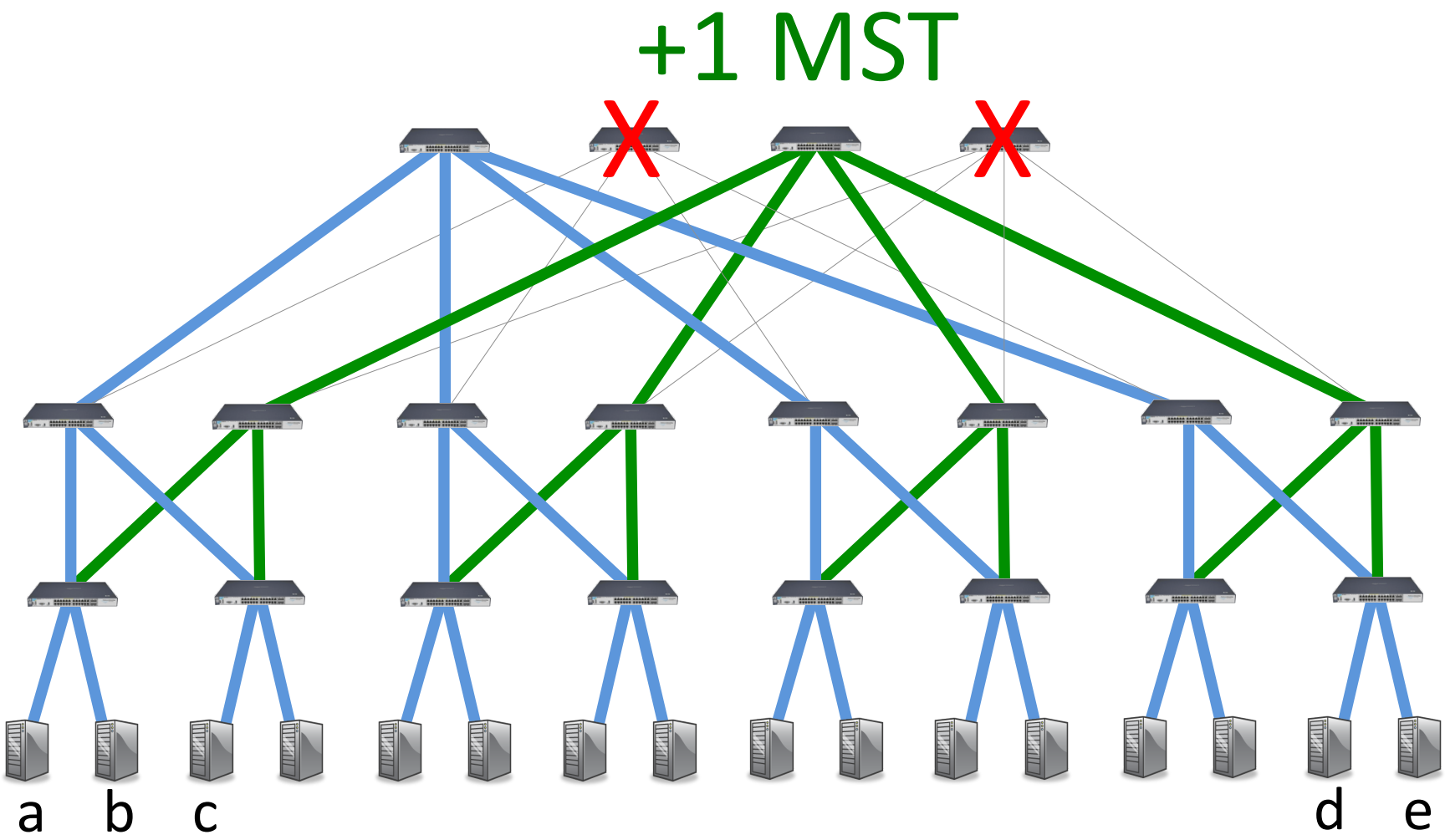
	a	b	c	d	e
a				0.4	
b				0.4	
c					0.4
d					
e					



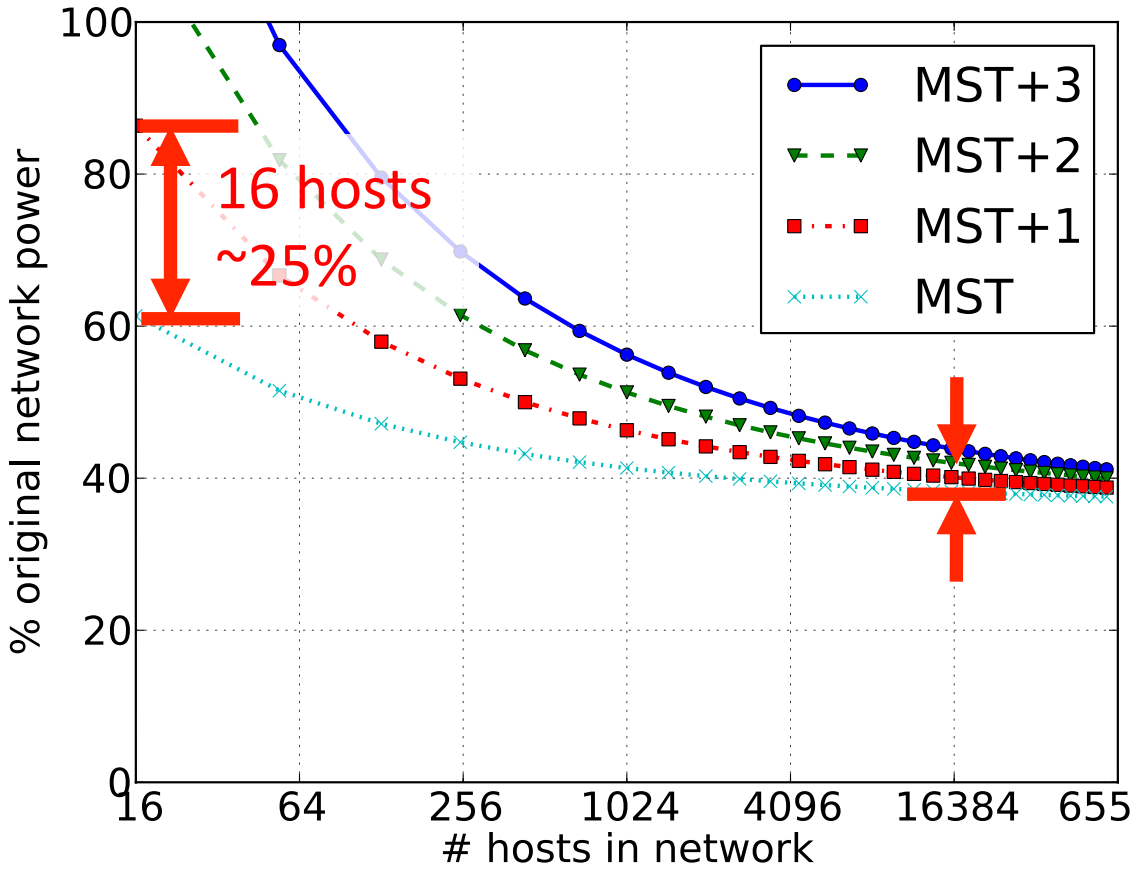
Power Savings



adding fault tolerance



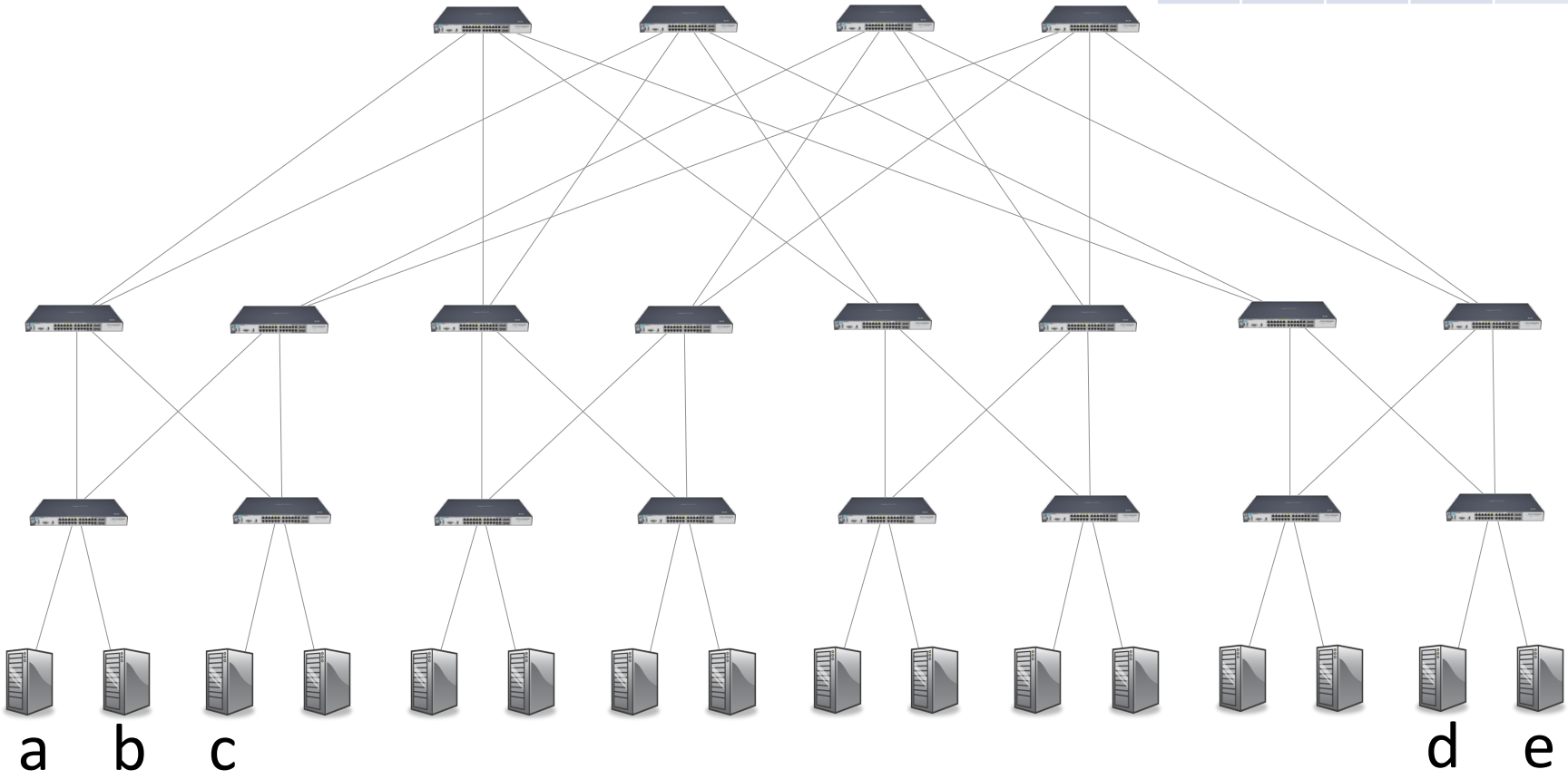
Cost of Fault Tolerance



3 flows, each 0.3Gbps

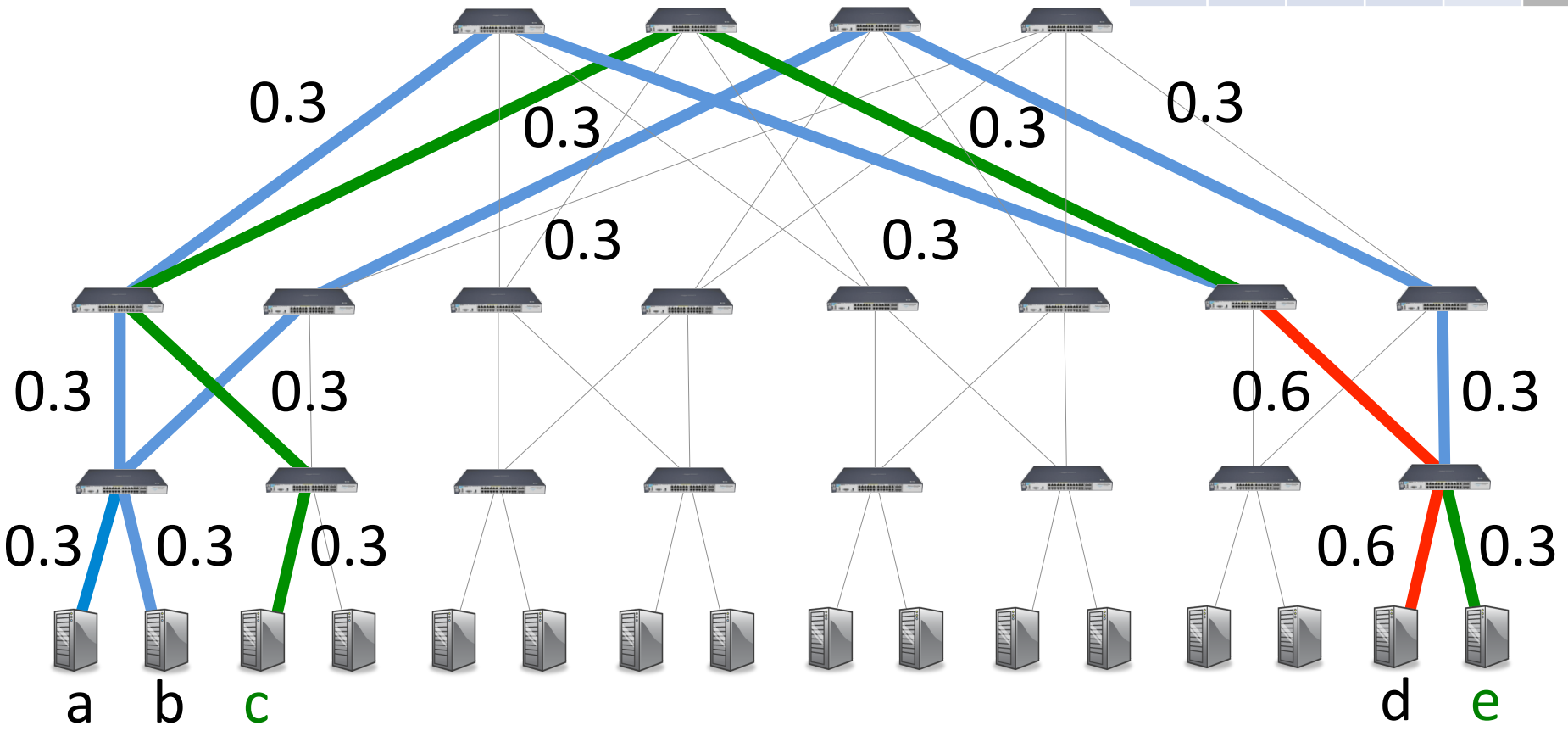
utilization bound: 0.5 Gbps

	a	b	c	d	e
a				0.3	
b				0.3	
c					0.3
d					
e					

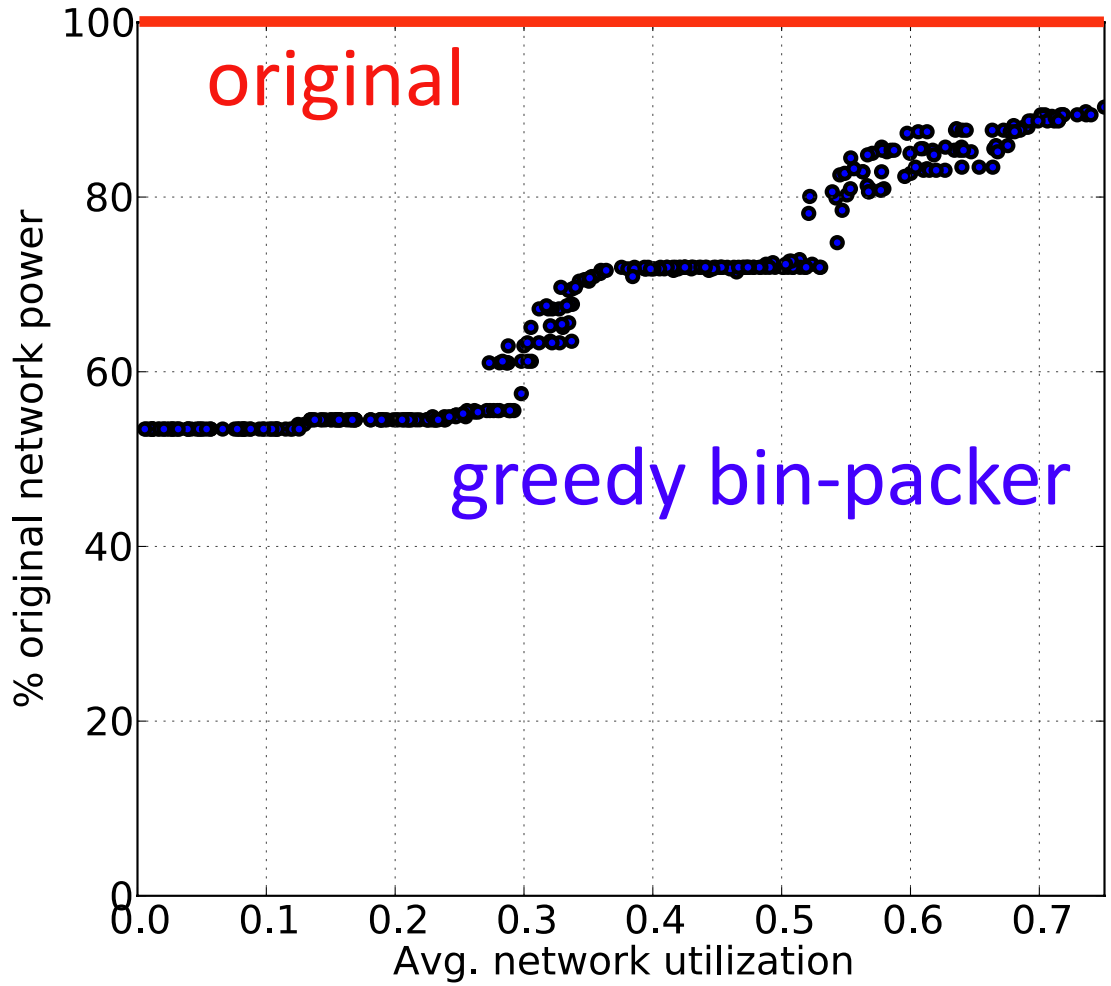


utilization bound: 0.5 Gbps
place flow [c,e]

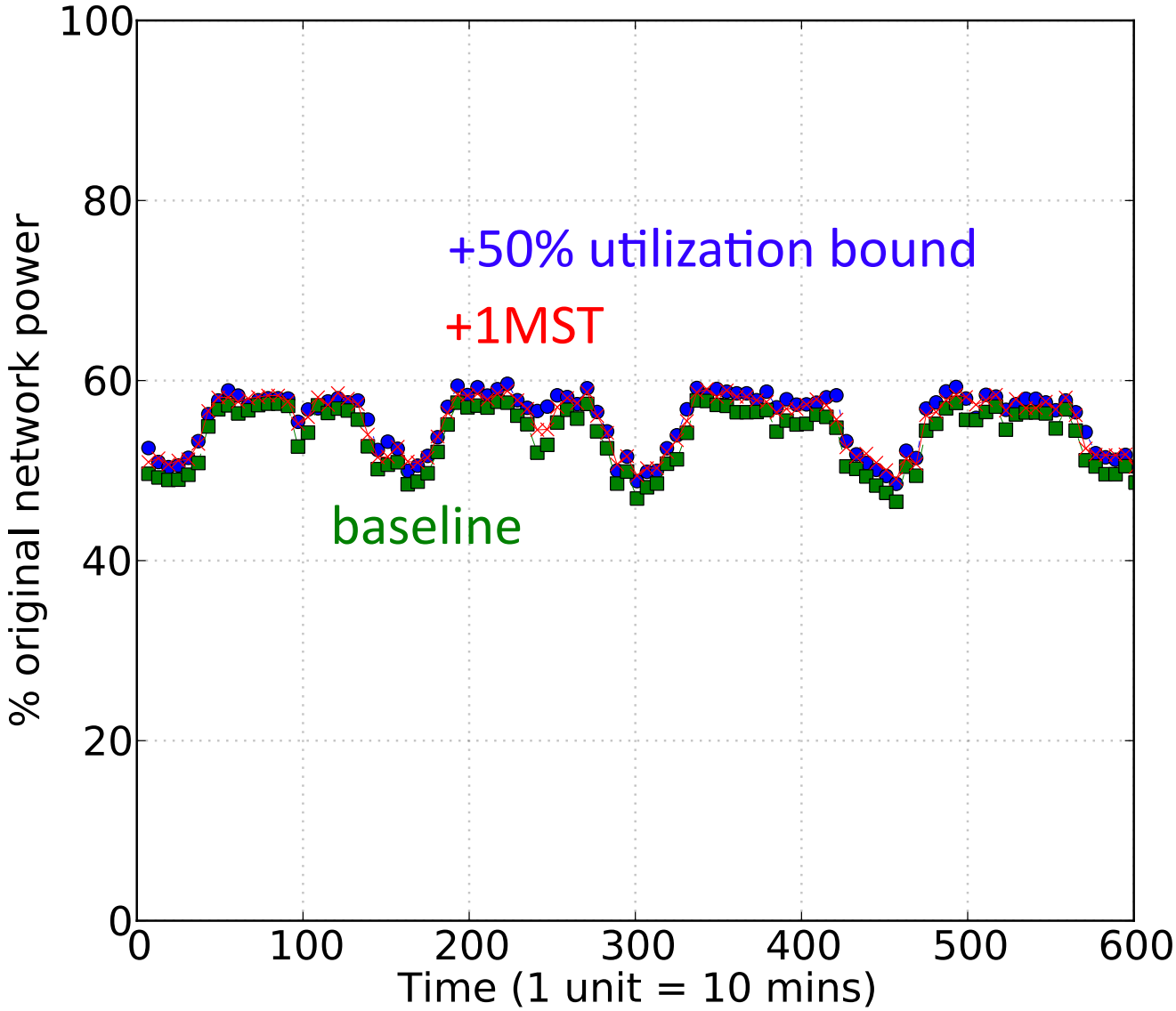
	a	b	c	d	e
a				0.3	
b				0.3	
c					0.3
d					
e					



54-node 3-layer Fat Tree (+1 MST)



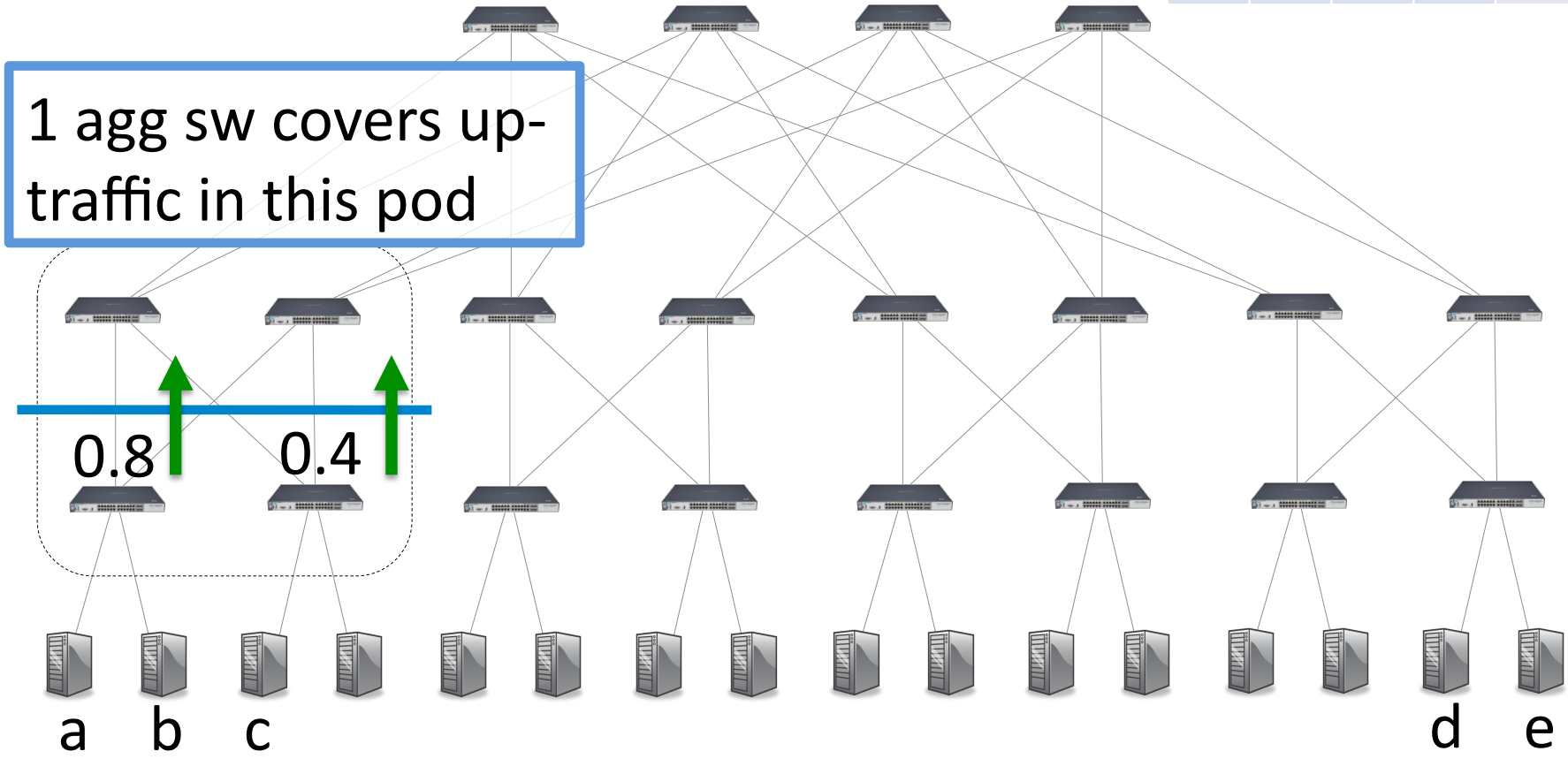
E-commerce trace, augmented



Topology-aware Heuristic

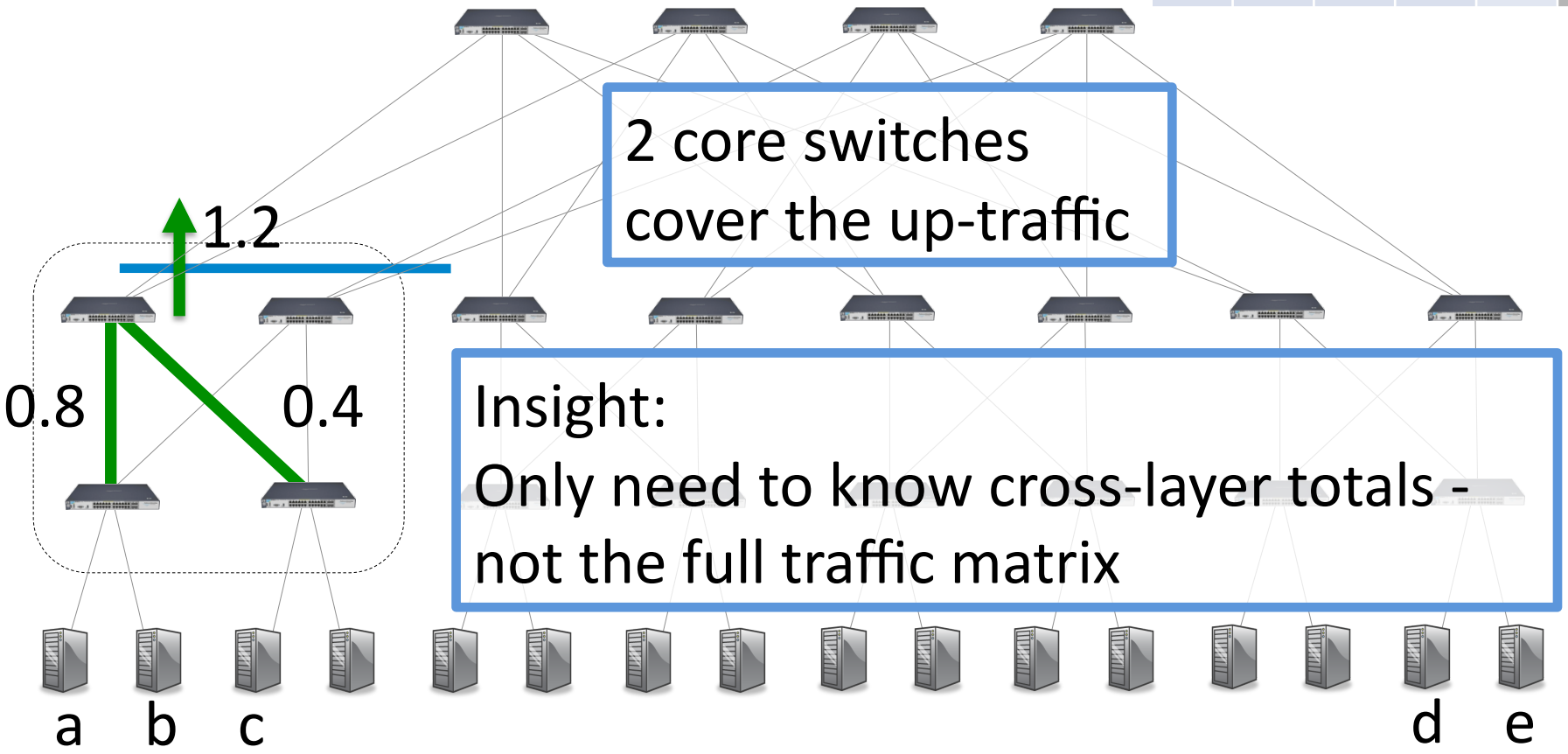
Topo-aware Heuristic

	a	b	c	d	e
a				0.4	
b				0.4	
c					0.4
d					
e					



Topo-aware Heuristic

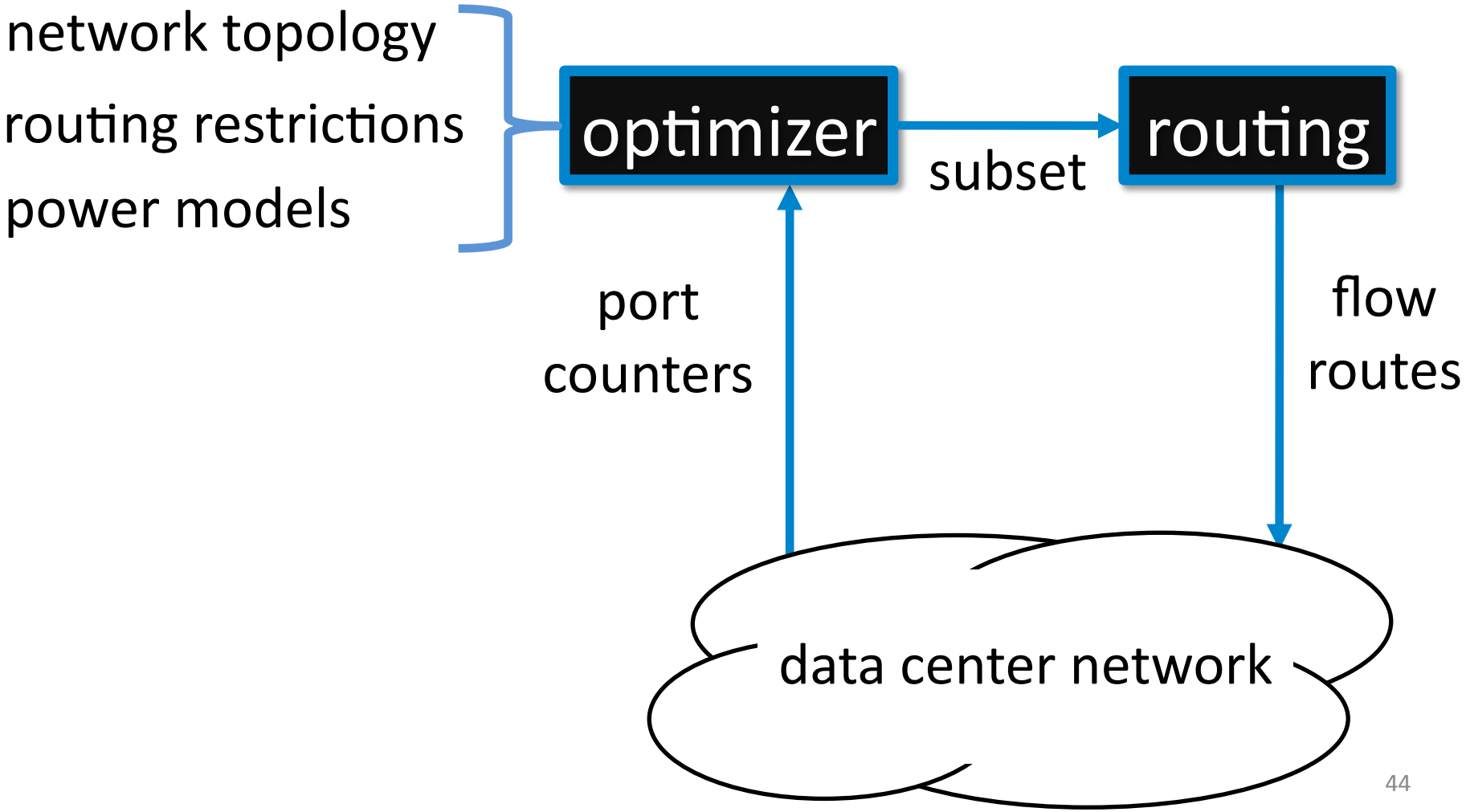
	a	b	c	d	e
a				0.4	
b				0.4	
c					0.4
d					
e					



2 core switches
cover the up-traffic

Insight:
Only need to know cross-layer totals - not the full traffic matrix

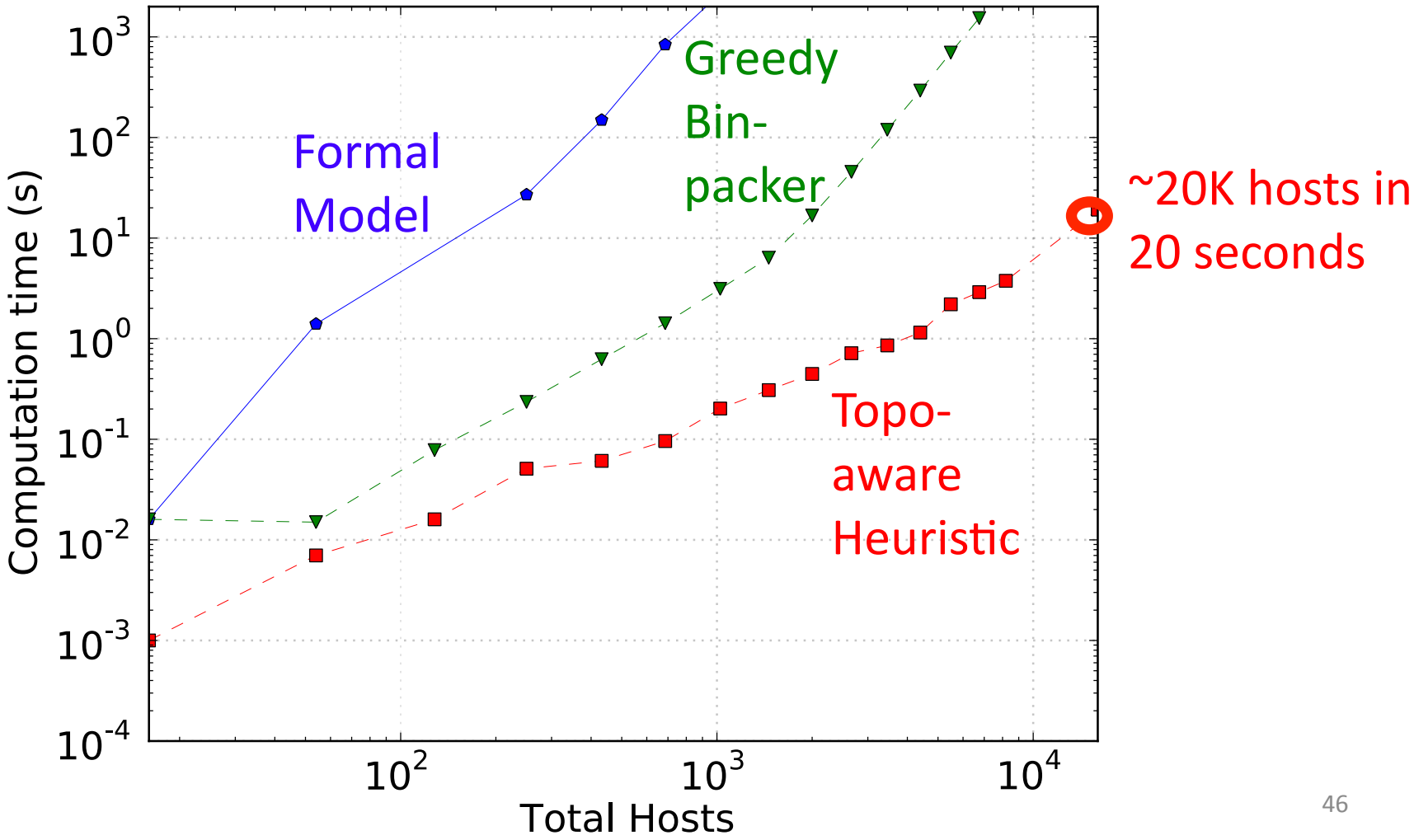
Decoupled Optimization and Routing

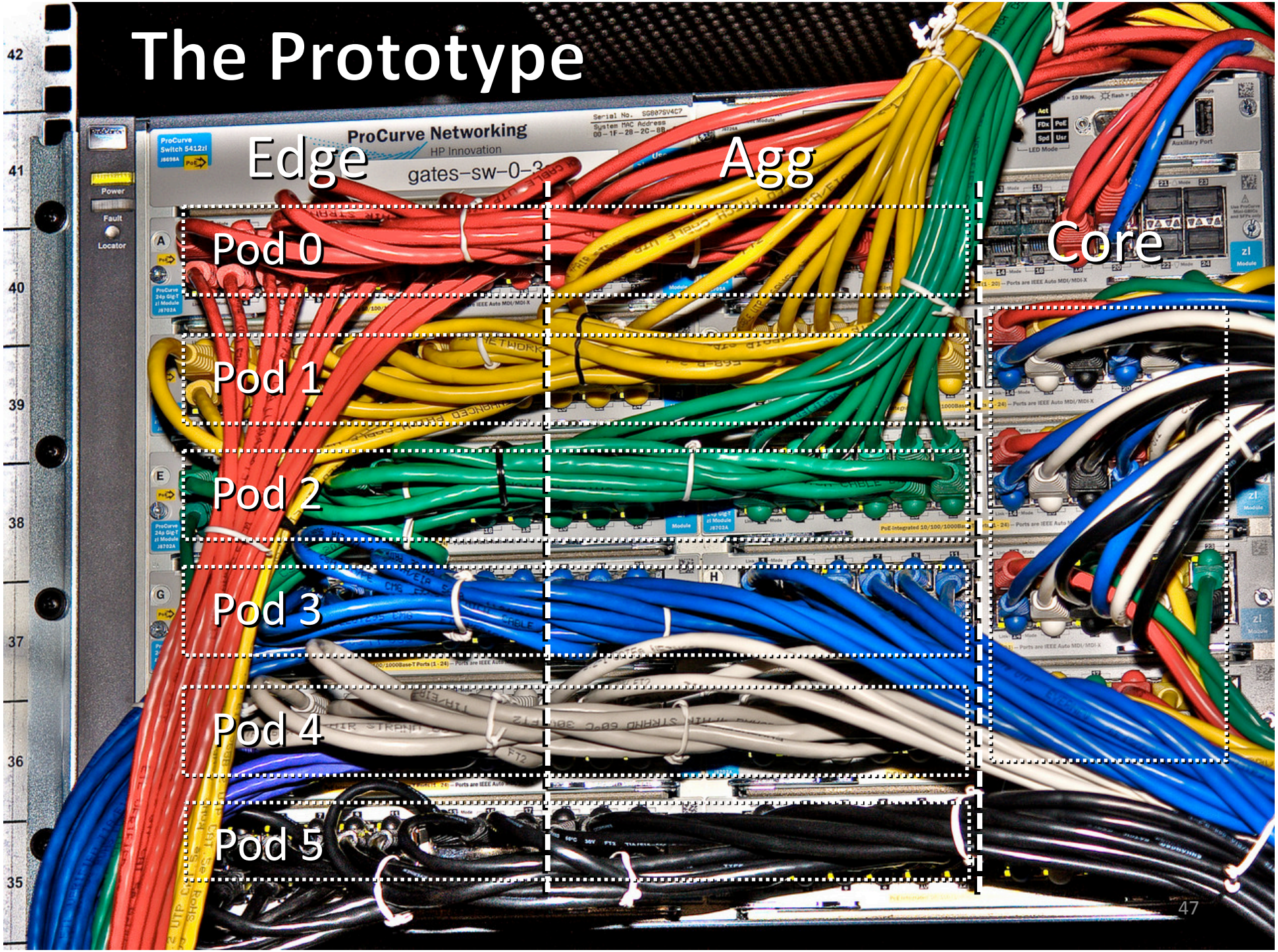


Optimizer Comparison

Type	Quality	Topo	Scalability	Input
Formal	Optimal	Any	Poor ~O($n^{3.5+}$)	Traffic Matrix
Greedy	Good	Any	Good ~O(n^{2+})	Traffic Matrix
Topo-aware	OK	Structured	Best O(n)	Port Counters

Scalability




The Prototype
Edge
Agg
Core
Pod 0
Pod 1
Pod 2
Pod 3
Pod 4
Pod 5

Example

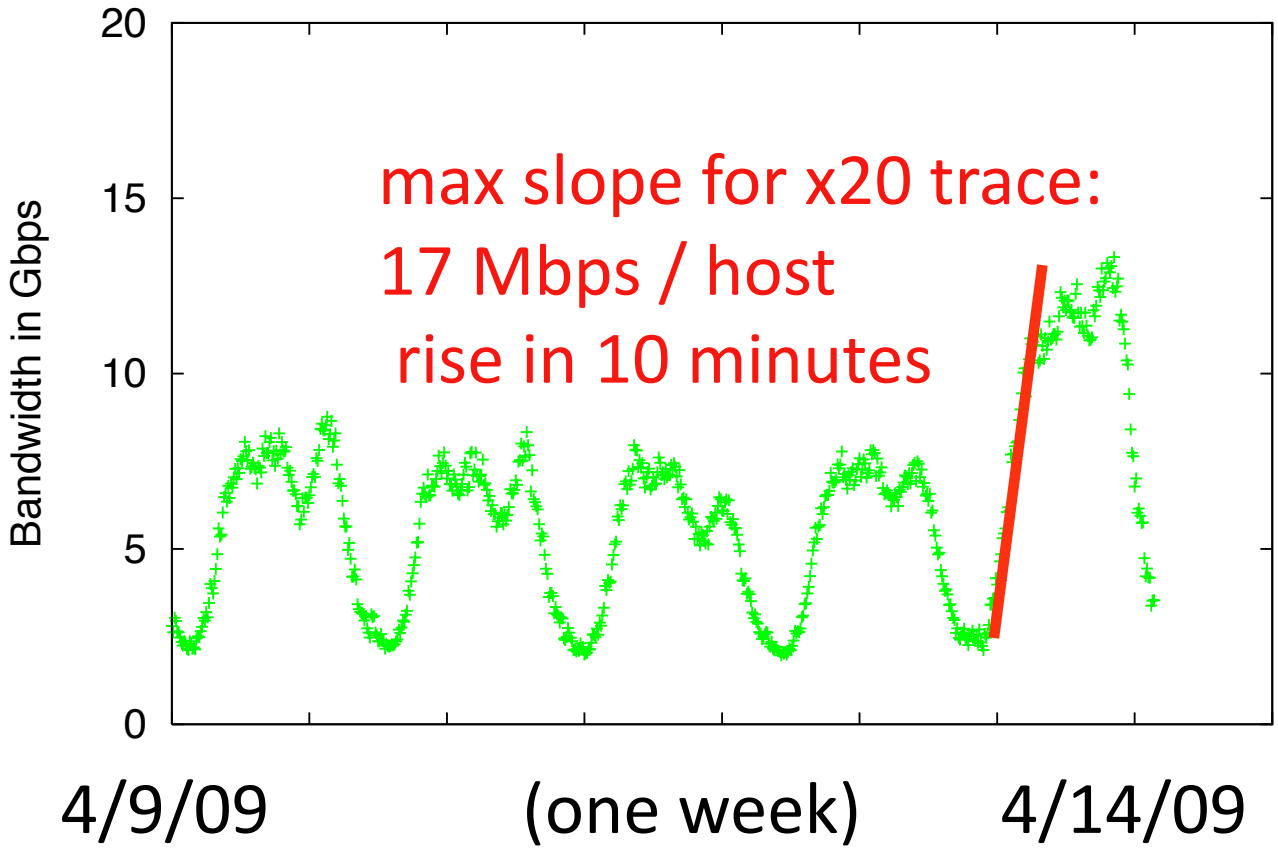
Your goal: configure ElasticTree to

- Minimize power (duh!)
- Handle bursts 5x larger than those seen during one-week traffic history
- Increase latency no more than 10%
- Maintain equivalent fault tolerance to 2N tree

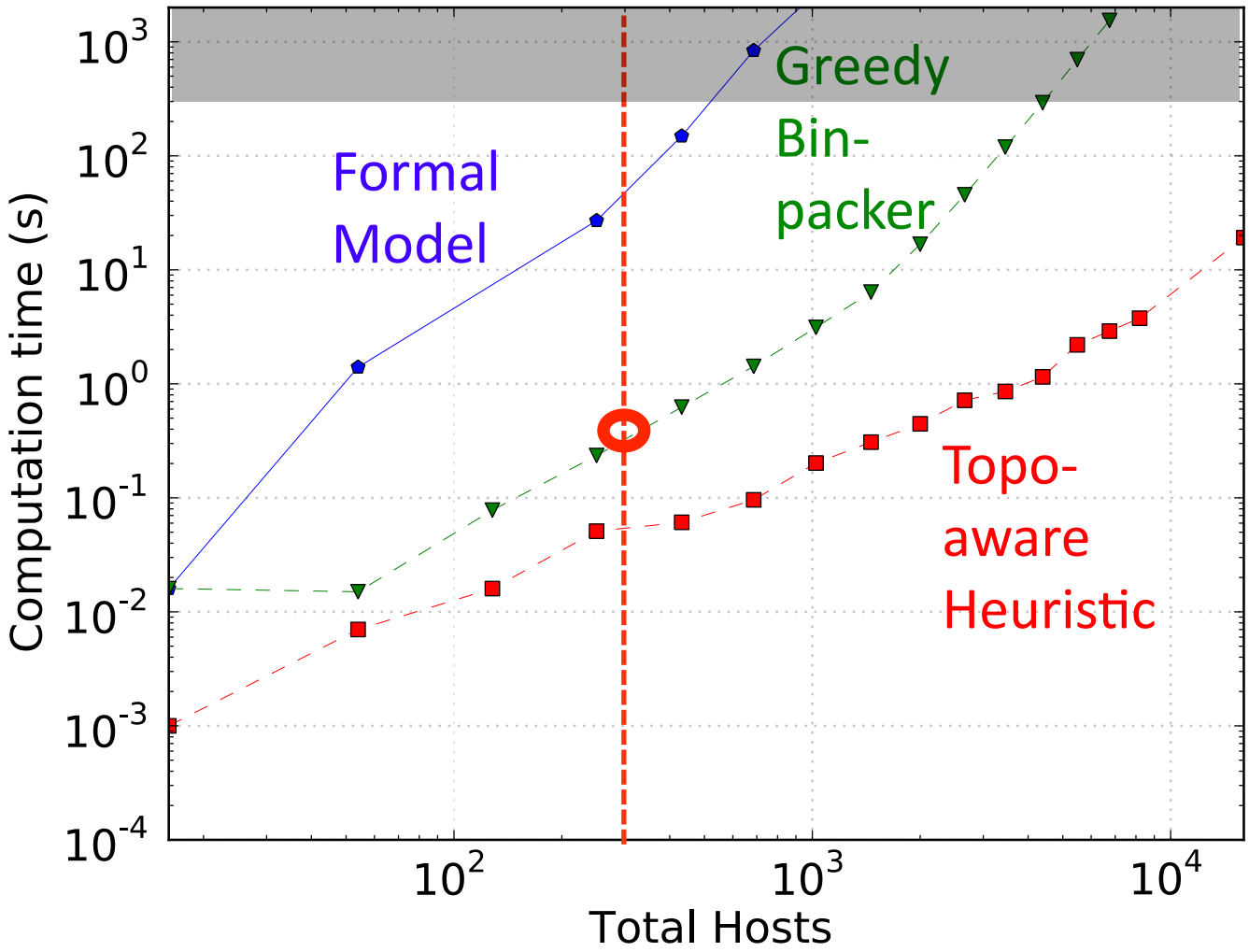
You have:

- 300 servers
- Switches with 5-minute boot time

(1) How fast does traffic change?

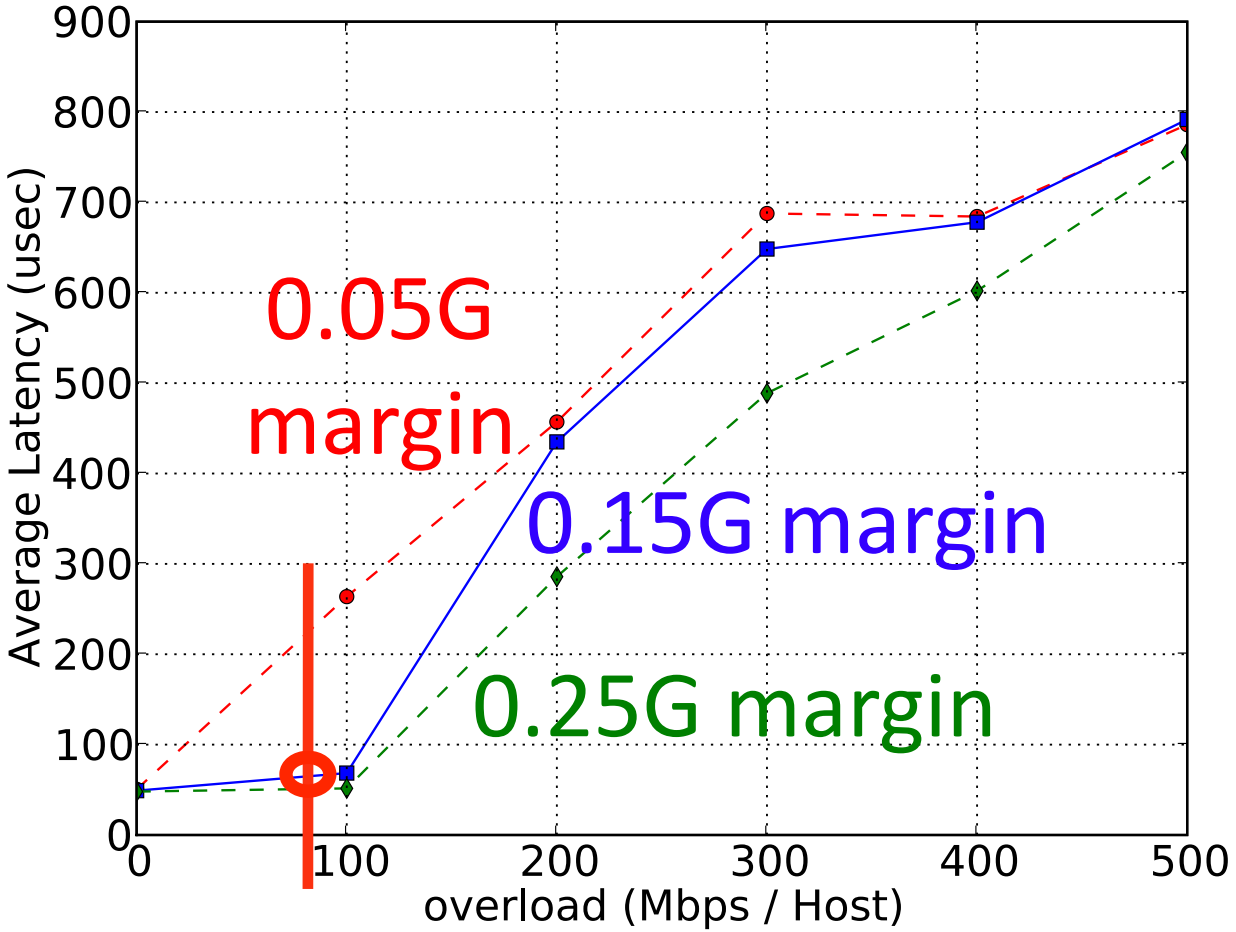


(2) Which Optimizer?



(3) What utilization bound?

17 Mbps rise in 10 min
5x faster rise = 85 Mbps



What makes this real?

- Better understanding of:
  - algorithmic tradeoffs
  - application behavior
  - protocol interactions
- Switches with sleep mode

Summary

Contributions

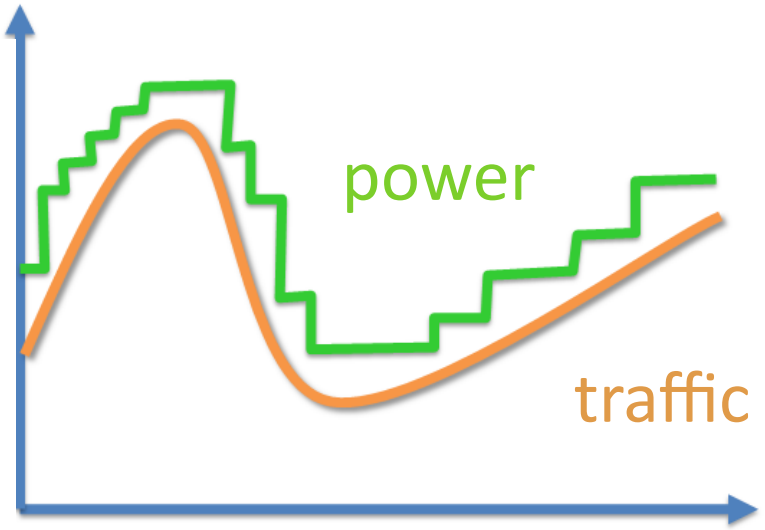
Identified an opportunity

Three algorithms: Model, Greedy, Heuristic

Initial understanding of tradeoffs

Running prototype

ElasticTree is a first step toward an **energy-proportional data center network** from non-proportional components.



Switch sleep makes it practical.

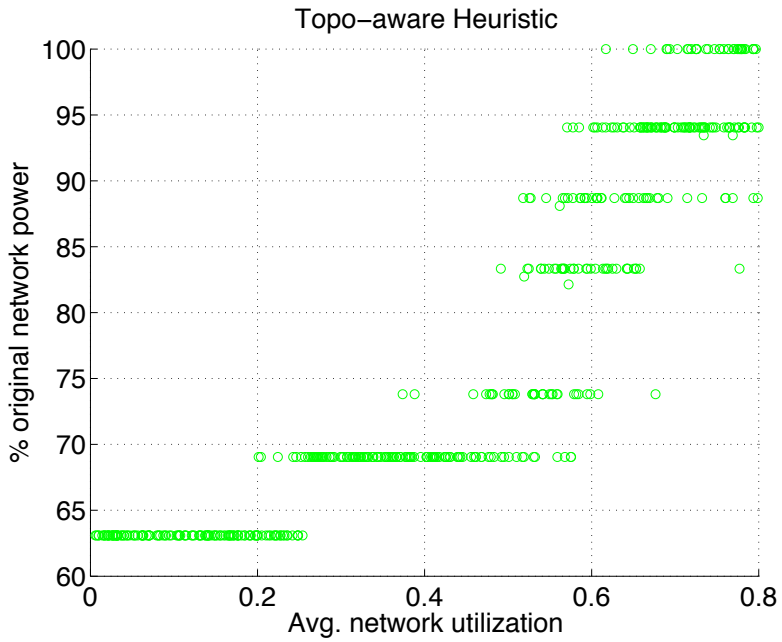
Proportional components are even better.

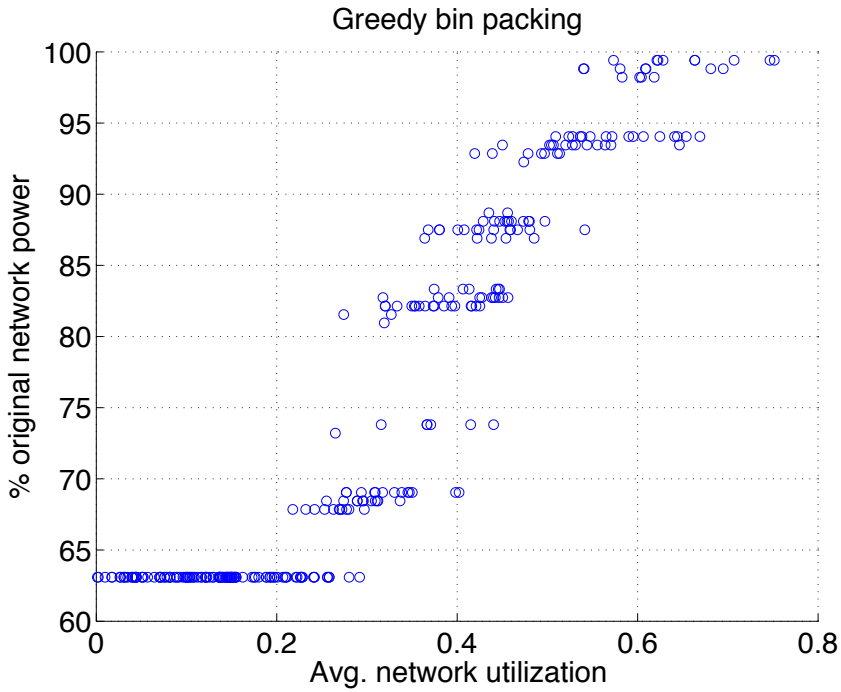
Acknowledgements

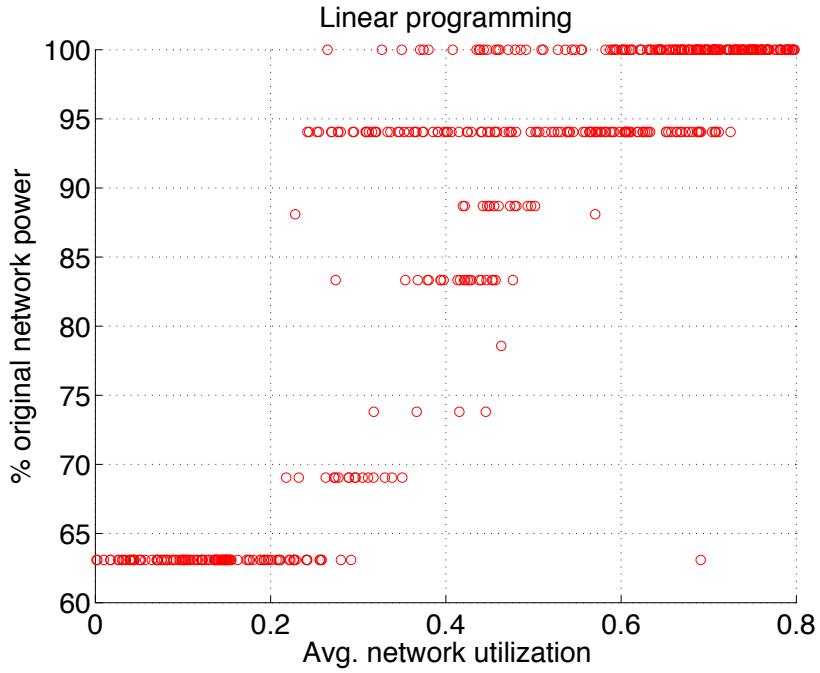
- Ant Rowstron (MSR)
- Xiaoyun Zhu (VMWare)
- Ram Swaminathan (HP Labs)
- Partha Ranganathan (HP Labs)
- David Underhill (Stanford)
- HP, NEC, and Stanford OpenFlow Teams

Questions

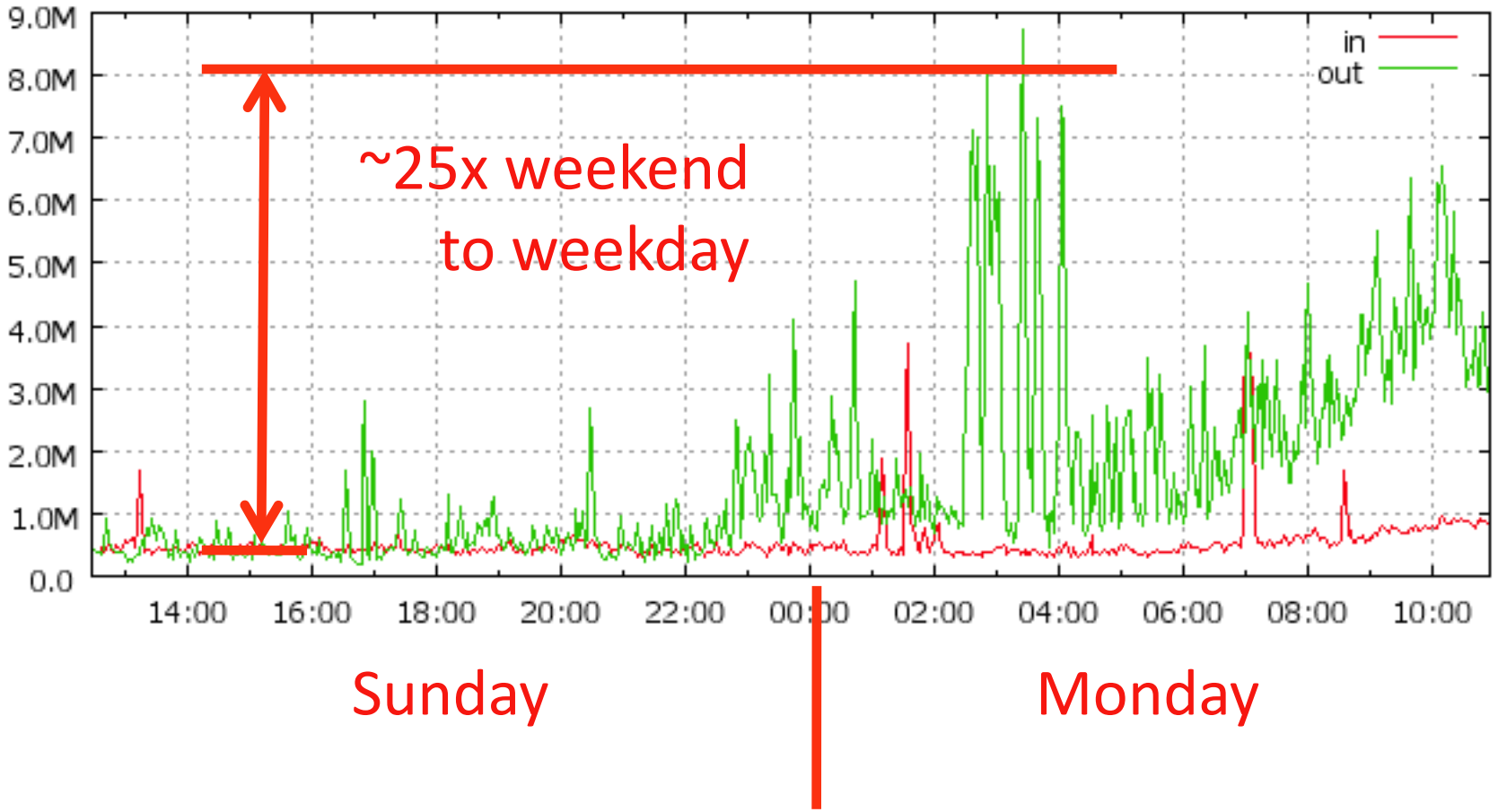
Backup Slides


Topo−aware Heuristic
% original network power
Avg. network utilization
100
95
90
85
80
75
70
65
60
0
0.2
0.4
0.6
0.8


Greedy bin packing
% original network power
Avg. network utilization
100
95
90
85
80
75
70
65
60
0
0.2
0.4
0.6
0.8


Linear programming
% original network power
Avg. network utilization
100
95
90
85
80
75
70
65
60
0
0.2
0.4
0.6
0.8

Google Router port



Formal Model

Capacity constraints: $\sum_{i=1}^{k} f_i(u,v) \leq c(u,v)$

The sum total flow of all commodities in each link must not exceed the edge capacity.

Flow conservation: $\sum_{w \in V} f_i(u,w) = 0 \quad \text{when} \quad u \neq s_i, t_i$

Commodities are neither created nor destroyed at intermediate nodes.

Demand satisfaction: $\sum_{w \in V} f_i(s_i,w) = \sum_{w \in V} f_i(w,t_i) = d_i$

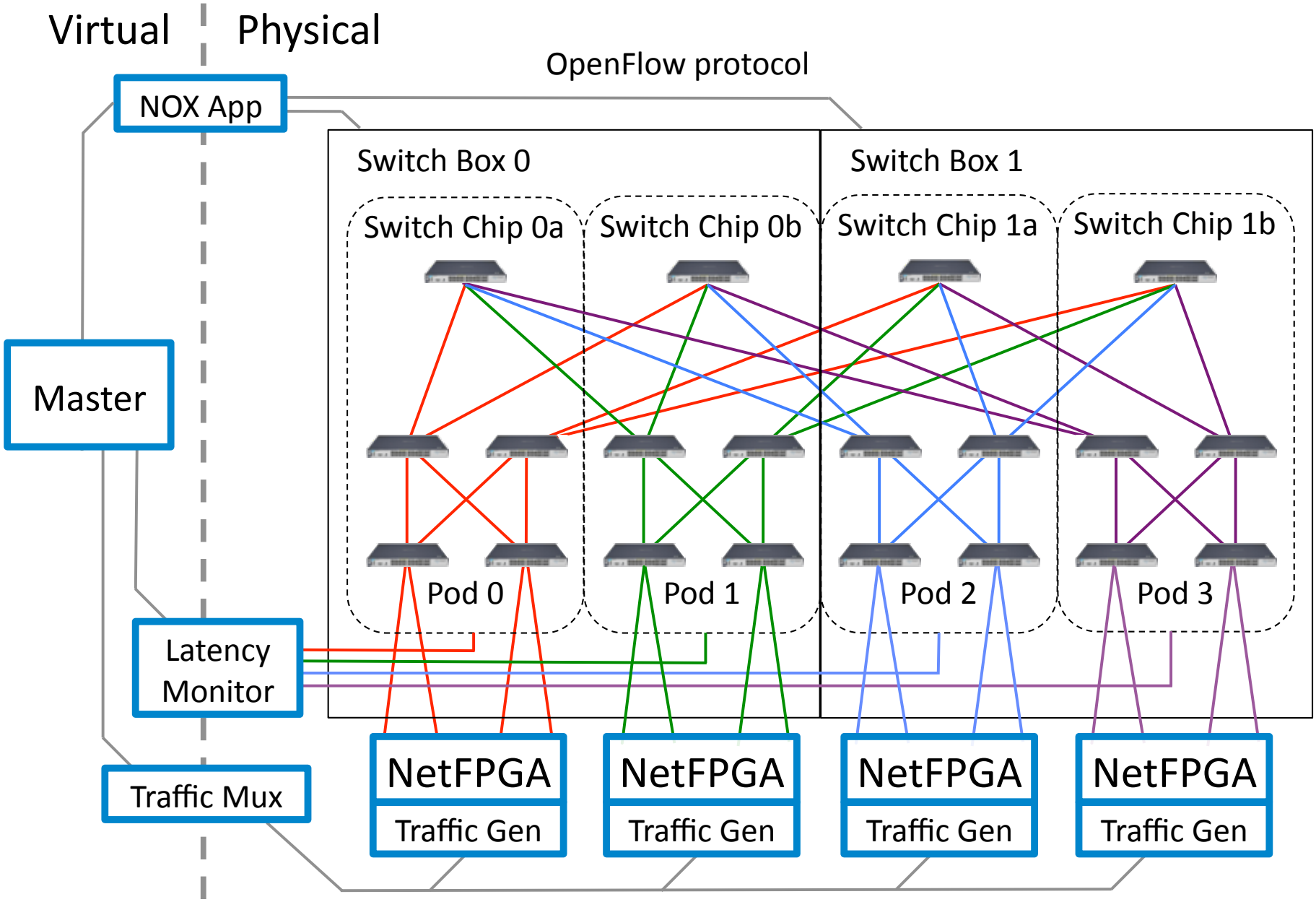
Each source and sink sends or receives an amount equal to its demand.

Deactivated links have no traffic: $X_{u,v} = 0 \rightarrow f_i(u,v) = 0, \forall i.$

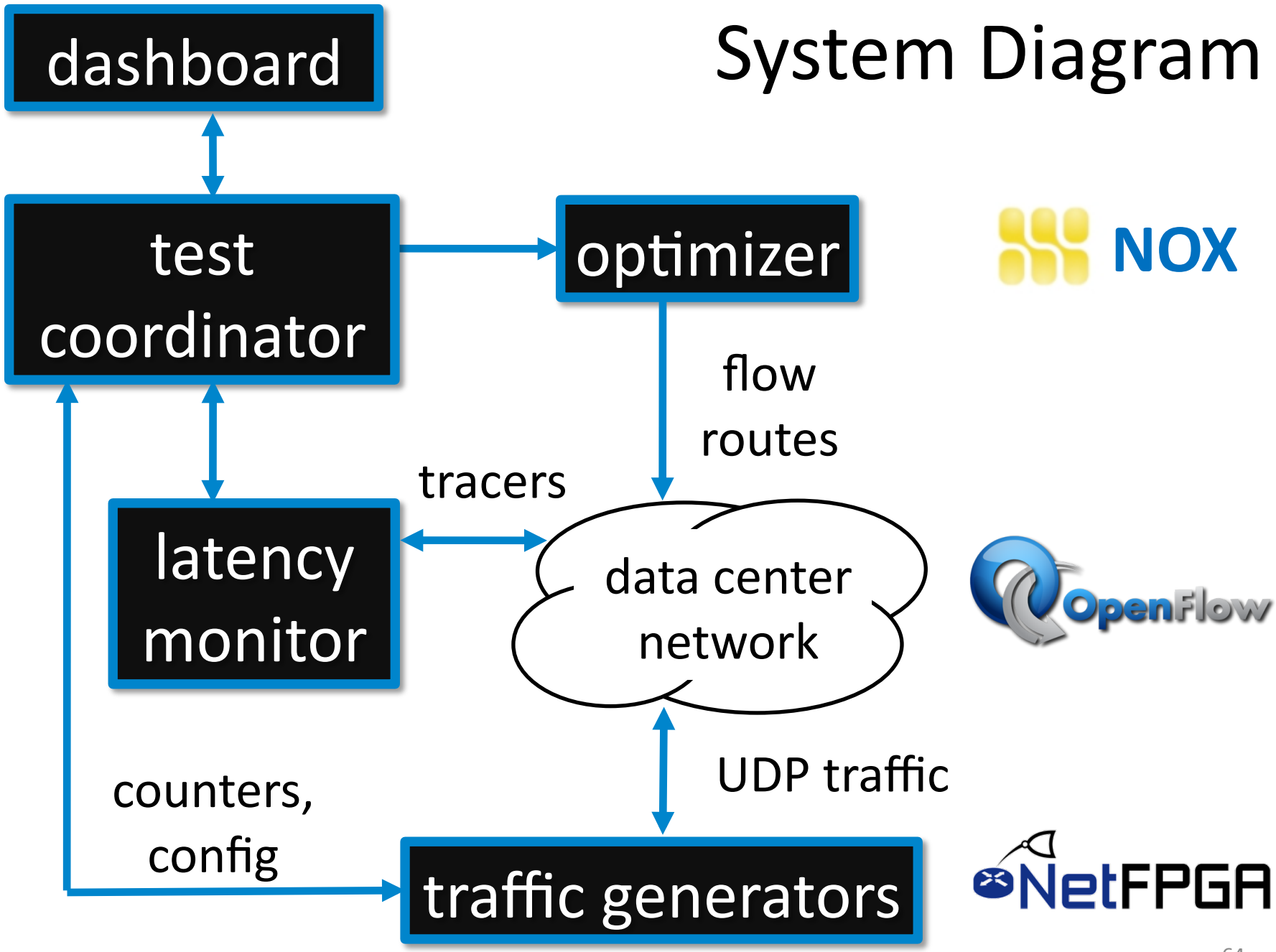
Deactivated switches have no active links: $\forall u \in V, Y_u = 0 \rightarrow \sum_{w \in V} X_{w,u} = 0.$

Both "halves" of a link must be turned on if traffic is flowing in either direction of a physical link: $\forall (u,v) \in E, X_{u,v} = X_{v,u}.$

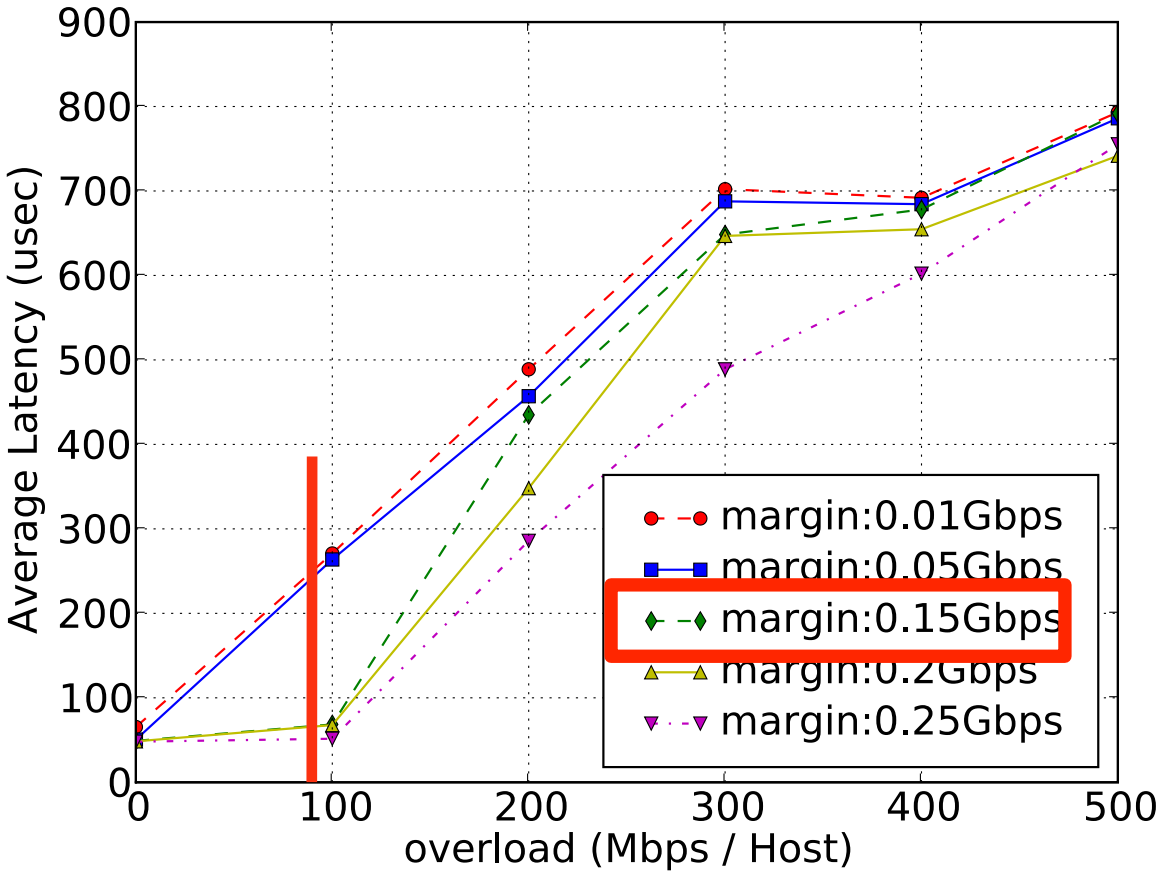
Minimize $\sum_{(u,v) \in E} X_{u,v} \times a(u,v) + \sum_{u \in V} Y_u \times b(u)$


Virtual
Physical
OpenFlow protocol
NOX App
Switch Box 0
Switch Box 1
Switch Chip 0a
Switch Chip 0b
Switch Chip 1a
Switch Chip 1b
Master
Pod 0
Pod 1
Pod 2
Pod 3
Latency Monitor
Traffic Mux
NetFPGA
Traffic Gen
NetFPGA
Traffic Gen
NetFPGA
Traffic Gen
NetFPGA
Traffic Gen

System Diagram



latency under extra load



Decoupled Optimization and Routing