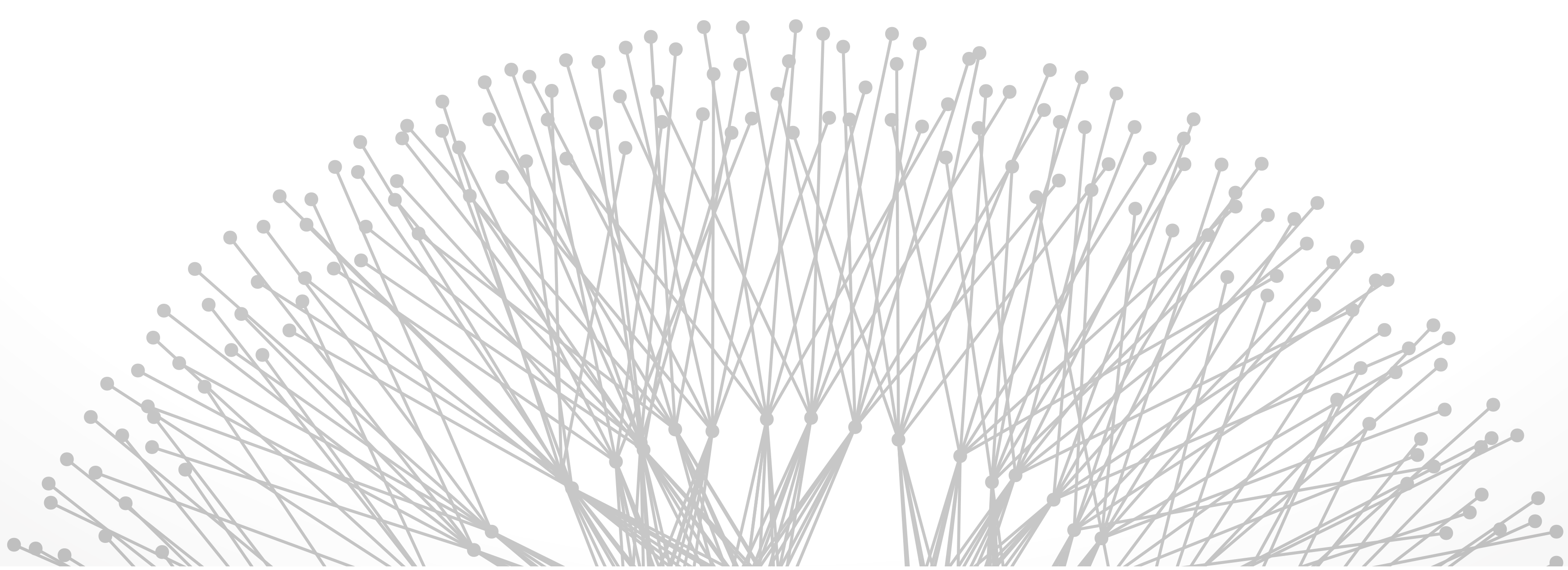
Data Center Networks

Brighten Godfrey

CS 538 October 29 2013



Introduction:
The Driving Trends

Cloud Computing

Cloud Computing: Computing as a utility

- Purchase however much you need, whenever you need it
- Service ranges from access to raw (virtual) machines, to higher level: distributed storage, web services

Implications

- Reduces barrier to entry to building large service
  - No need for up-front capital investment
  - No need to plan ahead
- May reduce cost
- Compute and storage becomes more centralized

The physical cloud: Data centers

Facebook data center, North Carolina

National Petascale Computing Facility, UIUC


google.com/datacenters


google.com/datacenters

Key advantage: economy of scale

One technician for each 15,000 servers [Facebook]

Facility / power infrastructure operated in bulk

- Power usage efficiency (PuE) ~ 1.8 in average DCs
- Pushed down to ~ 1.1 in large cloud DCs

Ability to custom-design equipment

- Facebook (servers), Google (servers & networking gear)

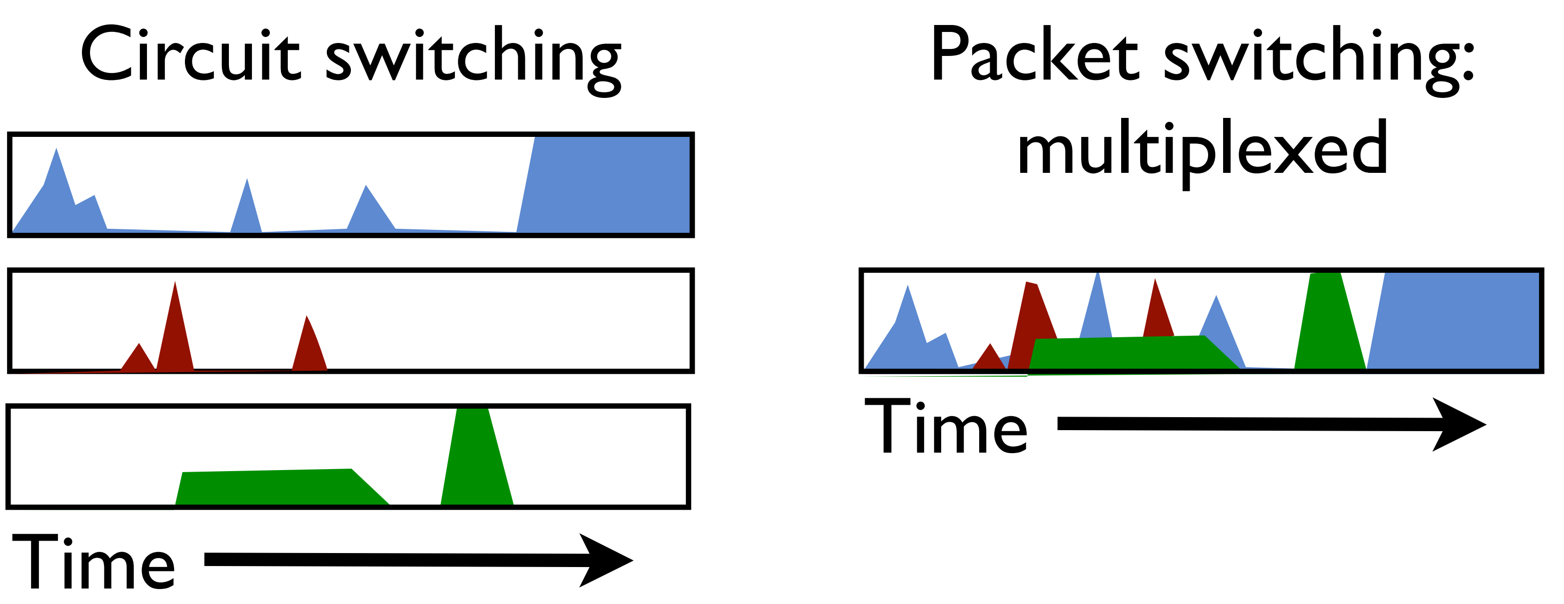
Statistical multiplexing

- Must provision for peak load
- Many users sharing a resource are unlikely to have their peaks all at the same time

Key advantage: economy of scale

Statistical multiplexing

- Must provision for peak load
- Many users sharing a resource are unlikely to have their peaks all at the same time
- Just as in packet switching



Challenges for Cloud Computing

Challenges

- Confidentiality of data and computation
- Isolation of resources
- Integration with existing systems
- Robustness
- Latency
- Bandwidth
- Programmability
- ...

Opportunities

- New systems and architectures
- Optimizations matter

Costs in a data center

Servers are expensive!

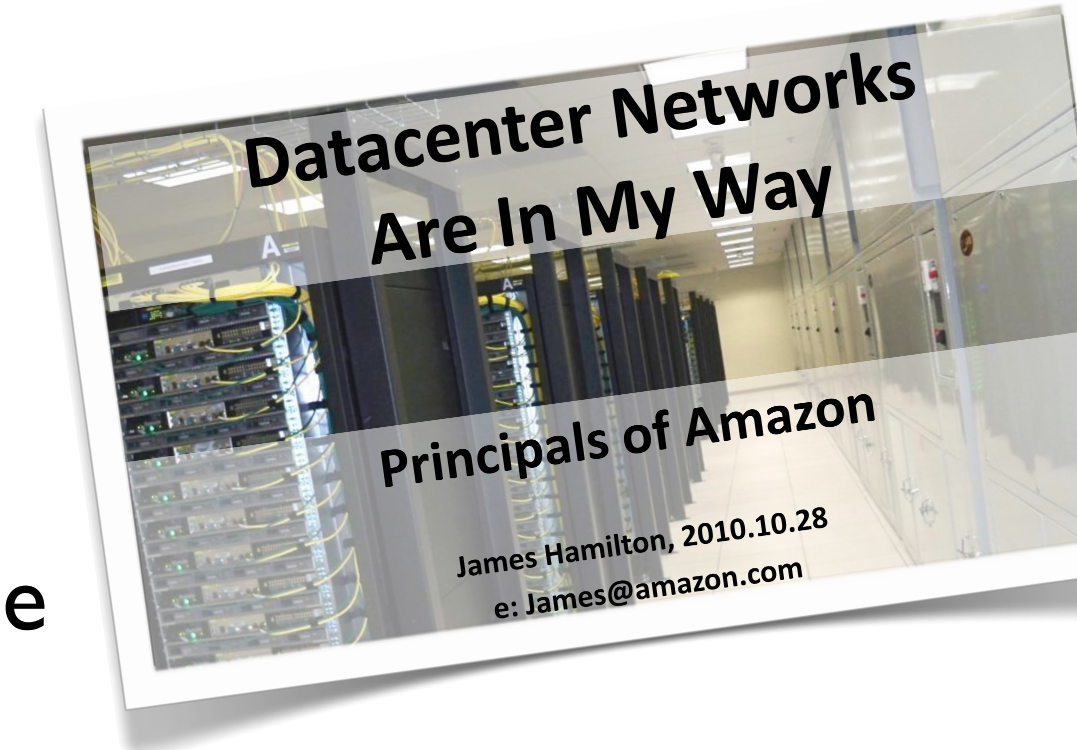
Amortized Cost	Component	Sub-Components
~45%	Servers	CPU, memory, storage systems
~25%	Infrastructure	Power distribution and cooling
~15%	Power draw	Electrical utility costs
~15%	Network	Links, transit, equipment

[Greenberg, CCR Jan. 2009]

A key goal: Agility

Agility: Use any server for any service at any time

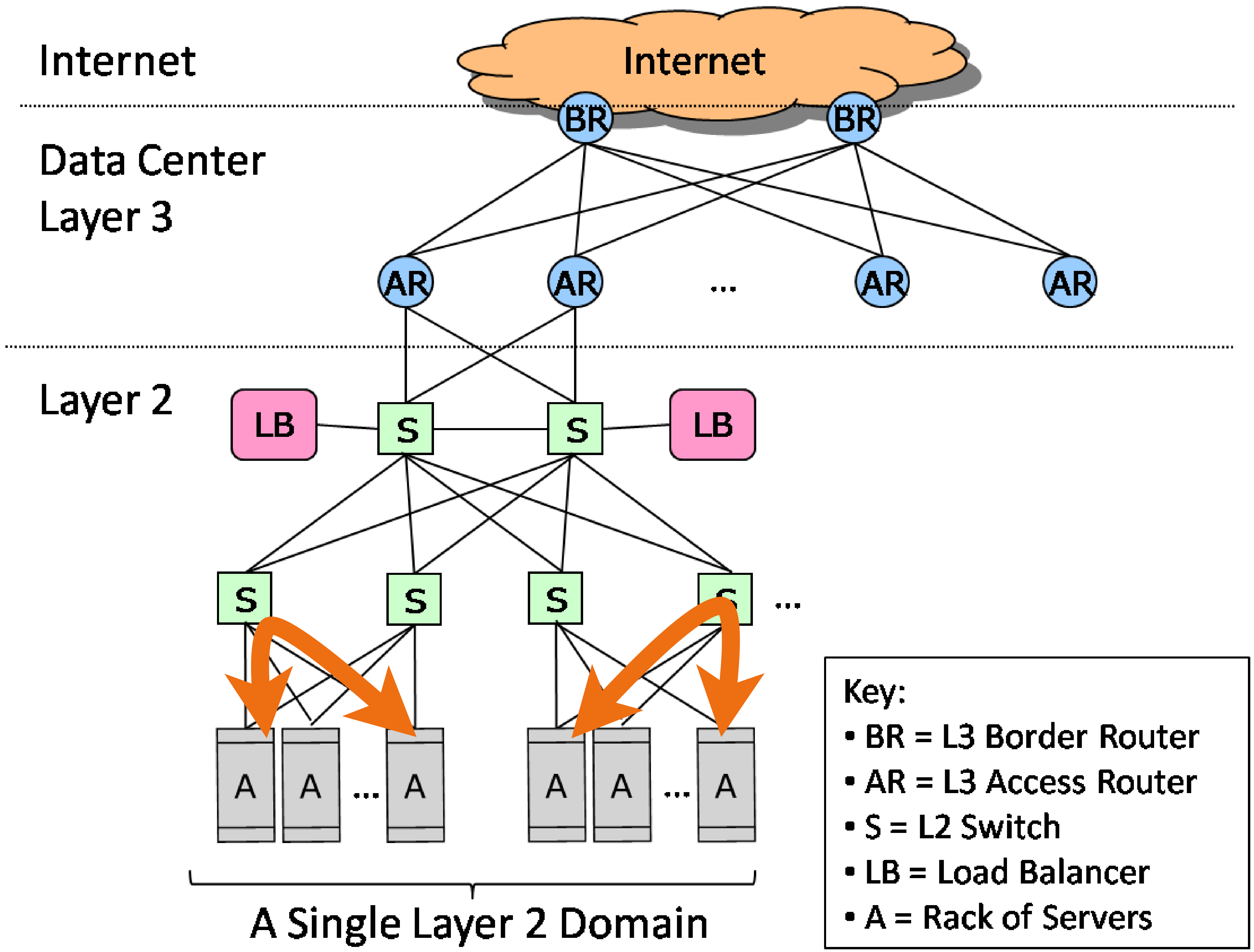
- Increase utilization of servers
- Reduce costs, increase reliability



What we need [Greenberg, ICDCS'09]

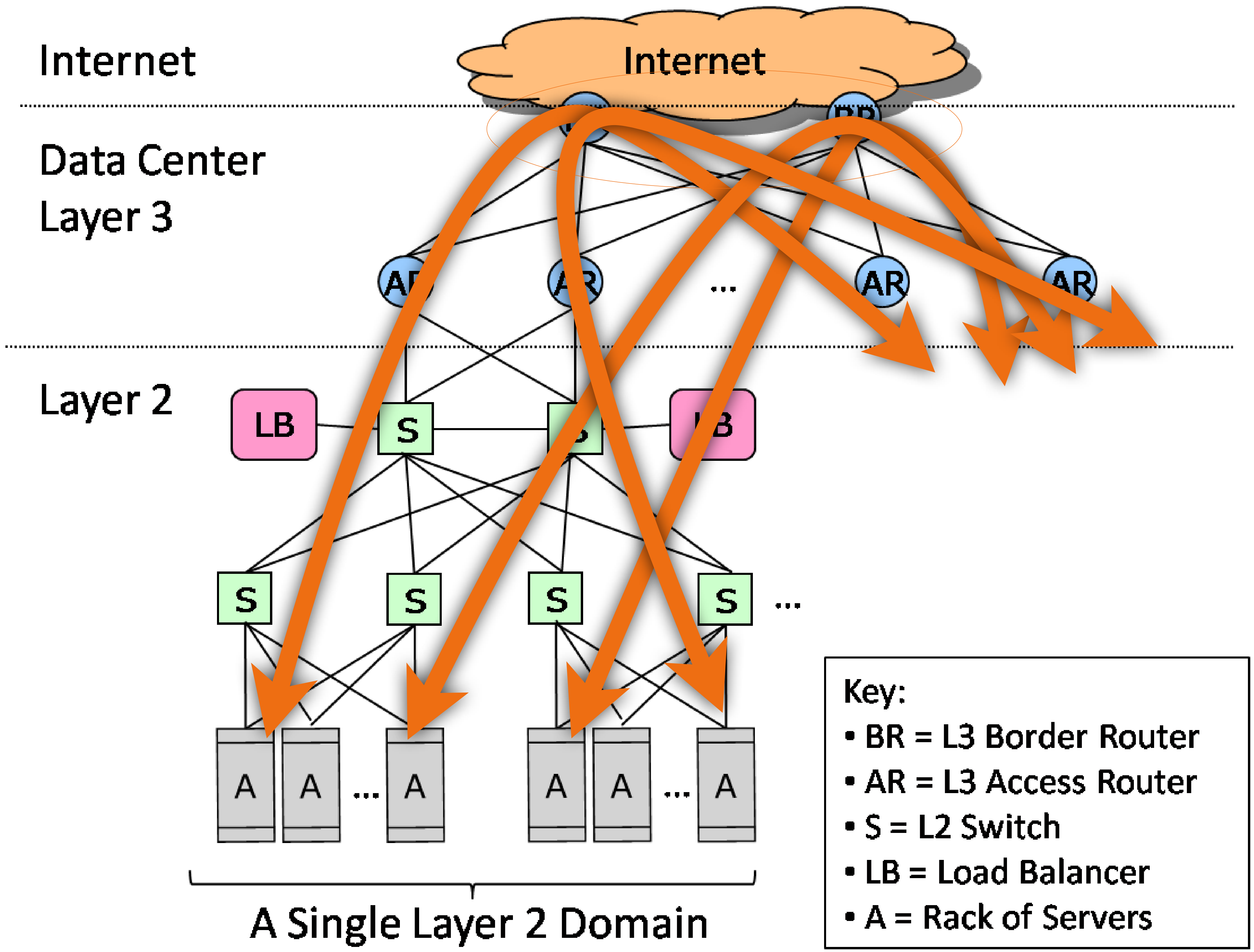
- Rapid installation of service's code
  - Solution: virtual machines
- Access to data from anywhere
  - Solution: distributed filesystems
- Ability to communicate between servers quickly, regardless of where they are in the data center

Traditional data center network



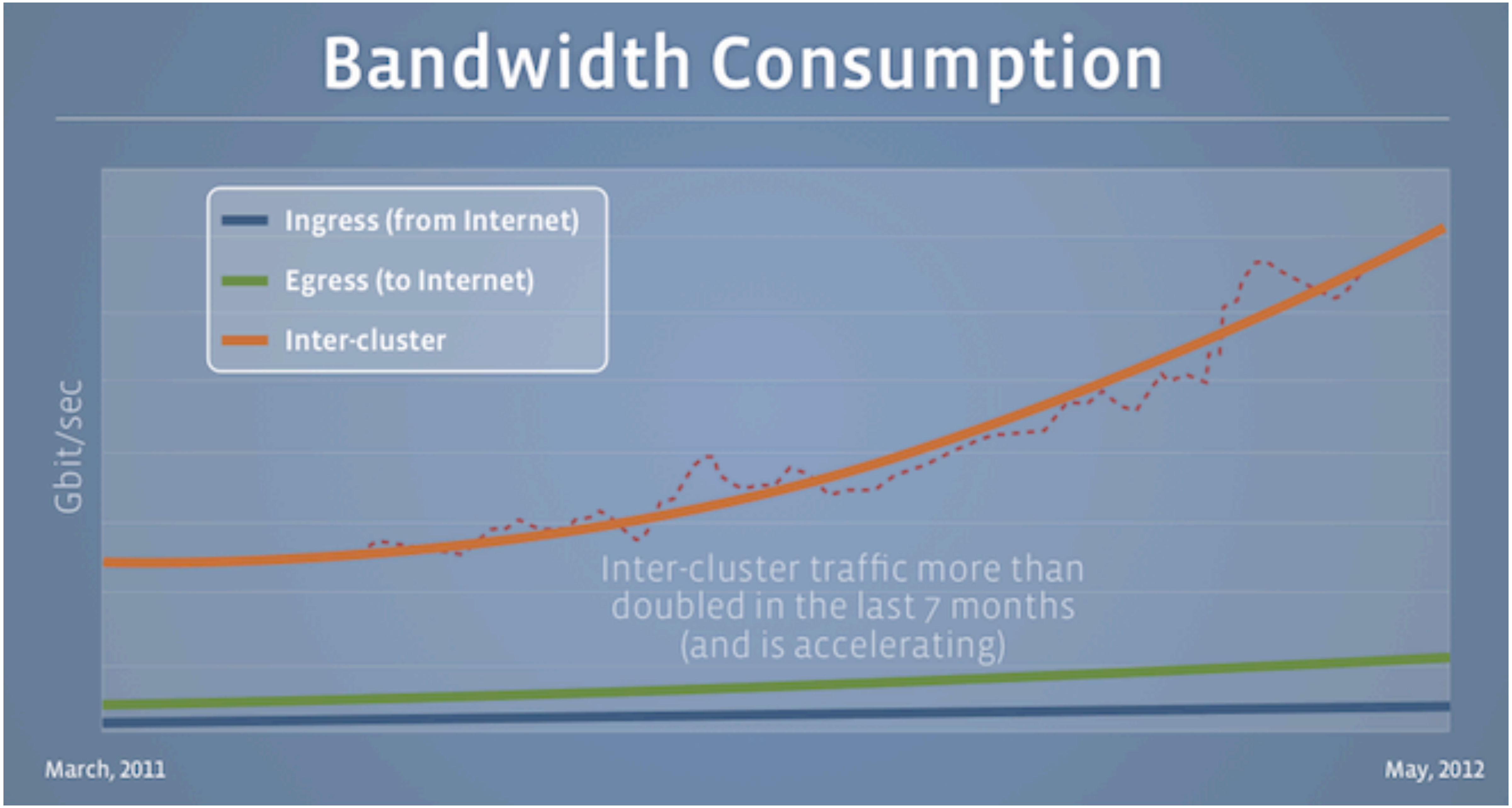
[Greenberg et al, CCR Jan. 2009]

Traditional data center network



[Greenberg et al, CCR Jan. 2009]

The need for performance



March 2011

May 2012

[Facebook, via Wired]

Modern Data Center Networks

Scalable, commodity DC net arch.

[Al-Fares, Loukissas, Vahdat, SIGCOMM 2008]

Argued for nonblocking bandwidth

- servers limited only by their network card's speed, regardless of communication pattern between servers
- also known as full throughput in the "hose model"

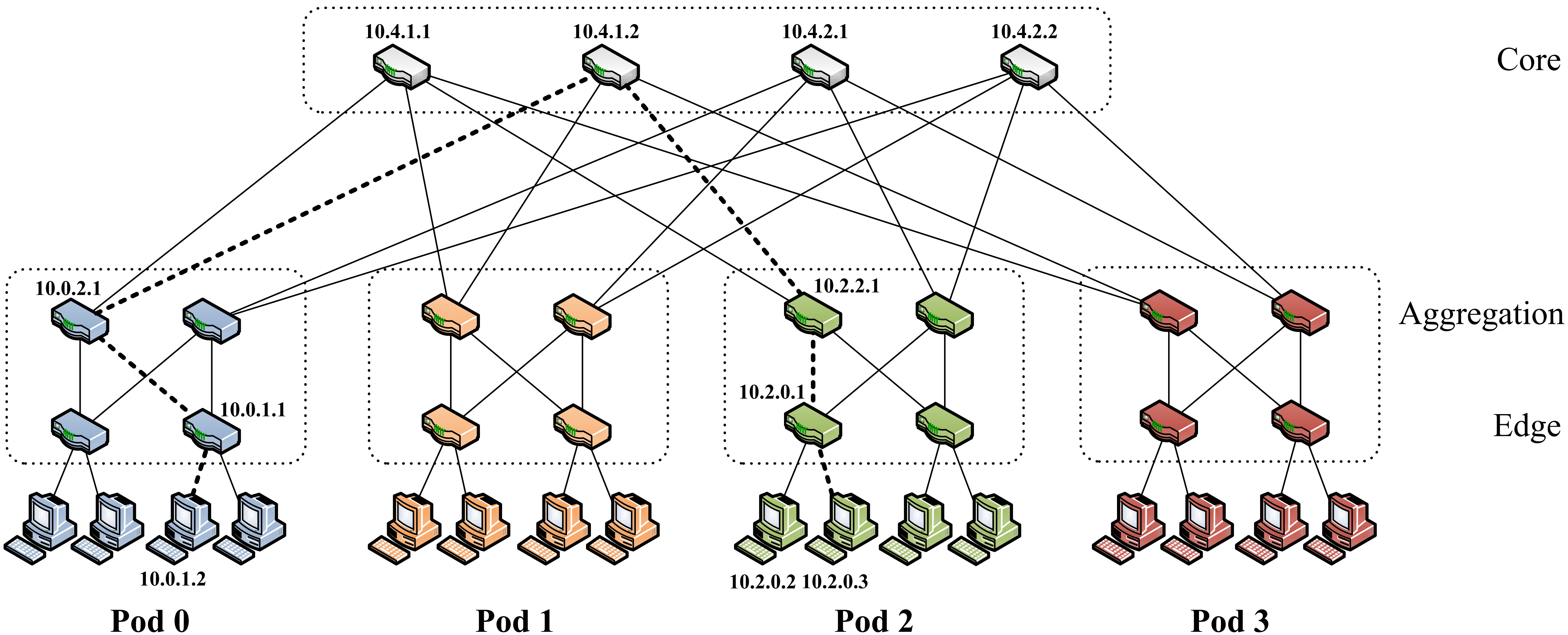
Employed large number of commodity switches

- rather than "big iron"

Arranged in Clos topology

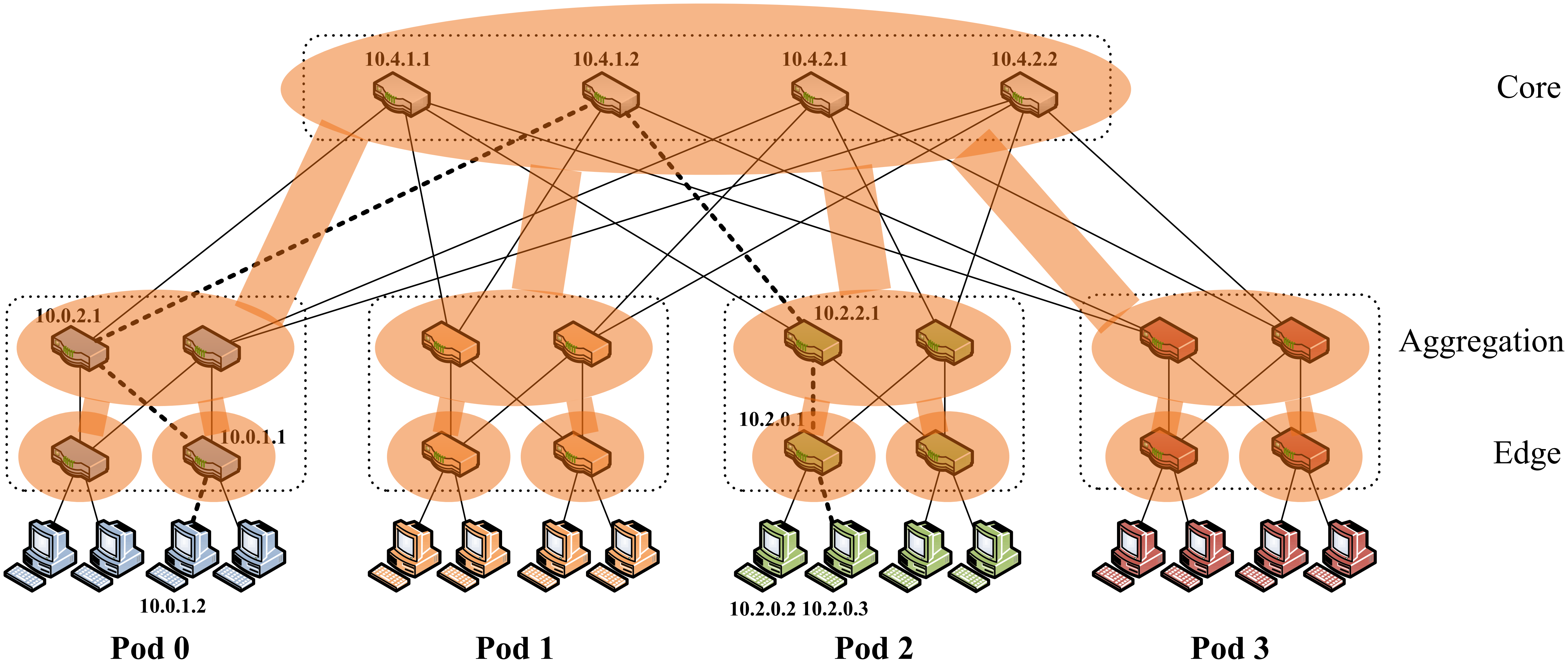
- specifically, a "fat tree"

Fat tree network



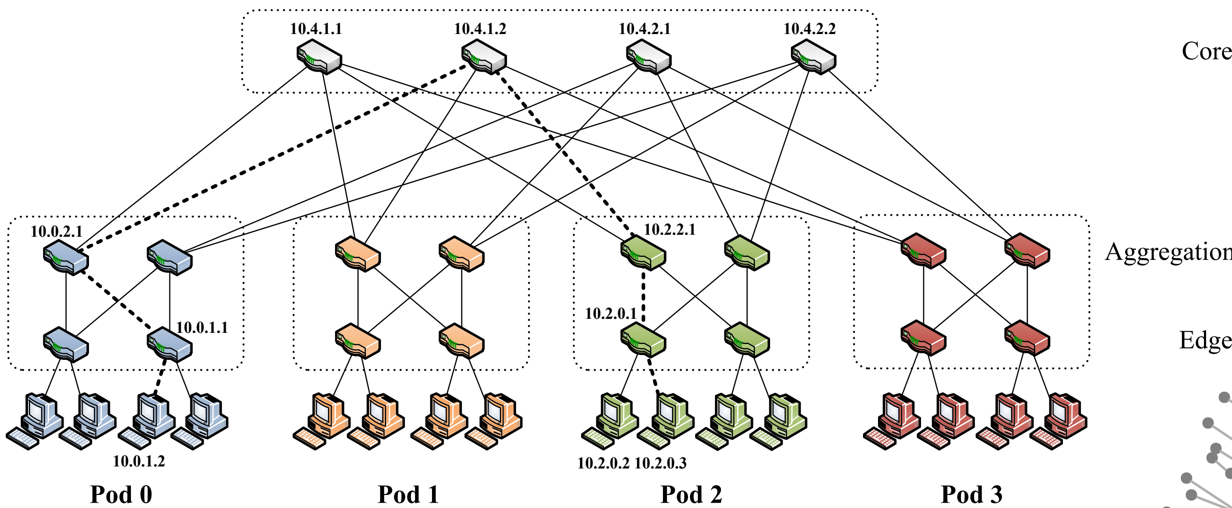
[Al-Fares, Loukissas, Vahdat, SIGCOMM '08]

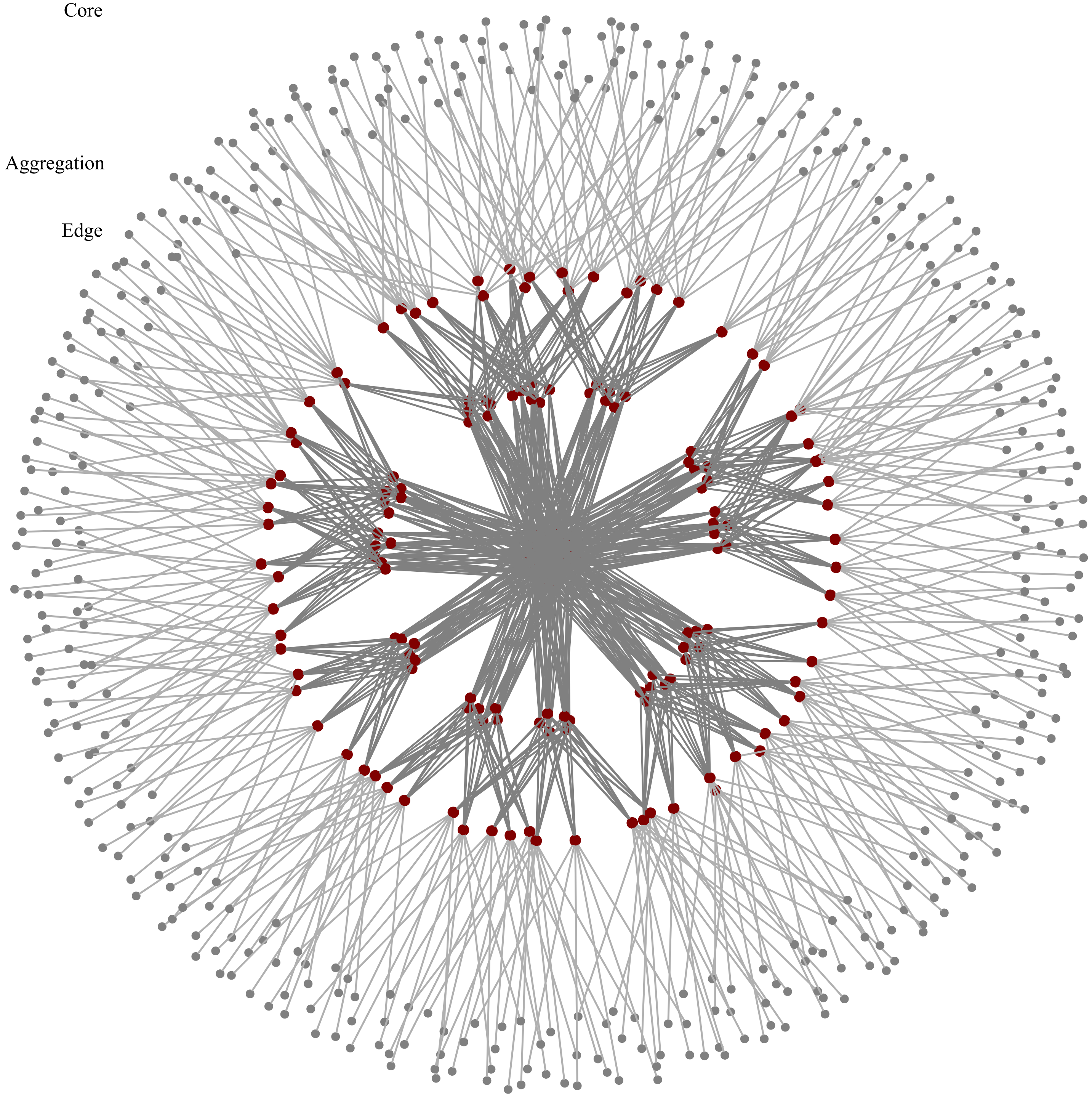
Fat tree network



[Al-Fares, Loukissas, Vahdat, SIGCOMM '08]

Fat tree network





VL2

[Greenberg, Hamilton, Jain, Kandula, Kim, Lahiri, Maltz, Patel, Sengupta, SIGCOMM 2009]

VL2 claims (in its title!) that it is "flexible". In what ways is VL2 flexible? How does it achieve these notions of flexibility?

VL2

[Greenberg, Hamilton, Jain, Kandula, Kim, Lahiri, Maltz, Patel, Sengupta, SIGCOMM 2009]

Key features

- High bandwidth network
  - Another folded Clos network
  - Slightly different than fat tree (e.g., uses 10 Gbps links)
- Randomized (Valiant) load balancing
  - Makes better use of network resources
- Flat addressing
  - Ethernet-style (layer 2) addresses to forward data, rather than IP addresses
  - Separates names from locations

VL2 discussion

Does VL2 need to adjust its randomized routing over time? [Yiying]

- More generally, how much could you improve routing?

How does VL2 compare to MPLS-based TE and 'fabric'?

Rest of this lecture:

Jellyfish

[Singla, Hong, Popa, Godfrey, NSDI 2012]
[Singla, Godfrey, Kolla, manuscript, 2013]

Two difficult goals

High throughput
with minimal cost

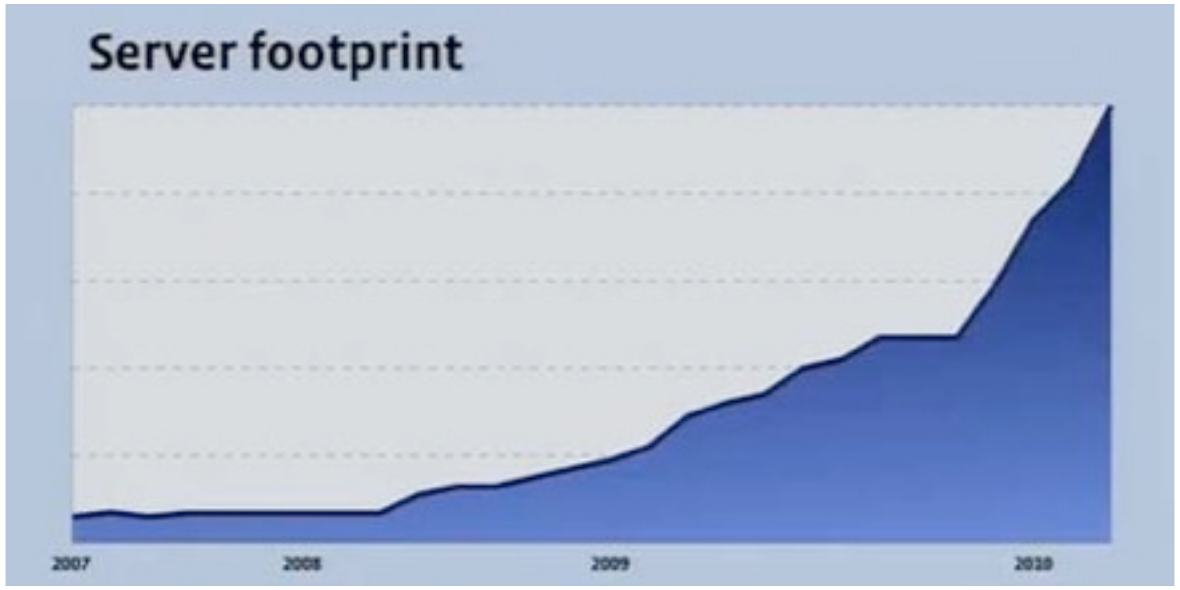
Support big data analytics
Agile placement of VMs

Flexible incremental
expandability

Easily add/replace
servers & switches

Incremental expansion

Facebook "adding capacity on a daily basis"



2007 08 09 10

Reduces up-front capital expenditure

Commercial products expand servers but not the net

- SGI Ice Cube ("Expandable Modular Data Center")
- HP EcoPod ("Pay-as-you-grow")

Structure hinders expansion

Coarse design points

- Hypercube: 2^k switches
- de Bruijn-like: 3^k switches
- 3-level fat tree: $5k^2/4$ switches

Fat trees by the numbers

- (3-level, with commodity 24, 32, 48, ... port switches)
- 3456 servers, 8192 servers, 27648 servers, ...

Unclear how to maintain structure incrementally

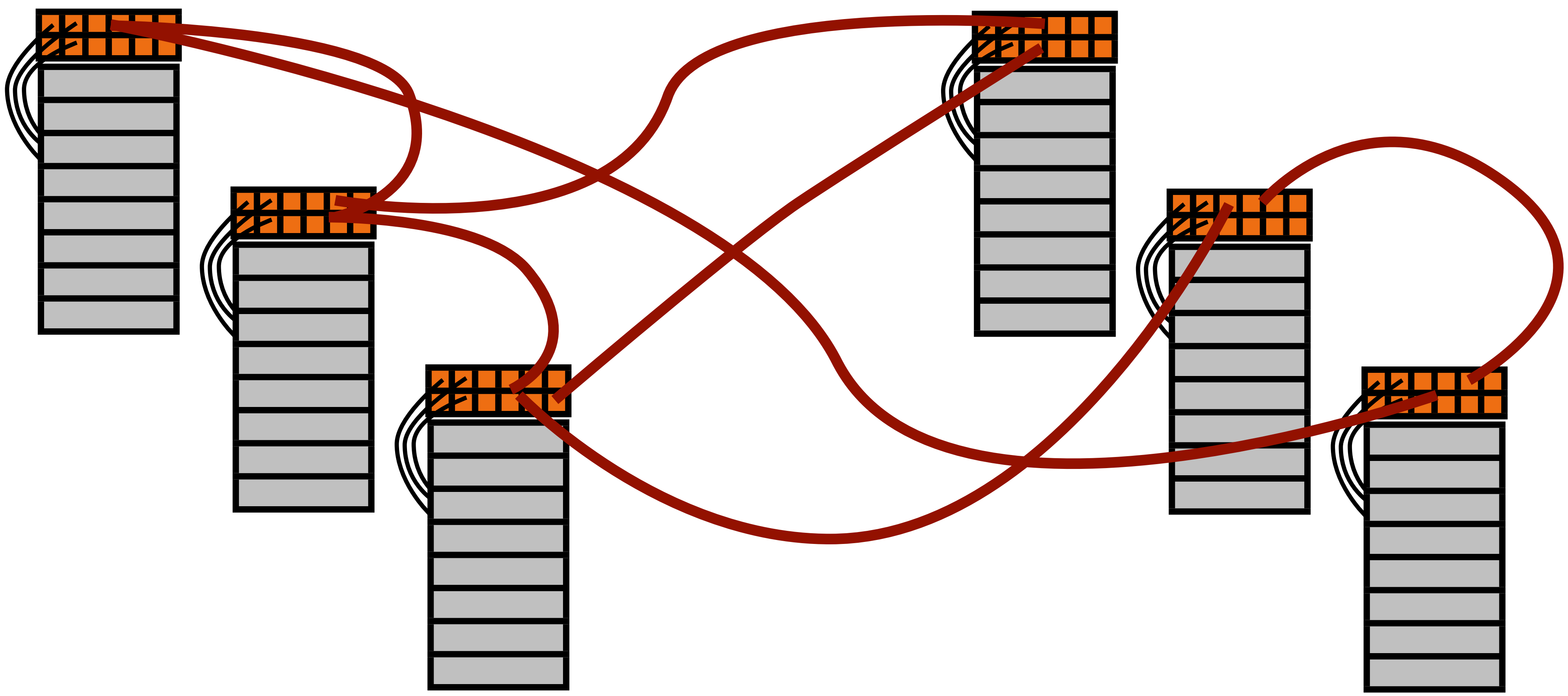
- Overutilize switches? Uneven / constrained bandwidth
- Leave ports free for later? Wasted investment

Our Solution

Forget about structure –
let's have no structure at all!

Jellyfish:
The Topology

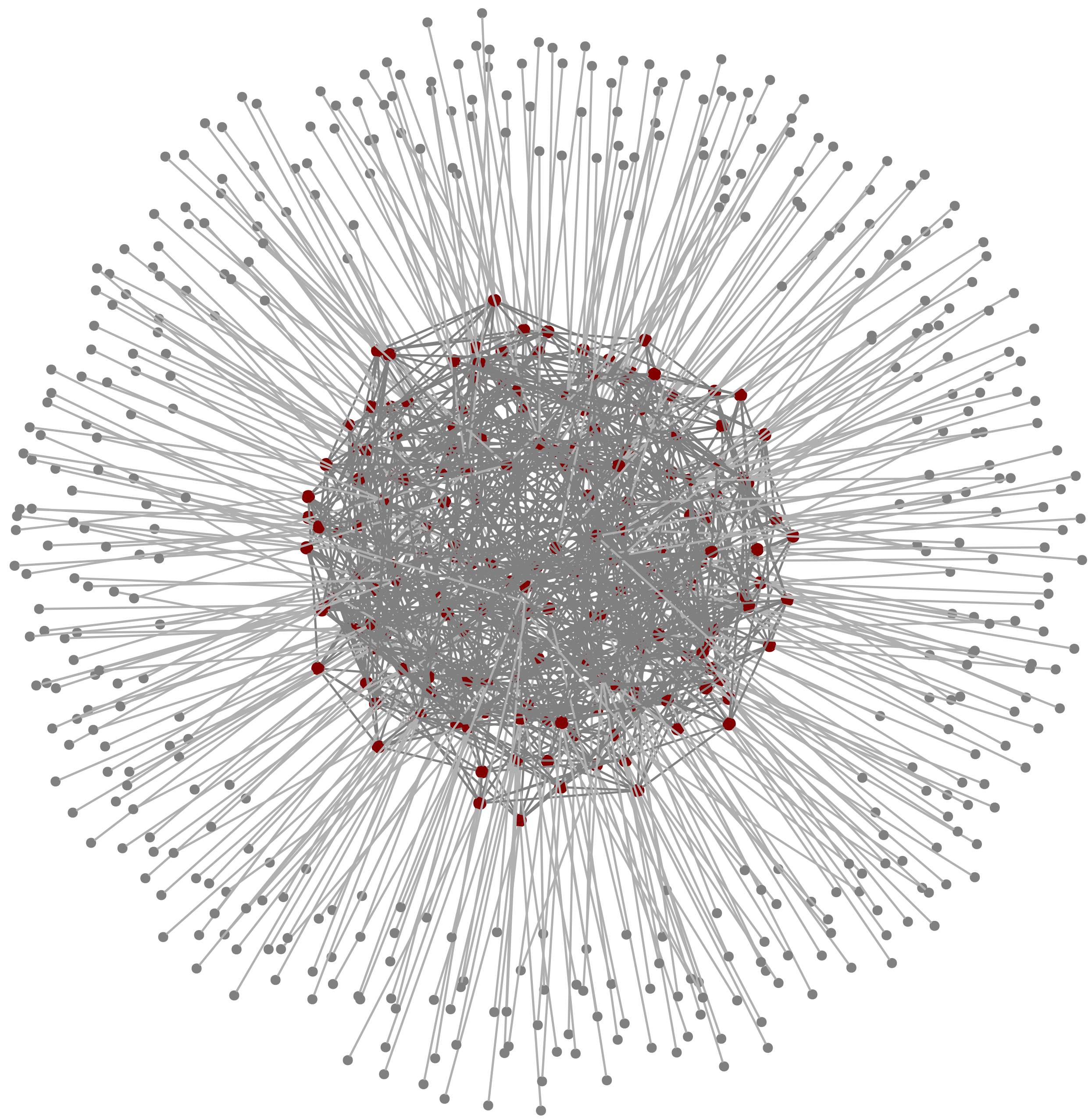
Jellyfish: The Topology



Servers connected to top-of-rack switch

Switches form uniform-random interconnections

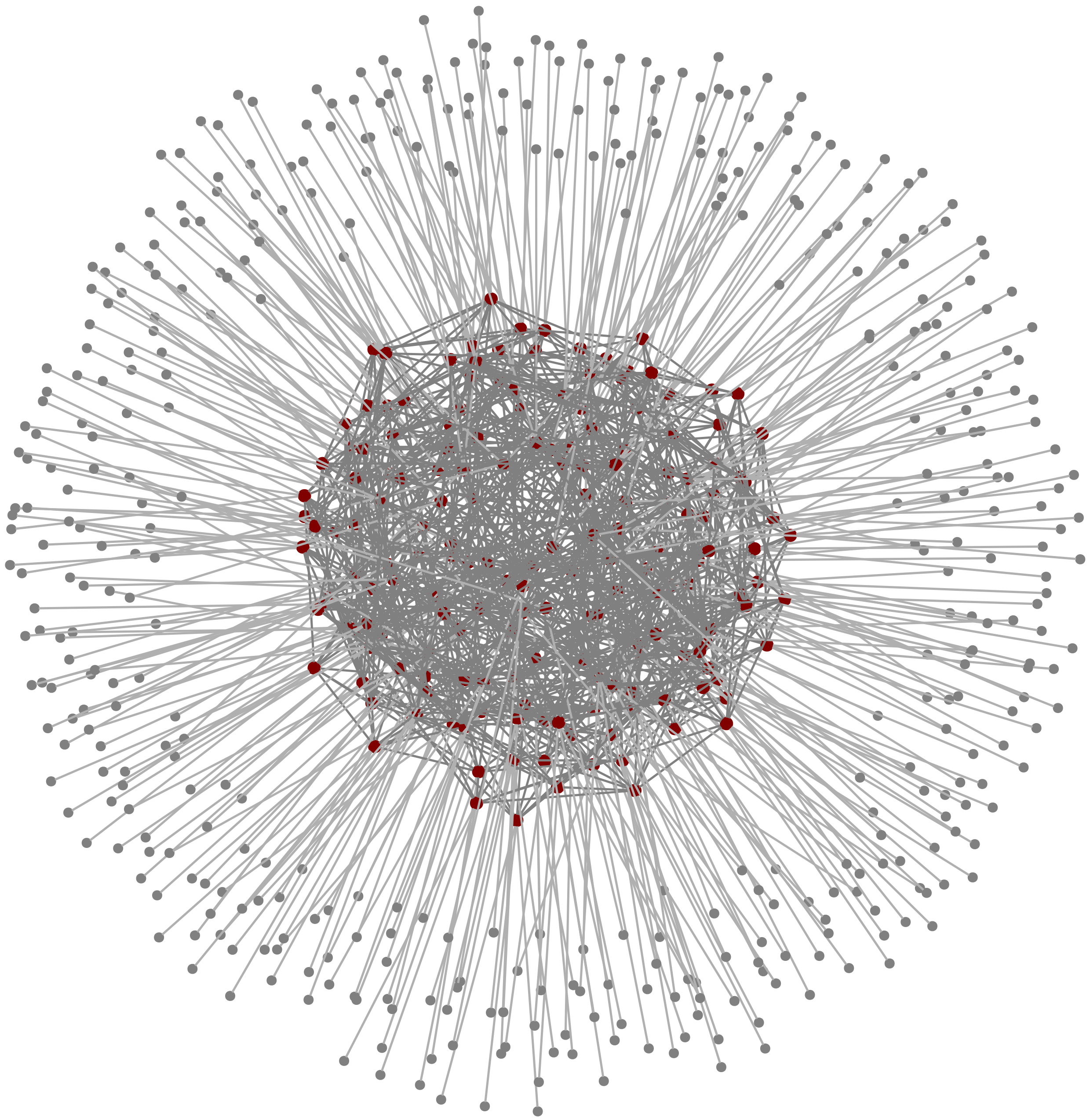
Capacity as a fluid



Jellyfish random graph

432 servers, 180 switches, degree 12

Capacity as a fluid



Jellyfish random graph

432 servers, 180 switches, degree 12

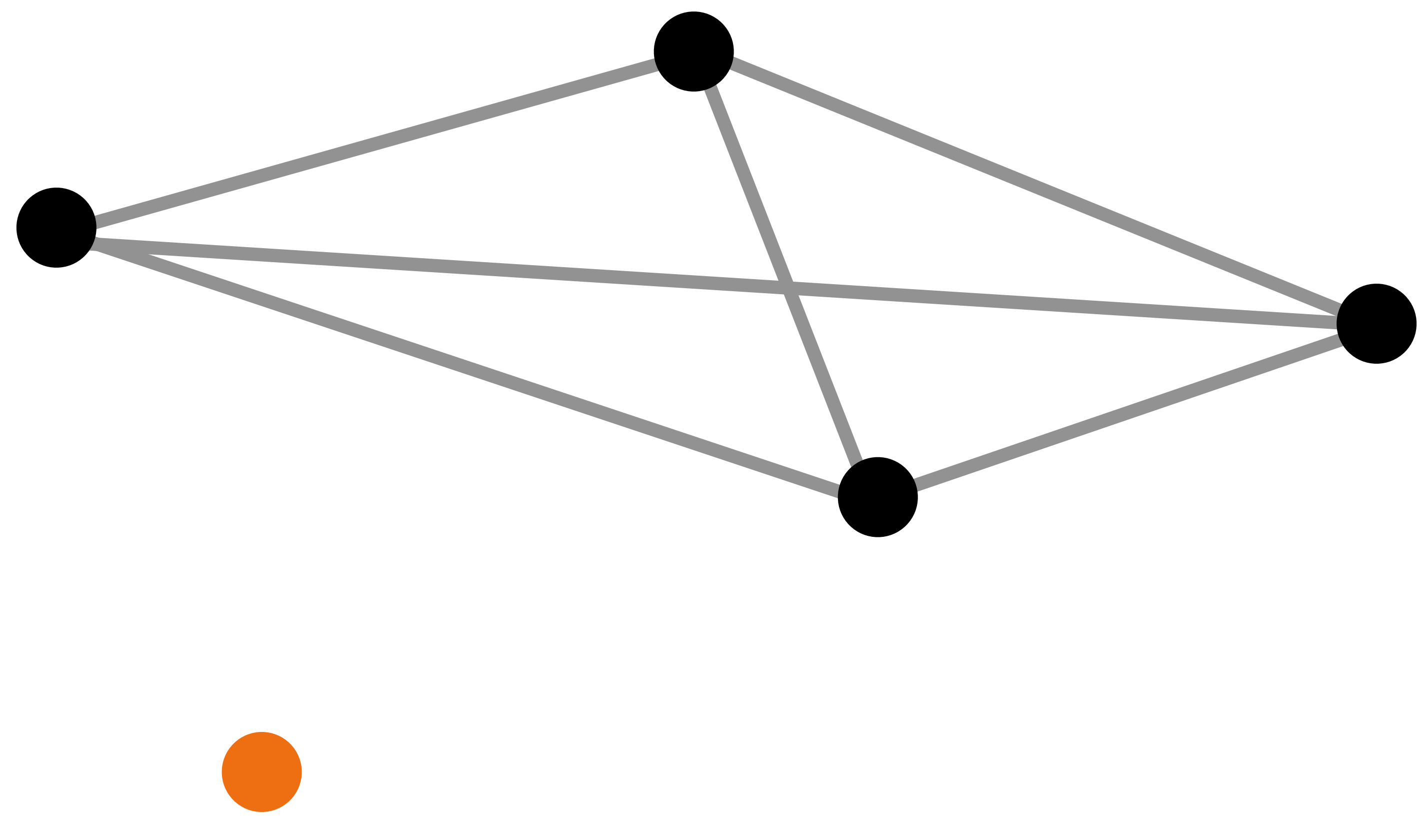
Jellyfish

Crossota norvegica

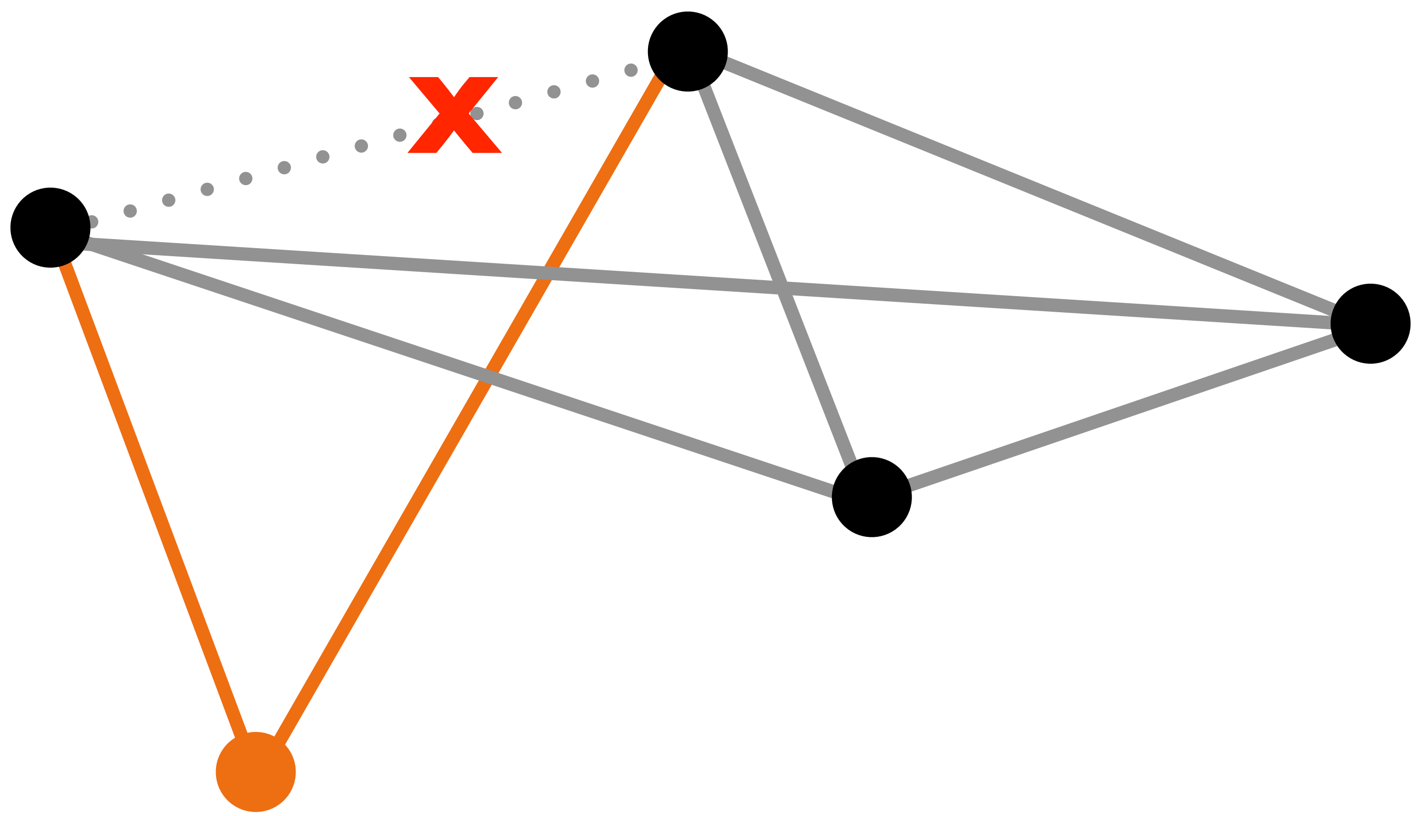
Photo: Kevin Raskoff

Construction & Expansion

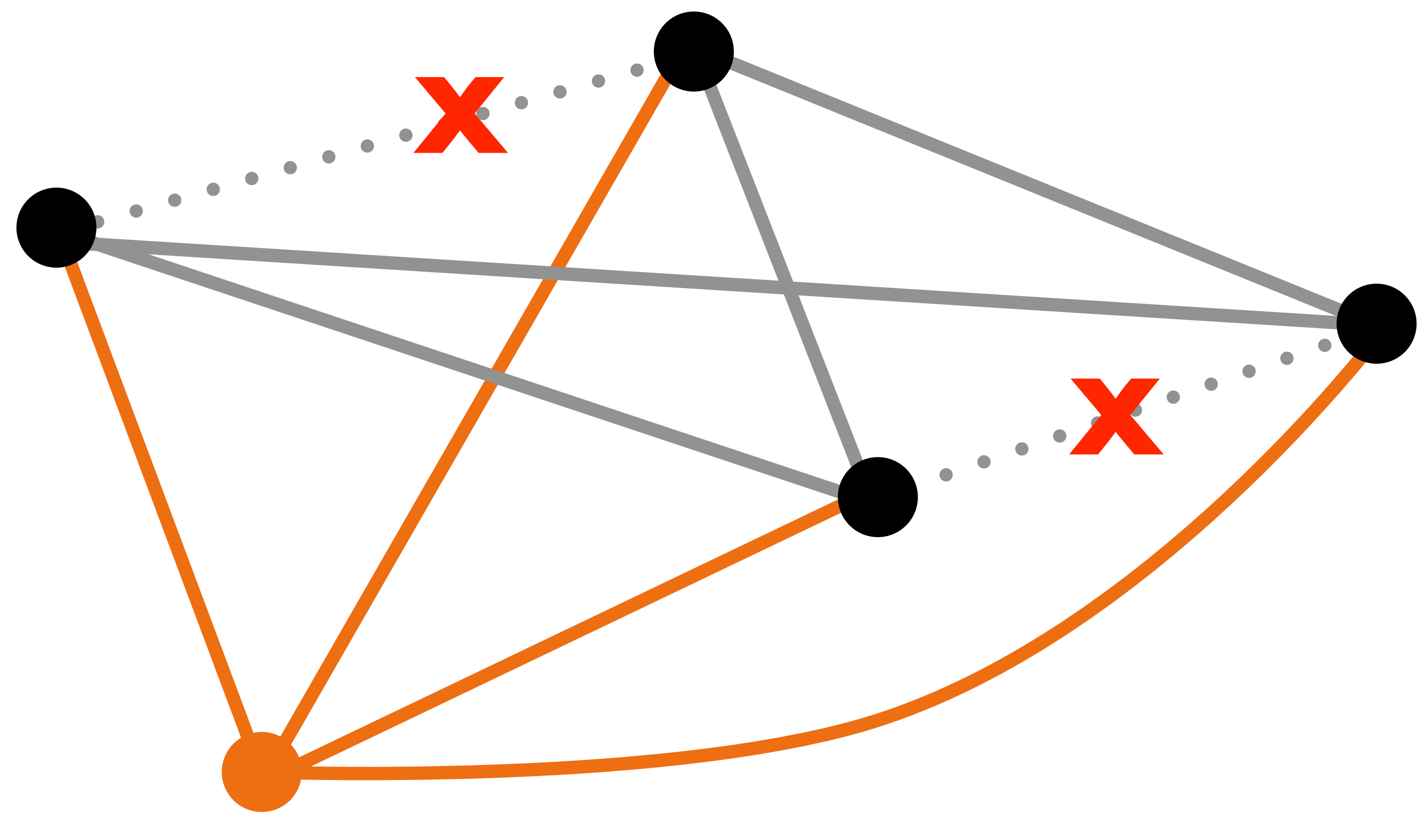
Building Jellyfish



Building Jellyfish



Building Jellyfish



Same procedure for initial construction
and incremental expansion

Can flexibly incorporate any type of equipment

Building Jellyfish

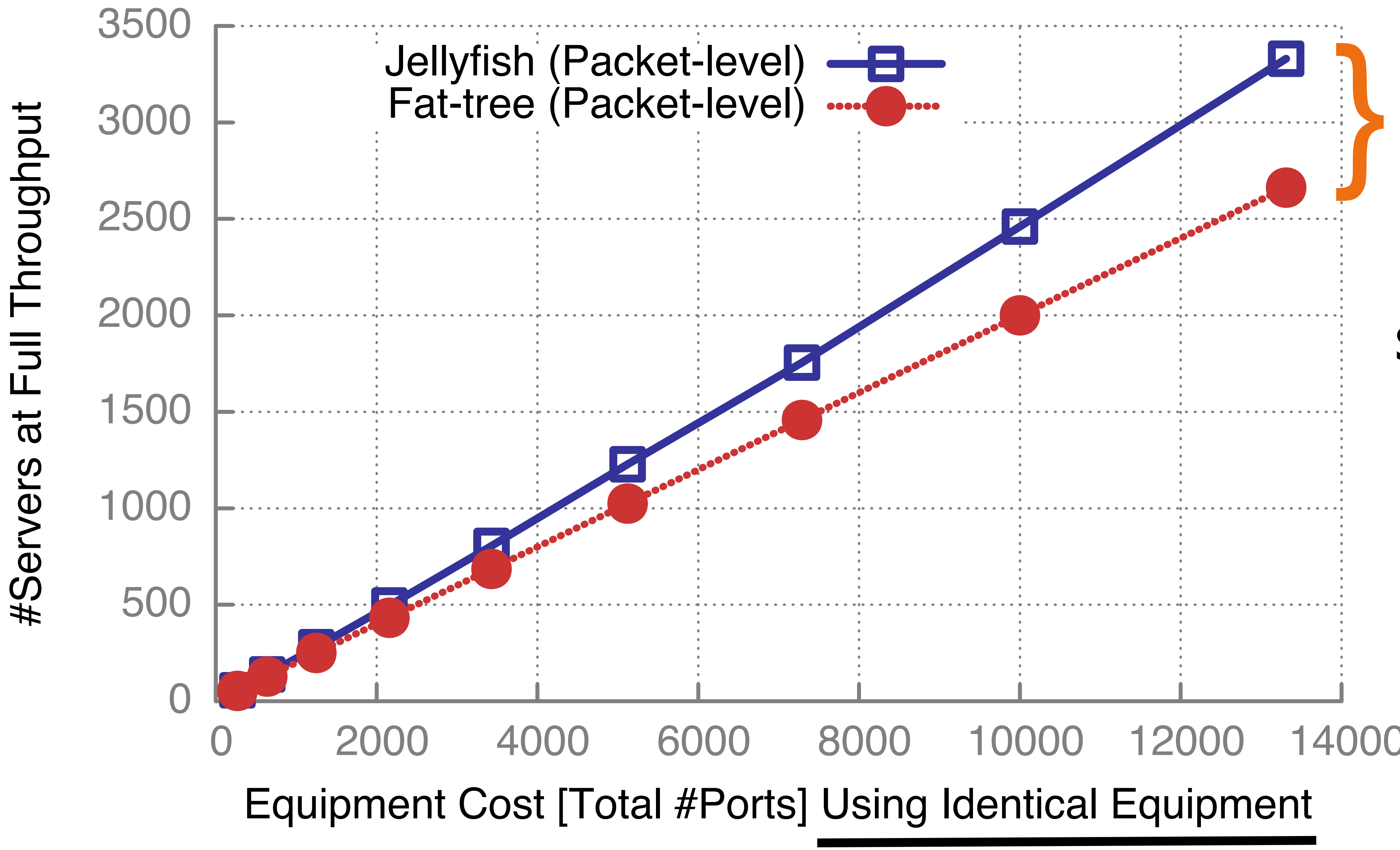
60% cheaper incremental expansion compared with past technique for traditional networks

LEGUP: [Curtis, Keshav, Lopez-Ortiz, CoNEXT'10]

Throughput

By giving up on structure,
do we take a hit on throughput?

Throughput: Jellyfish vs. fat tree

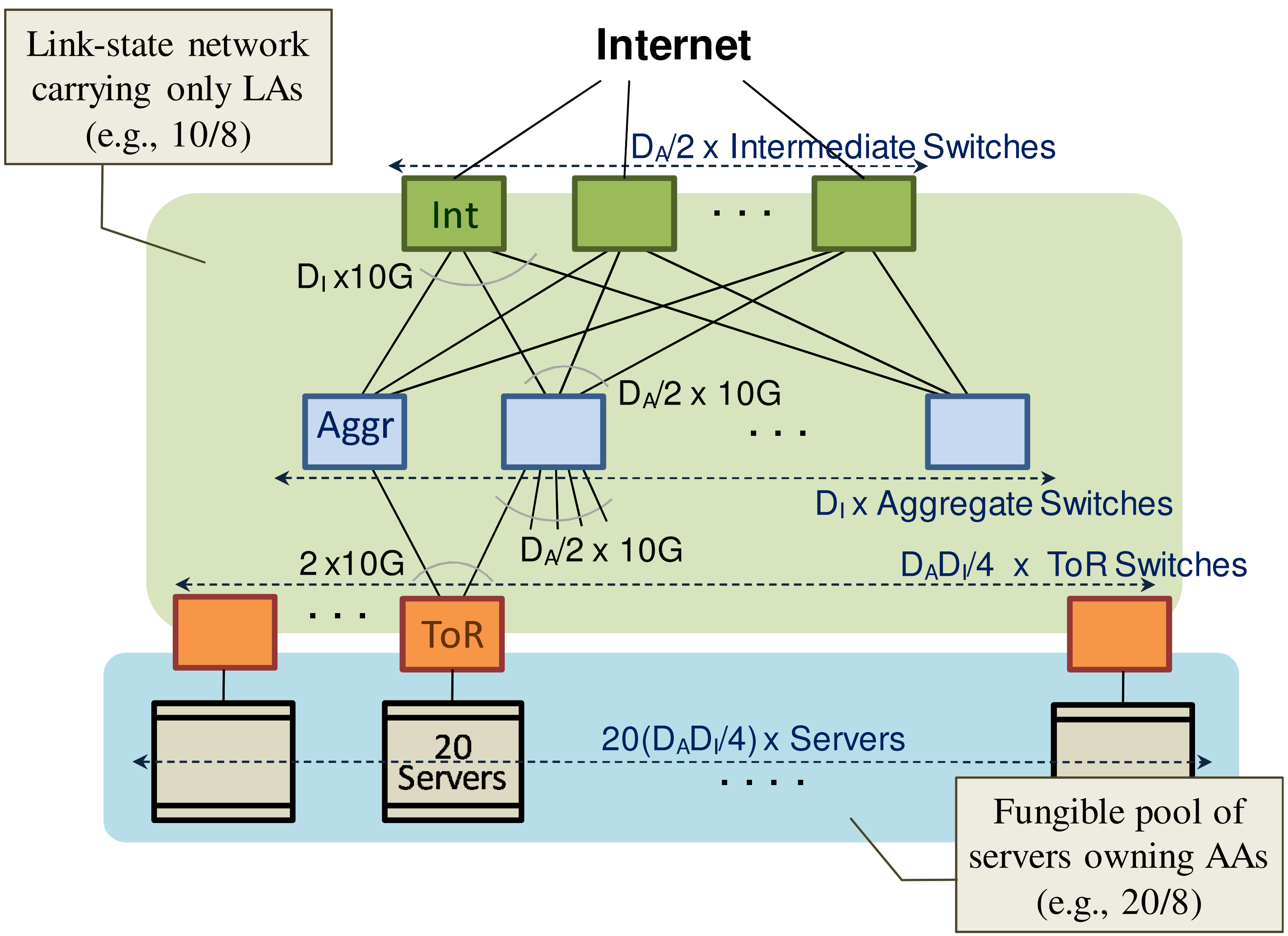


+25%

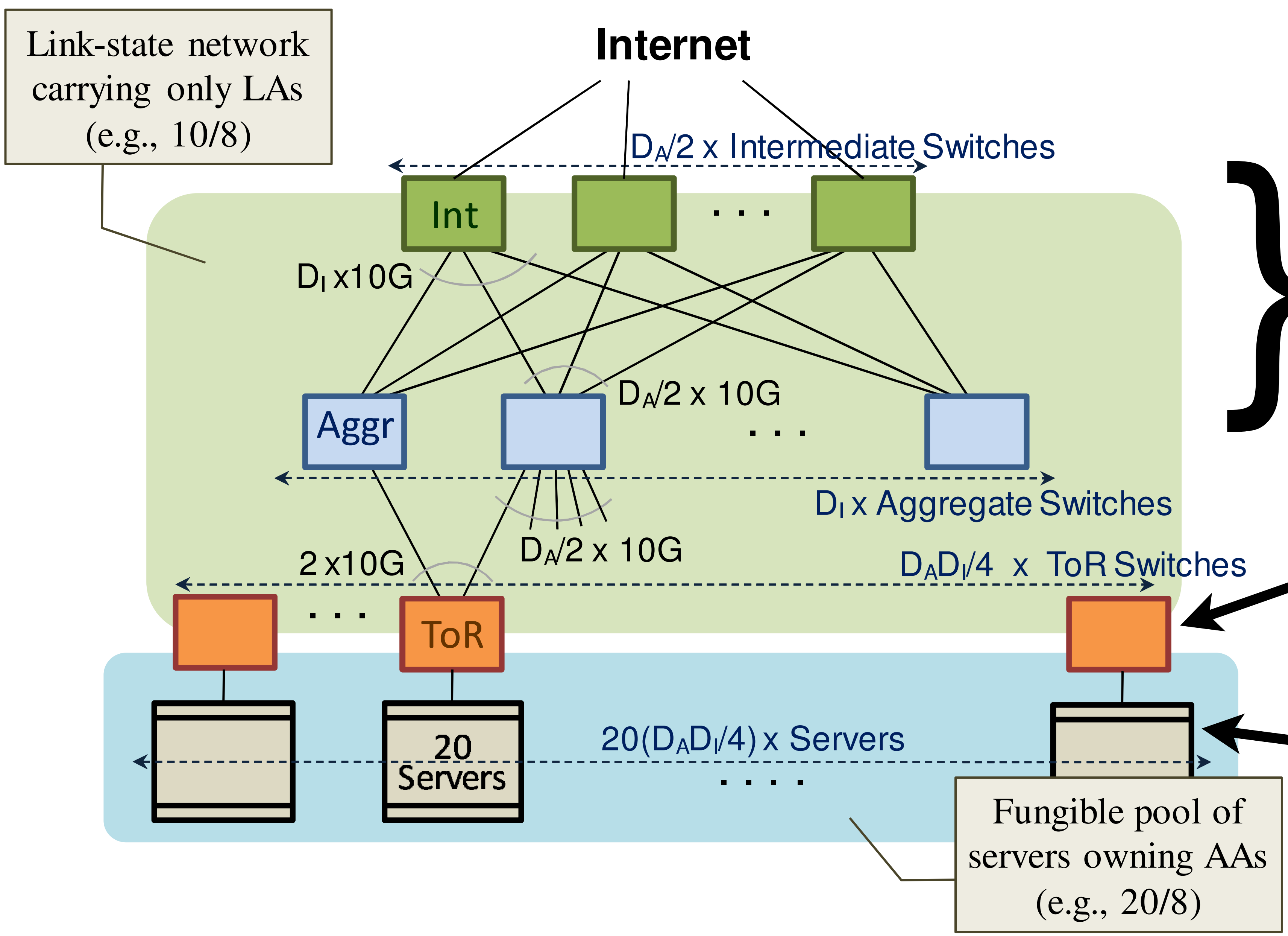
more servers

The VL2 topology

[Greenburg, Hamilton, Jain, Kandula, Kim, Lahiri, Maltz, Patel, Sengupta, SIGCOMM'09]



Rewiring VL2

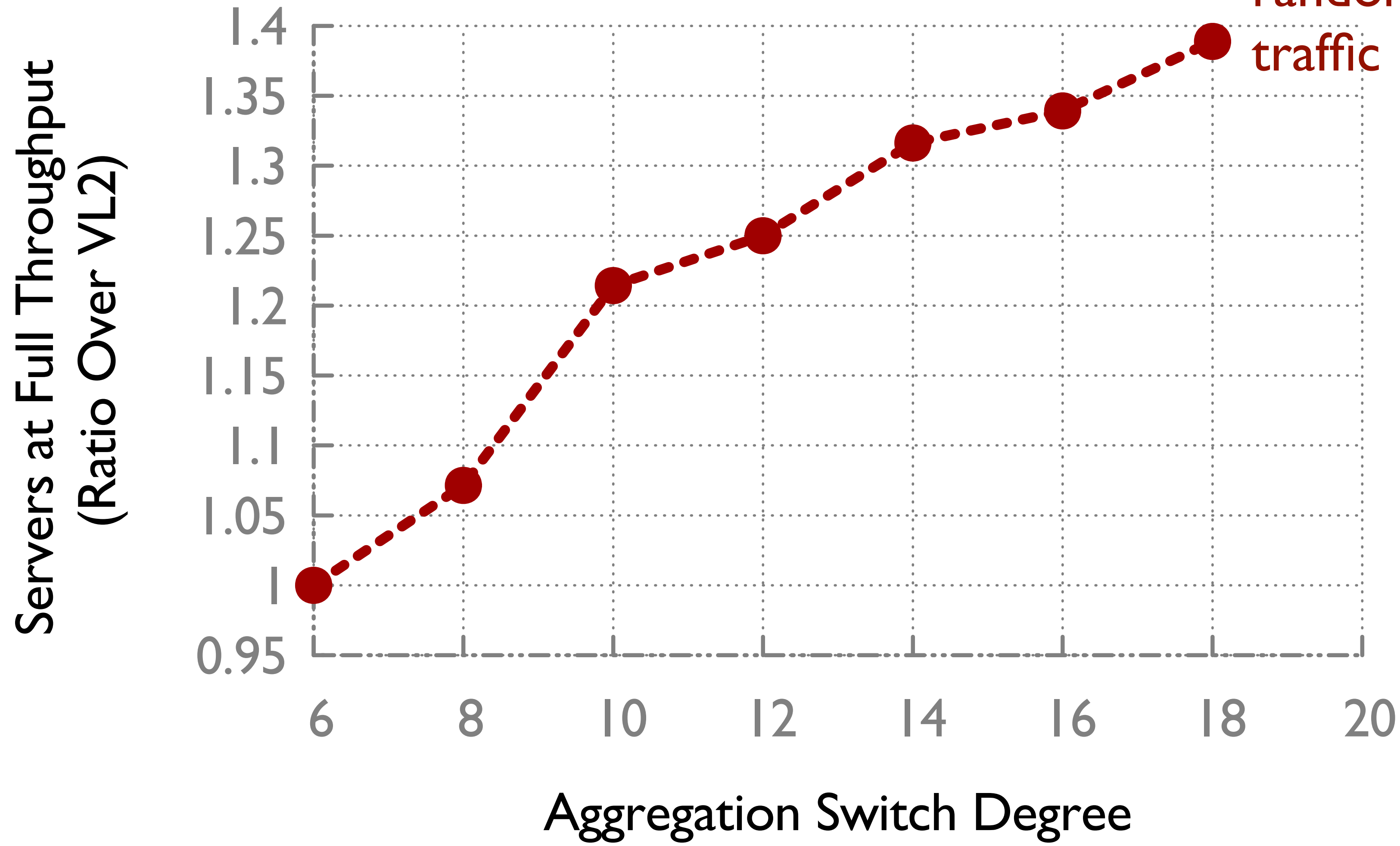


Uniform-random interconnection

Connect ToRs proportional to Intermediate/Agg degree

Servers unchanged (only ToRs have 1 Gbps ports)

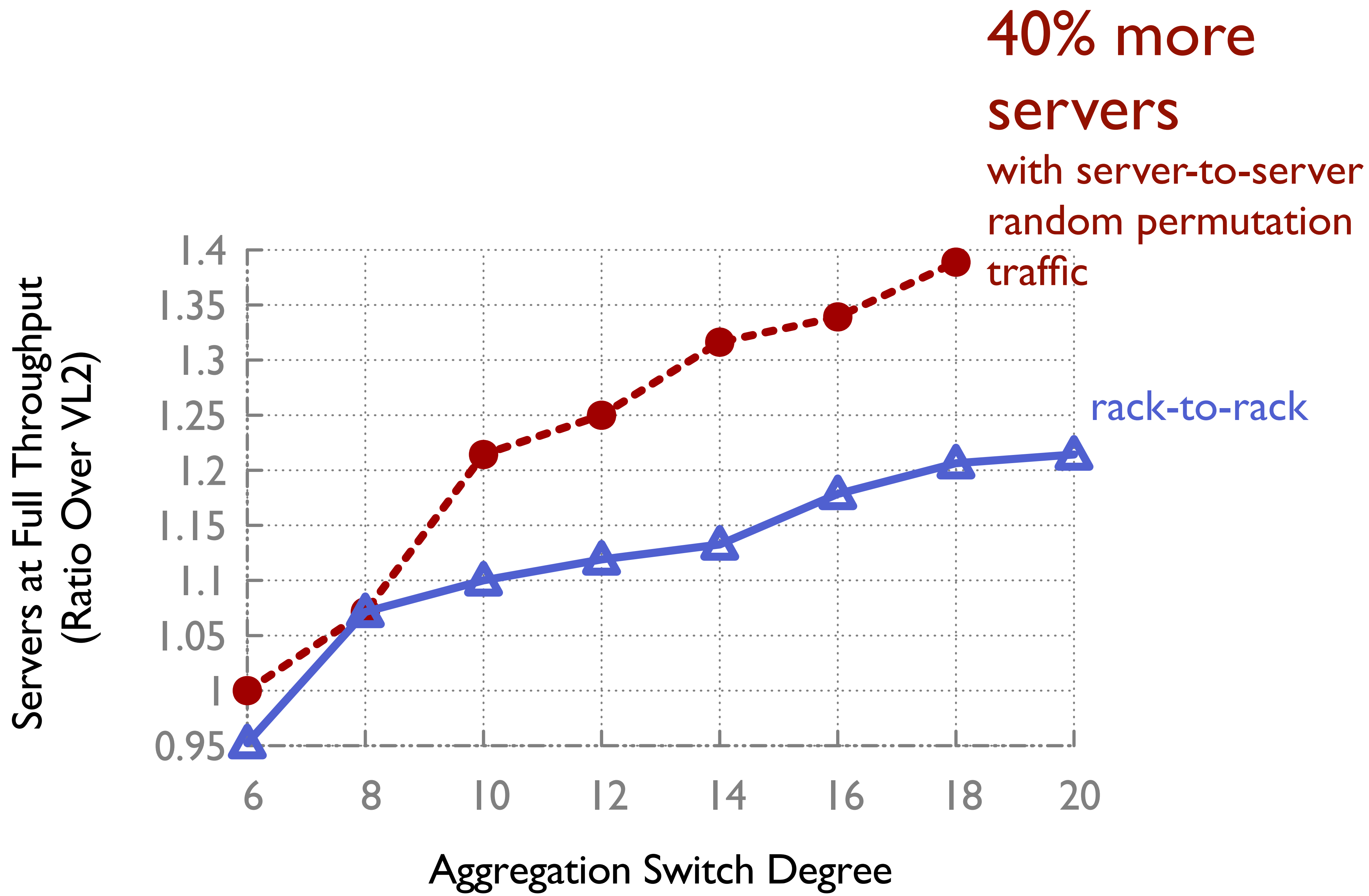
Rewiring VL2



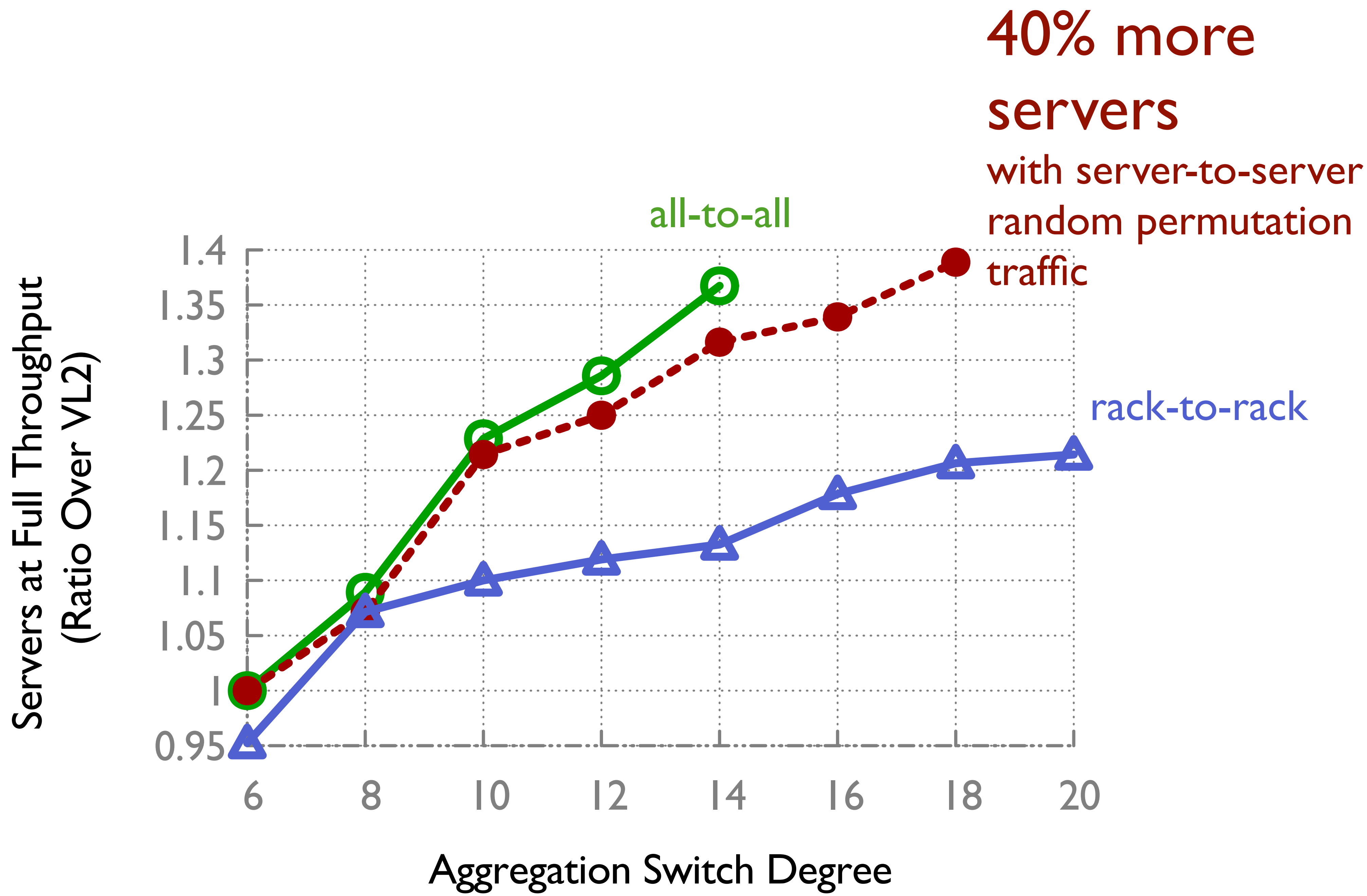
40% more servers

with server-to-server random permutation traffic

Rewiring VL2



Rewiring VL2



Just the beginning

Just the beginning

> *“Everything you just said is completely counterintuitive to everyone in this building.”*

– a large networking company

Just the beginning

Topology design

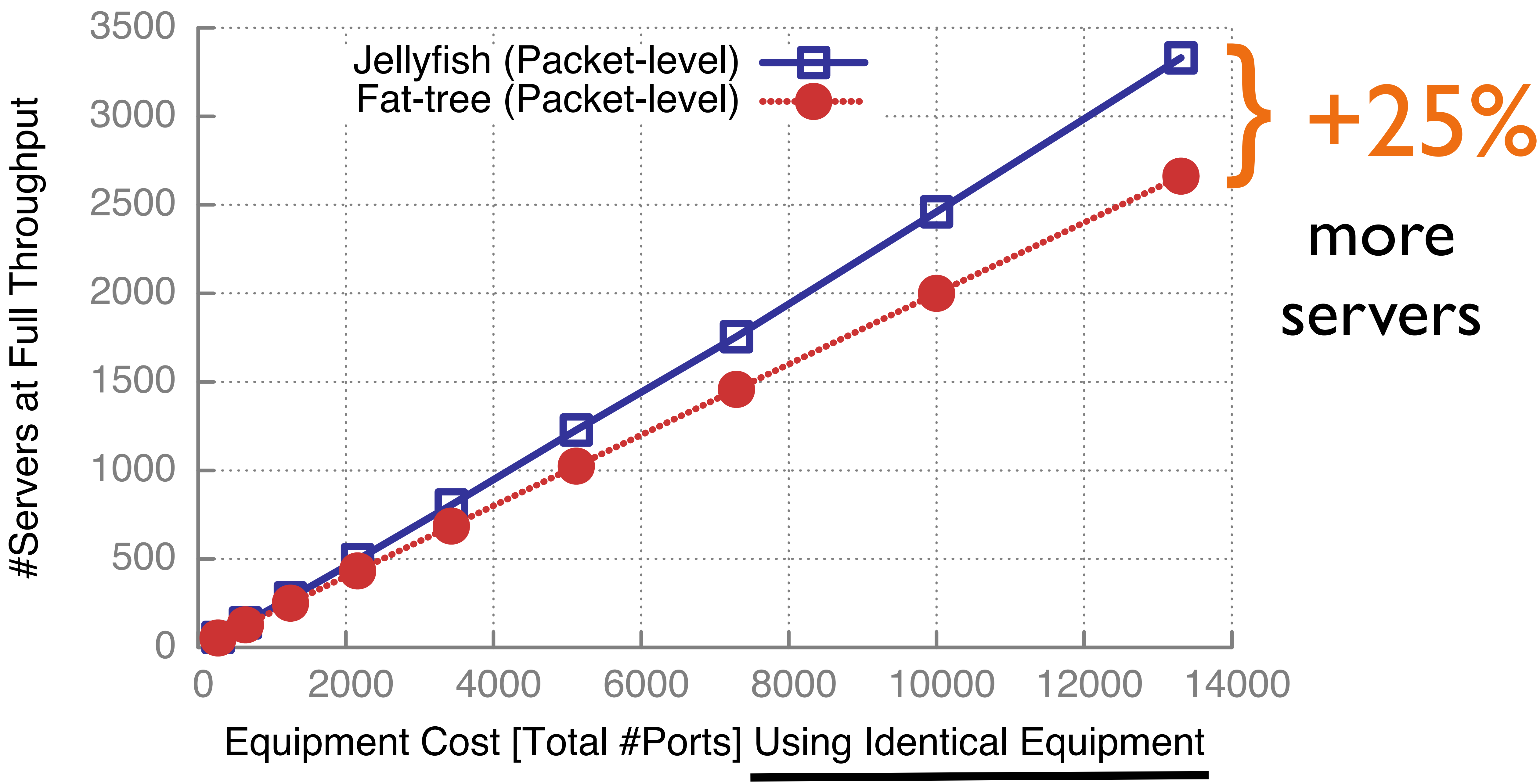
- How close are random graphs to optimal?
- What if switches are heterogeneous?

System design (or: *"But what about..."*)

- Performance consistency?
- Cabling spaghetti?
- Routing and congestion control without structure?

Understanding Throughput

Throughput: Jellyfish vs. fat tree



Intuition

if we fully utilize all available capacity ...

$$\#\ 1\ \text{Gbps flows} = \frac{\text{total capacity}}{\text{used capacity per flow}}$$

Intuition

if we fully utilize all available capacity ...

$$\# \text{ 1 Gbps flows} = \frac{\sum_{\text{links}} \text{capacity}(\textit{link})}{\text{used capacity per flow}}$$

Intuition

if we fully utilize all available capacity ...

$$\#\ 1\text{ Gbps flows} = \frac{\sum_{\text{links}} \text{capacity}(\textit{link})}{1\text{ Gbps} \cdot \text{mean path length}}$$

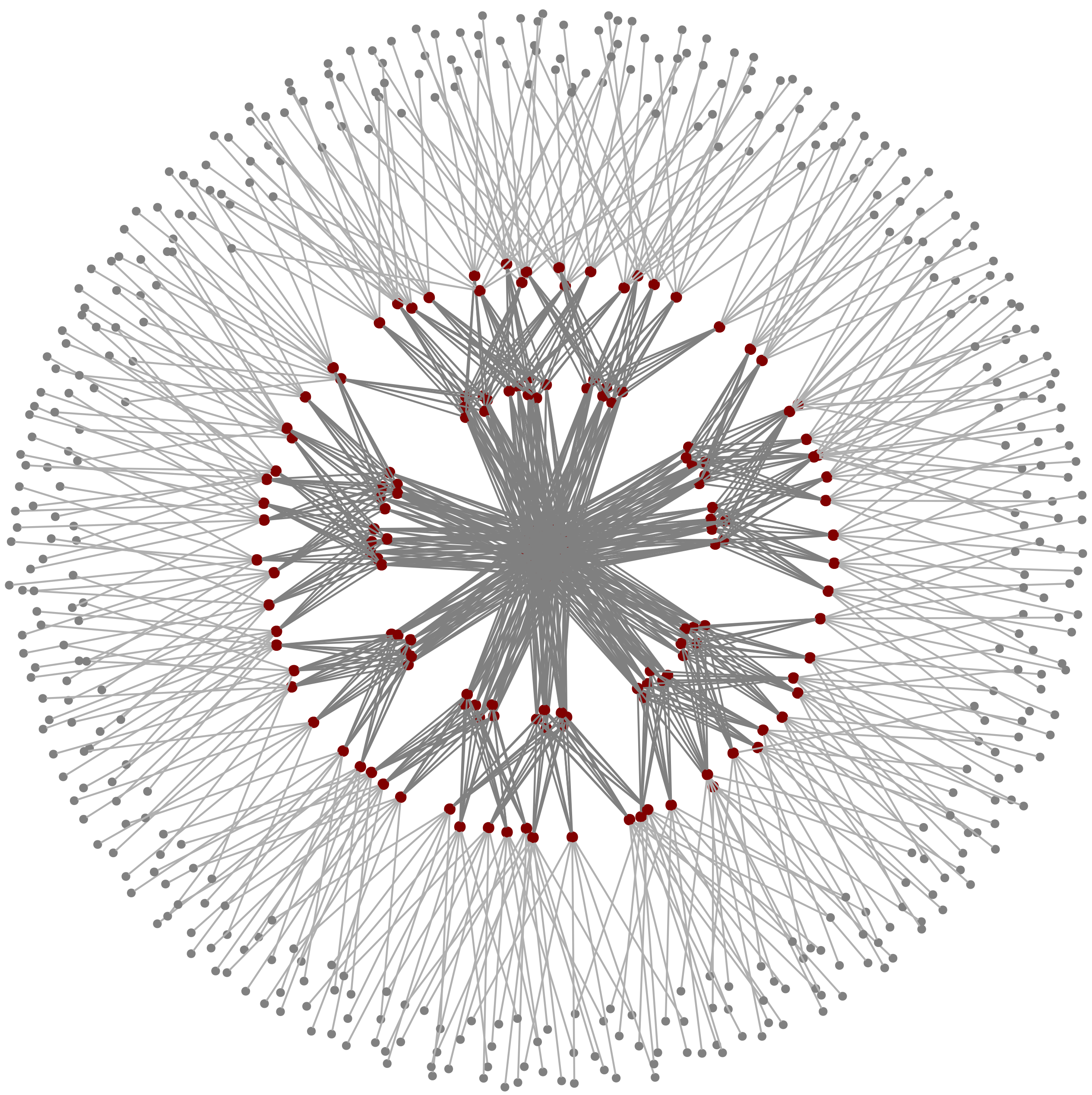
Intuition

if we fully utilize all available capacity ...

$$\#\ 1\ \text{Gbps flows} = \frac{\sum_{\text{links}} \text{capacity}(\textit{link})}{1\ \text{Gbps} \cdot \text{mean path length}}$$

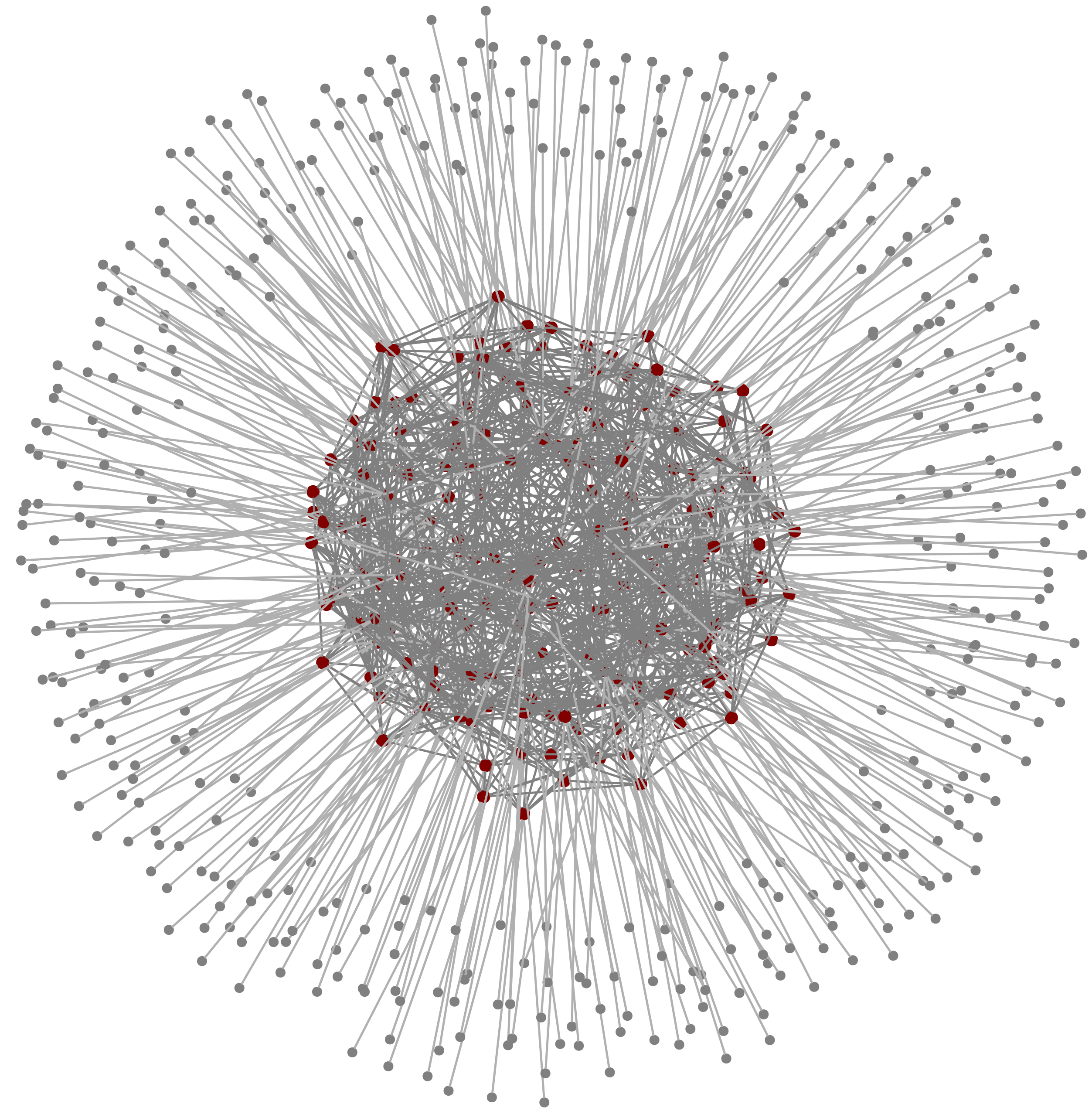
Mission:
minimize average path length

Example



Fat tree

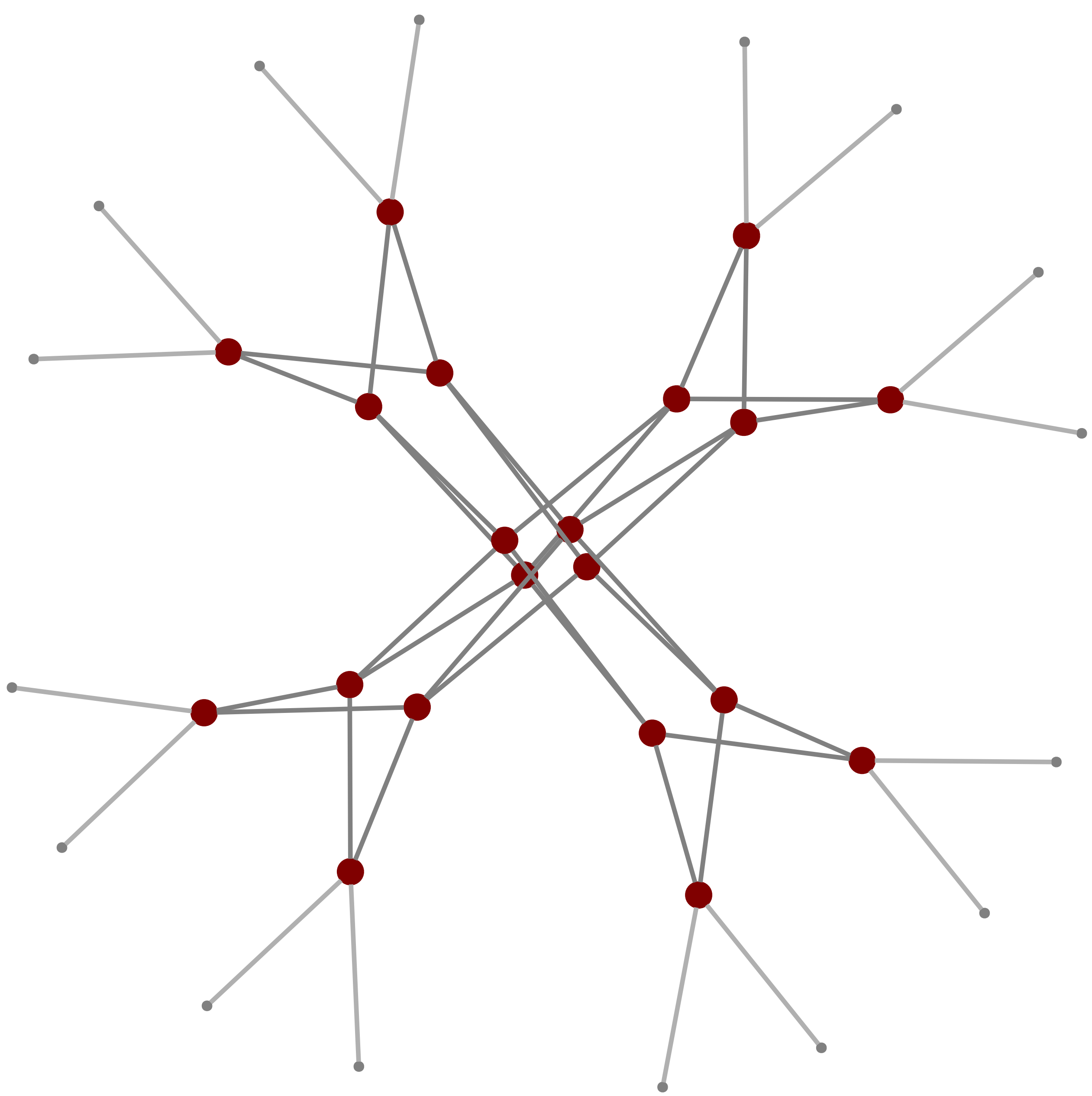
432 servers, 180 switches, degree 12



Jellyfish random graph

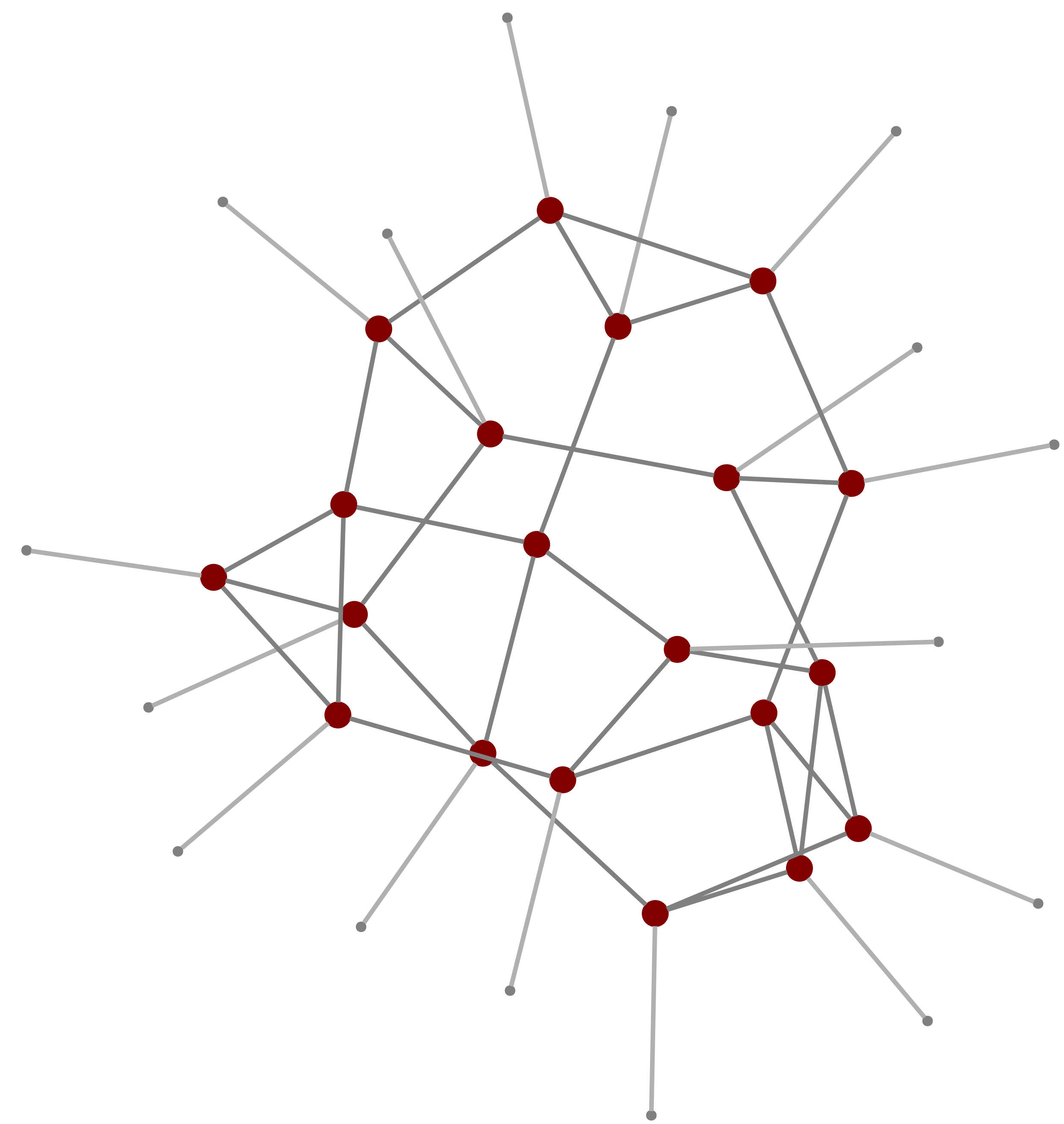
432 servers, 180 switches, degree 12

Example



Fat tree

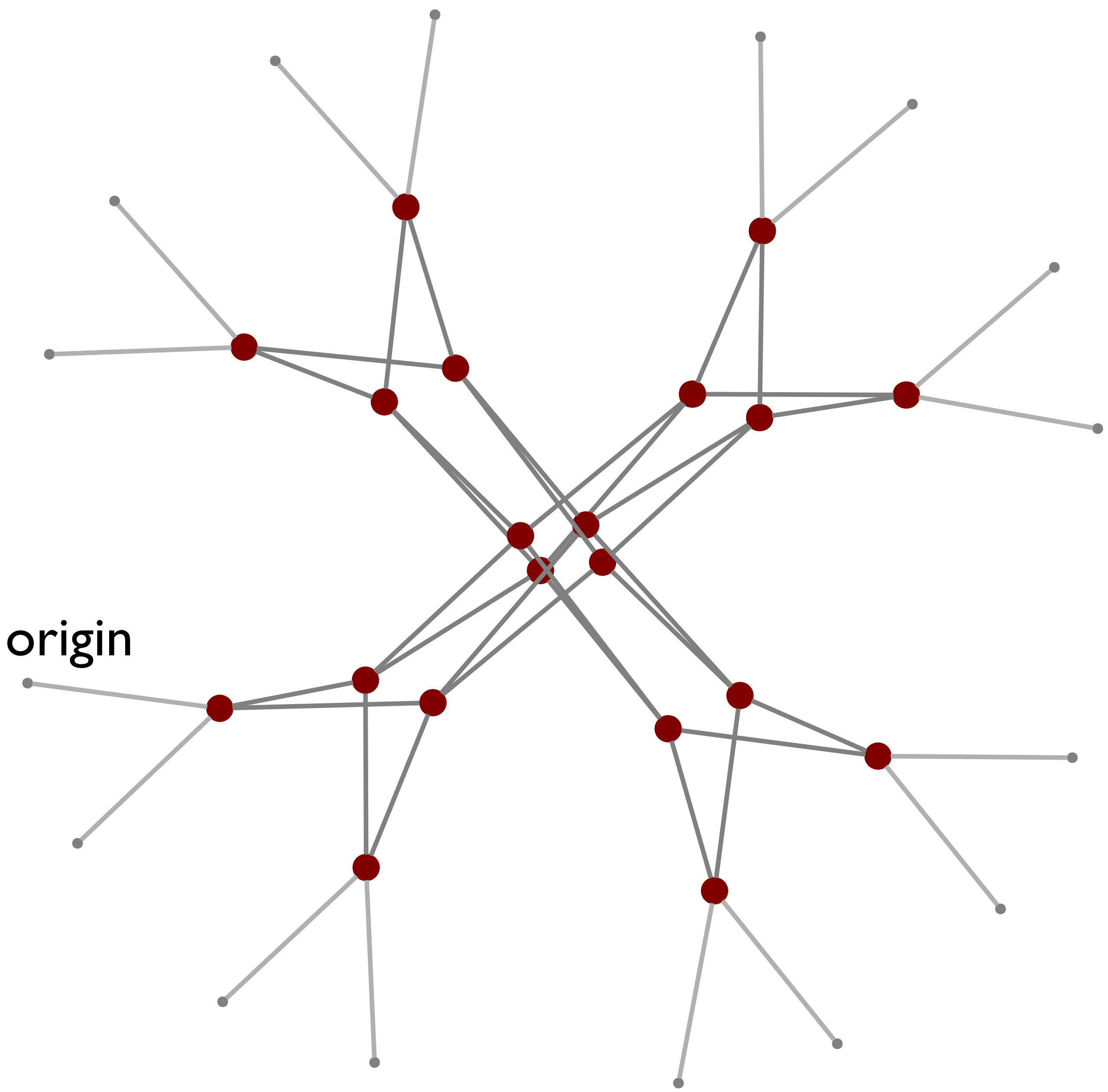
16 servers, 20 switches, degree 4



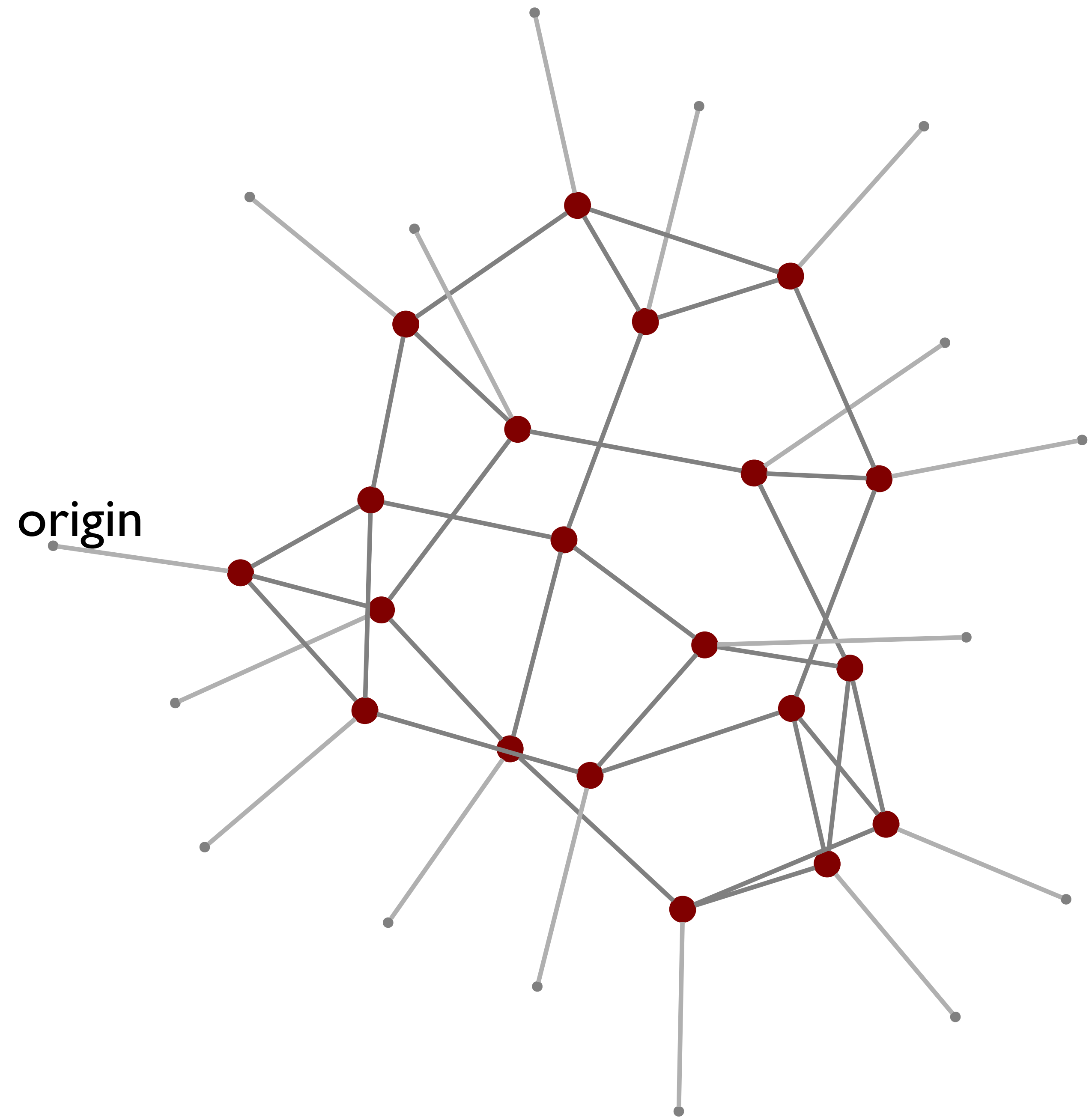
Jellyfish random graph

16 servers, 20 switches, degree 4

Example

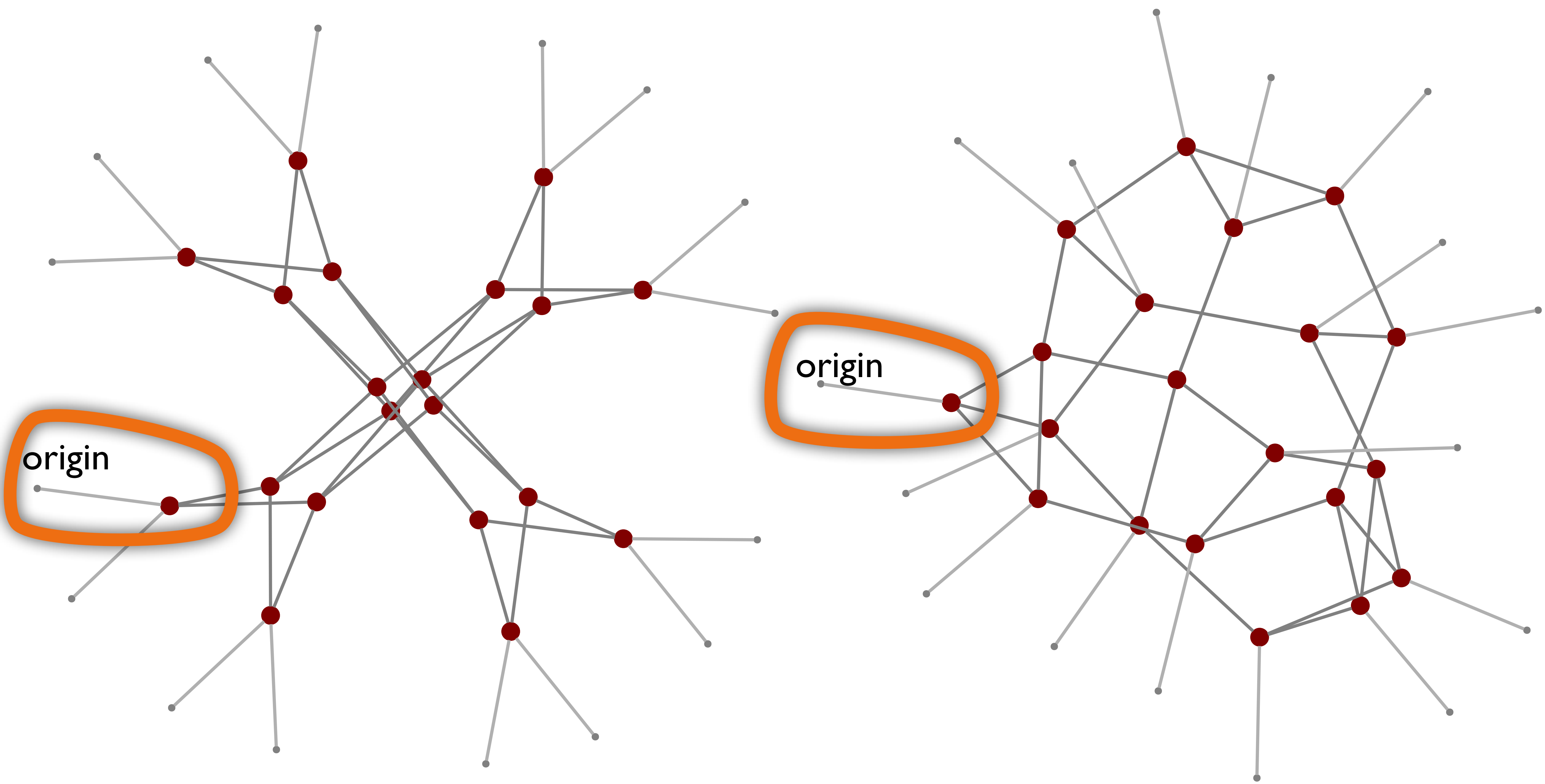


Fat tree
16 servers, 20 switches, degree 4



Jellyfish random graph
16 servers, 20 switches, degree 4

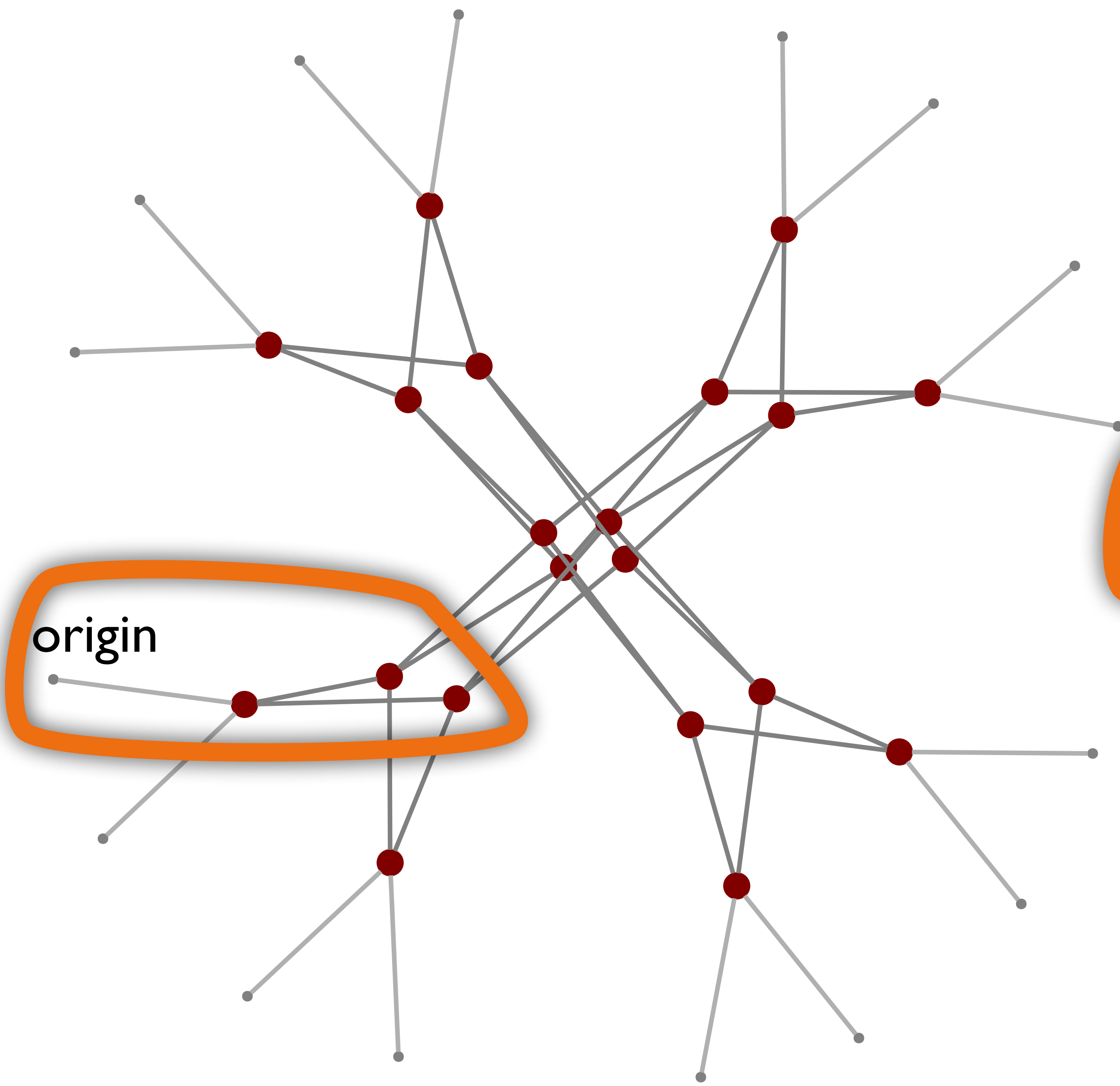
Example



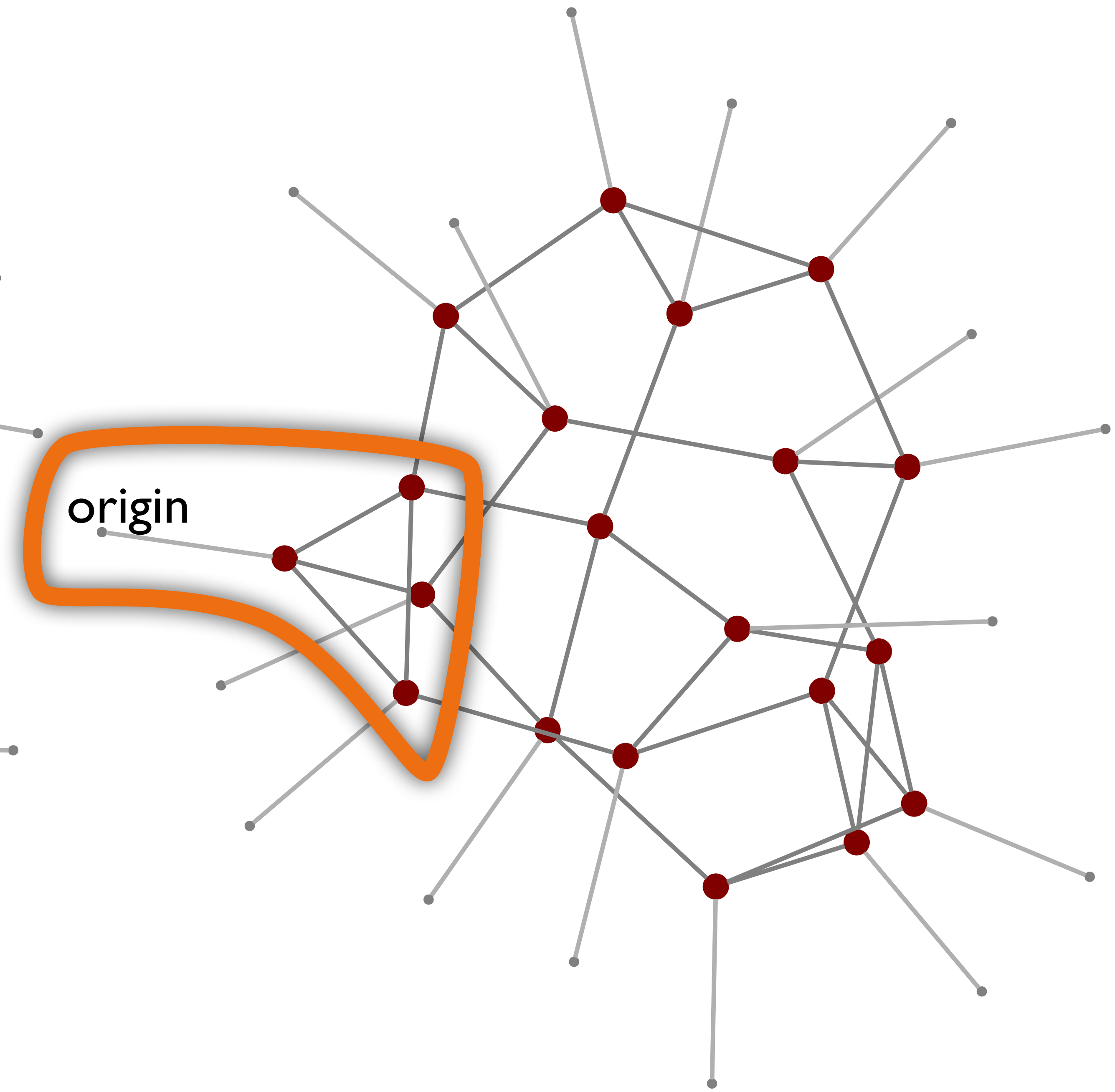
Fat tree
16 servers, 20 switches, degree 4

Jellyfish random graph
16 servers, 20 switches, degree 4

Example

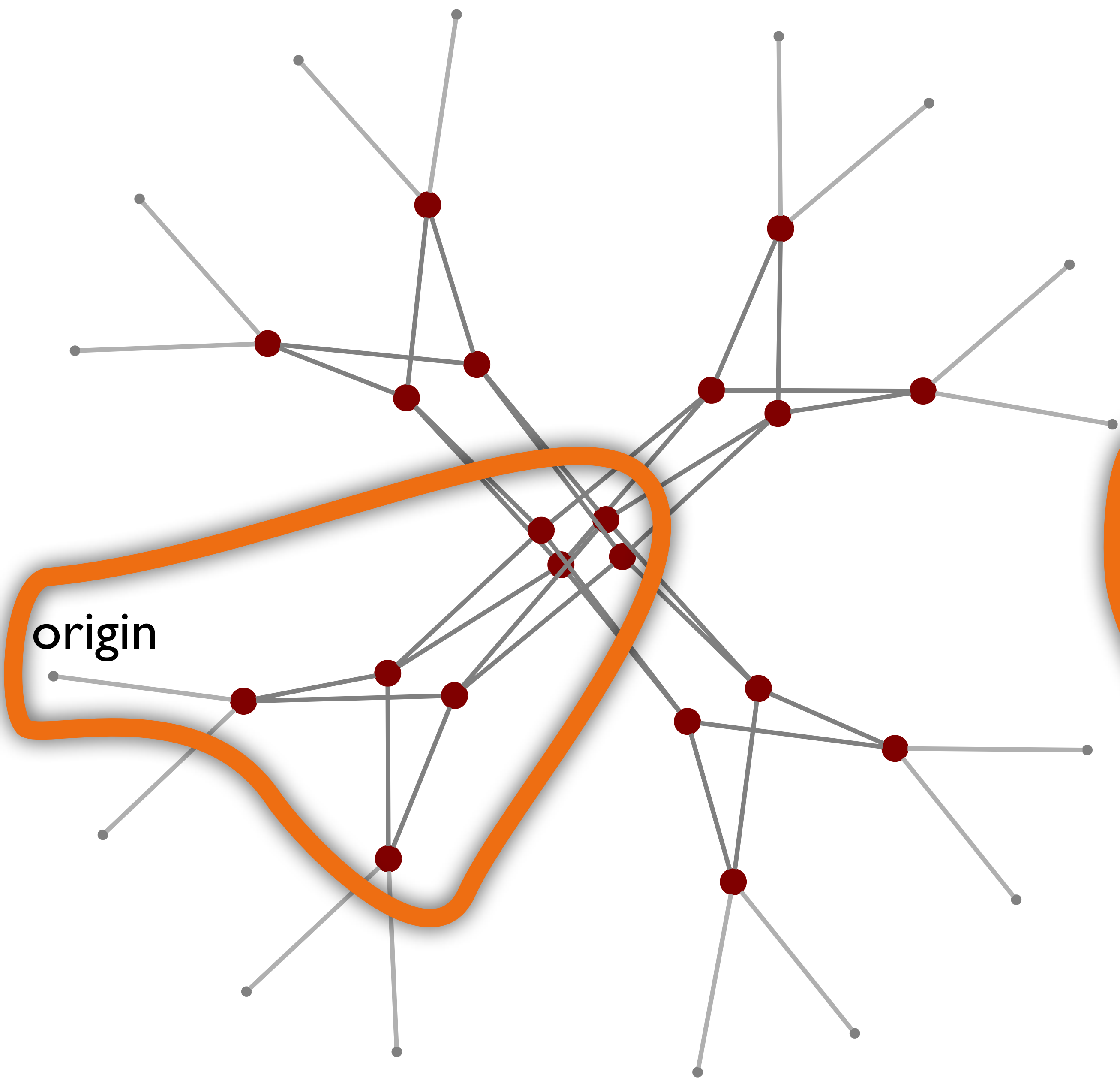


Fat tree
16 servers, 20 switches, degree 4

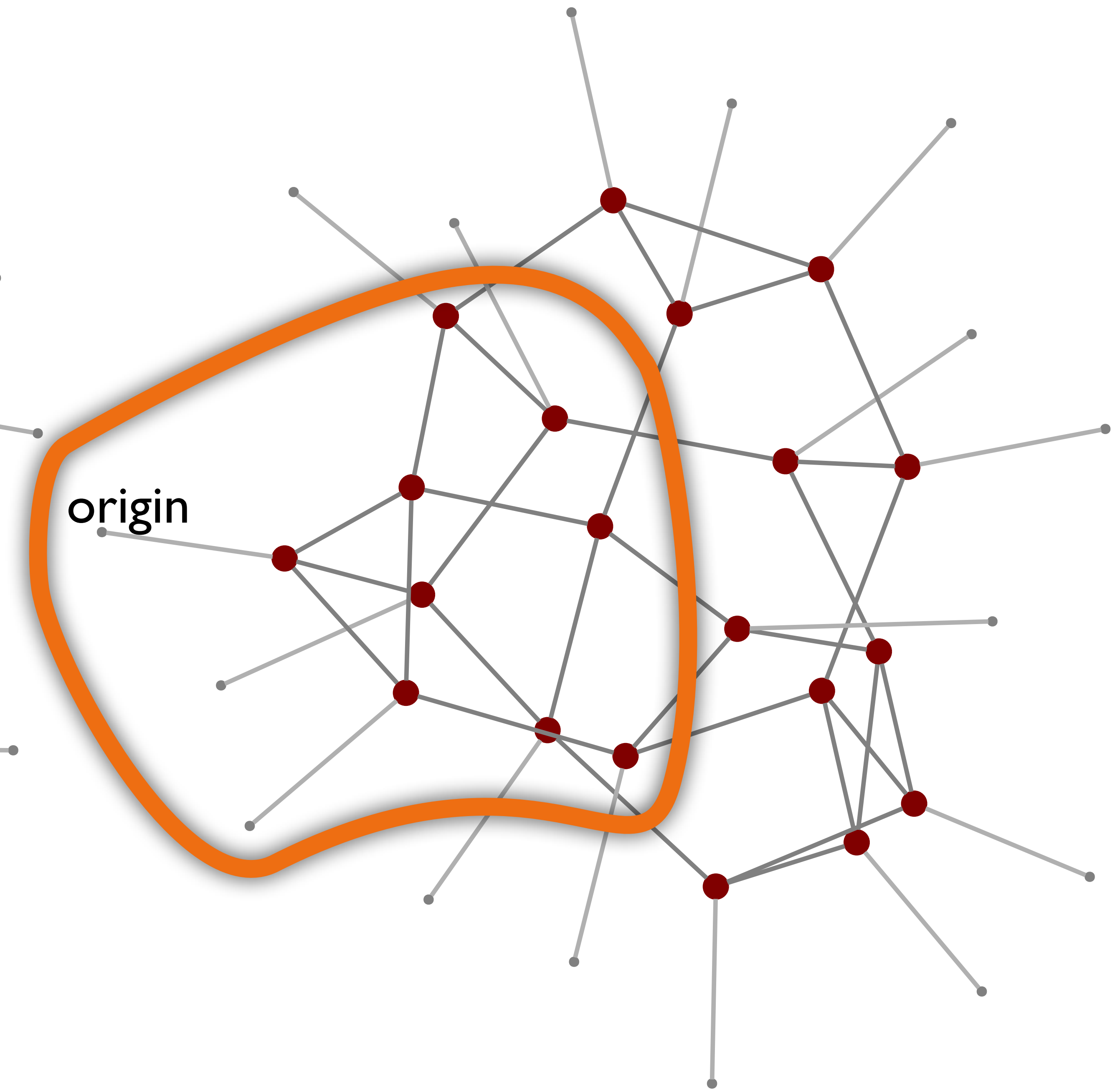


Jellyfish random graph
16 servers, 20 switches, degree 4

Example

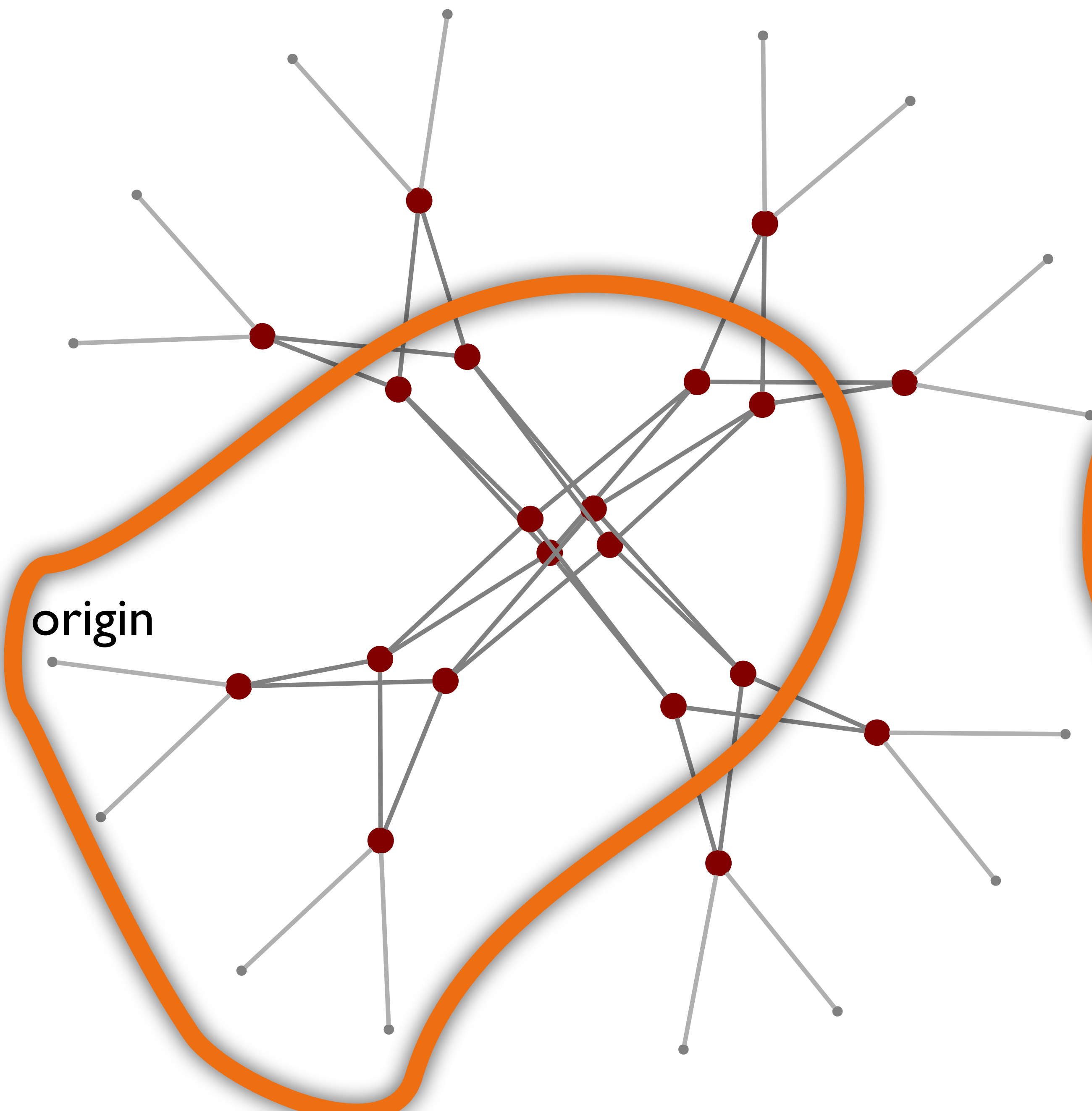


Fat tree
16 servers, 20 switches, degree 4

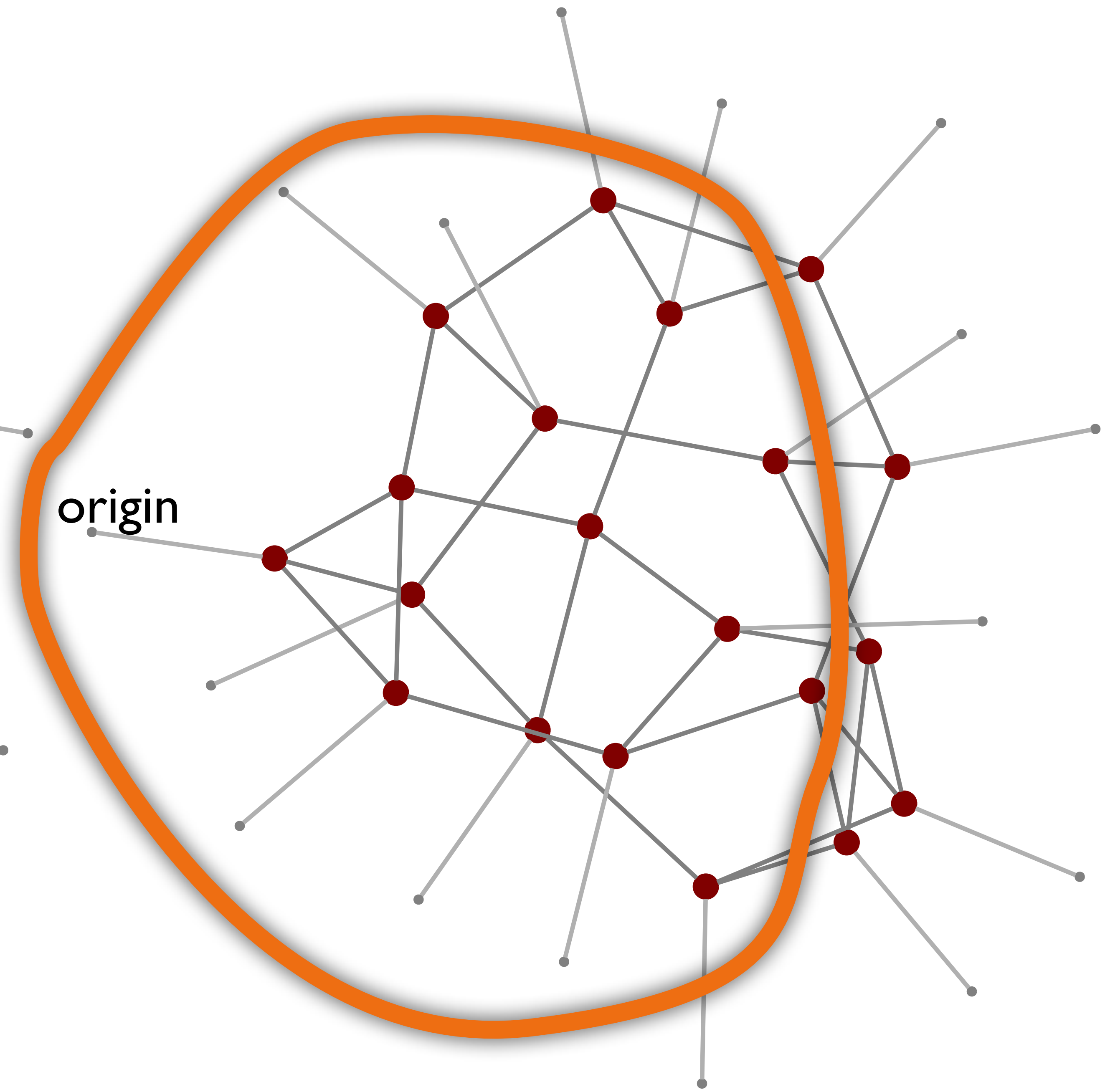


Jellyfish random graph
16 servers, 20 switches, degree 4

Example

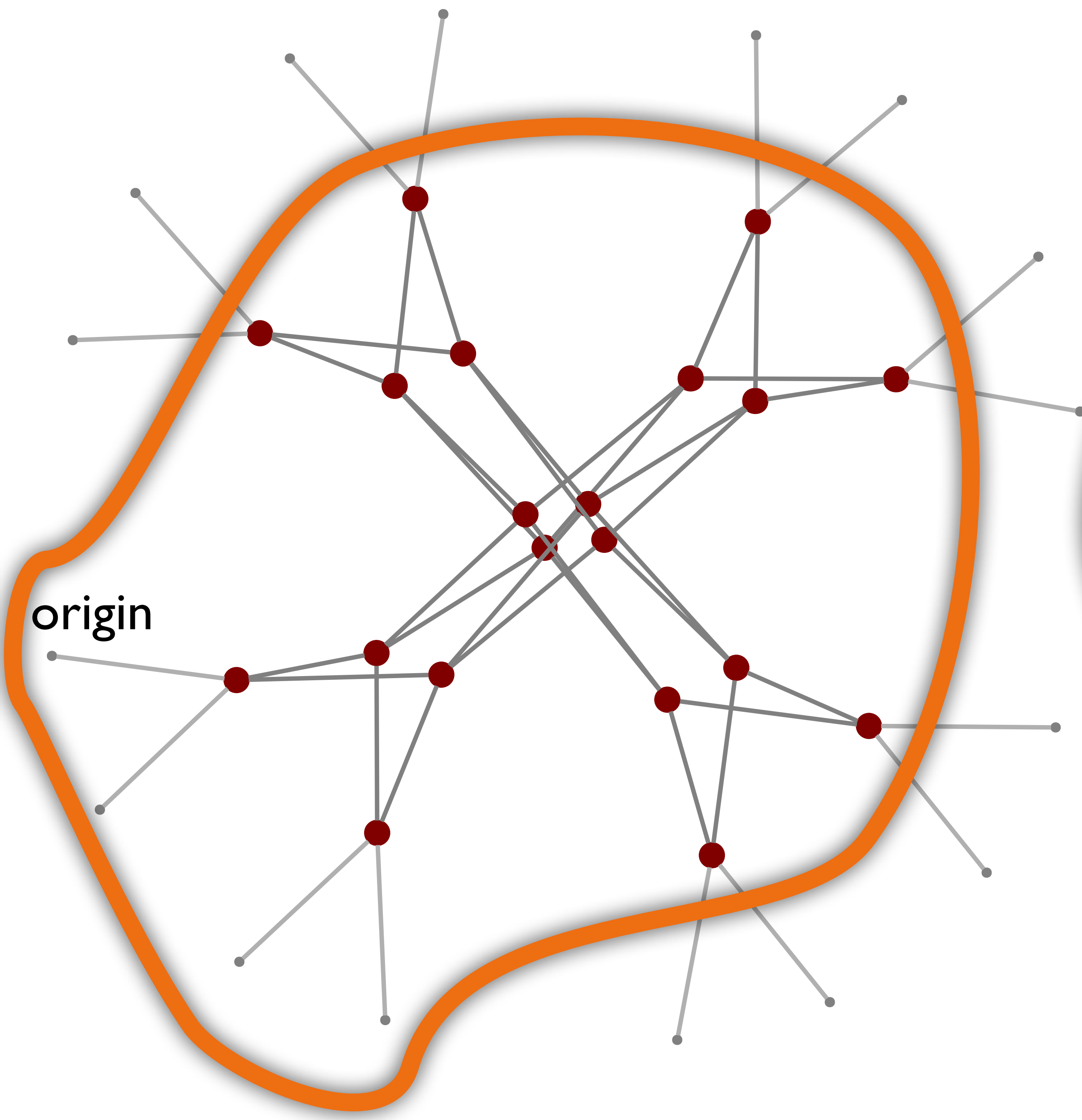


Fat tree
16 servers, 20 switches, degree 4



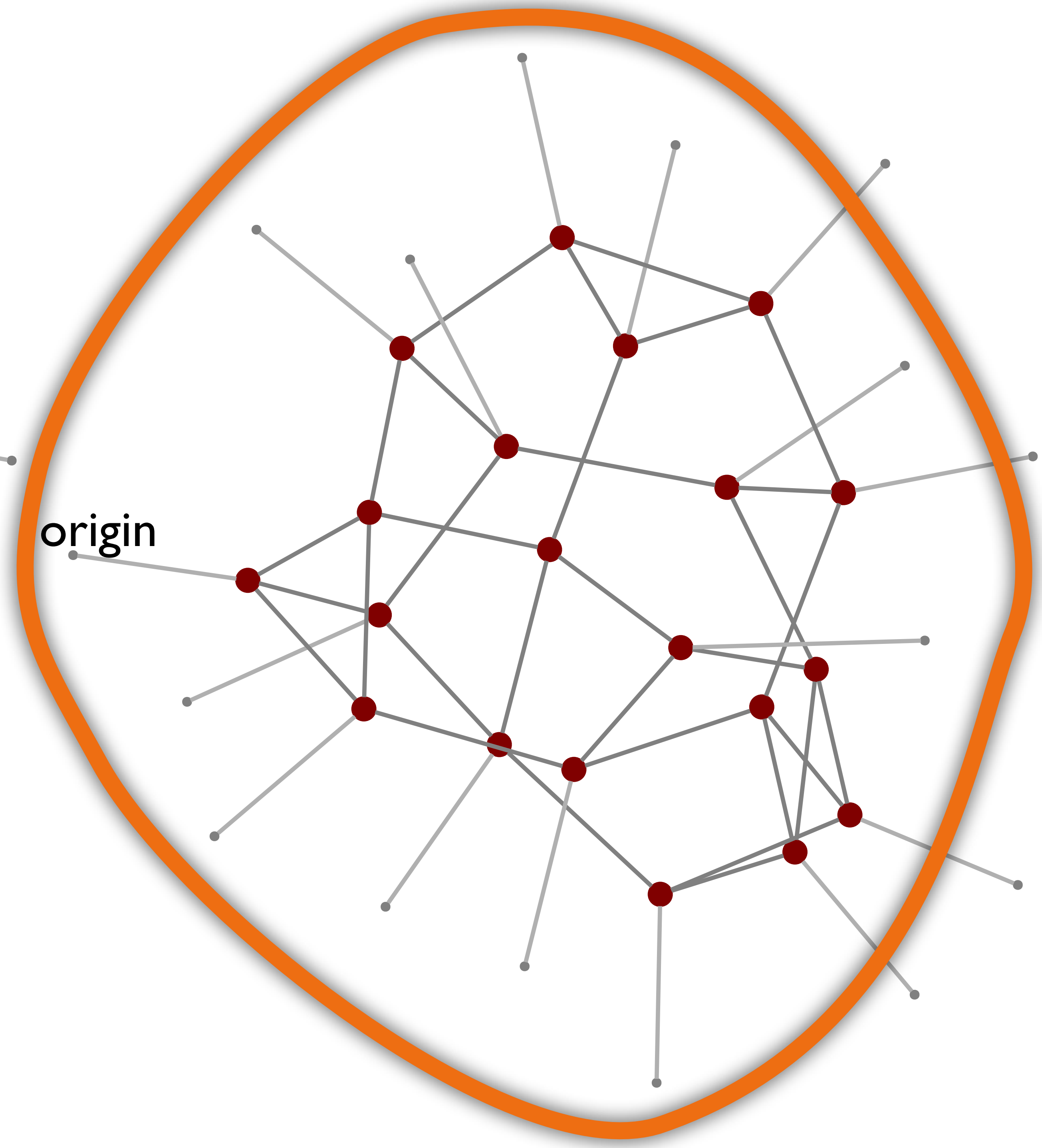
Jellyfish random graph
16 servers, 20 switches, degree 4

Example



Fat tree

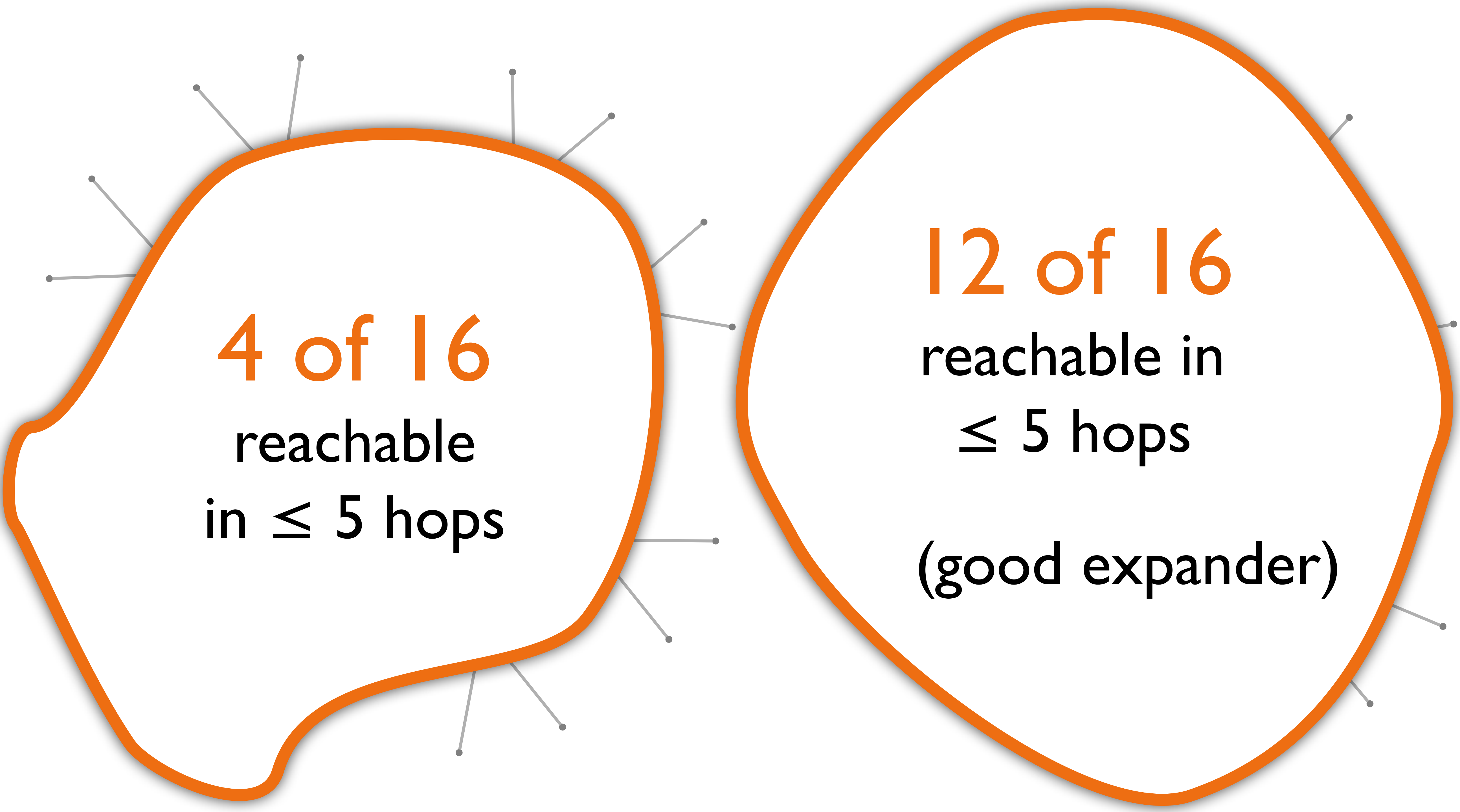
16 servers, 20 switches, degree 4



Jellyfish random graph

16 servers, 20 switches, degree 4

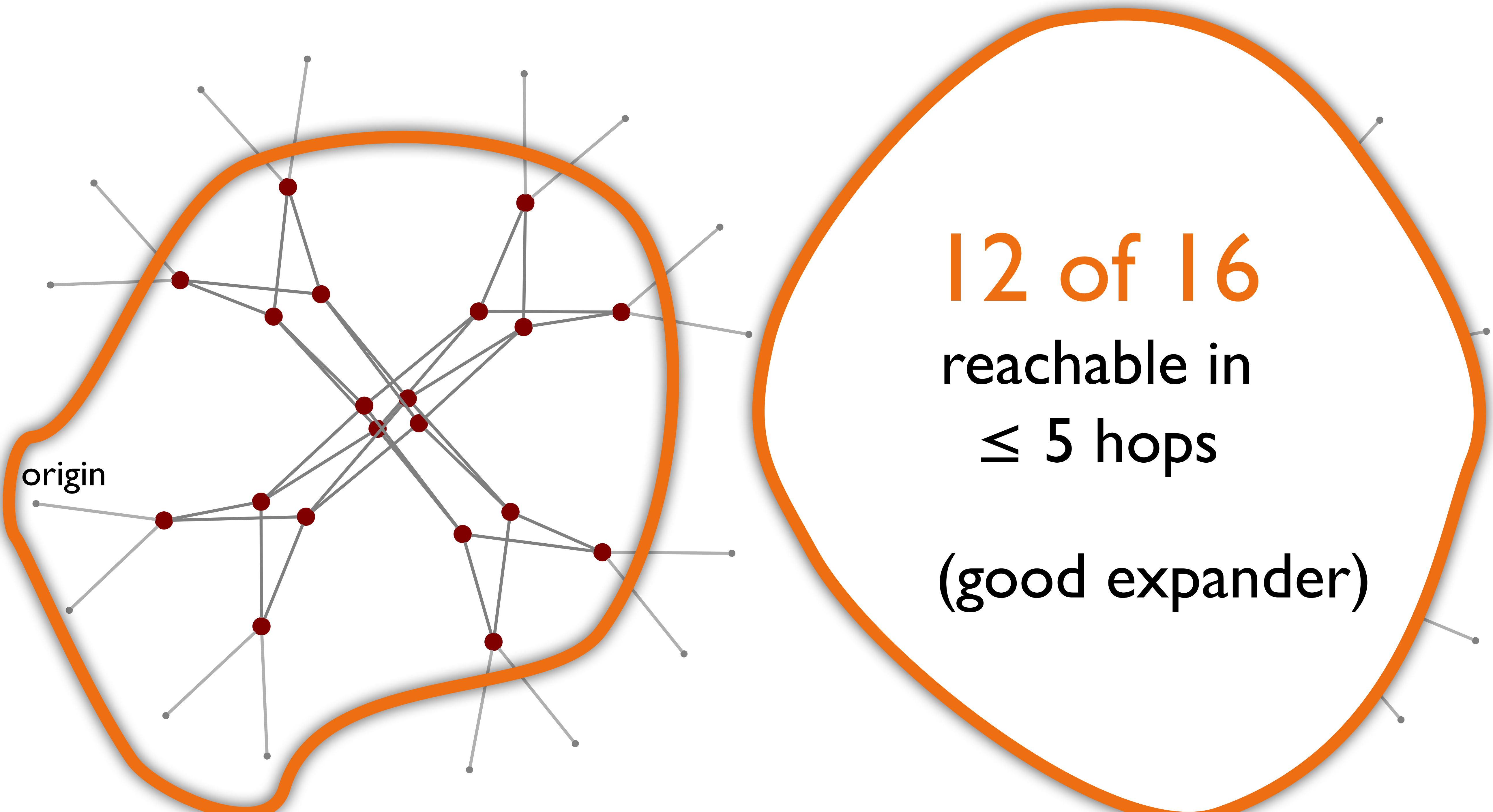
Example



Fat tree
16 servers, 20 switches, degree 4

Jellyfish random graph
16 servers, 20 switches, degree 4

Example



Fat tree

16 servers, 20 switches, degree 4

12 of 16

reachable in

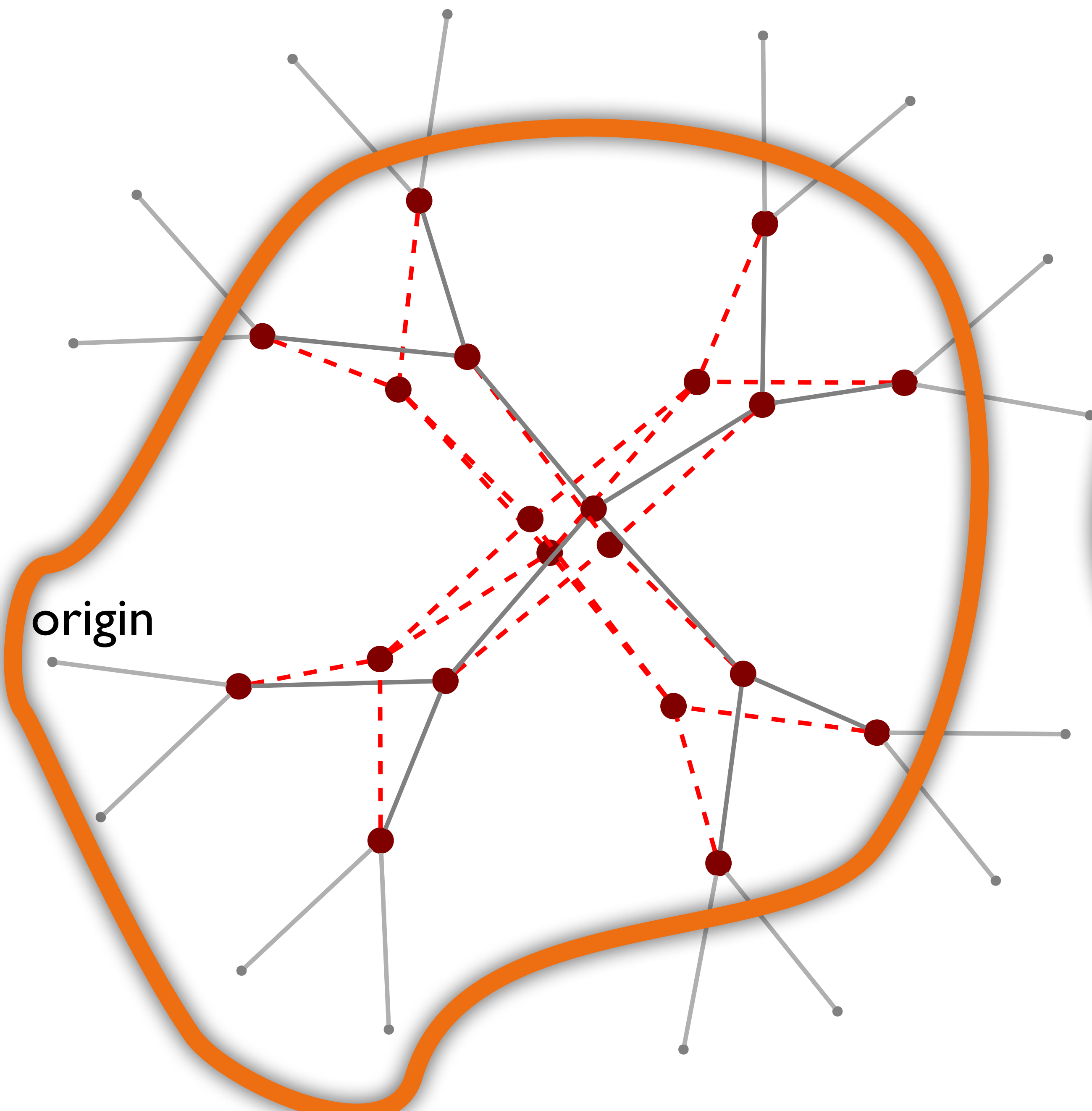
≤ 5 hops

(good expander)

Jellyfish random graph

16 servers, 20 switches, degree 4

Example



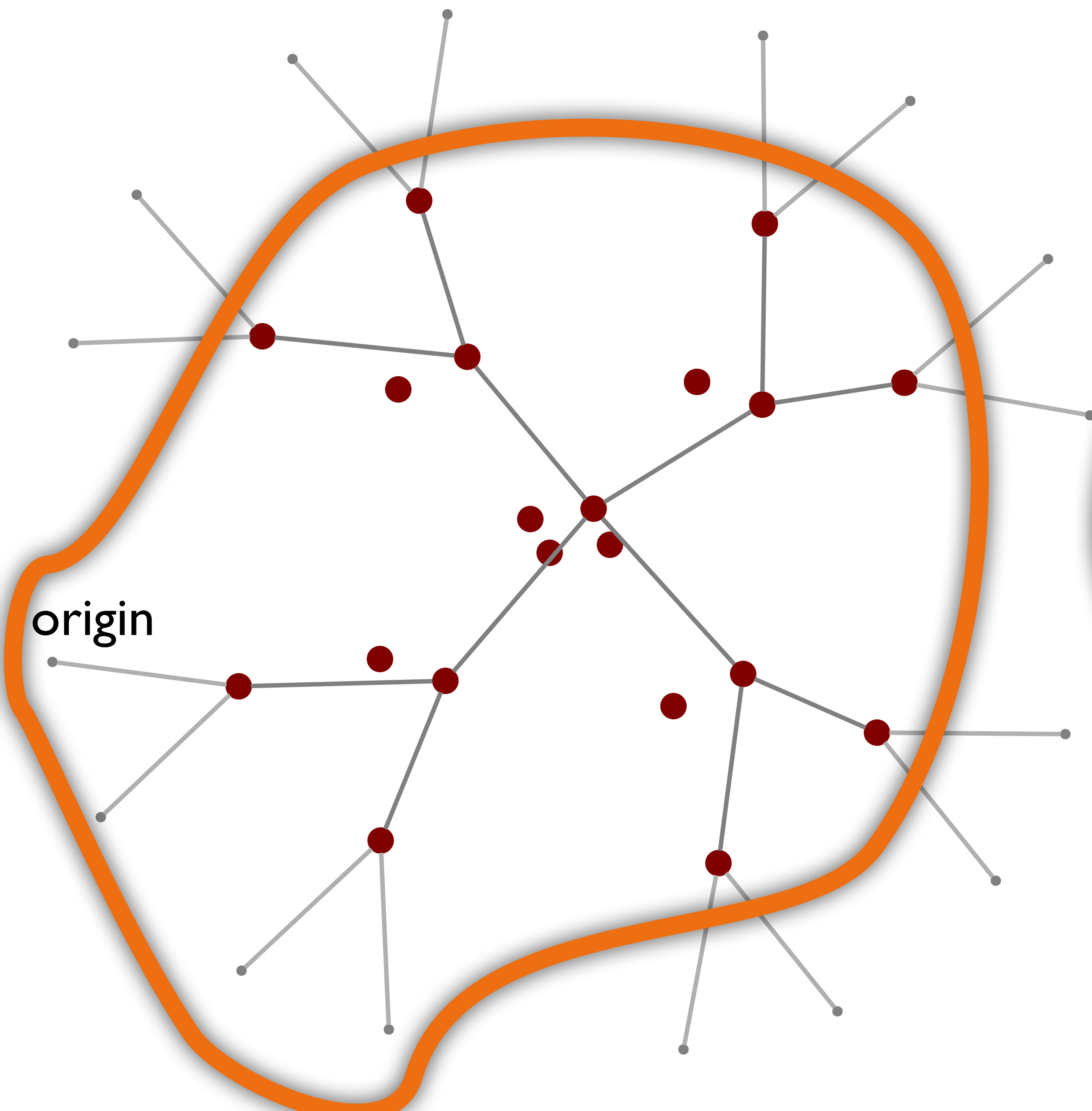
Fat tree

16 servers, 20 switches, degree 4

Jellyfish random graph

16 servers, 20 switches, degree 4

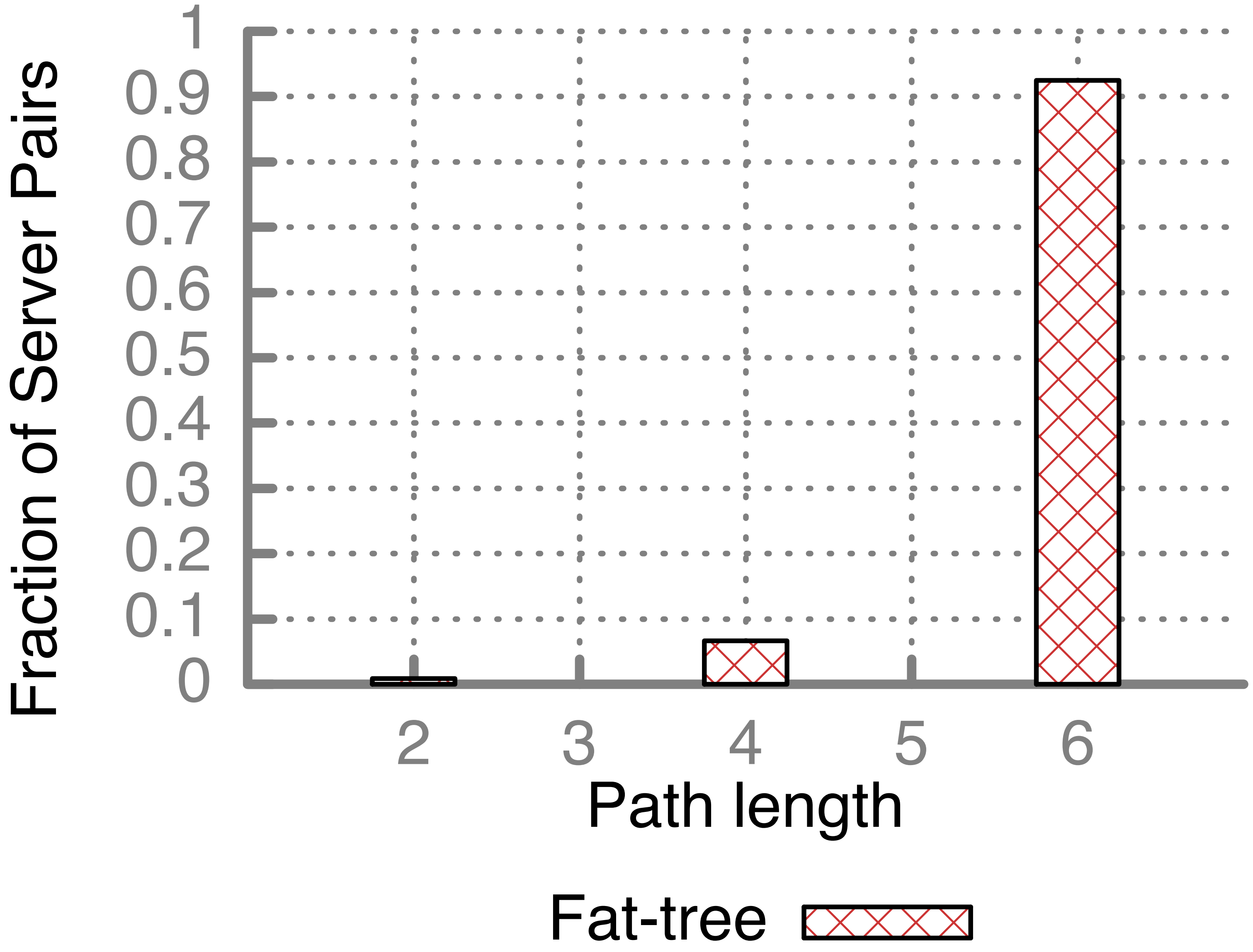
Example



Fat tree
16 servers, 20 switches, degree 4

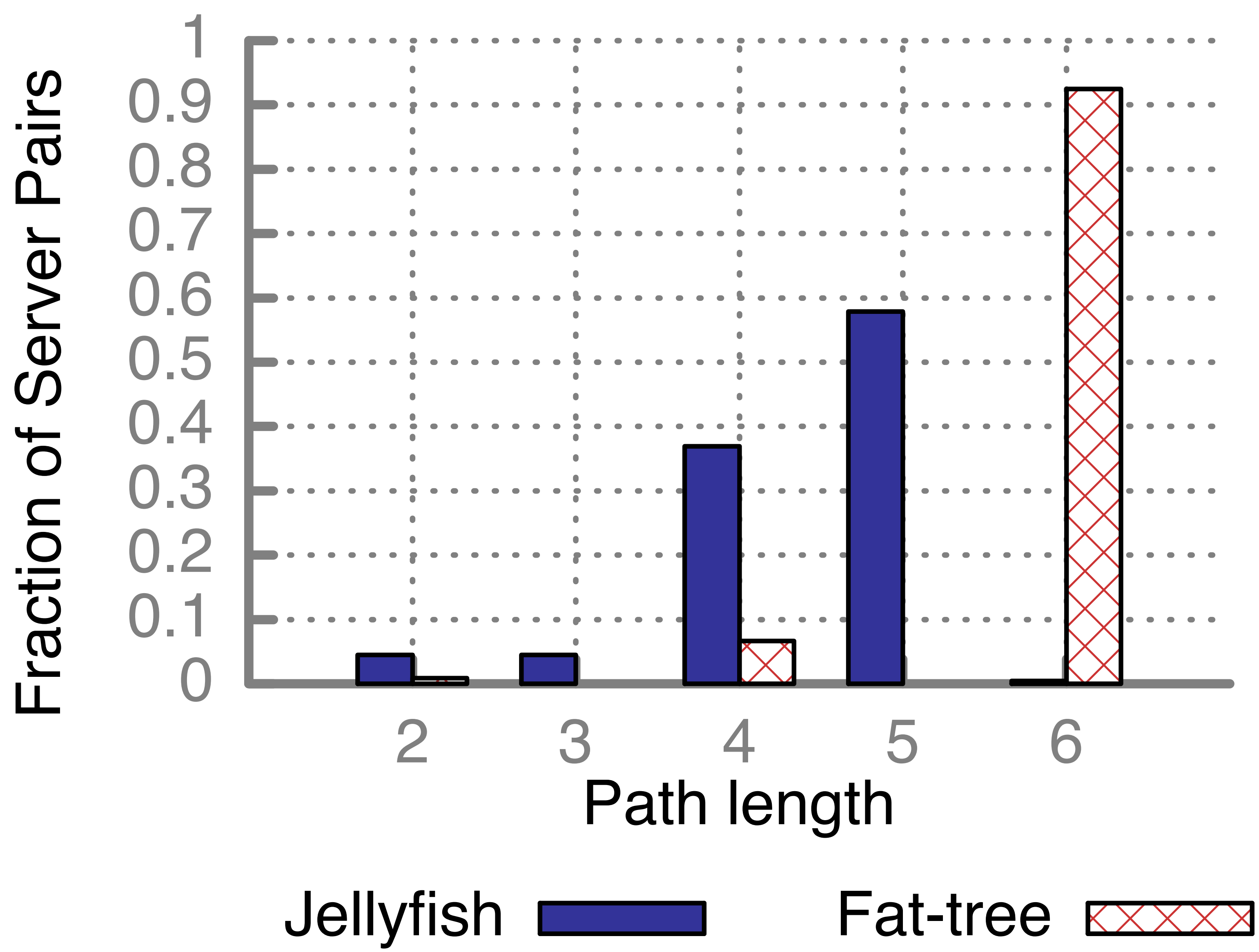
Jellyfish random graph
16 servers, 20 switches, degree 4

Jellyfish has short paths



Fat-tree with 686 servers

Jellyfish has short paths



Jellyfish, same equipment

Wrap up

Announcements

Now: David Patterson, UC Berkeley

- Distinguished Lecture
- 3 pm today, 2405 SC
- "Myths about MOOCs and Software Engineering Education"

Thursday: Nathan Farrington, Facebook

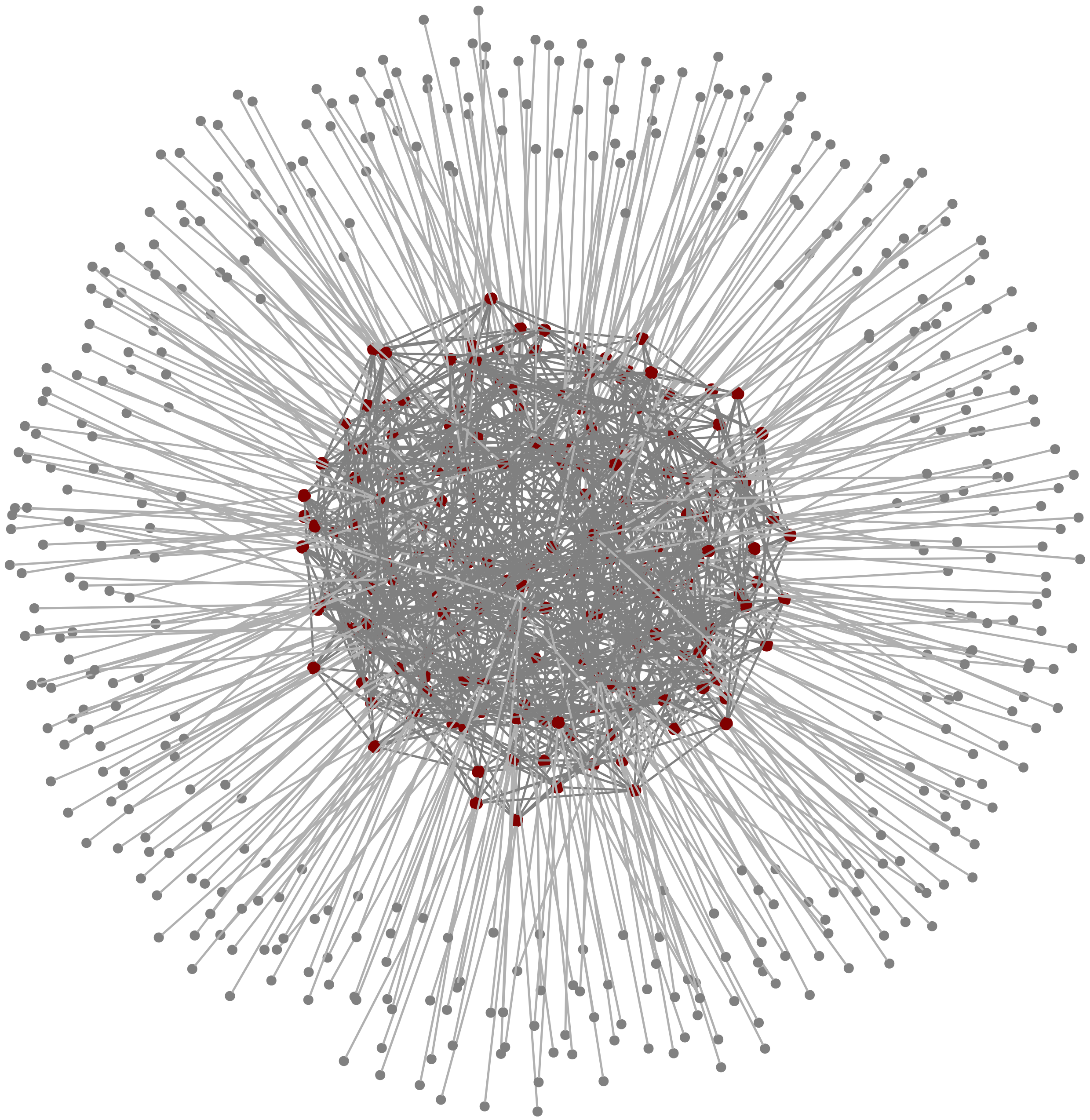
- Virtual Guest Lecture here
- Post review comment, and optionally a question for Nathan

System Design:

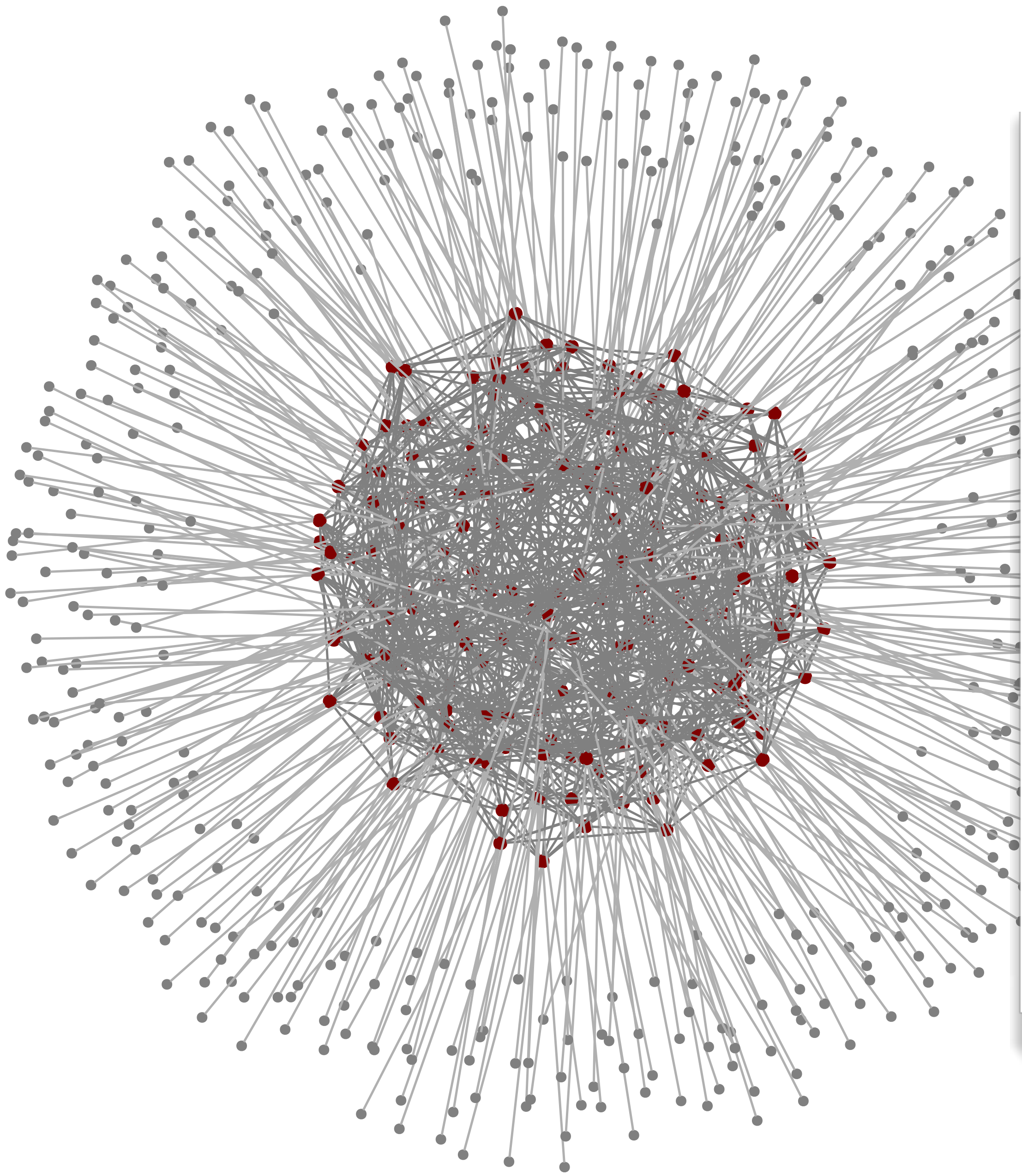
Cabling

(Not discussed in lecture)

Cabling



Cabling



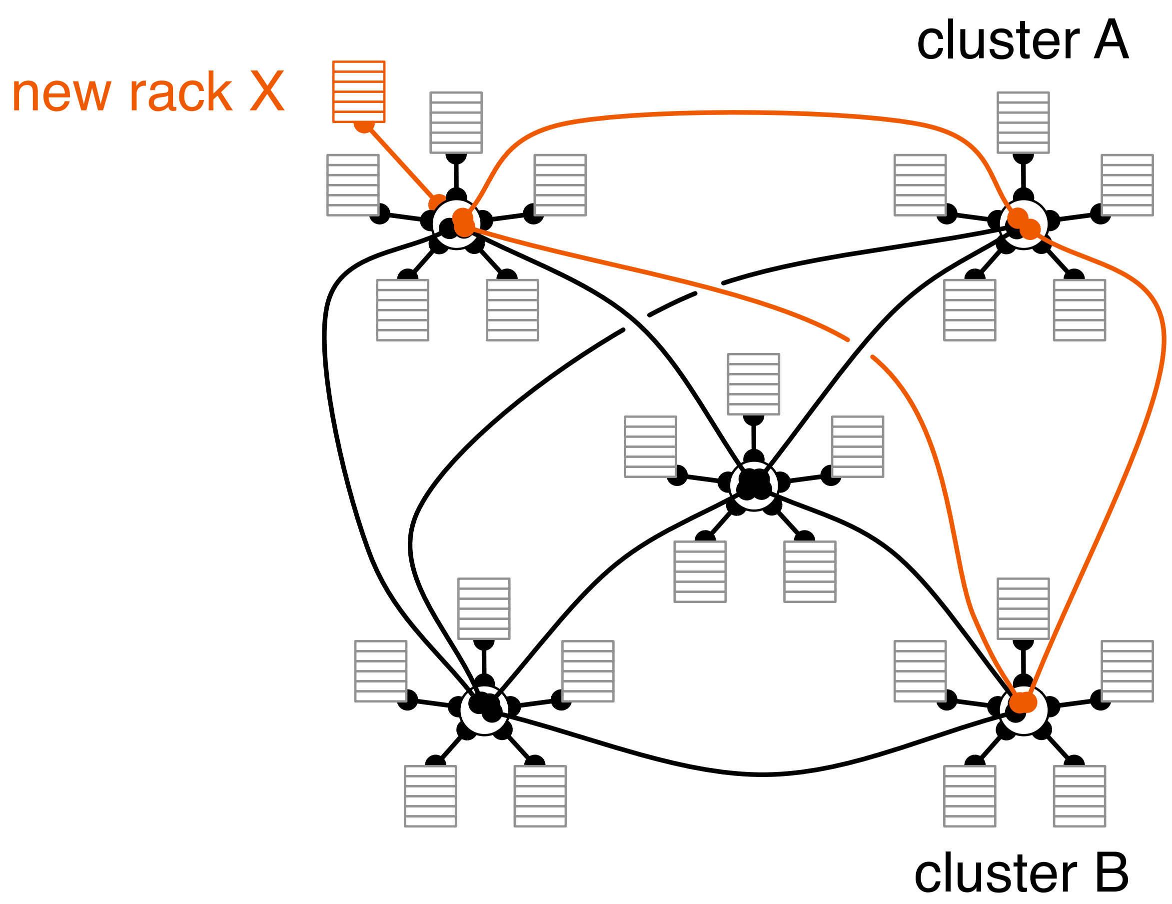
[Photo: Javier Lastras / Wikimedia]

Cabling solutions

Fewer cables

for same # servers as fat tree

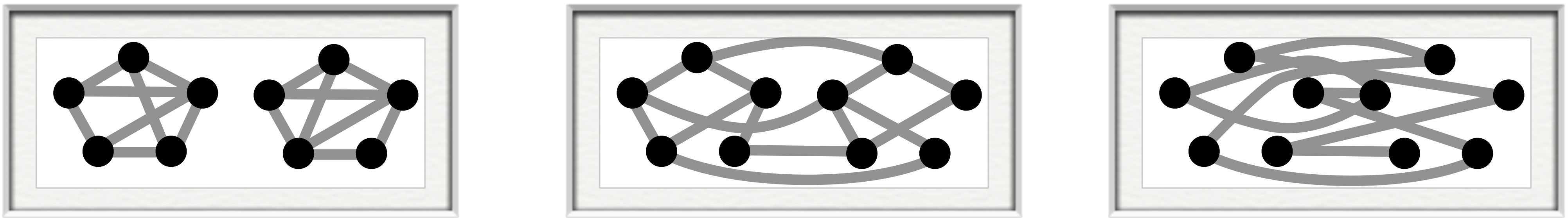
Aggregate bundles



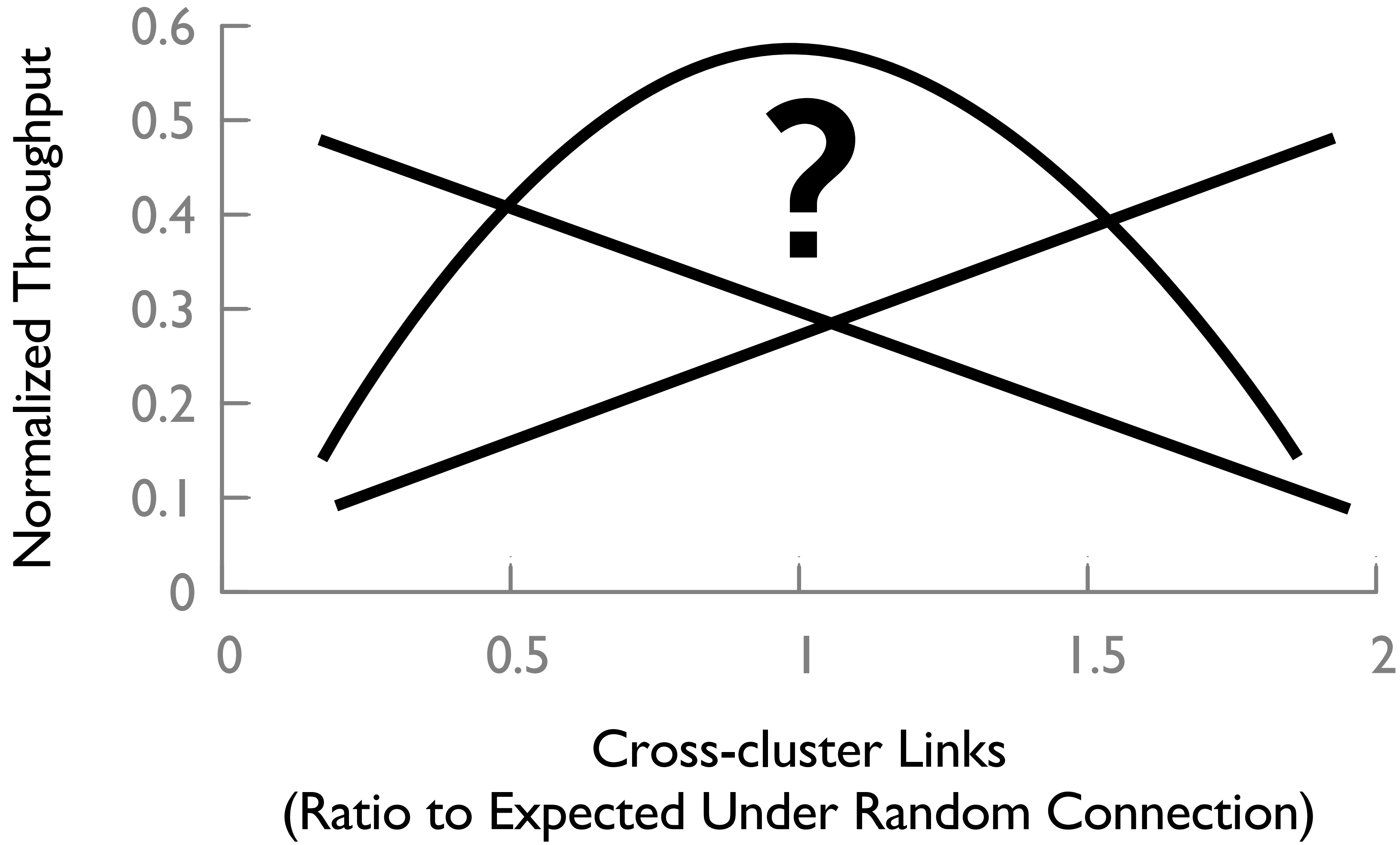
Generic optimization: Place all switches centrally

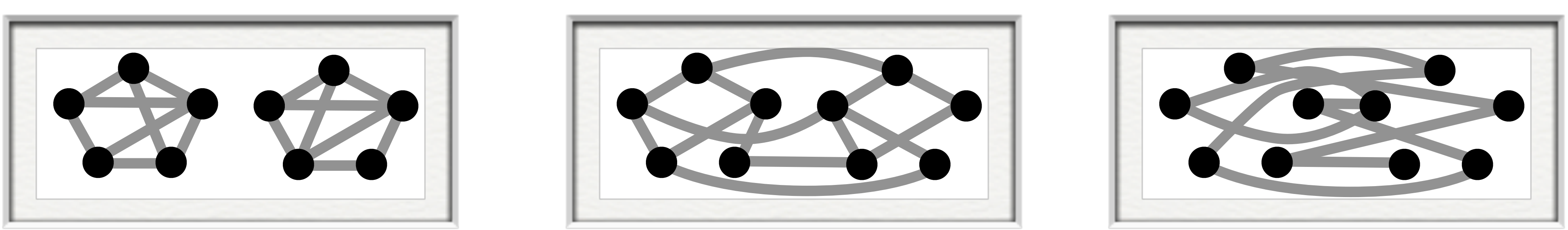
Interconnecting clusters

How many “long” cables do we actually need?

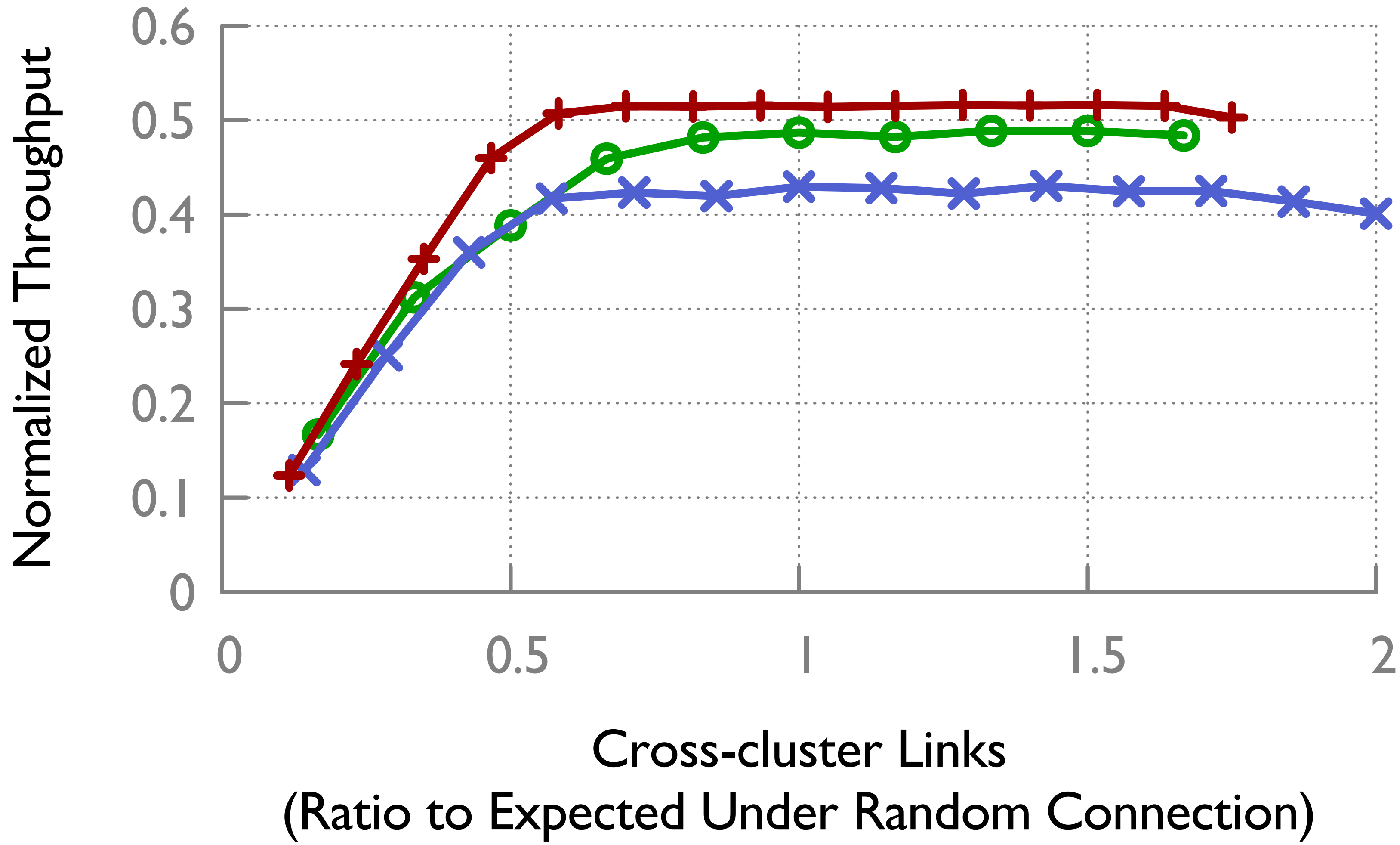


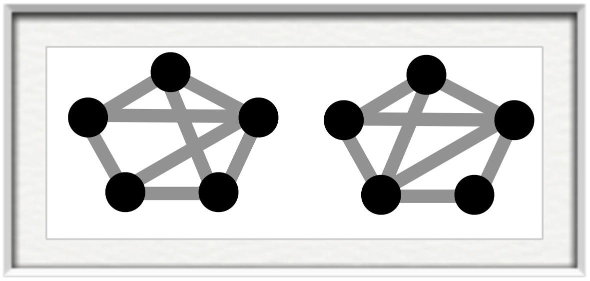
Interconnecting clusters

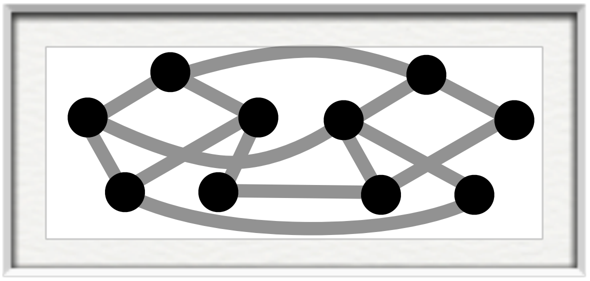


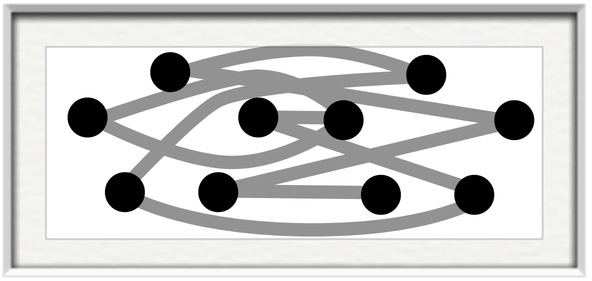


Interconnecting clusters

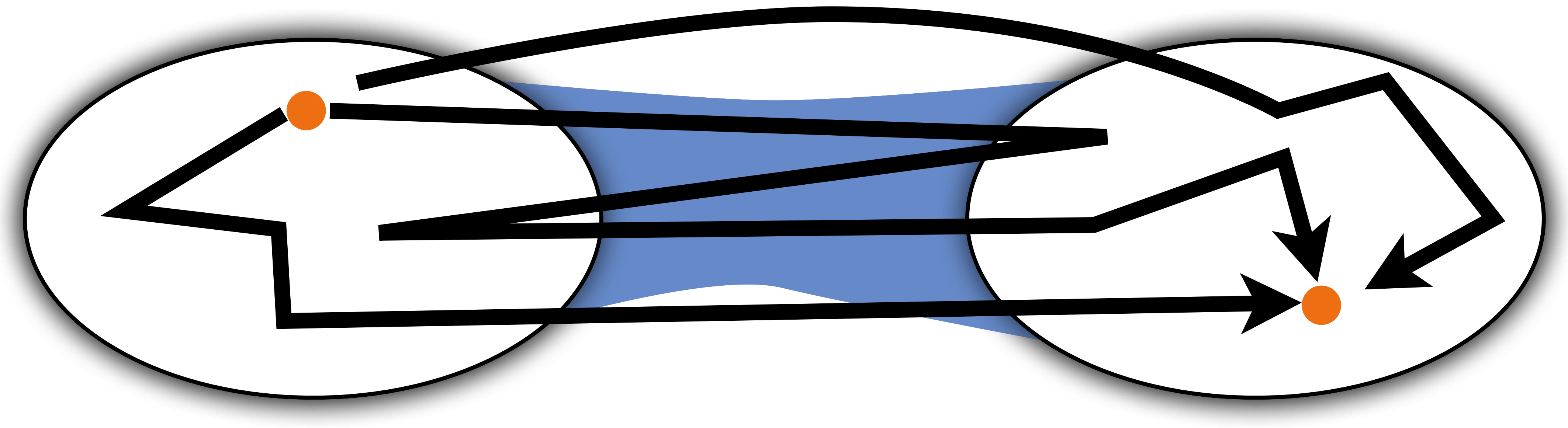




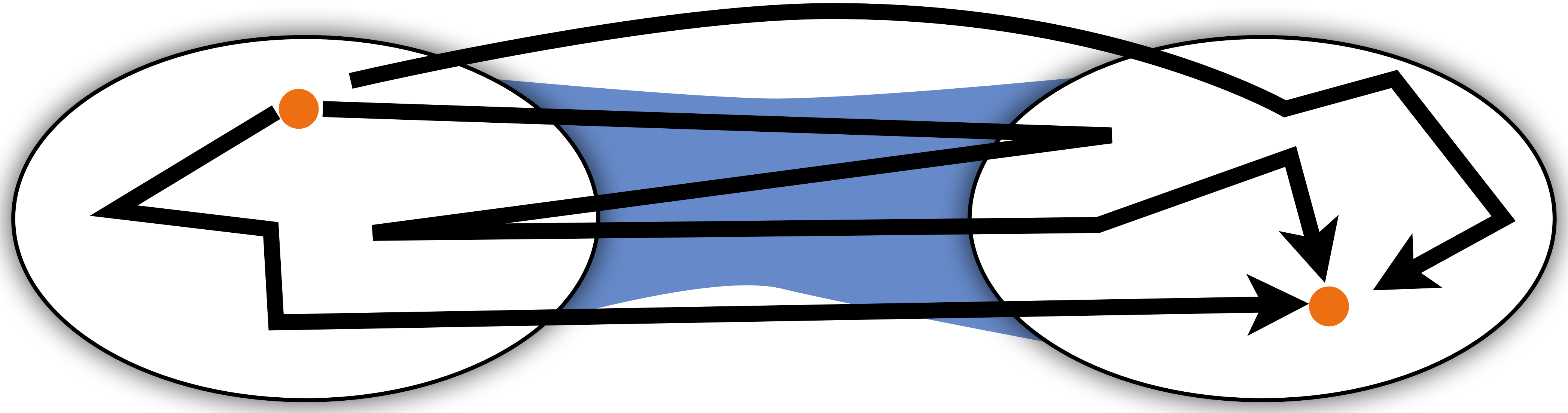


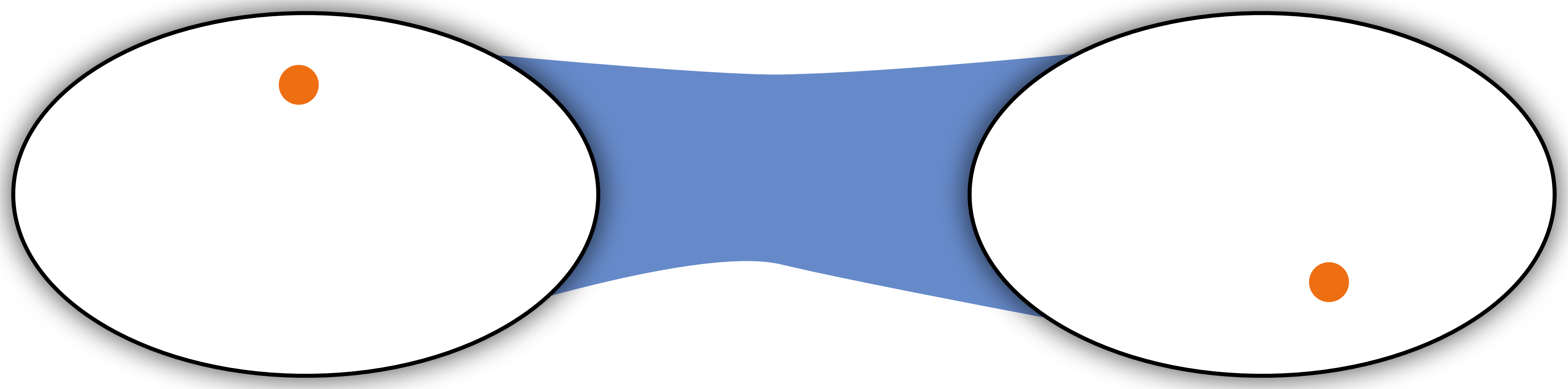


Intuition

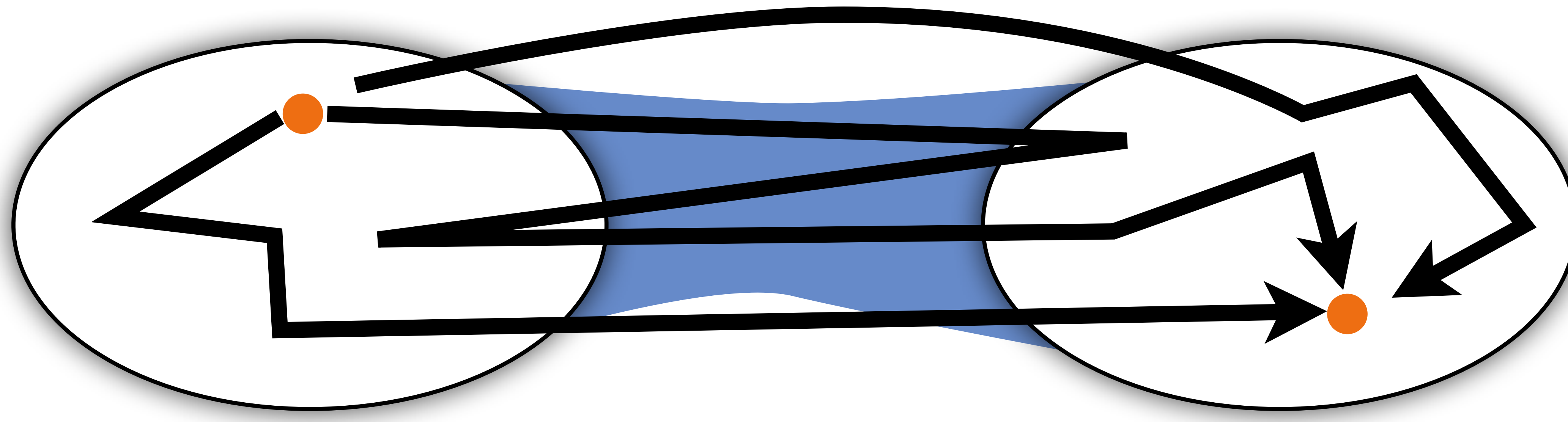


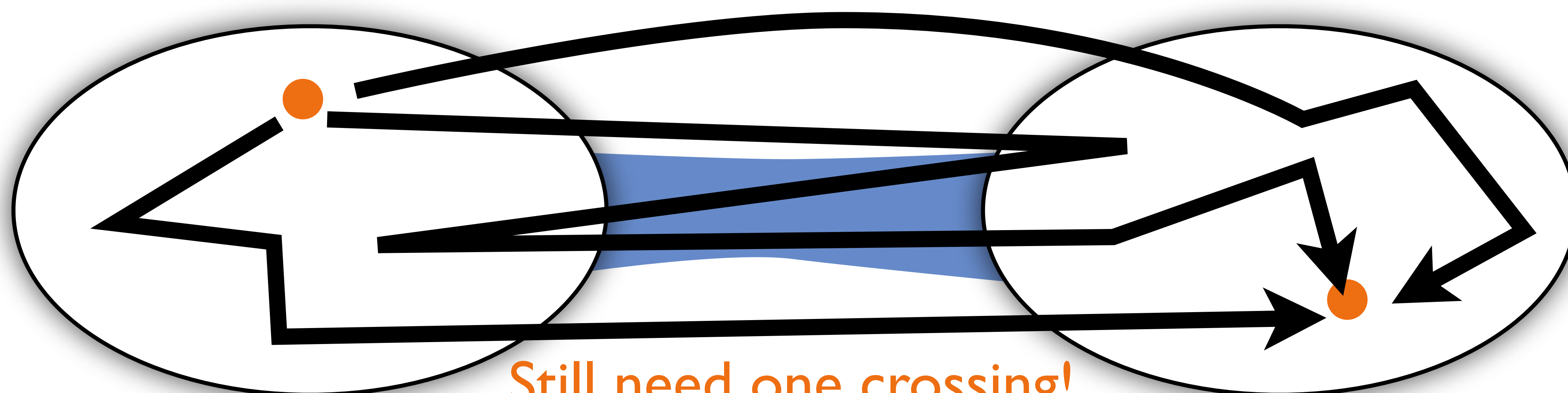
Intuition





Intuition

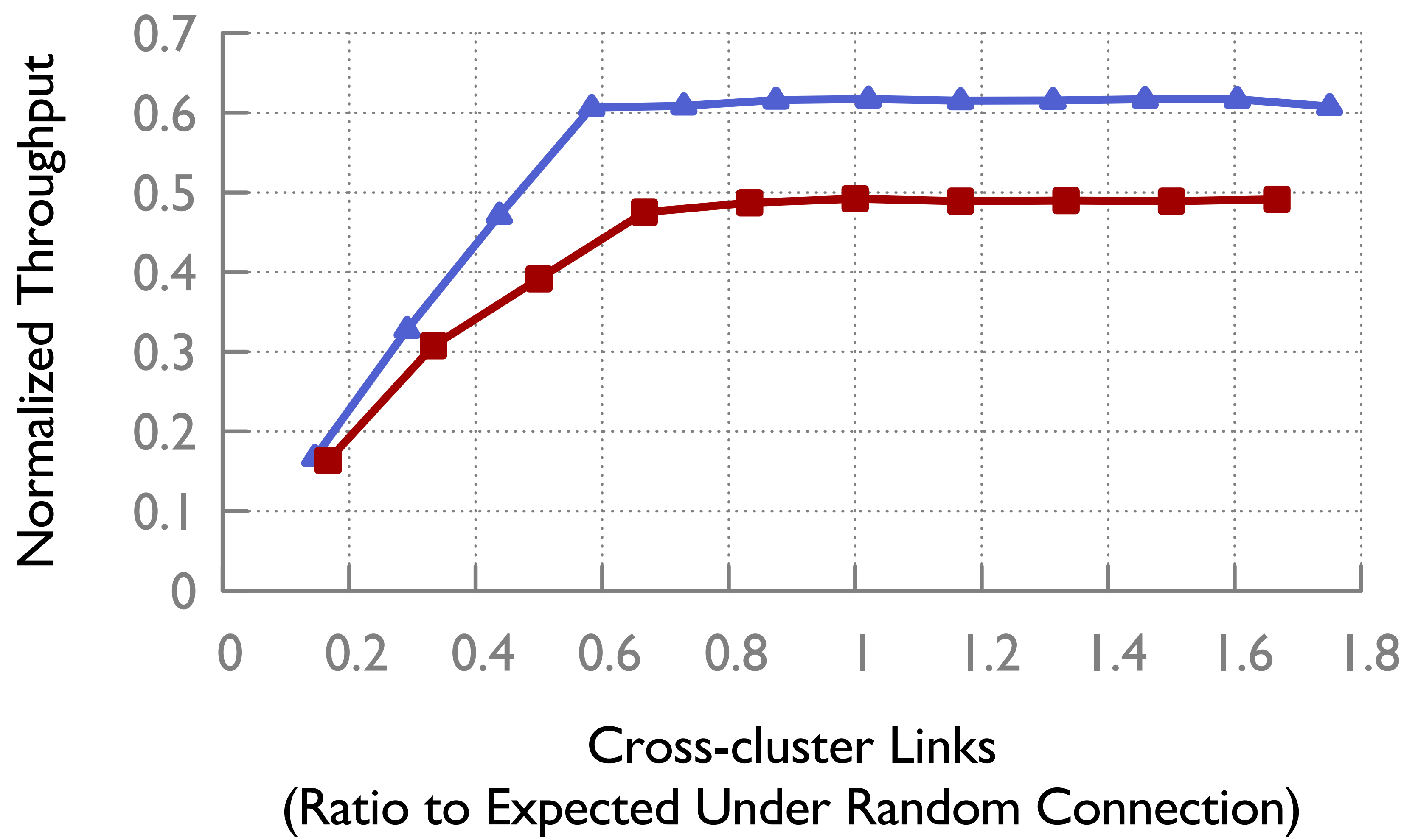




Still need one crossing!

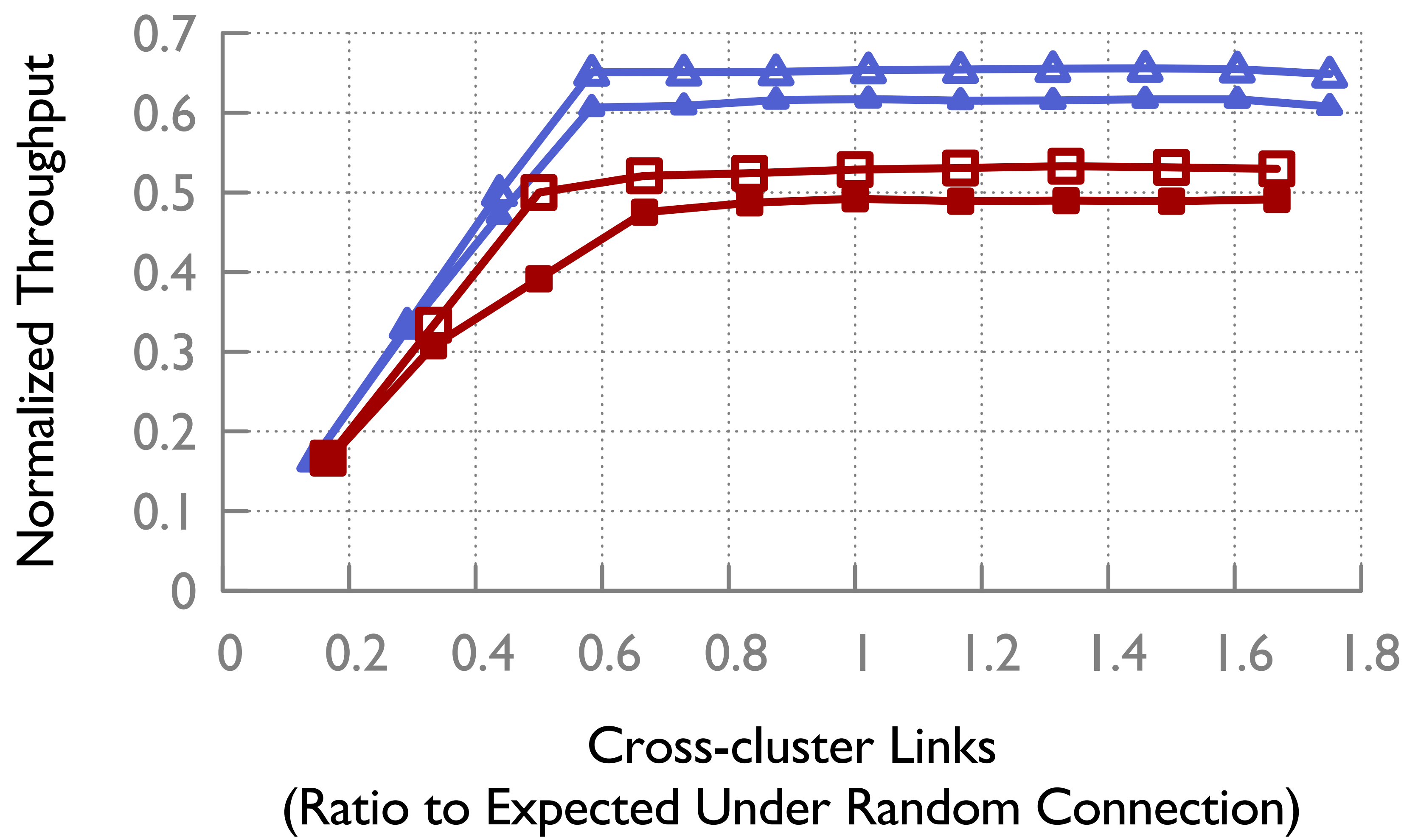
Throughput should drop when less than $\Theta\left(\frac{1}{APL}\right)$ of total capacity crosses the cut!

Explaining throughput



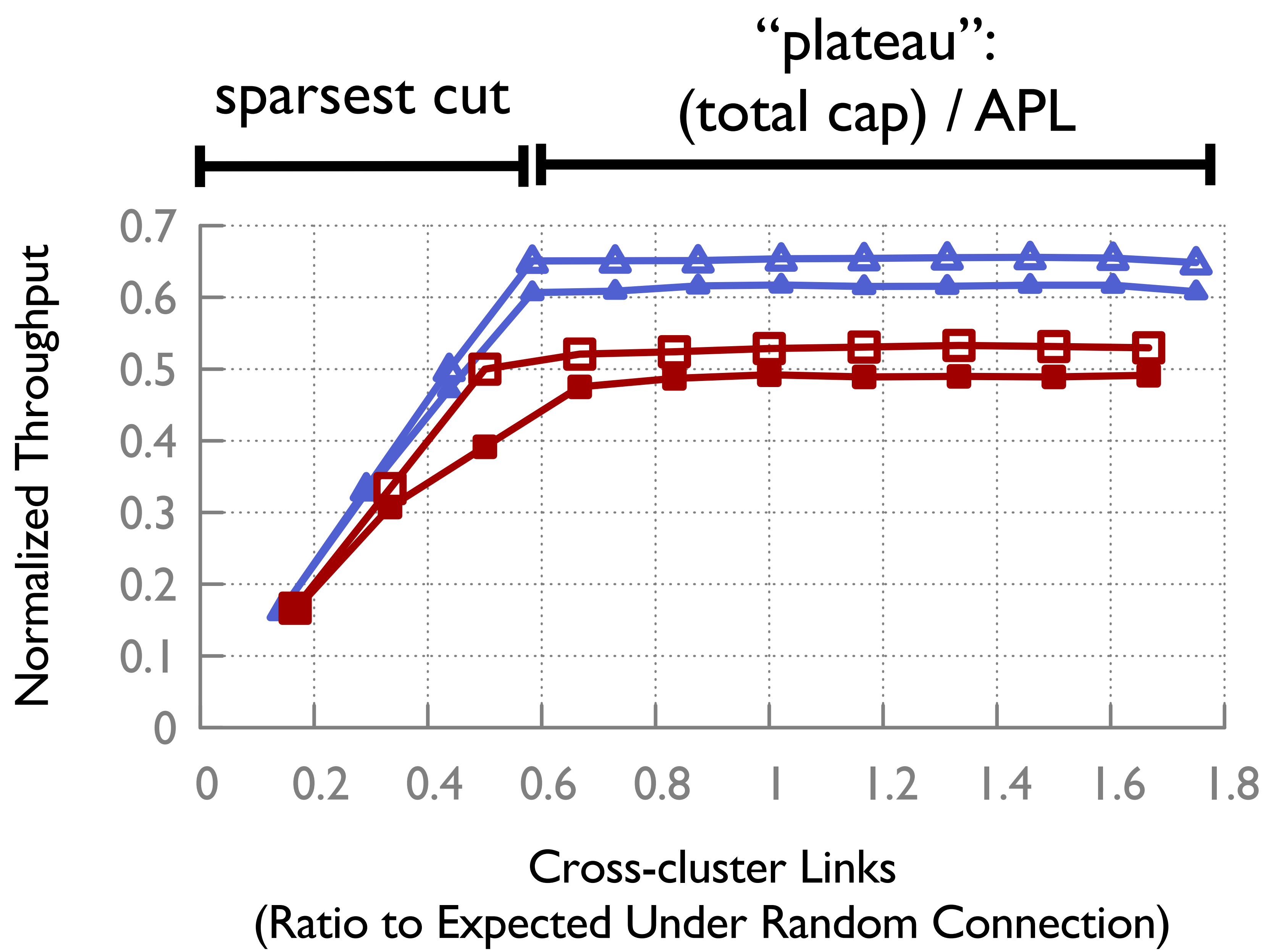
Explaining throughput

Upper bounds...

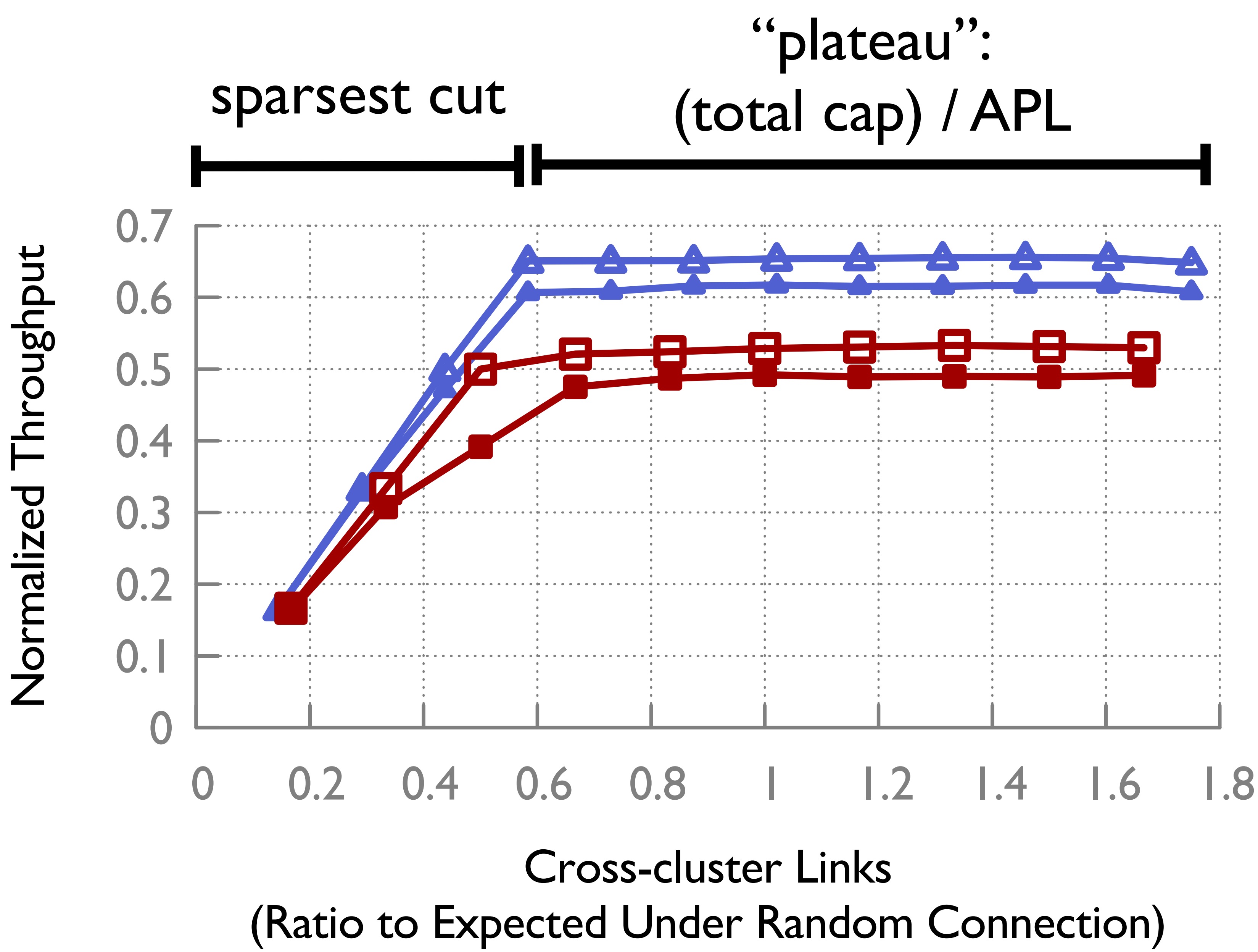


And constant-factor matching lower bounds in special case.

Two regimes of throughput



Two regimes of throughput



High-capacity switches needn't be clustered

Bisection bandwidth is poor predictor of performance!

Cables can be localized