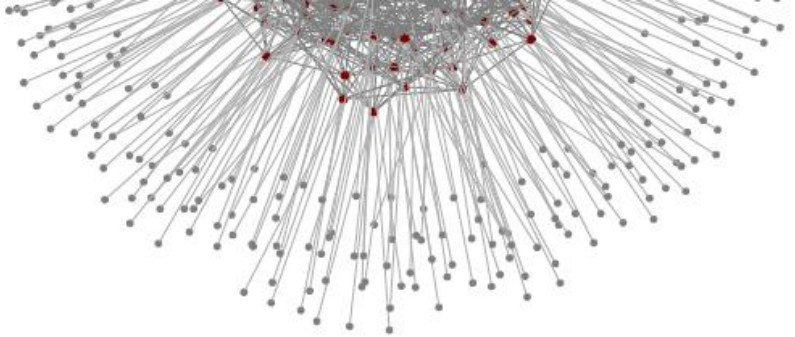
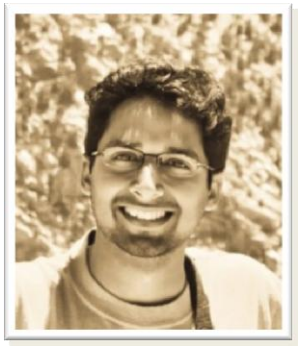


Jellyfish: Networking Data Centers Randomly

Speaker: Chi-Yao Hong, UIUC



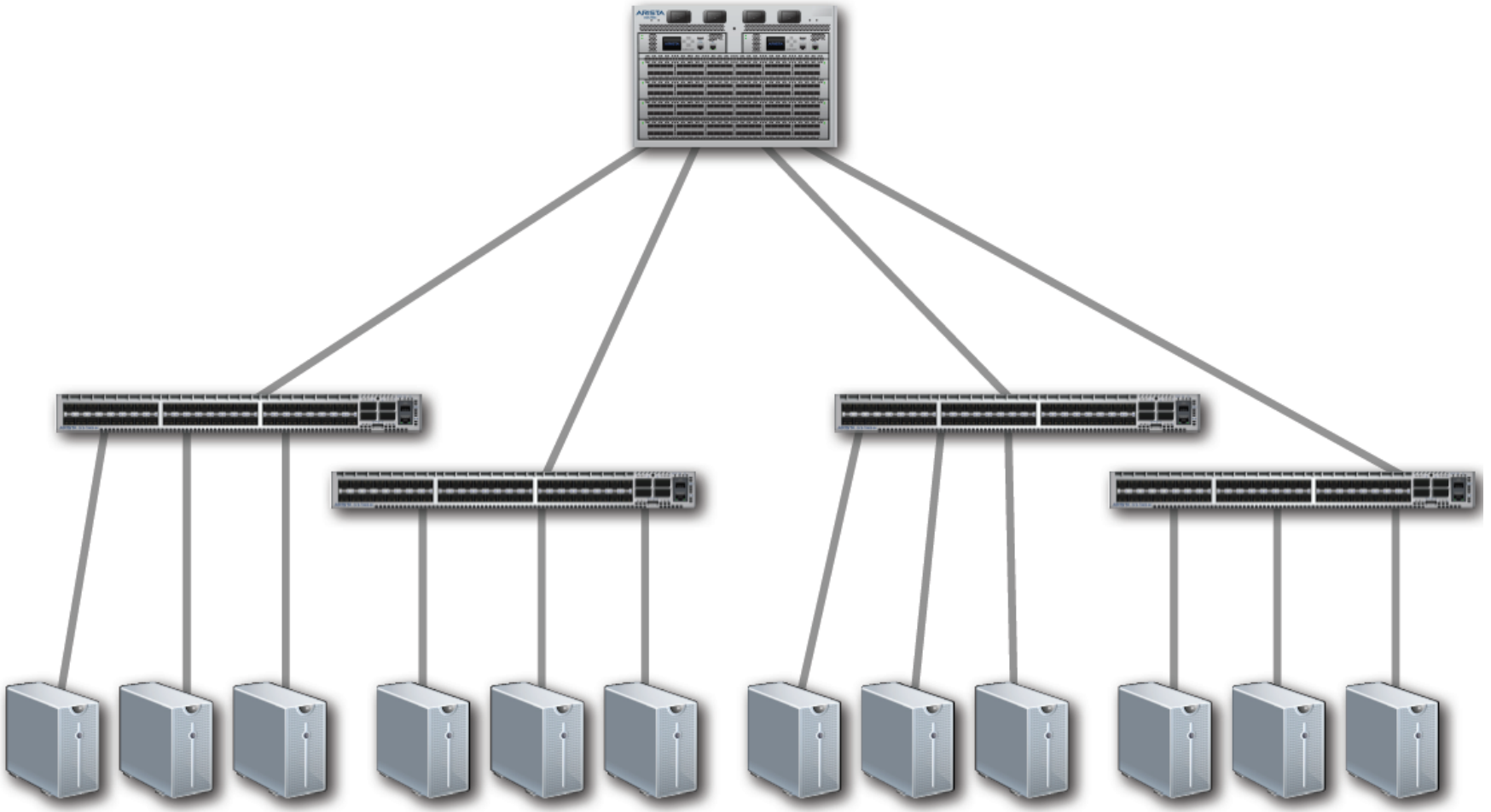
Ankit Singla
UIUC

Lucian Popa
HP Labs

Brighten Godfrey
UIUC

“It is anticipated that the whole of the populous parts of the United States will, within two or three years, be covered with net-work like a spider's web.”

— *The London Anecdotes*, 1848



We have opportunity to build
topologies with great deal more freedom

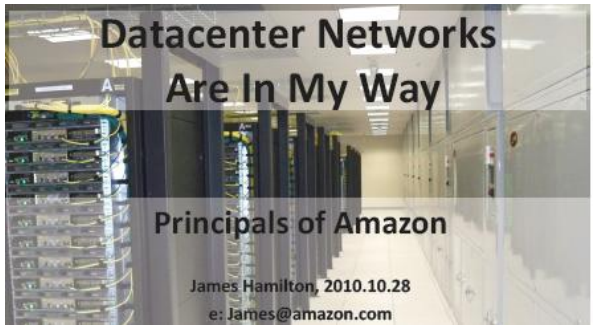
Two goals

High throughput

Eliminate bottlenecks
Agile placement of VMs

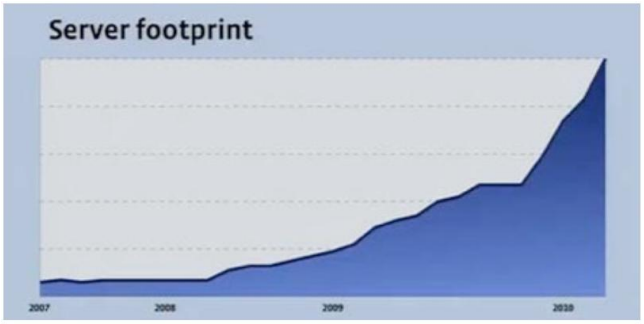
Incremental expansion

Easily add/replace servers & switches



Incremental expansion

Facebook "adding capacity on a daily basis"



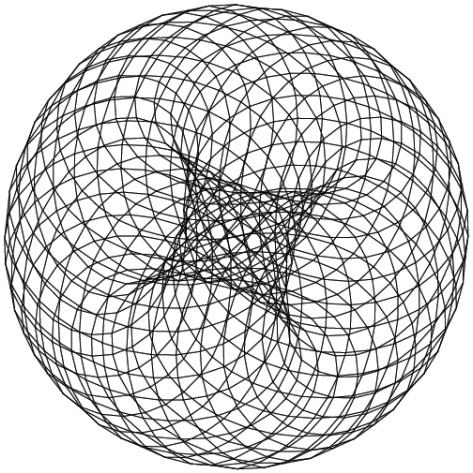
2007 08 09 10

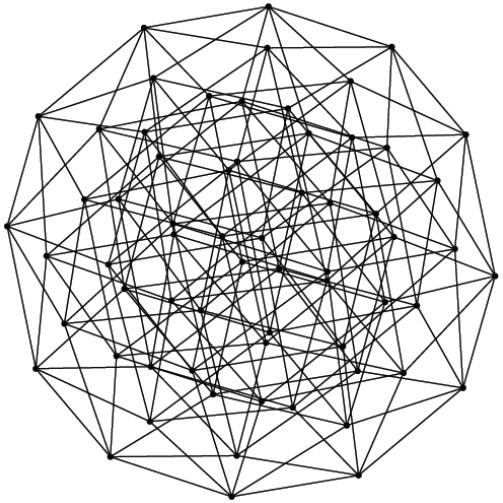
Commercial products

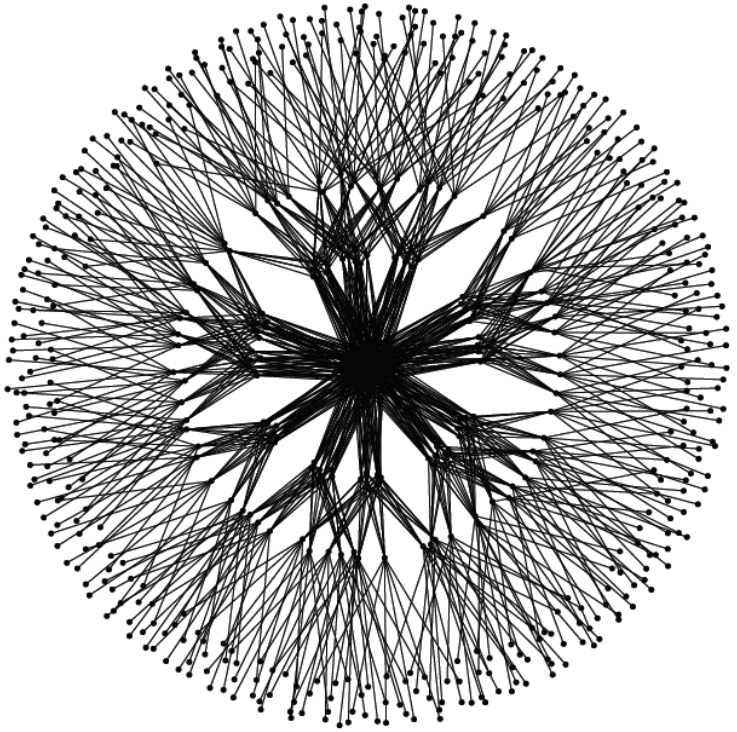
- SGI Ice Cube ("Expandable Modular Data Center")
- HP EcoPod ("Pay-as-you-grow")

You can add servers, but what about the network?

Today's structured networks







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

Structure constrains expansion

Coarse design points

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

3-level fat trees, commodity switches:

- 24-port switch ➔ 3,456 servers
- 32-port switch ➔ 8,192 servers
- 48-port switch ➔ 27,648 servers

Workarounds exist, but unclear how to maintain structure incrementally

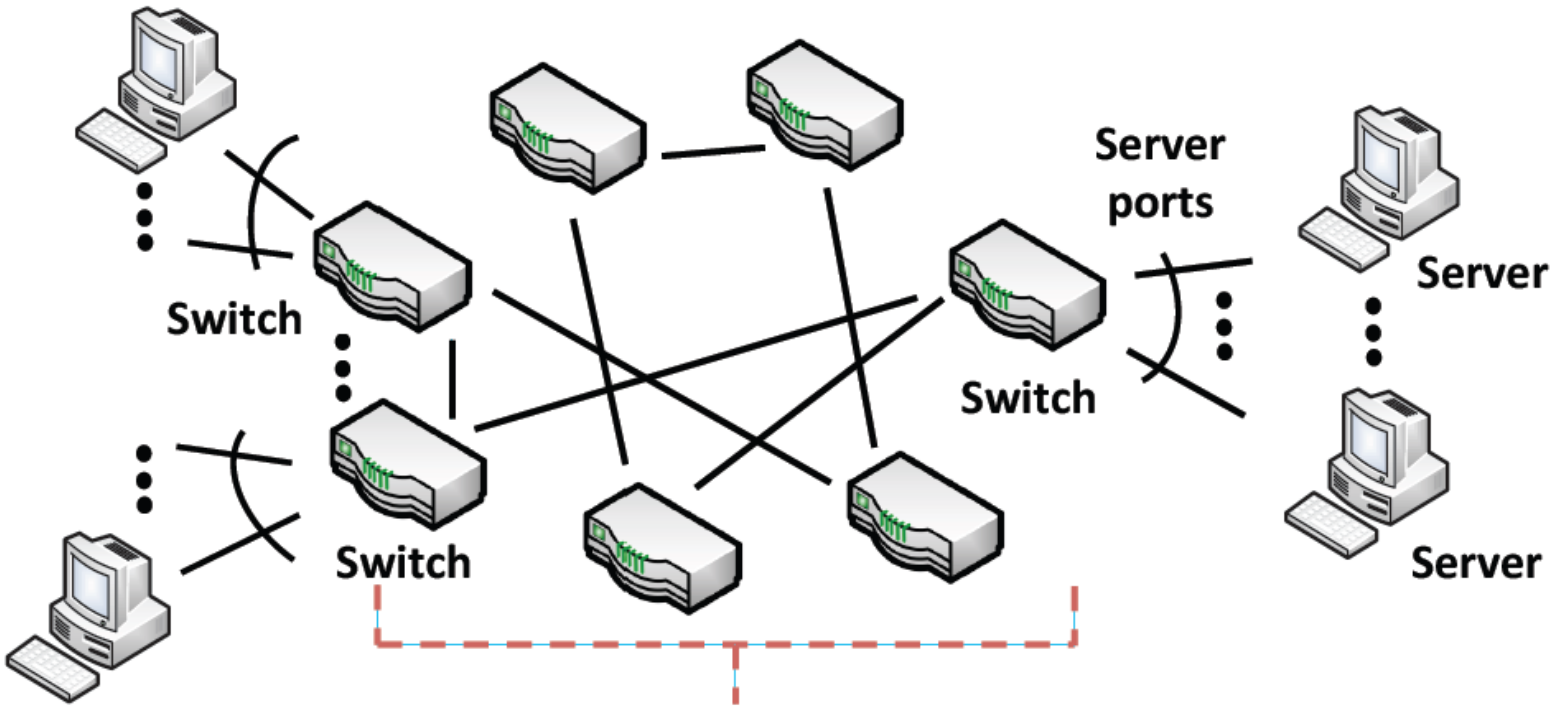
- Overutilize network? Uneven / constrained bandwidth
- Leave ports free for later? Wasted investment

Our solution

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

Jellyfish

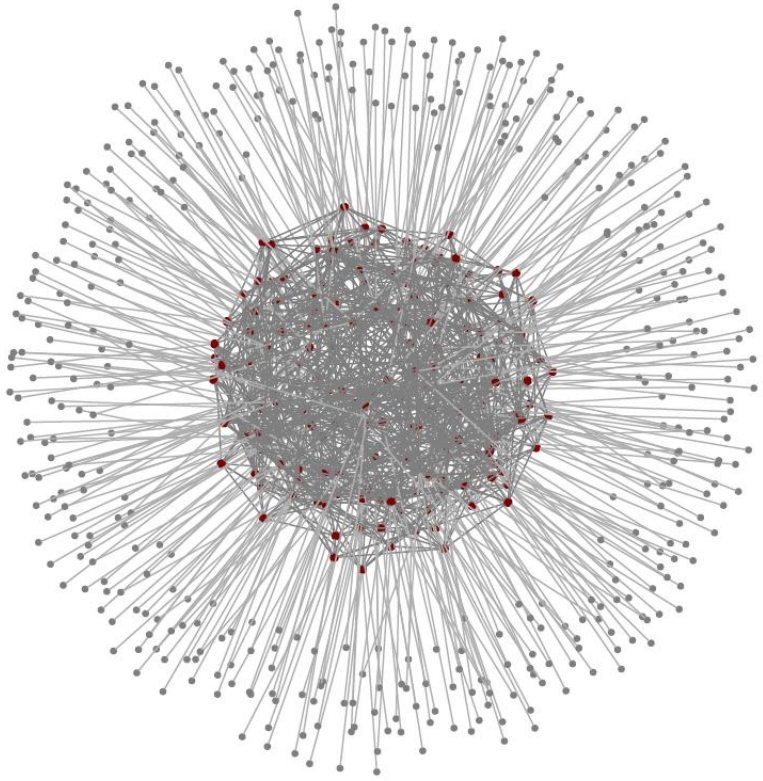
Jellyfish: The topology



Randomly selected from all valid graphs

Switches are nodes

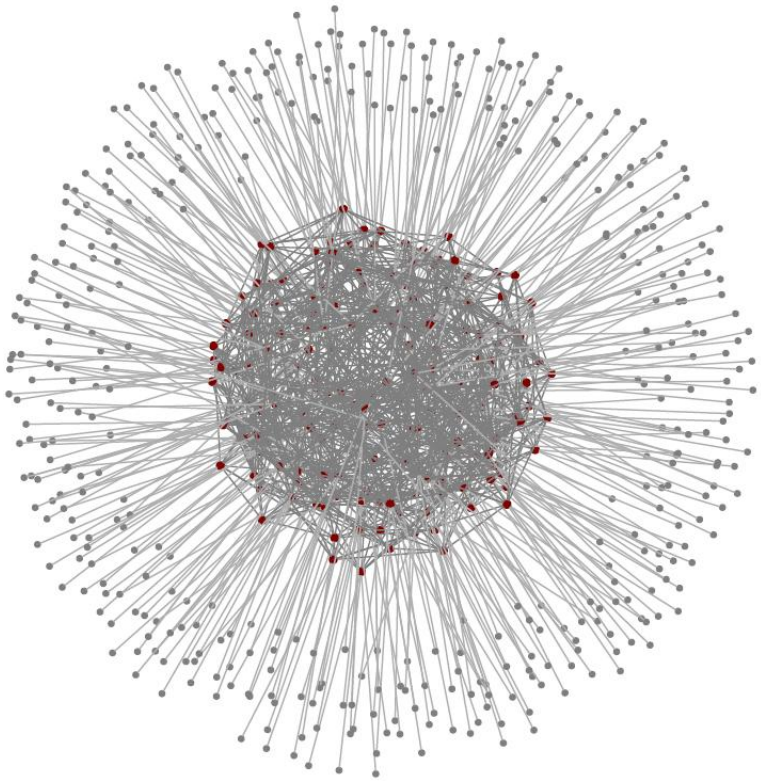
Capacity as a fluid



Jellyfish random graph

432 servers, 180 switches, degree 12

Capacity as a fluid



Jellyfish random graph

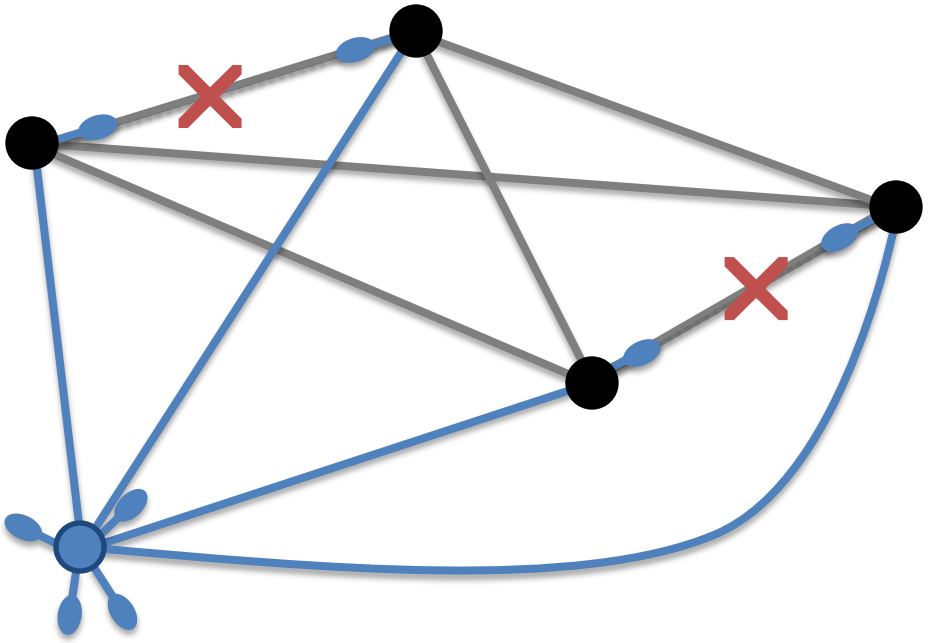
432 servers, 180 switches, degree 12

Jellyfish

Arctapodema
(http://goo.gl/KoAC3)

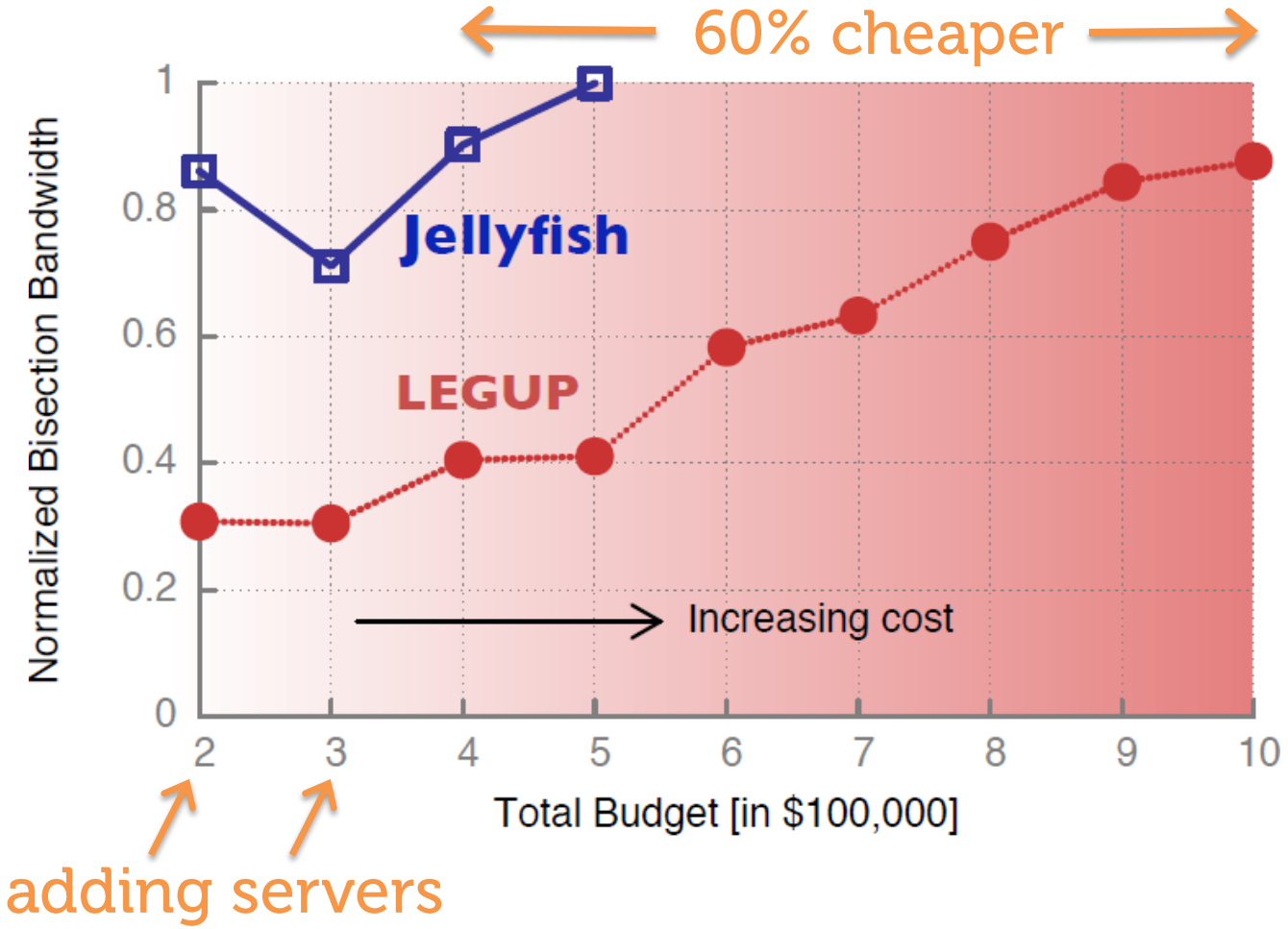
Construction & Expansion

Building Jellyfish



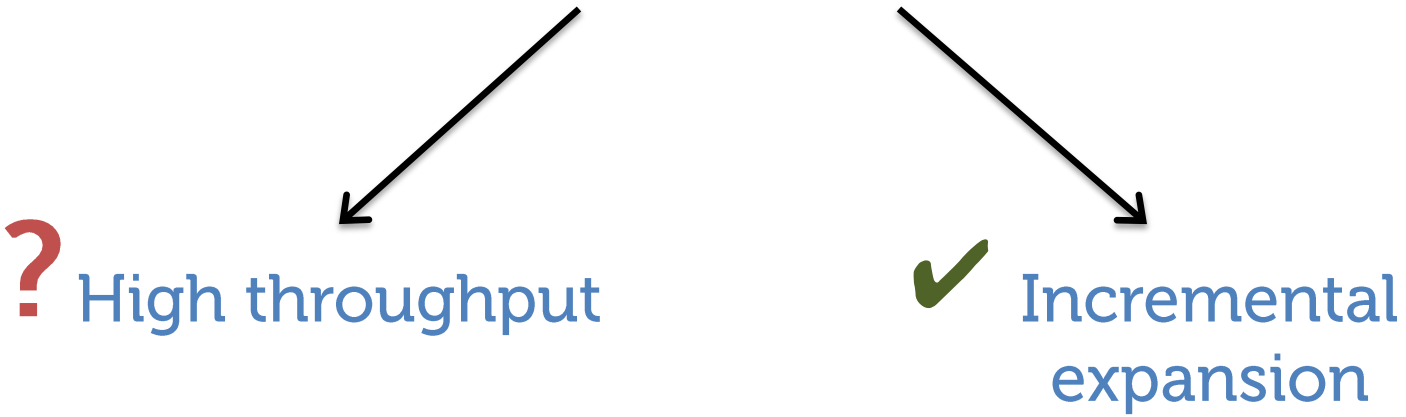
Same procedure for graph construction and incremental expansion

Quantifying expandability

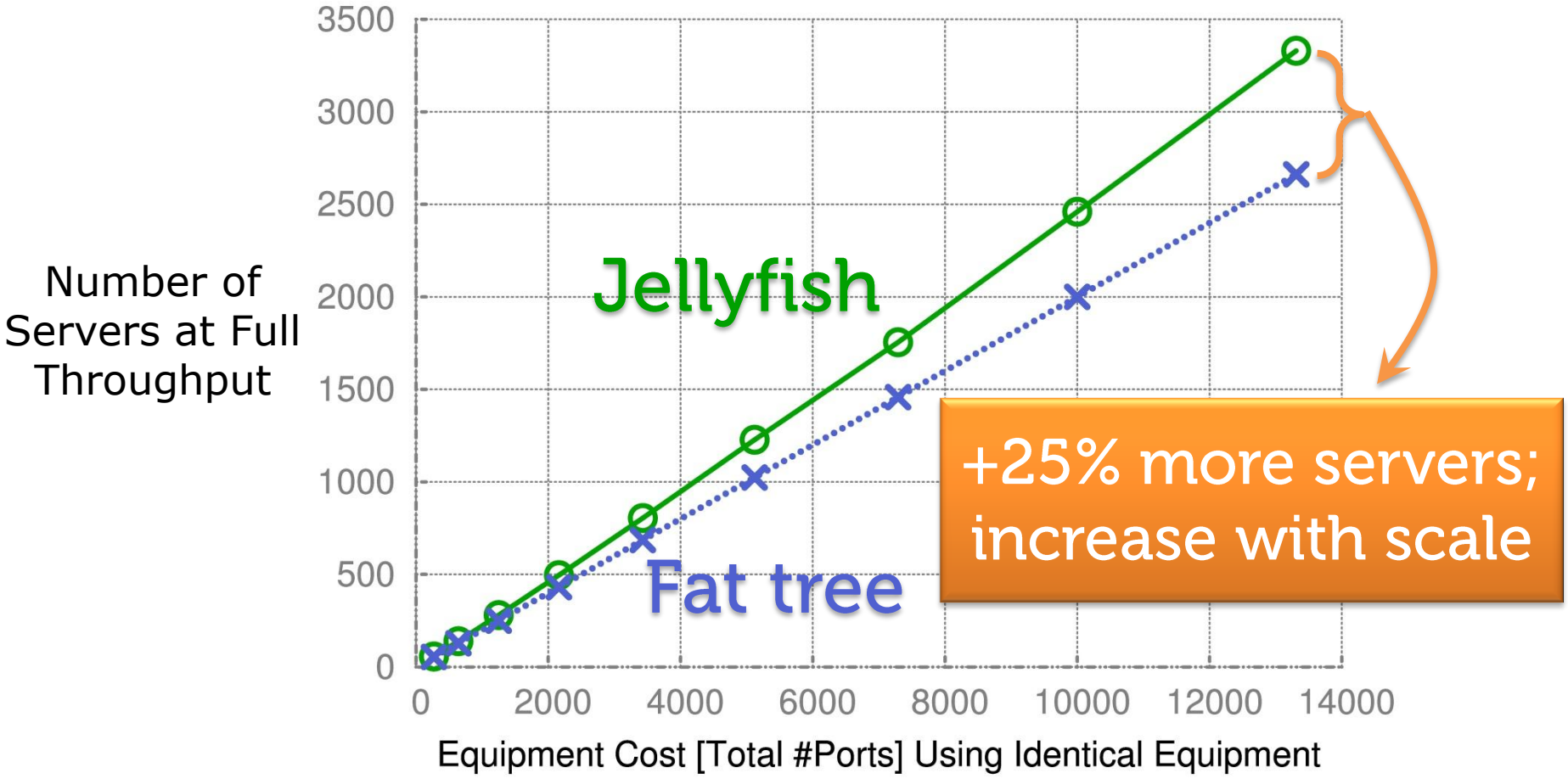


LEGUP: [Curtis, Keshav, Lopez-Ortiz, CoNEXT'10]

Two goals



Throughput: Jellyfish vs. Fat tree



Packet level simulation; random permutation traffic

Suspicious?

Intuition: Why Jellyfish works?

- … and can we do better?

From theory to systems

- Routing?
- Congestion Control?
- Cabling?
- Failure Resilience?

Intuition

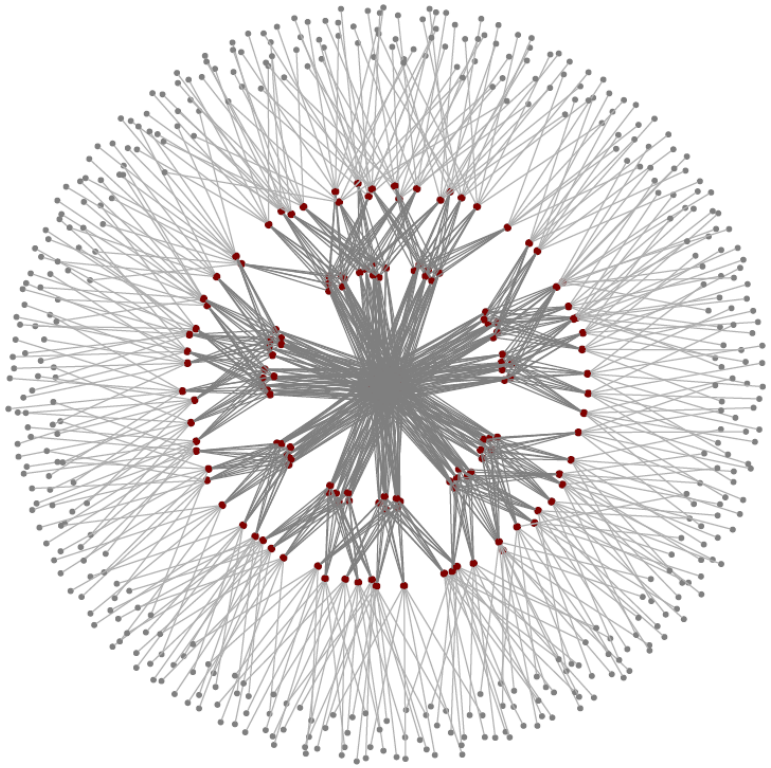
If we fully utilize all available capacity …

$$\sum_{\forall \text{links}} \text{capacity}(link)$$

$$\text{Number of flows at full throughput (1 Gbps)} = \frac{\text{total network capacity}}{\text{capacity used per flow}} = 1\ \text{Gbps} \bullet \text{mean path length}$$

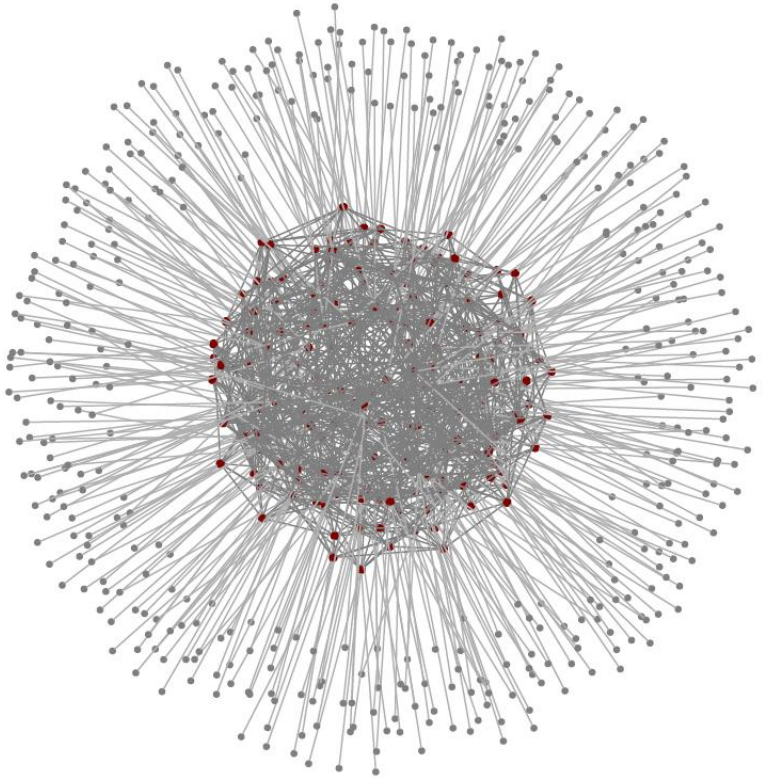
Mission:
minimize average path length

Example



Fat tree

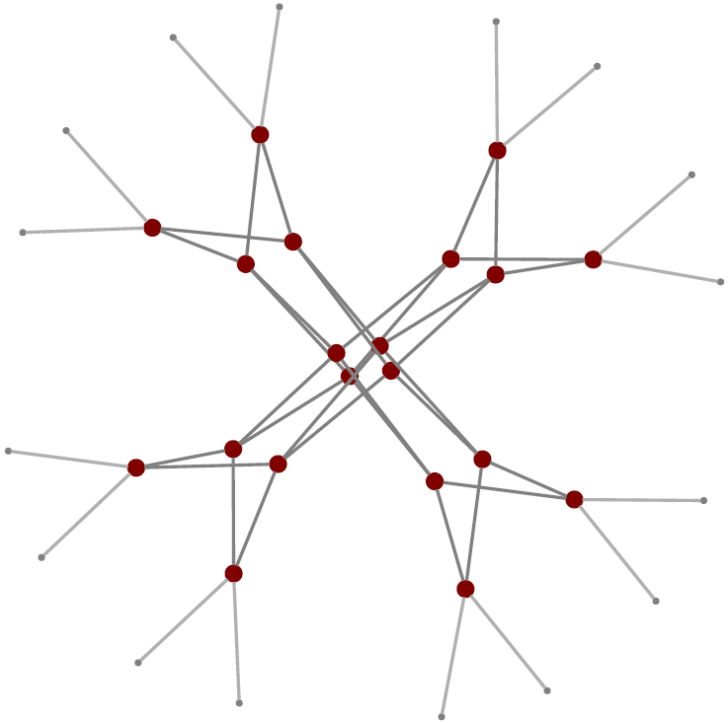
432 servers, 180 switches, degree 12



Jellyfish random graph

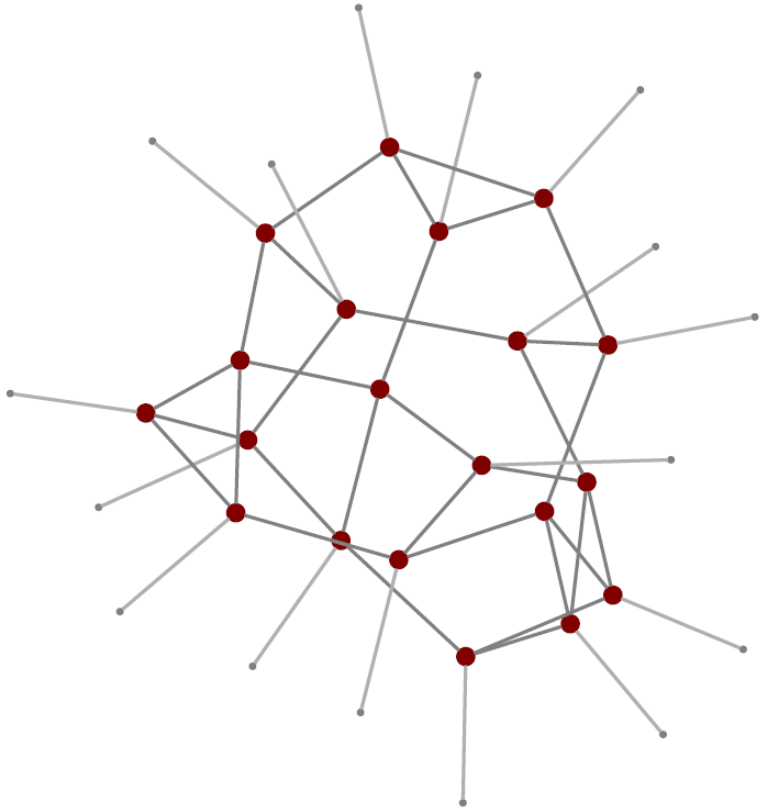
432 servers, 180 switches, degree 12

Example



Fat tree

16 servers, 20 switches, degree 4

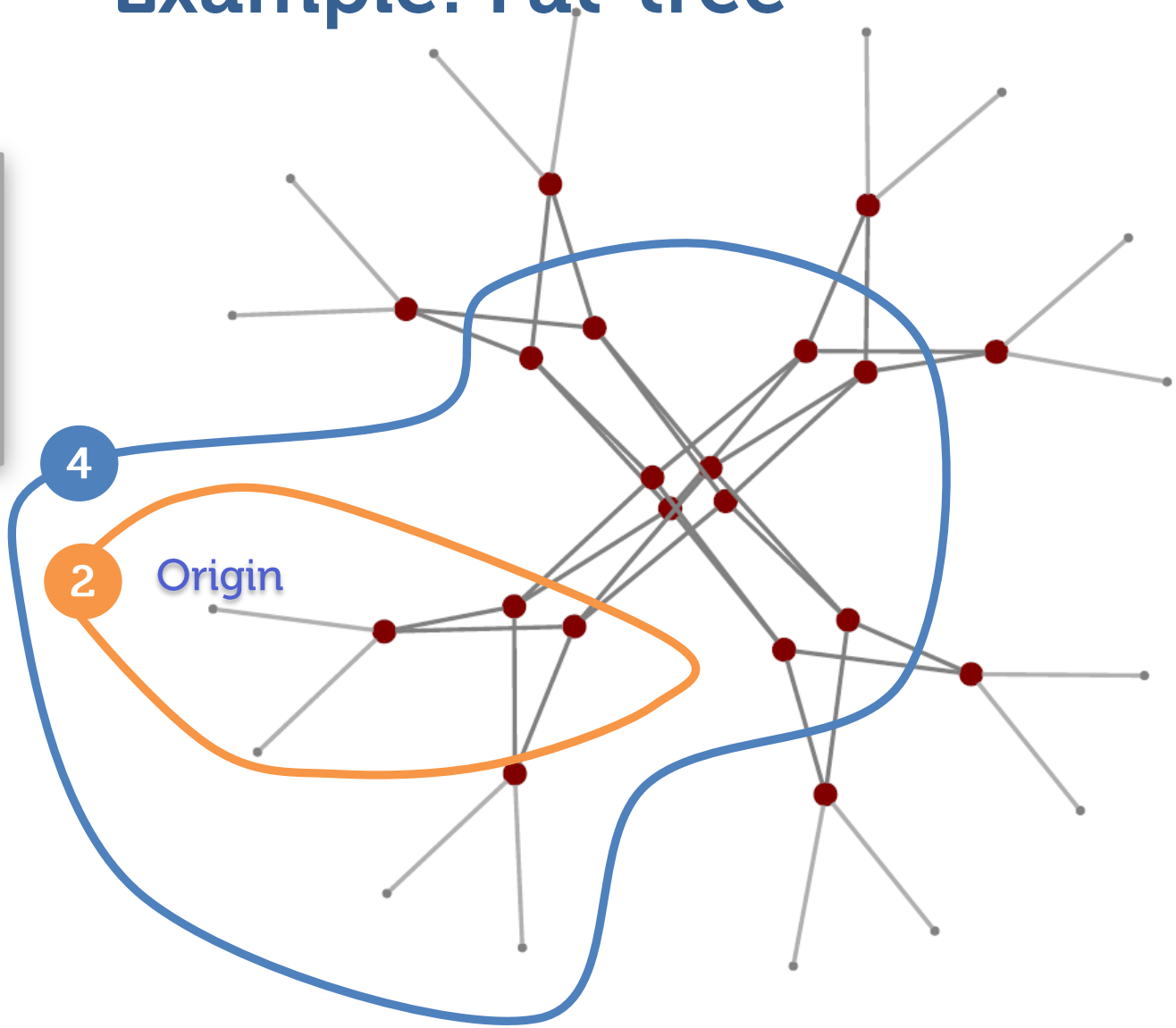


Jellyfish random graph

16 servers, 20 switches, degree 4

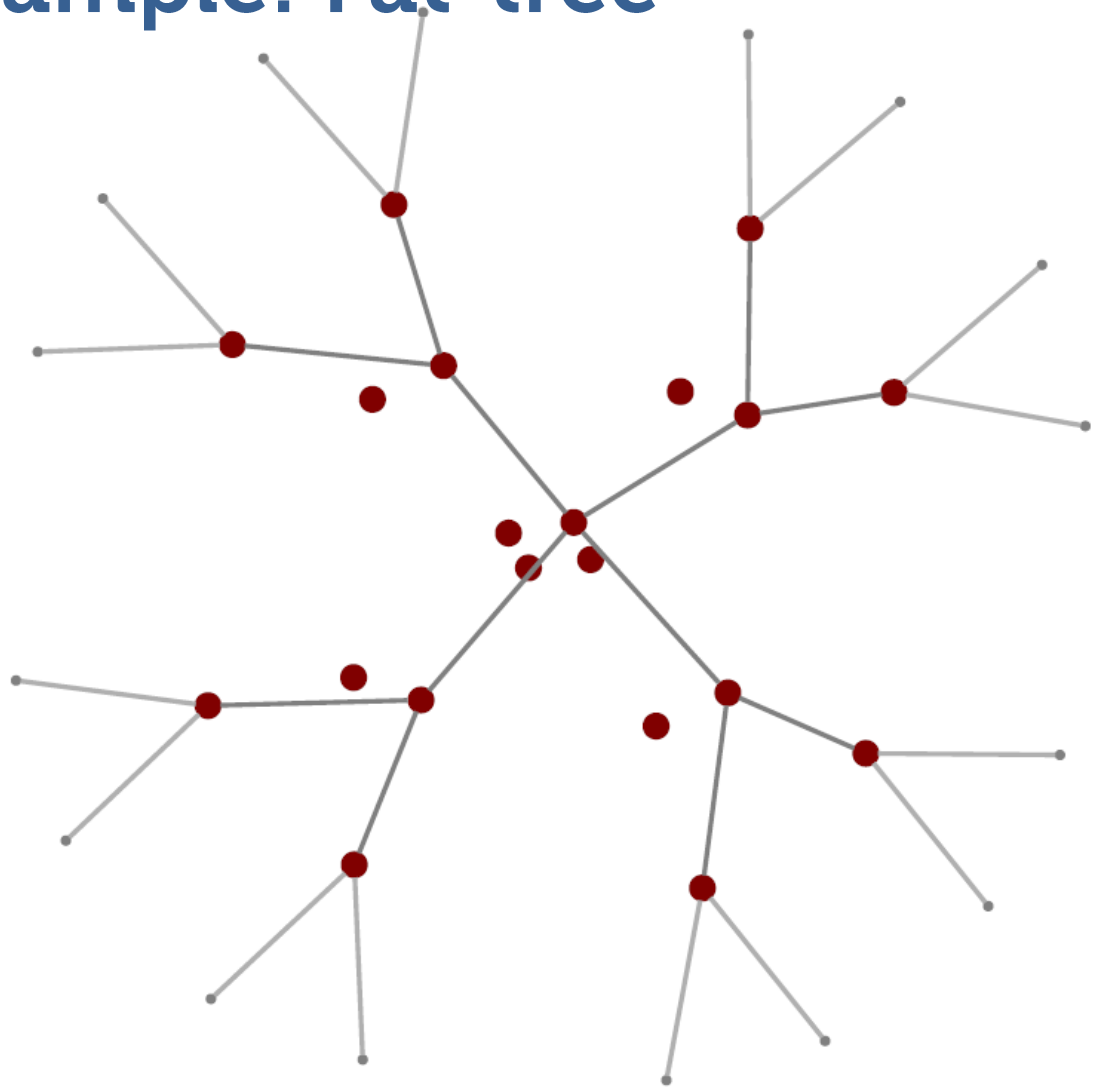
Example: Fat-tree

Fat-tree:
3 of 15
in < 6 hops



Example: Fat-tree

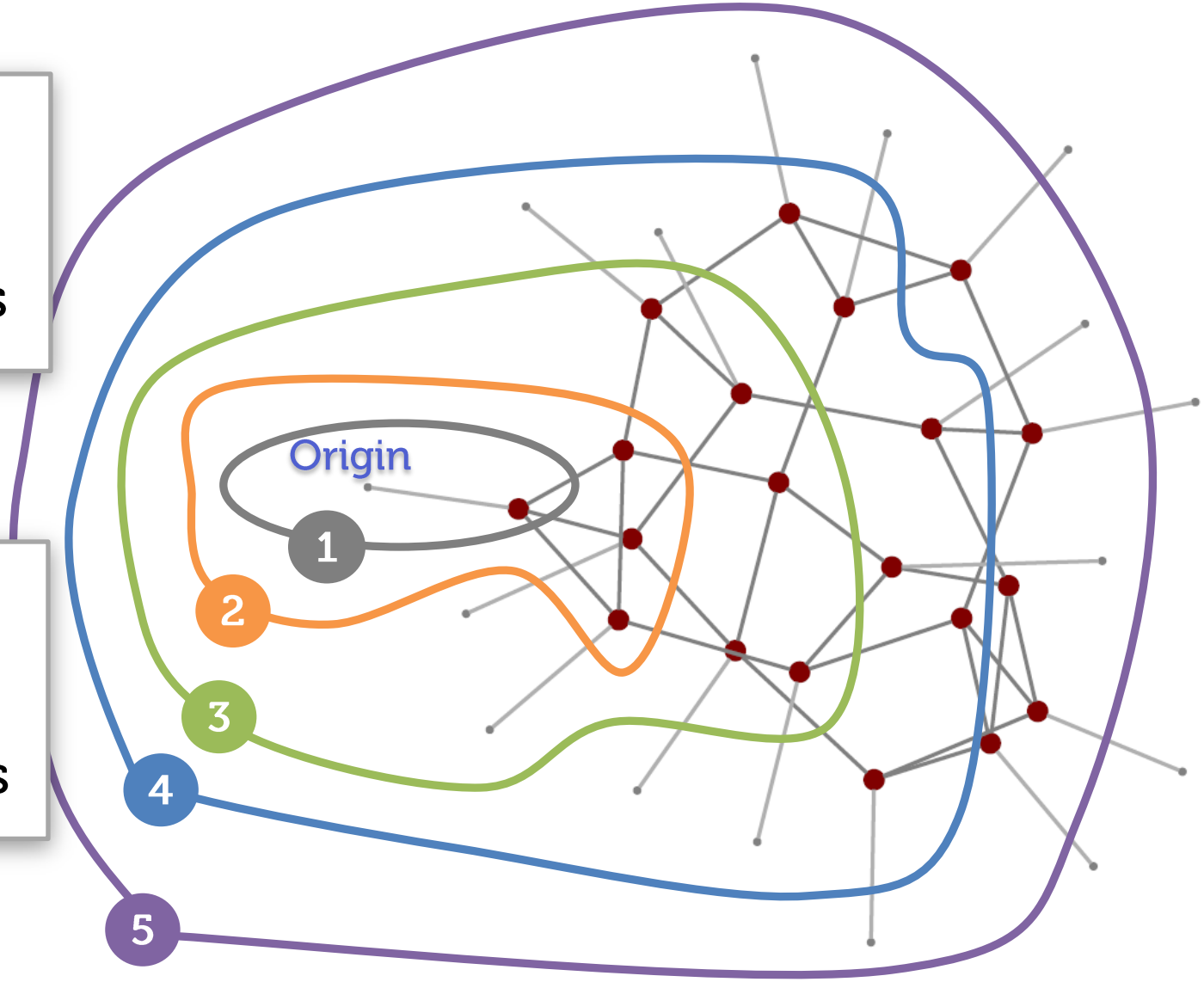
Fat-tree:
3 of 15
in < 6 hops



Jellyfish has smaller path length

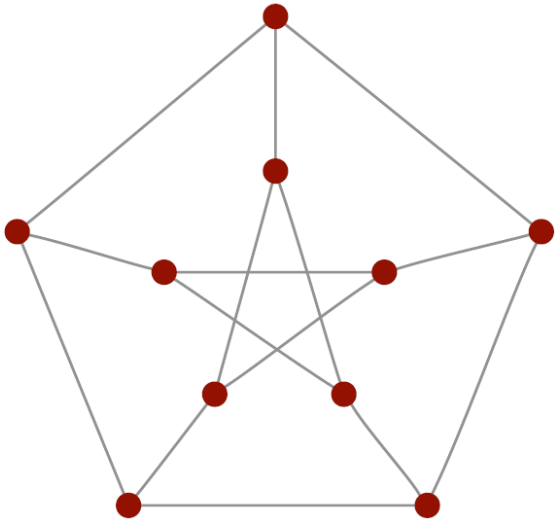
Fat-tree:
3 of 15
in < 6 hops

Jellyfish:
11 of 15
in < 6 hops



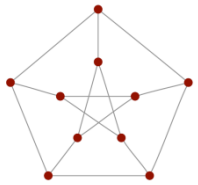
Can we do even better?

What's the maximum number of nodes in any graph with degree 3 and diameter 2?



Petersen graph

Degree-diameter bounded graph



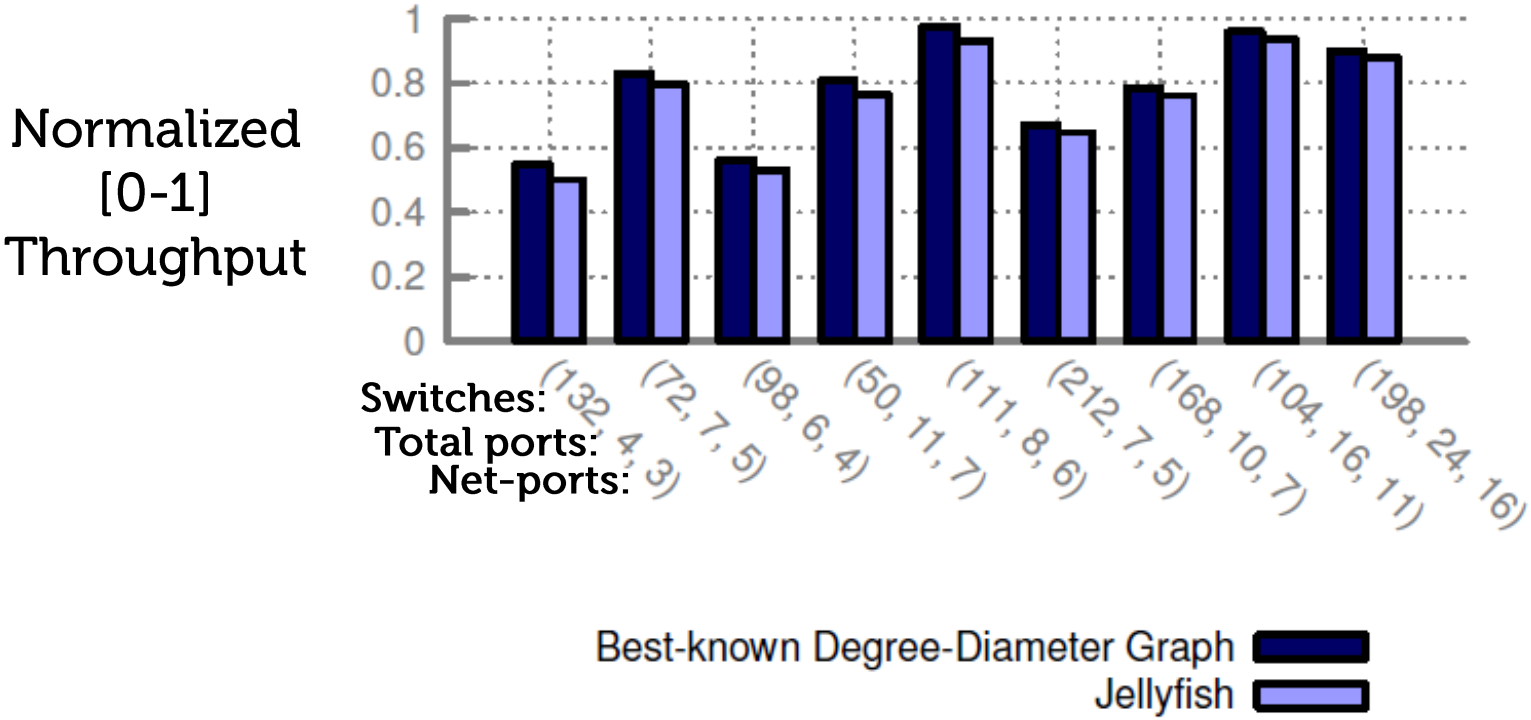
Largest known graph size with degree δ and diameter d, June 2010

Diameter

Degree	2	3	4	5	6	7	8	9	10
3	10	20	38	70	132	196	336	600	1 250
4	15	41	98	364	740	1 320	3 243	7 575	17 703
5	24	72	212	624	2 772	5 516	17 030	53 352	164 720
6	32	111	390	1 404	7 917	19 282	75 157	295 025	1 212 117
7	50	168	672	2 756	11 988	52 768	233 700	1 124 990	5 311 572
8	57	253	1 100	5 060	39 672	130 017	714 010	4 039 704	17 823 532
9	74	585	1 550	8 200	75 893	270 192	1 485 498	10 423 212	31 466 244
10	91	650	2 223	13 140	134 690	561 957	4 019 736	17 304 400	104 058 822
11	104	715	3 200	18 700	156 864	971 028	5 941 864	62 932 488	250 108 668
12	133	786	4 680	29 470	359 772	1 900 464	10 423 212	104 058 822	600 105 100
13	162	851	6 560	39 576	531 440	2 901 404	17 823 532	180 002 472	1 050 104 118
14	183	916	8 200	56 790	816 294	6 200 460	41 894 424	450 103 771	2 050 103 984
15	186	1 215	11 712	74 298	1 417 248	8 079 298	90 001 236	900 207 542	4 149 702 144
16	198	1 600	14 640	132 496	1 771 560	14 882 658	104 518 518	1 400 103 920	7 394 669 856

[Delorme & Comellas: http://www-mat.upc.es/grup_de_grafs/table_g.html/]

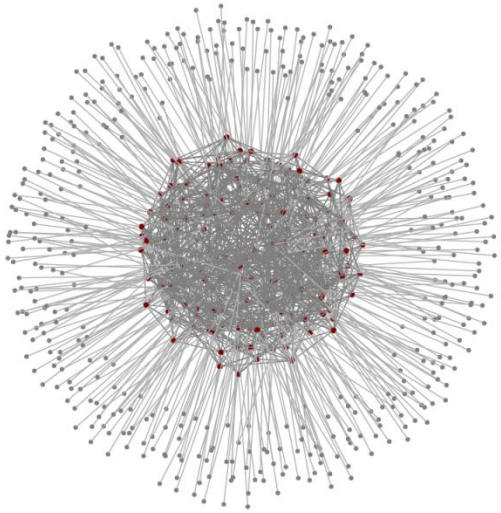
Degree-diameter vs. Jellyfish



D-D graphs do have high throughput

Jellyfish within 9%!

What we know so far



flexible, expandable

high throughput

OK, but ...

Routing and congestion control

Cabling

Intuition

If we fully utilize all available capacity …

$$\text{Number of flows at 1 Gbps rate} = \frac{\text{total capacity}}{\text{used capacity per flow}}$$

How to effectively utilize capacity without structure?

Routing and congestion control

Routing

✗ ~~ECMP~~

✔ *k* shortest paths
- MPLS TE
- OpenFlow
- SPAIN

[Mudigonda, Yalagandula, Al-Fares, Mogul; NSDI'10]

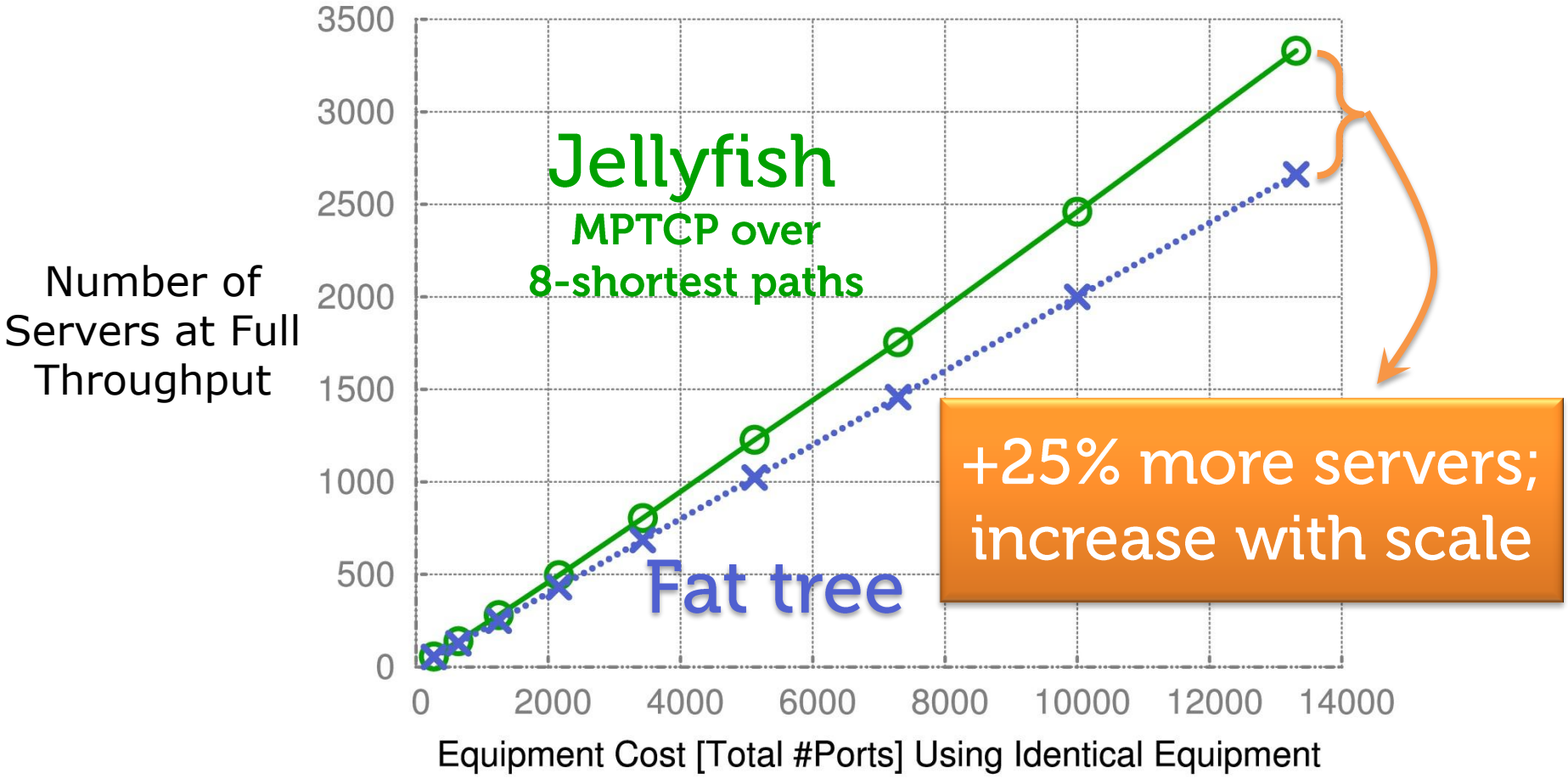
Congestion Control

TCP

Multipath TCP

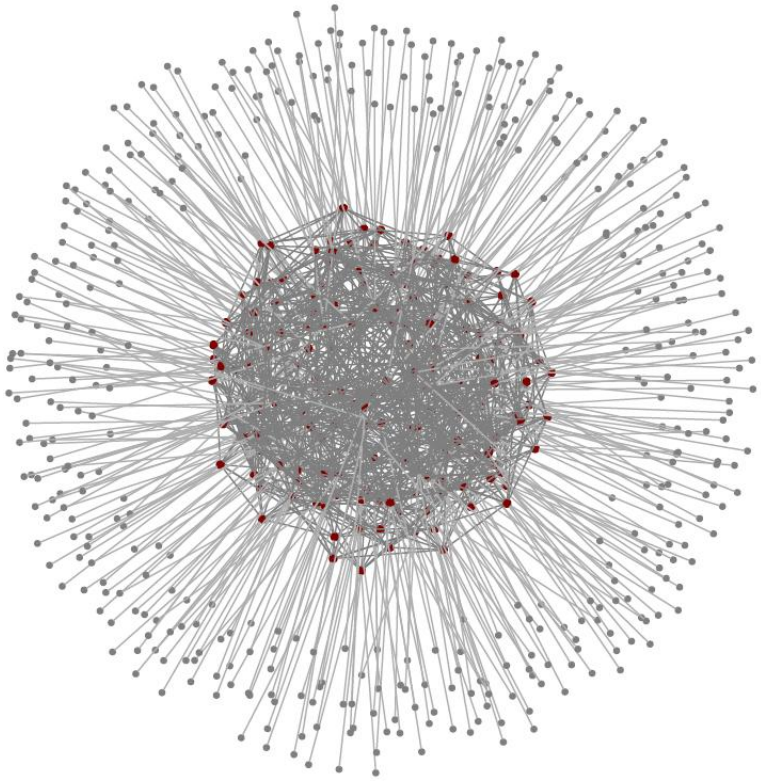
[Wischik, Raiciu, Greenhalgh, Handley; NSDI'11]
and SIGCOMM'11 and NSDI'12

Throughput: Jellyfish vs. Fat tree



Packet level simulation; random permutation traffic

Cabling



[Photo: Javier Lastras / Wikimedia]

~20% fewer switch-to-switch cables

For same # servers as fat tree

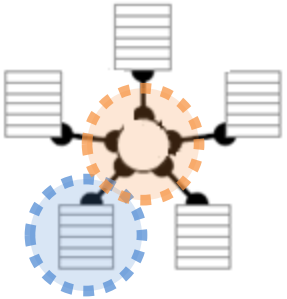
Optimization

Place switches centrally

Aggregate bundles

switches

server rack



cluster

Long optical cable: cost += ~$200

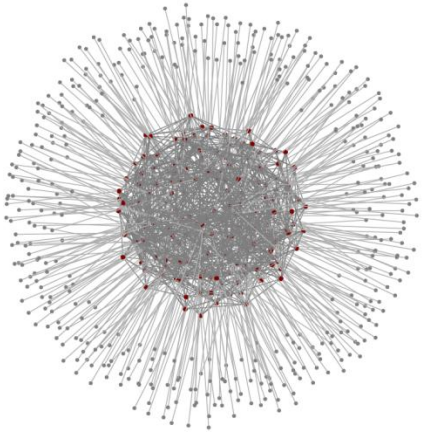
Idea: localize a fraction of connections

Result: Jellyfish retains almost all its gains after matching the same # short and long cables as the fat-tree

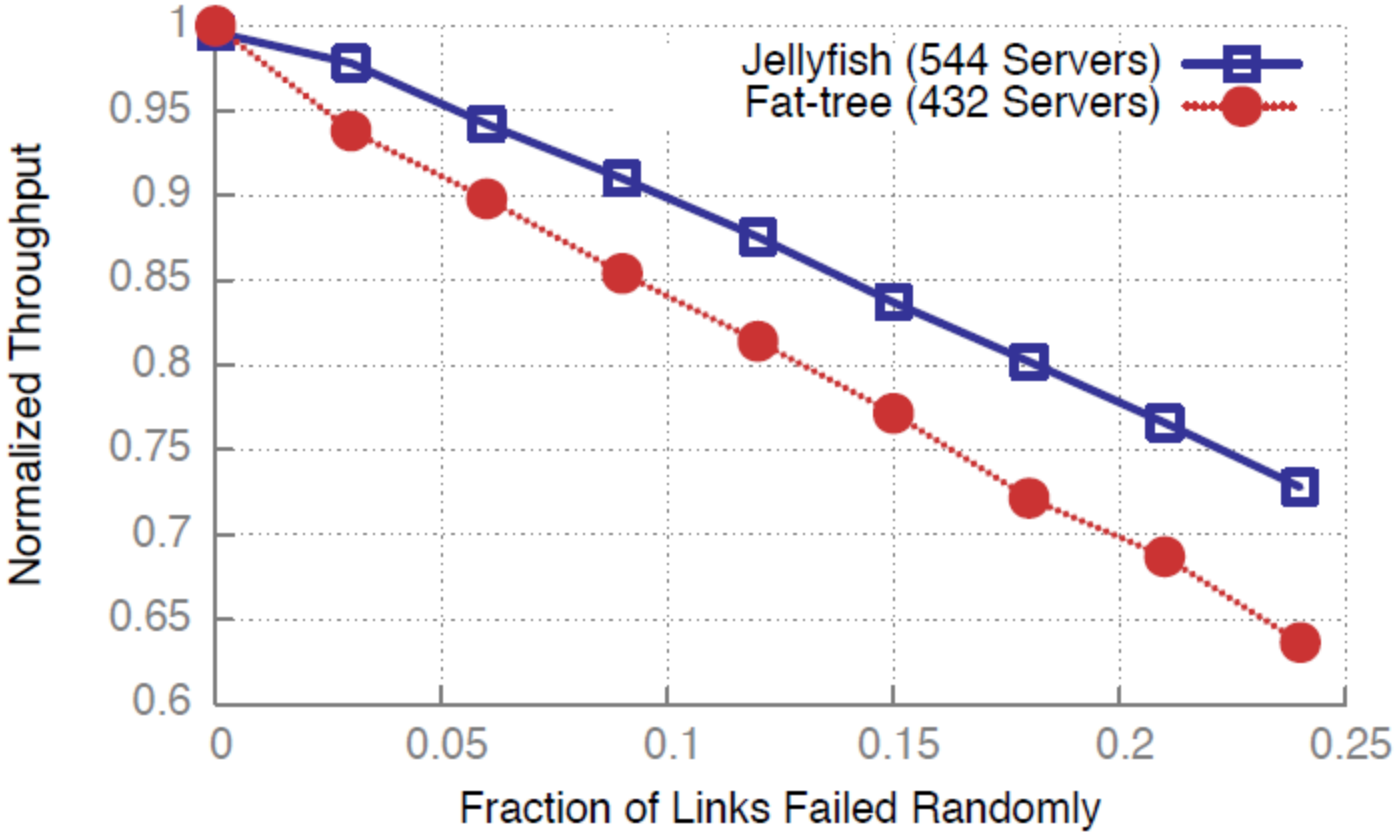
Conclusion

High throughput

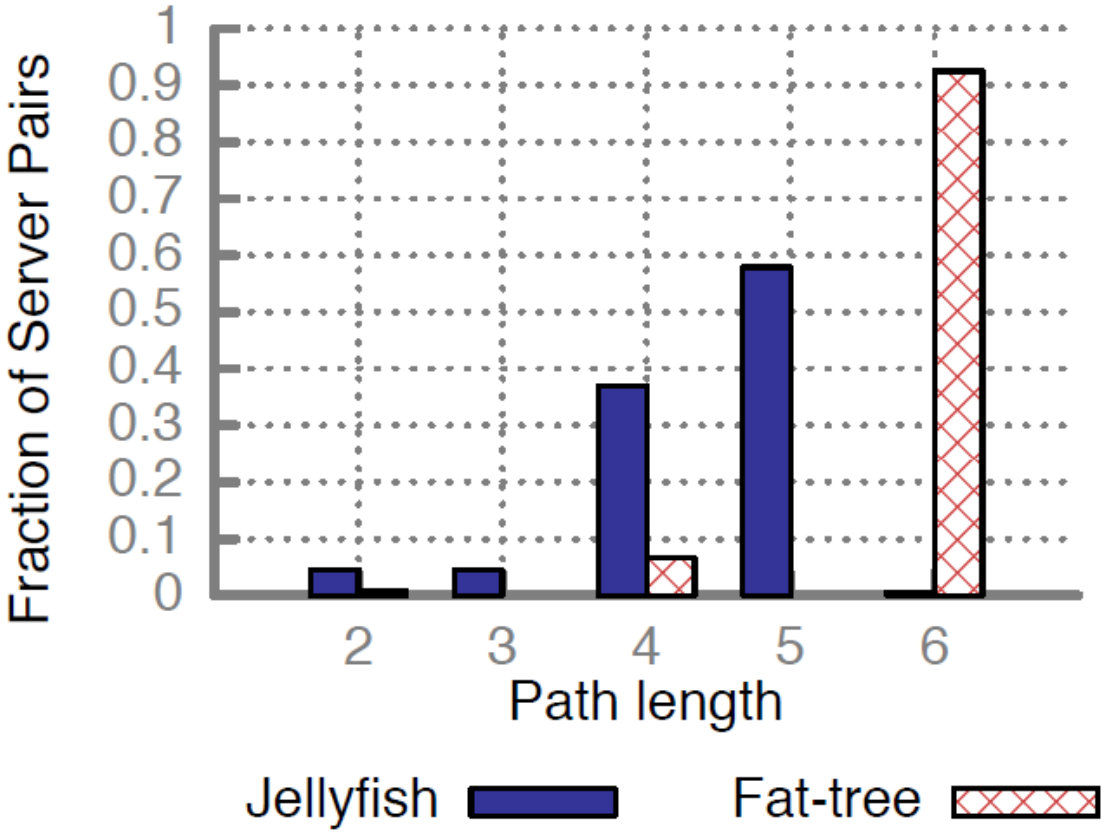
Expandability



Backup: Throughput under link failures

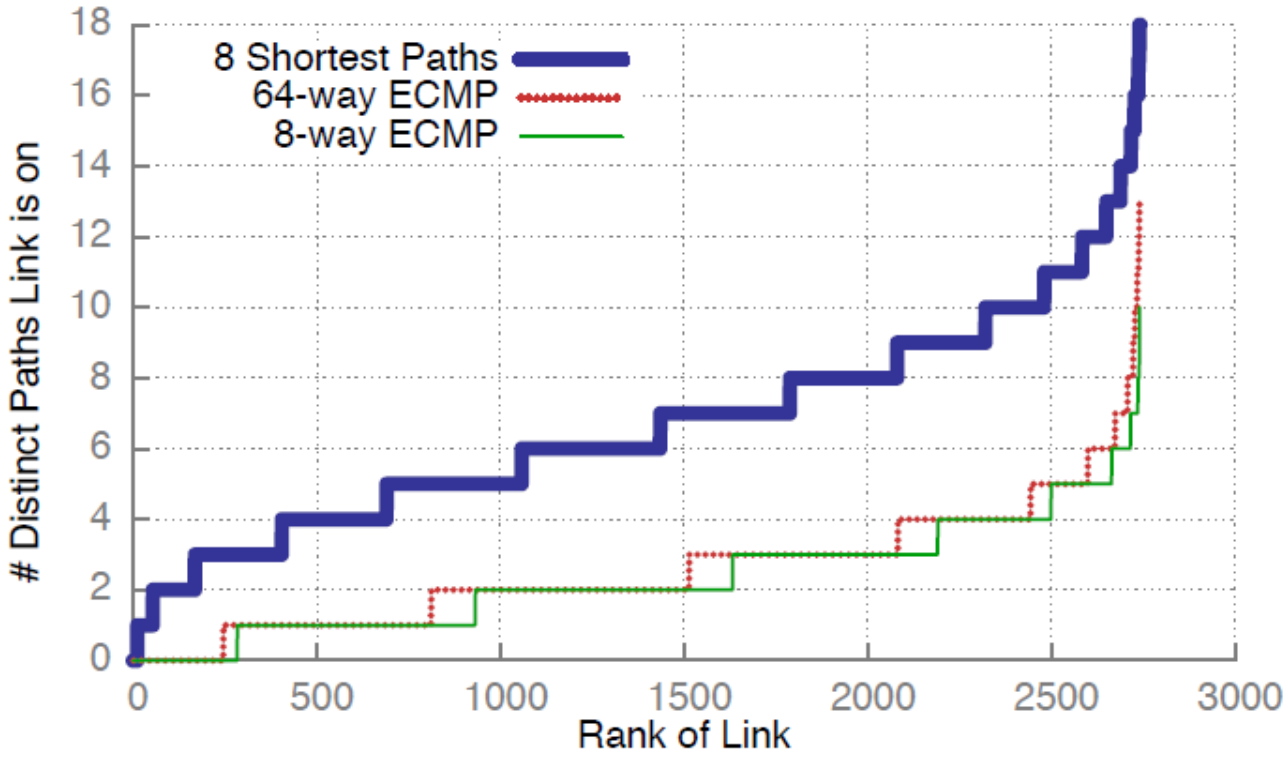


Backup: Jellyfish has short paths

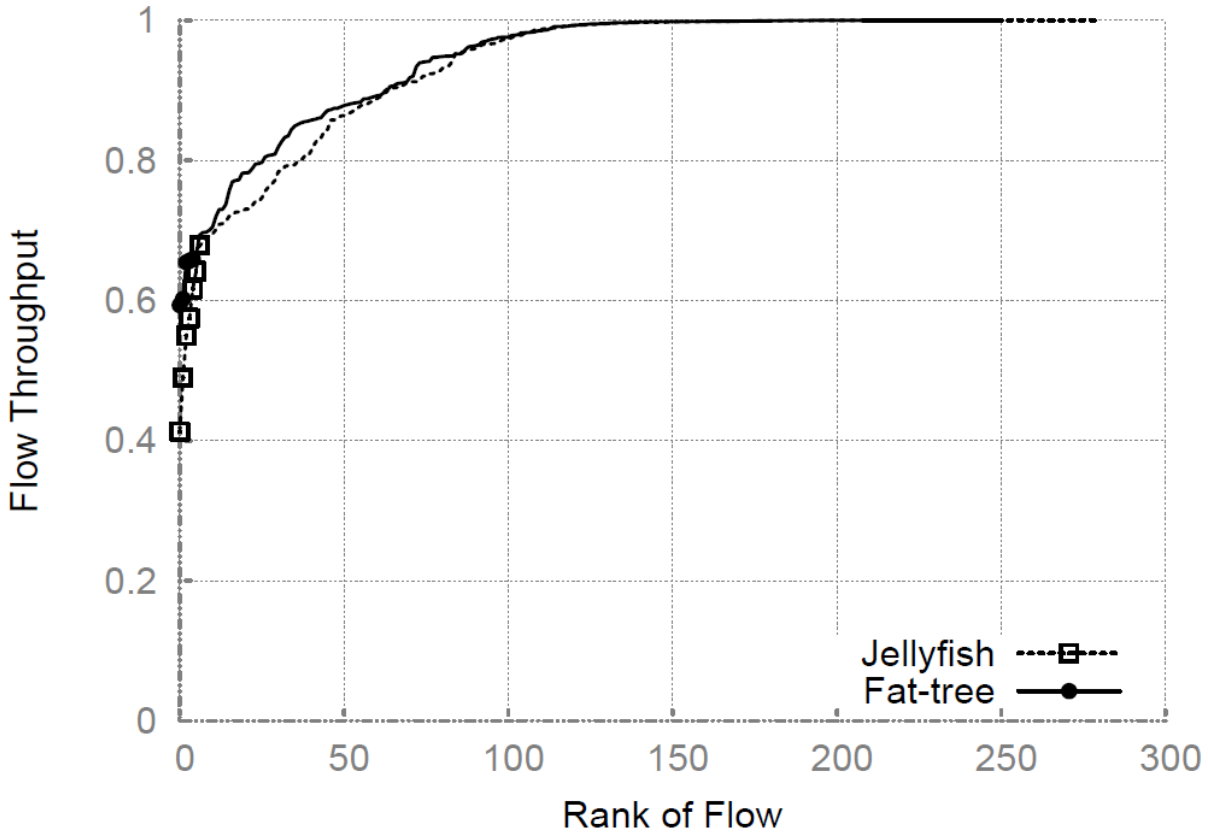


[686 servers, equal-equipment comparison]

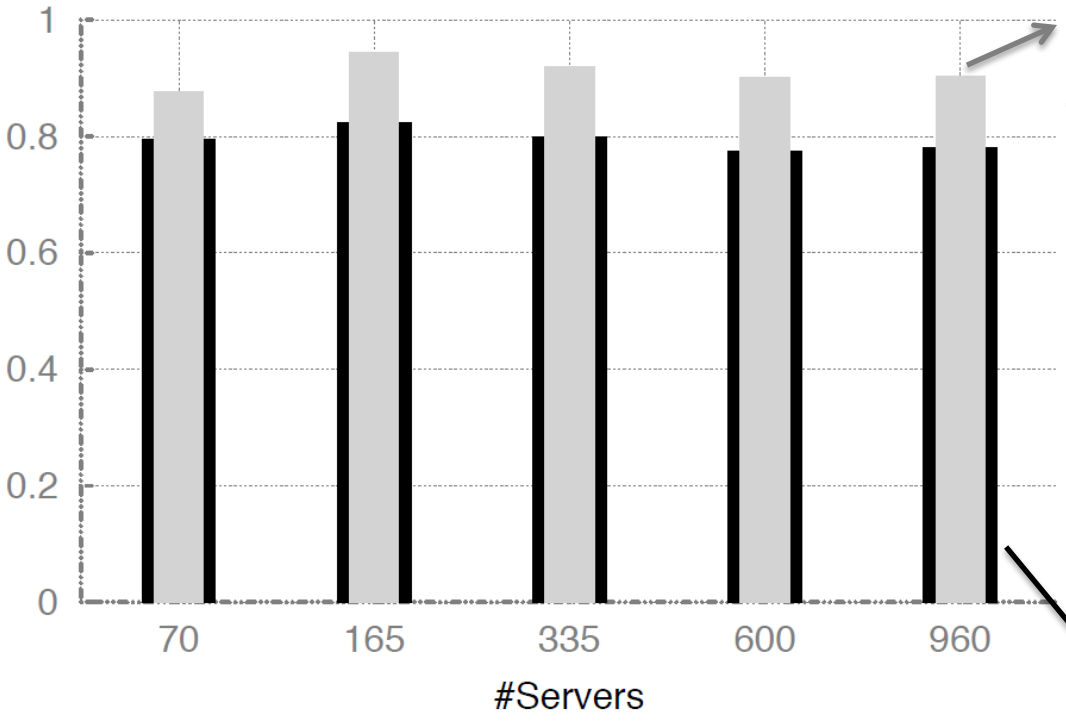
Backup: ECMP does not use enough Jellyfish's path diversity



Backup: Jellyfish provides good fairness



Backup: Simple protocol works



Optimal routing + Optimal congestion control (CPLEX solver)

MPTCP over 8-shortest paths (packet-level simulation)

86-90% of optimal!