Interconnection Topologies

Acknowledgement

Some of the images are taken from publicly available slides of CS5413 (Fall 2013) course page at Cornell University and CS4700/5700 course page at Northeastern University.

Before we start

- Topology defines the interconnection between nodes
- With each topology, there is also the very important issue of how do you assign addresses to the servers (what addresses to which servers) and how do you route between them
  - Can always assign addresses arbitrarily, and let a standard routing protocol like RIP/OSPF to form the routing table, and use that table for forwarding
  - May not be efficient for structured topologies in which specific ways of address assignment can simplify routing or allow finding better paths
  - Will not consider routing much here, though will give a simple example to illustrate what I mean

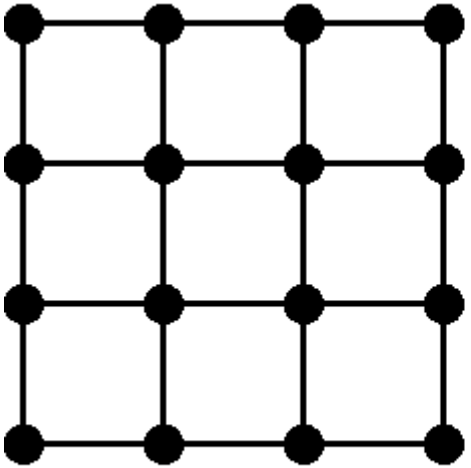
Some Terminologies

- Diameter of a network: Maximum value of a shortest path between two nodes (maximum over all such pairs of nodes)
- Bisection width: minimum number of links that have to be taken out to partition the network into two approximately equal sized networks
- Bisection Bandwidth: Minimum total bandwidth of links that have to be taken out to partition the network into two approximately equal halves (Bisections width x link bandwidth if all link bandwidths are the same)

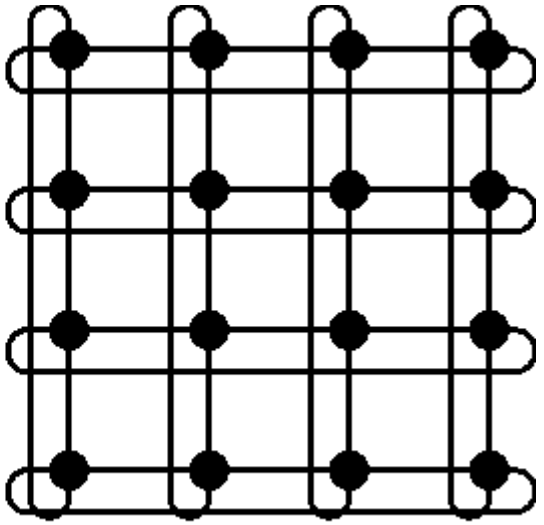
Some Properties of a Good Topology

- Low diameter
  - Also, low average number of hops between any two nodes
- High bisection width and bandwidth
- Low total number of links
  - Implies lower cabling complexity
- Low number of connections per node
  - Implies less number of ports per node
- Easy scalability: should be able to incrementally add nodes as the network grows

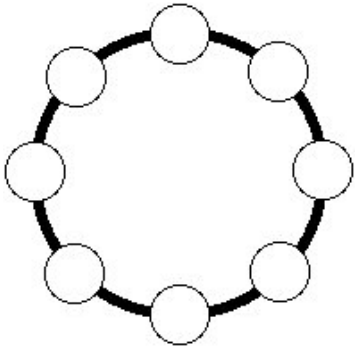
Linear



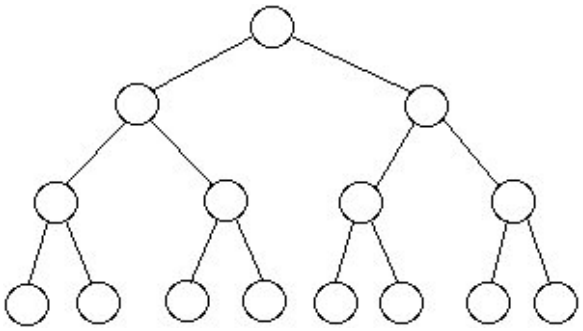
(a) 2D-Mesh



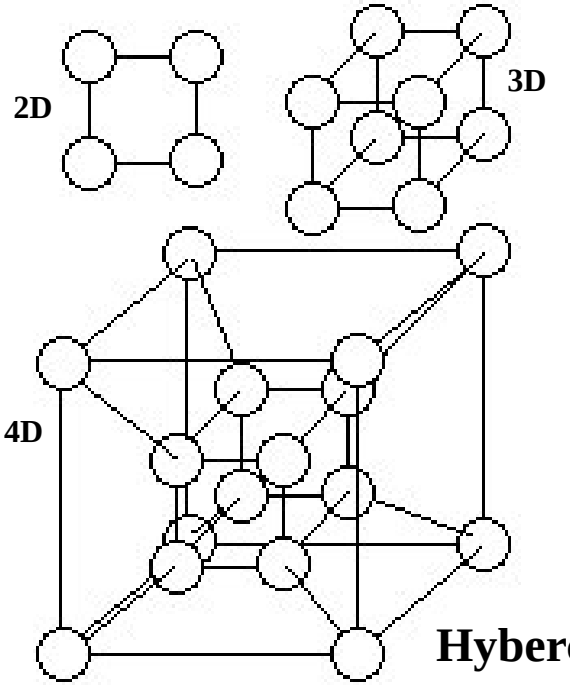
(b) 2D-Torus



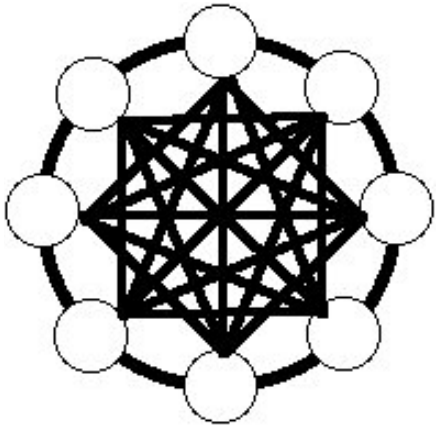
Ring



Binary Tree



Hybercube

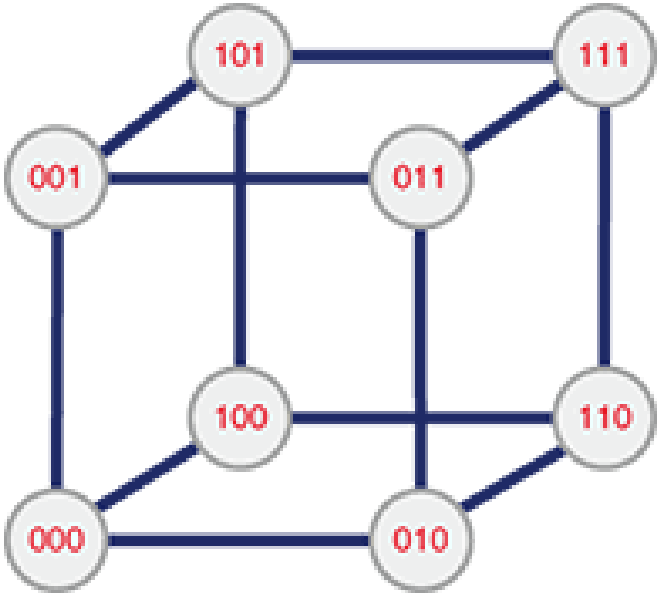


Fully Connected

	Diameter	Bisection Width	No. of Links	Ports per Node
Linear	N-1	1	N-1	2
Ring	N/2	2	N	2
2-d Mesh	$2(\sqrt{N}-1)$	$\sqrt{N}$	$2\sqrt{N}(\sqrt{N}-1)$	4
2-d Torus	$\sqrt{N}-1$	$2\sqrt{N}$	2N	4
Hypercube	$\log_2 N$	N/2	$N(\log_2 N)/2$	$\log_2 N$
Binary Tree	$2\log_2 N$	1	N – 1	3
Fully Connected	1	N	N^2	N – 1

Example of Routing: Hypercube

- Consider a 3-d hypercube with 8 nodes
- How should you assign the ids 0 to 7 to these 8 nodes?
- Option 1: Assign arbitrarily
  - Need to run a background routing protocol to find routes
  - Never done for hypercubes
- Option 2: Assign ids so that neighboring nodes differ only in 1 bit in binary representation of the ids
  - No background routing protocol needed
  - To route, just think of the 3 bits as the 3 dimensions, and forward the packet in one of the dimensions in which the source and destination differs at each step to make that dimension the same as that in destination id



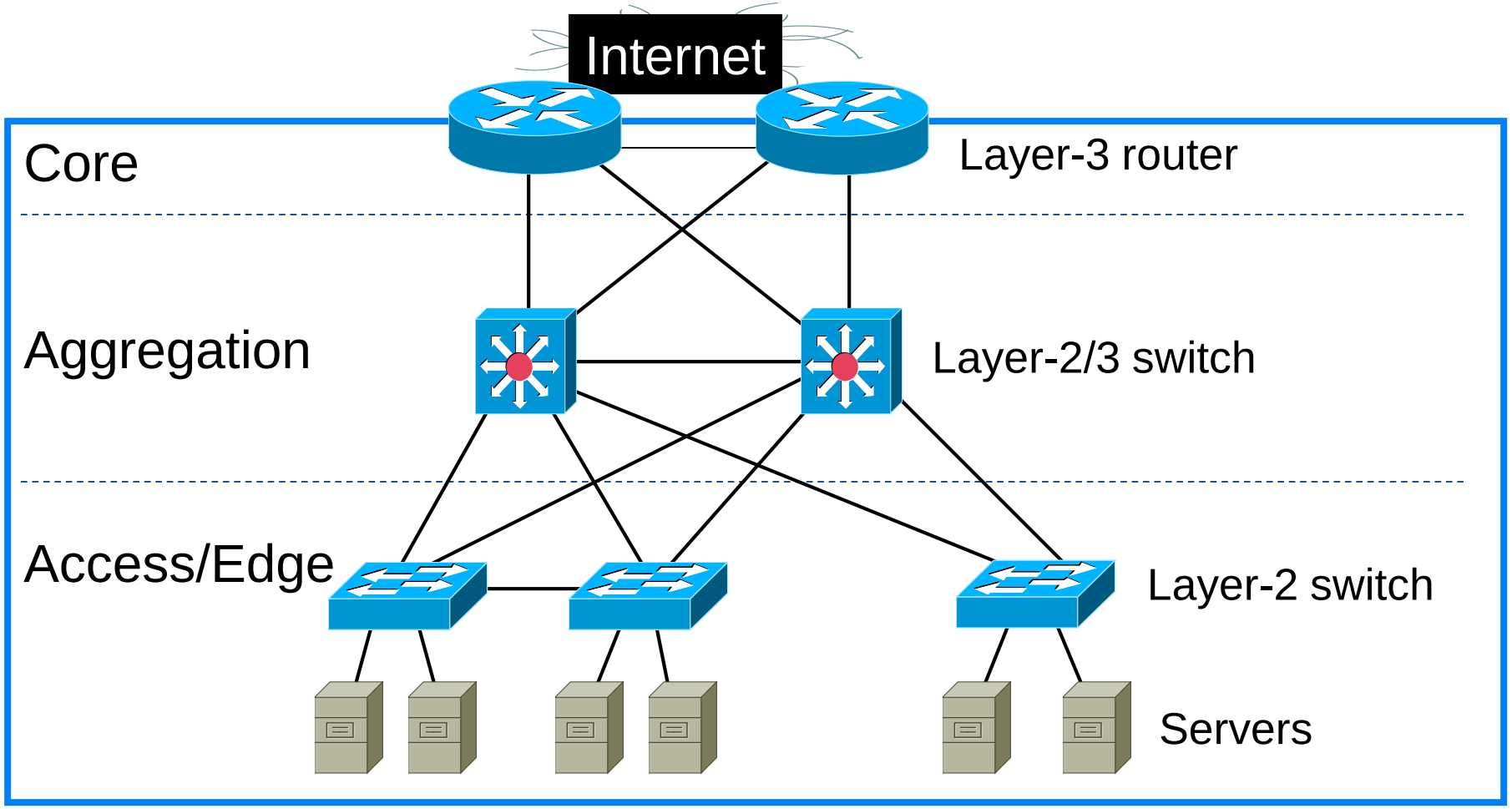
- Example: To route from 001 to 100
  - 001 sends to 101 (make 1st dimension same as destination)
  - 101 sends to 100 (make 3rd dimension same as destination)
- Could have also taken the route 000 (make 3rd dimension same as destination), 100 (make 1st dimension same as destination)
- Always gives shortest path (obvious)
- Easy to extend to any dimension

Switched Networks

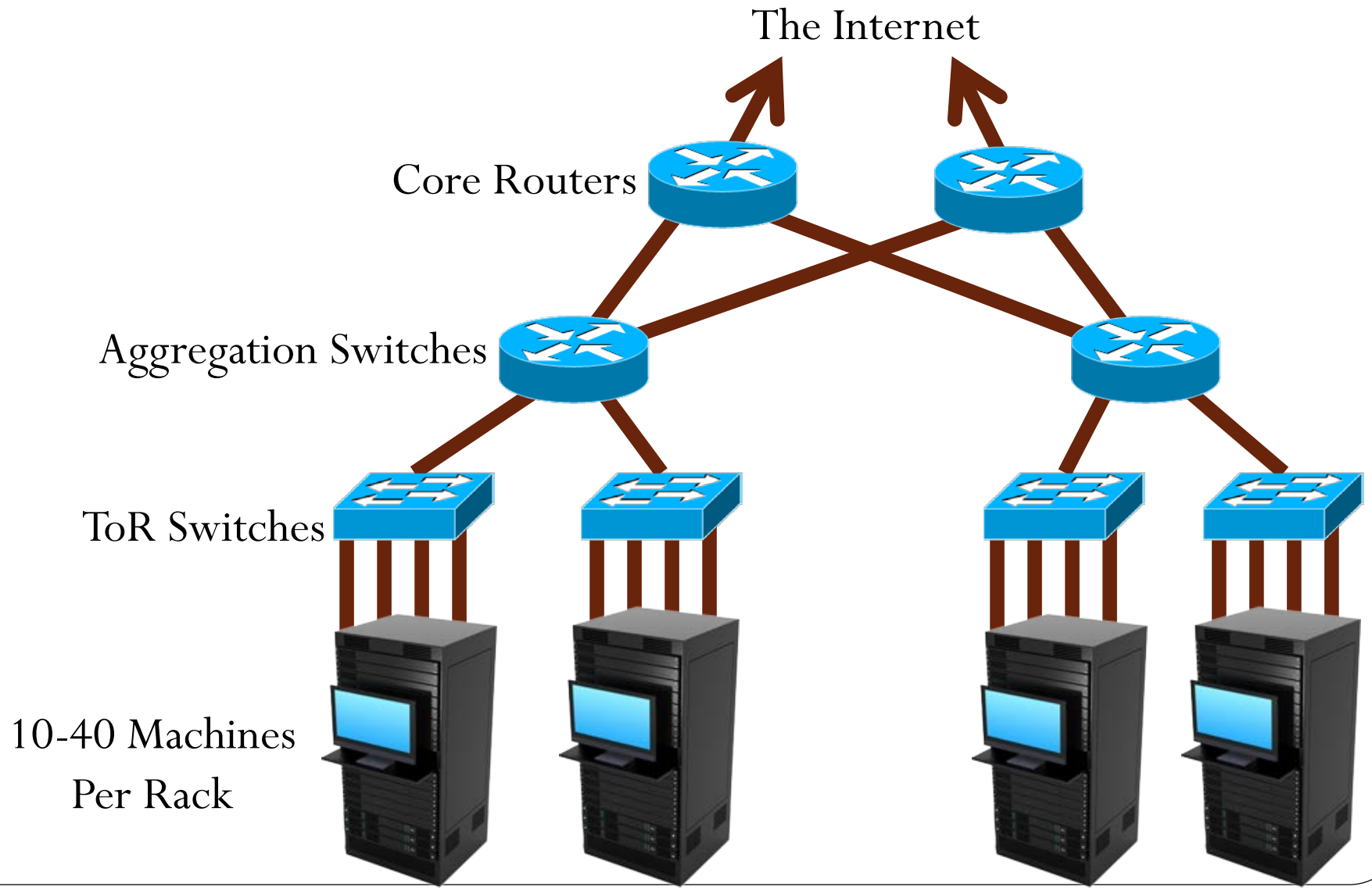
- What are nodes?
  - Can be servers or switches
- Direct networks
  - When the switching component is integrated with the server
- Modern networks
  - Switched networks, where servers are interconnected by switches

A Typical 3-Layer Switched Network

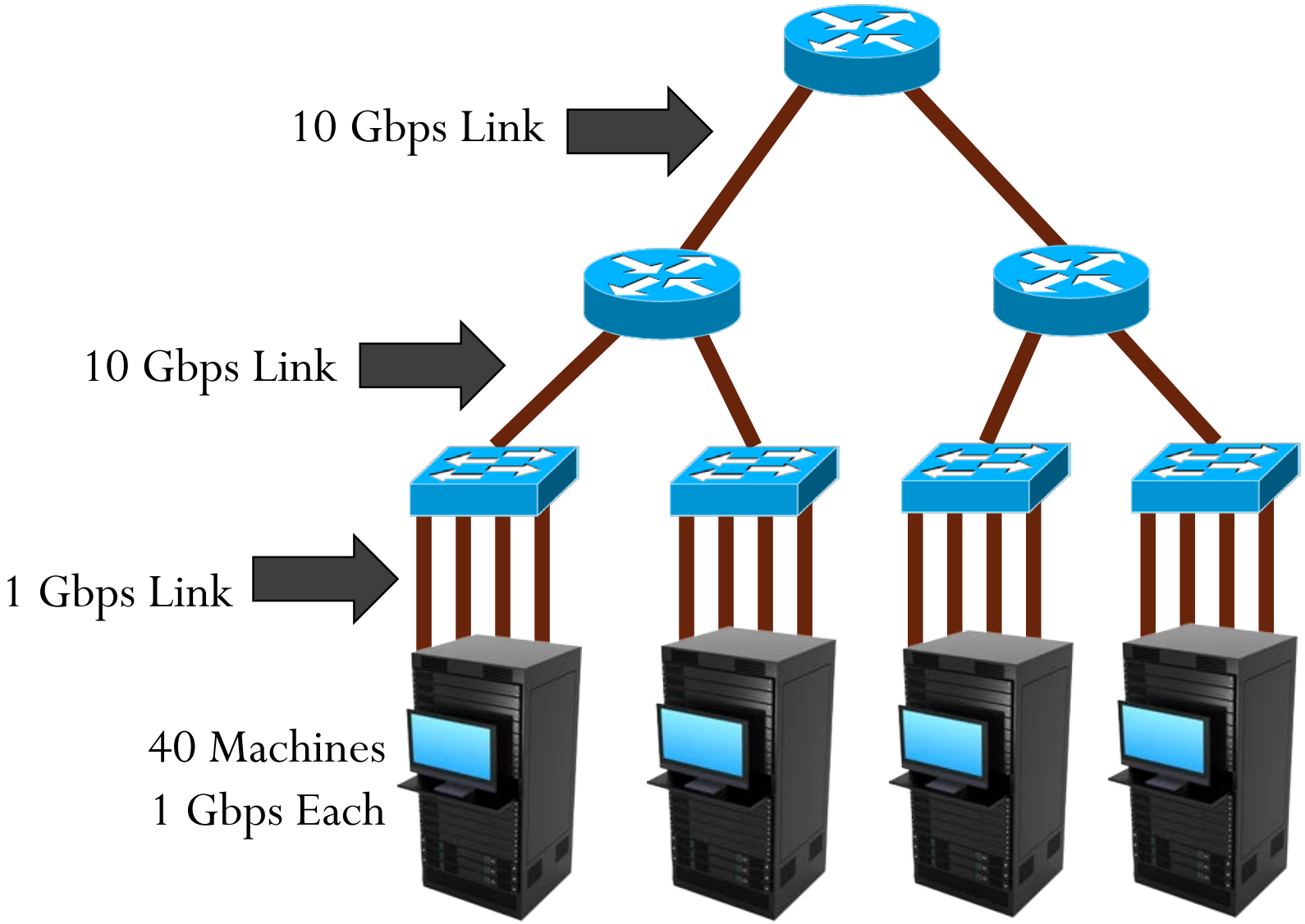
- With redundant links



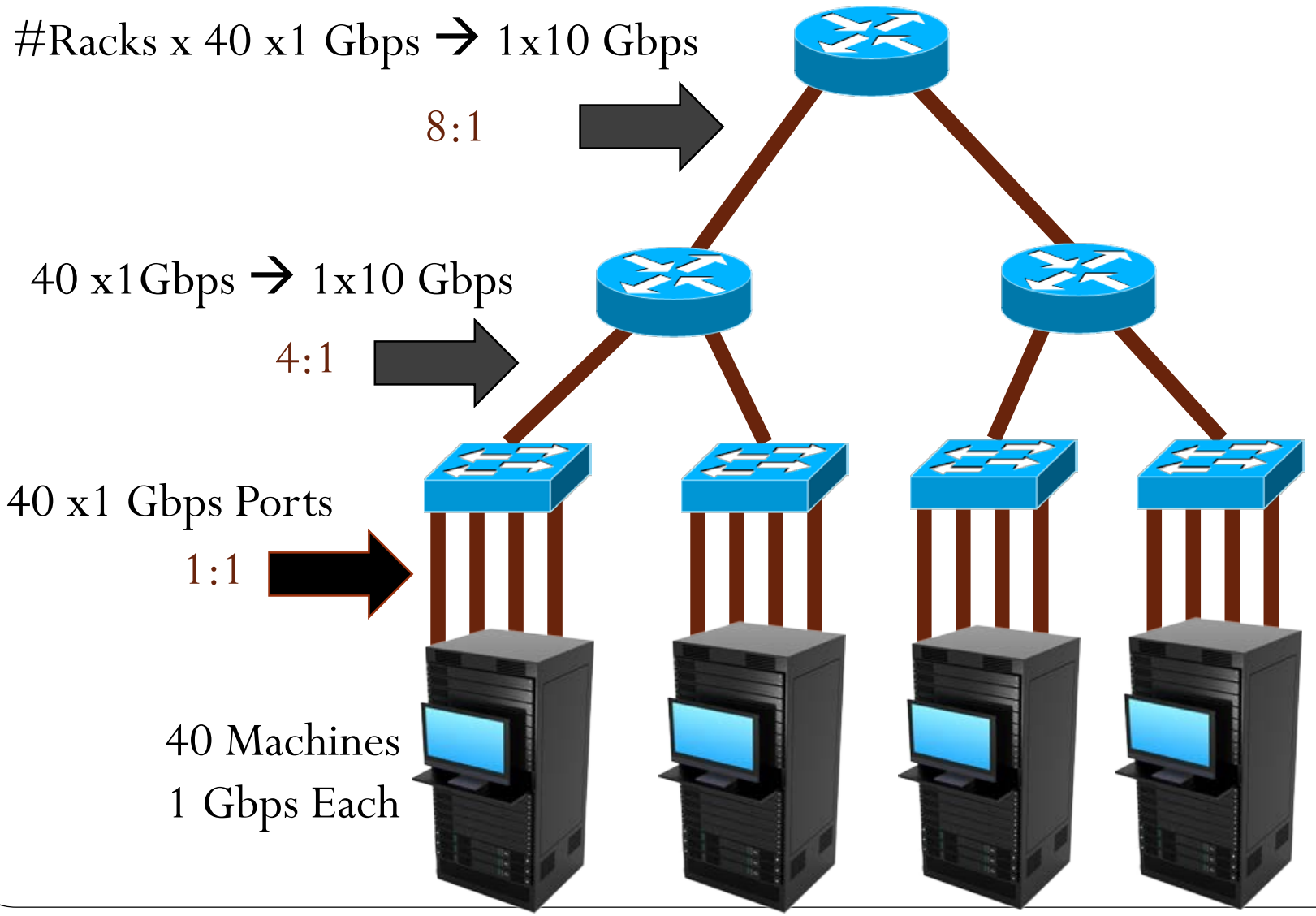
Typical 3-layer Rack Interconnection (with redundancy only at core)



Consider this Network



Oversubscription Problem



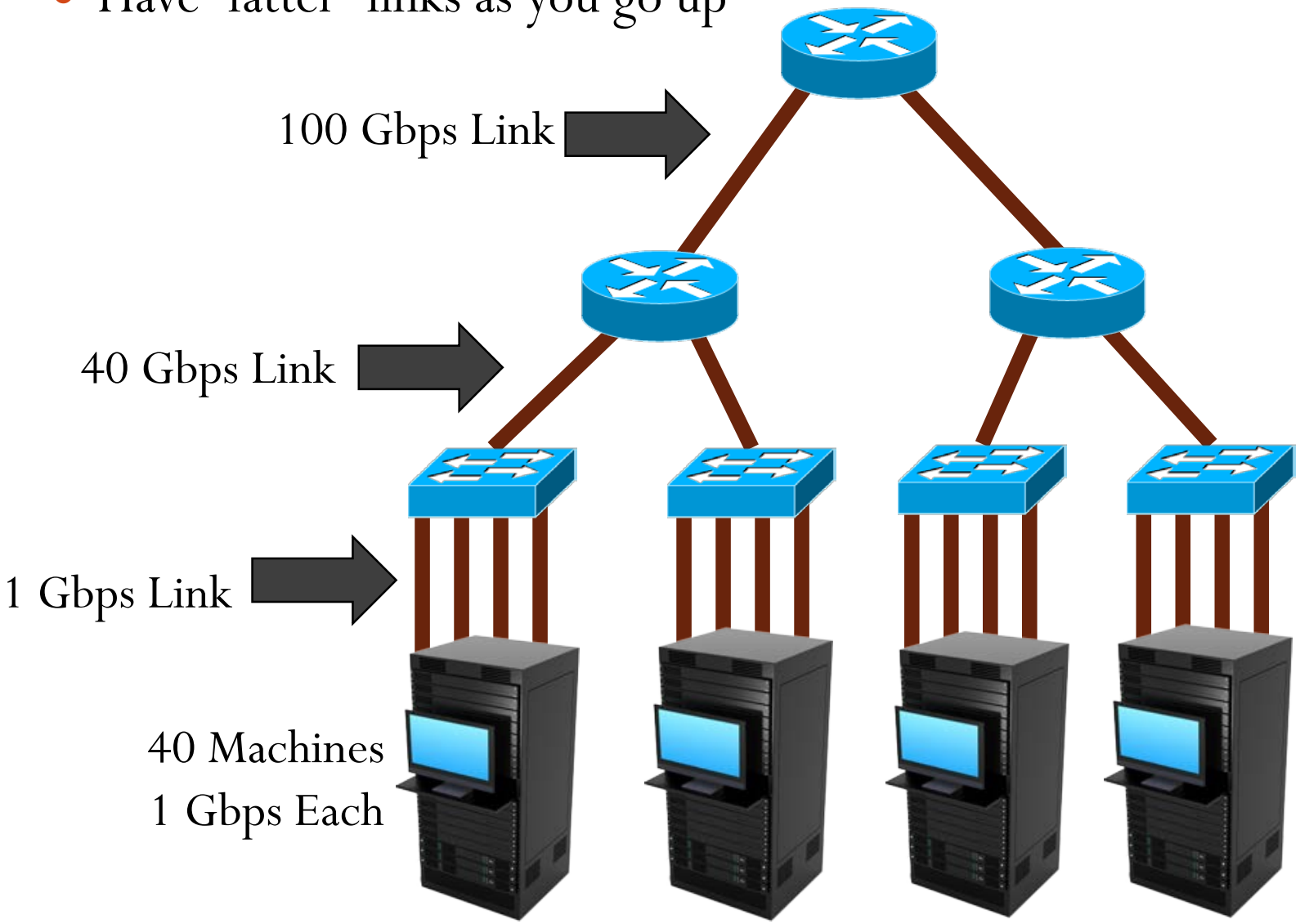
Is Oversubscription Always Bad?

- Bandwidth gets scarce as you move up the tree
  - So all-to-all communication is very bad
- Problem is becoming more acute as servers with 10Gbps links are becoming more common
  - Access switch ports are 10 Gbps also instead of 1 Gbps then
- But that's just the available capacity. Applications may not use the whole capacity
  - Common for many small enterprise data centers
  - But for large data centers like cloud data centers, a server may run 10's of VMs with high network loads

- Can you avoid moving up the tree for most communication?
  - Keep communications local (within racks, between nearby racks,..)
  - Whether you can do this or not depends on many other things like application communication pattern, fault tolerance etc.
    - Lot of research on mapping communication to topology to take advantage of topology
  - But difficult to ensure always

How can we address oversubscription?

- Have "fatter" links as you go up



- Problems with single Fat links
  - Very costly
  - Switch costs do not scale linearly with capacity
    - Access switches with a single 40 Gbps uplink port are hard to find and very costly (much more than a switch with 4 x 10G uplinks)
    - In addition, for higher layer switches, cost also increases with number of ports (much more than access switches)
  - Also, the topology we considered has no redundancy
    - A core switch failure can bring down the entire data center
  - Having redundancy at some levels will add to the cost
  - In summary, does not scale to large systems
- So what can be done?
  - Have a logical "fat" link, with more than one physical "thinner" link
  - What's the catch?

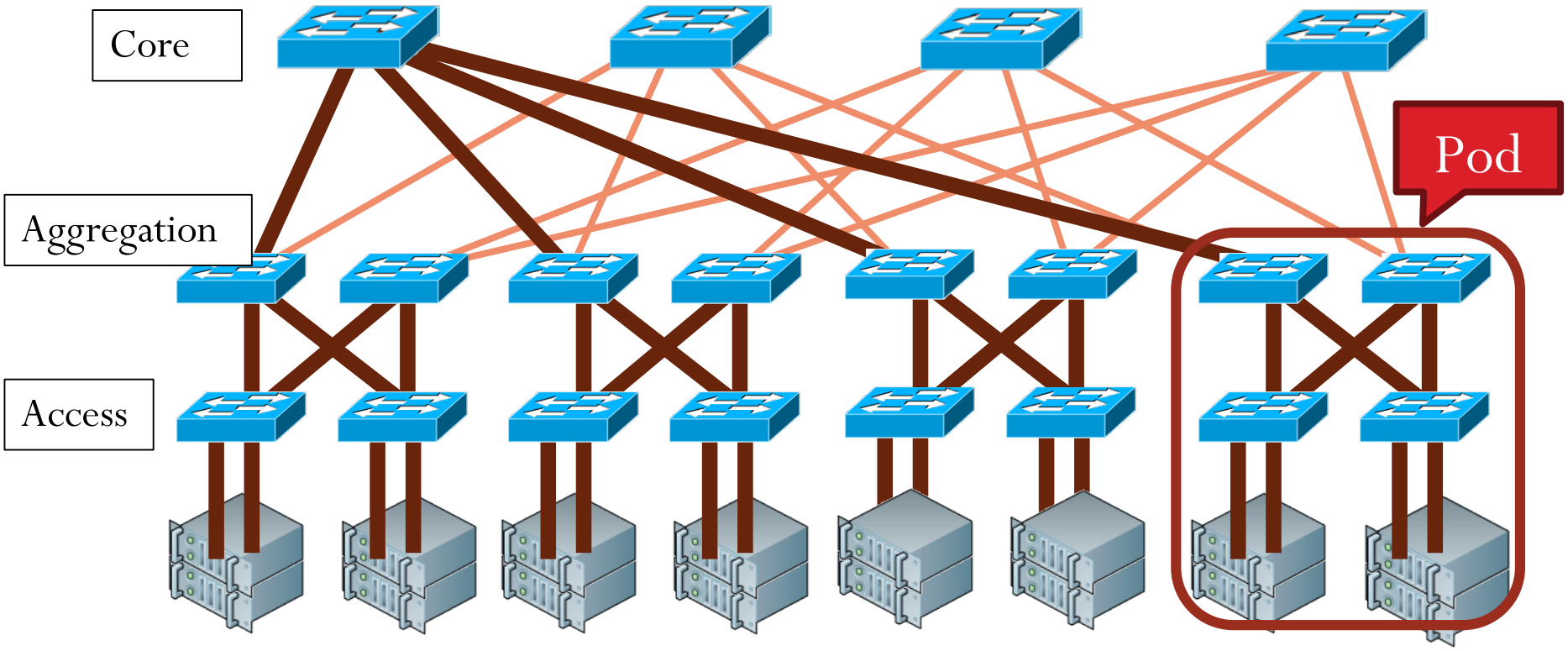
K-ary Fat Tree Topology

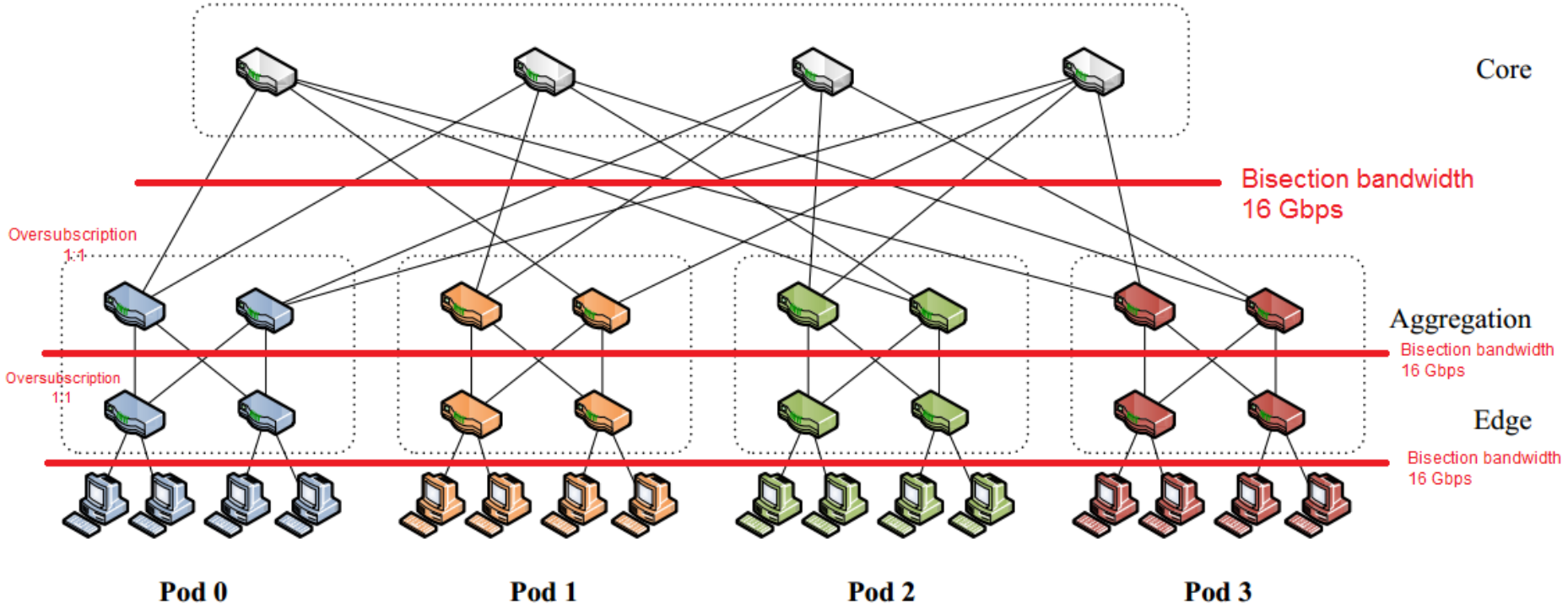
- K-port switches
- $K^3/4$ servers
- K pods, each with K switches and connecting K servers
  - K/2 access switch, K/2 aggregation switch
- $(K/2)^2$ core switches
- Interconnection within a pod
  - Connect K/2 servers to each access switch
  - Connect the other K/2 ports of each K/2 access switch to K/2 aggregation switches in a redundant fashion
- Pod to core interconnection
  - Connect the other K/2 ports of each aggregation switch from each pod to K/2 different core switches
- No direct inter-pod connections

Fat Tree Topology

In this example K=4

- 4-port switches
- $K^3/4 = 16$ servers
- $(K/2)^2 = 4$ core switches
- 4 pods, each with 4 switches




Core
Bisection bandwidth
16 Gbps
Oversubscription
1:1
Aggregation
Bisection bandwidth
16 Gbps
Oversubscription
1:1
Edge
Bisection bandwidth
16 Gbps
Pod 0
Pod 1
Pod 2
Pod 3

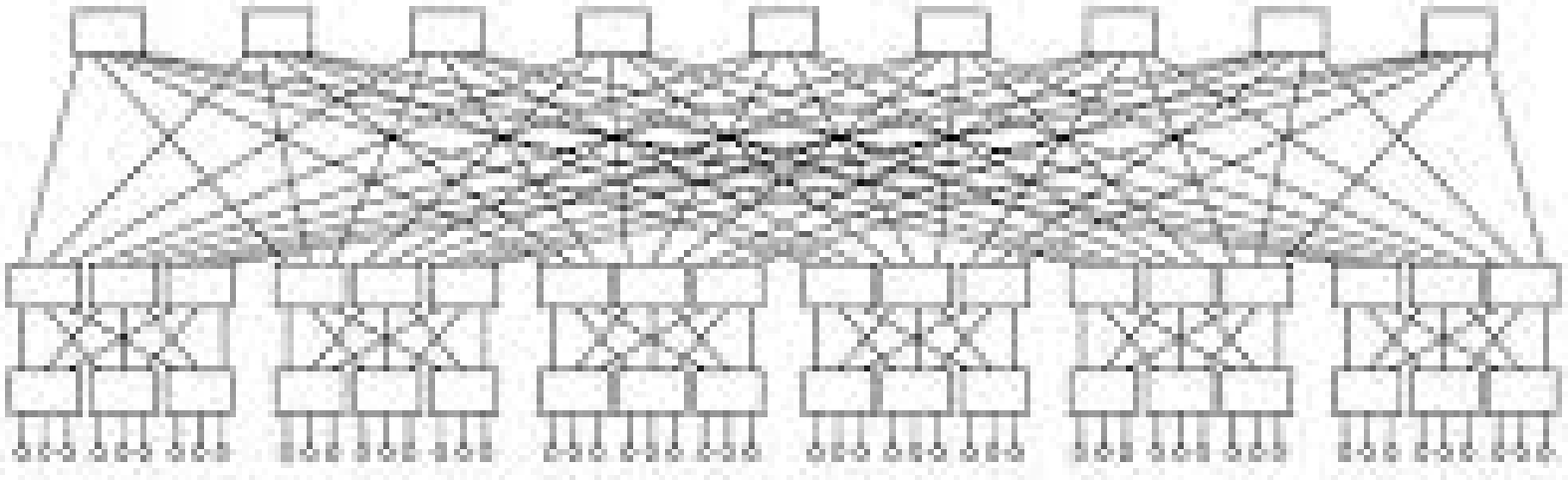
Pros of Fat Tree

- Each layer of Fat tree has the same aggregated bandwidth
- Can be built using cheaper switches with the same port capacity
  - Each port supports same speed as end host
- All devices can transmit at line speed if packets are distributed uniformly along available paths
- Scalability: K-port switch supports $K^3/4$ servers
  - With 48 port switches = 27,648
- Lots of path for failover redundancy

Cons of Fat Tree

- Large number of switches → $3K^2/2$ switches
  - With 48 port switches = 3456
- Huge number of wires → $(K^3+2K^2)/4$
  - With 48 port switches = 28800
- Requires specialized addressing and routing schemes to ensure no bottleneck in practice
- Even though theoretically all switches can be the same, in practice, aggregation to core links will be of higher capacity as full uniform utilization of all links is difficult to achieve
  - Will give different access, aggregation, and core switches
  - But the bandwidth difference will be much smaller than what is needed in single-root 3-tier networks (ex. 1G/10G or 10G/40G at edge/aggregation)

A 6-ary Fat Tree



Topologies Used in DCs

- Some HPC DCs are built for a very specific class of applications. For such DCs, application communication patterns can be predicted and regular topologies like mesh, torus etc. have been used.
- Other than that, fat tree and its variants are very common in large DCs
- Small DCs often use a standard 3-tier network with or without partial/full redundancy
- Many other network topologies and associated schemes proposed for DCs, hot area of research
- Many large corporations have also used their own proprietary topologies, though often based on clos/fat-tree.