Emerging Networks: Data Center Networks

CS204: Advanced Computer Networks

Nov 20, 2023

Agenda

- Characteristics of a datacenter (DC)
  - Goals, workloads, hardware, …
- Specific goals of DC networking (or DCN)
- Fat-tree based DCN architecture

What’s a datacenter

- A data center is a physical location that stores computing machines and their related hardware equipment. It contains the computing infrastructure that IT systems require, such as servers, data storage drives, and network equipment [1].

https://aws.amazon.com/what-is/data-center/

Early Datacenters

1961, Information Processing Center at the National Bank of Arizona

Today's Datacenters

Amazon's Datacenter in Oregon (left) and Virginia (right)

Key differences between the two

- Scale
  - 1M servers/site [Microsoft/Amazon/Google]
  - > $1B to build one site [Facebook]
  - >$20M/month/site operational costs [Microsoft ’09]
- Emerging applications
  - Large-scale computations (AI, big data)
  - Customer-facing, revenue generating services
- Service model
  - Cloud: Infrastructure as a Service (IaaS), SaaS, PaaS, …
  - Multi-tenancy

Implications of Scale

- Scalable designs
  - e.g., avoid flooding
- Low cost designs: e.g., use commodity technology
- High efficiency: e.g., >80% avg. utilization
  - Contrast: avg. utilization on Internet links often ~30%
- Tolerate very frequent failure
  - Large number of low-cost, error-prone components

Implications of Service model

- Typical service model: clouds / multi-tenancy
  - Performance guarantees
  - Isolation guarantees
  - Portability

- Key enabler: Network virtualization
  - Software-Defined Radio (SDN) and Network Function Virtualization (NFV)

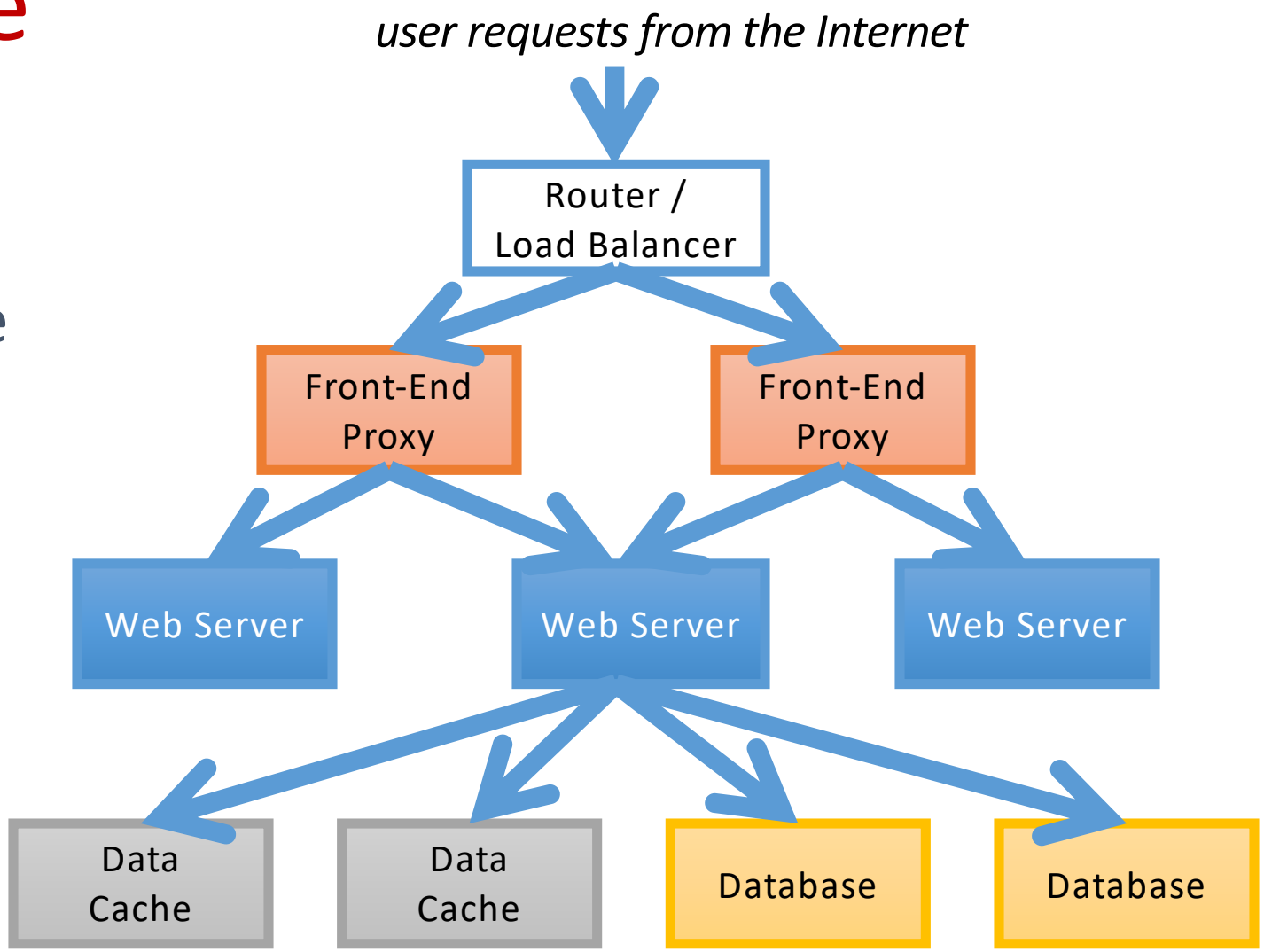
Applications

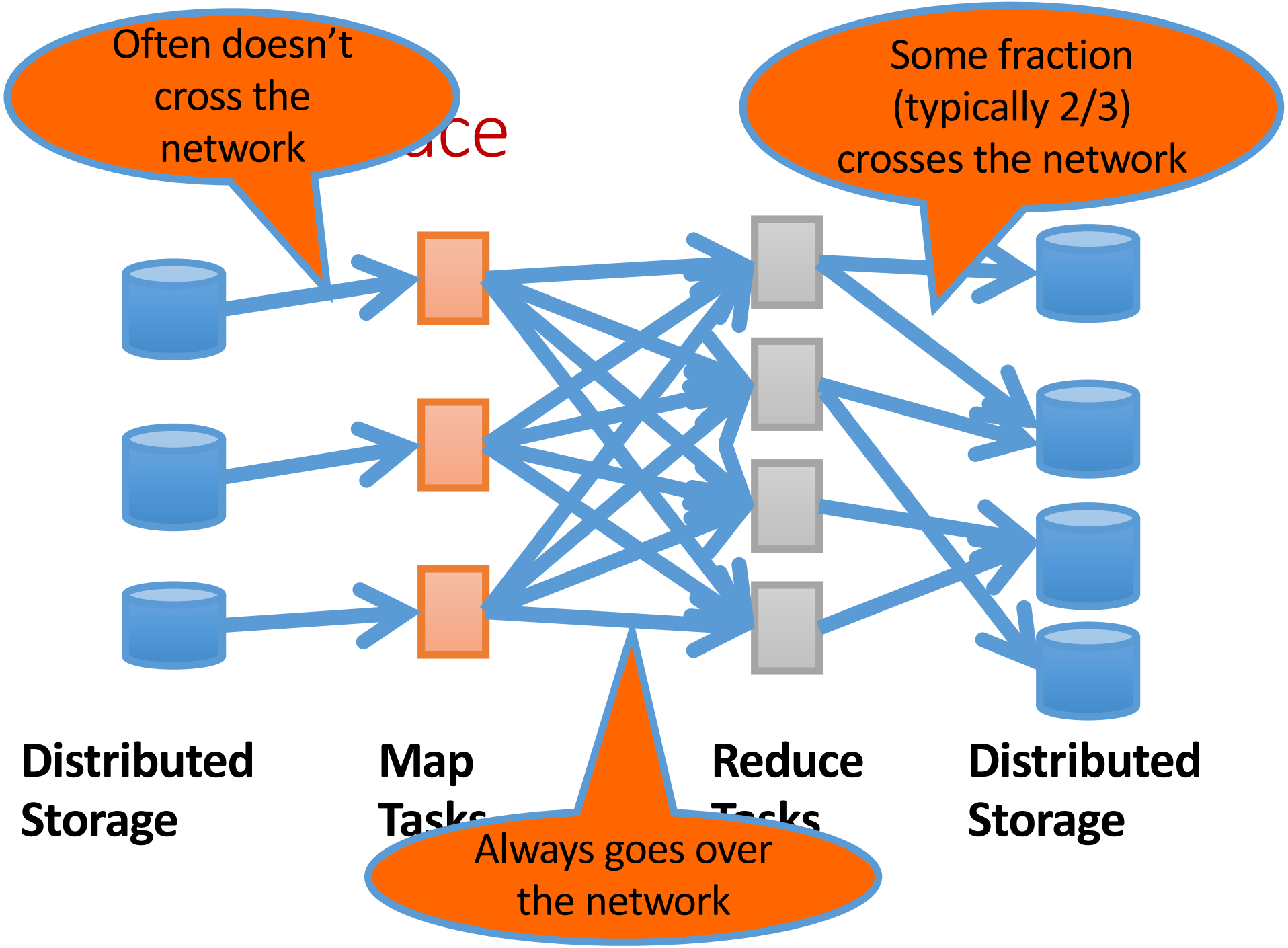
- Common theme: parallelism
  - Applications decomposed into tasks
  - Running in parallel on different machines
- Two common paradigms
  - Partition-Aggregate
  - Map Reduce

Partition-Aggregate

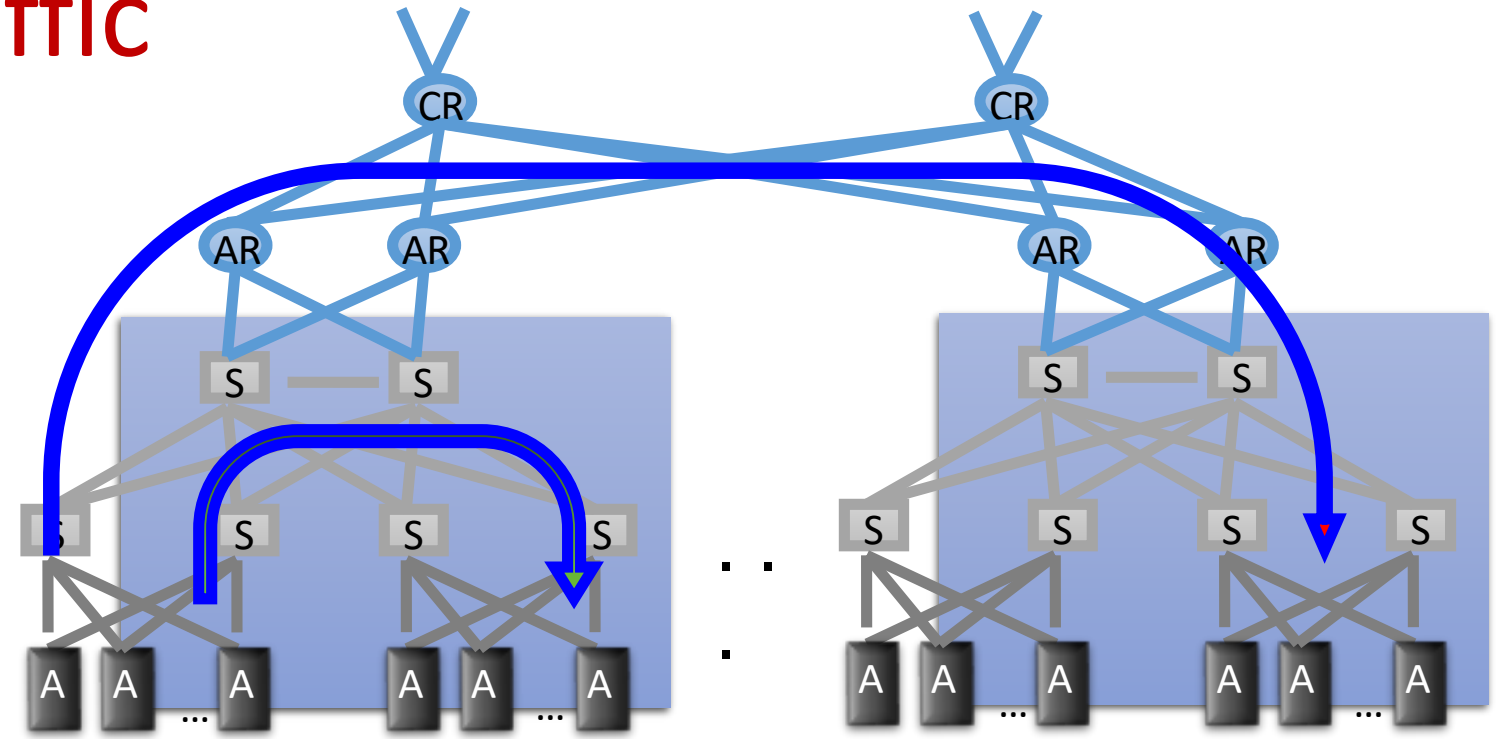
North-South Traffic

- Interactive / query-response exchange between external clients and datacenter
- Handled by front-end (web) servers, mid-tier application servers, and back-end databases

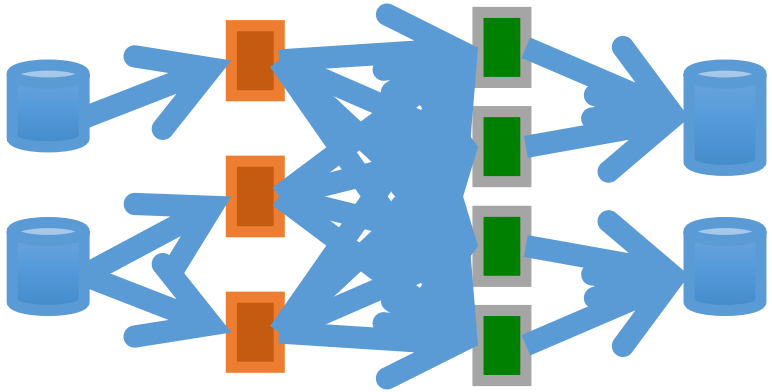



Often doesn't cross the network
Some fraction (typically 2/3) crosses the network
Always goes over the network
Distributed Storage
Map Tasks
Reduce Tasks
Distributed Storage

East-West Traffic



- Traffic between servers in the datacenter
- Communication *within* "big data" computations
- Traffic may shift on small timescales (< minutes)



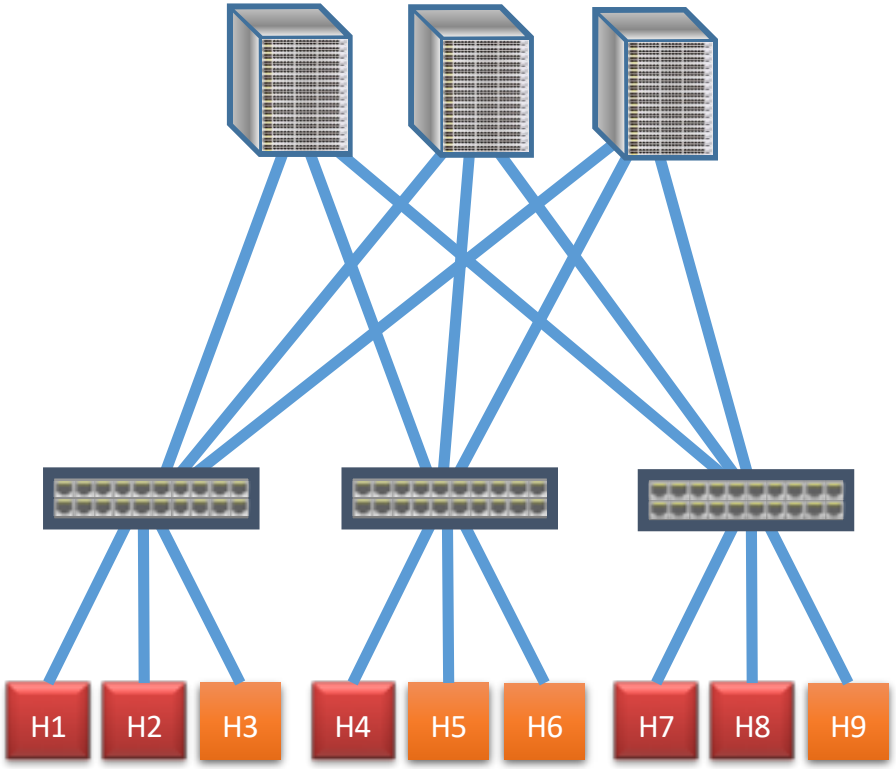
Implications of Scale

- Low latency
  - Very low RTTs within the DC (approaching 1µsec)
    - BW x delay: 10Gbps x 1µsec = 10000 bits = 2.5 packets
    - Consider TX 500B @ 10Gbps = 0.4µs per hop = 2µs if a packet traverses 5 hops and waits behind one packet at every hop
  - What does this mean for congestion control, buffering, switch design?

Implications of Scale

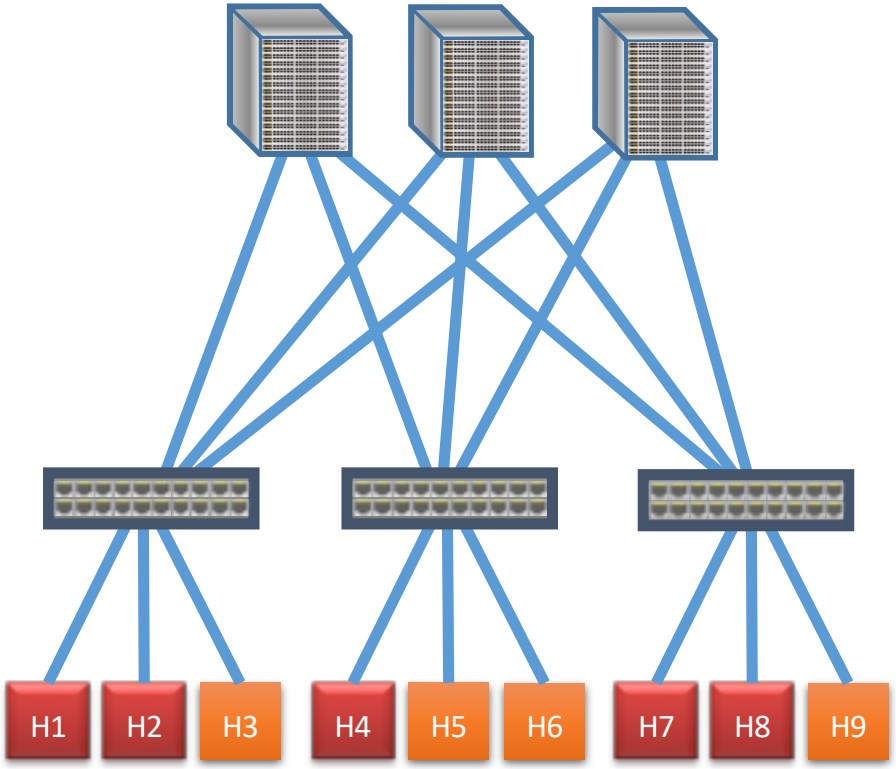
- High bandwidth communication
  - Ideally: Each server can talk to any other server at its full access link rate

DC Network: Just a Giant Switch!



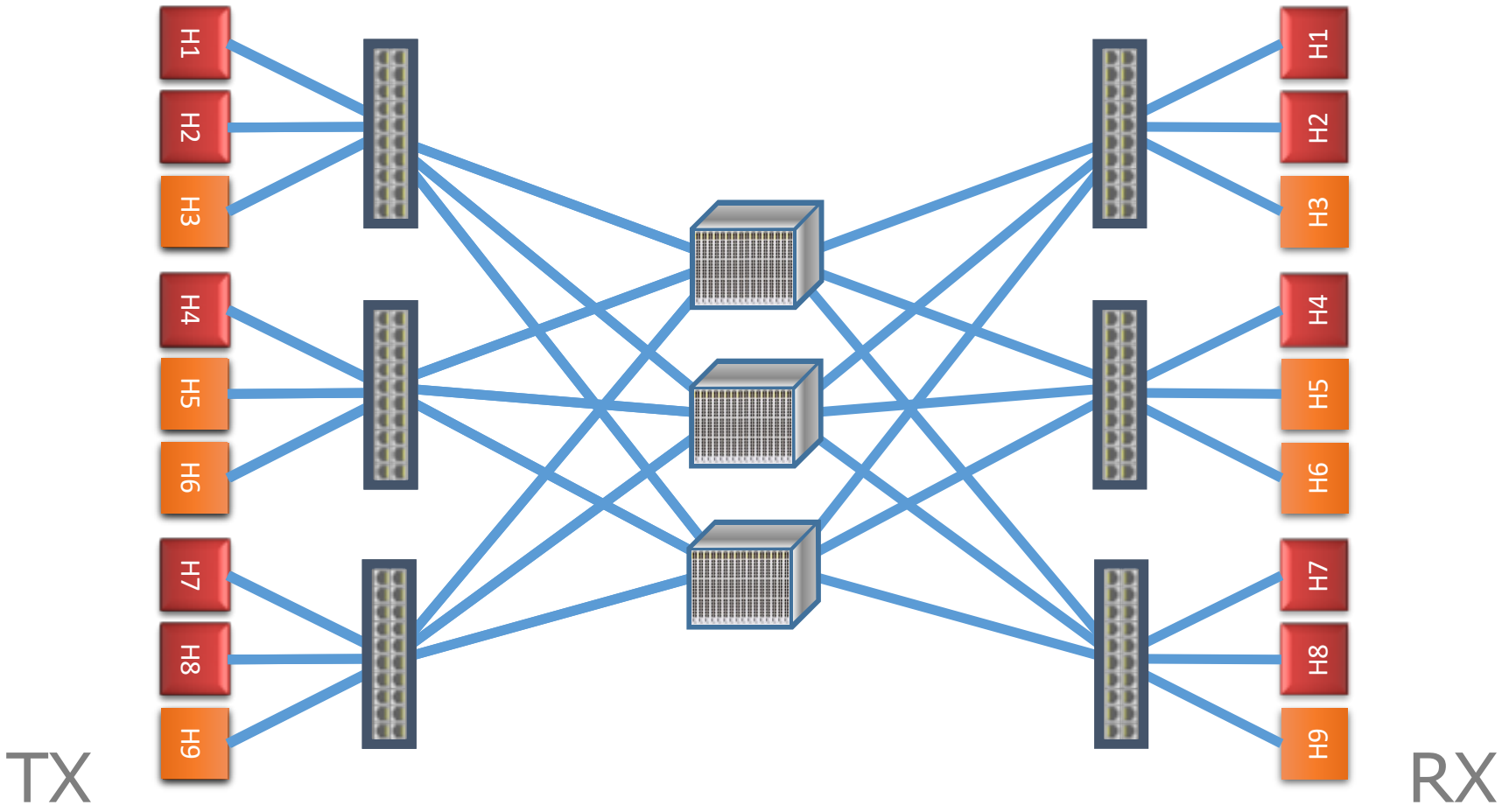
Slides from: Alizadeh, HotNets 2012

DC Network: Just a Giant Switch!



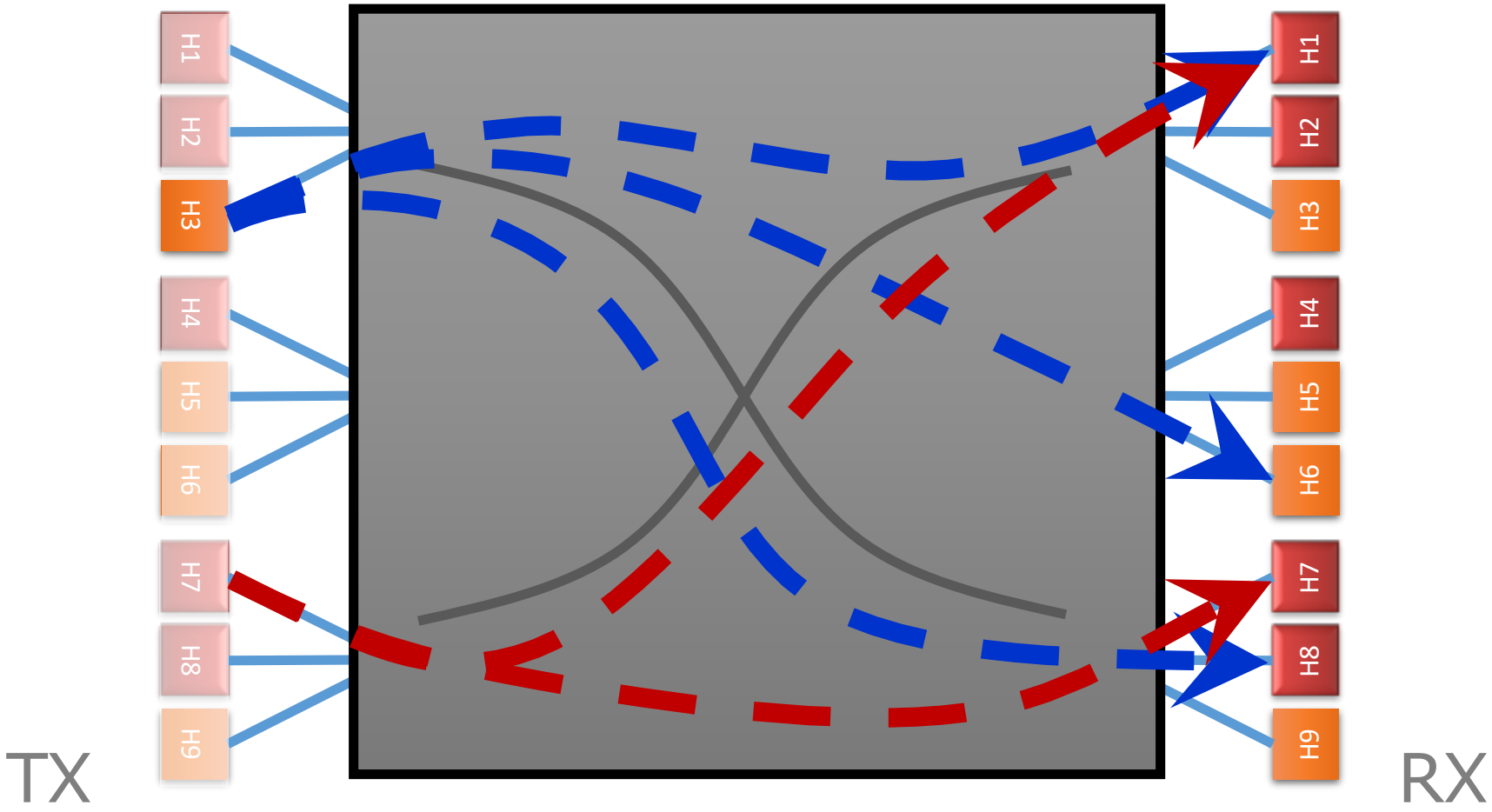
Slides from: Alizadeh, HotNets 2012

DC Network: Just a Giant Switch!



Slides from: Alizadeh, HotNets 2012

DC Network: Just a Giant Switch!

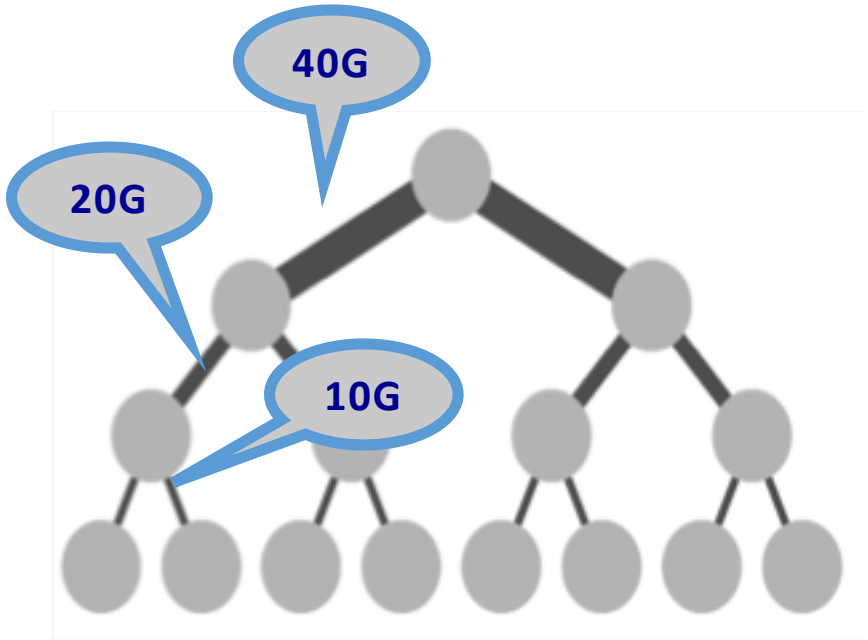


Slides from: Alizadeh, HotNets 2012

Implications of Scale

- High bandwidth communication
  - Ideally: Each server can talk to any other server at its full access link rate
- Conceptually: DC network as one giant switch
  - Would require a 10 Pbits/sec switch!
    - 1M ports (one port/server)
    - 10Gbps per port
- Practical approach: build a network of switches ("fabric") with high "bisection bandwidth"
  - Each switch has practical #ports and link speeds

Achieving Full Bisection Bandwidth

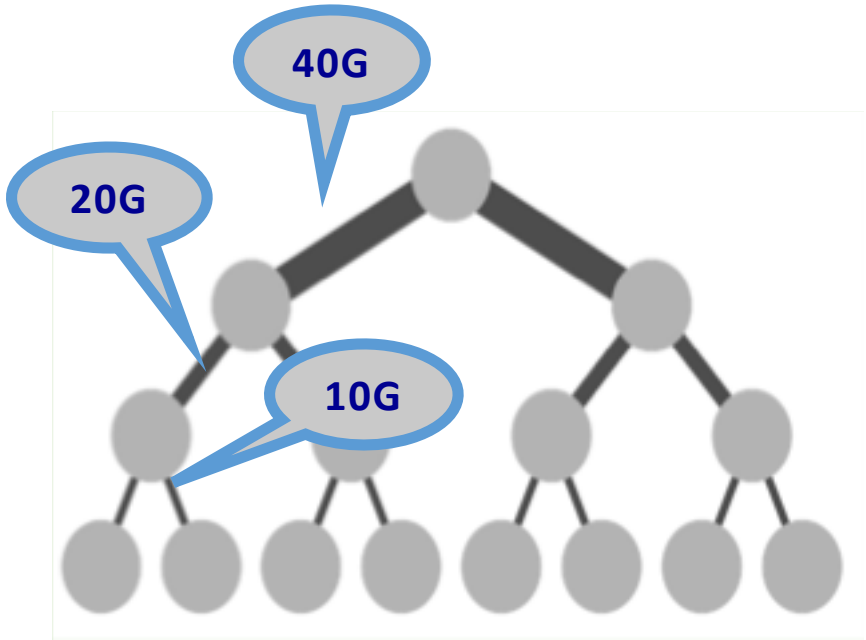


"scale up" approach

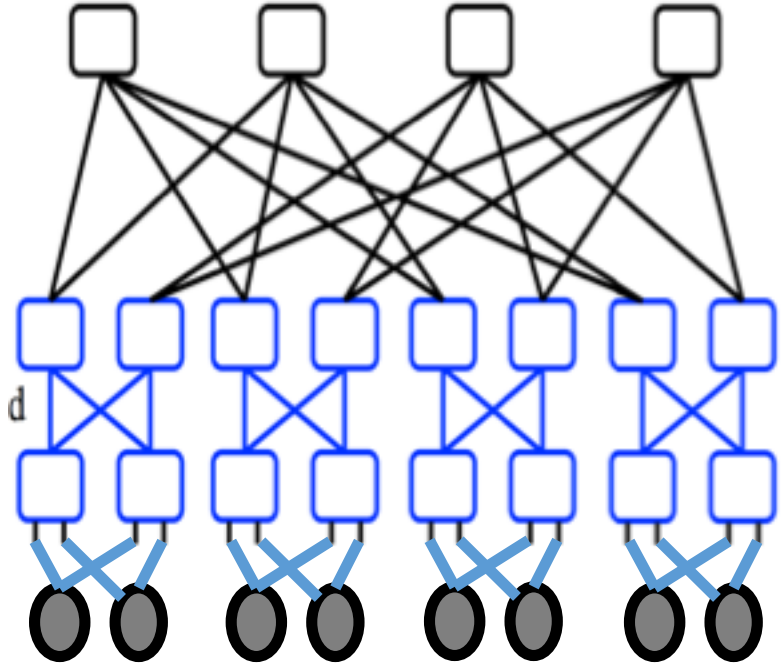
- Partition a network into two equal parts
- Minimum bandwidth between the partitions is the bisection bandwidth

- Problem: "Scaling up" a traditional tree topology is expensive!
  - Over-subscription
  - Single point of failure

Potential Solution: Better Topology



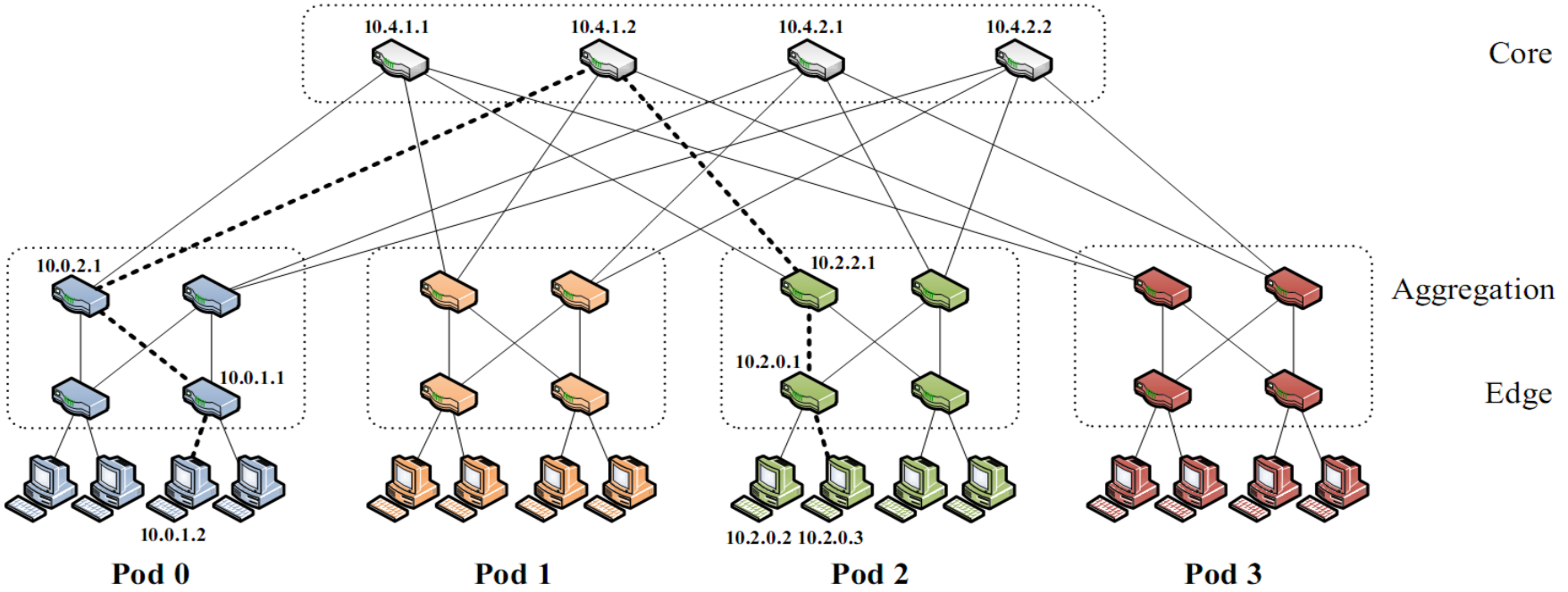
“scale up” approach



“scale out” approach

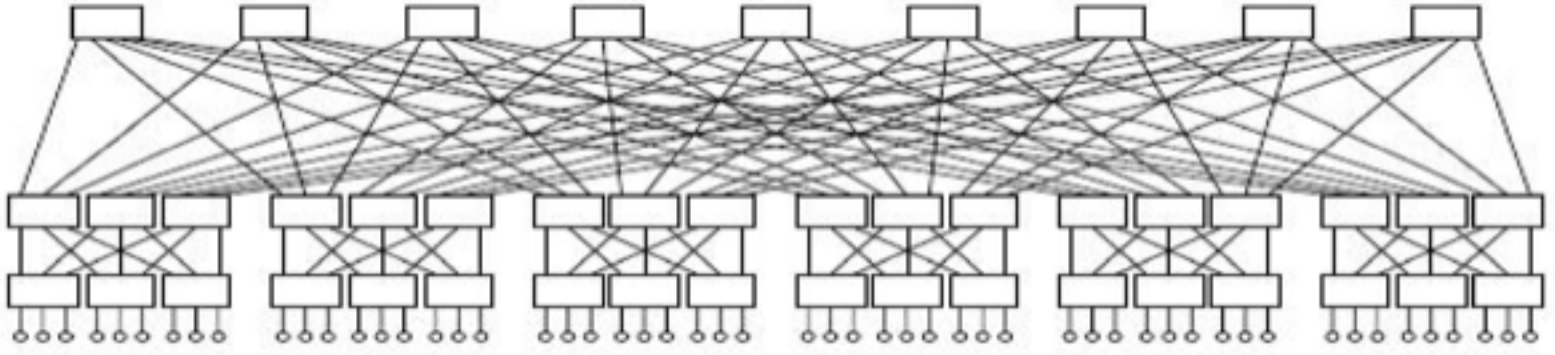
Fat-Tree topology

- K-ary fat tree: three-layer topology (edge, aggregation and core)
  - each pod consists of (k/2)2 servers & 2 layers of k/2 k-port switches
  - each edge switch connects to k/2 servers & k/2 aggr. Switches
  - each aggr. switch connects to k/2 edge & k/2 core switches
  - (k/2)2 core switches: each connects to k pods



Nice properties of fat-tree

- Fat tree has identical bandwidth at any bisections
  - Each layer has the same aggregated bandwidth
- Can be built using cheap devices with uniform capacity
  - Each port supports same speed as end host
  - All devices can transmit at line speed if packets are distributed uniform along available paths
- Great scalability: k-port switch supports k^3/4 servers



However, this is insufficient

- What routing protocols should we run on these switches?
- Layer 2 switch algorithm: data plane flooding!
- Layer 3 IP routing
  - shortest path IP routing will typically use only one path despite the path diversity in the topology
  - if using equal-cost multi-path routing at each switch independently and blindly, packet re-ordering may occur; further load may not necessarily be well-balanced
  - Aside: control plane flooding!

FAT-tree Modified

- Enforce special addressing scheme in DC
  - Allows host attached to same switch to route only through switch
  - Allows intra-pod traffic to stay within pod
  - unused.PodNumber.switchnumber.Endhost
- Use two level look-ups to distribute traffic and maintain packet ordering

2-level lookups

- First level is prefix lookup
  - Used to route down the topology to endhost
  - Highest 3 bytes (prefix /24)
- Second level is a suffix lookup
  - Lowest 1 byes (suffix /8)
  - Used to route up towards core
  - Diffuses and spreads out traffic
  - Maintains packet ordering by using the same ports for the same endhost

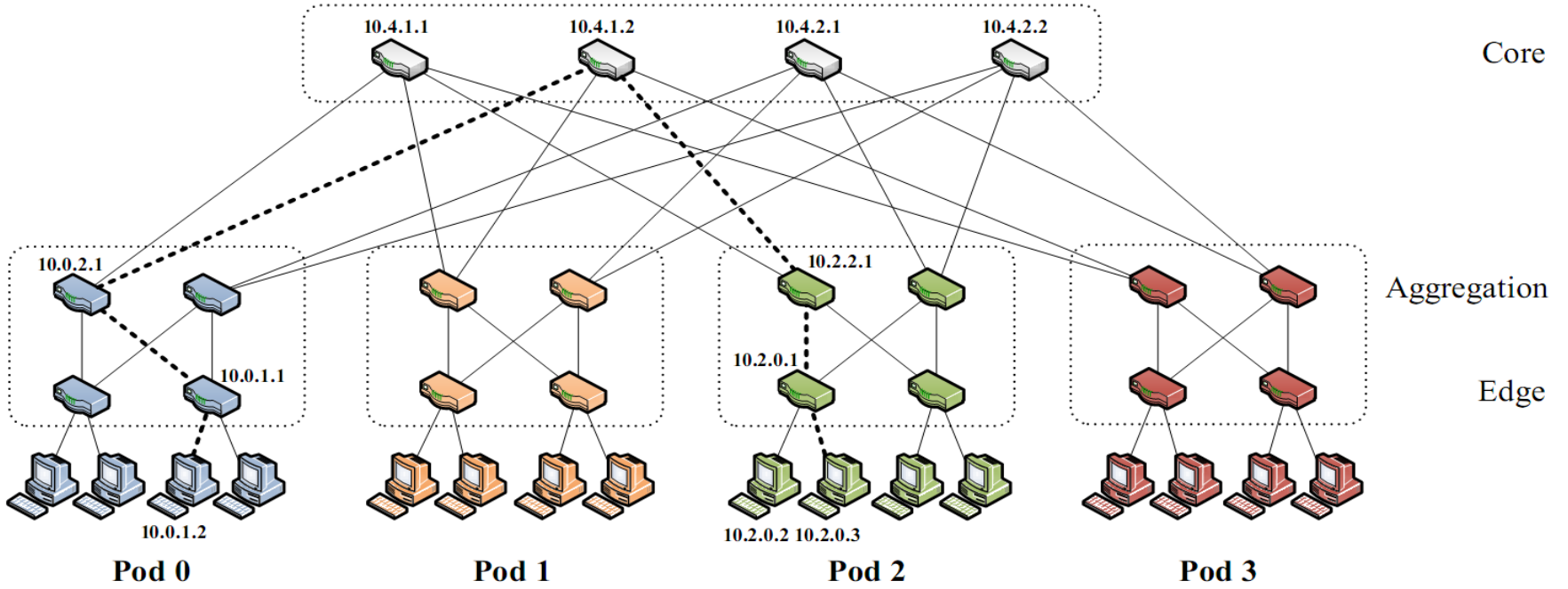
Routing example

unused.PodNumber.switchnumber.Endhost

- Prefix lookup: route down the topology to endhost
- Suffix lookup: Used to route up towards core

Prefix	Output port
10.2.0.0/24	0
10.2.1.0/24	1
0.0.0.0/0	

Suffix	Output port
0.0.0.2/8	2
0.0.0.3/8	3



Further Optimizations

- Flow classification
  - Eliminates local congestion
  - Assign to traffic to ports on a per-flow basis instead of a per-host basis
- Flow scheduling
  - Eliminates global congestion
  - Prevent long lived flows from sharing the same links
  - Assign long lived flows to different links