Data Centers &
Co-designed Distributed Systems

A Data Center

Inside a Data Center

Data center

10k - 100k servers: 250k – 10M cores

1-100PB of DRAM

100PB - 10EB storage

1- 10 Pbps bandwidth (>> Internet)

10-100MW power

- 1-2% of global energy consumption

100s of millions of dollars

Servers

Limits driven by the power consumption

1-4 multicore sockets

20-24 cores/socket (150W each)

100s GB – 1 TB of DRAM (100-500W)

40Gbps link to network switch

Servers in racks

19” wide

1.75” tall (1u)

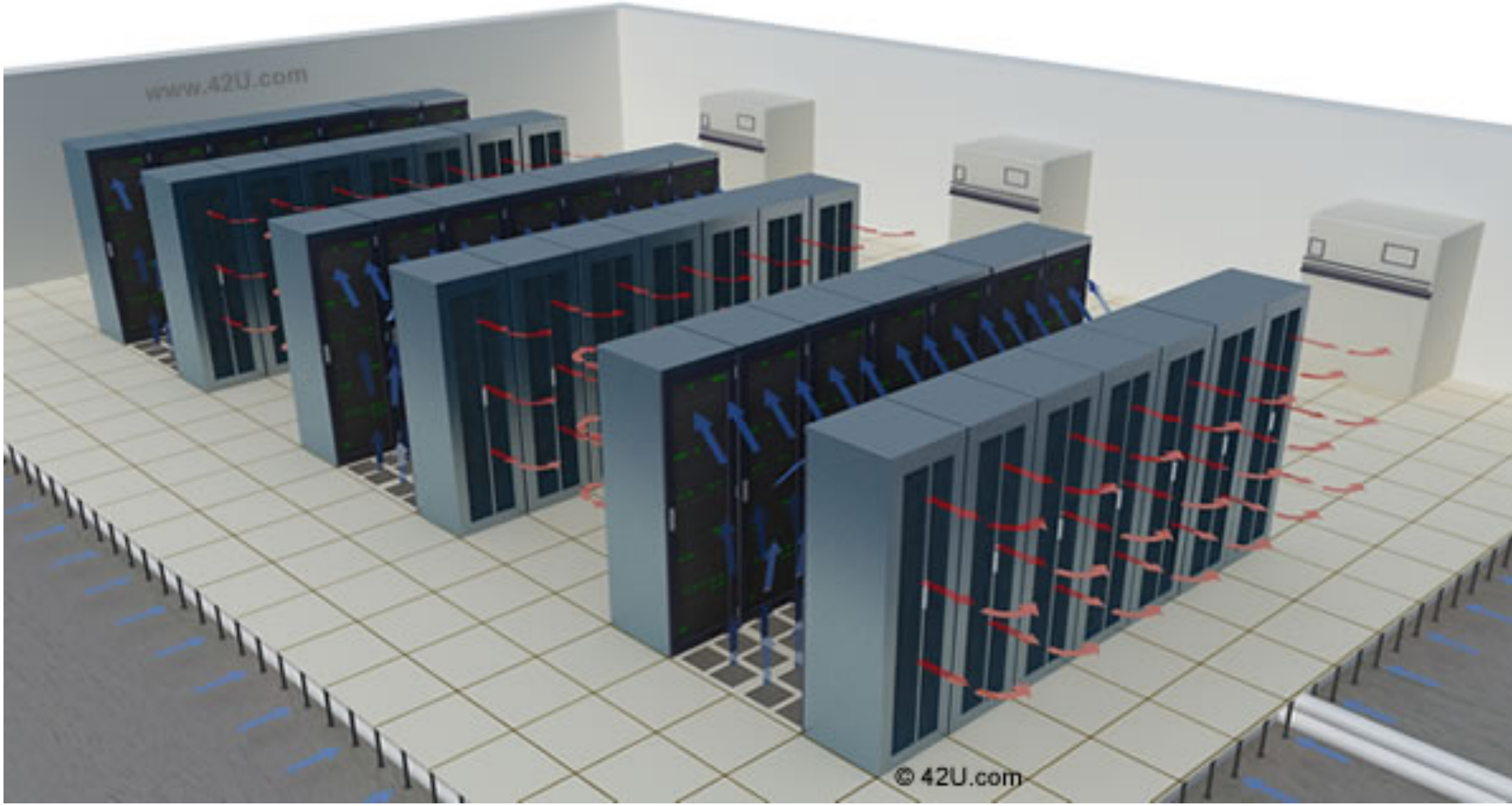
(defined decades back!)

40-120 servers/rack

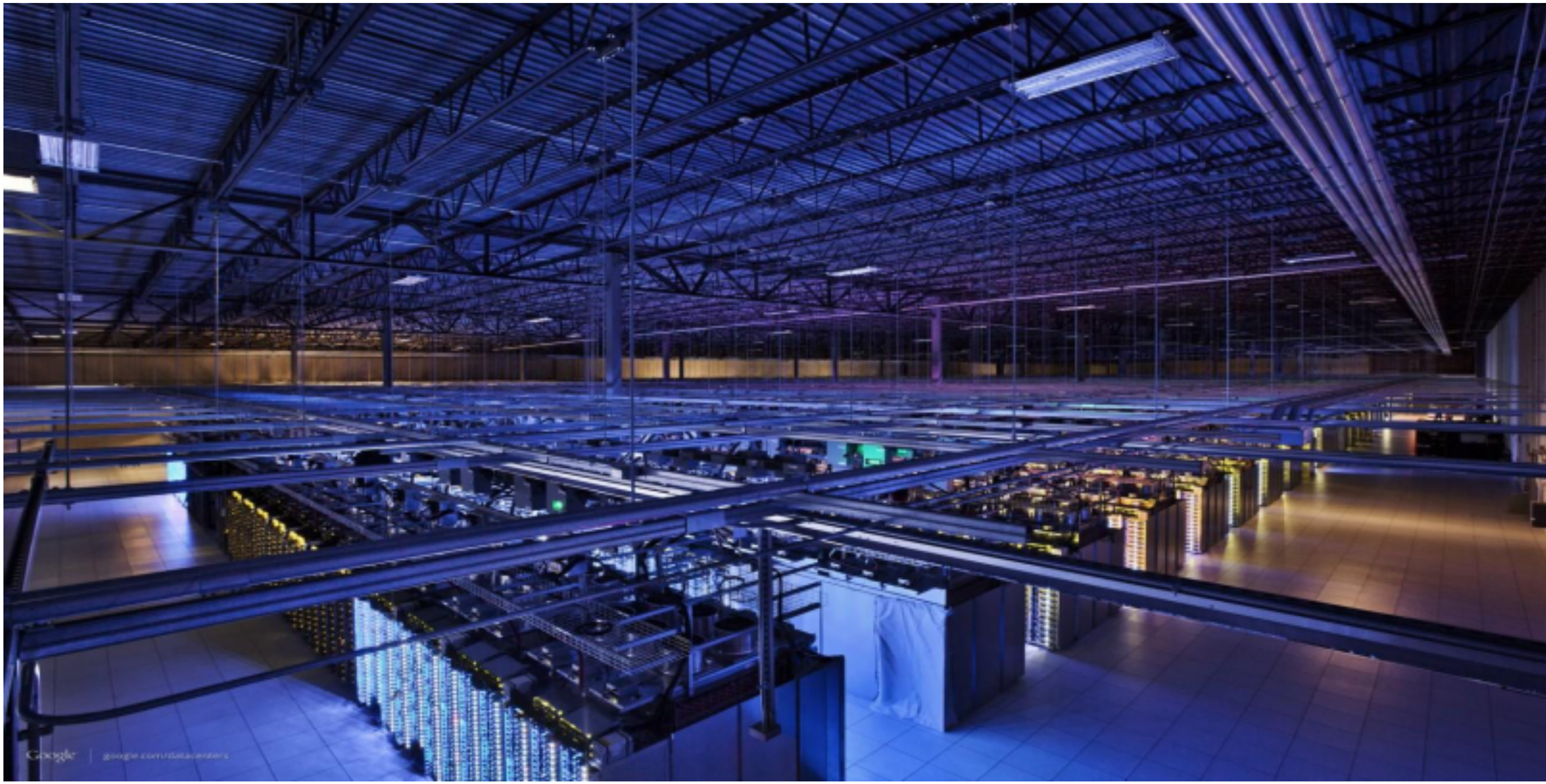
network switch at top

Racks in rows

Rows in hot/cold pairs



Hot/cold pairs in data centers



Where is the cloud?

Amazon, in the US:

- Northern Virginia
- Ohio
- Oregon
- Northern California

Many reasons informing the locations.

MTTF/MTTR

Mean Time to Failure/Mean Time to Repair

Disk failures (not reboots) per year ~ 2-4%

- At data center scale, that's about 2/hour.
- It takes 10 hours to restore a 10TB disk

Server crashes

- 1/month * 30 seconds to reboot => 5 mins/year
- 100K+ servers

Data Center Networks

Every server wired to a ToR (top of rack) switch

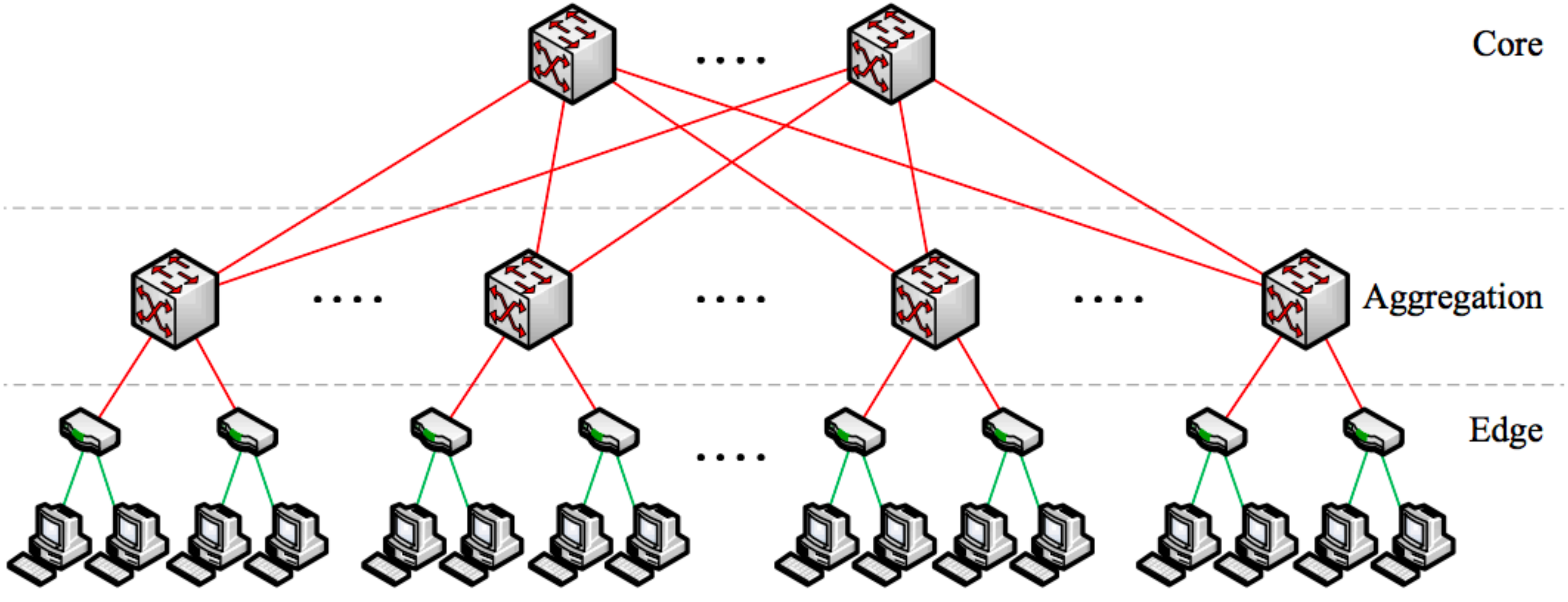
ToR's in neighboring aisles wired to an aggregation switch

Agg. switches wired to core switches

Early data center networks

3 layers of switches

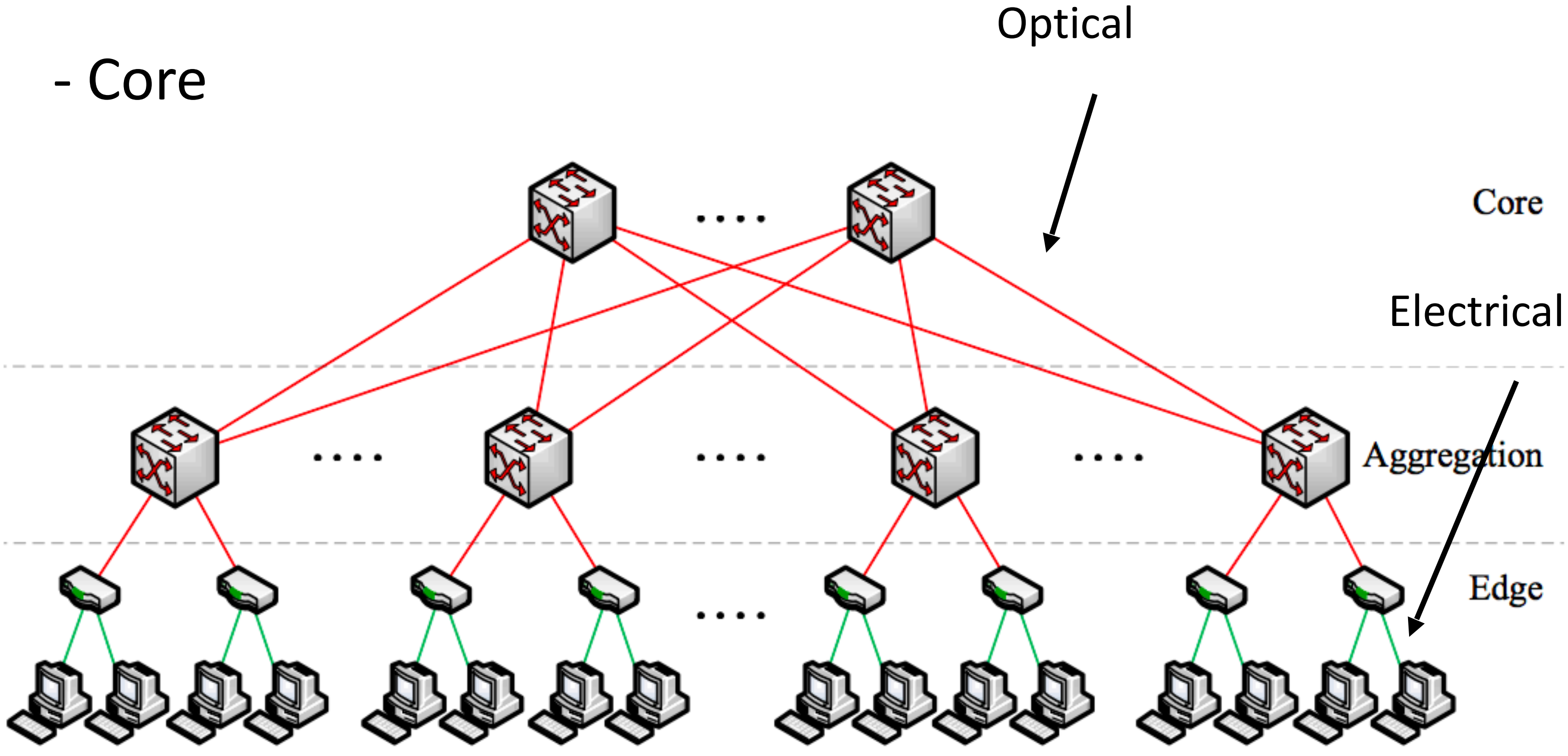
- Edge (ToR)
- Aggregation
- Core



Early data center networks

3 layers of switches

- Edge (ToR)
- Aggregation
- Core



Early data center limitations

Cost

- Core, aggregation routers = high capacity, low volume

- Expensive!

Fault-tolerance

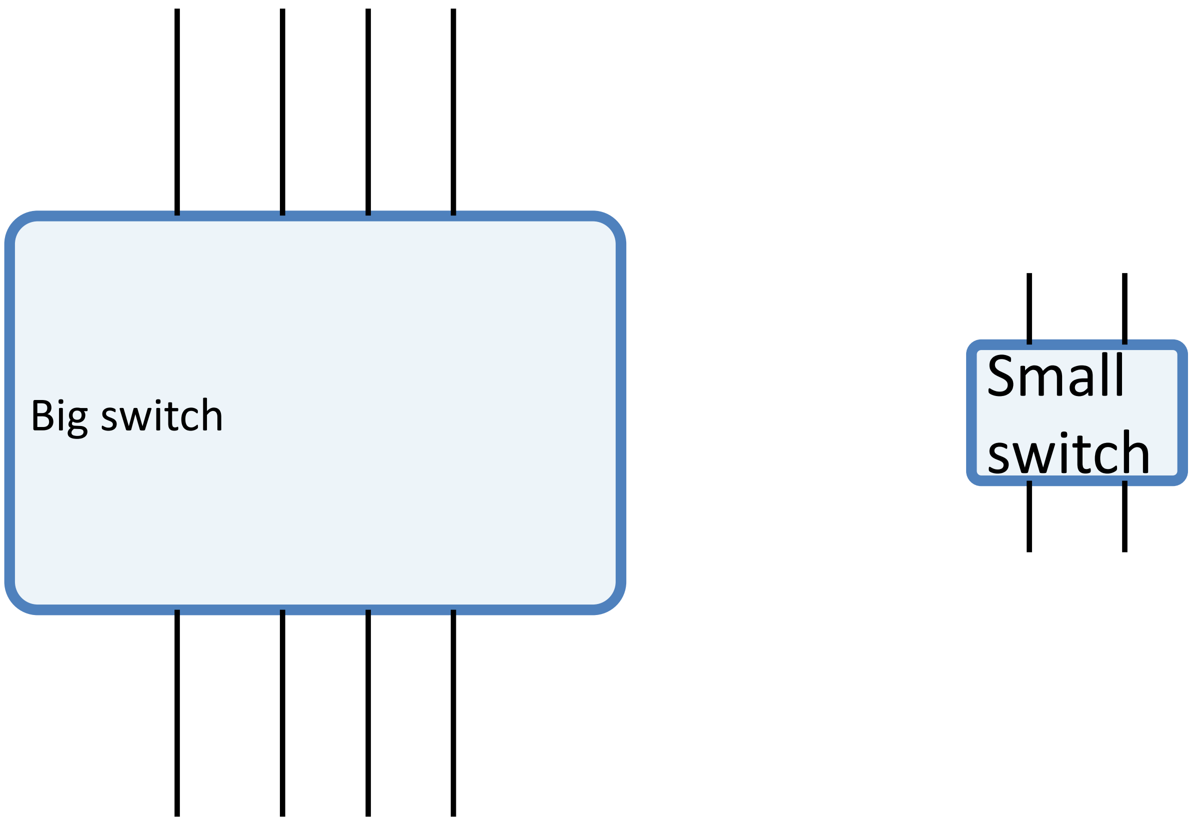
- Failure of a single core or aggregation router = large bandwidth loss

Bisection bandwidth limited by capacity of largest available router

- Google's DC traffic doubles every year!

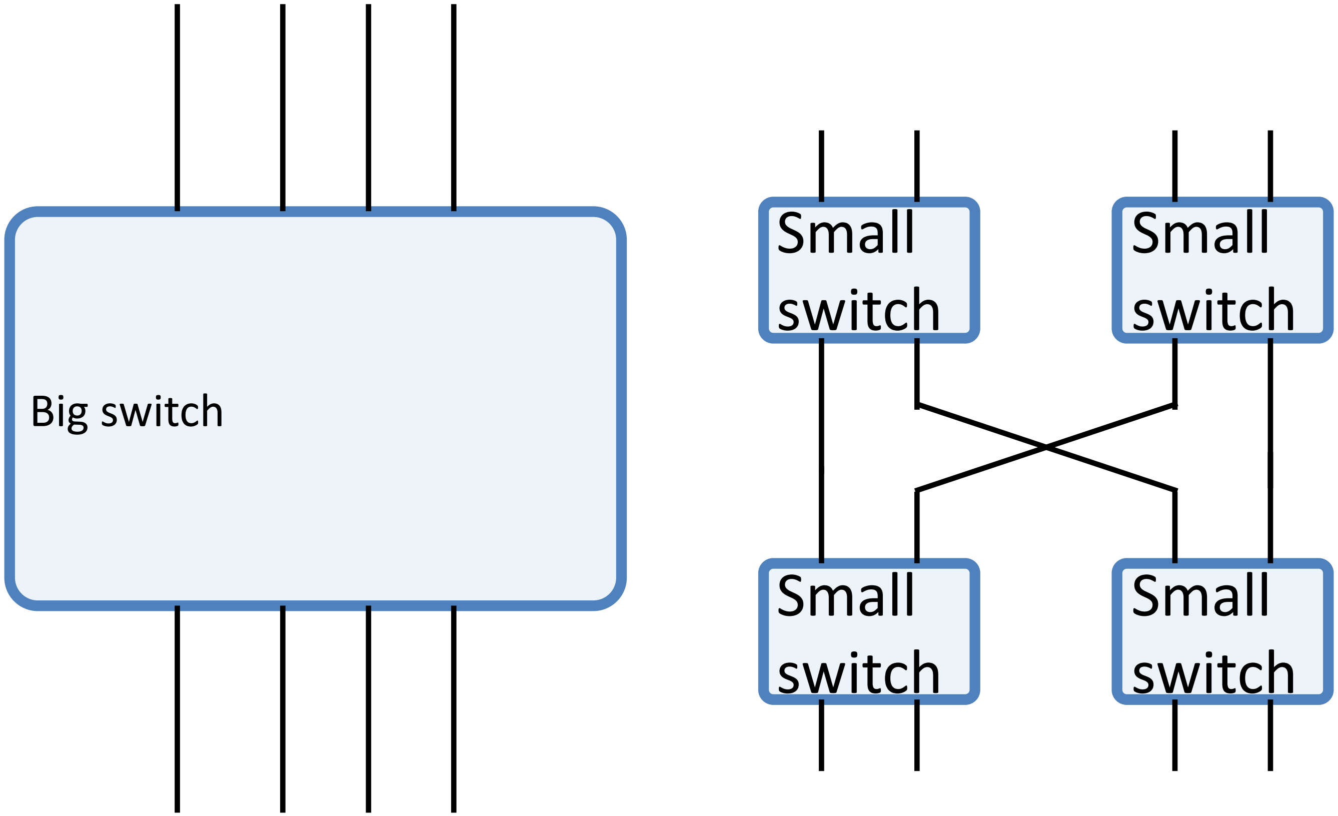
Clos networks

How can I replace a big switch by many small switches?



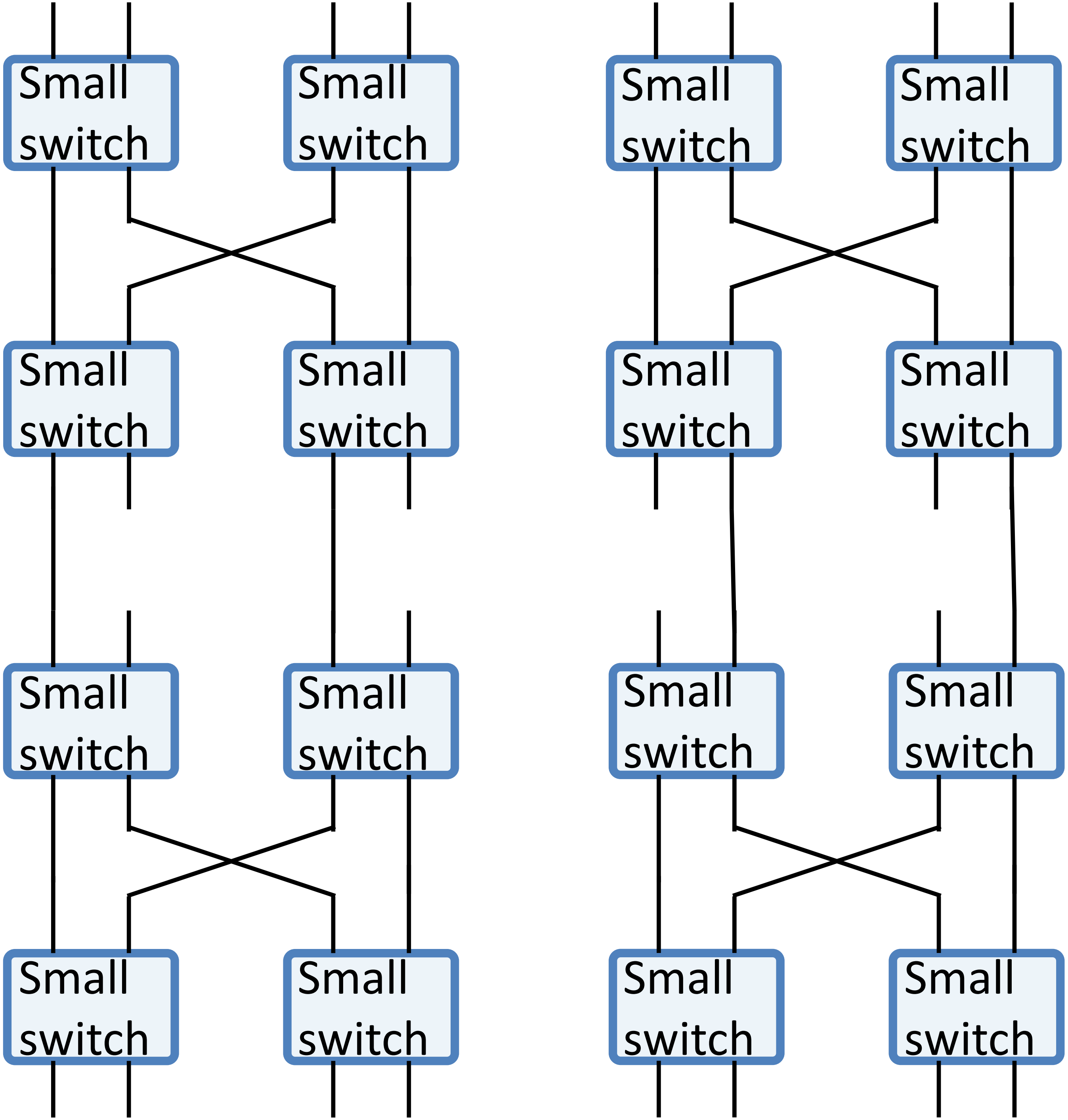
Clos networks

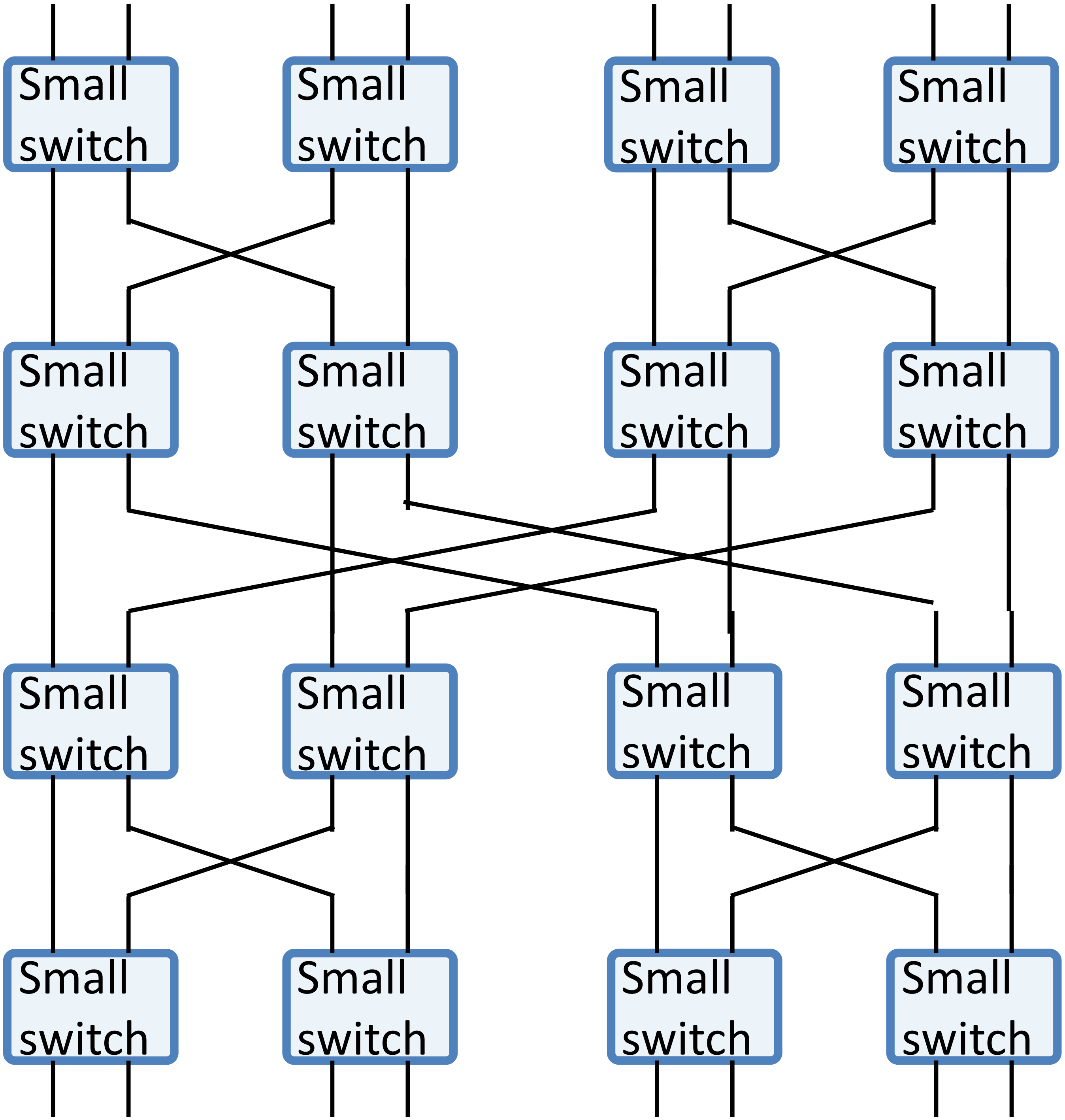
How can I replace a big switch by many small switches?



Clos Networks

What about bigger switches?


Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch


Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch
Small switch

Multi-rooted tree

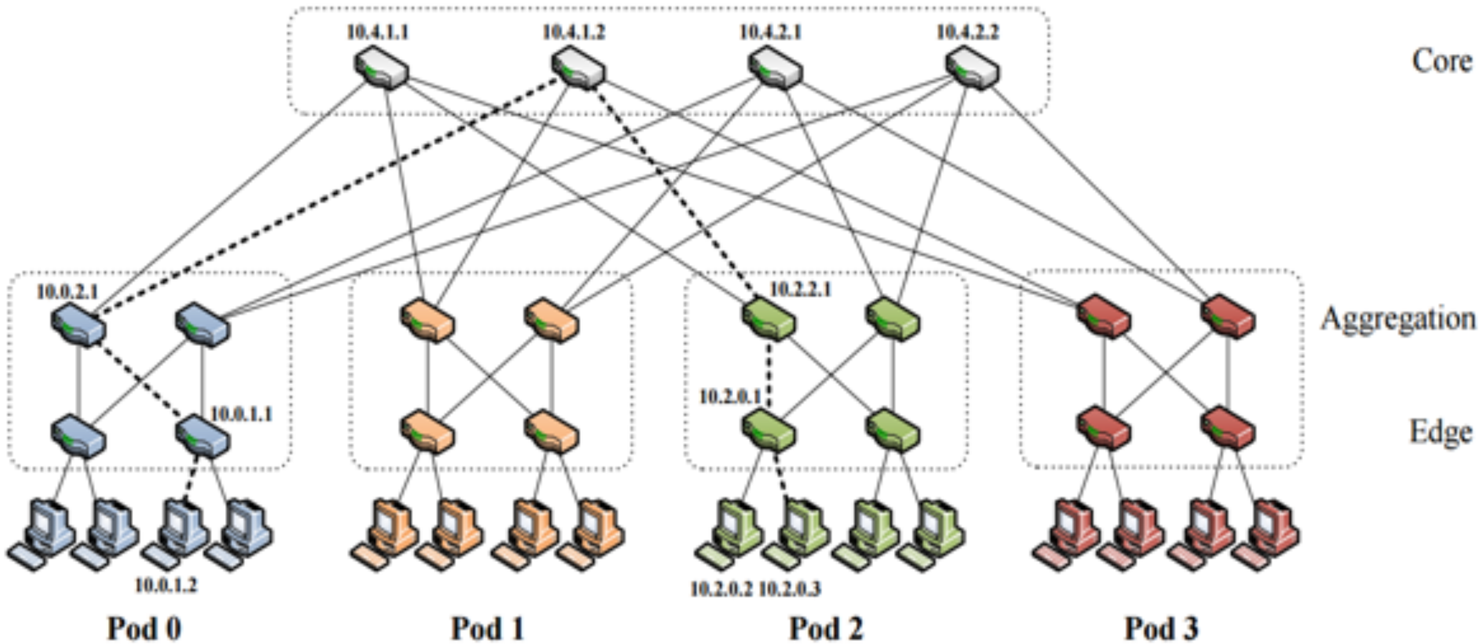


Figure 3: Simple fat-tree topology. Using the two-level routing tables described in Section 3.3, packets from source 10.0.1.2 to destination 10.2.0.3 would take the dashed path.

Every pair of nodes has many paths

Fault tolerant! But how do we pick a path?

Multipath routing

Lots of bandwidth, split across many paths

ECMP: hash on packet header to determine route

- (5 tuple): Source IP, port, destination IP, port, prot.
- Packets from client – server usually take same route

On switch or link failure, ECMP sends subsequent packets along a different route

=> Out of order packets!

Data Center Network Trends

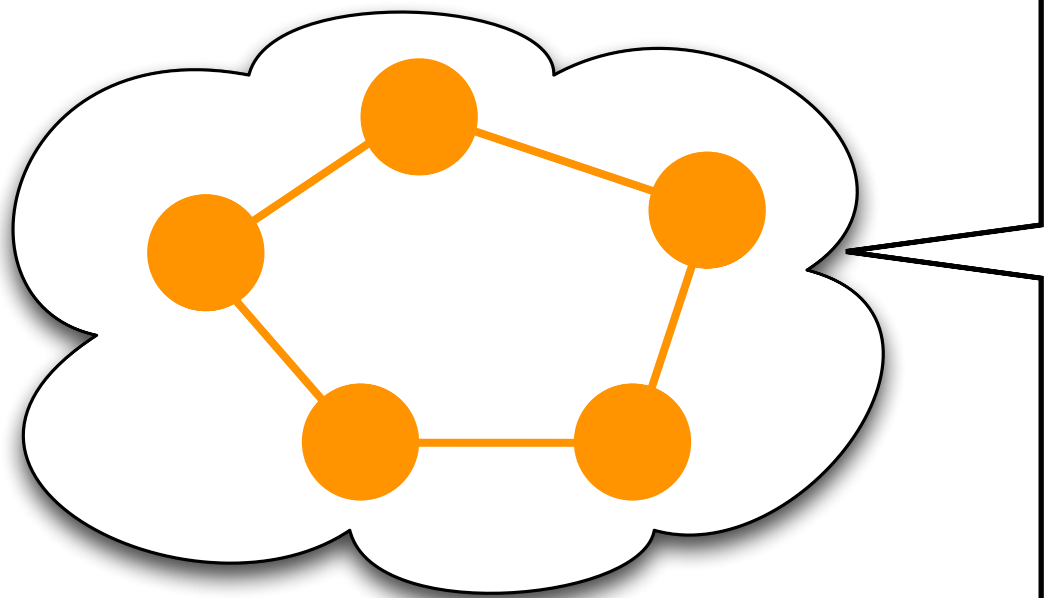
- RT latency across data center ~ 10 usec
- 40 Gbps links common, 100 Gbps on the way
  - 1KB packet every 80ns on a 100Gbps link
  - Direct delivery into the on-chip cache (DDIO)
- Upper levels of tree are (expensive) optical links
  - Thin tree to reduce costs
- Within rack > within aisle > within DC > cross DC
  - Latency and bandwidth: keep communication local

Local Storage

- Magnetic disks for long term storage
  - High latency (10ms), low bandwidth (250MB/s)
  - Compressed and replicated for cost, resilience
- Solid state storage for persistence, cache layer
  - 50us block access, multi-GB/s bandwidth
- Emerging NVM
  - Low energy DRAM replacement
  - Sub-microsecond persistence

Co-designing Systems inside the Datacenter

Network is minimalistic

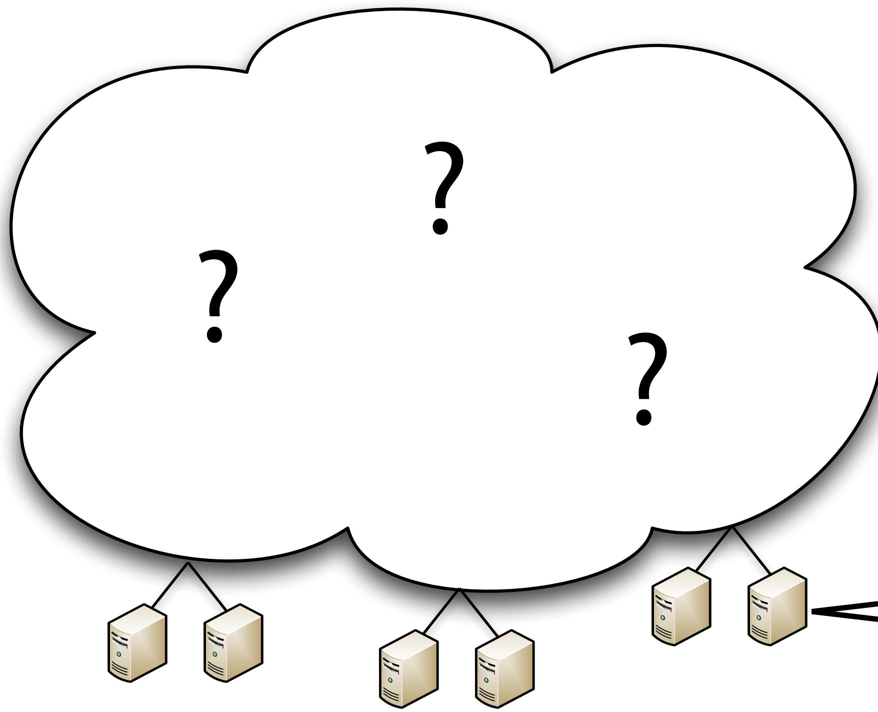


best effort delivery

simple primitives

minimal guarantees

Distributed Systems assume the worst



packets may be arbitrarily

- dropped
- delayed
- reordered

asynchronous network!

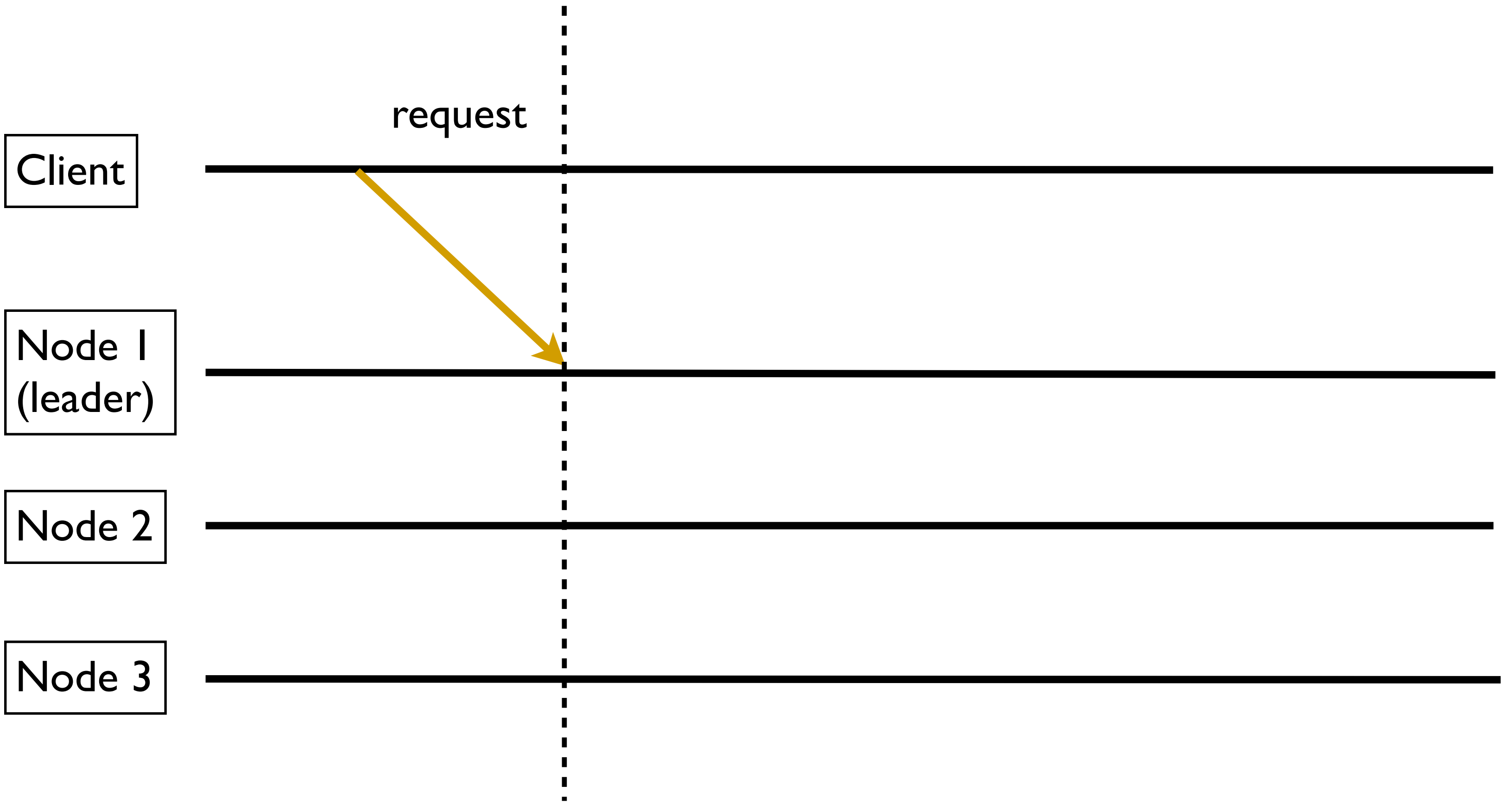
Data Center Networks

- DC Networks can exhibit stronger properties:
  - controlled by single entity
  - trusted, extensible
  - predictable, low latency

Research Questions

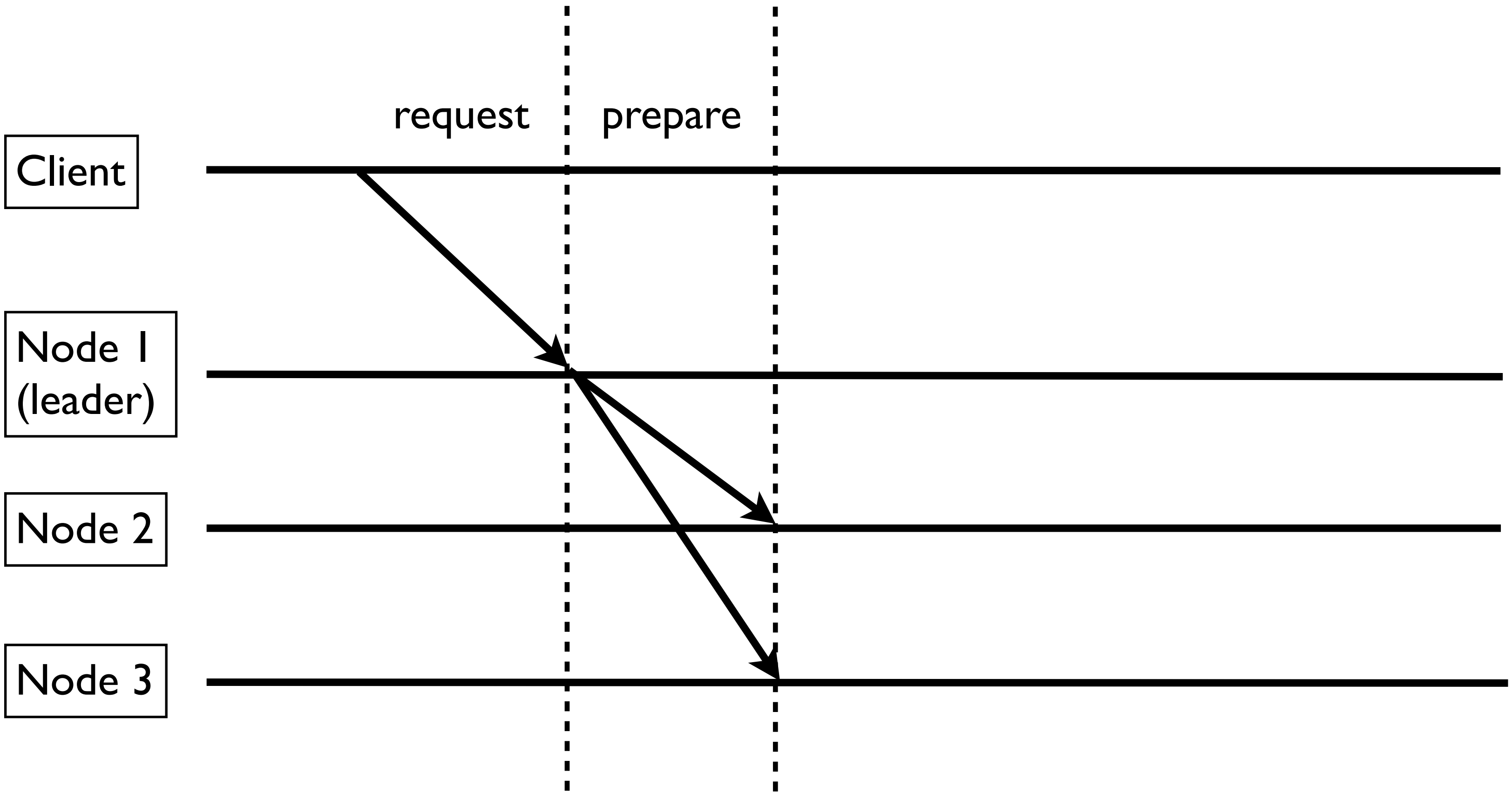
- *Can we build an approximately synchronous network?*
- *Can we co-design networks and distributed systems?*

Paxos



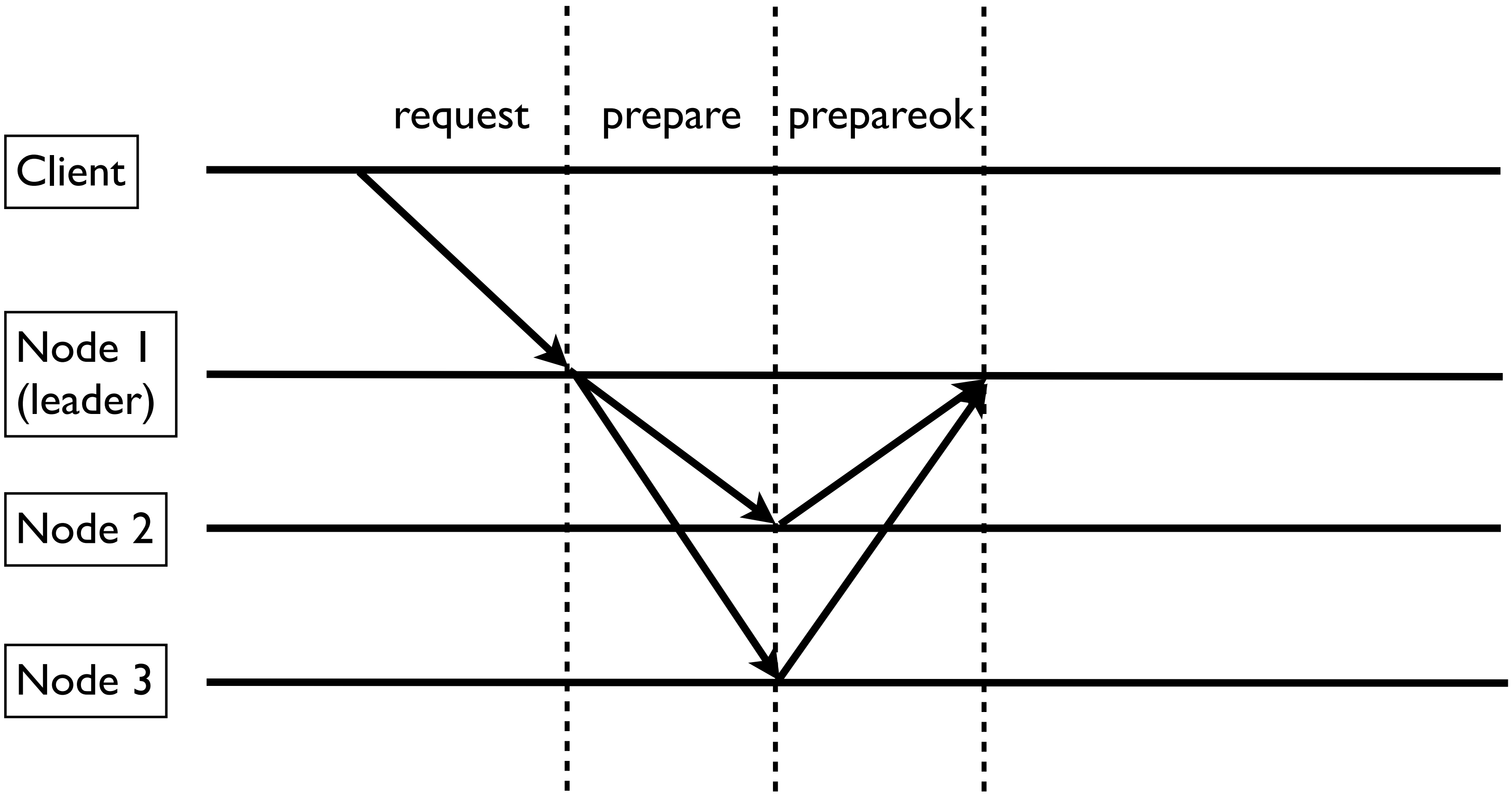
- Paxos typically uses a leader to order requests
- Client request sent to the leader

Paxos



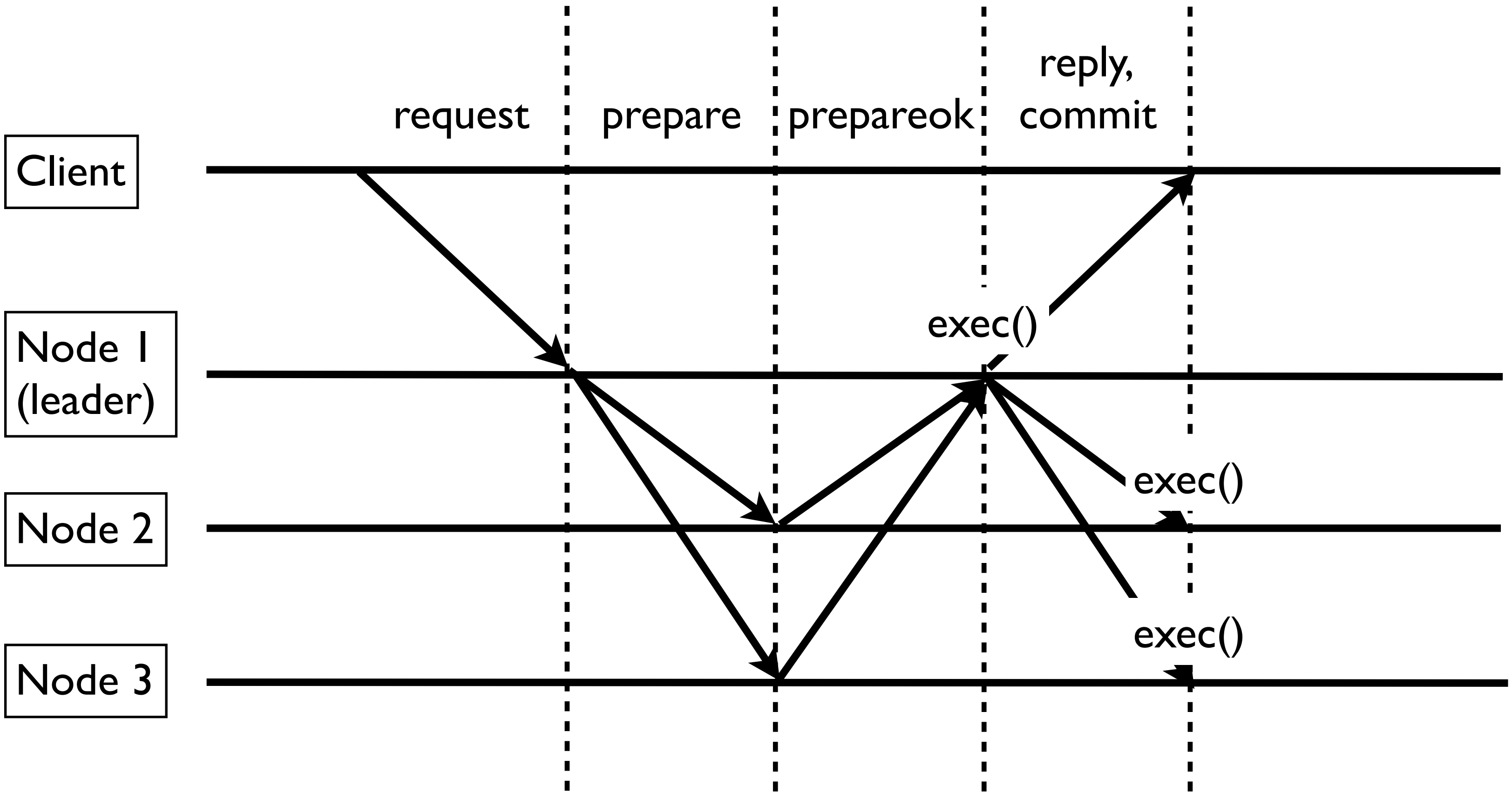
- Leader sequences operations; sends to replicas

Paxos



- Replicas respond; leader waits for *f+1* replies

Paxos



- Leader executes; replies to client; commits to nodes

Performance Analysis

- End-to-end latency: *4 messages*
- Leader load: *2n messages*

- Leader sequencing increases latency and reduces throughput

- Can we design a “leader-less” system?

- Can the network provide stronger delivery properties?

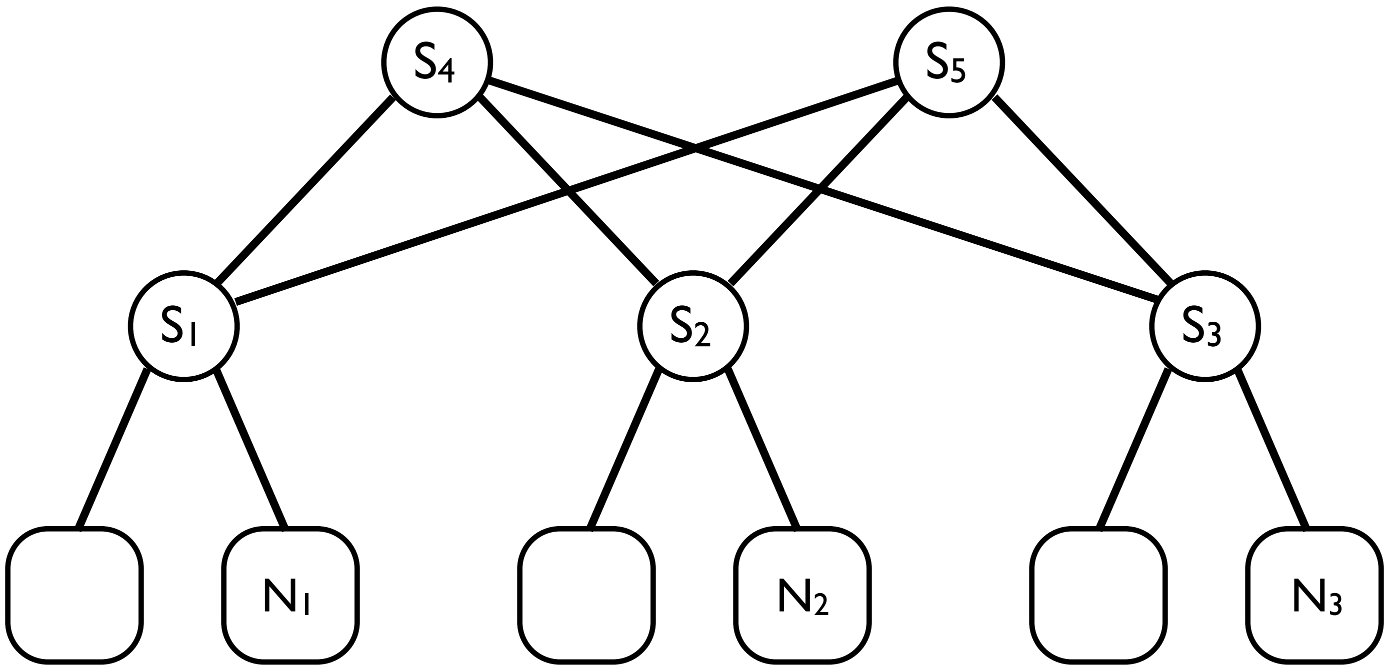
Mostly Ordered Multicasts

- *Best-effort ordering* of concurrent multicasts
- Given two concurrent multicasts m_1 and m_2

  If a node receives m_1 and m_2, then all other nodes will process them in the same order with *high probability*

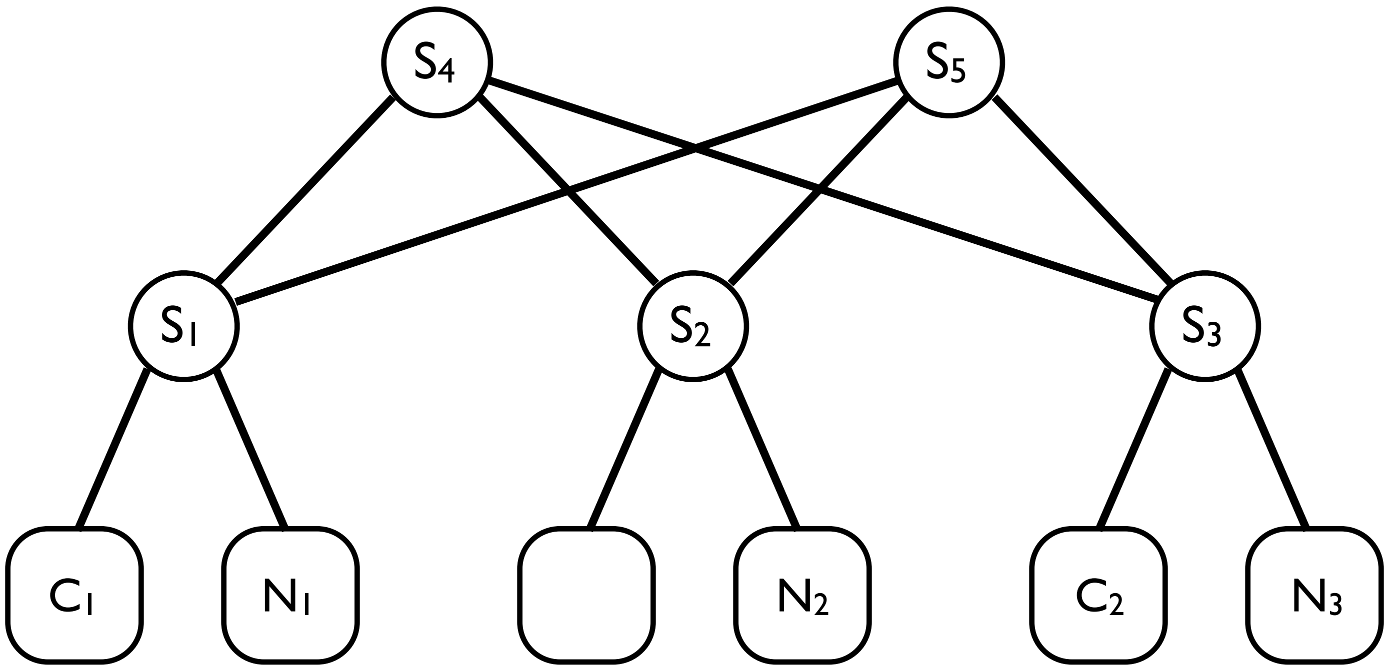
- More practical than *totally ordered multicasts;* but not satisfied by existing multicast protocols

Traditional Network Multicast



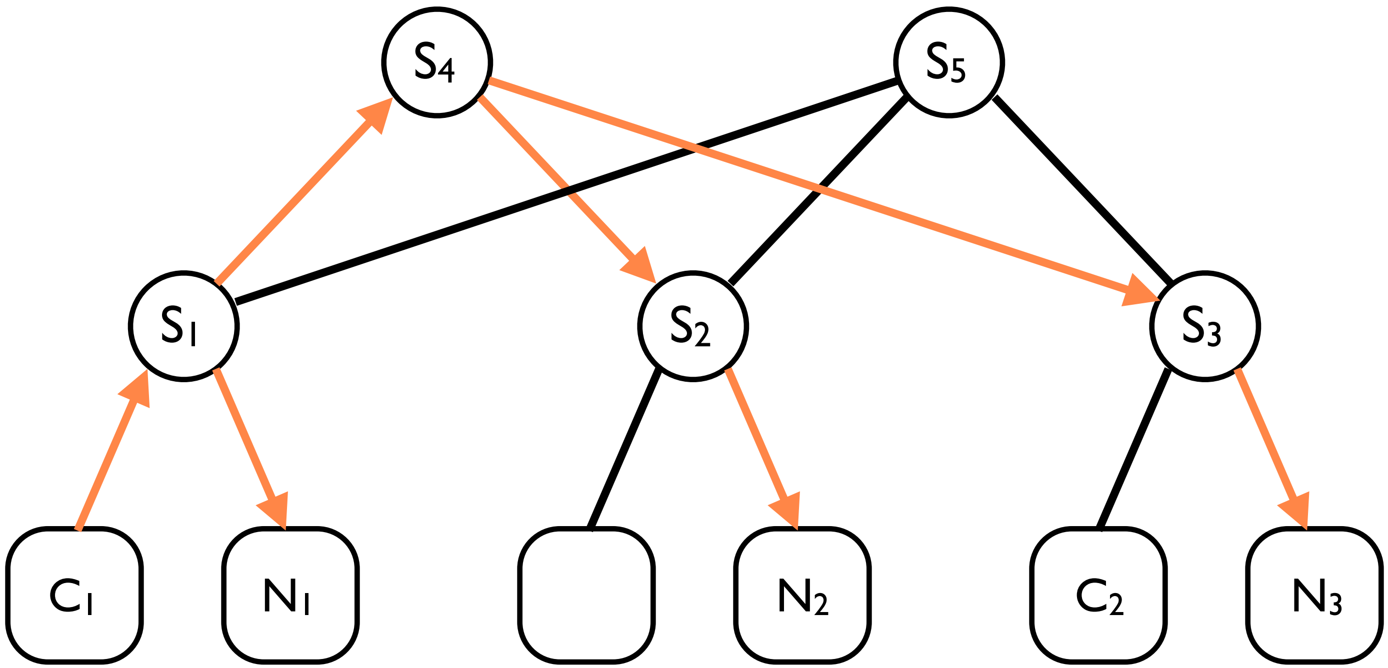
Consider a symmetric DC network with three replica nodes

Traditional Network Multicast



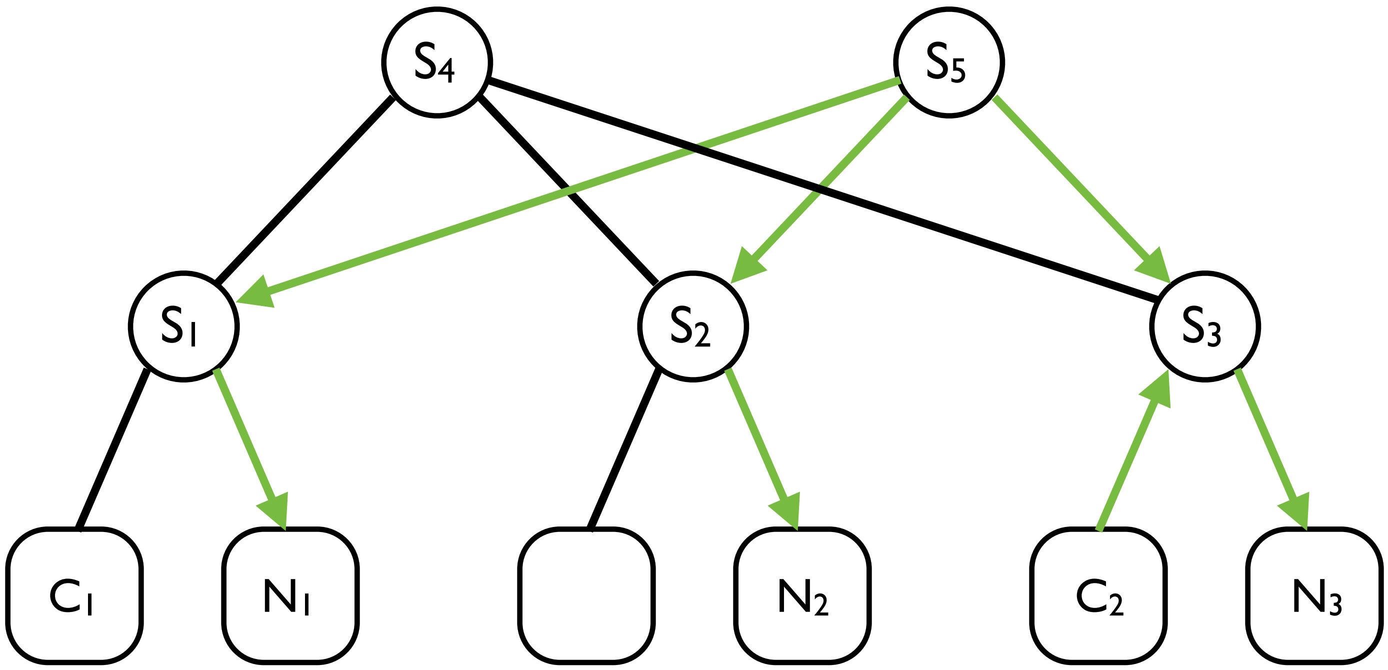
Let two clients issue concurrent multicasts

Traditional Network Multicast



Multicast messages travel different path lengths

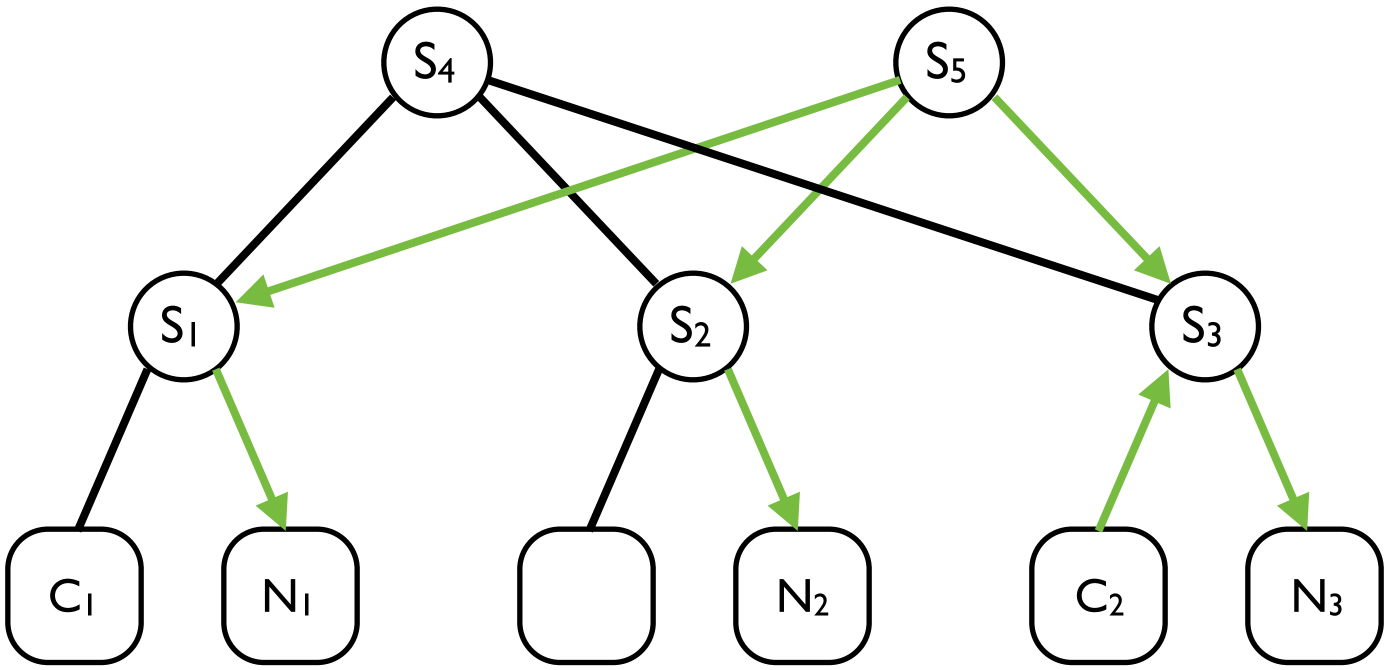
Traditional Network Multicast



N_1 is closer to C_1 while N_3 is closer to C_2

Different multicasts traverse links with different loads

Traditional Network Multicast

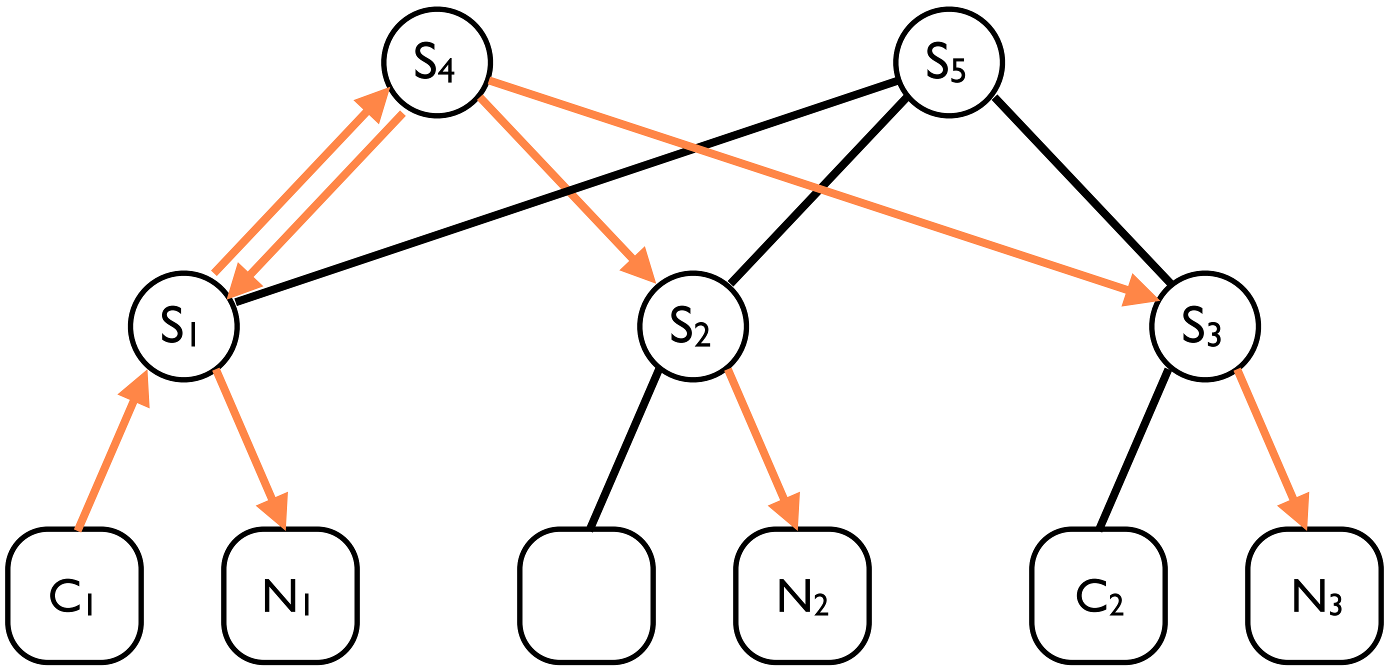


Simultaneous multicasts will be received in arbitrary order by replica nodes

Mostly Ordered Multicast

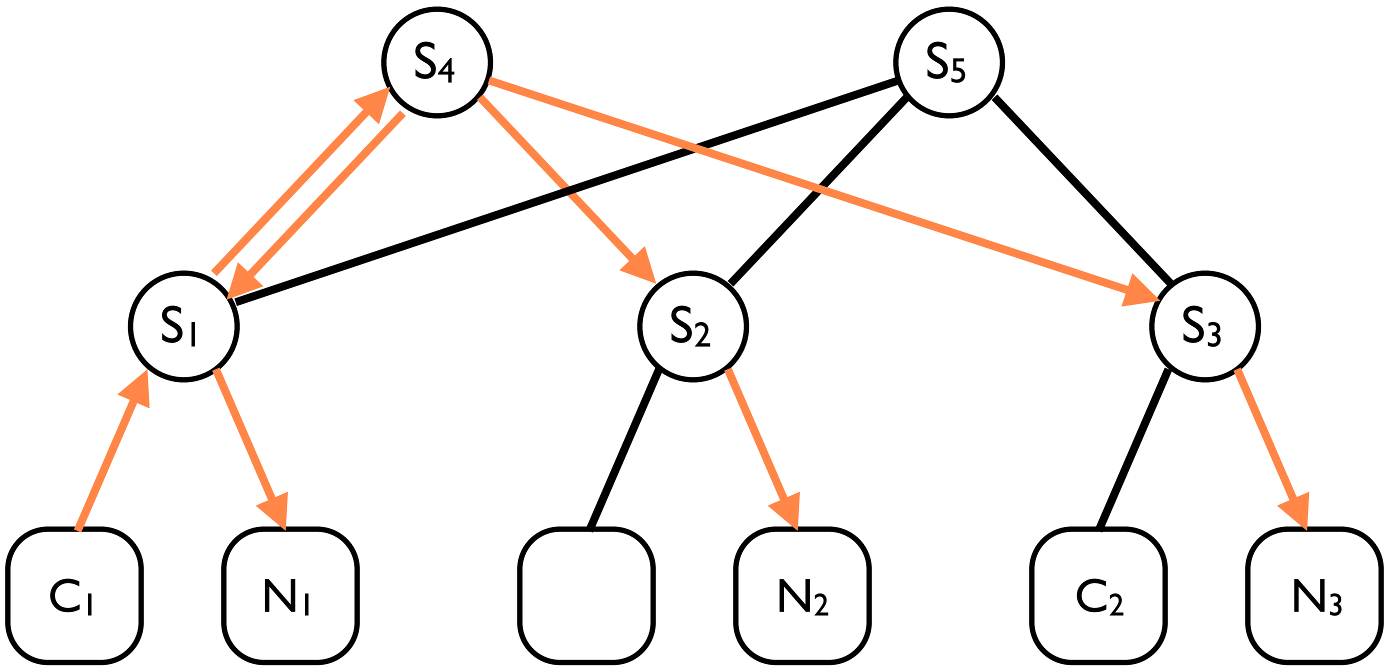
- Ensure that all multicast messages traverse the same number of links
- Minimize reordering due to congestion induced delays

Mostly Ordered Multicast



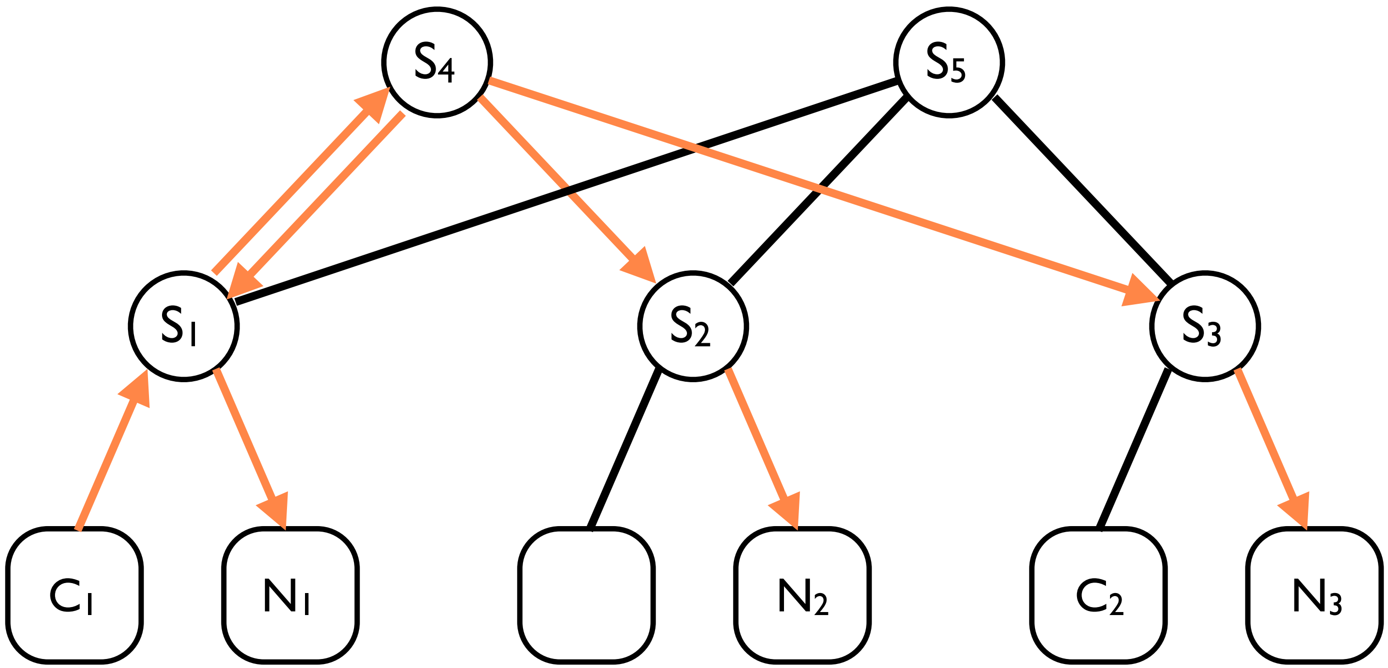
Step 1: Route multicast messages always through a root switch equidistant from receivers

Mostly Ordered Multicast



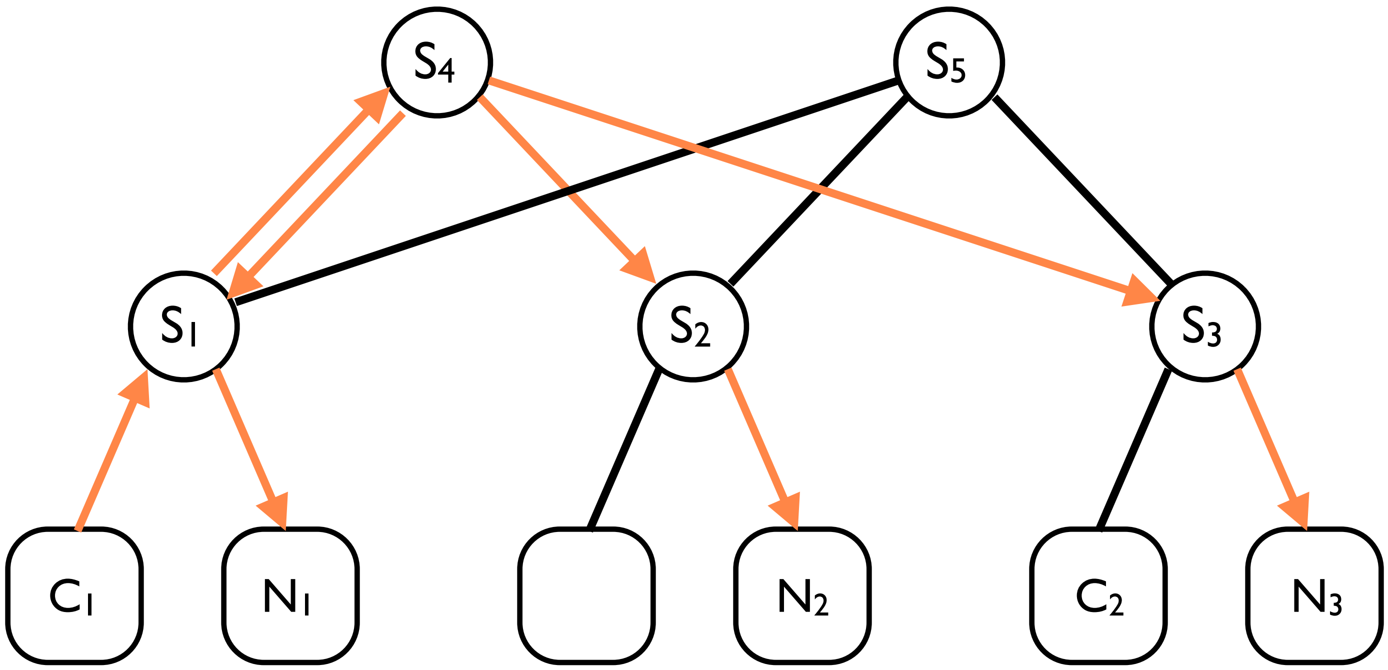
Step 2: Perform in-network replication at the root switch or on the downward path

Mostly Ordered Multicast



Step 3: Use the same root switch if possible (especially when there are multiple multicast groups)

Mostly Ordered Multicast



Step 4: Enable QoS prioritization on multicast messages on the downward path; queueing delay at most one message/switch

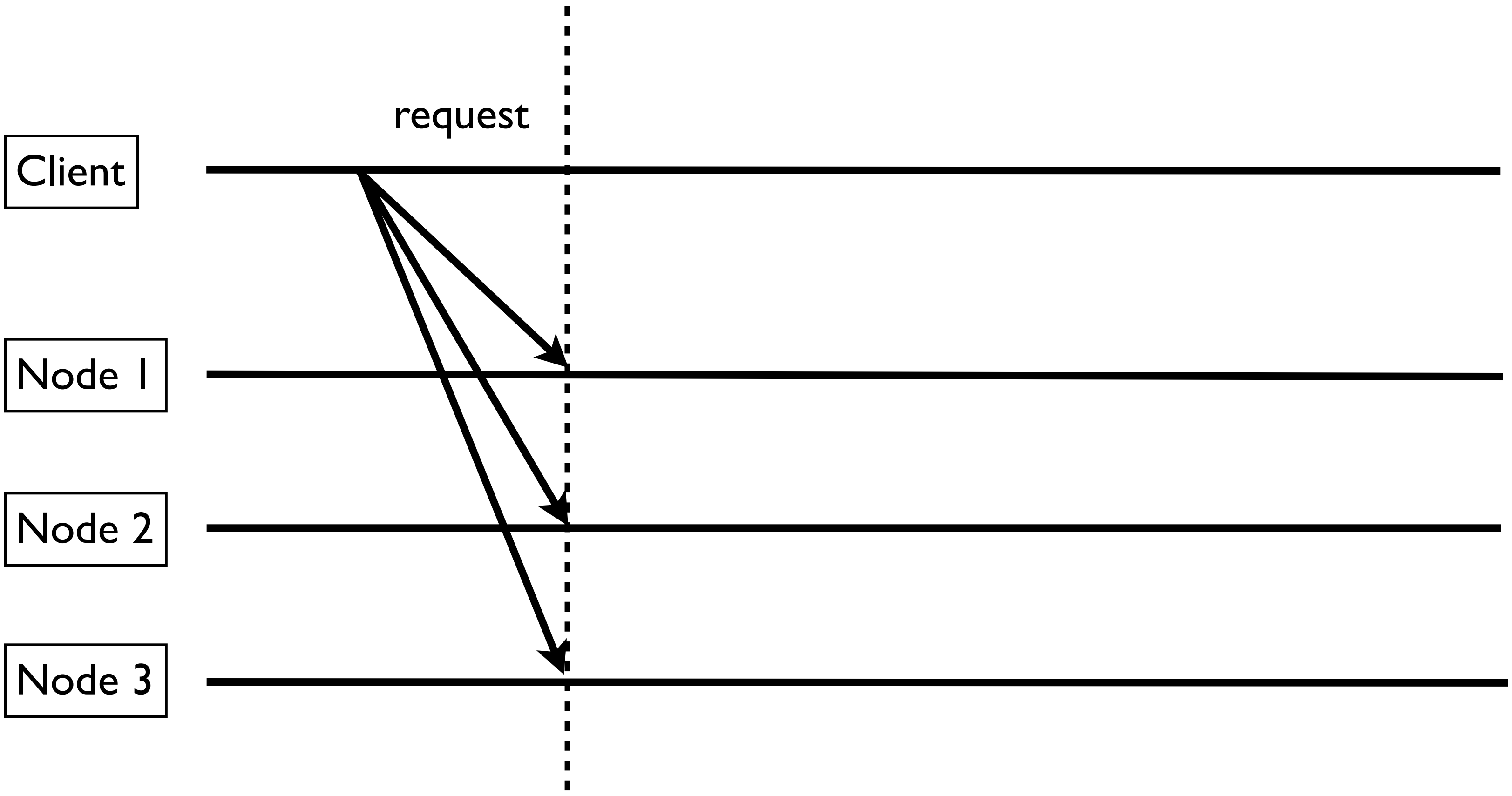
MOM Implementation

- Easily implemented using OpenFlow/SDN
- Multicast groups represented using virtual IPs
- Routing based on both the destination and the direction of traffic flow

Speculative Paxos

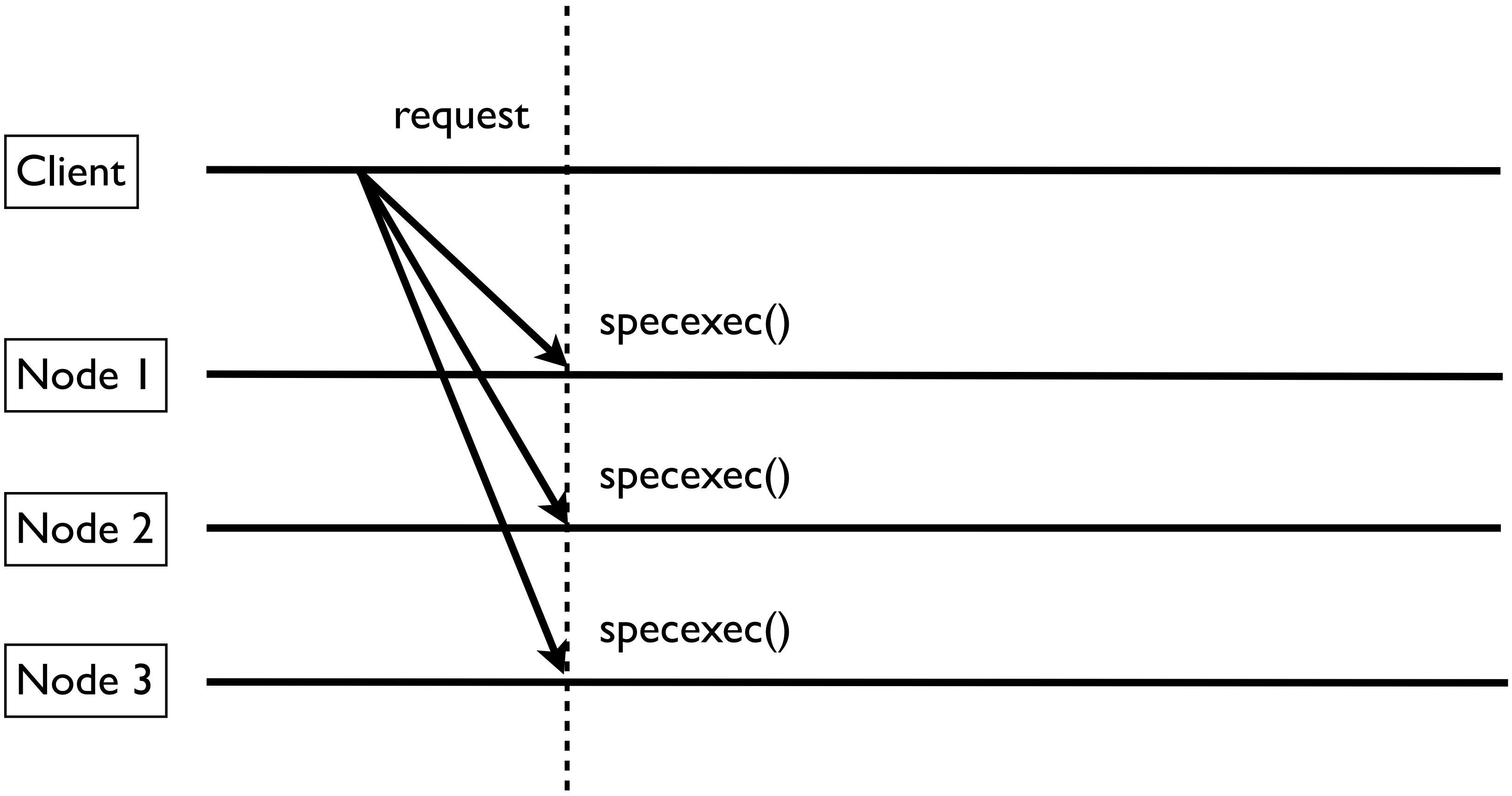
- New consensus protocol that relies on MOMs
- Leader-less protocol in the common case
- Leverages approximate synchrony:
  - If no reordering, leader is avoided
  - If there is reordering, leader-based reconciliation
  - Always safe, but more efficient with ordered multicasts

Speculative Paxos



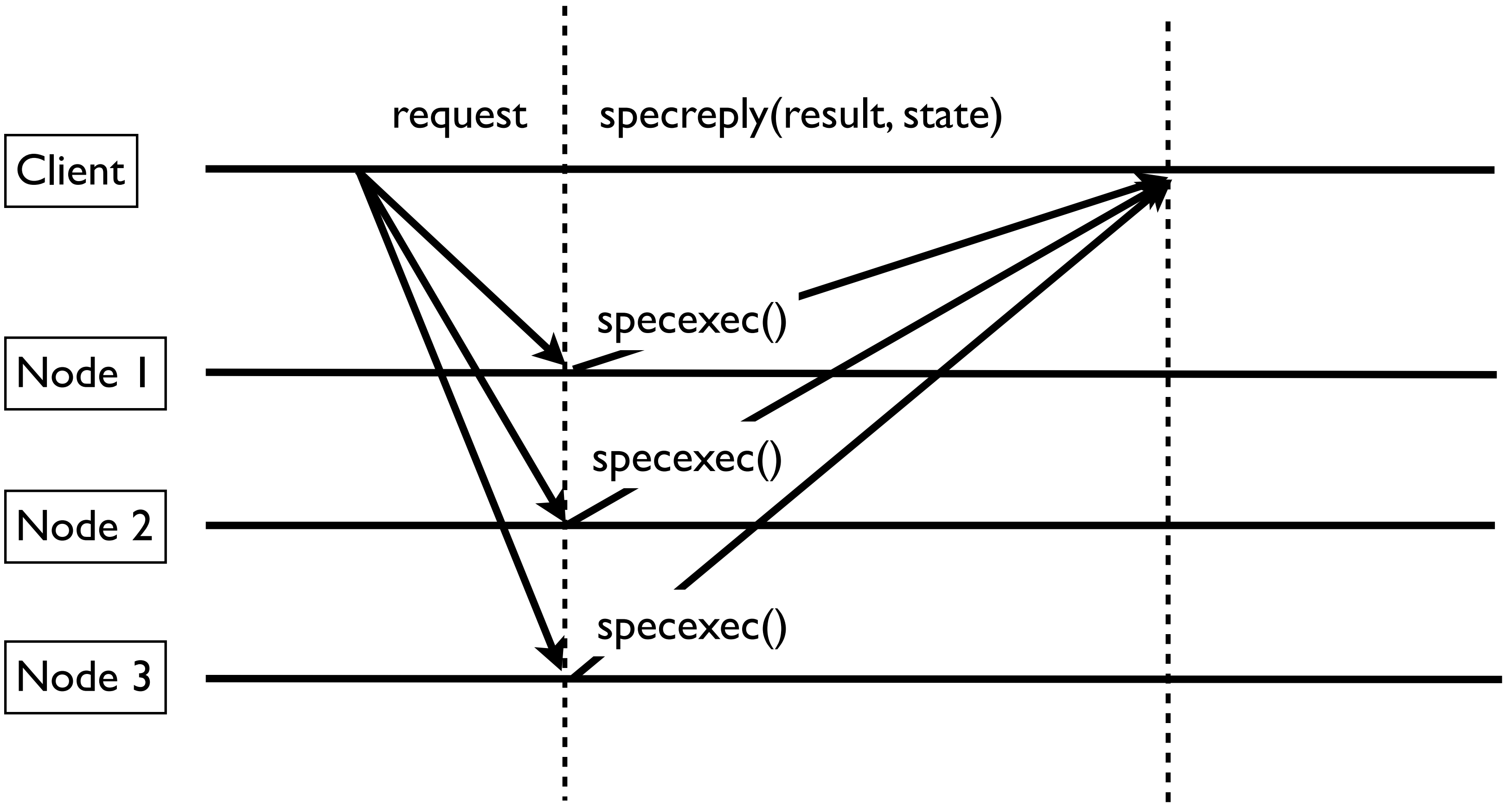
- Client sends request through a MOM to all nodes

Speculative Paxos



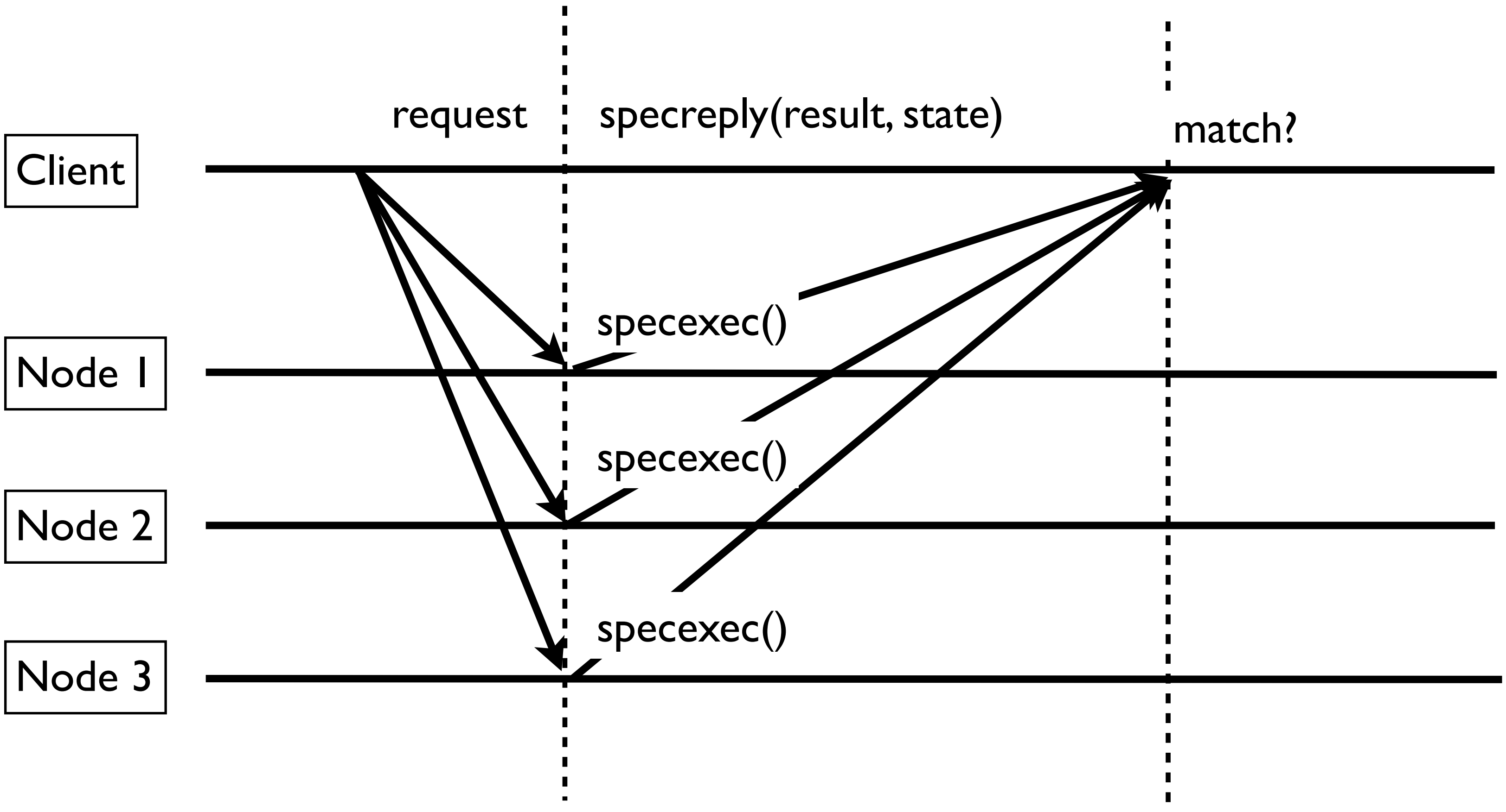
- Nodes speculatively execute assuming correct order

Speculative Paxos



- Nodes reply with result and a compressed digest of all prior commands executed by each node

Speculative Paxos



- Client checks for matching responses; operation committed if responses match from *3/2*f+1* nodes

Speculative Execution

- Only clients know immediately as to whether their requests succeeded
- Replicas periodically run *synchronization protocol* to commit speculative commands
- If there is divergence, trigger a *reconciliation protocol*
  - leader node collects speculatively executed commands
  - leader decides ordering and notifies replicas
  - replicas rollback and re-execute requests in proper order

Summary of Results

- Testbed and simulation based evaluation
- Speculative Paxos outperforms Paxos when reorder rates are low
  - *2.6x* higher throughput, *40%* lower latency
  - effective up to reorder rates of 0.5%