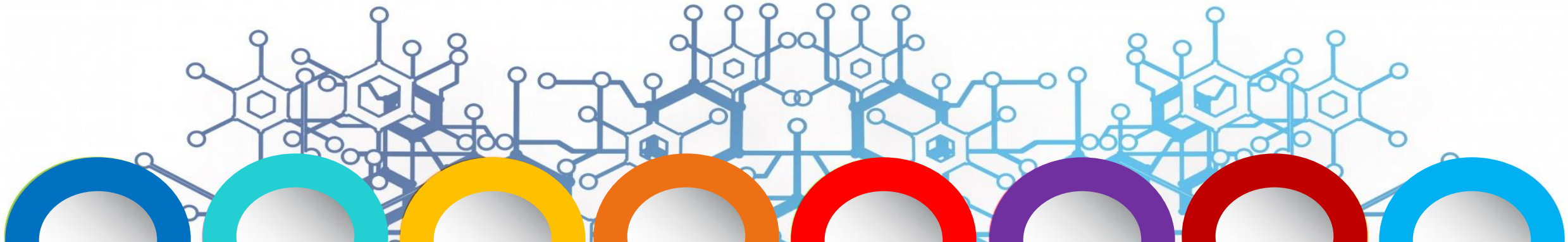
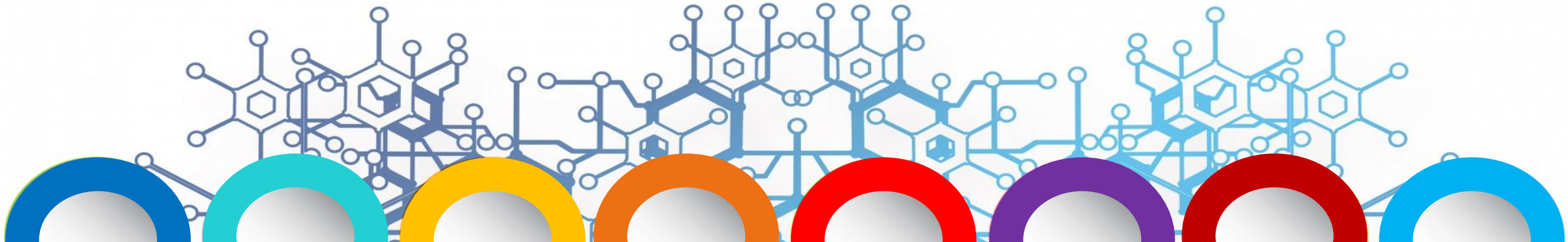
Fundamentals of Distributed Systems & The Cloud:

Network . Compute . Store



Network



Data Center Networks (DCN)

- Tens to hundreds of thousands of hosts, often closely coupled, in close proximity
  - E-business (Amazon)
  - Content-servers (YouTube, Akamai, Apple, Microsoft)
  - Search engines, data mining (Google)

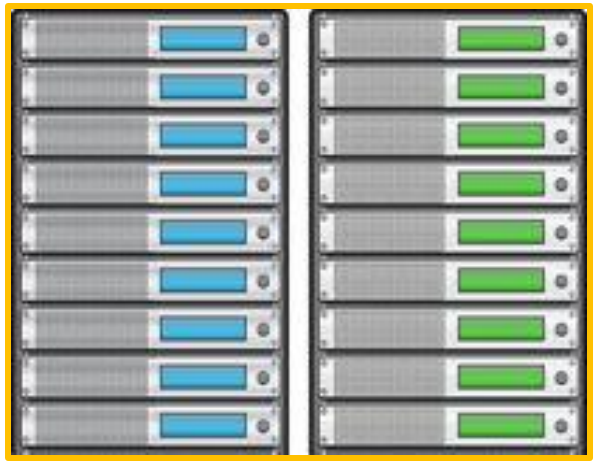
Google Douglas County, Georgia data center

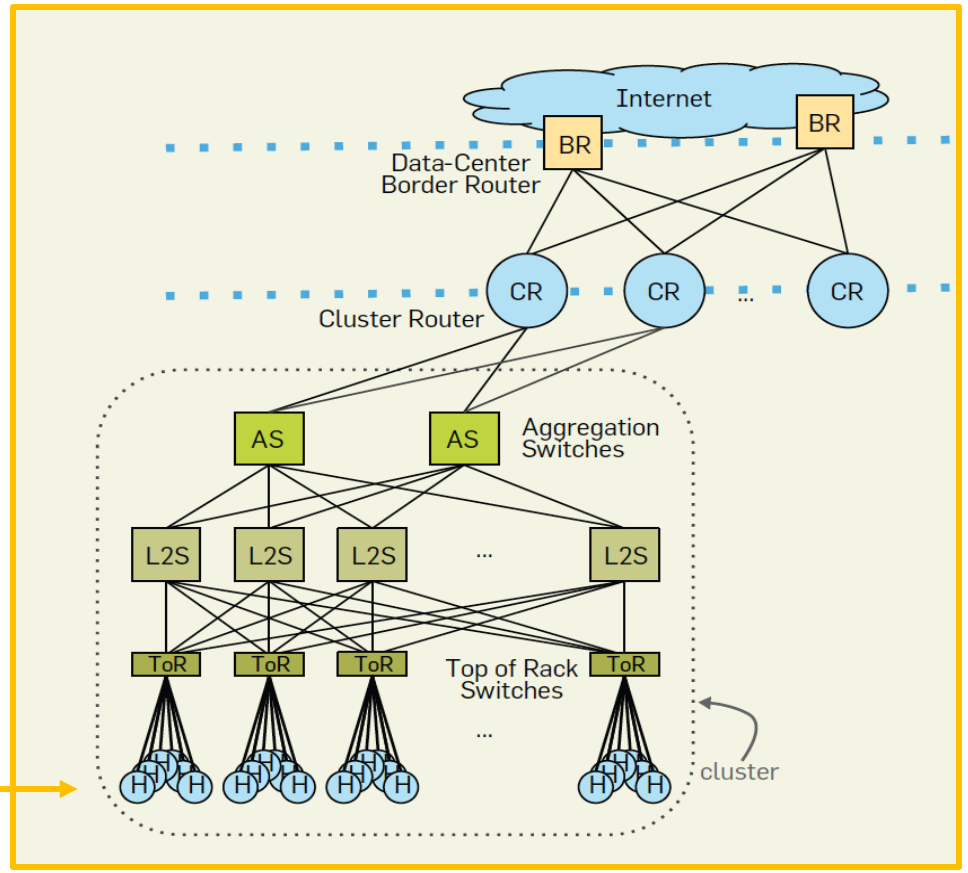
Microsoft, Chicago data center

Facebook, Mexico data center

Data Center Hosts

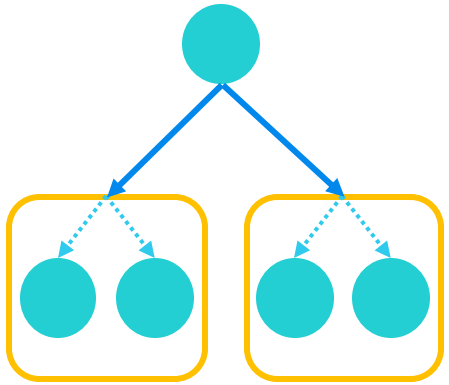
- Commodity Hosts: Blades (Pizza Box)
- Each rack ~20-40 blades
- Top Of the Rack (TOR) switch for each rack
- Each host has a connected interface to the TOR
- Each host has its own DCN IP address



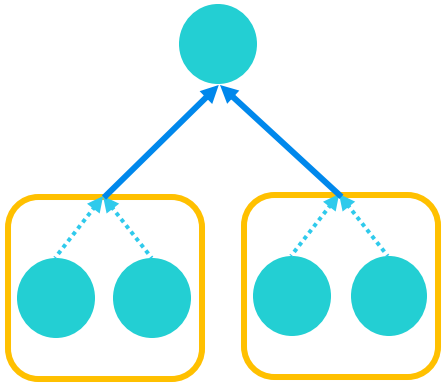


Reference: A Guided Tour of Data-Center Networking (June 2012)

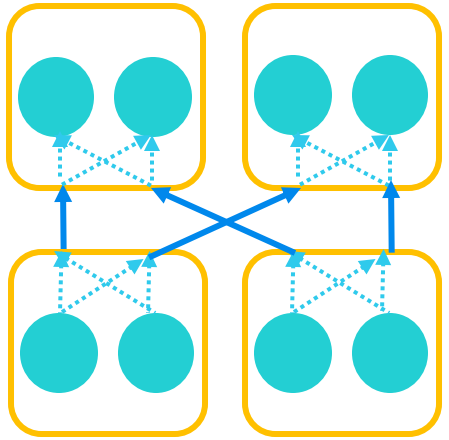
Review: Communication Patterns



Broadcast

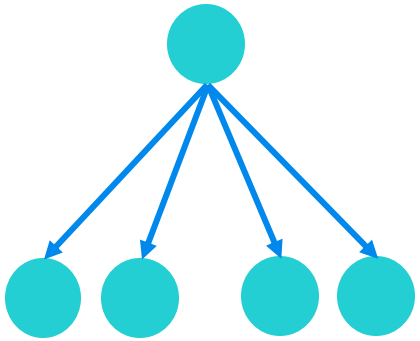


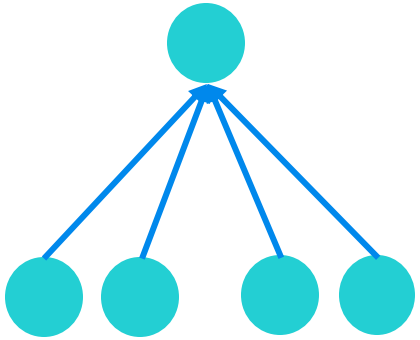
Aggregation

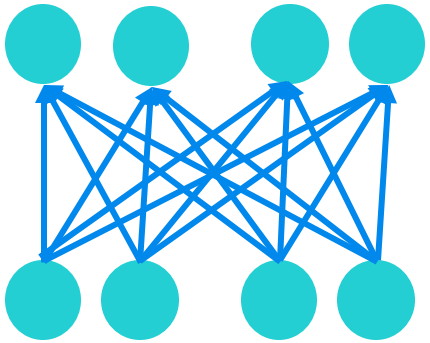


Shuffle

VM-based







Function-based

VM: Virtual Machine

Reference: Cloud Programming Simplified: A Berkeley View on Serverless Computing (2019)

Communication Patterns

Some Suggested Solutions

- Provide cloud functions with a larger number of cores similar to VM instances
  Multiple tasks can combine and share data among them before networking

- Allow developer to explicitly place cloud functions on same VM instance
  Offer distributed communication primitives for allocating cloud functions to the same VM instance

- Let applications provide a computation graph
  Enables the cloud provider to co-locate the cloud functions to minimize communication overhead

VM: *Virtual Machine*

From: *Cloud Programming Simplified: A Berkeley View on Serverless Computing (2019)*

DCN Agility: Any Service, Any Server

- Turn the servers into a single large fungible pool
  - Dynamically expand and contract service footprint as needed

- Benefits
  - Increase service developer productivity
  - Achieve high performance and reliability
  - Lower cost

***Reference**: web.mit.edu/6.829/www/currentsemester/materials/datacenter-networking.pptx (last accessed: January 2024)*

DCN: Achieving Agility

- Workload management
  - Rapidly installing a service's code on a server
  - Virtual machines, disk images, containers
- Storage Management
  - Server to access persistent data
  - Distributed filesystems (e.g., HDFS, BLOB stores)
- Network
  - Communication among servers, regardless of location in the data center

HDFS: Hadoop Distributed File System
BLOB: Binary Large Object
Reference: Presentation for VL2 Paper. web.mit.edu/6.829/ (2018 offering)

DCN: Routing & Switching

- **Ethernet switching (layer 2)**
  - ✓ Fixed IP addresses and auto-configuration (plug and play)
  - ✓ Seamless mobility, migration, and failover
  - x Broadcast limits scale (ARP)
  - x Spanning Tree Protocol

- **IP routing (layer 3)**
  - ✓ Scalability through hierarchical addressing
  - ✓ Multipath routing through equal-cost multipath
  - x More complex configuration
  - x Can not migrate without changing IP address

ARP: *Address Resolution Protocol*

Reference*: Presentation for VL2 Paper. web.mit.edu/6.829/ (2018 offering)*

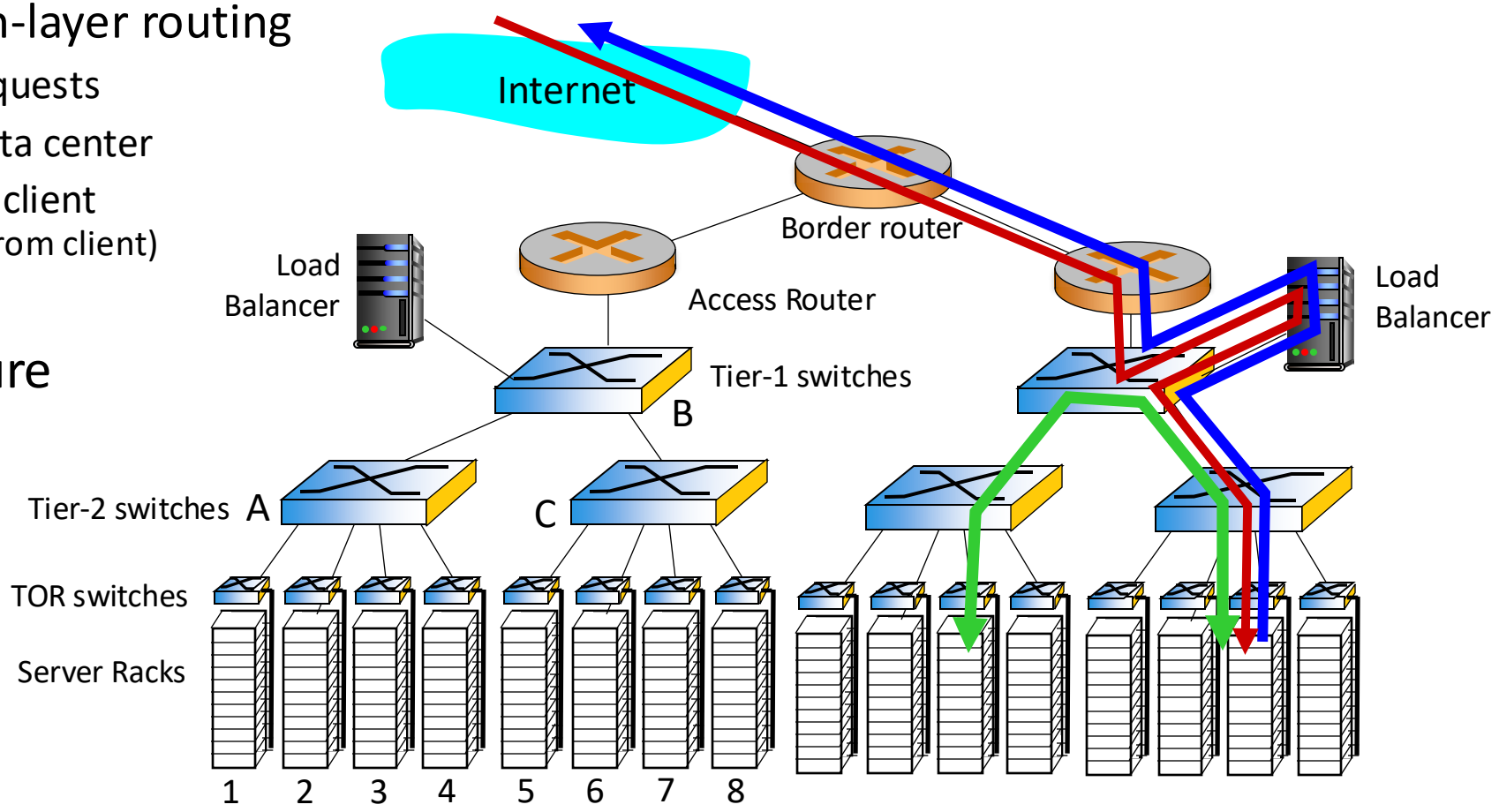
Data Center Network

- Load balancer: Application-layer routing
  - Receives external client requests
  - Directs workload within data center
  - Returns results to external client
    (Hiding data center internals from client)

- Conventional Architecture
  - **Good:** Scalable
  - **Not So Good**

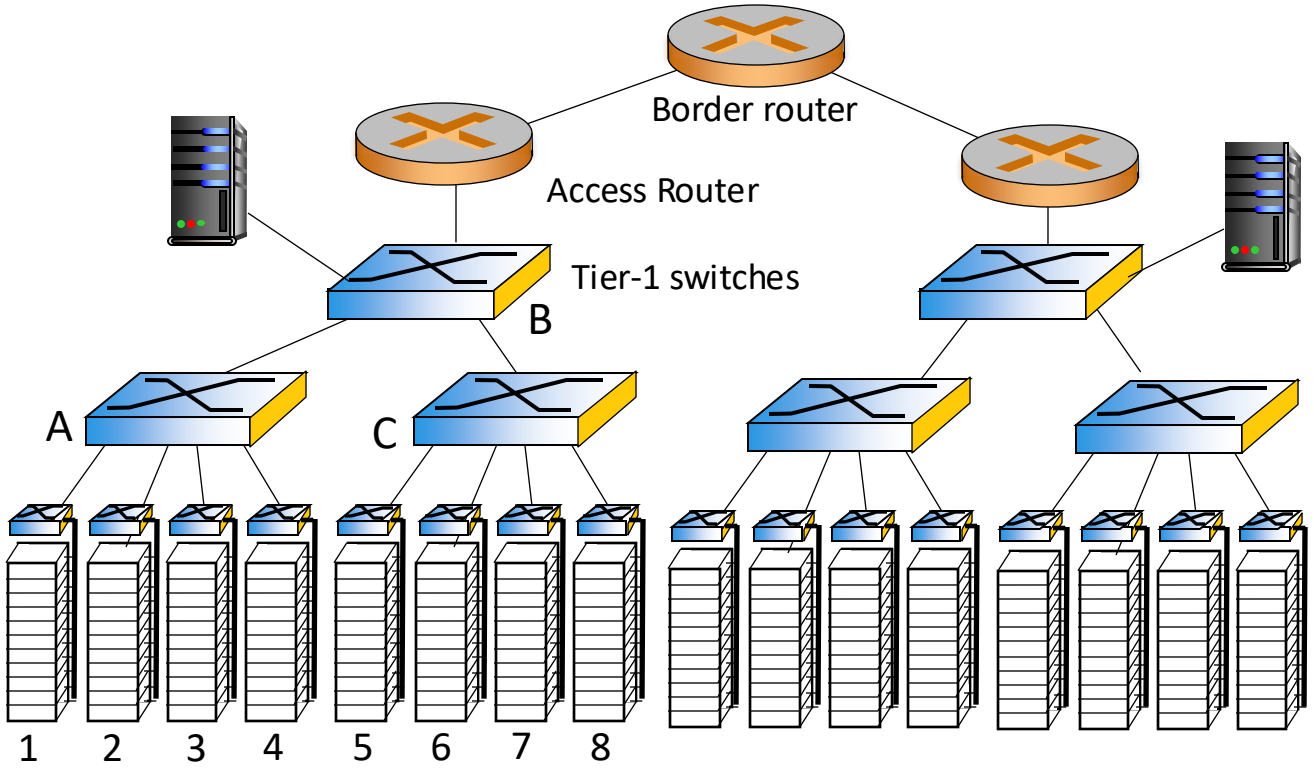
Limited host-to-host capacity

No Service Agility



Example

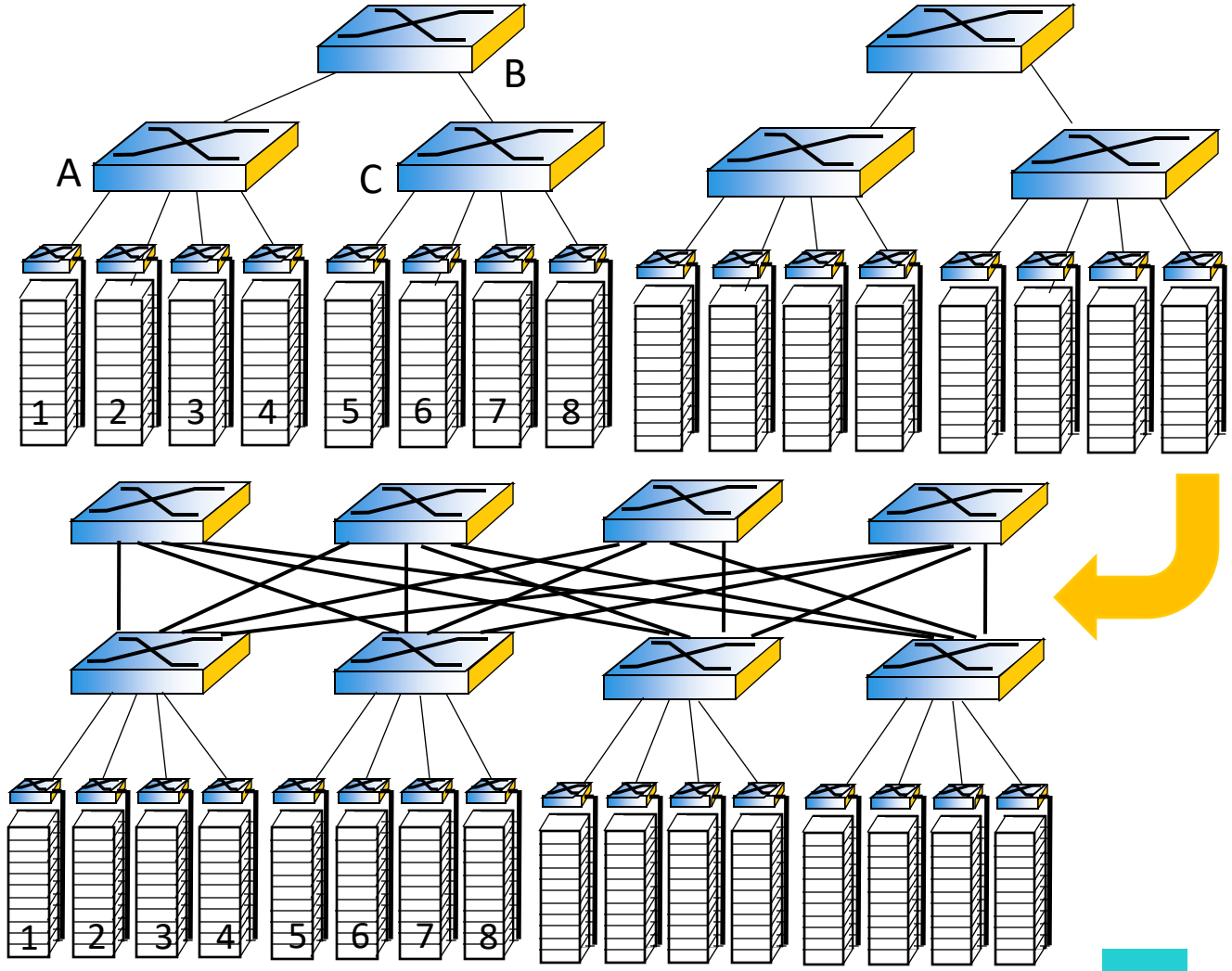
- Host-TOR: 1Gbps
- Between switches: 10Gbps
- 40 Flows
  - 10 Hosts in Rack 1 - 10 Hosts in Rack 5
  - 10 Hosts in Rack 2 - 10 hosts in Rack 6
  - 10 Hosts in Rack 3 - 10 hosts in Rack 7
  - 10 Hosts in Rack 4 - 10 hosts in Rack 8
- Performance?



Reference: Computer Networking : A Top-Down Approach. James F. Kurose, Keith W. Ross, 7th Edition, Pearson, 2017

Example

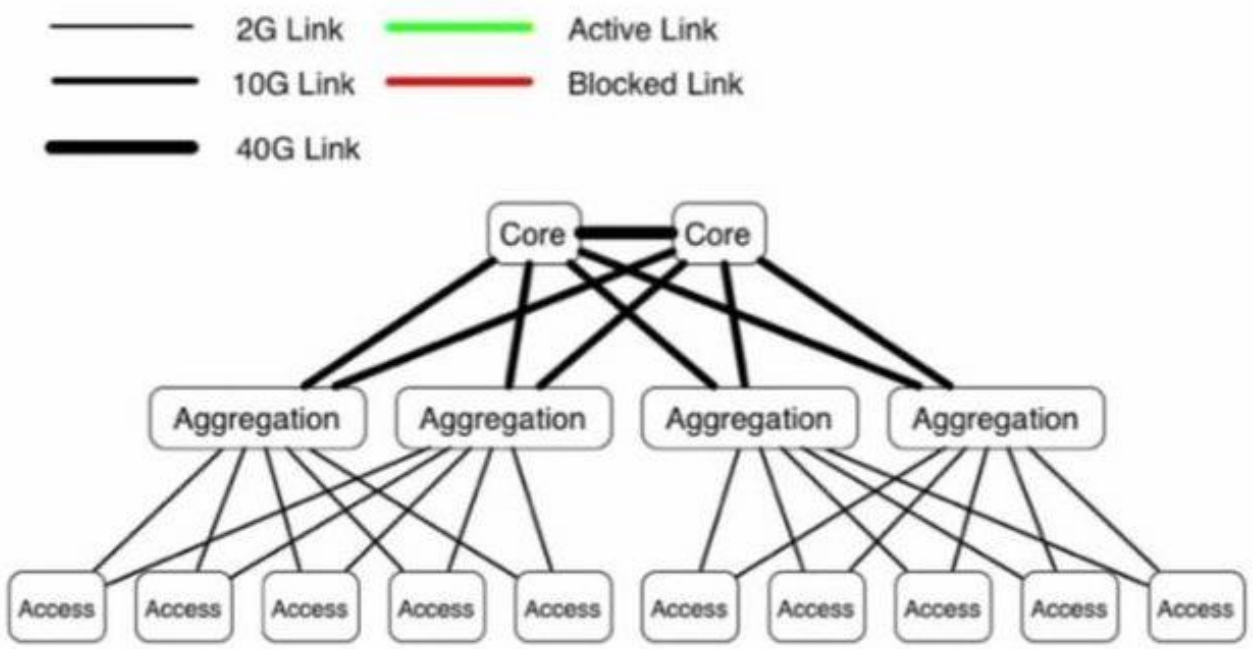
- Host-TOR: 1Gbps
- Between switches: 10Gbps
- 40 Flows
  - 10 Hosts in Rack 1 - 10 Hosts in Rack 5
  - 10 Hosts in Rack 2 - 10 hosts in Rack 6
  - 10 Hosts in Rack 3 - 10 hosts in Rack 7
  - 10 Hosts in Rack 4 - 10 hosts in Rack 8
- Performance?



Reference: Computer Networking : A Top-Down Approach. James F. Kurose, Keith W. Ross, 7th Edition, Pearson, 2017

DCN Architectures: Fat Tree

- Every layer above a layer has to support sum of the capacity of the lower layer
- Scale up!
  - Expensive for higher layers

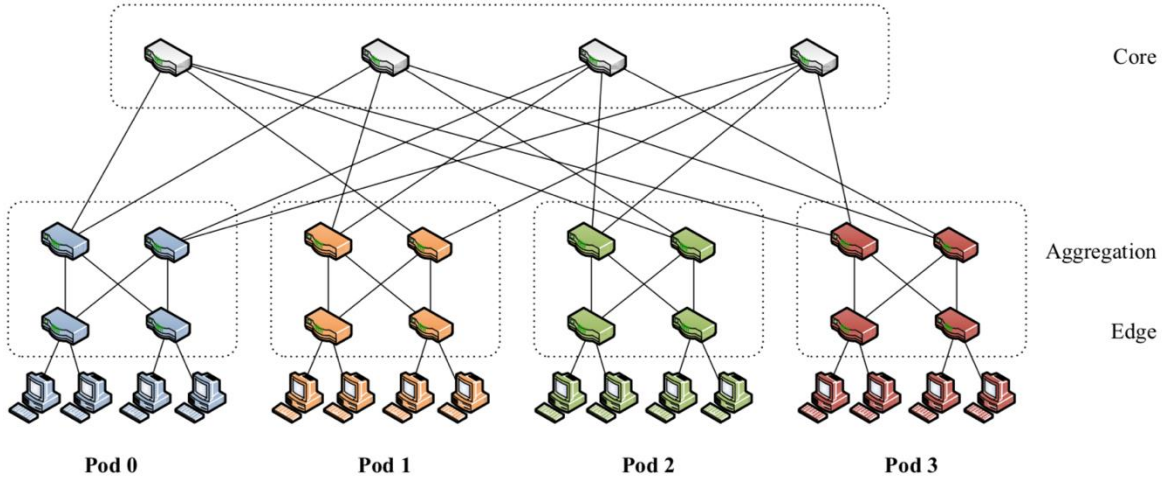


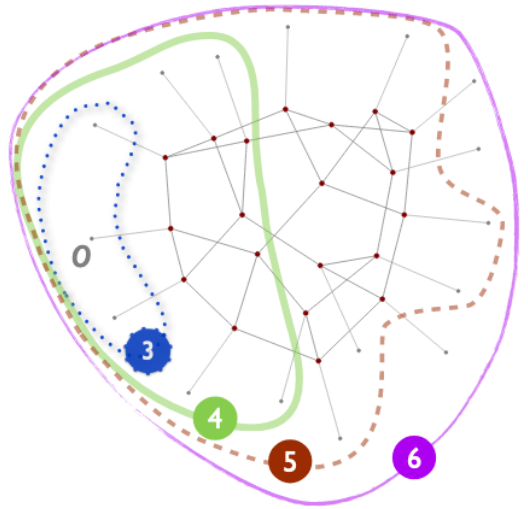
- Why Fat-Tree?
  - Identical bandwidth at bisections
  - Same aggregated bandwidth at each layer
  - All devices **can** transmit at line speed if packets are distributed properly along available paths

Reference: https://www.cs.cornell.edu/courses/cs5413/2014fa/lectures/08-fattree.pdf (by Hakim Weatherspoon)

Other DCN Architecture Proposals

- **MDC** (Modular Data Center) Network
  - Container-internal network
  - Core network to connect containers
- **BCube** (Container-based)
  - A standard 12-meter shipping container
  - A few thousand hosts
  - Graceful performance degradation over time
  - Replaceable
- **PortLand**: Plug and play large scale data center networks
- **JellyFish**
  - Network interconnect
  - Degree-bounded random graph topology among TOR switches
  - Different degrees of oversubscription





Reference: *PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric (SIGCOMM 2009)*

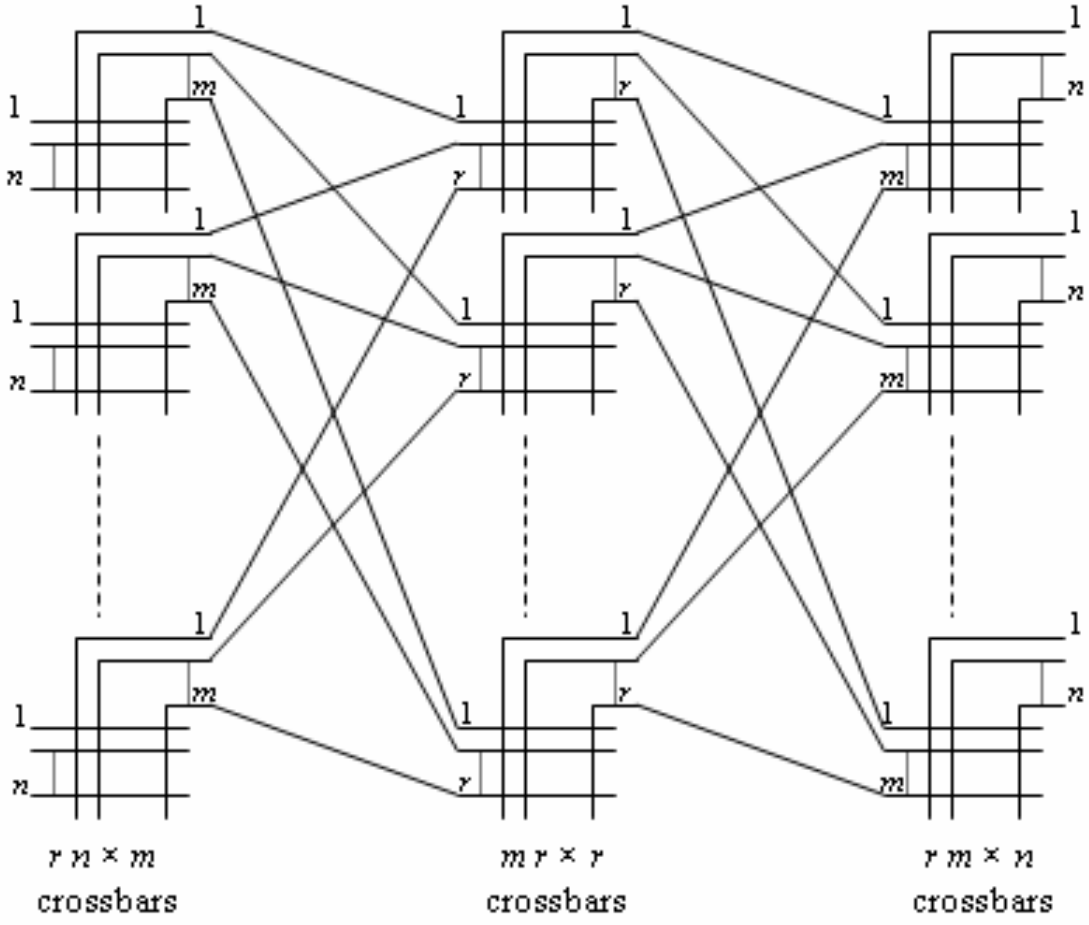
Reference: *Jellyfish: Networking Data Centers Randomly (NSDI 2012)*

Reference: *Computer Networking : A Top-Down Approach. James F. Kurose, Keith W. Ross, 7th Edition, Pearson, 2017*

Reference & Figure: *BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers (SIGCOMM 2009)*

Clos Topology

- Historical: Telephone Circuit Switching
- Parameters
  - Stages
  - Inputs
  - Outputs
- Non-blocking properties



Picture from Wikipedia

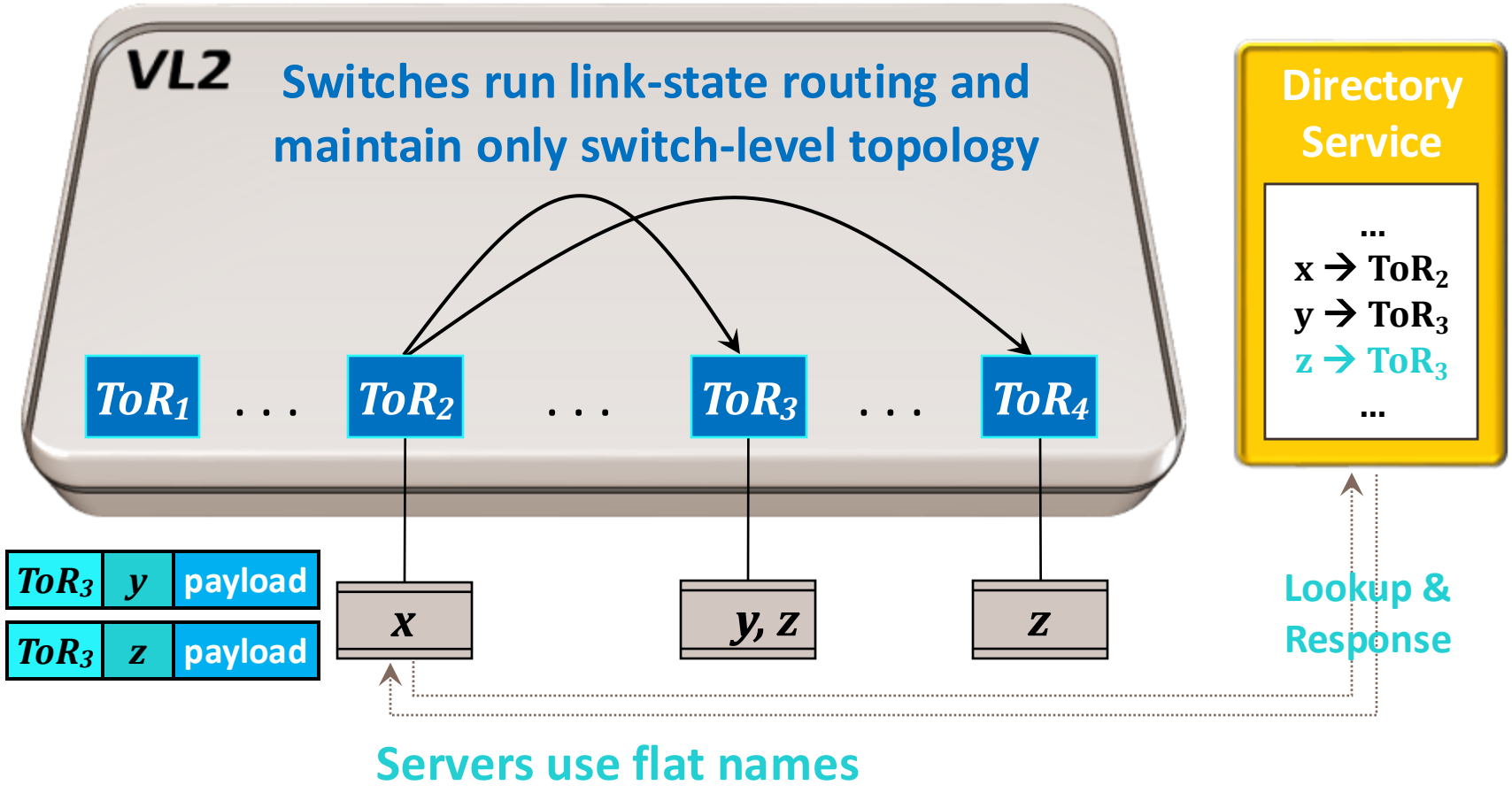
VL2: Solutions

Objective	Approach	Solution
1. Layer-2 semantics	Employ flat addressing	Name-location separation & resolution service
2. Uniform high capacity between servers	Guarantee bandwidth for hose-model traffic	Flow-based random traffic indirection (Valiant LB)
3. Performance Isolation	Enforce hose model using existing mechanisms only	TCP

TCP: *Transmission Control Protocol,* ***LB****: Load Balancing*

Reference*: web.mit.edu/6.829/www/currentsemester/materials/datacenter-networking.pptx*

VL2: Addressing & Routing

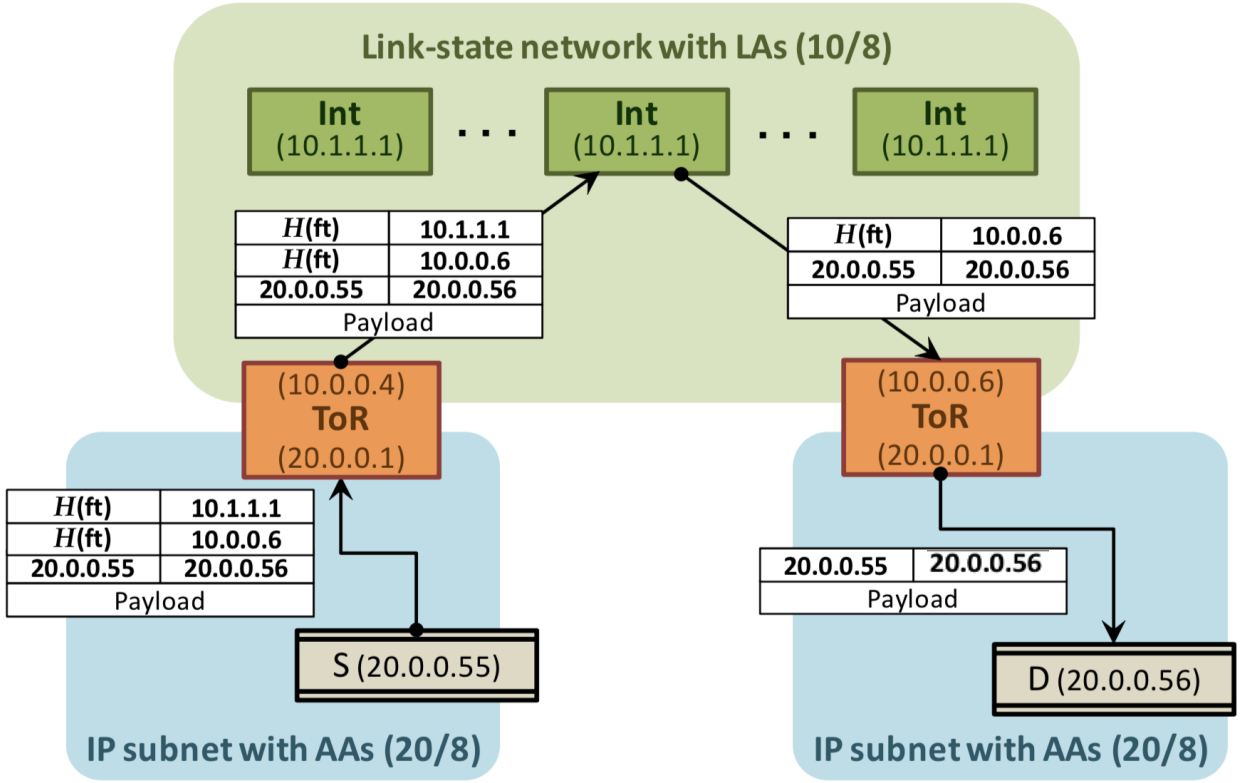


Reference: *VL2: A Scalable and Flexible Data Center Network (SIGCOMM 2009)*

Figure: *web.mit.edu/6.829/www/currentsemester/materials/datacenter-networking.pptx*

VL2: Summary

- Network built from low-cost switch ASICs arranged into a **Clos topology**
- **Valiant Load Balancing (VLB)**: Spread traffic uniformly across network paths without central coordination or traffic engineering (sending server picks random path for each flow)
- **Flat addressing**: Allow service instances to be placed anywhere in the network
- **End-system based** address resolution to scale to large server pools without introducing complexity to the network control plane



ASIC: Application Specific Integrated Switch

Reference: VL2: A Scalable and Flexible Data Center Network (SIGCOMM 2009)

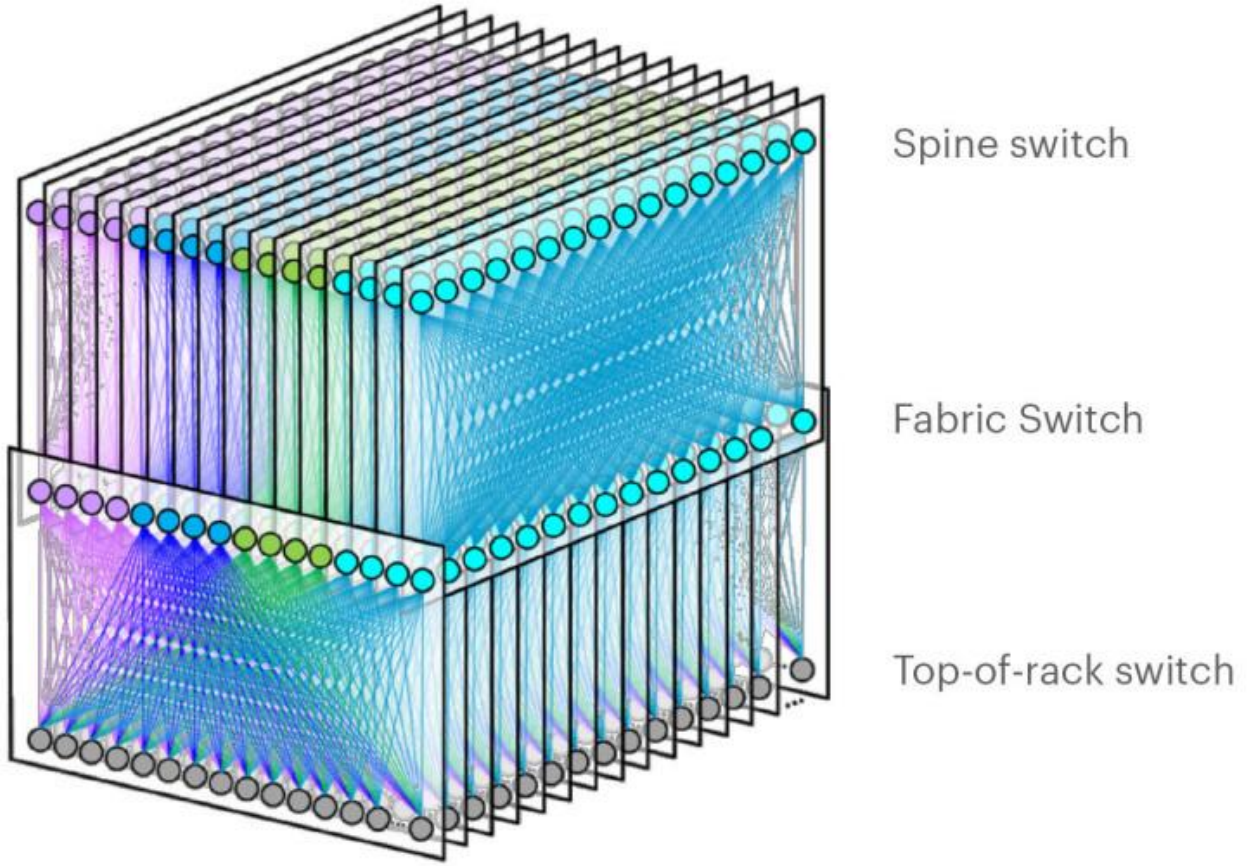
VL2: Summary

- Benefits
  - Uniform High Capacity
    - Max rate of server-to-server traffic flow limited only by available capacity of the NICs of servers involved
    - Assigning servers to a service is independent of network topology
  - Performance Isolation
    - Traffic of one service is not affected by the traffic of any other service (just as if each service was connected by a separate physical switch)
  - Layer 2 Semantics
    - Just as if servers on a LAN: IP address can connect to any port of Ethernet switch due to flat addressing
    - Virtual machines able to migrate to any server while keeping the same IP address

Reference: VL2: A Scalable and Flexible Data Center Network (SIGCOMM 2009)

Data Center Network Elements

Facebook F16 data center network topology:



https://engineering.fb.com/data-center-engineering/f16-minipack/ (posted 3/2019)

Jupiter (Google)

- Software Defined Networking
  - Logically centralized and hierarchical control plane
- Clos Topology
  - Non-blocking multistage switching topology
- Merchant Switch Silicon
  - Cost-effective, commodity general-purpose Ethernet switching

Google Cloud Blog (1): *https://cloudplatform.googleblog.com/2015/06/A-Look-Inside-Googles-Data-Center-Networks.html* (2015)

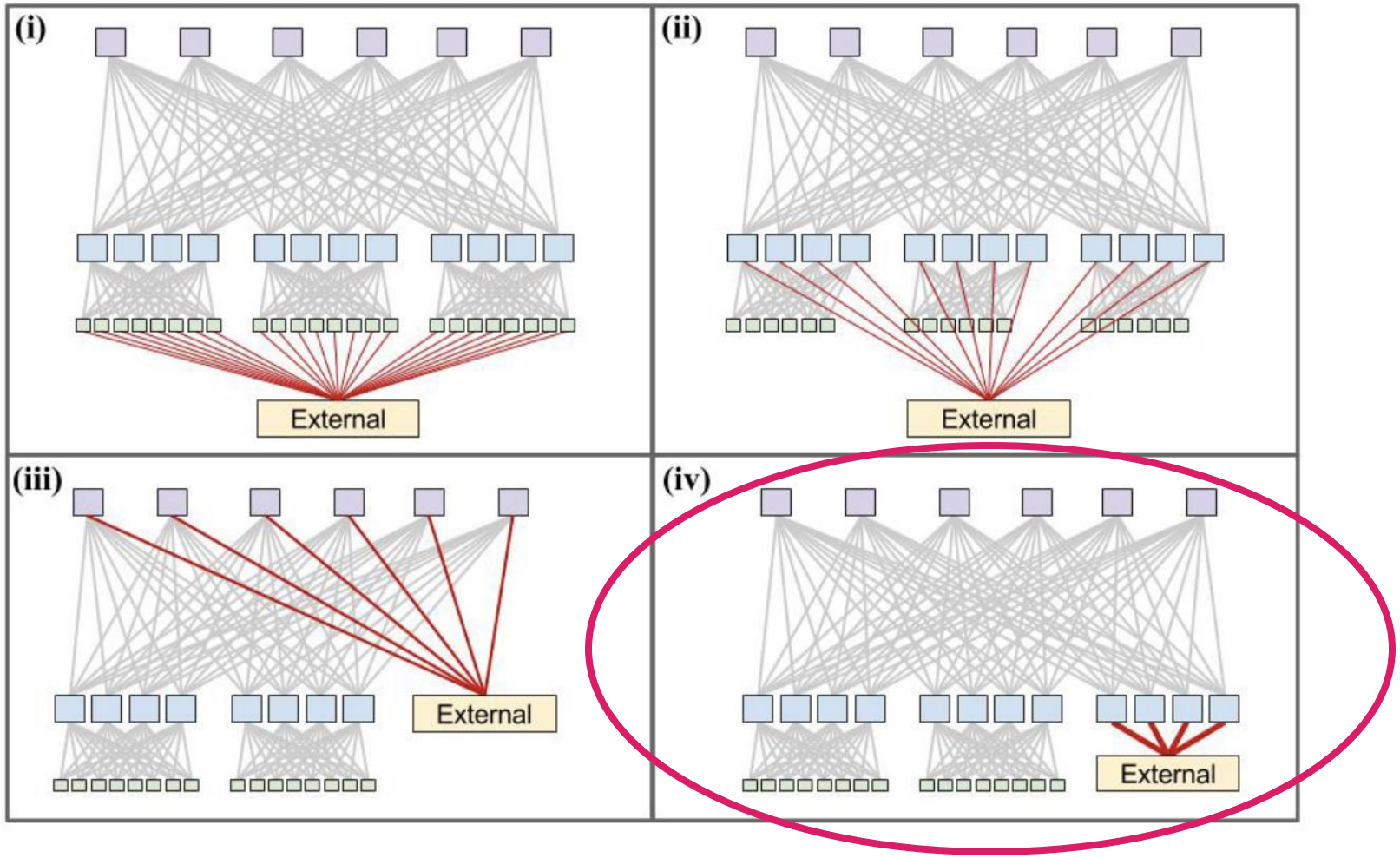
Google Cloud Blog (2): *https://cloud.google.com/blog/topics/systems/the-evolution-of-googles-jupiter-data-center-network* (2022)

***Reference**: Jupiter evolving: Reflecting on Google's data center network transformation (2022)*

Jupiter

External Connectivity

Can it handle burst external bandwidth?

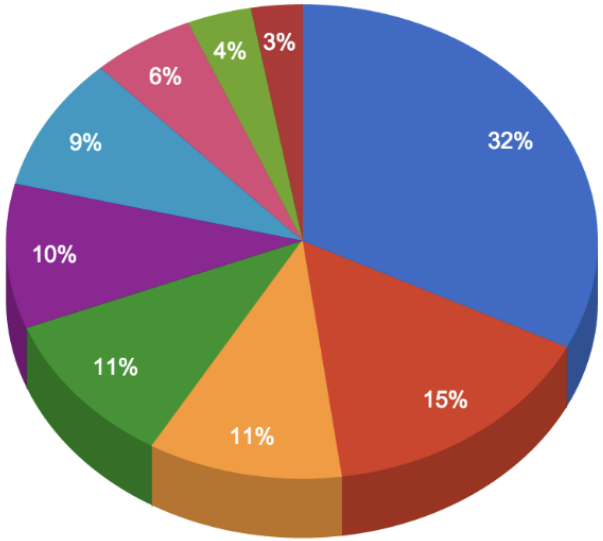


Reference: *Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network (2015)*

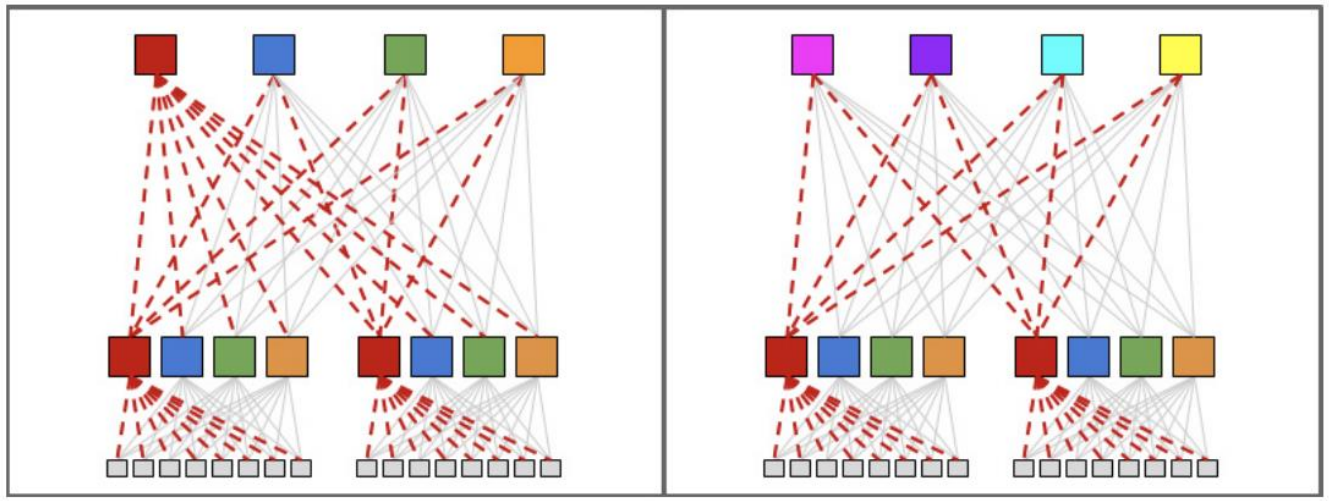
Jupiter

- Chassis linecard failure
- ToR failure
- Chassis reboot
- ToR reboot
- Planned Operation Alert
- Server issue
- CPN switch linecard failure
- Congestion alerts
- Link repair or bounce

Alerts in a Cluster over 9 months



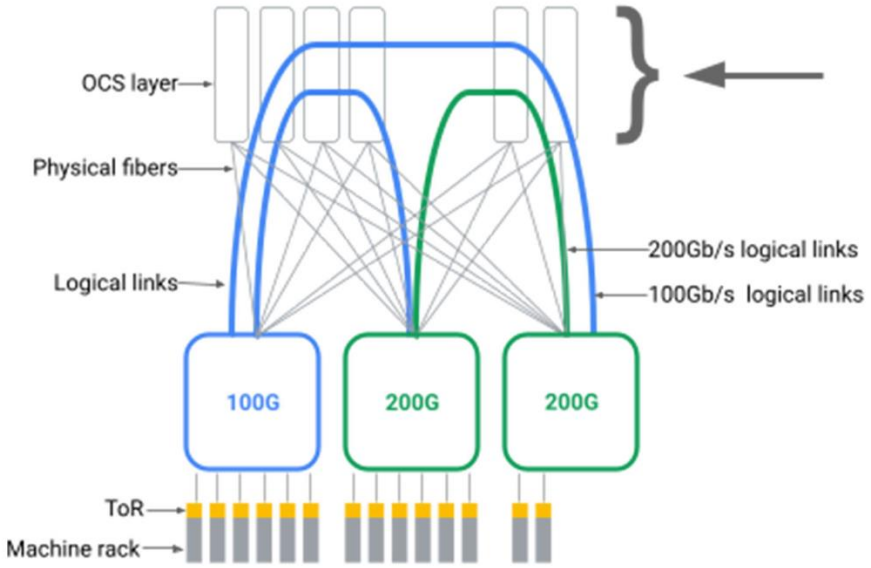
Fabric Software Redundancy Upgrades

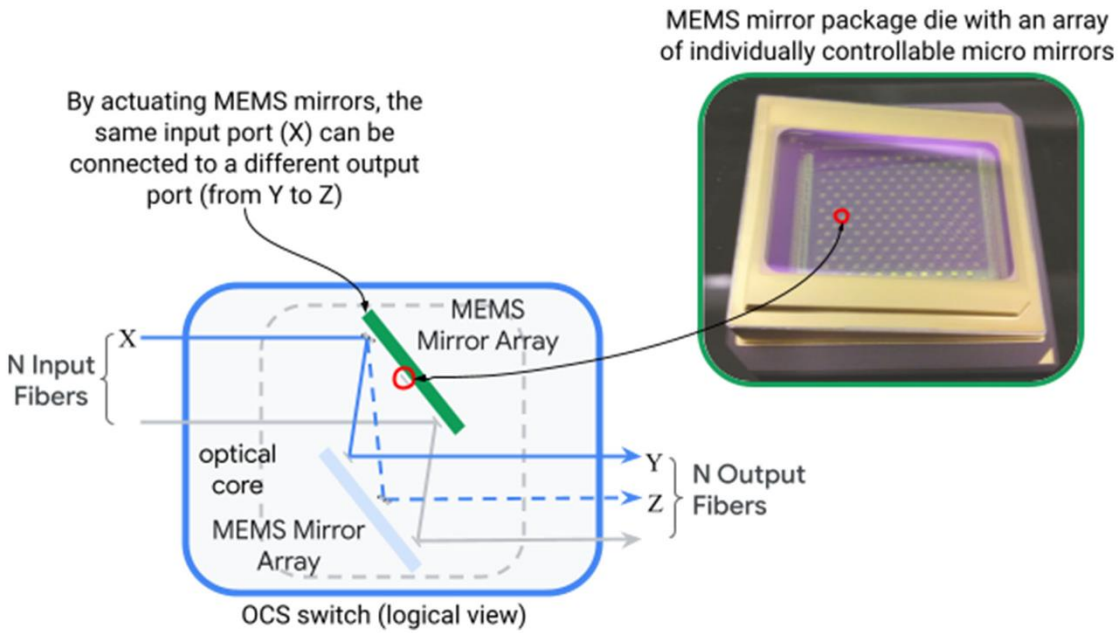


Reference: *Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network (2015)*

Jupiter Evolving

Direct-connect topology & Optical Circuit Switching (OCS)





5x Higher Speed, 30% Capex Reduction, 41% Power Reduction

Reference: Jupiter evolving: Reflecting on Google's data center network transformation (2022)

Jupiter Evolving

- Optical Circuit Switches (OCS)
- Direct **mesh-based** network topologies for higher performance, lower latency, lower cost, and lower power consumption
- Real-time topology and traffic engineering to simultaneously adapt network connectivity and pathing to match application priority and communication patterns
- Hitless network upgrades with localized add and removal of capacity

***Reference**: Jupiter evolving: Reflecting on Google's data center network transformation (2022) + Blog Post*

Cloud Resources

- Cloud Building Blocks
  - Compute
  - Store
  - Network

- Resources
  - CPU (Example: 4 Cores)
  - Memory (Example: 16GB RAM)
  - Storage (Example: 1TB HDD)
  - **Network (Example: 20 Gbps BW)**
  - I/O (Example: 1TB SATA)

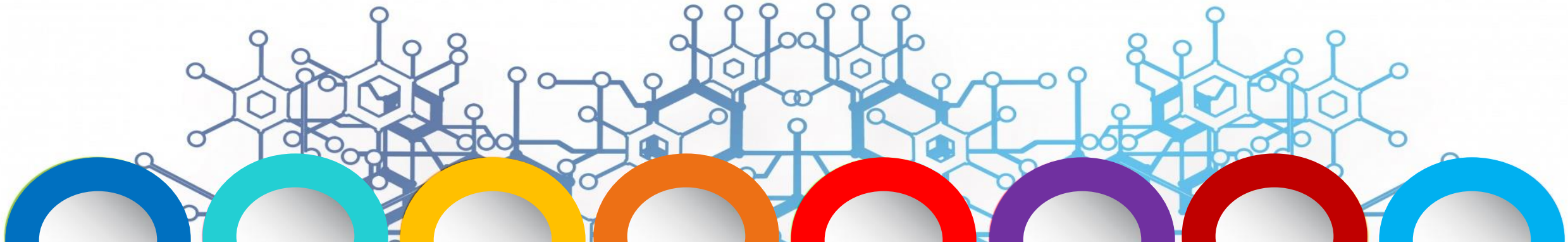
HDD: Hard Disk Drive, SATA: Serial Advanced Technology Attachment, BW: Bandwidth

Network Resources in The Cloud

- Virtual Private Cloud (Cloud VPC)
  - Connectivity for VM instances
  - Built-in Internal TCP & UDP Load Balancing
  - Proxy systems for Internal HTTP(S) Load Balancing
  - Distributes traffic
- Virtual Private Network (Cloud VPN)
  - Connect VPC network to on-premises or another cloud by a secure virtual private network
- Cloud NAT
  - Software-defined **network address translation** support
- Cloud Router
  - Border Gateway Protocol (BGP) exchange of routes between Virtual Private Cloud (VPC) and peer network
- Cloud Interconnect
  - Connect VPC network to an on-premises network by high-speed physical connection

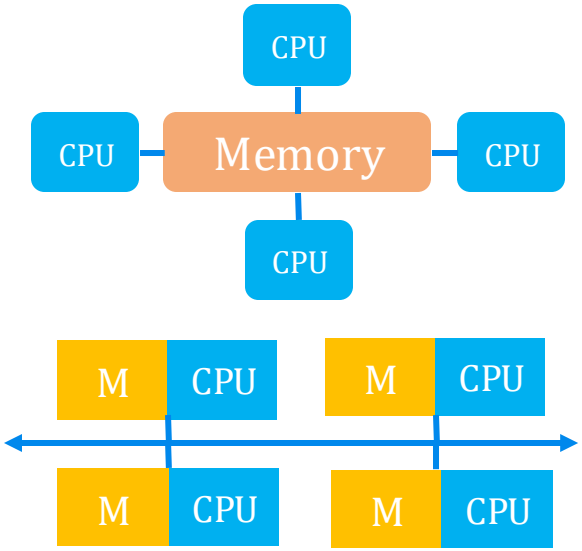
Reference: Google Cloud VPC Documentation: https://cloud.google.com/vpc/docs/overview (accessed January 2025)

Compute



Parallel Systems

- Multi-processor system
  - No Common Clock
  - **Access to Shared Memory (Unified Memory Access: UMA)**
- Array processors
  - **Common Clock**
  - No Shared Memory
- Multi-computer parallel system
  - No Common Clock
  - No Shared Memory
    (Non-Unified Memory Access: NUMA)



Reminder: Characteristic of Distributed Systems

- **Lack of Global Clock:** Action Coordination
- **No Shared Memory:** Communication (message passing)
- **Geographical Separation:** Network
- **Autonomy & Heterogeneity:** Communication (message passing)

Reference: Distributed Computing: Principles, Algorithms, and Systems, Kshemkalyani & Singhal (2008)

Distributed Program

- **Coupling:** Interdependency and binding among modules
  - Local (Tight) or Distant (Loose)
  - Homogeneity (Tight) or Heterogeneity (Loose)

		Processors	
		Loose	Tight
OS & SW	Loose	*Network*	*Multi-Processor*
	Tight	*Distributed*	*Local*

- **Parallelism:** Speedup on a specific system
  - Ratio of the time $T(1)$ with a single processor to the time $T(n)$ with n processors
  - Expectation: Percentage of time spent on processing more than wait for communication and control for Distributed Program

- **Concurrency (Distributed Systems)**
  - Ratio of the number of ***local operations*** to the ***total number of operations***

- **Granularity**
  - Ratio of the amount of computation to the amount of communication
    - Coarse (More CPU instructions to communication), Fine (Fewer)

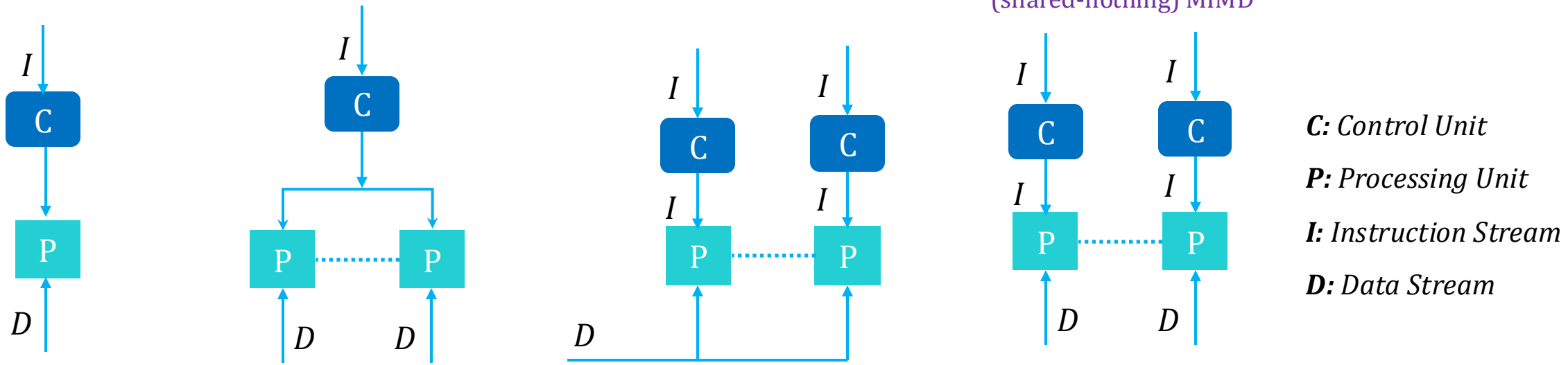
Reference: Distributed Computing: Principles, Algorithms, and Systems, Kshemkalyani & Singhal (2008)

Flynn's Taxonomy (1972)

Shared-memory MIMD
(Coordinator-Worker, Symmetric Multiprocessing)

Easy Data Partitioning

Distributed memory
(shared-nothing) MIMD



SISD
Single Instruction, Single Data

SIMD
Single Instruction, Multiple Data

MISD
Multiple Instructions, Single Data

MIMD
Multiple Instructions, Multiple Data

C: *Control Unit*

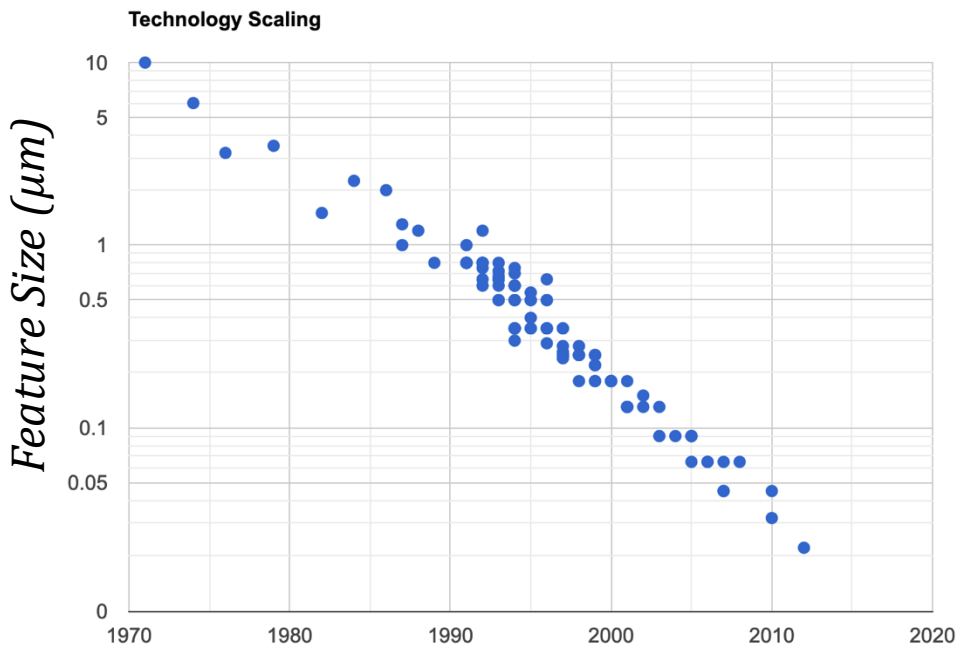
P: *Processing Unit*

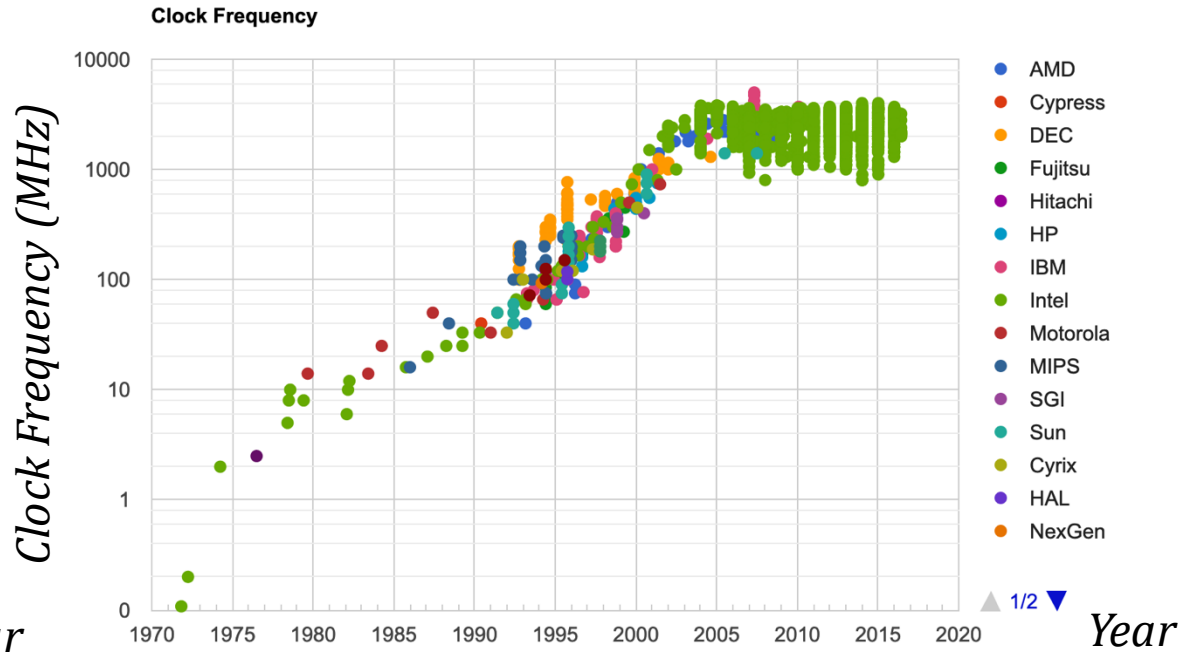
I: *Instruction Stream*

D: *Data Stream*

__Reference__: Distributed Computing: Principles, Algorithms, and Systems, Kshemkalyani & Singhal (2008)

Technology





CPUs (Central Processing Unit)	*GPUs (Graphics Processing Unit)*
Several cores	*Many (Thousands) cores*
Low latency	***High throughput***
*Good for **serial** processing*	*Good for **parallel** processing*

Other Options? *Field Programmable Gate Array (FPGA), and Application-Specific Integrated Circuit (ASIC)*

Figures: http://cpudb.stanford.edu/visualize/technology_scaling (Accessed: Jan 2023)

Algorithm & State

- The global state of a distributed computation is composed of
  - The states of the processes (in atomic sequential execution)
    State of local memory at a specific time
  - The state of communication channels
    State of messages in the channel at a specific time
    - Channel Services Models
      - Causal Ordering, FIFO, Non-FIFO

- Algorithm
  - Application Algorithm
    - Main Application Output
  - Control Algorithm (Protocol)
    - Superimposed on the main algorithm
      Global State Recording, Checkpointing, Consensus Algorithm, Termination Detection, Creating Spanning Tree, . . .

Reference: Distributed Computing: Principles, Algorithms, and Systems, Kshemkalyani & Singhal (2008)

A Model of A Distributed Program

- A distributed program is composed of a set of n asynchronous processes $p_1, p_2, \ldots, p_i, \ldots, p_n$ that communicate by message passing over a communication network

  - Each process is running on a different processor
  - Processes p_i and p_j communicate over channel C_{ij} through a message m_{ij}

  - **Execution** and **message passing** are ***asynchronous***
  - Processes do not share a ***global memory***

***Reference**: Distributed Computing: Principles, Algorithms, and Systems, Kshemkalyani & Singhal (2008)*

Process Communication

- Communication
  - Shared Memory: Common Shared Address Space
    - Communication through shared data and control variables
    - Easier than message passing
  - **Message Passing**
    - Synchrony
      - Synchronous: If ***Send()*** and ***Receive()*** handshake with each other
      - Asynchronous: Control returns back to the invoking process immediately
    - Blocking
      - Blocking: Control returns to invoking process after the processing for the primitive completes
      - Non-Blocking: Control returns back to the invoking process immediately after invocation

***Reference**: Distributed Computing: Principles, Algorithms, and Systems, Kshemkalyani & Singhal (2008)*

Algorithms

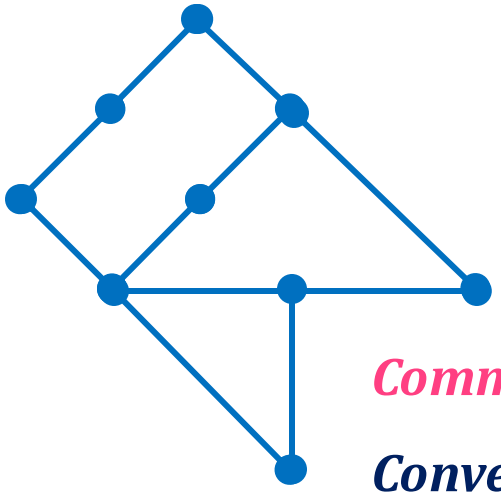
- Centralized vs. Distributed (considerable amount of work by a few processes)
- Symmetric vs. Asymmetric (same or different logical functions)
- Deterministic vs. Non-Deterministic (communication, repeated execution result)
- Synchronous vs. Asynchronous (upper bound on clock, execution, communication drift)
- Online vs. Offline (data availability)

- Other
  - Wait-free (bound on steps irrespective of failures)
  - Execution Inhibition (freezing)
  - Failure Mode: Fail-stop, Crash, Receive Omission, Send Omission, General Omission, Byzantine or Malicious Failure (w/wo authentication)
  - Uniform (scalability transparency: the number of processes is not a parameter of code)
  - Adaptive (participation-dependent)

Reference: Distributed Computing: Principles, Algorithms, and Systems, Kshemkalyani & Singhal (2008)

Examples

- Spanning Tree *Limiting Broadcast Domain*
  - Synchronous, Single-initiator
  - Asynchronous
    - Single-initiator
    - Concurrent-initiator
- Shortest Path *Distributed Routing*
  - Single source shortest path: [Asynchronous] **Bellman–Ford**
  - All sources shortest paths: [Asynchronous distributed] Floyd–Warshall
- Weighted Spanning Tree *Routing*
- Maximal Independent Set *Hypergraphs* *Wireless broadcast*



Communication

Convergence

Termination

Complexity

Reference: Distributed Computing: Principles, Algorithms, and Systems, Kshemkalyani & Singhal (2008)

Process Execution

- Synchronous
  - All processes work and communicate in lock-step with clock synchrony (bounded clock drift)
  - Synchronous execution involves degrees of blocking and delaying
  - Limited State Space
  - Easier to design and handle

- Asynchronous
  - No upper bound on messaging and execution clock drift
  - Could be non-Blocking
  - Higher Parallelism
    - Likely to have higher performance
  - Difficult to design, verify, and implement

Reference: Distributed Computing: Principles, Algorithms, and Systems, Kshemkalyani & Singhal (2008)

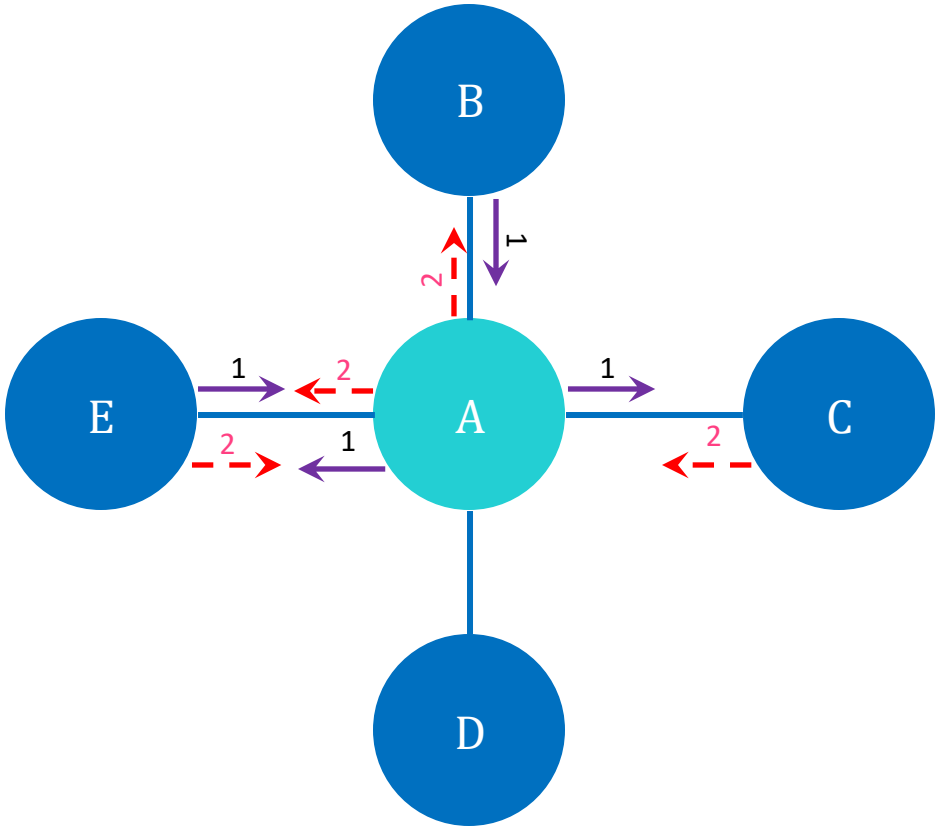
Synchronizers

- Class of transformation algorithms to run **synchronous algorithms** on **asynchronous systems**
  - A mechanism that indicates to each process when it is safe to proceed to the next round of execution of the synchronous algorithm

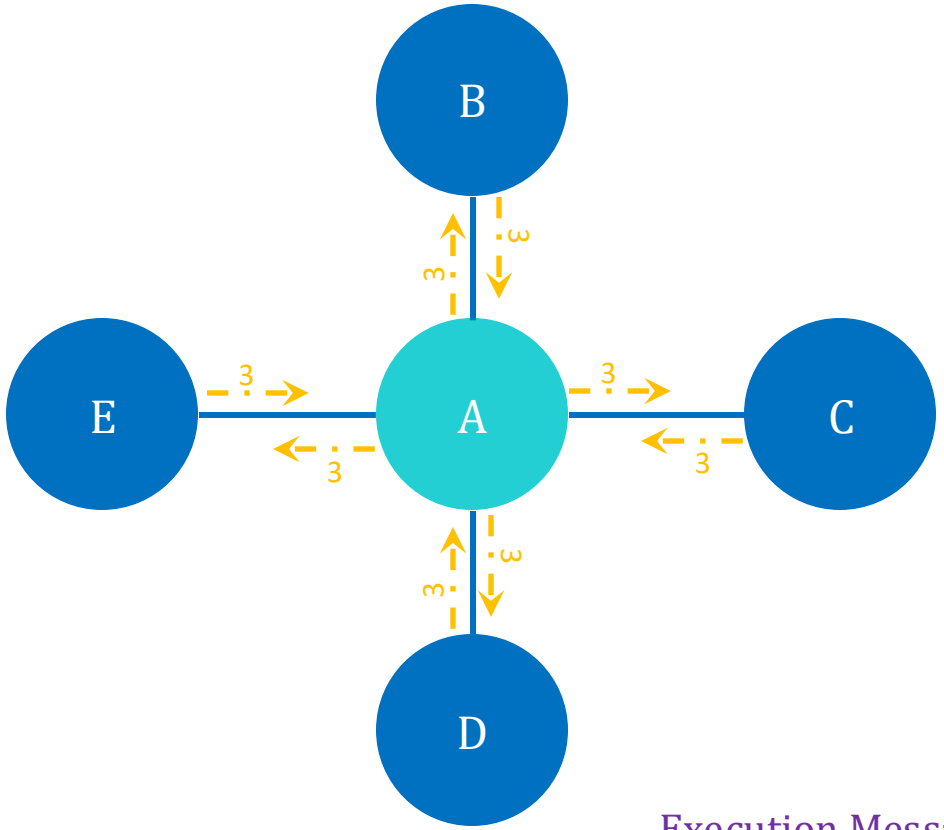
- Synchronizers
  - Simple synchronizer
  - **α** synchronizer
  - **β** synchronizer
  - **γ** synchronizer

Reference: Distributed Computing: Principles, Algorithms, and Systems, Kshemkalyani & Singhal (2008)

Example



(a)



(b)

Execution Message

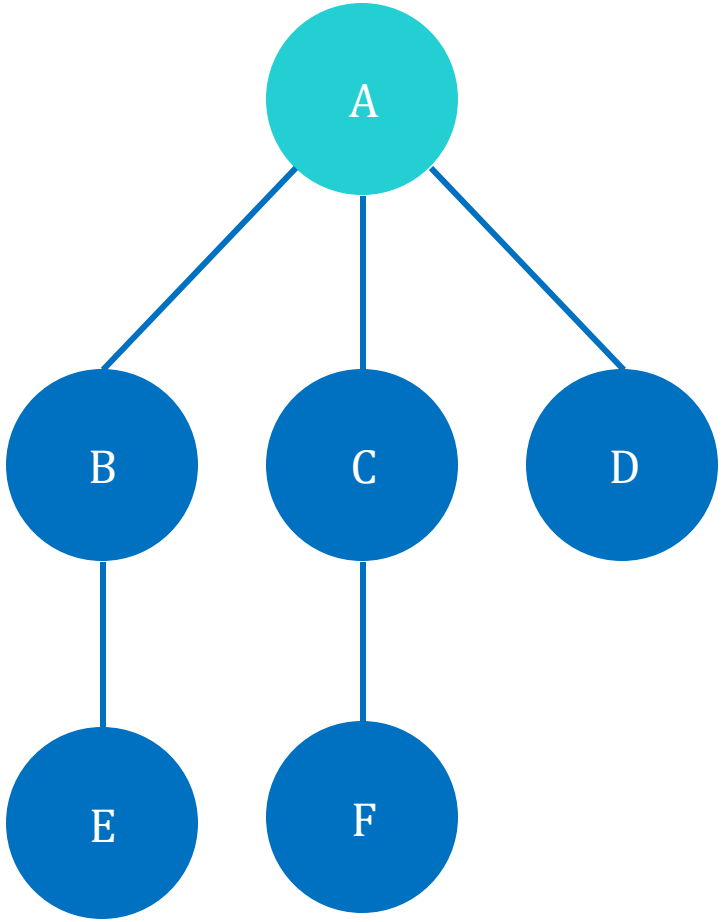
Acknowledgement

Safe

Reference: Distributed Computing: Principles, Algorithms, and Systems, Kshemkalyani & Singhal (2008)

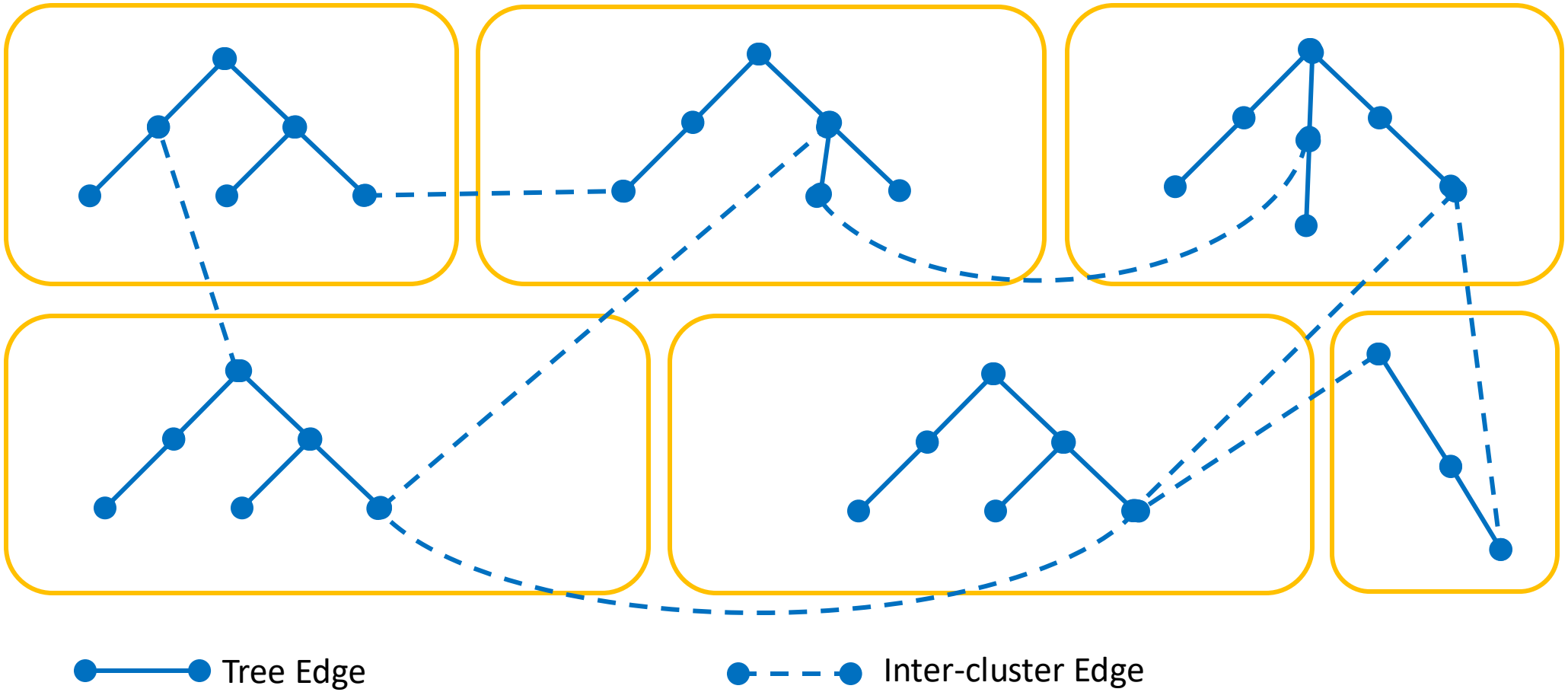
Example

- Rooted Spanning Tree
- Safe leaf nodes initiate a converge-cast
- An intermediate node converge-casts **to its parent** when all the nodes in its subtree (including itself) are safe
- **Root safe:** Received the converge-cast from all its children
  - Tree broadcast to inform all the nodes to move to the next phase



Reference: Distributed Computing: Principles, Algorithms, and Systems, Kshemkalyani & Singhal (2008)

Example

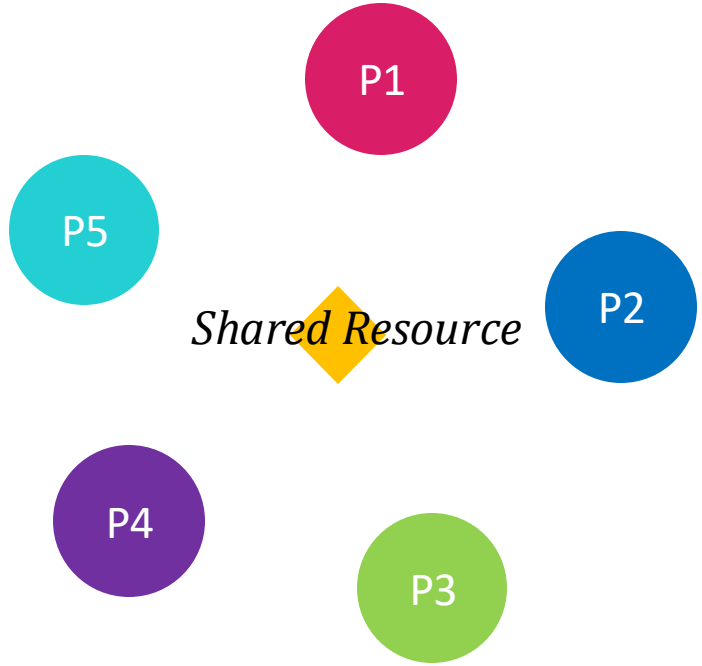


Reference: Distributed Computing: Principles, Algorithms, and Systems, Kshemkalyani & Singhal (2008)

Symmetry

- Symmetry
  - Symmetric: All processors execute the same logical functions
  - Asymmetric: Processors execute the logically different functions

- Breaking the Symmetry
  - Why? No deterministic algorithm to solve the problem
  - Caution
    - Deadlocks
    - Lockouts



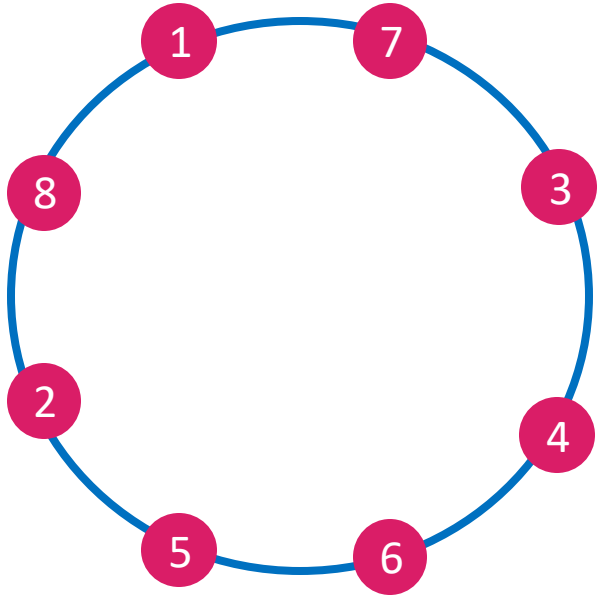
Reference: Distributed Computing: Principles, Algorithms, and Systems, Kshemkalyani & Singhal (2008)

Leader Election

- Asymmetry in distributed execution
  - Example: RAFT for Replicated State Machines (Leader, Follower, Candidate)

- Need for a role of a leader process
  - Example: Role of root node in tree algorithms
    Role of coordinator among worker processes

- All processes agree on the leader process
  - Examples:
    - Ring Topology (LCR): Forwarding to right neighbor, greater node ID wins
    - RAFT: Voting for a majority win of a leader in the election, heartbeat messages to establish authority



***Reference**: Distributed Computing: Principles, Algorithms, and Systems, Kshemkalyani & Singhal (2008), In Search of an Understandable Consensus Algorithm (2014)*

Reminder: Cloud Resources

- Instance Types
  - General Purpose
    Web servers, code repositories
  - Compute Optimized (vCPUs)
    Media Transcoding, High Performance Computing (HPC), Scientific Modeling, Machine Learning (ML)
  - Memory Optimized
    Big Data Processing
  - Accelerated Computing (includes GPUs)
    Graphic processing, floating point computations, data pattern matching
  - Storage Optimized (IOPS optimized)
    High sequential read and write access to very large data sets
  - HPC Optimized
    Deep learning workloads

***Reference**: Amazon Instance offering at https://aws.amazon.com/ec2/instance-types/ (Accessed Jan 2024)*

Acknowledgement

The list of resources used in preparation of this slide set are provided on:

https://canvas.sfu.ca/courses/88212/pages/references

Pictures and quoted resources are mentioned in each use.