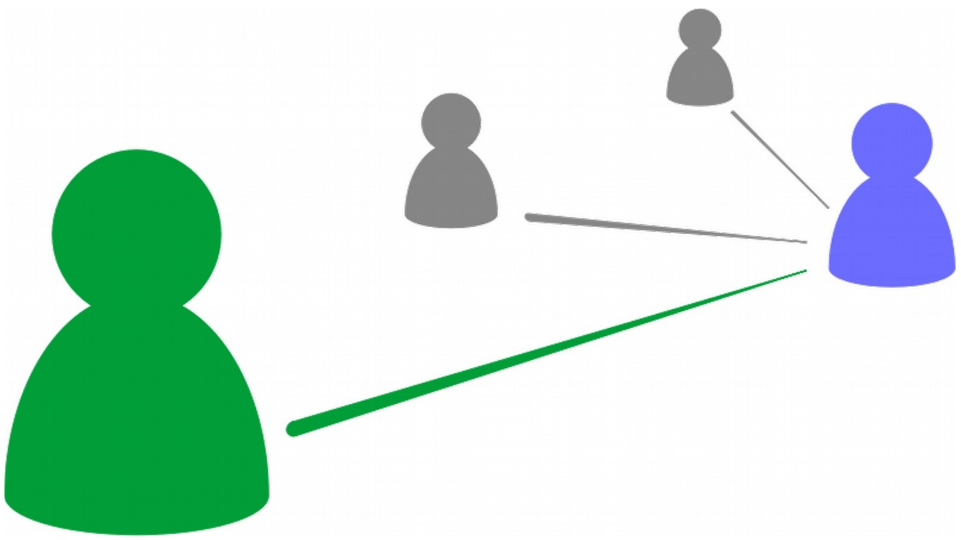
CS 470
Spring 2025

Mike Lam, Professor



Networks

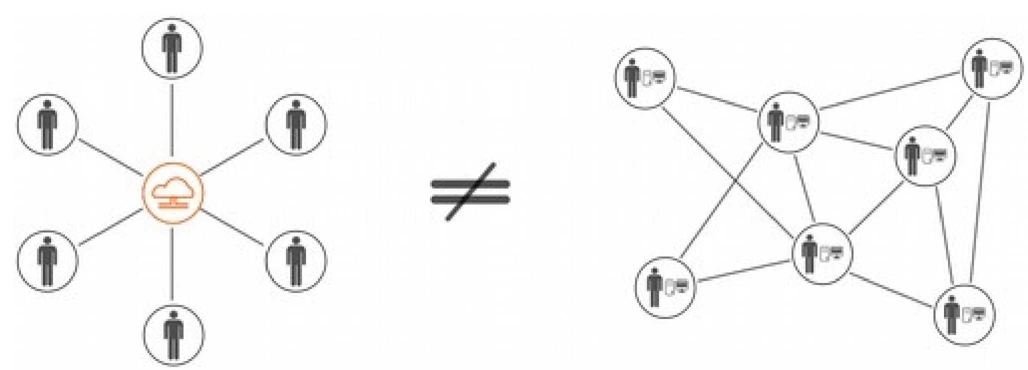
Content taken from IPP 2.3.3 and the following:

"Distributed Systems: Principles and Paradigms" by Andrew S. Tanenbaum and Maarten Van Steen (Chapter 4)

Various online sources (including wikipedia.org and openclipart.org)

Overview

- Topologies – how a network is arranged (hardware)
- Routing – how traffic navigates a network (hardware and software)
- Protocols – how machines communicate (software, low-level)
- IPC paradigms – how processes communicate (software, high-level)

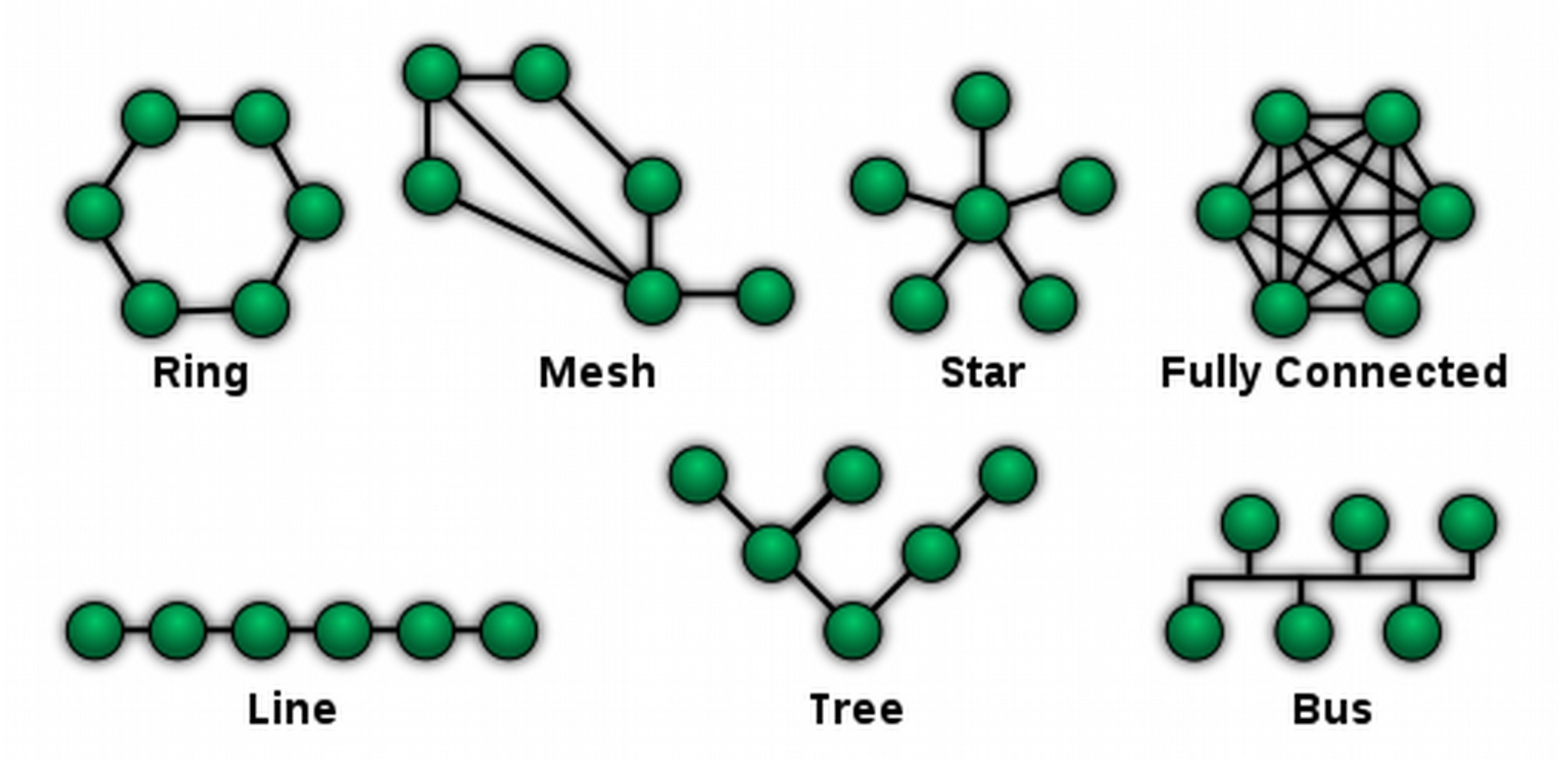


Part 1

- Topologies – how a network is arranged (hardware)

Network topologies

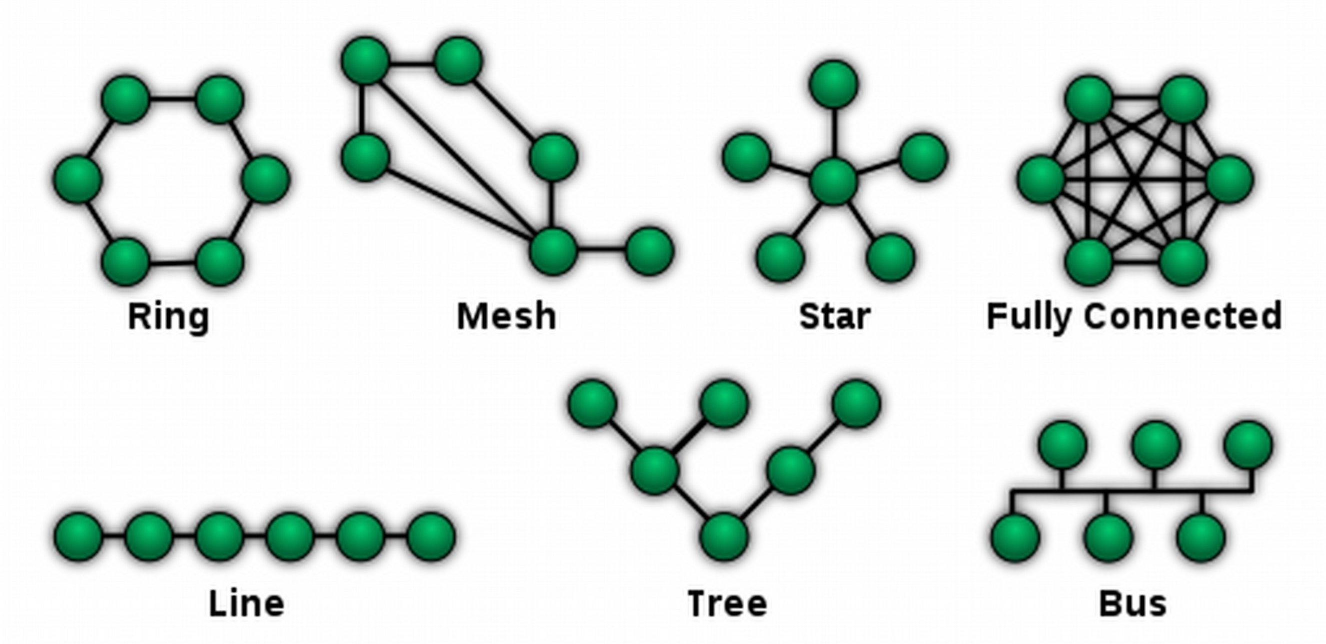
- A network topology is an arrangement of components or nodes in a system and their connections (e.g., a graph)



from `https://en.wikipedia.org/wiki/Network_topology`

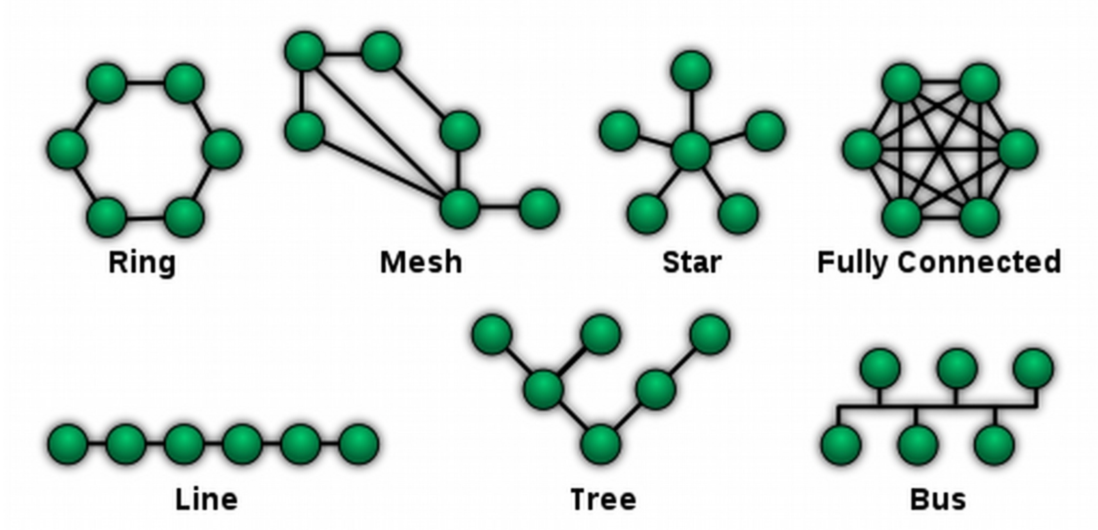
Network topologies

- In which topology is every node connected to exactly two other nodes?
  - A. Ring
  - B. Star
  - C. Fully connected
  - D. Line
  - E. Tree



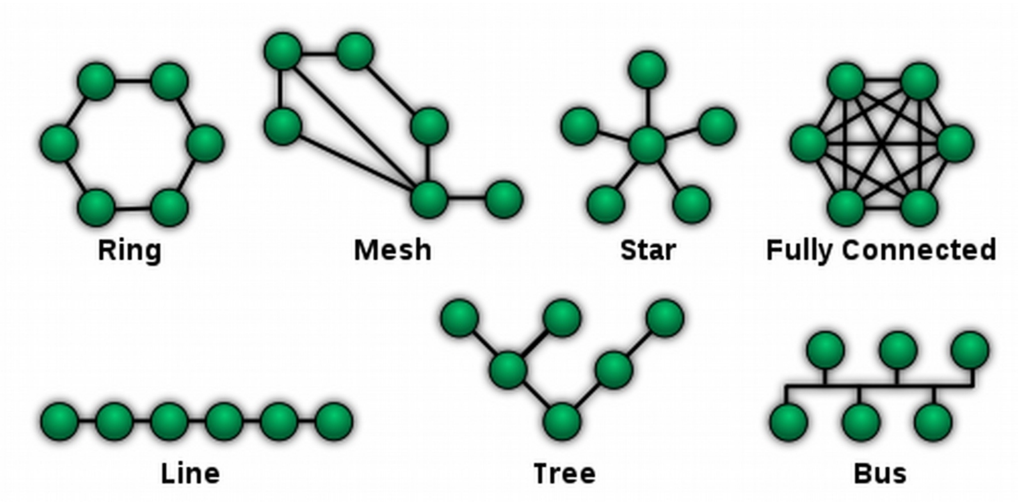
Network topologies

- A network topology is an arrangement of components or nodes in a system and their connections (e.g., a graph)
  - Ring, star, line, and tree allow simultaneous connections but disallow some pairs of point-to-point communication
  - Fully connected and bus allow direct any-to-any communication but do not scale well



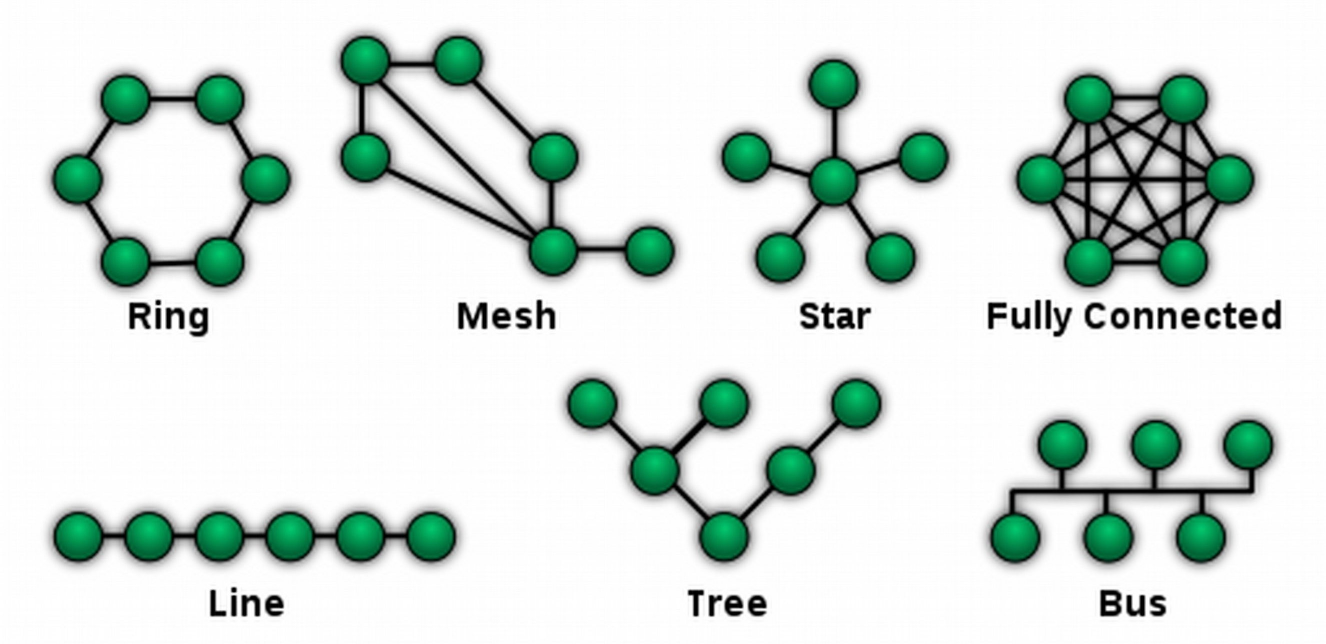
Evaluating topologies

- Full network:
  - Diameter: maximum number of hops between nodes on a network
  - Total number of links required (every link costs money!)
- Between two nodes:
  - Bandwidth: maximum rate at which data can be transmitted
    - Throughput: measured rate of actual data transmission (usually less than theoretical maximum)
  - Latency: time between start of send and reception of first data
- **Important**: how do these metrics scale as you add nodes?



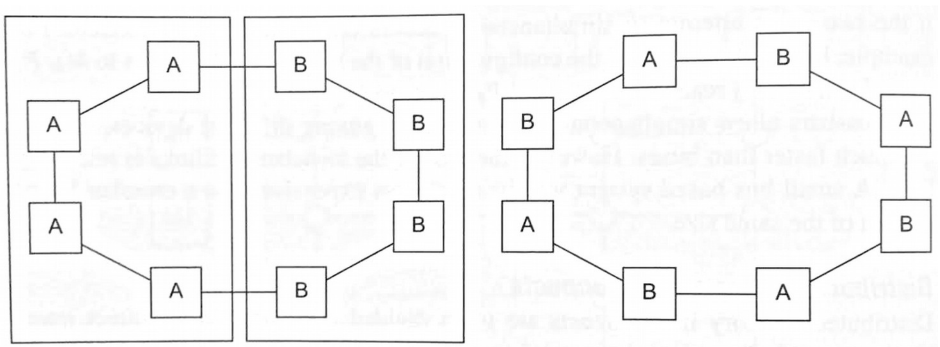
Evaluating topologies

- In which topology (or topologies) does the diameter remain unchanged as you scale up the number of nodes?
  - A. Ring
  - B. Star
  - C. Fully connected
  - D. Line
  - E. Tree



Evaluating topologies

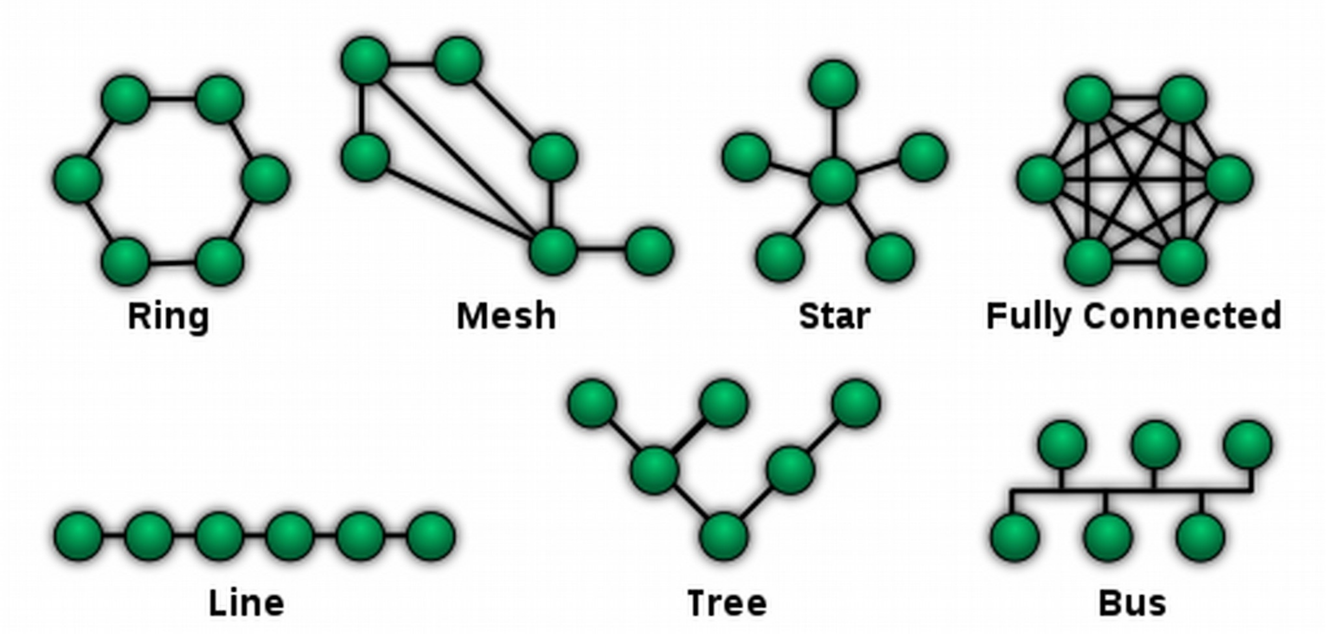
- Bisection: divide the network into two equal partitions
  - Bisection width: number of simultaneous communications between the two partitions
  - Bisection bandwidth: total data rate between the partitions
  - Typically done in such a way that minimizes bisection bandwidth
  - This represents network performance with a worst-case bottleneck



Two different bisections of a network

Evaluating topologies

- In which topology is the minimum bisection bandwidth the highest as you scale up the number of nodes?
  - A. Ring
  - B. Star
  - C. Fully connected
  - D. Line
  - E. Tree

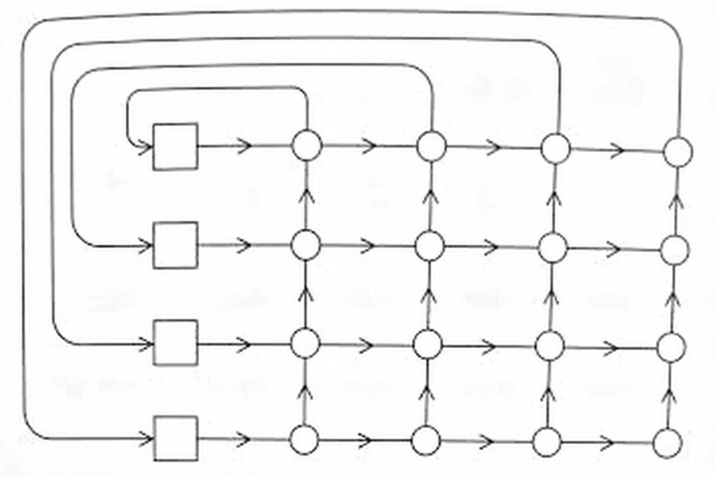


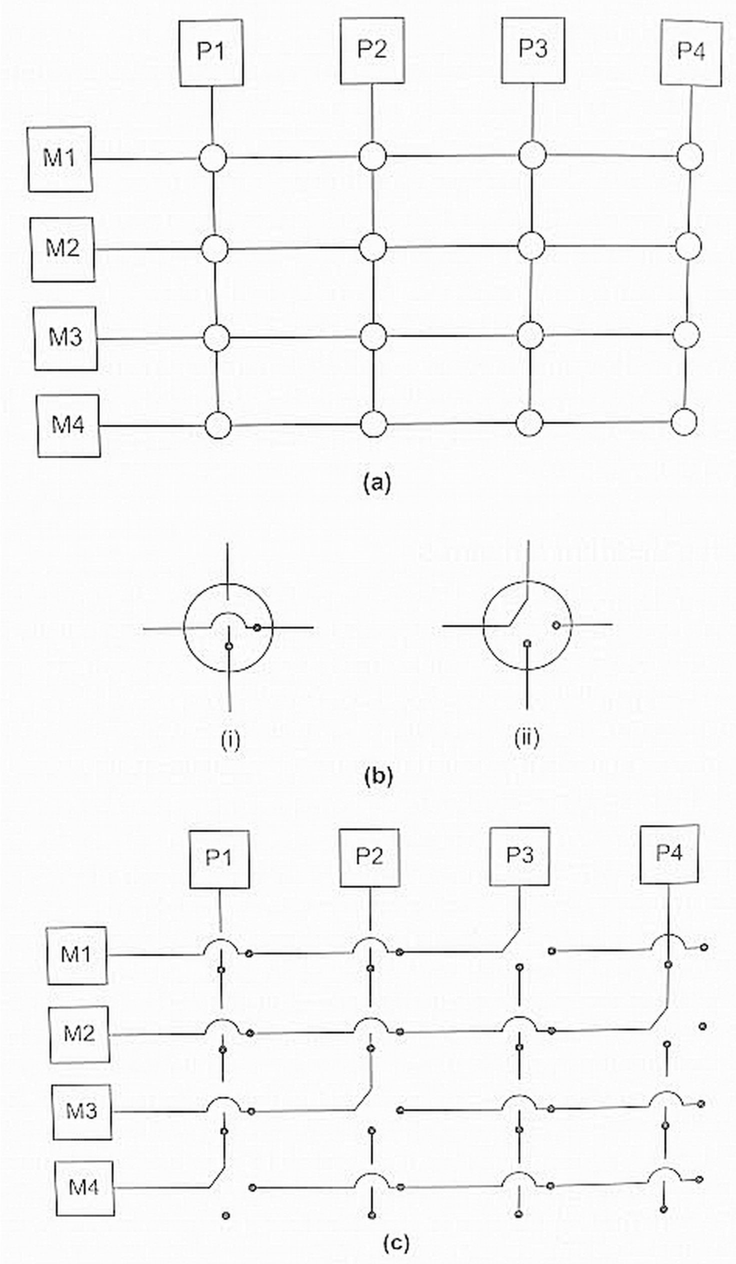
Evaluating topologies

- What is the most important network metric for a realtime distributed health monitoring system where the system must respond as soon as possible to changes in any user's heart rate, blood pressure, etc.?
  - A. Bandwidth
  - B. Latency
  - C. Diameter
  - D. Bisection width
  - E. Bisection bandwidth

Crossbar switches

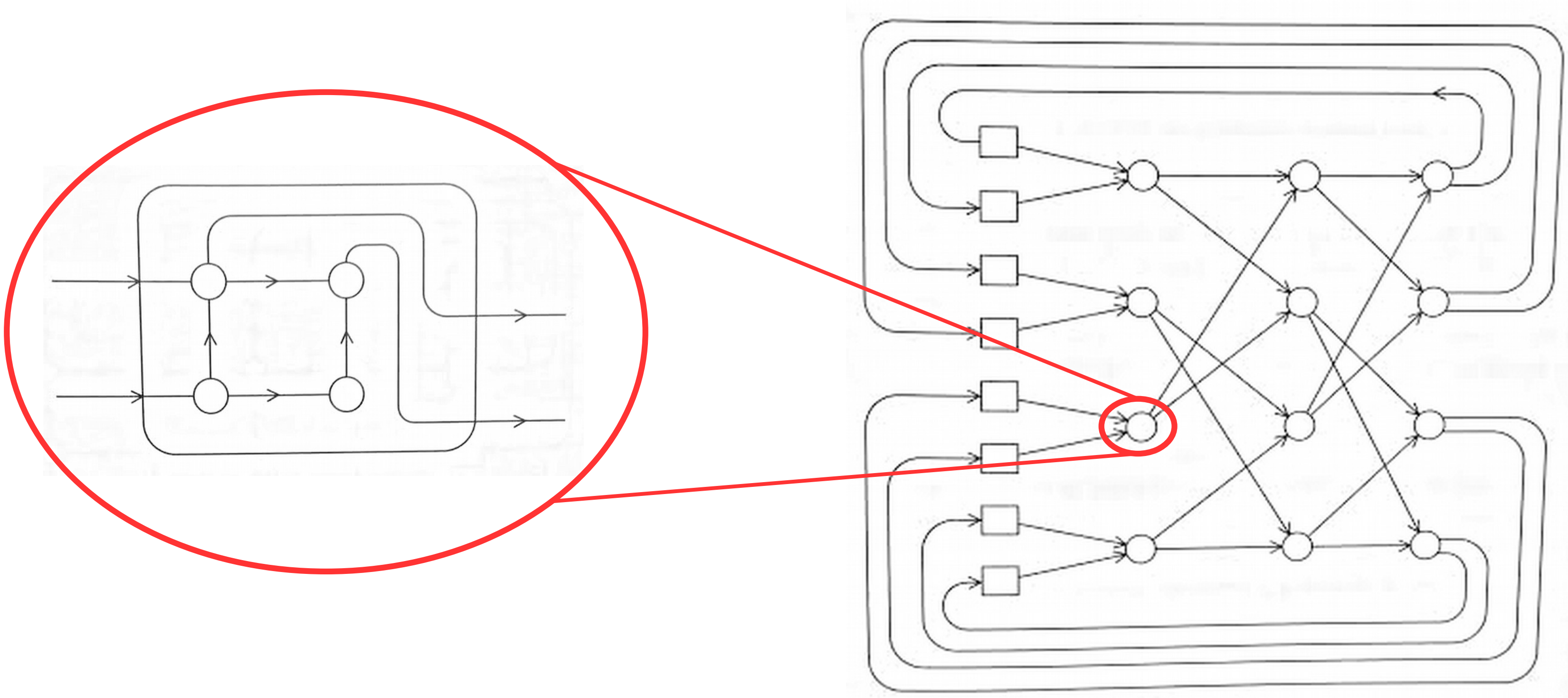
- Switched interconnects allow multiple simultaneous paths between components
  - *(Graphically, use squares for nodes and circles for switches)*
- A crossbar switch uses a matrix of potential connections to create ad-hoc paths between nodes





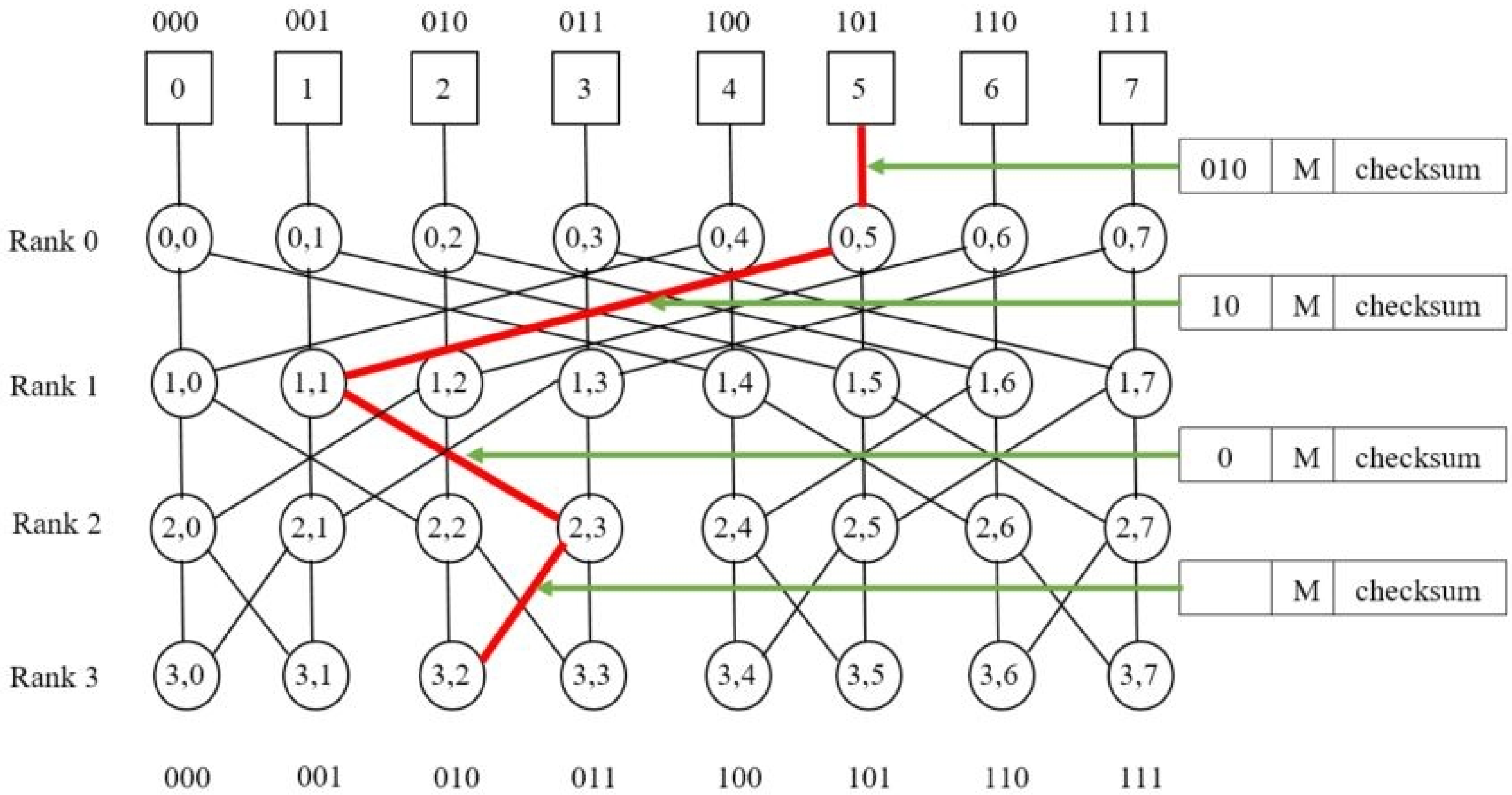
Omega networks

- Omega network: crossbar of crossbars
  - Each individual switch is a 2-by-2 crossbar



Butterfly networks

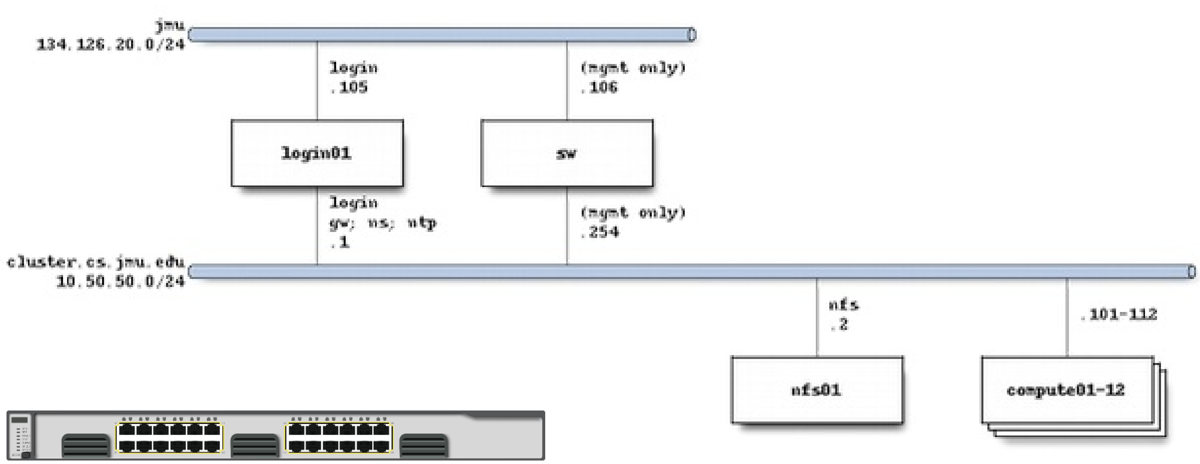
- Multi-stage network w/ dedicated switching nodes
  - Easy routing based on binary host numbers (0=left, 1=right)



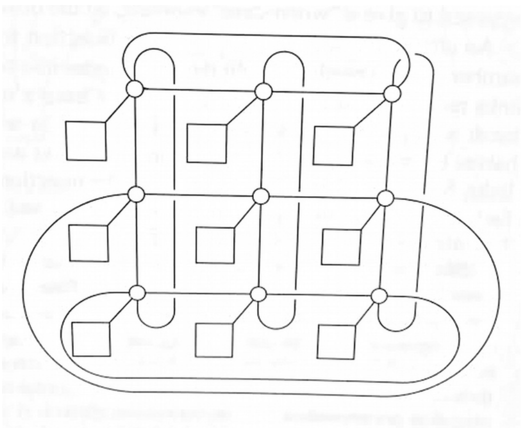
from https://en.wikipedia.org/wiki/Butterfly_network

HPC interconnects

- In an HPC system, the network is called an interconnect
  - Common patterns: switched bus, mesh/torus, hypercube
  - Connected via switches vs. connected directly



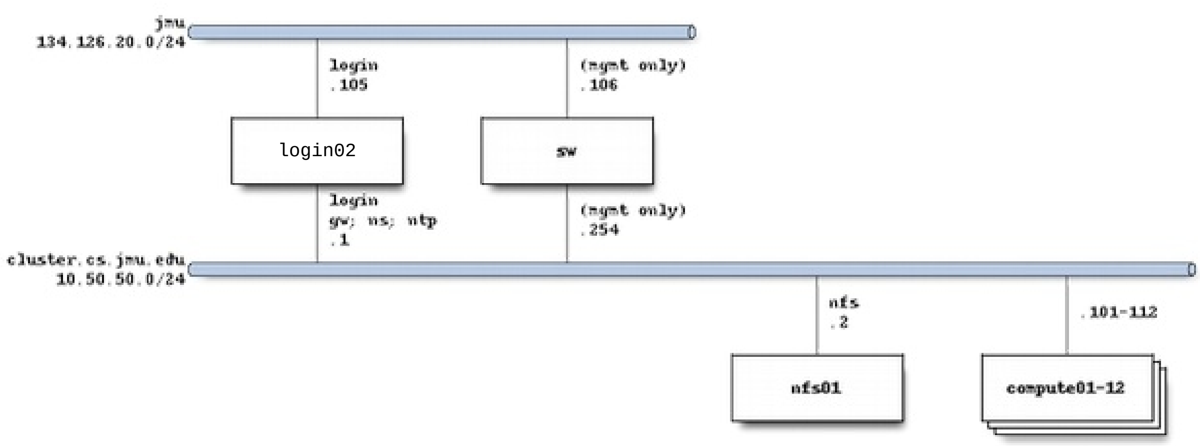
Our cluster (switched bus)



Toroidal Mesh

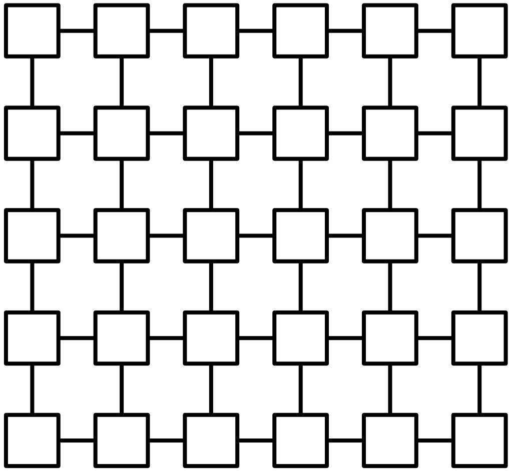
HPC interconnects

- In our cluster, which of the following is compute01 NOT connected to via a single bus hop?
  - A. compute02.cluster.cs.jmu.edu
  - B. compute08.cluster.cs.jmu.edu
  - C. nfs01.cluster.cs.jmu.edu
  - D. login02.cluster.cs.jmu.edu
  - E. stu.cs.jmu.edu

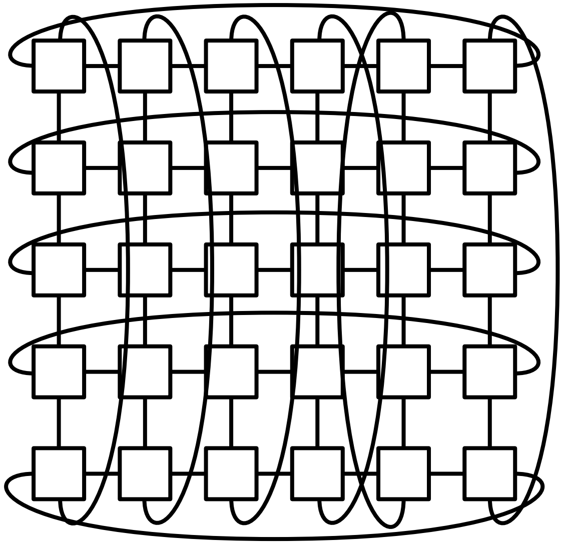


Meshes and tori

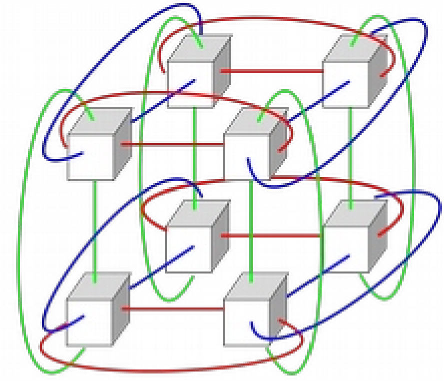
- Nodes are connected to several neighbors
  - Non-uniform memory access to non-immediate neighbors



2D Regular Mesh



2D Torus
(or "toroidal mesh")

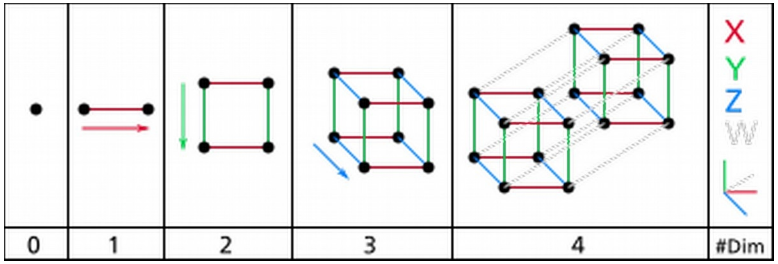


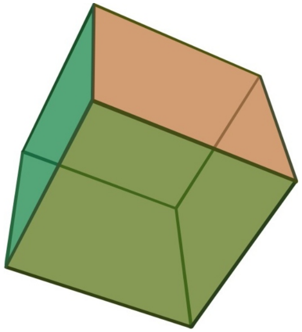
3D Torus

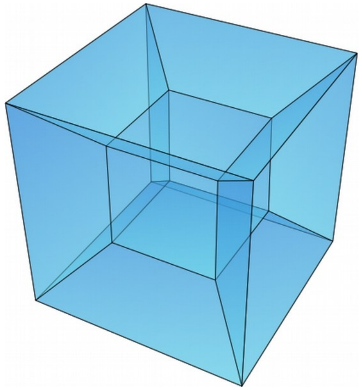
https://en.wikipedia.org/wiki/Torus_interconnect

Hypercubes

- Inductive definition:
  - 0-D hypercube: a single node
  - n-D hypercube: two (n-1)-D hypercubes with connections between corresponding nodes
    - E.g., a 3-D hypercube contains two 2-D hypercubes



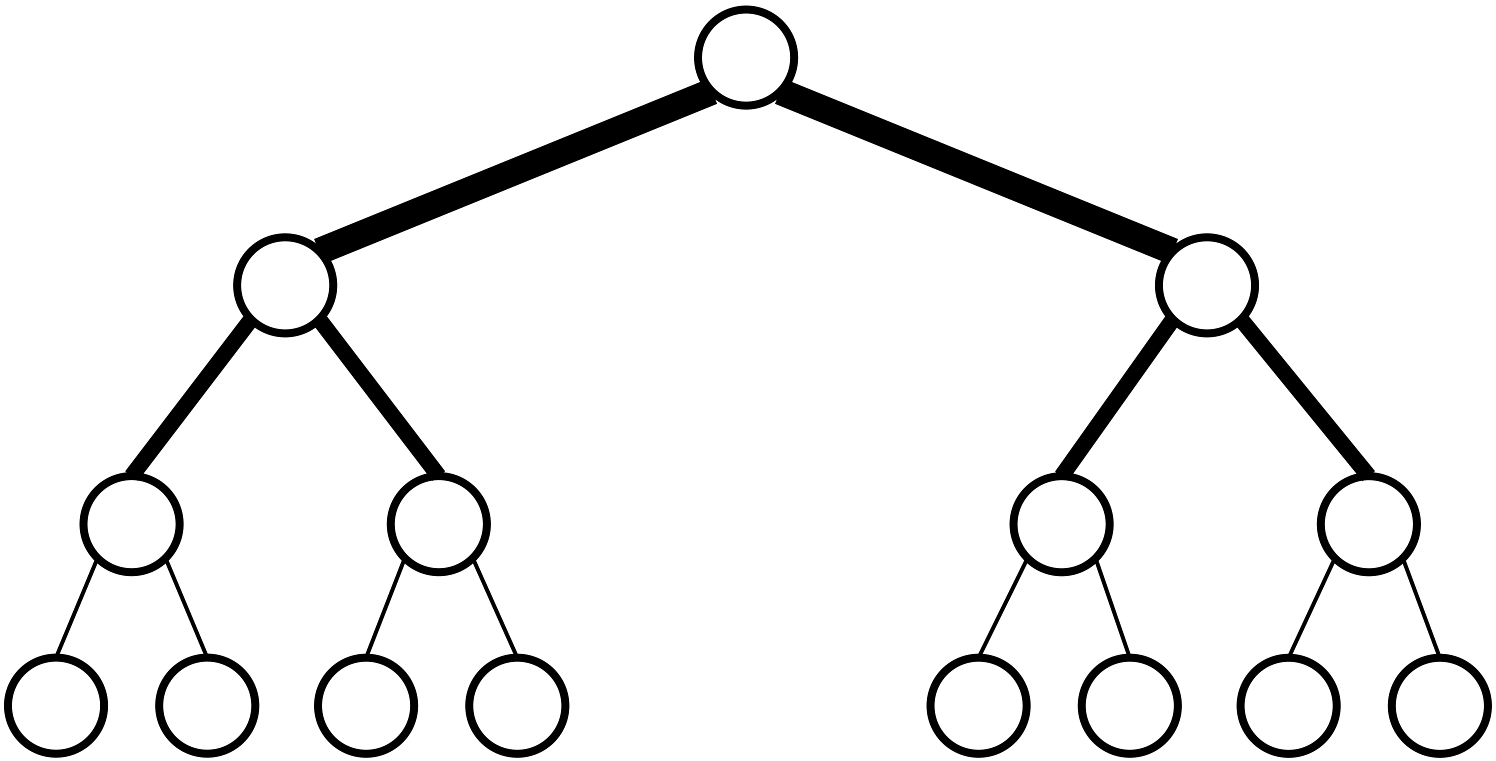




https://en.wikipedia.org/wiki/Hypercube

Fat trees

- Hierarchical tree-based topology
  - Links near the root have a higher bandwidth



Topology summary

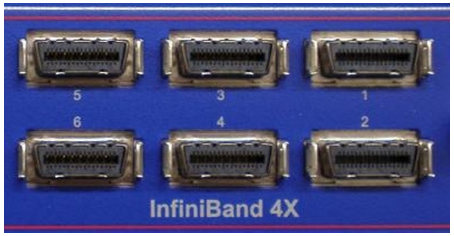
Evaluation category	Bus	Ring	2D mesh	2D torus	Hypercube	Fat tree	Fully connected
Performance							
$BW_{Bisection}$ in # links	1	2	8	16	32	32	1024
Max (ave.) hop count	1 (1)	32 (16)	14 (7)	8 (4)	6 (3)	11 (9)	1 (1)
Cost							
I/O ports per switch	NA	3	5	5	7	4	64
Number of switches	NA	64	64	64	64	192	64
Number of net. links	1	64	112	128	192	320	2016
Total number of links	1	128	176	192	256	384	2080

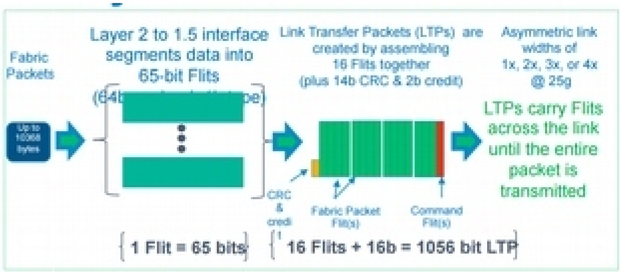
Figure E.15 Performance and cost of several network topologies for 64 nodes. The bus is the standard reference at unit network link cost and bisection bandwidth. Values are given in terms of bidirectional links and ports. Hop count includes a switch and its output link, but not the injection link at end nodes. Except for the bus, values are given for the number of network links and total number of links, including injection/reception links between end node devices and the network.

One port per node; nodes attached to switches.
Hennessy and Patterson, 2007.

HPC Interconnect Technologies

- Ethernet: 10/100 Mbps – 100 Gbps
  - Early versions used shared-medium coaxial cable
  - Newer versions use twisted pair or fiber optic with hubs or switches
- InfiniBand (IB): 24-300 Gbps w/ 0.5μs latency
  - Packet-based switched fabric (bus, fat tree, or mesh/torus)
  - Very loose API; more formal spec provided by OpenFabrics Alliance
  - Used on many current high-performance clusters
  - Vendors: Mellanox, Intel, and Oracle
- OmniPath (Intel) or Aries / Slingshot (Cray)
  - Proprietary interconnects for HPC machines





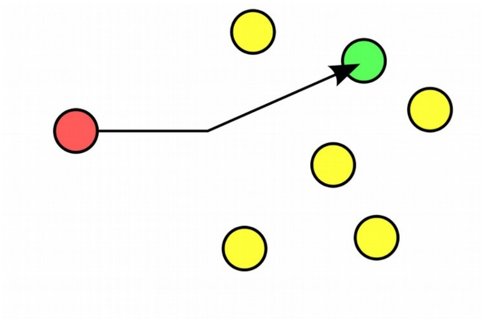
Part 2

- Routing – how traffic navigates a network (hardware and software)

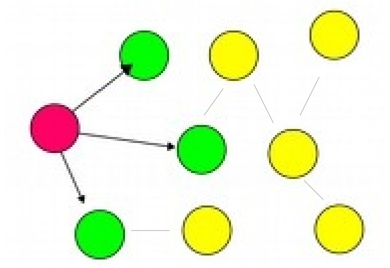
Routing

- Circuit switching
  - Paths are pre-allocated for an entire session
  - All data is routed along the same path
  - Higher setup costs and fewer simultaneous communications
  - Constant latency and throughput
- Packet switching
  - Break data into independent, addressed packets
  - Packets are routed independently
  - No setup costs and no restriction on simultaneous communications
  - Resiliency to network failures and changing conditions
  - Variable (and often unpredictable) latency and throughput

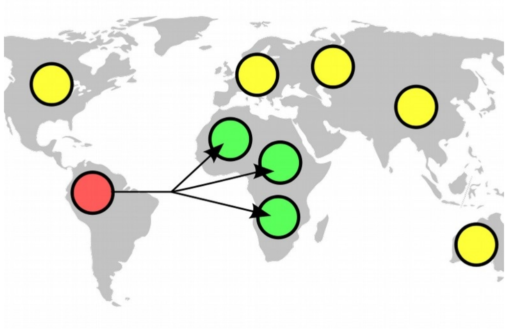
Routing



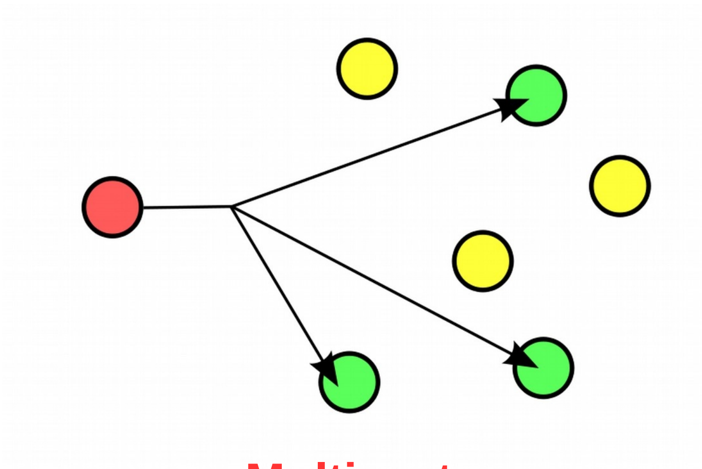
Unicast
(one-to-one)



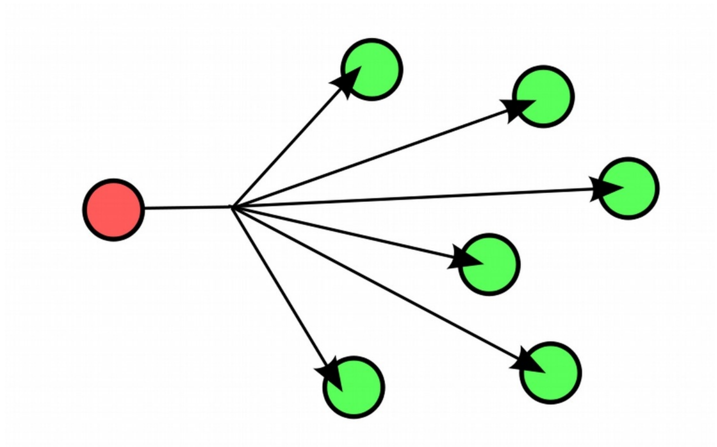
Anycast
(one-to-nearest)



Geocast
(one-to-proximate)



Multicast
(one-to-many)

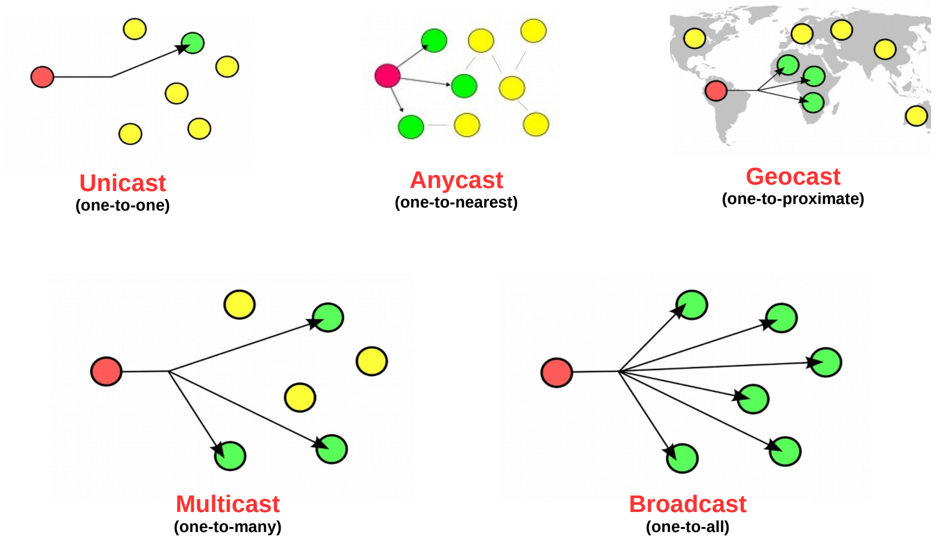


Broadcast
(one-to-all)

from `https://en.wikipedia.org/wiki/Routing#Delivery_schemes`

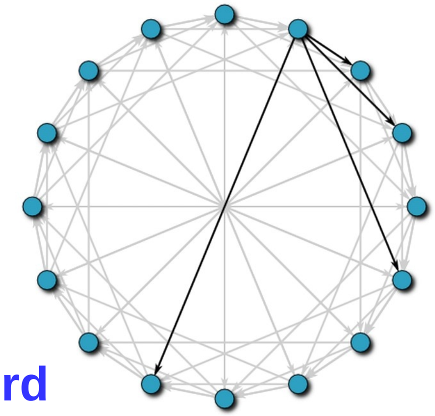
Routing

- Which routing paradigm is most appropriate for streaming an esports championship?
  - A. Unicast
  - B. Anycast
  - C. Geocast
  - D. Multicast
  - E. Broadcast



Overlays

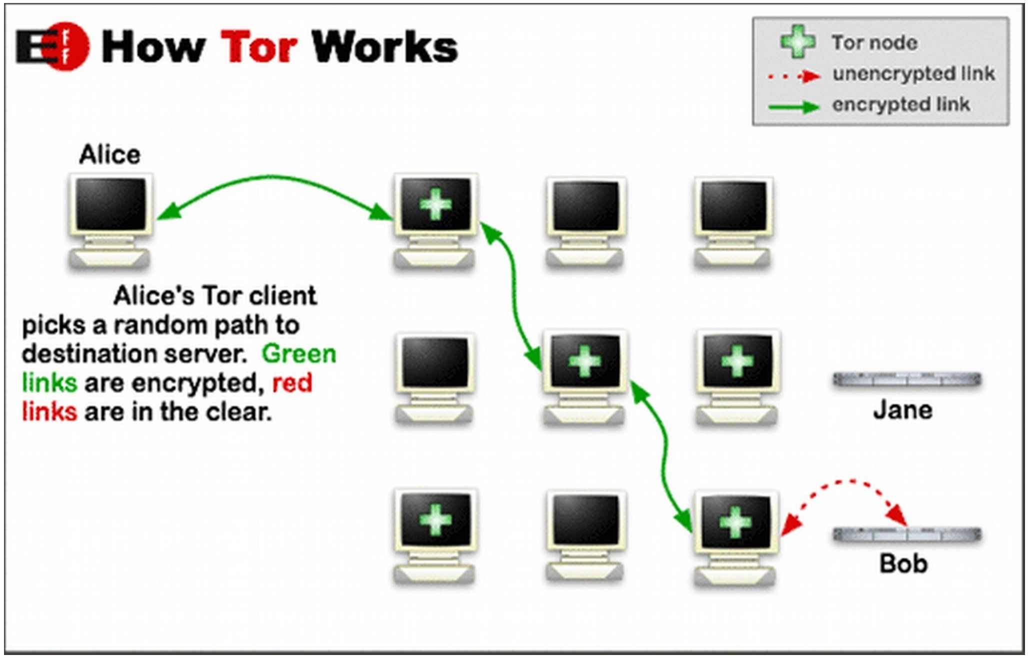
- Overlay: a network built on top of another network
  - IP multicast: technique for sending data to multiple recipients over an IP network using UDP
    - Group addressing (IGMP)
    - Tree-based distribution
  - Distributed hash tables (e.g., Chord)
  - XMPP – Jabber/Gtalk chat protocol
  - Tor network



Chord

Tor network

- Overlay network for anonymity
- Onion routing: multiple layers of obfuscation
  - At each layer, data is encrypted and sent to a random Tor relay
  - Sequence of relays form a virtual circuit that is difficult to trace
  - No single relay connects the source and destination directly



Part 3

- Protocols – how machines communicate (software, low-level)

Networking principles

- Distributed system components are often unreliable
- How do we build a **reliable** network using **unreliable** hardware and software?
  - Abstraction helps by hiding details where possible
  - Protocols define well-structured communication patterns
  - Layered / stacked protocols build on each other
  - Each layer adds metadata to help solve a specific problem
- Another guiding principle: the end-to-end principle
  - *Application-specific functions ought to reside in the end hosts of a network rather than in intermediary nodes whenever possible.*

For more info:
`http://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf`

Networking principles

- Which of the following is a violation of the end-to-end principle?
  - A. Public key encryption
  - B. ISP-based acceleration of Netflix traffic
  - C. Client-server chat programs
  - D. Web browser ad-blockers
  - E. Cross-platform video game multiplayer

QoS concerns

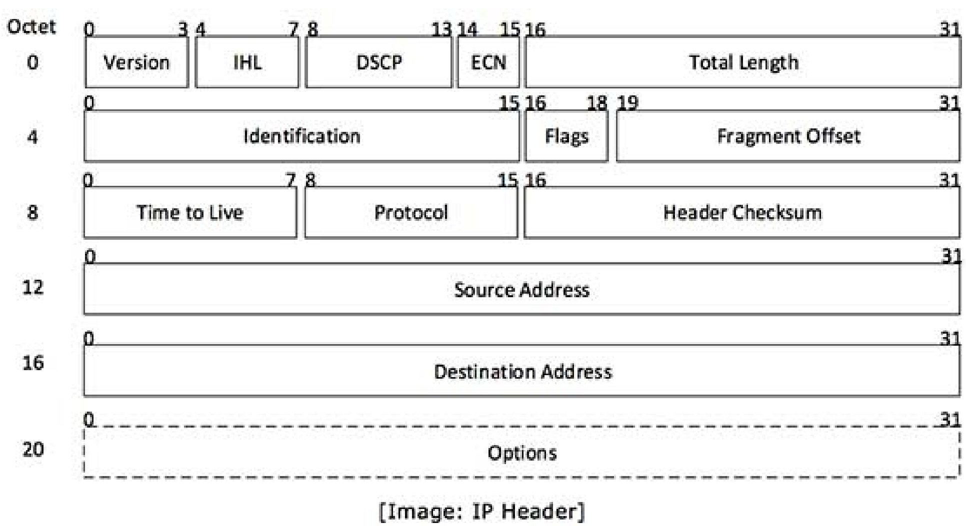
- Quality of Service (QoS) guarantees
  - Possible reasons to violate end-to-end principle
  - Minimum required bit rate (bandwidth)
  - Maximum delay to set up a session
  - Maximum end-to-end delay (latency)
  - Maximum delay variance (jitter)
  - Maximum round-trip delay
  - Possibility of expedited forwarding
  - Synchronization mechanisms
  - Examples: MPEG-2, HLS

QoS concerns

- Which QoS concern would be most important for streaming video?
  - A. Minimum bitrate
  - B. Maximum setup delay
  - C. Maximum latency
  - D. Maximum jitter
  - E. Possibility of expedited forwarding

Networking protocols

- Routing: choosing a path through a network
- Datagram: self-contained, encapsulated package of data and metadata capable of being routed
  - Also called a frame: (layer 2), a packet (layer 3), or a segment (layer 4)
- Protocol: rules for exchanging data (often using datagrams)
- Checksums: data integrity verification mechanism



[Image: IP Header]

IPv4 header

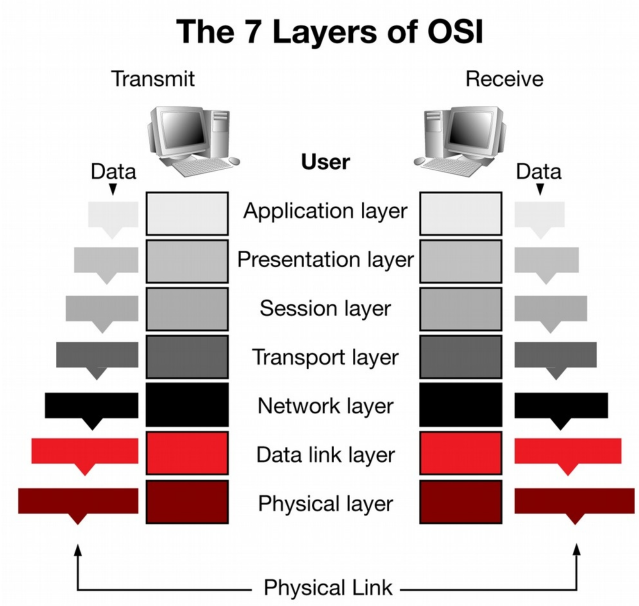
(from `https://www.tutorialspoint.com/ipv4/ipv4_packet_structure.htm`)

Protocol design issues

- Connectionless vs. connection-oriented
  - Is there a setup/teardown procedure required for communication?
  - No setup costs vs. faster speed after connection
- Synchronous vs. asynchronous
  - Does the sender block after sending?
    - E.g., `MPI_Ssend` vs. `MPI_Isend`
  - Easier to debug and verify vs. faster communication
- Persistent vs. transient communication
  - Are messages stored by the middleware?
  - Guaranteed delivery vs. simplicity of middleware

OSI model layers

1) Physical: Transmission of raw bits over a physical medium (Ethernet, 802.11)
2) Data link: Reliable transmission of frames between two nodes (FC, 802.11)
3) Network: Structured transmission on a multi-node network (IP, ICMP)
4) Transport: Reliable transmission on a multi-node network (TCP, UDP)
5) Session: Managed communication sessions (RPC, NFS)
6) Presentation: Encoding and conversion of data (HTML, XML, JSON)
7) Application: Application-level abstractions (FTP, HTTP, SSH, MPI)



Part 4

- IPC paradigms – how processes communicate (software, high-level)

IPC paradigms

- Inter-process communication (IPC)
  - Message-passing (explicit)
    - Symmetric (SPMD) vs. asymmetric (differentiated hosts)
  - Remote procedure calls (implicit)
    - Synchronous vs. asynchronous

Berkeley / POSIX Sockets

- API for inter-process communication
  - Originally designed for BSD
  - Later evolved into a POSIX standard
  - Often used for low-level TCP and UDP communication
  - Hosts identified by address (usually IP) and port number
  - Passes "messages" (packets) between hosts
  - Can use Unix pipes if both endpoints are on a single host

Socket primitives

- Server
  - Socket: Create a new endpoint
  - Bind: Attach a local address to a socket
  - Listen: Announce readiness for connections
  - Accept: Block until a request arrives
- Client
  - Connect: Attempt to establish a connection
- Server & client
  - Write: Send data over a connection
  - Read: Receive data over a connection
  - Close: Destroy a connection

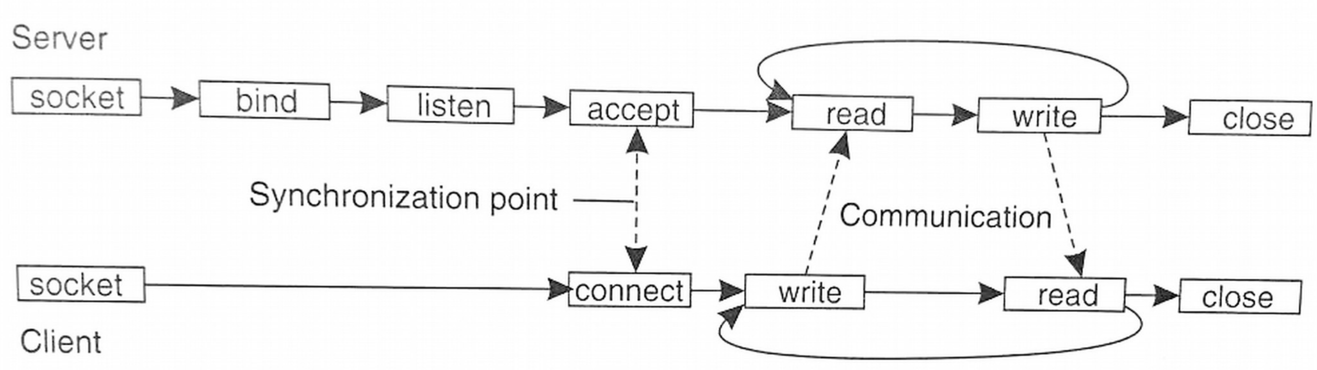


Figure 4-15. Connection-oriented communication pattern using sockets.

Socket primitives

- Which of the following is NOT a valid event sequence using sockets? (other events may occur between the events in the sequence)
  - A. accept, write, read
  - B. accept, read, write
  - C. accept, listen, read
  - D. listen, accept, read
  - E. socket, connect, write

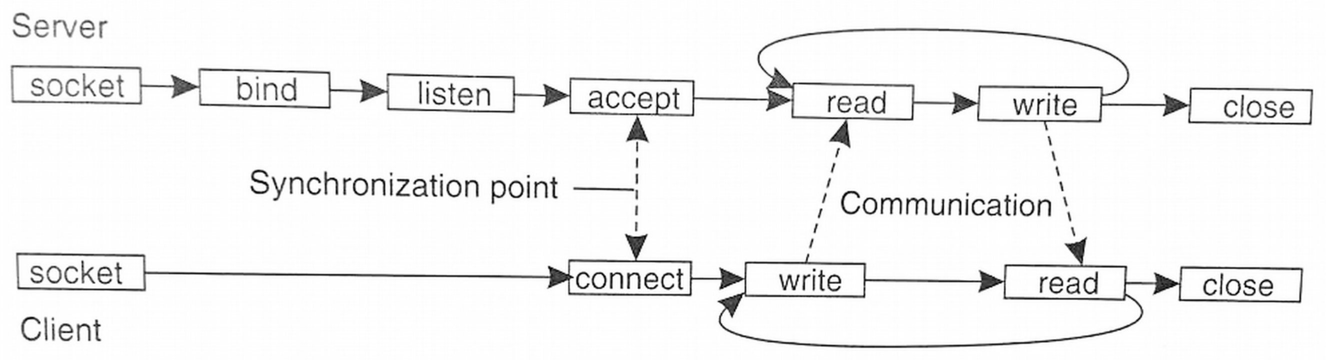
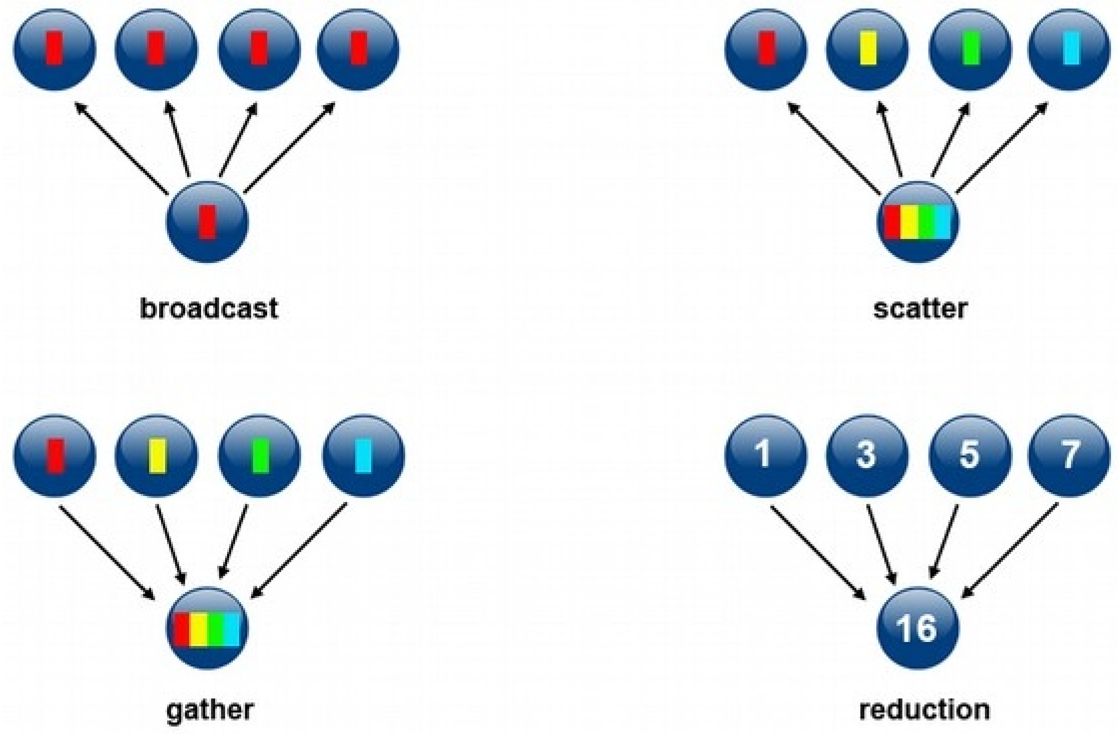


Figure 4-15. Connection-oriented communication pattern using sockets.

MPI (Message Passing Interface)

- MPI_Send
- MPI_Recv
- MPI_Bcast
- MPI_Scatter
- MPI_Gather
- MPI_Allgather
- MPI_Reduce
- MPI_Allreduce
- MPI_Alltoall



from https://computing.llnl.gov/tutorials/parallel_comp/

Remote Procedure Call (RPC)

- Key idea: transparency
  - It should look like the procedure call is happening locally
  - Similar in spirit to PGAS remote memory accesses
  - Implement server / client stubs to handle the call
- Parameter marshalling
  - Preparing parameters for transmission over a network

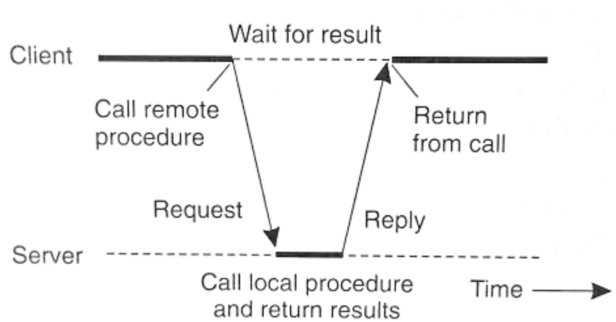


Figure 4-6. Principle of RPC between a client and server program.

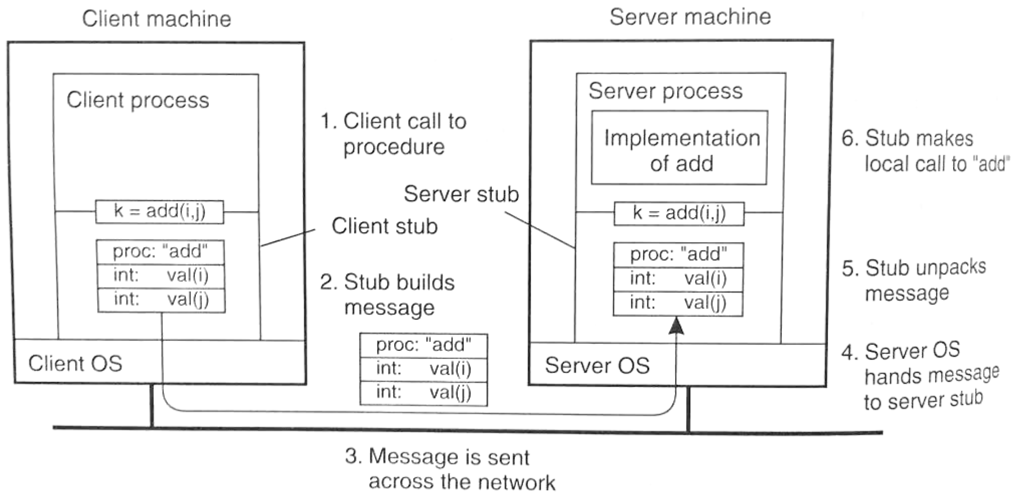


Figure 4-7. The steps involved in a doing a remote computation through RPC.

Asynchronous RPC

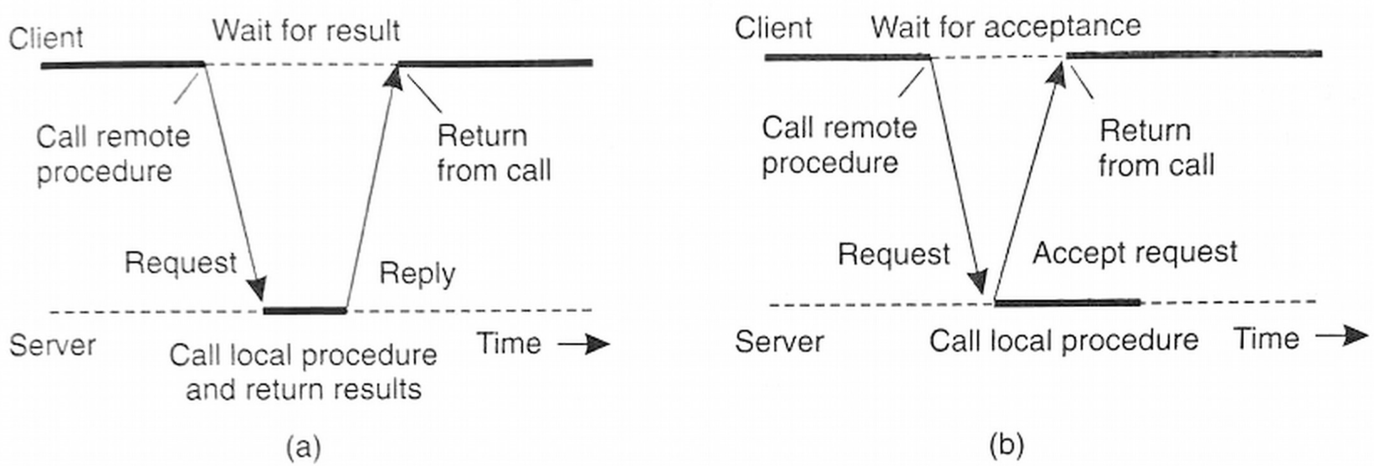


Figure 4-10. (a) The interaction between client and server in a traditional RPC. (b) The interaction using asynchronous RPC.

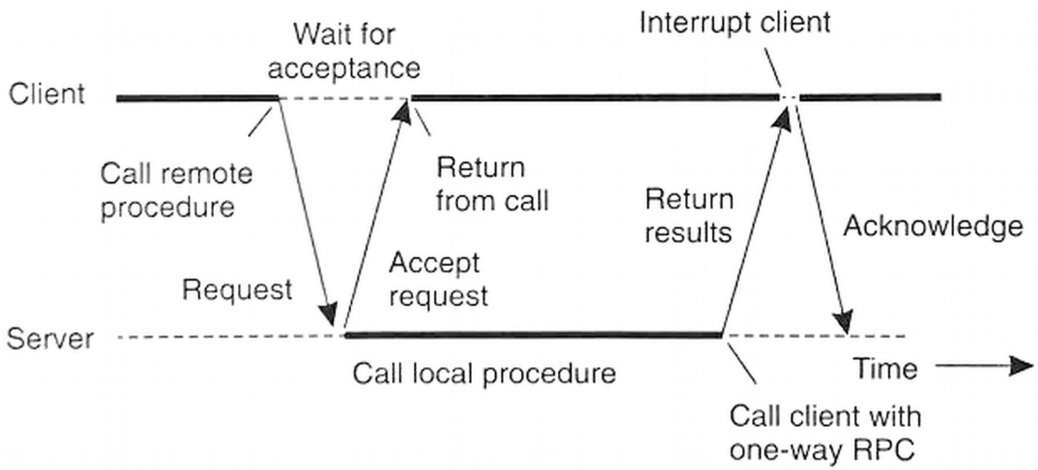


Figure 4-11. A client and server interacting through two asynchronous RPCs.

Summary

- Topologies – how a network is arranged (hardware)
- Routing – how traffic navigates a network (hardware and software)
- Protocols – how machines communicate (software, low-level)
- IPC paradigms – how processes communicate (software, high-level)

Next time: how do we **identify** hosts on a network?
(e.g., what is a host’s name)