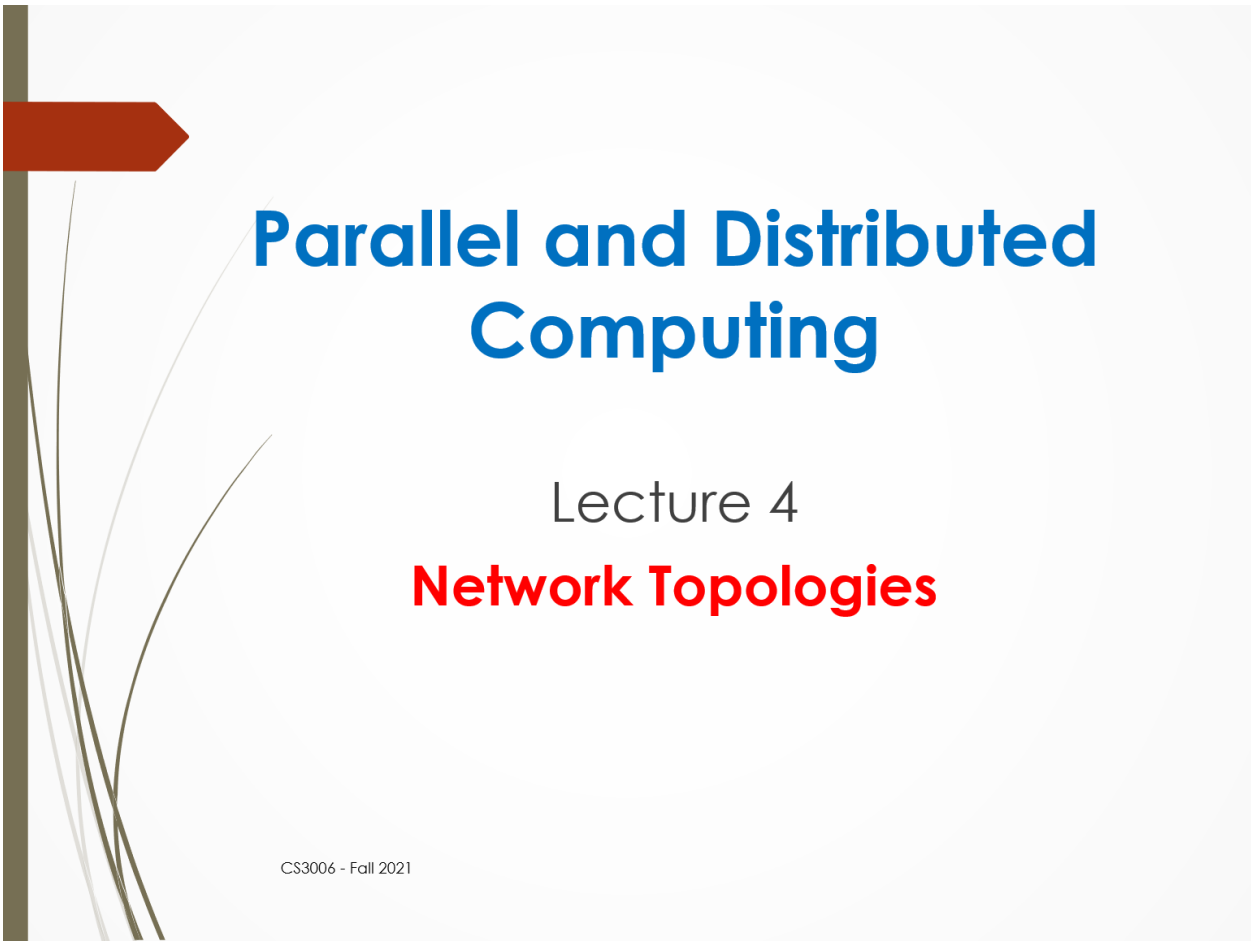


Absolutely! Let's break down **network topologies in parallel and distributed computing** in a **clear, exam-friendly way**, with simple diagrams you can imagine.

---

1. Network Topologies

In **parallel and distributed systems**, network topology describes **how processors (nodes) are connected** and how they **communicate**.

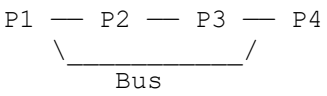
Communication performance depends heavily on the topology.

A) Bus Topology

- **Structure:** All nodes connected to a **single communication line** (bus).
- **Advantages:** Simple, cheap, easy to implement.
- **Disadvantages:** Only one communication at a time → contention; poor scalability.

- **Example:** Small LANs, early multi-processor systems.

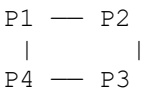
Diagram:



B) Ring Topology

- **Structure:** Nodes connected in a **closed loop**. Each node connected to **two neighbors**.
- **Advantages:** Simple, deterministic path, equal access.
- **Disadvantages:** Slow for distant nodes, failure of one node may break network.
- **Example:** Token ring networks.

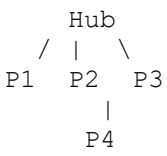
Diagram:



C) Star Topology

- **Structure:** All nodes connected to a **central hub/switch**.
- **Advantages:** Easy to manage, hub handles traffic, failure of one node does not affect others.
- **Disadvantages:** Hub failure → whole network down, hub can become bottleneck.
- **Example:** Ethernet LANs with switches.

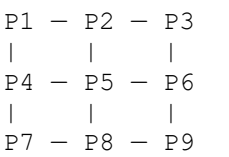
Diagram:



D) Mesh Topology

- **Structure:** Each node connected to **neighboring nodes** in a grid pattern. Can be **2D or 3D mesh**.
- **Advantages:** Multiple paths → fault tolerant, scalable.
- **Disadvantages:** More complex wiring, more links.
- **Example:** Supercomputer interconnects (Cray, Blue Gene).

Diagram (2D Mesh):



E) Hypercube (n-cube) Topology

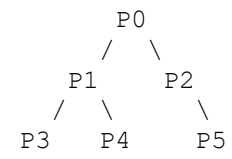
- **Structure:** Nodes connected in a hypercube pattern; for n-dimensions, each node has n links.
- **Advantages:** Logarithmic diameter → low max hops, high connectivity.
- **Disadvantages:** Complex wiring for high dimensions.
- **Example:** Some parallel supercomputers, high-performance clusters.

Diagram (3-cube):

F) Tree Topology

- **Structure:** Nodes connected hierarchically like a **tree**.
- **Advantages:** Scalable, easy to manage, good for routing.
- **Disadvantages:** Root failure affects entire network; longer paths for distant nodes.
- **Example:** Multistage interconnection networks in multiprocessors.

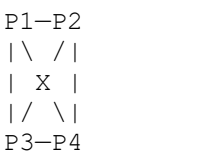
Diagram:



G) Fully Connected / Clique

- **Structure:** Every node connected to **every other node**.
- **Advantages:** Minimum communication delay, direct path.
- **Disadvantages:** Very expensive for large networks, huge number of links.
- **Example:** Small multiprocessor systems.

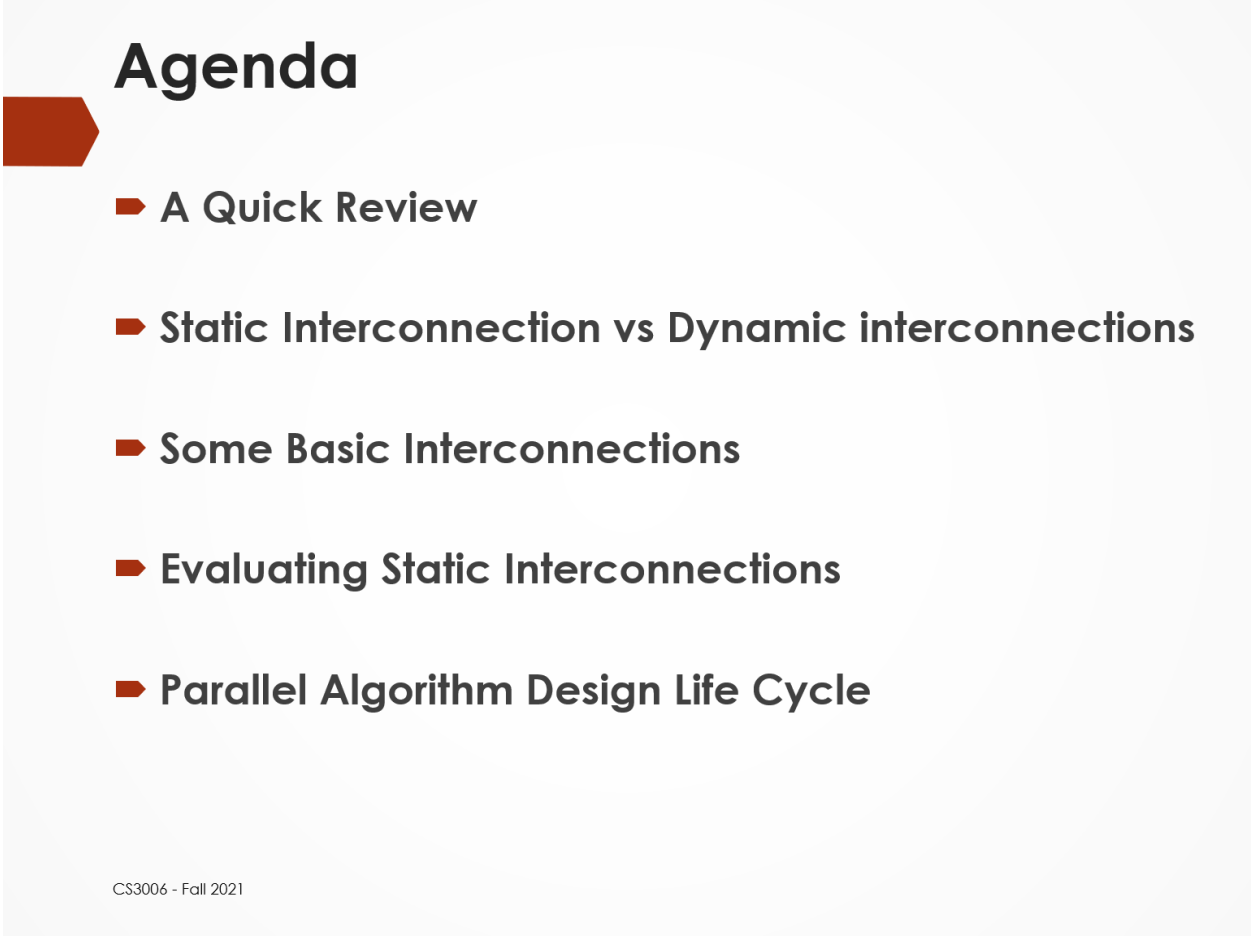
Diagram:



Summary Table for Exam

Topology	Links per Node	Advantages	Disadvantages	Example
Bus	1	Simple, cheap	Contention, poor scalability	Early LANs
Ring	2	Equal access	Slow for distant nodes, failure sensitive	Token ring
Star	1 to hub	Easy management	Hub is bottleneck	Ethernet LAN
Mesh	2-4	Fault tolerant, scalable	Complex wiring	Cray supercomputers
Hypercube	log2(N)	Low diameter, high connectivity	Complex wiring	Parallel clusters
Tree	1-2	Scalable, easy routing	Root failure affects network	Multistage interconnect
Fully Connected	N-1	Minimum delay	Expensive, many links	Small multiprocessor system

---



---

1. Static Interconnection vs Dynamic Interconnection

Interconnection = How processors are connected to each other and to memory in a parallel system.

A) Static Interconnection

- **Definition:** The links between processors are **fixed** and do not change at runtime.
- **Advantages:** Simple design, predictable performance, low hardware cost.
- **Disadvantages:** Less flexible, may suffer from **communication bottlenecks** if traffic is uneven.
- **Examples:**
  - **Bus** (all processors connected via single bus)
  - **Ring** (each processor connected to two neighbors)
  - **Mesh / Hypercube**

Key Point: Topology is fixed → paths do not change.

B) Dynamic Interconnection

- **Definition:** Connections between processors can **change at runtime** depending on the communication needs.
- **Advantages:** Flexible, avoids congestion, can optimize communication.
- **Disadvantages:** Complex hardware, higher cost.
- **Examples:**
  - **Switch-based networks**
  - **Crossbar switches** (can connect any processor to any memory dynamically)
  - **Multistage interconnection networks**

Key Point: Links are programmable or switchable → better utilization.

---

2. Some Basic Interconnections (Static)

Interconnection	Description	Example
Bus	Single line connecting all processors	Early LAN, simple multiprocessors
Ring	Each node connected to 2 neighbors	Token ring networks
Mesh	2D grid of nodes	Cray supercomputers
Hypercube	n-dimensional cube	Parallel clusters
Tree	Hierarchical connections	Multistage interconnection
Fully Connected	Every node connected to every other node	Small multiprocessor systems

Note: Static → fixed; Dynamic → switchable / programmable.

---

3. Evaluating Static Interconnections

When analyzing interconnection networks, we check:

1. **Degree (d):** Number of links per processor.
2. **Diameter (D):** Maximum number of hops between two processors.
3. **Cost (C):** Total number of links = Number of processors × Degree ÷ 2.
4. **Scalability:** Can the network grow easily?
5. **Reliability:** Can the network tolerate node/link failures?

Example Table:

Topology	Degree	Diameter	Cost	Scalability
Bus	1	N-1	N-1	Poor
Ring	2	N/2	N	Medium
Mesh	2-4	2√N	2(N-√N)	Good
Hypercube	log2N	log2N	N*log2N/2	Excellent

---

4. Parallel Algorithm Design Life Cycle

Designing a parallel program involves **structured steps**:

1. **Problem Analysis:**
   - Understand the task, data dependencies, and compute-intensive parts.
2. **Decomposition:**
   - Break the problem into **smaller sub-tasks**.
   - Types of decomposition:
     - **Data decomposition:** Divide data among processors (array, matrix).
     - **Task decomposition:** Divide tasks among processors.
3. **Mapping / Assignment:**
   - Assign sub-tasks to processors and plan **communication** between them.
4. **Design of Communication & Synchronization:**
   - Decide how processors exchange data (message passing, shared memory).
   - Synchronization mechanisms: barriers, locks, semaphores.
5. **Algorithm Formulation:**
   - Develop **parallel algorithm** considering computation and communication.
6. **Performance Evaluation:**
   - Measure **speedup, efficiency, scalability**.
   - Identify bottlenecks → refine algorithm.

Example (Matrix Multiplication):

- **Decomposition:** Divide matrix into blocks.
- **Mapping:** Assign each block to a processor.
- **Communication:** Send/receive necessary blocks.
- **Synchronization:** Ensure all processors finish before combining results.

✅ Exam-Friendly Summary

Static vs Dynamic Interconnection:

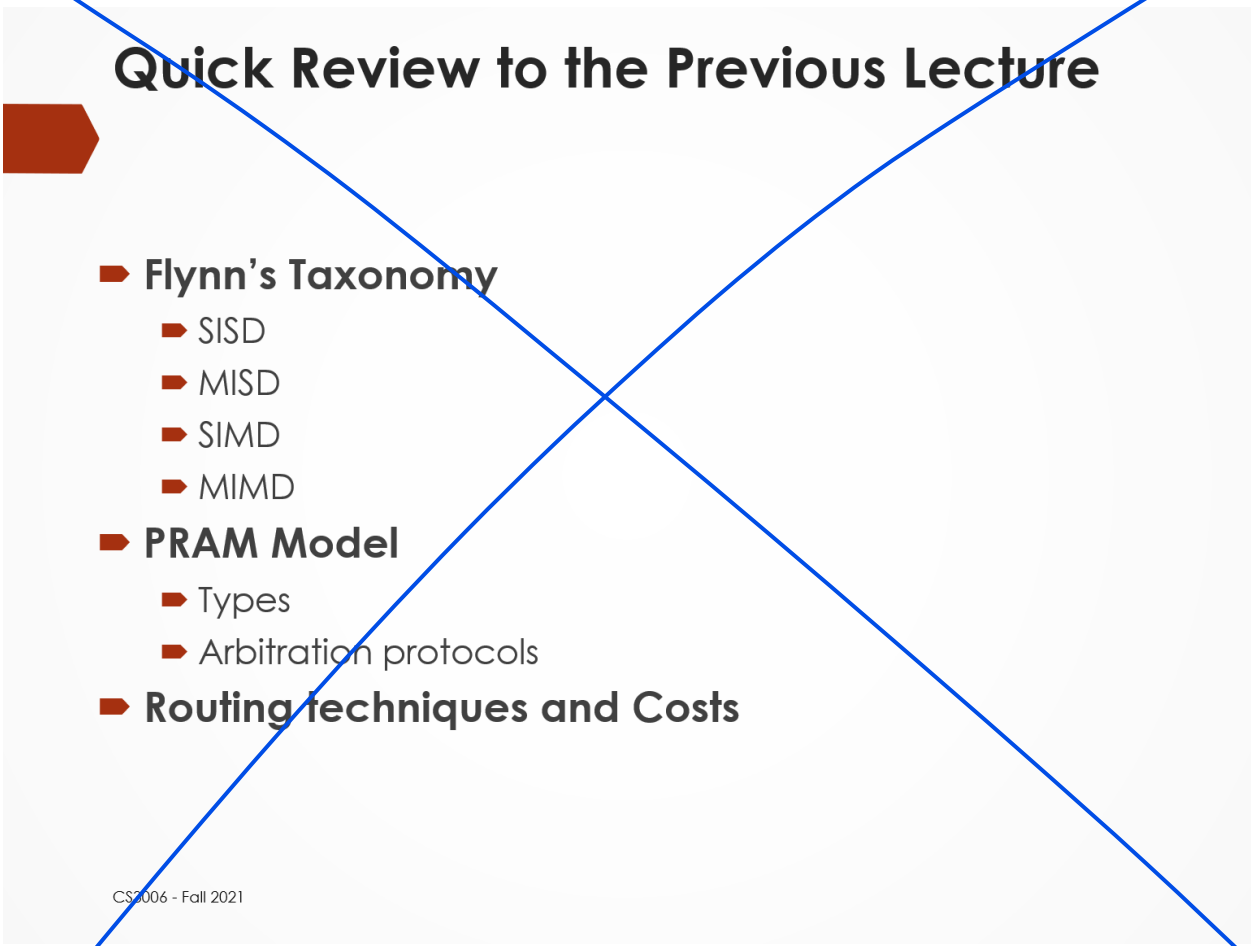
Feature	Static	Dynamic
Links	Fixed	Programmable/switchable

Feature	Static	Dynamic
Complexity	Low	High
Flexibility	Low	High
Example	Bus, Ring, Mesh	Crossbar, Switch, Multistage

Parallel Algorithm Design Life Cycle:

1. Problem Analysis → 2. Decomposition → 3. Mapping → 4. Communication/Synchronization → 5. Algorithm Formulation → 6. Performance Evaluation

---



Sure! Let's put all these topics together in a **clear, concise, exam-friendly way**, with definitions, types, and examples.

---

1. Flynn's Taxonomy

Classifies computer architectures based on **instruction streams** and **data streams**.

Type	Instruction Stream	Data Stream	Example	Notes
SISD	Single	Single	Traditional serial computer	Classical von Neumann architecture; single core
SIMD	Single	Multiple	GPU, Intel MMX, Cray vector machines	Same instruction applied to different data; good for arrays, graphics
MISD	Multiple	Single	Systolic arrays	Rarely used; pipelines of functional units operating on same data
MIMD	Multiple	Multiple	Multi-core CPUs, Clusters	Each processor executes independent instructions on independent data; most modern parallel systems

Note: SIMD uses **single control unit**, simpler hardware; MIMD is more flexible.

---

2. PRAM Model (Parallel Random Access Machine)

- Idealized parallel computer for algorithm design.
- **Components:**
  - p processors
  - Shared global memory (uniform access)
  - Single synchronized clock

Problem: Multiple processors may try to **read/write the same memory location simultaneously** → need arbitration.

PRAM Types (Based on Memory Access)

Type	Read	Write	Notes
EREW	Exclusive	Exclusive	No two processors read/write same memory simultaneously; simplest model
CREW	Concurrent	Exclusive	Multiple reads allowed; writes serialized
ERCW	Exclusive	Concurrent	Writes allowed concurrently; reads serialized
CRCW	Concurrent	Concurrent	Most powerful; multiple reads and writes; arbitration required

---

Arbitration Protocols (for concurrent writes)

Protocol	Description	Example
Common	Write only if all processors want same value	All want 5 → memory = 5
Arbitrary	Randomly pick one processor's value	5 or 10 randomly chosen
Priority	Processor with highest priority wins	Highest ID processor value written
Sum / Associative	Write sum (or other associative operation) of values	5 + 10 → 15

---

3. Routing Techniques & Costs

Communication is a **major overhead** in parallel programs.

Cost components:

1. **Startup time (ts):** Time at sending/receiving node (prepare headers, trailers, parity, establish links).
2. **Per-hop time (th):** Time for message header to traverse each network hop (switch/router delay, decision latency).
3. **Per-word transfer time (tw):** Time per word of message (bandwidth, buffering overhead).

$$
T_{\text{comm}} \approx t_s + l \cdot t_h + m \cdot t_w
$$

Where:

- (l) = number of hops
- (m) = message size (words)

Routing Techniques

Technique	Description	Example
Store-and-Forward	Each node receives the entire message before forwarding	Early network protocols
Packet Routing	Divide message into packets, pipeline through network; each packet has routing info, sequence, error checks	TCP/IP networks

Technique	Description	Example
Cut-Through Routing	Divide message into small units (flits), all flits follow same path; minimal headers, error check at end	Supercomputer interconnects (InfiniBand)

Notes:

- Cut-through routing → **faster than store-and-forward**, reduces latency.
- Communication cost increases if network is **congested**.

✅ Exam-Friendly Summary Table

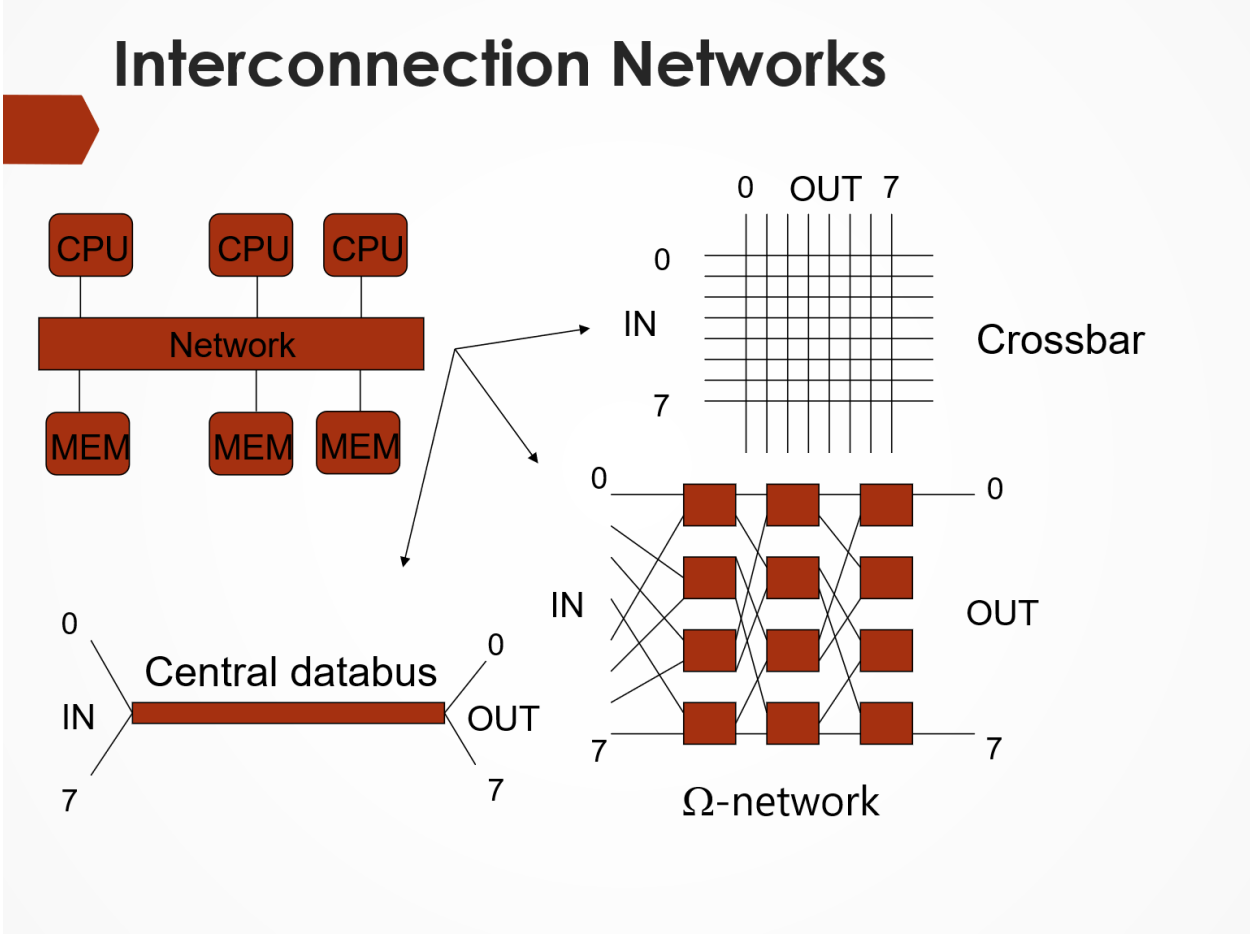
Topic	Key Points	Example
Flynn's Taxonomy	SISD, SIMD, MISD, MIMD	CPU, GPU, Cluster
PRAM Types	EREW, CREW, ERCW, CRCW	Ideal parallel algorithms
Arbitration	Common, Arbitrary, Priority, Sum	Resolve concurrent writes
Routing	Store-and-forward, Packet, Cut-through	TCP/IP, InfiniBand
Communication Cost	$T \approx t_s + l\,t_h + m\,t_w$	Message passing between nodes

---

Interconnection Networks

- Main problem is how to do interconnections of the CPUs to each other and to the memory

- There are three main network topologies available:
  - Crossbar (n^2 connections – data path without sharing)
  - Multi-stages network ($n \log_2 n$ connections - $\log_2 n$ switching stages and shared on a path)
  - Central databus (1 connections - n shared)



Got it! Let's make this **super simple and exam-friendly**.

Interconnection of CPUs to Memory

Problem: How to connect multiple CPUs to memory efficiently for **parallel processing**.

- Goal: Minimize **contention**, **latency**, and **hardware cost**.
- Three main **network topologies** for connecting CPUs to memory:

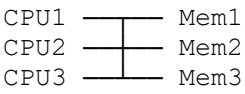
---

1. Crossbar Network

- **Structure:** Every CPU connected to **every memory module** via a dedicated path.
- **Connections:** (n^2) for n CPUs and n memory modules.
- **Advantages:**
  - No contention (each CPU-memory pair has its own path)
  - Very fast

- **Disadvantages:**
  - Expensive for large n (hardware cost grows as n^2)
- **Example:** Small high-speed multiprocessors

Diagram (simplified for 3 CPUs & 3 Memory Modules):



(Each CPU can access any memory independently via separate paths.)

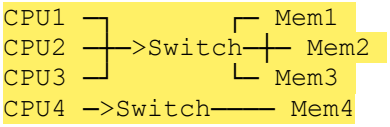
---

2. Multi-stage Network (MSN)

n log 2 n (2 base may ha

- **Structure:** Uses **log2 n switching stages** to connect CPUs and memory modules.
- **Connections:** (n \cdot \log_2 n) switches (cheaper than crossbar).
- **Advantages:**
  - Supports simultaneous connections for multiple CPU-memory pairs
  - Less expensive than crossbar
- **Disadvantages:**
  - Paths are **shared**, so contention possible
- **Example:** Omega network, Butterfly network

Diagram (simplified for 4 CPUs & 4 Memory Modules):

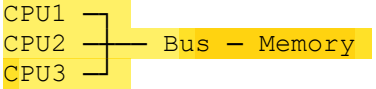


---

3. Central Databus

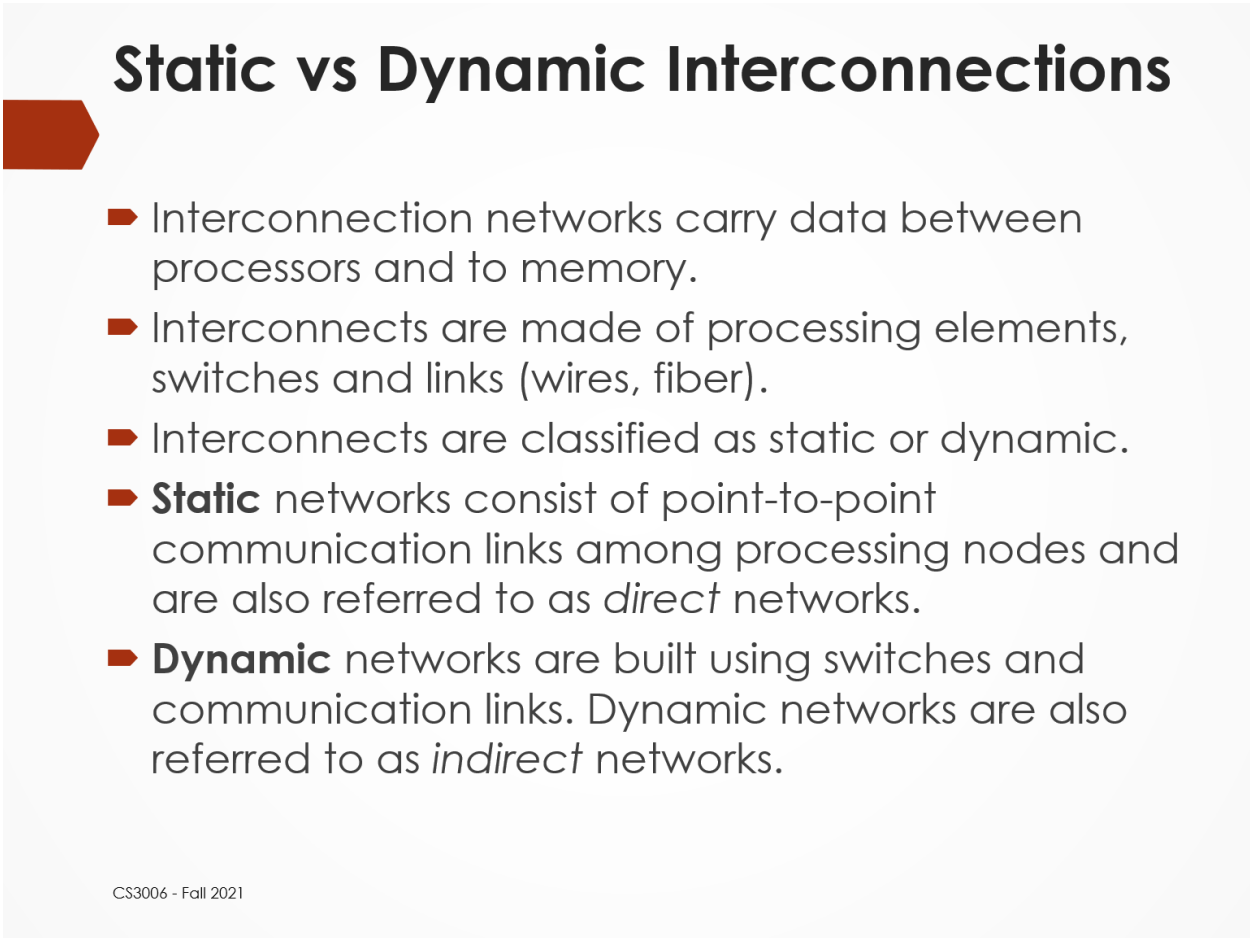
- **Structure:** All CPUs connected to **one shared bus**.
- **Connections:** Only **1 bus** → n CPUs share it.
- **Advantages:** Simple, cheap hardware
- **Disadvantages:**
  - Only **one CPU can access memory at a time** → contention, slow for large n
- **Example:** Early microprocessor systems, small multiprocessors

Diagram:



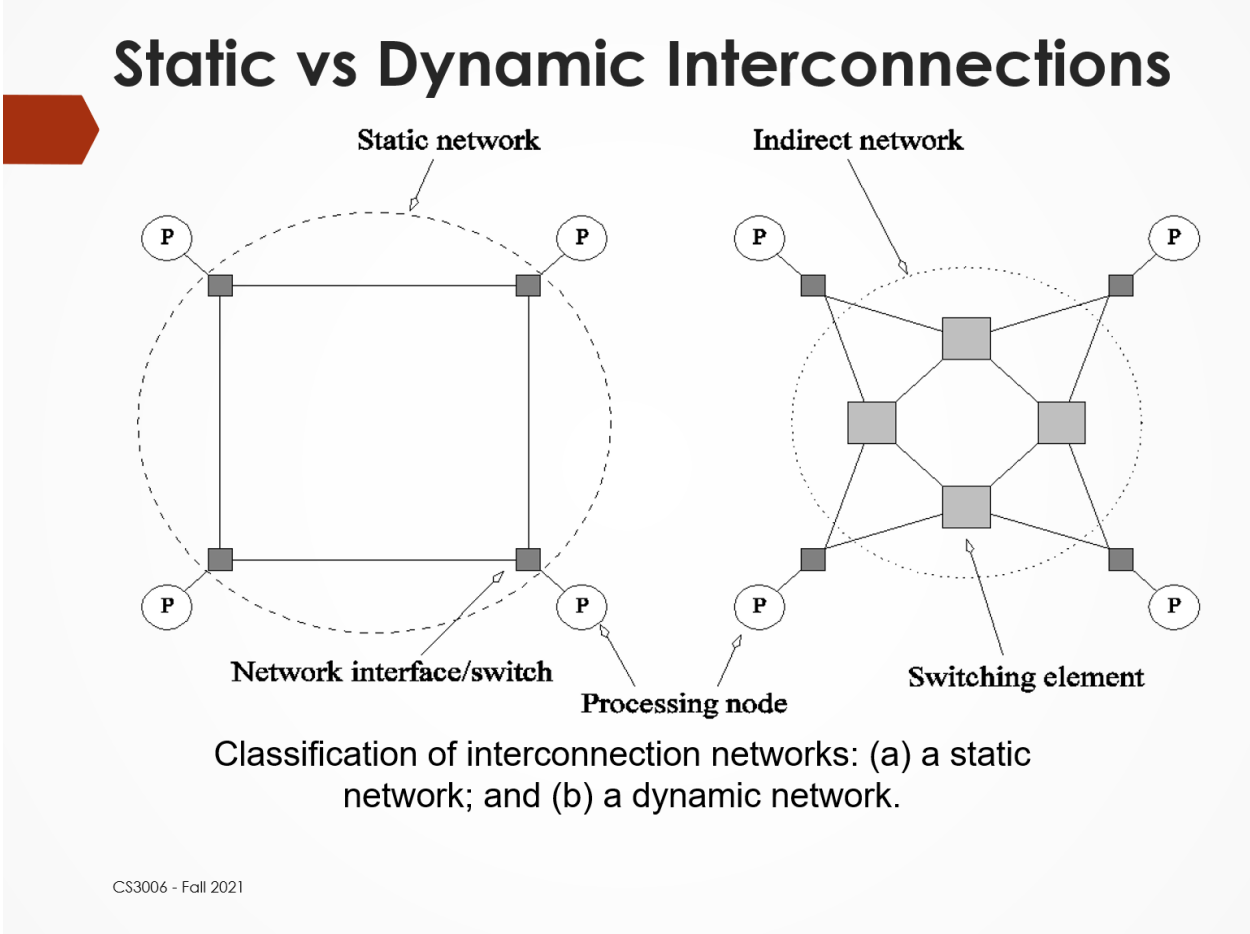
Summary Table (Exam-Friendly)

Topology	Connections	Advantages	Disadvantages	Example
Crossbar	(n^2)	No contention, very fast	Expensive for large n	Small high-speed multiprocessors
Multi-stage	(n \log_2 n)	Less expensive, multiple simultaneous connections	Shared paths → contention possible	Omega / Butterfly networks
Central Databus	1	Very simple, cheap	Only one CPU at a time → slow for large n	Early multiprocessors, microprocessors



Static: point to point

Dynamic: indirect



Interconnections in Parallel Computers

Purpose: Carry data between **processors** and **memory**.

Components:

- **Processing elements (PEs)** – the CPUs or nodes
- **Switches** – devices that route data
- **Links** – wires, fiber optics, or other connections

Two main types: Static and **Dynamic**

---

1. Static Interconnections (Direct Networks)

- **Definition:** Processors are connected by **fixed point-to-point links**.
- **Characteristics:**

    - Links are **fixed**
    - Each processor has a direct path to some other processors
    - Predictable, simple design
- **Advantages:**
    - Low complexity
    - Minimal control logic needed
- **Disadvantages:**
    - Less flexible
    - Poor scalability for large systems
- **Examples:**
    - **Bus network** – single line connecting all processors
    - **Ring network** – each processor connected to two neighbors
    - **Mesh / Hypercube** – processors connected in grid or cube patterns

Key point: "Direct paths exist between nodes, topology fixed"

---

2. Dynamic Interconnections (Indirect Networks)

- **Definition:** Use **switches** to connect processors to memory and to each other.
- **Characteristics:**
    - Links are **shared** and **programmable**
    - Paths can change dynamically depending on traffic
    - More complex hardware
- **Advantages:**
    - Flexible → can connect any processor to any memory module
    - Avoids contention better than static networks
- **Disadvantages:**
    - Higher hardware cost
    - More control logic needed
- **Examples:**
    - **Crossbar switch network** – any processor can connect to any memory module dynamically
    - **Multistage interconnection network (MSN)** – log2 n stages of switches connecting n processors and n memory modules

Key point: "Indirect paths via switches, topology can change dynamically"

Quick Comparison Table

Feature	Static (Direct)	Dynamic (Indirect)
Links	Fixed point-to-point	Shared via switches

Feature	Static (Direct)	Dynamic (Indirect)
Flexibility	Low	High
Complexity	Simple	Complex
Contention	Possible if links overlap	Reduced by dynamic routing
Example	Bus, Ring, Mesh, Hypercube	Crossbar, Multistage network

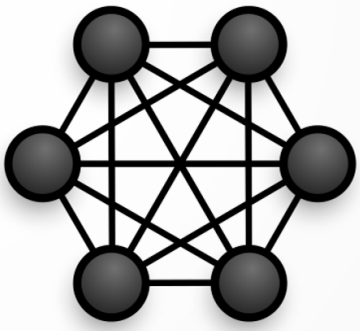
---

Tip for Exam:

- **Static → "direct, fixed"** → think of **grid/mesh connections**
- **Dynamic → "switch-based, flexible"** → think of **crossbar or switching network**

Network Topologies: Linear Arrays, Meshes, and *k-d* Meshes

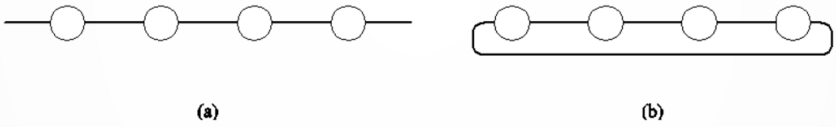
- Each processor is connected to every other processor (Complete connected network).
- The number of links in the network scales as $O(p^2)$.
- While the performance scales very well, the hardware complexity is not realizable for large values of p.
- Star connected networks



CS3006 - Fall 2021

Network Topologies: Linear Arrays, Meshes, and *k-d* Meshes

- In a linear array, each node has two neighbors, one to its left and one to its right.
- If the nodes at either end are connected, we refer to it as a 1-D torus or a ring.



Linear arrays: (a) with no wraparound links; (b) with wraparound link.

Network Topologies in Parallel Computing

Purpose: Connect processors so they can **communicate with each other and memory** efficiently.

---

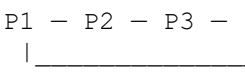
1. Linear Array (1-D Array)

- **Structure:** Each processor connected to **two neighbors** (left and right).
- **Advantages:** Simple, easy to build.
- **Disadvantages:** Long communication paths for distant nodes → slower communication.
- **Special Case:** If the two ends are connected → forms a **1-D Torus / Ring**.

Diagram: Linear Array

```
P1 — P2 — P3 — P4 — P5
```

Diagram: 1-D Torus / Ring



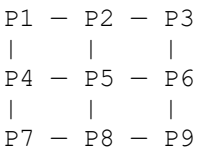
- **Example:** Small multiprocessor systems, token ring networks

---

2. Mesh Networks (2-D Mesh)

- **Structure:** Processors arranged in a **2D grid**; each processor connected to **up to 4 neighbors** (left, right, top, bottom).
- **Advantages:** Fault-tolerant, multiple paths → better performance than linear array.
- **Disadvantages:** More complex wiring.

Diagram: 2D Mesh (3x3)



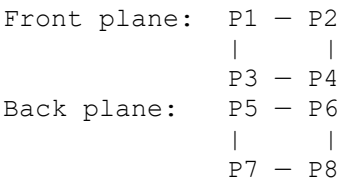
- **Example:** Cray supercomputers, parallel clusters

---

3. k-Dimensional Mesh (k-D Mesh / Hypermesh)

- **Structure:** Extension of 2D mesh to **k dimensions**.
- **Advantages:** Reduces **diameter** (maximum hops) → faster communication in large systems.
- **Disadvantages:** Wiring and hardware complexity increases with k.

Example: 3D Mesh for 8 processors: 2x2x2 cube



- **Use:** Modern high-performance clusters, supercomputers

---

4. Fully Connected Network

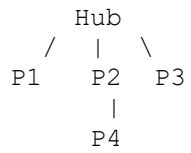
- **Structure:** Each processor connected to **every other processor**.
- **Links:** (O(p^2)) for p processors.
- **Advantages:** Maximum performance; direct communication between any pair.
- **Disadvantages:** Hardware not feasible for large p (complex wiring).
- **Example:** Small multiprocessor systems

---

5. Star Connected Network

- **Structure:** All processors connected to a **central hub**.
- **Advantages:** Simple, hub manages communication
- **Disadvantages:** Hub is a **single point of failure**, may become bottleneck
- **Example:** Small LANs with switch/hub

Diagram: Star Topology



Switch in Networks

- **Definition:** Device that **opens/closes access to memory/data banks**.
- **Purpose:** Helps **route messages** between processors efficiently.
- **Used in:** Dynamic / switch-based networks, multistage networks

Example: Crossbar switch or multistage interconnection network

---

Exam-Friendly Summary Table

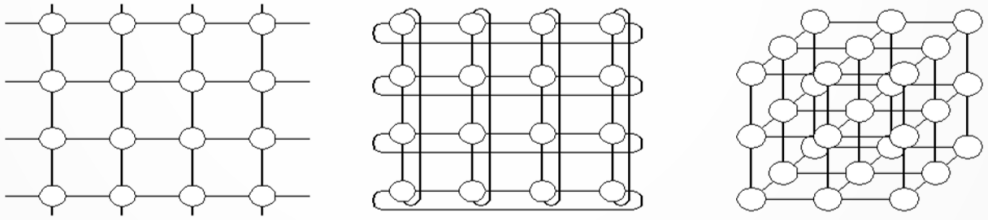
Topology	Connections	Advantages	Disadvantages	Example
Linear Array	2 neighbors	Simple	Slow for distant nodes	Small multiprocessor system
1-D Torus / Ring	Ends connected	Cyclic path, equal access	Long paths still exist	Token ring network
2D Mesh	Up to 4 neighbors	Fault tolerant, scalable	Wiring complexity	Cray supercomputer

Topology	Connections	Advantages	Disadvantages	Example
k-D Mesh	k-dim neighbors	Low diameter, fast	Complex wiring	Hypermesh cluster
Fully Connected	Every processor to every other	Maximum performance	Very high hardware cost	Small multiprocessor
Star	Central hub	Simple, manageable	Hub is bottleneck/failure point	LAN with switch

Network Topologies: Linear Arrays, Meshes, and *k-d* Meshes

Mesh

- A generalization has nodes with 4 neighbors, to the north, south, east, and west.
- A further generalization to *d* dimensions has nodes with 2*d* neighbors (i.e., 6 neighbors in case of 3d cube).



(a) (b) (c)

Two and three dimensional meshes: (a) 2-D mesh with no wraparound; (b) 2-D mesh with wraparound link (2-D torus); and (c) a 3-D mesh with no wraparound.

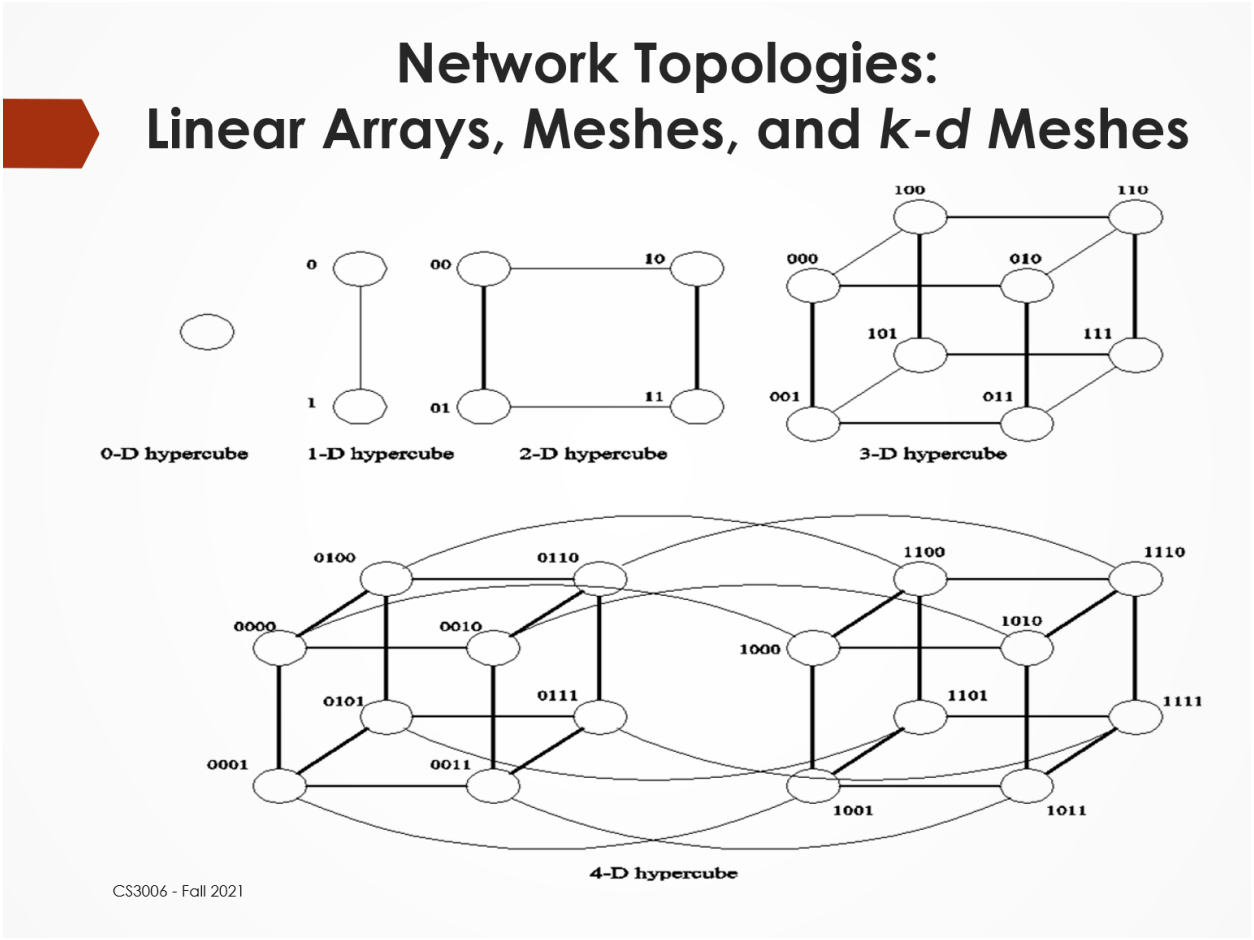
CS3006 - Fall 2021

3d-weather modeling, 3d-structure modeling

Network Topologies: Linear Arrays, Meshes, and *k-d* Meshes

Hypercube

- The hypercube has two nodes along each dimension except *0d* hypercube.
- *d = log p (dimensions = log(nodes))*
- The distance between any two nodes is at most *log p*.
- Each node has *log p* neighbors.
- The distance between two nodes is given by the number of bit positions at which the two nodes differ.
- Rule of thumb is: "d-dimensional hypercube can be constructed by connecting corresponding nodes of two (d-1)-dimensional hypercubes"



---

1. Mesh Networks

- **Structure:** Processors arranged in a **grid**, each processor connected to **its neighbors**.
- **2D Mesh:** Each node has **up to 4 neighbors** (north, south, east, west).
- **3D Mesh:** Each node has **6 neighbors** (front/back, left/right, top/bottom).
- **k-D Mesh:** Each node has **2k neighbors**, generalizing to k dimensions.

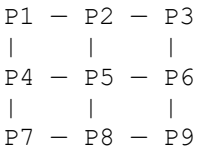
---

2. Mesh Variants

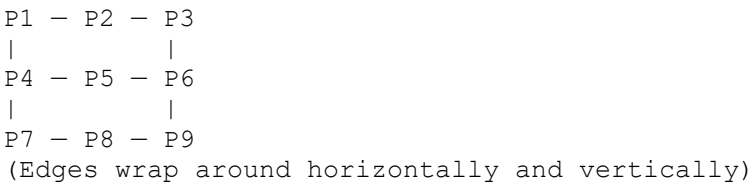
Variant	Structure	Notes
2D Mesh	Grid, no wraparound	Edges have fewer neighbors
2D Torus	Grid, wraparound connections at edges	Reduces diameter, improves communication
3D Mesh	Cube, no wraparound	Each node has 6 neighbors in 3D space

Example Diagrams:

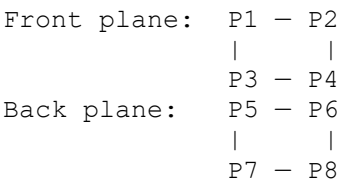
2D Mesh (3x3, no wraparound)



2D Torus (wraparound)



3D Mesh (2x2x2 cube)

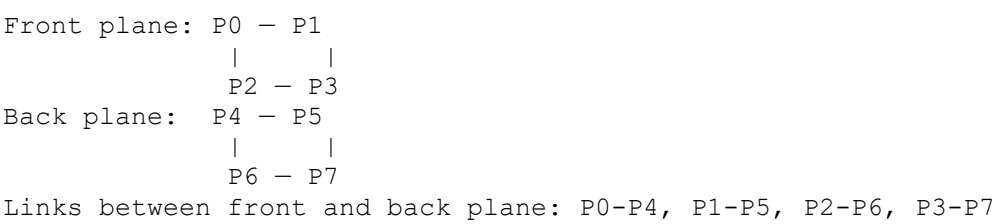


---

2. Hypercube Network

- **Definition:** Multi-dimensional generalization where **d = log2 p**, p = number of nodes.
- **Properties:**
  - Each node has **d neighbors**
  - **Maximum distance** between nodes = d
  - Distance = **number of differing bits** in node addresses (Hamming distance)
- **Construction Rule:**
  - A **d-dimensional hypercube** can be built by connecting corresponding nodes of **two (d-1)-dimensional hypercubes**.

Example: 3D Hypercube (8 nodes, d=3)



Distance between nodes:

- Node 010 and 110 → differ in 1 bit → distance = 1

Rule of Thumb:

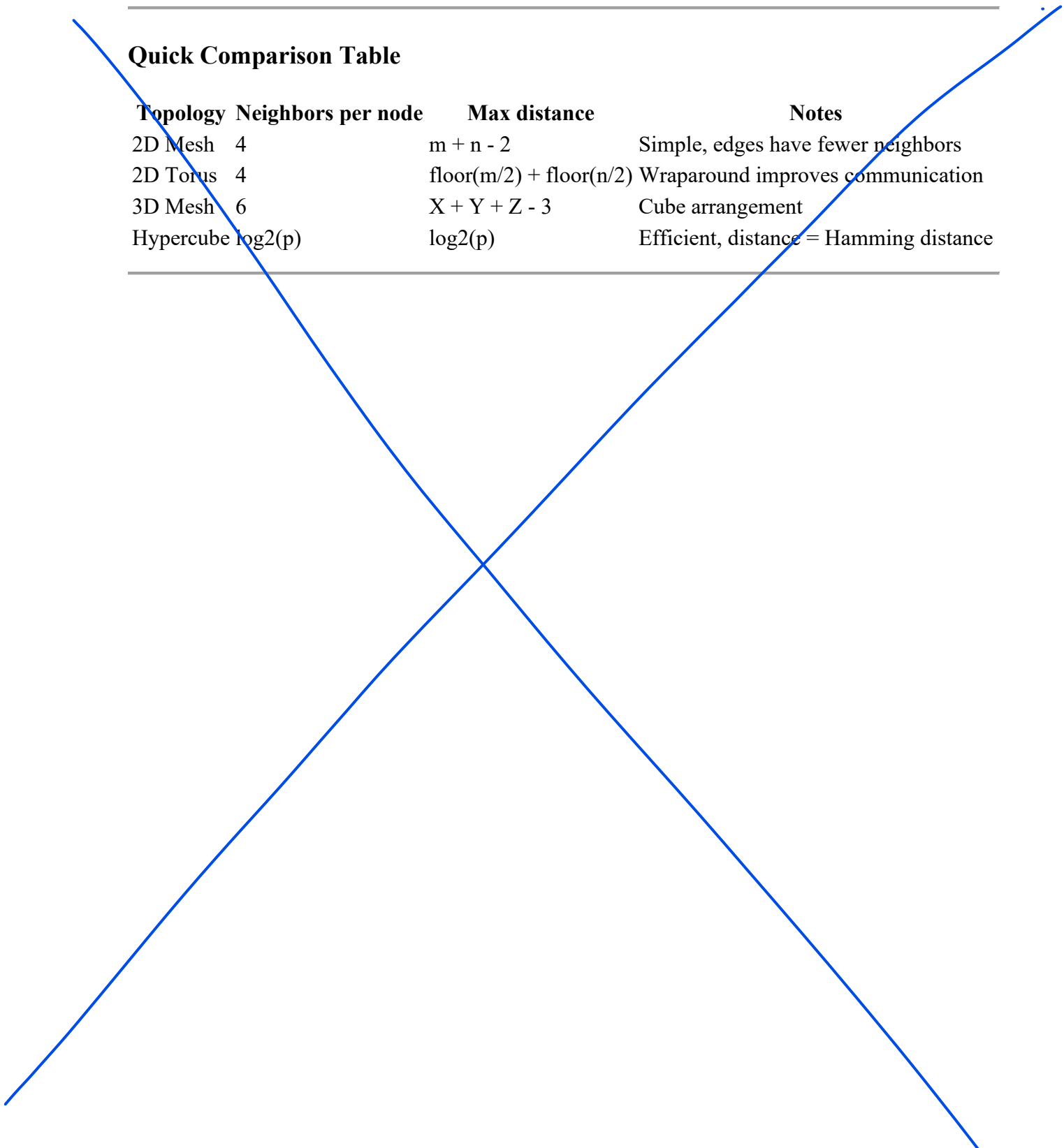
- Hypercube has **log2(p) neighbors** per node
- Maximum hops between any two nodes = **log2(p)** → efficient communication

---

Quick Comparison Table

Topology	Neighbors per node	Max distance	Notes
2D Mesh	4	m + n - 2	Simple, edges have fewer neighbors
2D Torus	4	floor(m/2) + floor(n/2)	Wraparound improves communication
3D Mesh	6	X + Y + Z - 3	Cube arrangement
Hypercube	log2(p)	log2(p)	Efficient, distance = Hamming distance

---

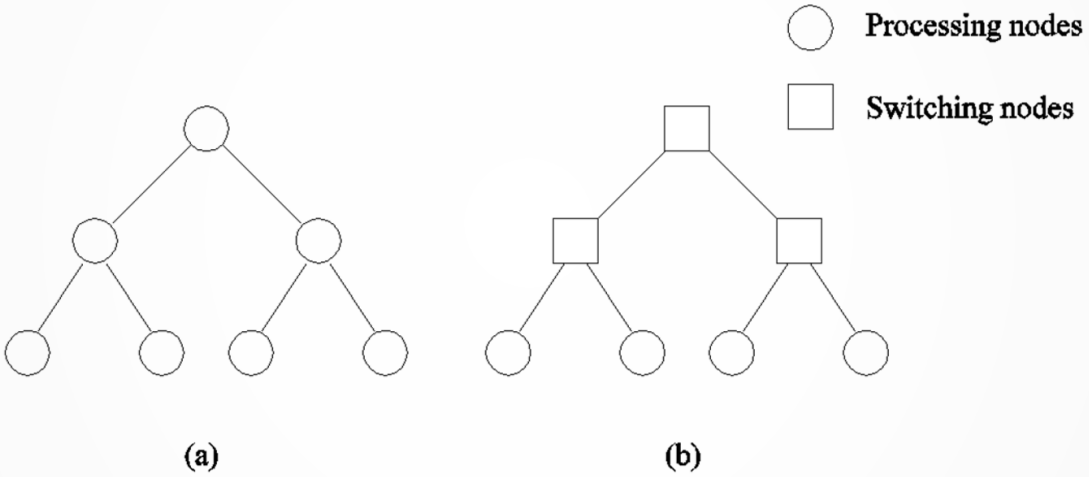


Network Topologies: Tree based Networks

- A tree network is one in which there is one path between any pair of nodes
- Linear arrays and star-connected networks are special cases of tree-based networks
- In static tree network, each node represent a processing element
- In dynamic tree network, leaf nodes represent processing element while internal nodes are switching elements.
- The source node sends the message up the tree until it reaches the node at the root of the smallest subtree containing both the source and destination nodes.

Network Topologies: Tree based Networks

Complete Binary Tree

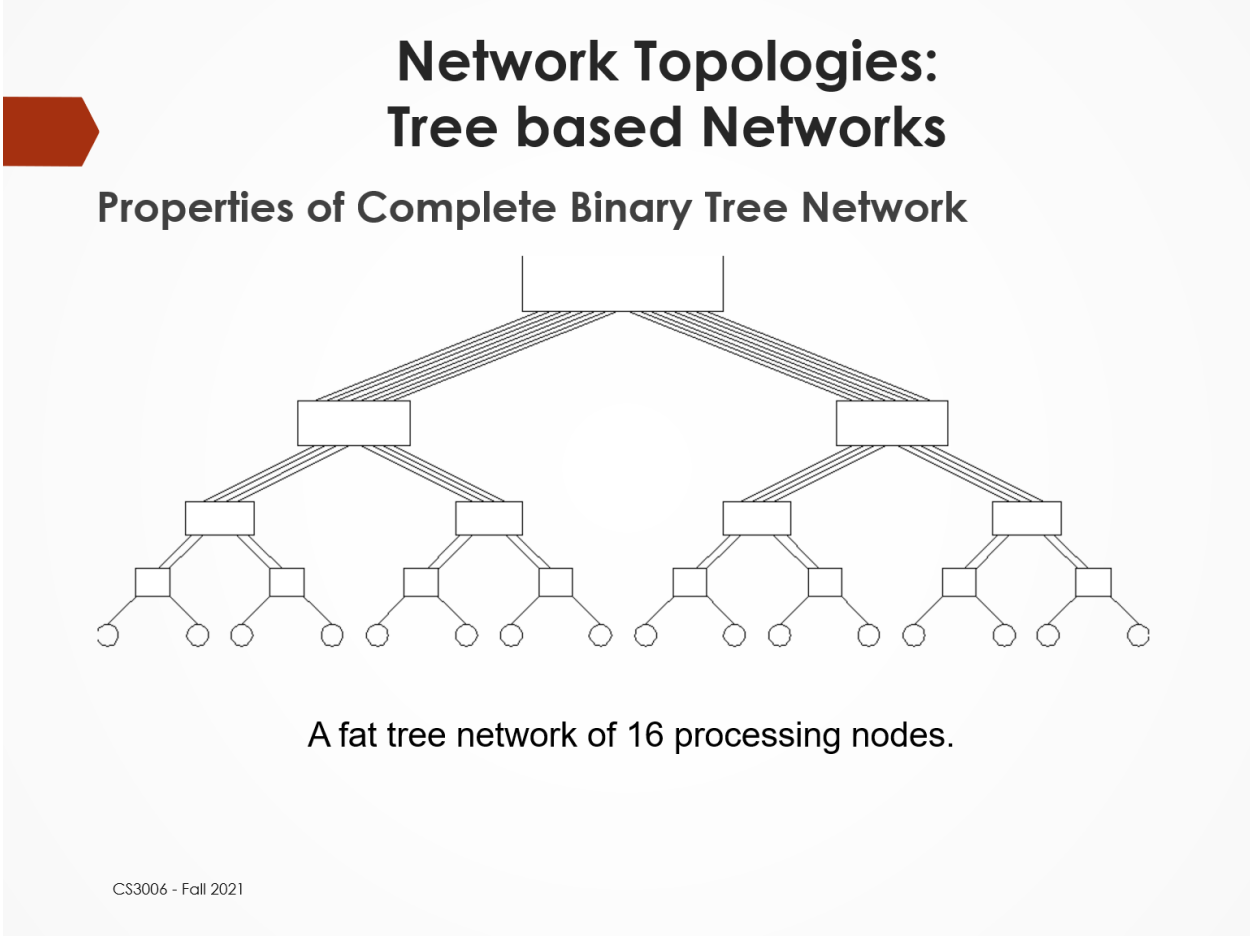


Complete binary tree networks: (a) a static tree network; and (b) a dynamic tree network.

Network Topologies: Tree based Networks

Properties of Complete Binary Tree Network

- The distance between any two nodes is no more than $2 log p$.
- Links higher up the tree potentially carry more traffic than those at the lower levels.
- For this reason, a variant called a fat-tree, fattens the links as we go up the tree.
- Trees can be laid out in 2D with no wire crossings. This is an attractive property of trees.



Tree-Based Networks

Definition:
A network organized like a **tree**, where there is **exactly one path between any two nodes**.

- **Special cases:**
  - Linear arrays → degenerate tree
  - Star-connected networks → root at center

---

1. Static Tree Network

- **Structure:** Each node = **processing element (PE)**
- **Use:** Direct communication between PEs along tree paths
- **Message Routing:**
  - Source sends message **up the tree** until it reaches **lowest common ancestor** (smallest subtree containing source & destination)
- **Example:** Early hierarchical multiprocessor systems

---

2. Dynamic Tree Network

- **Structure:**
  - **Leaf nodes = processing elements**
  - **Internal nodes = switches**
- **Use:** Leaf PEs communicate via **switching internal nodes**
- **Advantage:** Efficient use of switches, reduces hardware cost

---

Properties of Complete Binary Tree Network

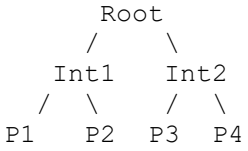
- **Distance:** Maximum distance between any two nodes = (2 \log p), where p = number of leaf nodes
- **Traffic:** Links near root carry **more traffic** than lower-level links
- **Wire layout:** Can be laid out in 2D with **no wire crossings** → simplifies hardware design

---

Fat Tree

- **Definition:** A **variant of tree** that **fattens the links** as you go up the tree
- **Purpose:**
  - Links near root handle **more traffic** → reduce congestion
- **Use:** Modern parallel and cluster networks

Diagram (Simplified 8-node binary tree):



- **Static:** All nodes are PEs
- **Dynamic:** Internal nodes (Int1, Int2) are switches, leaf nodes are PEs

---

Key Points for Exam

Feature	Tree Network	Fat Tree
Path	One path between any two nodes	Same, but higher links widened

Feature	Tree Network	Fat Tree
Max Distance	2 log p	Same
Traffic	Higher near root	Reduced by widening links
Layout	Can be 2D, no wire crossings	Same
Use	Hierarchical multiprocessors	Modern clusters and supercomputers

---

Tip:

- Remember: **Linear array & Star → special tree cases**
- Fat tree → “fattens the links near root” → handles more traffic

Evaluating Static Interconnections

The parameters to evaluate a static interconnection:-

- ***Cost:*** Usually depends on number of links for communication. E.g., cost for linear array is *p-1*.
  - *Lower values are favorable*
- ***Diameter:*** The shortest distance between the farthest two nodes in the network. The diameter of a linear array is $p - 1$.
  - *Lower values are favorable*
- ***Bisection Width:*** The minimum number of wires you must cut to divide the network into two equal parts. The bisection width of a linear array is *1*.
  - What it tells about performance of a topology?

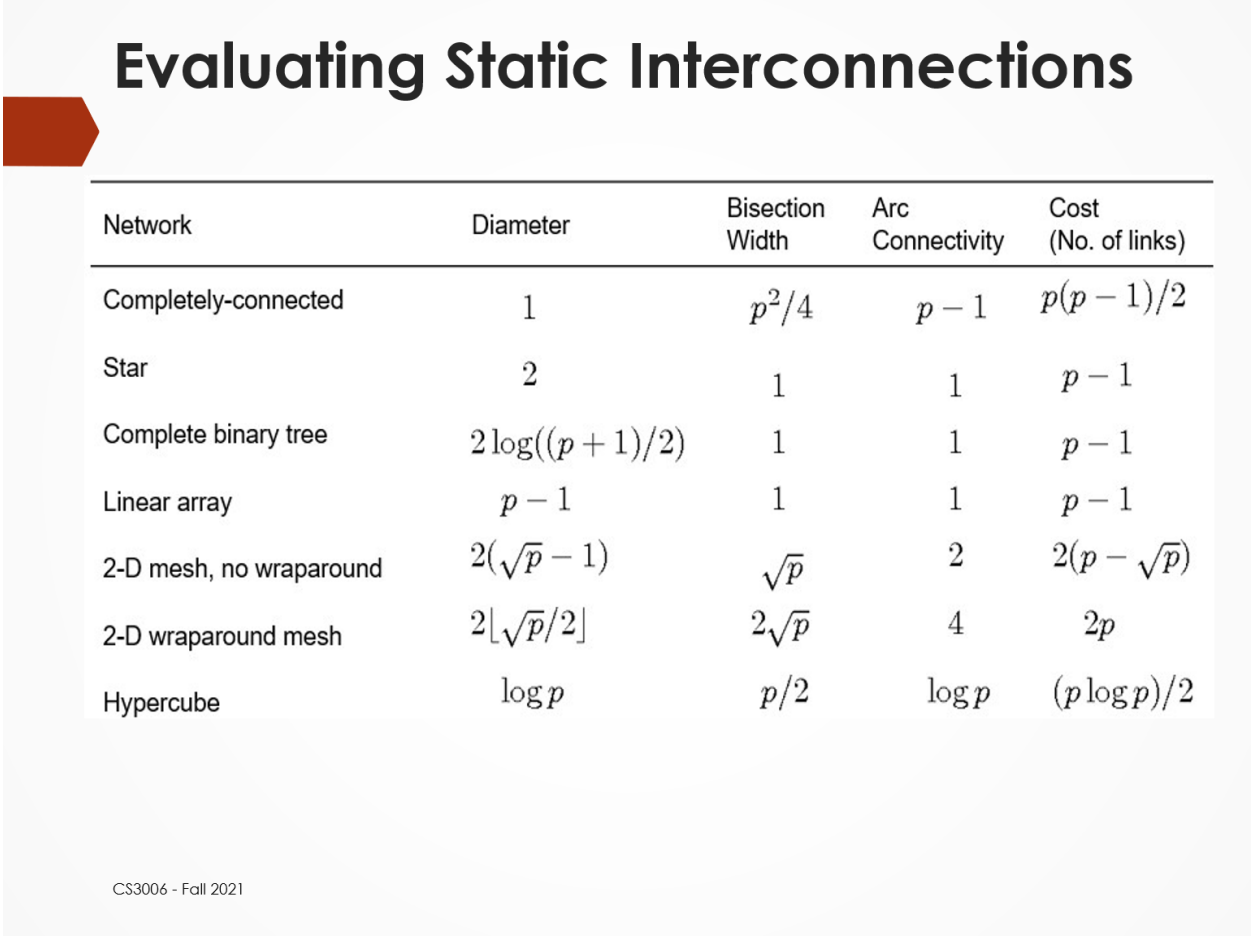
Evaluating Static Interconnections

The parameters to evaluate a static interconnection:-

- ***Arc-connectivity:*** *The minimum number of arcs or links that must be removed from the network, to break the network into two disconnected networks*
  - *Higher value are desirable*
  - It is minimum number of the links that must be cut to separate the single node from the network
  - Higher values means, that incase of link failure there are multiple other routes to the node.
  - Arc-connectivity of linear array is *1* and *2* for ring.

Bisection width is related to available bandwidth of the network when first half of network communicates with second half of the network

Our performance increases as no. of dimensions increase



→ log(p+1)-1 floors of links in binary trees→ **Exponential formula**=2^0 + 2^1+…+2^n=2^(n+1) -1

→ To solve hypercube induce concept of directed links

Star bisection= not $\lfloor p/2 \rfloor$ **but** 1 because we only have one link to transfer data to second half

Determine expressions for all the interconnections

Discuss scalability issues→ How adding or removing a node effects overall network

Evaluating Static Interconnections

When analyzing a **network topology**, we check how efficient it is using several parameters:

---

1. Cost

- **Definition:** Total number of **communication links** in the network.

- **Formula / Example:**
  - Linear array with p nodes → cost = p − 1 links
  - Star network → cost = p (each node to central hub)
- **Interpretation: Lower cost → cheaper hardware**

---

2. Diameter

- **Definition: Longest shortest path** between any two nodes in the network.
- **Example:**
  - Linear array with p nodes → diameter = p − 1
  - Ring → diameter = floor(p / 2)
- **Interpretation: Lower diameter → faster communication**

---

3. Bisection Width

- **Definition:** Minimum number of links you must **cut to divide the network into two equal halves**.
- **Example:**
  - Linear array → 1
  - Ring → 2
- **Interpretation:** Related to **available bandwidth** between two halves of the network.
  - **Higher bisection width → better parallel performance**
  - Networks with more dimensions (e.g., 2D or 3D meshes) → larger bisection width → better performance

---

4. Arc-Connectivity

- **Definition:** Minimum number of **links that must be removed to disconnect the network** or isolate a node.
- **Example:**
  - Linear array → 1
  - Ring → 2
- **Interpretation:**
  - **Higher value → fault-tolerant**, multiple alternative routes exist if a link fails

---

5. Special Notes / Formulas

- **Binary Tree:**

    - Levels = log(p + 1) − 1
    - Total links = (2^0 + 2^1 + ... + 2^n = 2^{n+1} - 1)
- **Hypercube:**
    - Can use **directed links** for easier analysis
    - Each node has log(p) neighbors
    - Diameter = log(p)

Quick Comparison Table (Linear Array, Ring, 2D Mesh, Hypercube)

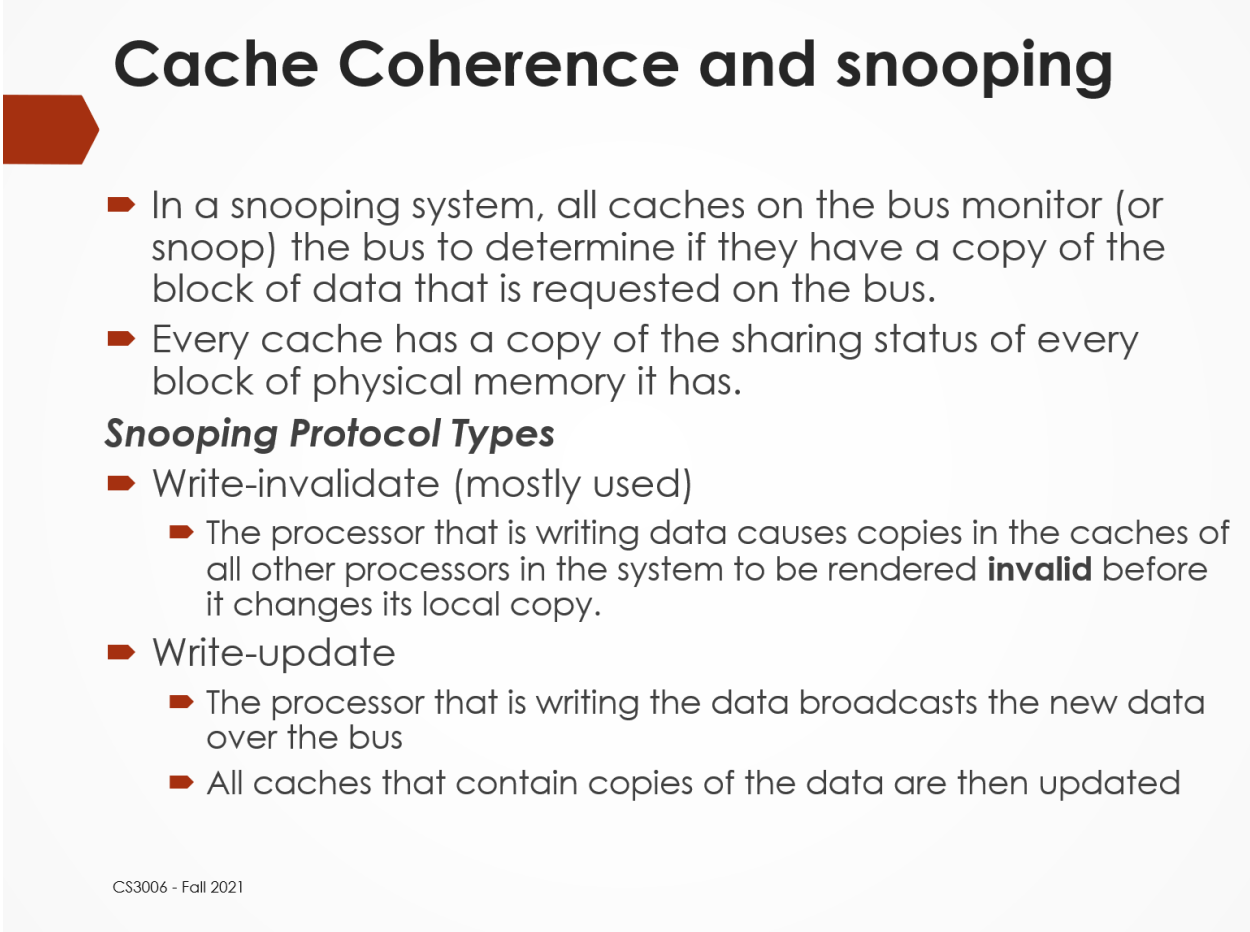
Topology	Cost (Links)	Diameter	Bisection Width	Arc-Connectivity	Notes
Linear Array	p − 1	p − 1	1	1	Simple, cheap, slow for distant nodes
Ring	p	floor(p/2)	2	2	Wraparound reduces diameter, better fault tolerance
2D Mesh (√p x √p)	2√p(√p − 1)	2(√p − 1)	√p	2	More paths → better bandwidth
Hypercube	p log p / 2	log p	p/2	log p	Very scalable, low diameter

---

Exam Tips

- **Cost:** "How many wires do I need?" → cheaper = better
- **Diameter:** "How far is the farthest processor?" → smaller = faster
- **Bisection Width:** "How many wires cut if half network talks to other half?" → bigger = better performance
- **Arc-Connectivity:** "How robust is the network?" → bigger = more fault-tolerant

---



Cache Coherence

Problem:

- In a multiprocessor system, each processor has its **own cache**.
- **Issue:** If two or more caches store the **same memory block**, updates in one cache **may not be reflected** in others → data inconsistency.

Goal: Ensure that **all processors see a consistent view of memory**.

Snooping System

- **Definition:** Each cache **monitors (snoops) the bus** to detect if any other processor is reading or writing a block it has.
- **Mechanism:**

- Each cache keeps track of the **sharing status** of its memory blocks.
- If a cache sees another processor accessing the block, it can **take action** to maintain coherence.

Snooping Protocol Types

1. Write-Invalidate (Mostly Used)

- **How it works:**
  1. Processor P1 wants to write to a block.
  2. It sends an **invalidate signal** on the bus.
  3. All other caches that have this block **mark it invalid**.
  4. P1 updates its local cache.
- **Advantages:**
  - Reduces **bus traffic** (only invalidates, doesn't send data)
  - Good for **write-heavy workloads**
- **Example:** P1 writes to variable X → all other caches mark X as invalid

---

2. Write-Update (Write-Broadcast)

- **How it works:**
  1. Processor P1 writes new data to its cache.
  2. It **broadcasts the new value** over the bus.
  3. All caches that have this block **update their copies**.
- **Advantages:**
  - Ensures all caches always have **latest value**
- **Disadvantages:**
  - More **bus traffic** (every write is broadcasted)

Summary Table

Protocol	Action on Other Caches	Bus Traffic	Use Case
Write-Invalidate	Invalidate copies	Low	Most systems, write-heavy workloads
Write-Update	Update copies with new value	High	Read-heavy workloads, all caches need latest data

---

Tip for Exam:

- **Invalidate → "others lose their copy"**
- **Update → "others get the new value"**
