Parallel Programming

Parallel Architectures

Diego Fabregat-Traver and Prof. Paolo Bientinesi

HPAC, RWTH Aachen
fabregat@aices.rwth-aachen.de

WS15/16

Parallel Architectures

Acknowledgements

- Prof. Felix Wolf, TU Darmstadt
- Prof. Matthias Müller, ITC, RWTH Aachen

References

- Computer Organization and Design. *David A. Patterson, John L. Hennessy*. Chapter 7.
- Computer Architecture: A Quantitative Approach. *John L. Hennessy, David A. Patterson*. Appendix F.

Outline

Flynn's Taxonomy

According to instructions and data streams

- *Single instruction stream, single data stream* (SISD): Classical single-core processor
- *Single instruction stream, multiple data stream* (SIMD): Vector extensions, GPUs
- *Multiple instruction stream, single data stream* (MISD): No commercial processor exists
- *Multiple instruction stream, multiple data stream* (MIMD): Multi-core, multi-processors, clusters

Flynn's Taxonomy

According to instructions and data streams

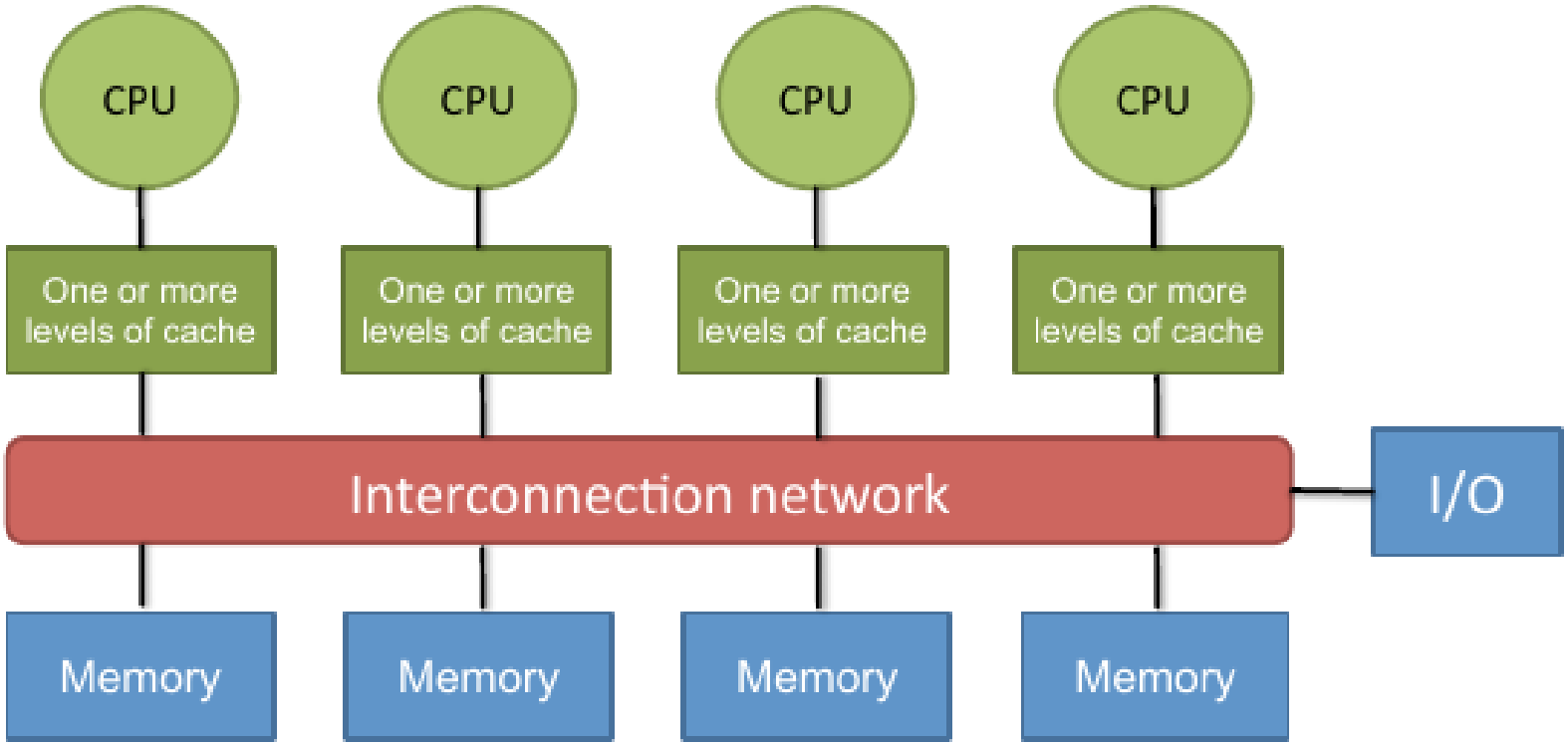
- *Single instruction stream, single data stream* (SISD): Classical single-core processor
- *Single instruction stream, multiple data stream* (SIMD): Vector extensions, GPUs
- *Multiple instruction stream, single data stream* (MISD): No commercial processor exists
- *Multiple instruction stream, multiple data stream* (MIMD): Multi-core, multi-processors, clusters

From a parallel programming perspective, only two are relevant: SIMD and MIMD. Focus of this course: MIMD.

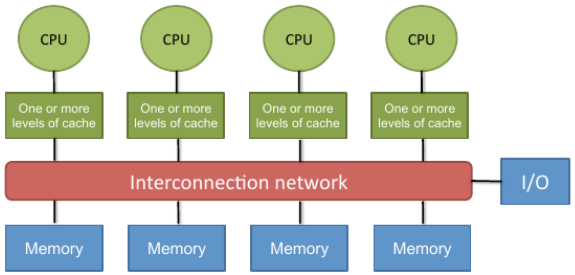
Multiple-Instruction Multiple-Data

- Most general model: Each processor works on its own data with its own instruction stream
- In practice: Single Program Multiple Data (SPMD)
  - All processors execute the same code stream
  - Just not the same instruction at the same time
  - Control flow relatively independent (can be completely different)
  - Amount of data to process may vary
- Further breakdown based on memory organization:
  - Shared-memory systems
  - Distributed-memory systems

Shared Memory Multiprocessors



Shared Memory Multiprocessors



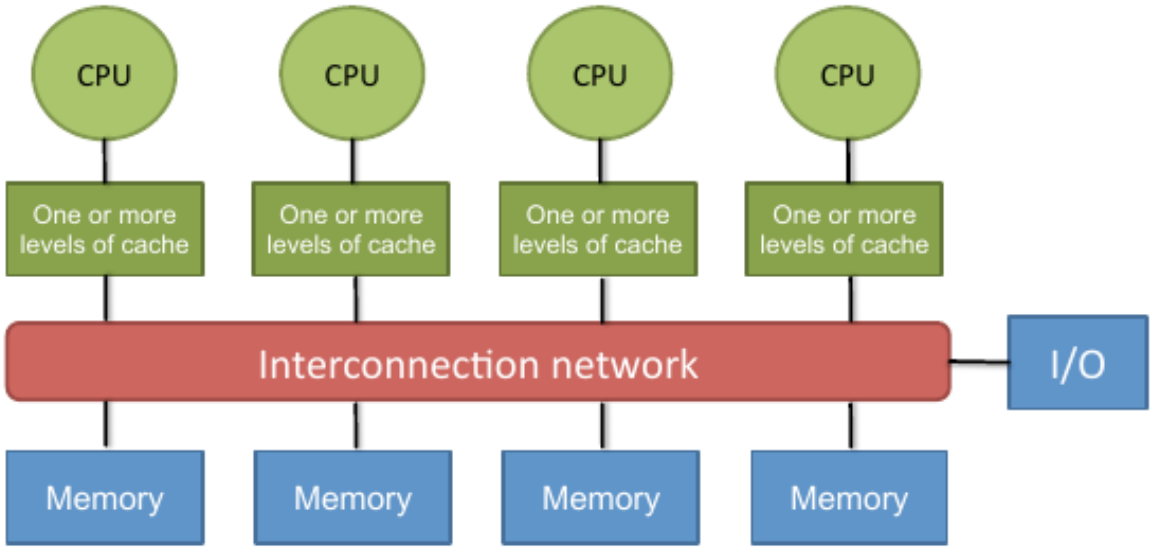
- Programmer's view: Single physical address space
- Processors communicate via shared variables in memory
- All processors can access any location via loads and stores
- Usually come in one of two flavors:
  - Uniform memory access(UMA)
  - Nonuniform memory access(NUMA)

Shared Memory Multiprocessors

Uniform Memory Access (UMA)

UMA

- About the same time to access main memory
- Does not matter which processor and address

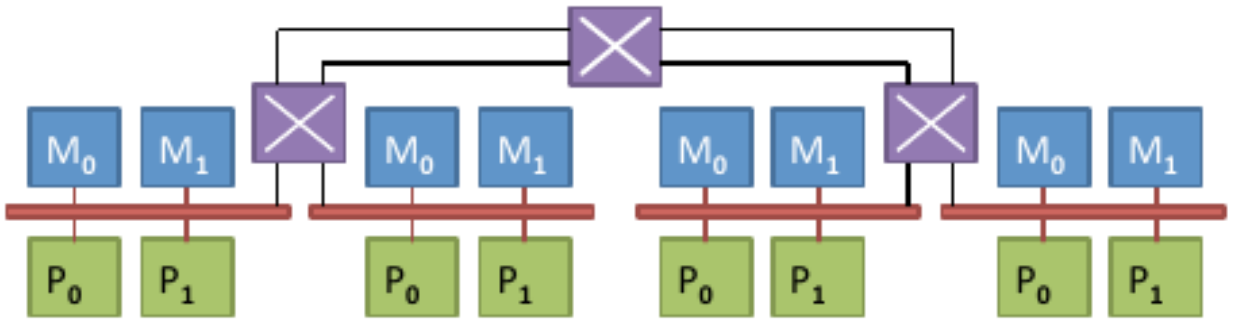


Shared Memory Multiprocessors

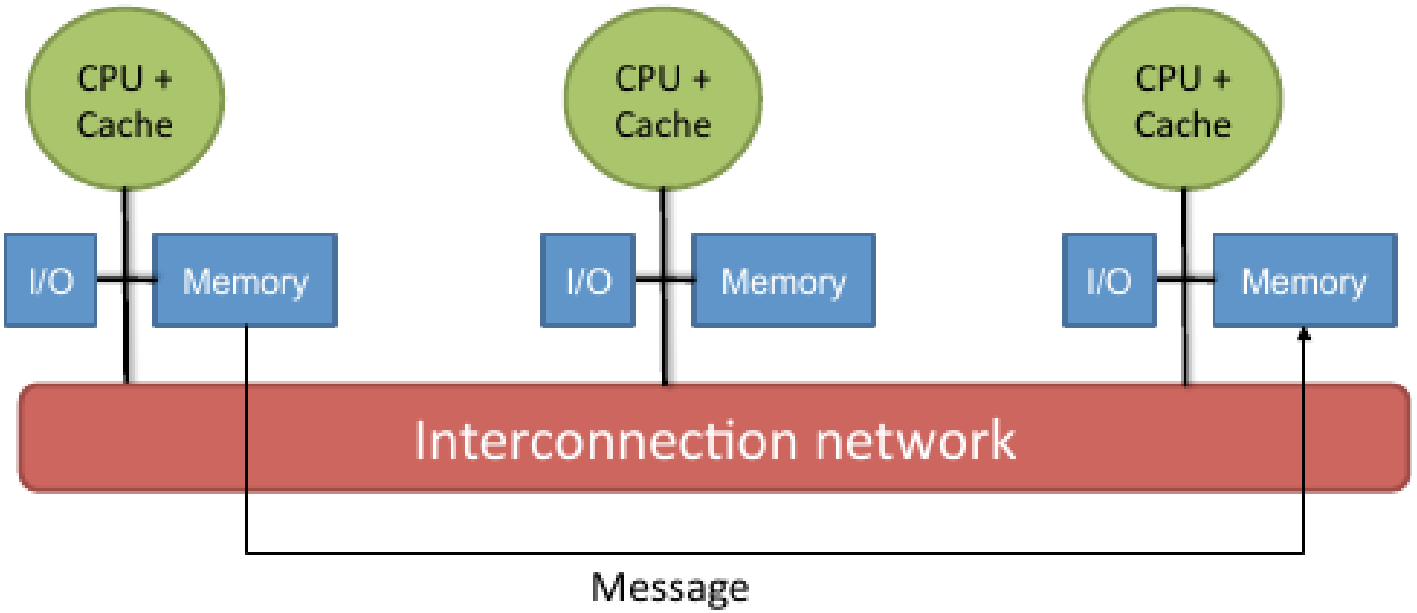
Nonuniform Memory Access (NUMA)

NUMA

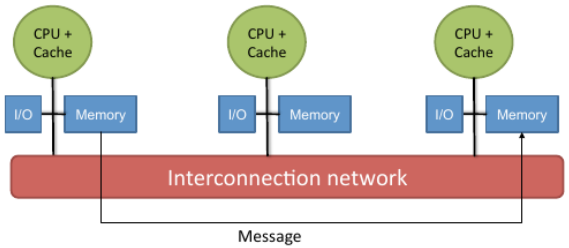
- Some memory accesses are faster than others
- Depends on which processor accesses which word



Cluster of Processors



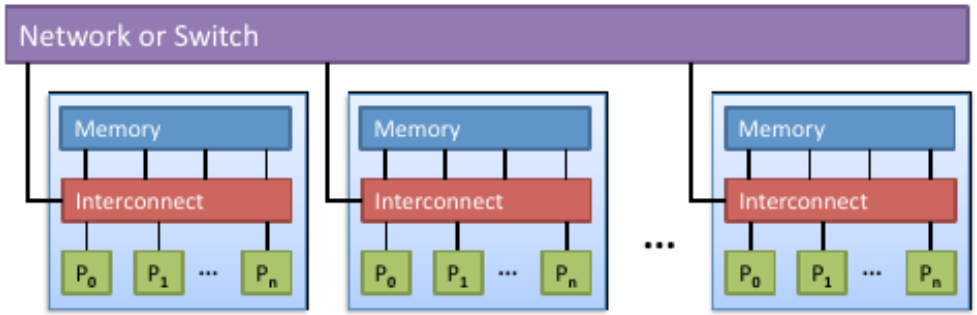
Cluster of Processors



- Each processor (node) has its own private address space
- Processors communicate via *message passing*
- Coordination/Synchronization via send/receive routines

Cluster of (Multi)Processors

Nowadays, we typically find hybrid configurations:

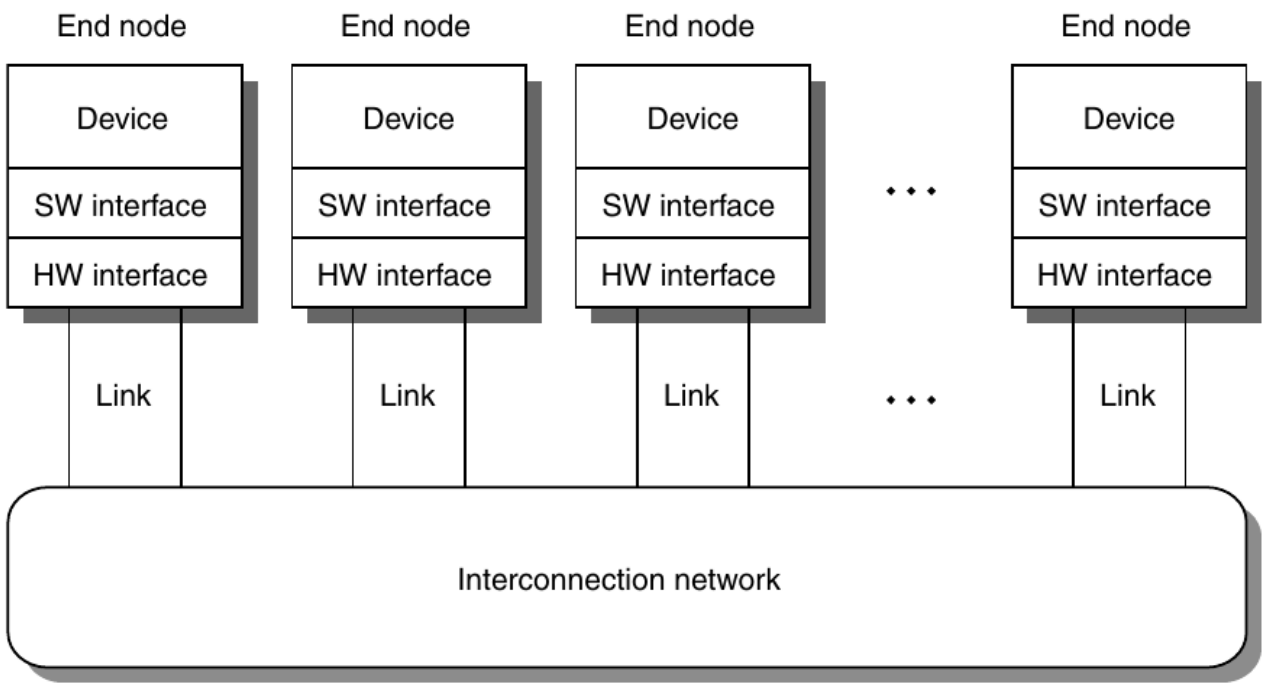


- Commodity clusters:
  - Standard nodes
  - Standard interconnection network
- Custom clusters:
  - Custom nodes
  - Custom interconnection network
  - Example: IBM BlueGene

Outline

Interconnection Networks

- Components of a network:
  - Nodes
  - Links
  - Interconnection

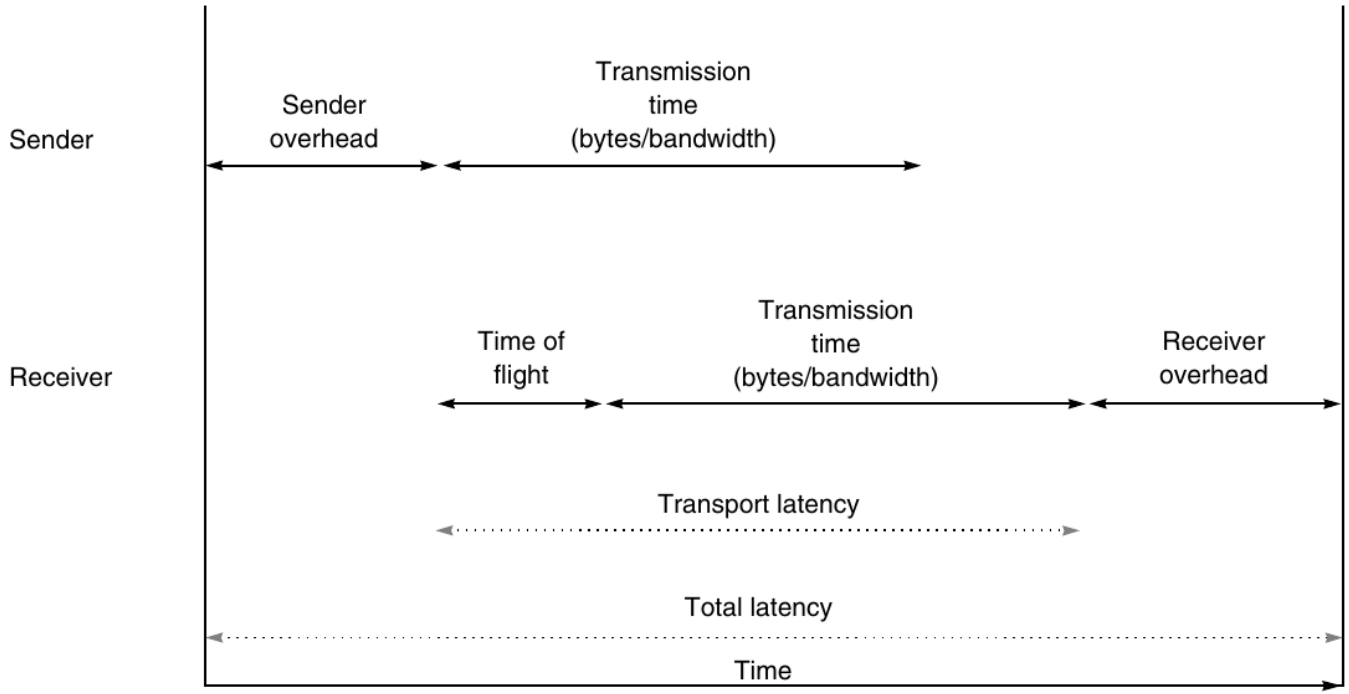


Network Performance

- Bandwidth
  - Maximum rate at which data can be transfered
  - Aggregate bandwidth: total data bw supplied by the network
  - Effective bandwidth (throughput): fraction of the aggregate bandwidth delivered to an application

Network Performance

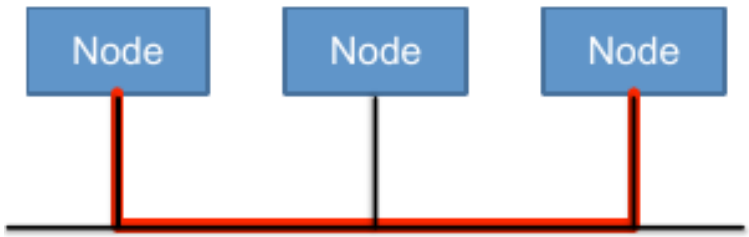
- Latency: Time to send and receive a message



Components of packet latency. Source: *Computer architecture, Appendix F.* Patterson, Hennessy.

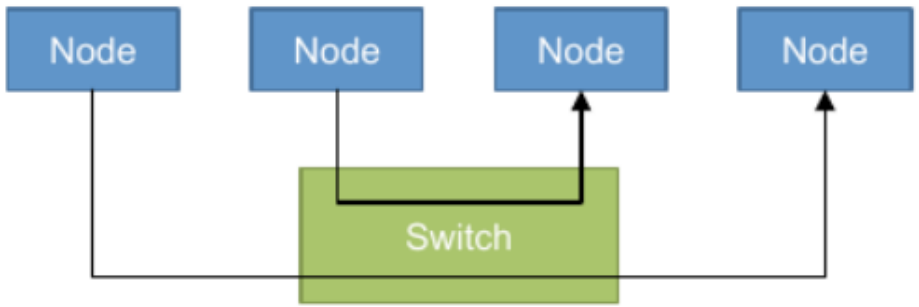
Shared-media networks

- Only one message at a time
  - Processors broadcast their message over the medium
- Each processor listens to every message and receives the ones for which it is the destination
- Decentralized arbitration
  - Before sending a message, processors listen until medium is free
- Message collision can degrade performance
- Low cost but does not scale
- Example: bus networks to connect processors to memory



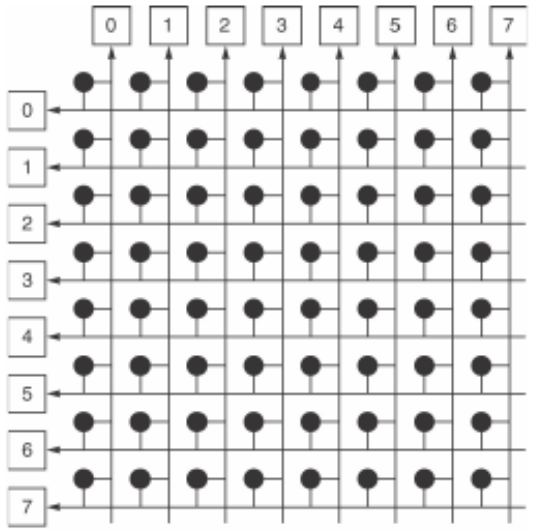
Switched-media networks

- Support point-to-point messages between nodes
- Each node has its own communication path to the switch
- Advantages
  - Support concurrent transmission of multiple messages among different node pairs
  - Scales much more than bus



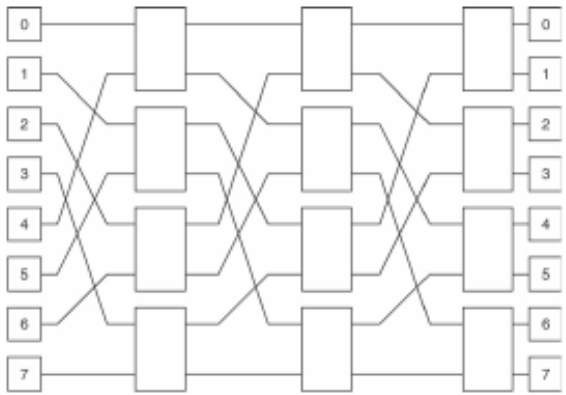
Crossbar switch

- Non-blocking
  - Links are not shared among paths to unique destinations
- Requires n^2 crosspoint switches
  - Limited scalability



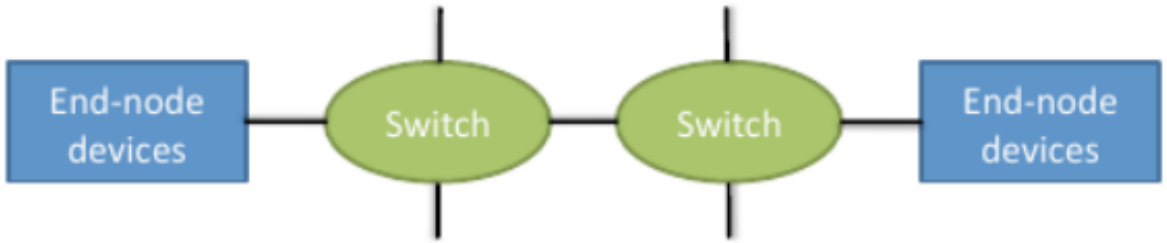
Omega network

- Multi-stage interconnection network (MIN)
- Splits crossbar into multiple stages with simpler switches
- Complexity $O(nlog(n))$
- Omega with $k \times k$ switches
  - $log_k(n)$ stages
  - $\frac{n}{k} log_k(n)$ switches
- Blocking due to paths between different sources and destinations simultaneously sharing network links



Distributed switched networks

- Each network switch has one or more end node devices directly attached to it
- Mostly used for distributed-memory architectures
  - End-node devices: processor(s) + memory
  - Network node: end-node + switch
- These nodes are directly connected to other nodes without going through external switches
  - Also called direct or static interconnection networks
  - Ratio of switches to nodes 1:1



Evaluation criteria

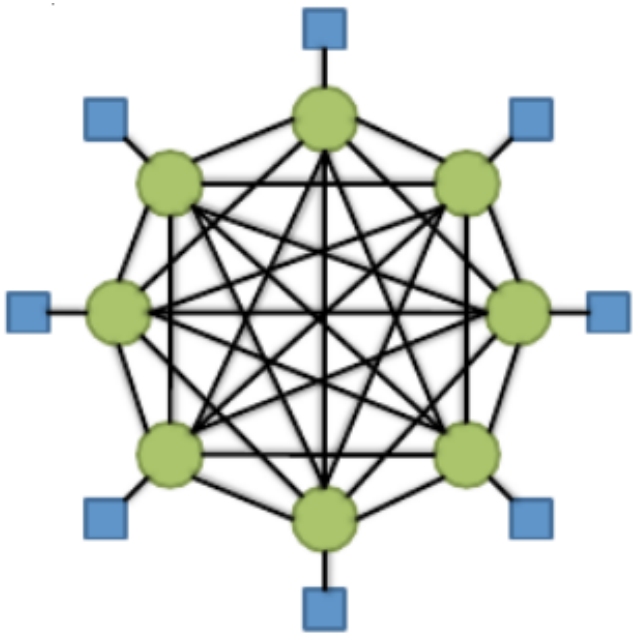
- Network degree
  - Maximum node degree
  - Node degree: number of adjacent nodes (in + out edges)
- Diameter
  - Largest distance between two nodes
- Bisection width (or bisection bandwidth)
  - Minimum number of edges between nodes that must be removed to cut the network into roughly two equal halves
- Edge/node connectivity
  - Minimum number of edges/nodes that need to be removed to render network disconnected

Requirements

- Low network degree to reduce hardware costs
- Low diameter to ensure low distance (i.e., latency) for message transfer
- High bisection bandwidth to ensure high-throughput
- High connectivity to ensure robustness
- Good scalability to connect large numbers of device nodes

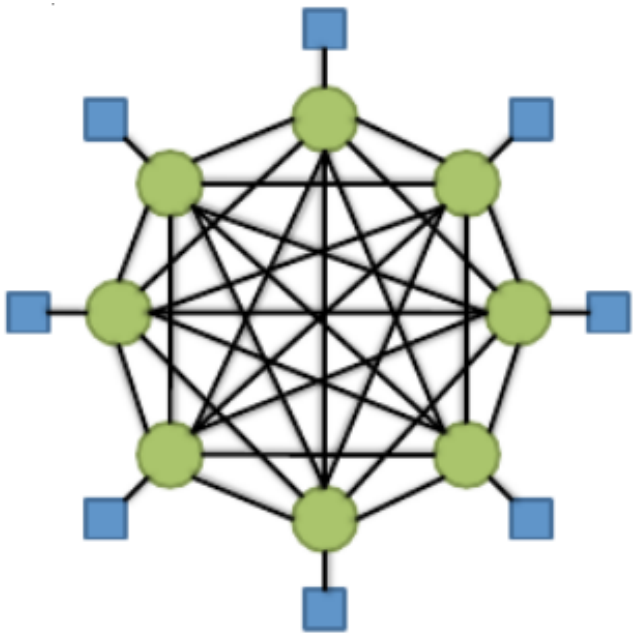
Fully connected topology

- Each node is directly connected to every other node
- Expensive for large numbers of nodes
- Dedicated link between each pair of nodes



Fully connected topology

- Each node is directly connected to every other node
- Expensive for large numbers of nodes
- Dedicated link between each pair of nodes



Assuming 64 nodes

Performance:

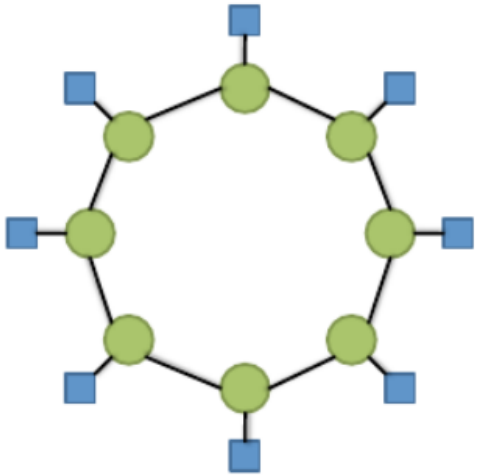
Diameter: 1	✓
$BW_{Bisection}$ (# links): 1024	✓
Edge connectivity: 63	✓

Cost:

# Switches: 64	✗
Network degree: 64	✗
# links: 2080	✗

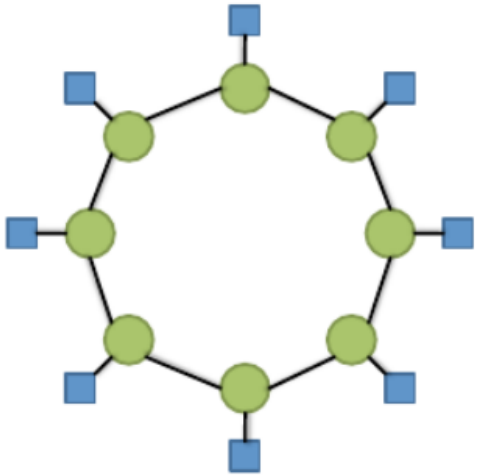
Ring topology

- Lower-cost
- n 3 $\times$ 3 switches, n network links
- Not a bus! simultaneous transfers possible



Ring topology

- Lower-cost
- n 3 $\times$ 3 switches, n network links
- Not a bus! simultaneous transfers possible



Assuming 64 nodes

Performance:

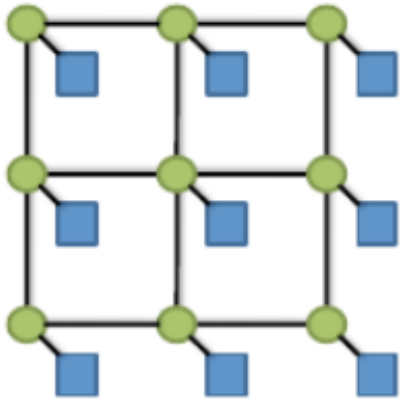
Diameter: 32	✗
$BW_{Bisection}$ (# links): 2	✗
Edge connectivity: 2	✗

Cost:

# Switches: 64	✓
Network degree: 3	✓
# links: 128	✓

N-dimensional meshes

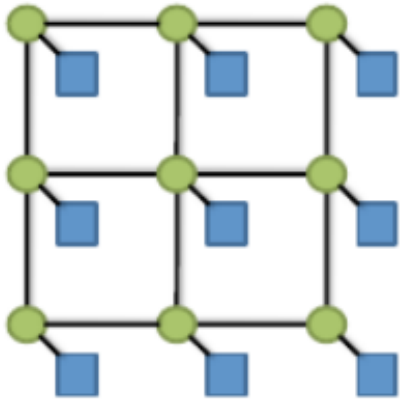
- Typically 2 or 3 dimensions
- Direct link to neighbors
- Each node has 1 or 2 neighbors per dimension
  - 2 in the center
  - Less for border or corner nodes
- Efficient nearest neighbor communication
- Suitable for large number of nodes



2D mesh

N-dimensional meshes

- Typically 2 or 3 dimensions
- Direct link to neighbors
- Each node has 1 or 2 neighbors per dimension
  - 2 in the center
  - Less for border or corner nodes
- Efficient nearest neighbor communication
- Suitable for large number of nodes



2D mesh

Assuming 64 nodes

Performance:

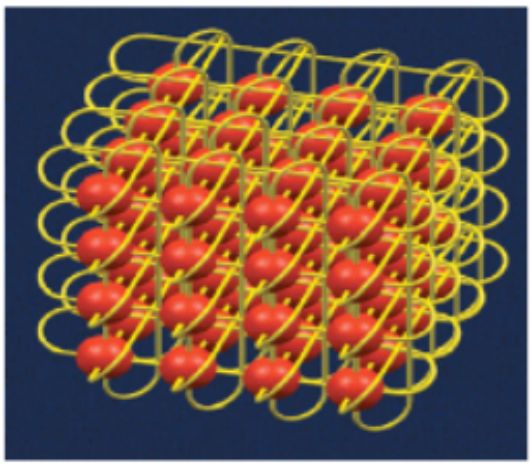
Diameter: 14
$BW_{Bisection}$ (# links): 8
Edge connectivity: 2

Cost:

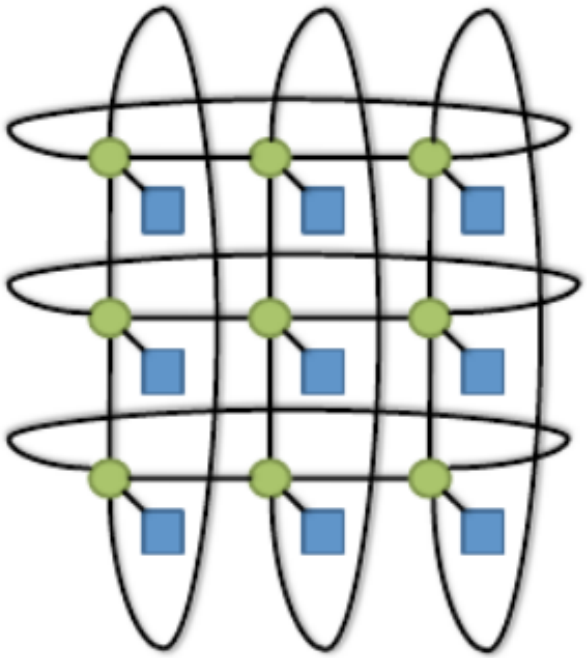
Switches: 64
Network degree: 5
links: 176

Torus

- Mesh with wrap-around connections
- Each node has exactly 2 neighbors per dimension



3D torus



2D torus

Torus

Assuming 64 nodes

Performance:

Diameter: 8

$BW_{Bisection}$ (# links): 16

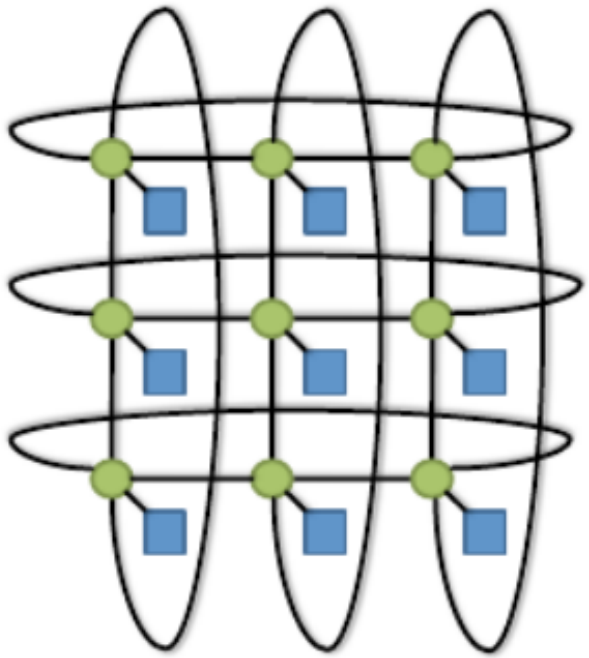
Edge connectivity: 4

Cost:

Switches: 64

Network degree: 5

links: 192



2D torus

Summary

- Flynn's classification
- This course: Focus on MIMD
  - Shared-memory architectures
    - Single address space
    - Communication: shared variables
  - Distributed-memory architectures
    - Multiple private address spaces
    - Communication: message passing
- Network topologies, performance and cost
  - latency, bandwidth
  - diameter, bisection bandwidth, connectivity
  - # switches, # links, network degree