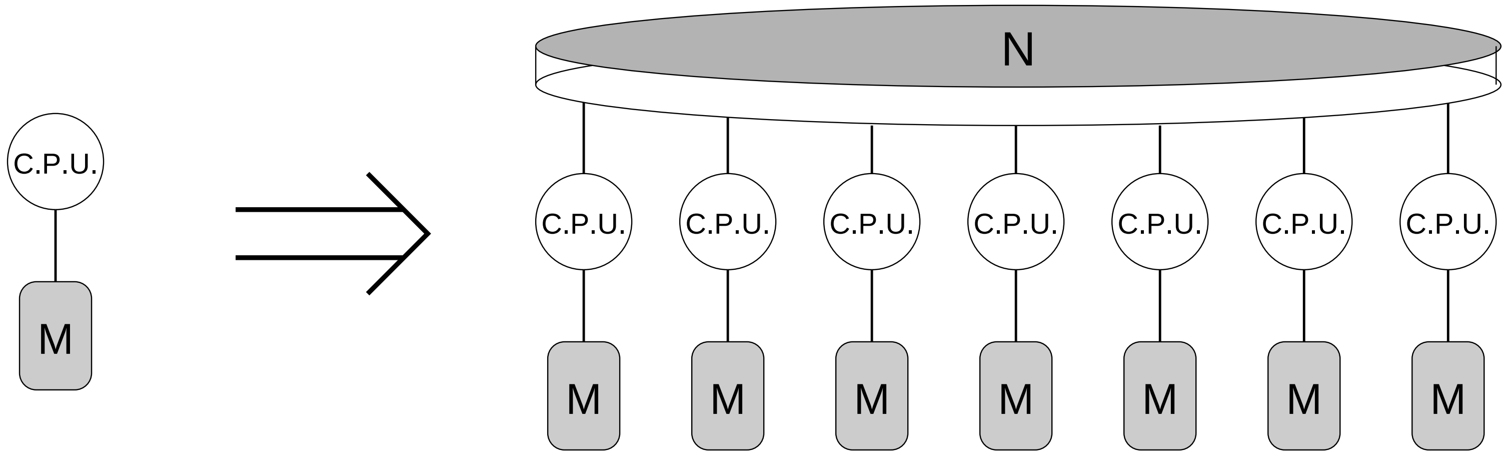
MIMD Distributed Memory Multiprocessor Systems



- Replicated Processing Elements
- Each PE is a full CPU with its own memory
- Communications Network - Processor to Processor

MIMD Distributed Memory Multiprocessor Systems

We have seen two models for parallel processing:

- Data Parallel Model
  - This model assumes that we have a *single process* which performs *calculations over whole data structures in parallel*.
  - Can use existing code following manual or automatic parallelization. Sometimes better to write new code.
- Shared Memory Model
  - Assumes that we have *multiple processes* all of which *share the same memory map*.
  - Can use existing code developed for multi-tasking on single processors.

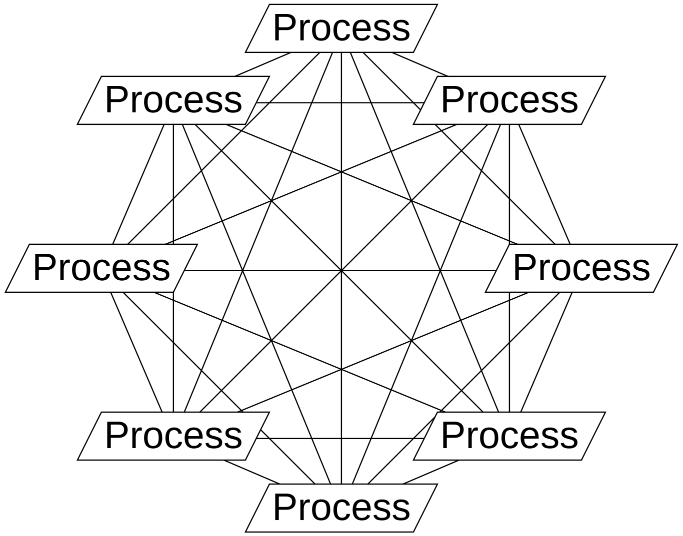
Although it is possible to simulate each of these models on a distributed memory machine, we need a new model in order to exploit such a machine more efficiently.

MIMD Distributed Memory Multiprocessor Systems

- Communicating Processes Model for Distributed Memory Machines
  - Assumes that we have *multiple independent processes* which *communicate and co-ordinate via the sending of messages*.
  - Each process can communicate with any other process, but can only access its own local memory.

In general we must write new code for this model.

Communications Network



We have suggested that our new model supports communication from any process to any other process.

It appears to follow that we will connect all processors to all other processors.

Connectivity - Some Options

- Full connection

  Each of *P* processors has *P-1* bi-directional communication links. One for each of the other processors.

- Bus Connection

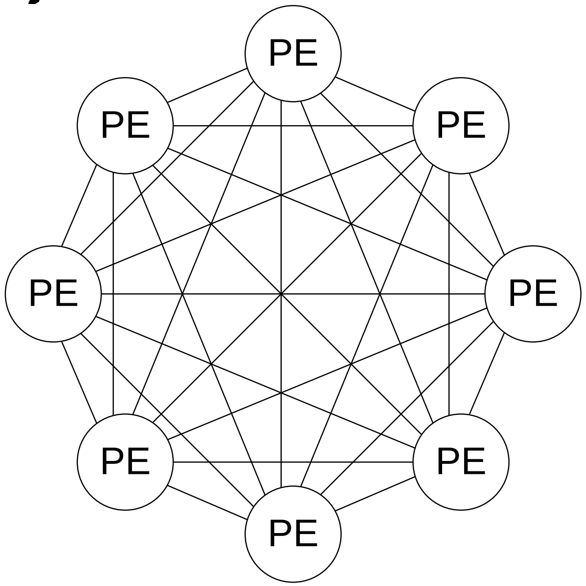
  All Processors share a single bus and compete for its use.

- Switched connection

  Each of *P* processors has one input link and one output link. The input and output links are connected into an *P by P* crossbar switch allowing any 1 to 1 connectivity.

MIMD Distributed Memory Multiprocessor Systems

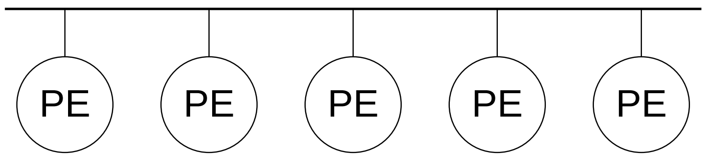
Fully Connected Network



- Expensive on link hardware ($(P-1)$ bi-directional links per PE).
- Complex wiring ($(P(P-1)/2)$ bi-directional links must be routed).
- Difficult to expand (each PE must be changed in order to add just one extra PE).

MIMD Distributed Memory Multiprocessor Systems

Bus Connection[1]

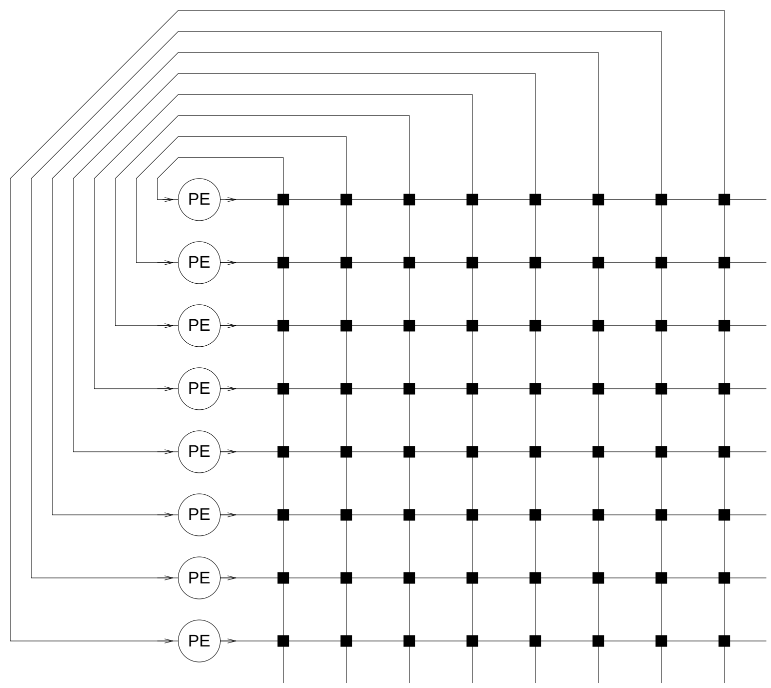


- Low Bandwidth.
- Bandwidth doesn't increase as more processors are added.
- Arbitration Problems.

[1] e.g. Ethernet

Communications Networks for MIMD Systems

Crossbar Switch

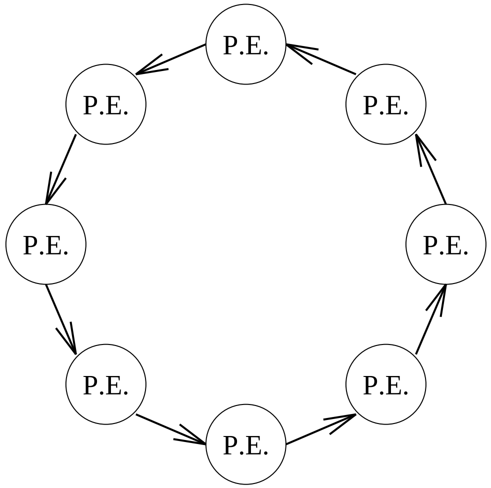


- Large numbers of switches (P^2).
- Difficult to expand.

Connectivity - Another Option

- Partial Connection
  - Each of P processors has a number of connections to a number of different processors supporting a fraction of the required interconnection.
  - The remainder of the connectivity is supported via message forwarding. Messages to distant nodes are forwarded from one node to the next until they reach their destination.

Uni-Directional Ring Network[2]



- Simplest Partial Connection Network.
- Only 1 input link and 1 output link per PE.
- To reach the preceding PE a message must travel via all other PEs.

[2]e.g. Cambridge Ring

Which Connectivity Subset

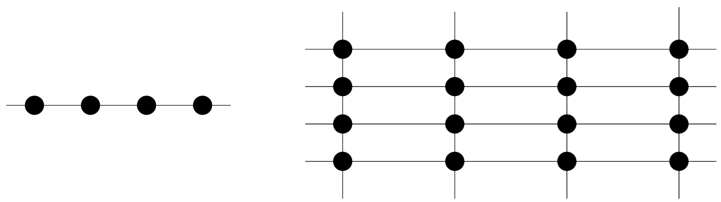
If we replace our uni-directional ring with a bi-directional ring we double the number of links and half the maximum distance between nodes.

What factors affect our choice of an optimum connectivity subset.

- Maximum Distance
- Connections per Processor
- Bandwidth
- Distribution of Messages
- Wiring Complexity
- Mapping Of Problems
- Route Choice Availability

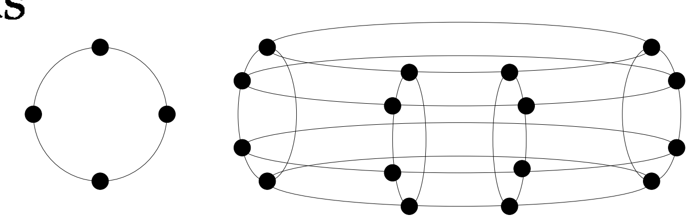
Open Networks & Closed Networks

- Open Networks



All links are bi-directional ($\rightleftharpoons$). Edge processors require fewer links than middle processors. Dangling links at ends of rows can be used for I/O.

- Closed Networks



Links may be uni-directional ($\rightarrow$) or bi-directional ($\rightleftharpoons$).All processors have equal numbers of links. I/O can be achieved by breaking a link or with extra hardware.

Trade-off Distance v Cost

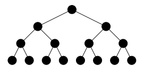
Topology	Maximum Distance d	Links per Processor l (in or out)	Product $d * l$
Closed Networks			
Ring ($\rightarrow$)	$P-1$	1	$P-1$
Ring ($\rightleftharpoons$)	$P/2$	2	P
2-D Torus ($\rightarrow$)	$2(\sqrt{P}-1)$	2	$4(\sqrt{P}-1)$
2-D Torus ($\rightleftharpoons$)	$\sqrt{P}$	4	$4(\sqrt{P})$
N-D Torus ($\rightarrow$)	$N(\sqrt[N]{P}-1)$	N	$N^2(\sqrt[N]{P}-1)$
N-D Torus ($\rightleftharpoons$)	$N(\sqrt[N]{P})/2$	$2N$	$N^2(\sqrt[N]{P})$
N-D Binary Hypercube	$N = log_2(P)$	$N = log_2(P)$	$N^2 = (log_2(P))^2$
Fully Connected	1	$(P-1)$	$P-1$

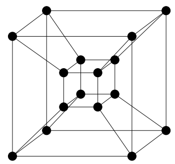
Trade-off Distance v Cost

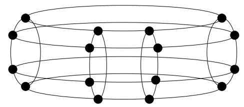
Topology	Maximum Distance d	Links per Processor l (in or out)	Product $d * l$
Open Networks			
Line	$P-1$	2	$2(P-1)$
2-D Grid	$2(\sqrt{P}-1)$	4	$8(\sqrt{P}-1)$
3-D Grid	$3(\sqrt[3]{P}-1)$	6	$18(\sqrt[3]{P}-1)$
N-D Grid	$N(\sqrt[N]{P}-1)$	$2N$	$2N^2(\sqrt[N]{P}-1)$
Binary Tree	$2log_2(P+1)$	3	$6log_2(P+1)$

Communications Networks for MIMD Systems

Favourite Networks







- Trees

  Although the Binary Tree (and other higher order trees) appear to do well on our $d * l$ measure, they suffer greatly from an imbalance of traffic - $1/2$ of all traffic passes through the root node. This is a popular network for *farms*.

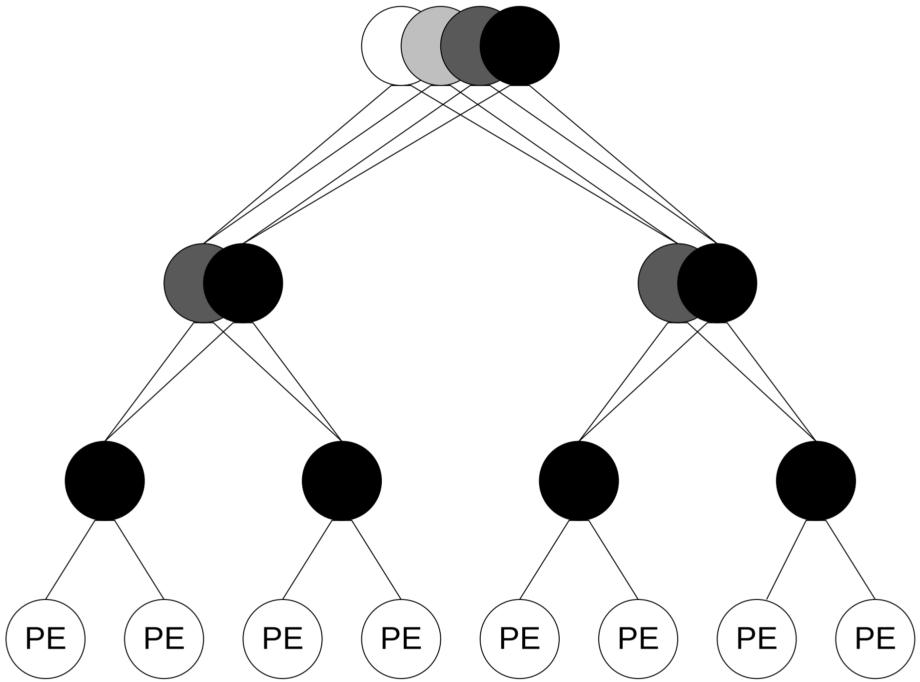
- Hypercubes

  These are very popular. Excellent $d * l$. Map well (if not easily) to most problems including FFTs.

- 2D Grid and Torus

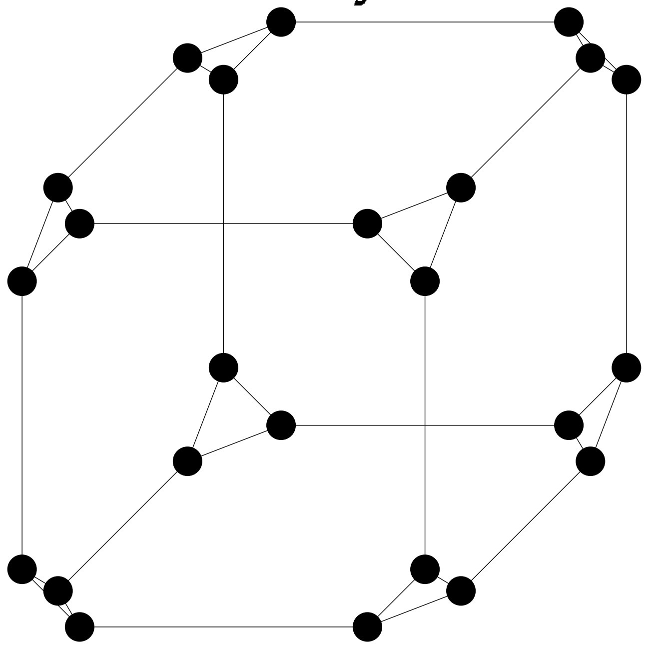
  The most popular constant *Valency* topologies. PEs need not be changed to cope with larger networks. Map well to many problems and are very easy to wire.

Hybrids - Fat Trees



- Separate processing nodes from message forwarding nodes.
- Bandwidth at a node is dependent upon traffic through it.

Hybrids - Cube connected cycles



- Pseudo-Hypercube with constant *Valency* nodes. When adding another dimension you increase by one the number of nodes in a cycle.
- An N-D Network will have $N * 2^N$ nodes.