Interconnects

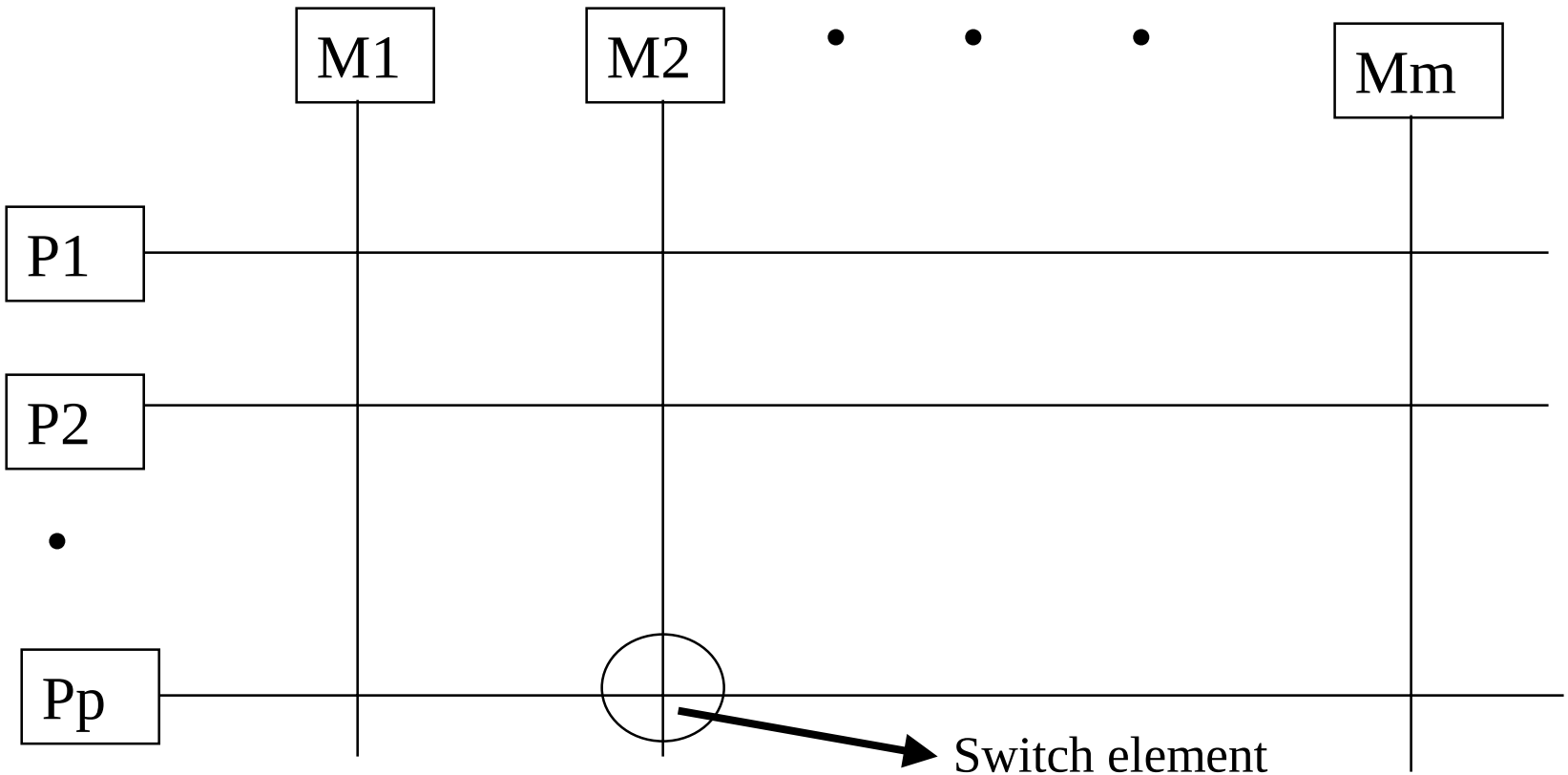
- **Shared address space and message passing computers can be constructed by connecting processors and memory unit using a variety of interconnection network**
    - **Dynamic/Indirect networks**:
        - cross bar
        - bus based
    - **Static/Direct networks**:
        - completely connected
        - star connected
        - linear array
        - ring
        - mesh
        - hypercube

More on Interconnects

- **Dynamic Interconnect:** Communication links are connected to one another dynamically by the switching elements to establish path among processors and memory banks. Normally used for share memory address space computers.
- **Static/Direct Interconnect** : Consists of point to point communication links among processors. Typically used for message passing computers.

Dynamic Interconnect

- **Cross bar switching : p processors, m memory banks**



Dynamic Interconnect

- **Cross Bar switch is a non blocking network i.e. connection of a processor to a memory bank does not block the connection of any other processor to any other memory bank.**

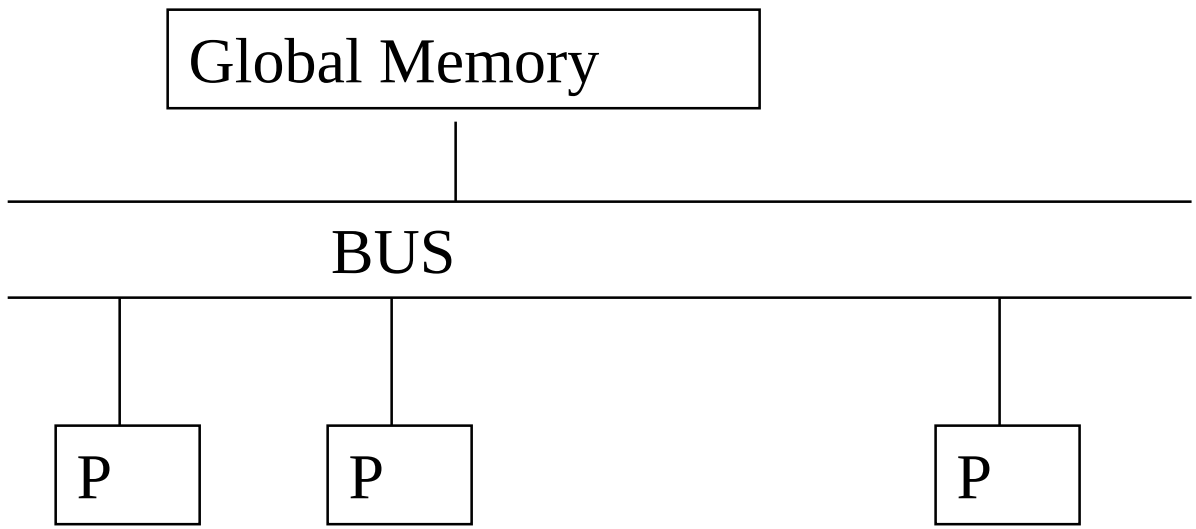
  Total # of switching elements required is
  f(p*m) = approx f(p*p) (assuming p = m)

  As p ↑ complexity of switching network f(p*p) ↑

  Cross bar switches are not scalable in terms of cost.

Dynamic Interconnect

- **Bus based network: Processors are connected to global memory by means of a common data path called a bus.**

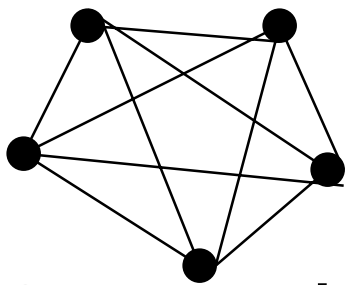


Dynamic Interconnect

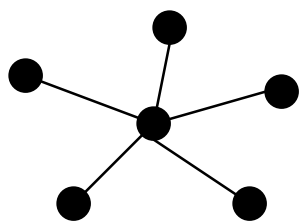
- **Bus based network**
- Simplicity of construction
- Provides uniform access to shared memory
- Bus can carry limited amount of data between the memory and processors
- As the number of processors increases each processor spends more time waiting for memory access while the bus is used by other processor

Static Interconnect

- **Completely Connected :** Each processor has direct communication link to every other processor

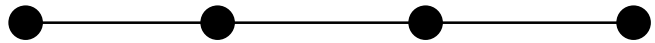


- **Star Connected Network** : The middle processor is the central processor. Every other processor is connected to it. Counter part of Cross Bar switch in Dynamic interconnect.

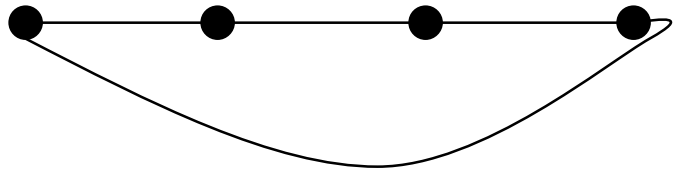


Static Interconnect

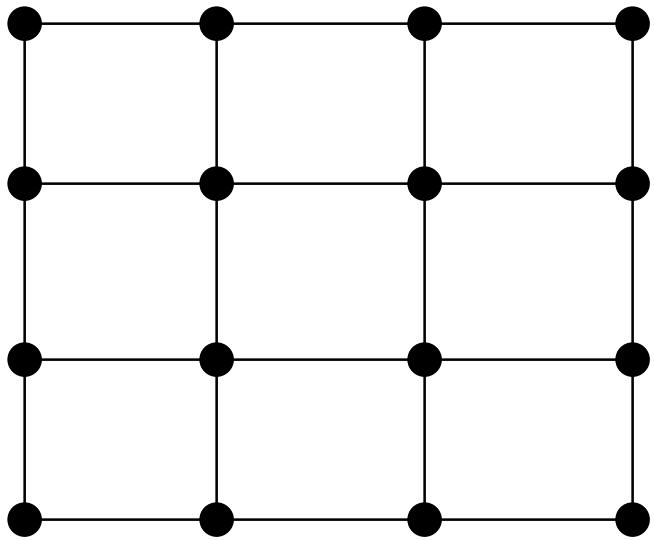
- **Linear Array :**



- **Ring :**

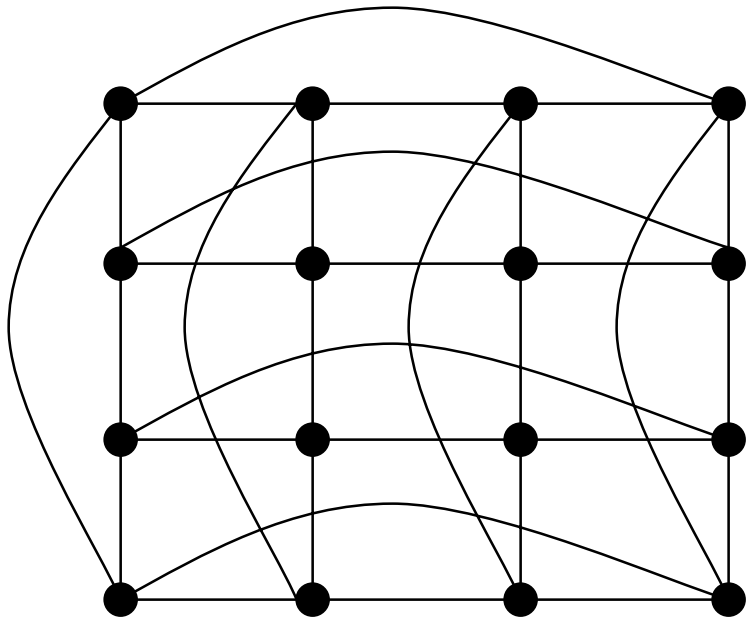


- **Mesh Network :**



Static Interconnect

- **Torus or Wraparound Mesh :**

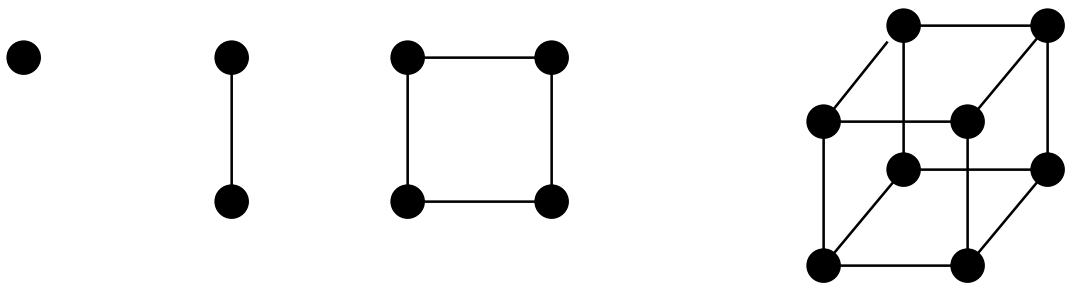


Static Interconnect

- **Hypercube Network :** A multidimensional mesh of processors with exactly two processors in each dimension. A d dimensional processor consists of

$$p = 2^d \text{ processors}$$

shown below are 0, 1, 2, and 3 dimensional hypercubes



0-D 1-D 2-D 3-D hypercubes

More on Static Interconnects

- **Diameter :** Maximum distance between any two processors in the network. (the distance between two processors is defined as the shortest path, in terms of links, between them). This relates to communication time. Diameter for completely connected network is 1, for star network is 2, for ring is p/2 (for p even processors)
- **Connectivity:** This is a measure of the multiplicity of paths between any two processors (# arcs that must be removed to break into two). High connectivity is desired since it lowers contention for communication resources. Connectivity is 1 for linear array, 1 for star, 2 for ring, 2 for mesh, 4 for torus.

More on Static Interconnects

- **Bisection width:** Minimum # of communication links that have to be removed to partition the network into two equal halves. Bisection width is 2 for ring, sq. root(p) for mesh with p (even) processors, p/2 for hypercube, (p*p)/4 for completely connected (p even).
- **Channel width:** # of bits that can be communicated simultaneously over a link connecting 2 processors
- **Channel rate:** Peak rate at which a single physical wire link can deliver bits
- **Channel BW : (**channel width) * (channel rate)
- **Bisection BW :** (bisection width) * (channel BW)

Communication Time Modeling

- **Tcomm = Nmsg * Tmsg**

  Nmsg = # of non overlapping messages
  Tmsg = time for one point to point communication

  L = length of message (for e.g in words)
  Tmsg = ts + tw * L

  latency = ts = startup time (size independent)
  tw = asymptotic time per word (1/BW)

Performance and Scalability Terms

- **Serial runtime(Ts):** Time elapsed between the begining and the end of execution of a sequential program
- **Parallel runtime(Tn):** Time that elapses from the moment that a parallel computer starts to execute to the moment that the last processor finishes execution.
- **Speedup(S):** Ratio of the serial runtime of the best sequential algorithm for solving a problem to the time taken by the parallel algorithm to solve the same problem on N processors.
  Ts = serial time
  Tn = parallel time on N processors
  S = Ts/Tn

Performance and Scalability Terms

- **Efficiency:** Measure of the fraction of time for which a processor is usefully employed. Defined as the ratio of speedup to the number of processor. E = S/N
- **Amdahl's law :** discussed before
- **Scalability :** An algorithm is scalable if the level of parallelism increases at least linearly with the problem size. An architecture is scalable if it continues to yield the same performance per processor, albeit used on a larger problem size, as the # of processors increases. Algorithm and architecture scalability are important since they allow a user to solve larger problems in the same amount of time by buying a parallel computer with more processors.

Performance and Scalability Terms

- **Superlinear speedup:** In practice a speedup greater than N (on N processors) is called superlinear speedup. This is observed due to
  1. Non optimal sequential algorithm
  2. Sequential problem may not fit in one processor's main memory and require slow secondary storage, whereas on multiple processors problem fits in main memory of N processors

Sources of Parallel Overhead

- **Interprocessor communication:** Time to transfer data between processors is usually the most significant source of parallel processing overhead.
- **Load imbalance: I**n some parallel applications it is impossible to equally distribute the subtask workload to each processor. So at some point all but one processor might be done and waiting for one processor to complete.
- **Extra computation:** Sometime the best sequential algorithm is not easily parallizable and one is forced to use a parallel algorithm based on a poorer but easily parallizable sequential algorithm. Sometimes repeatative work is done on each of the N processors instead of send/recv, which leads to extra computation.