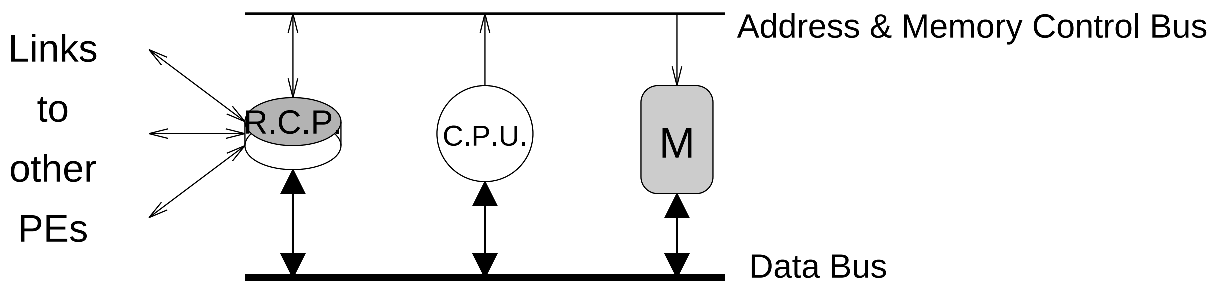
Hardware Routing

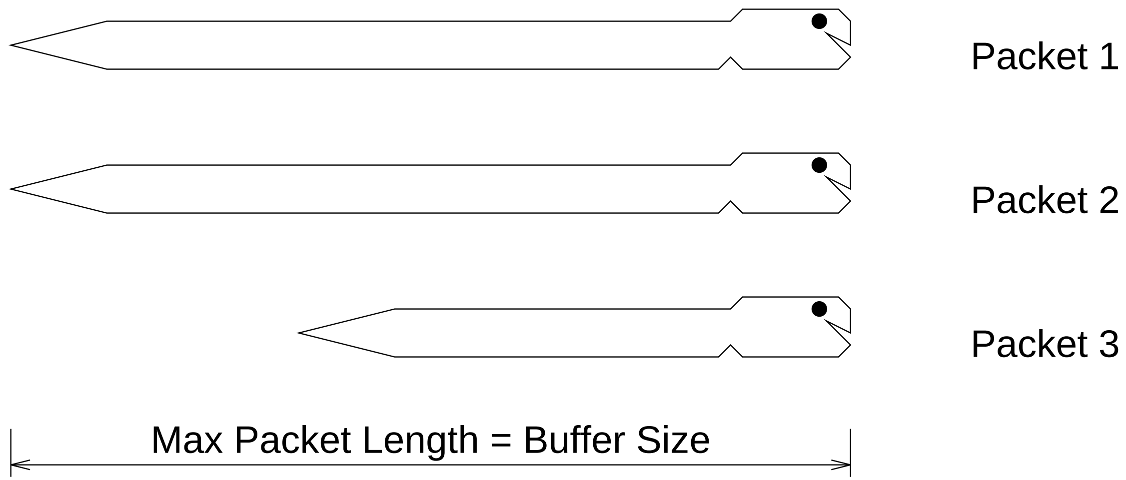
In a large network employing software routing, the task of message forwarding may significantly reduce the performance of the PEs.
As the network grows in size, hardware routing becomes more and more attractive:

- With hardware packet routing we provide each PE with a routing co-processor.
- Routing decisions can be made within one cycle thereby speeding message delivery.
- The PE's CPU need only deal with messages to or from the local node.



Packet Routing

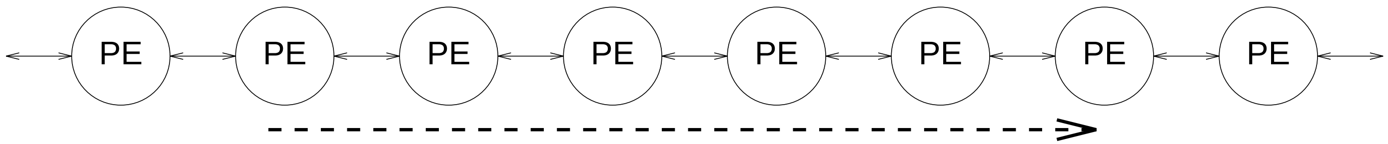
Most hardware routers deal with packets rather than complete messages.

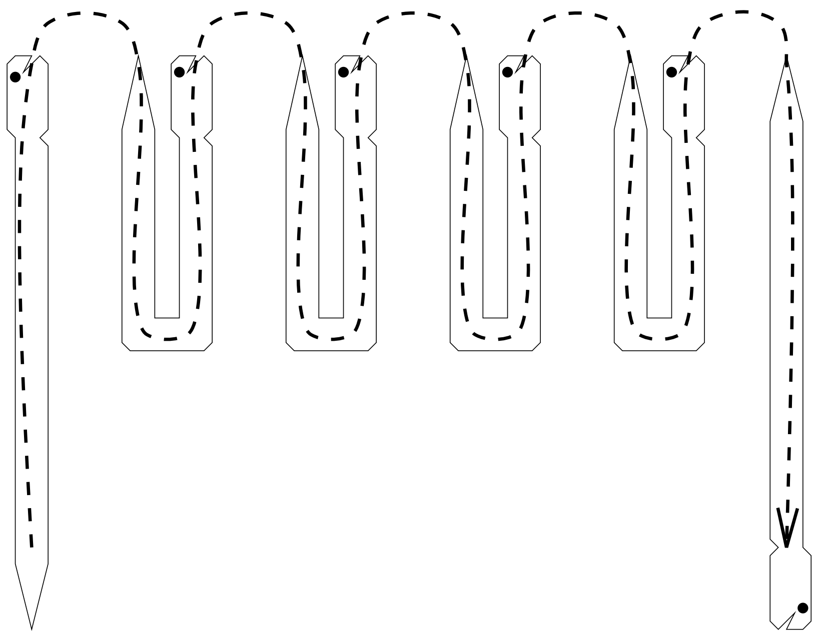


- A message is divided into packets before transmission.
- Each packet carries its own copy of the destination address.
- Packets have a fixed maximum length determined by the buffer size of the router.
- On arrival the packets are re-assembled into the original message.

Store and Forward Packet Routing

Let us examine the transport of a packet in a *store and forward* network.





Routing Performance – *Store and Forward*

- The time taken for the delivery of a message is calculated as follows;

$$T = n * \left(\frac{l + h}{B}\right)$$

  where

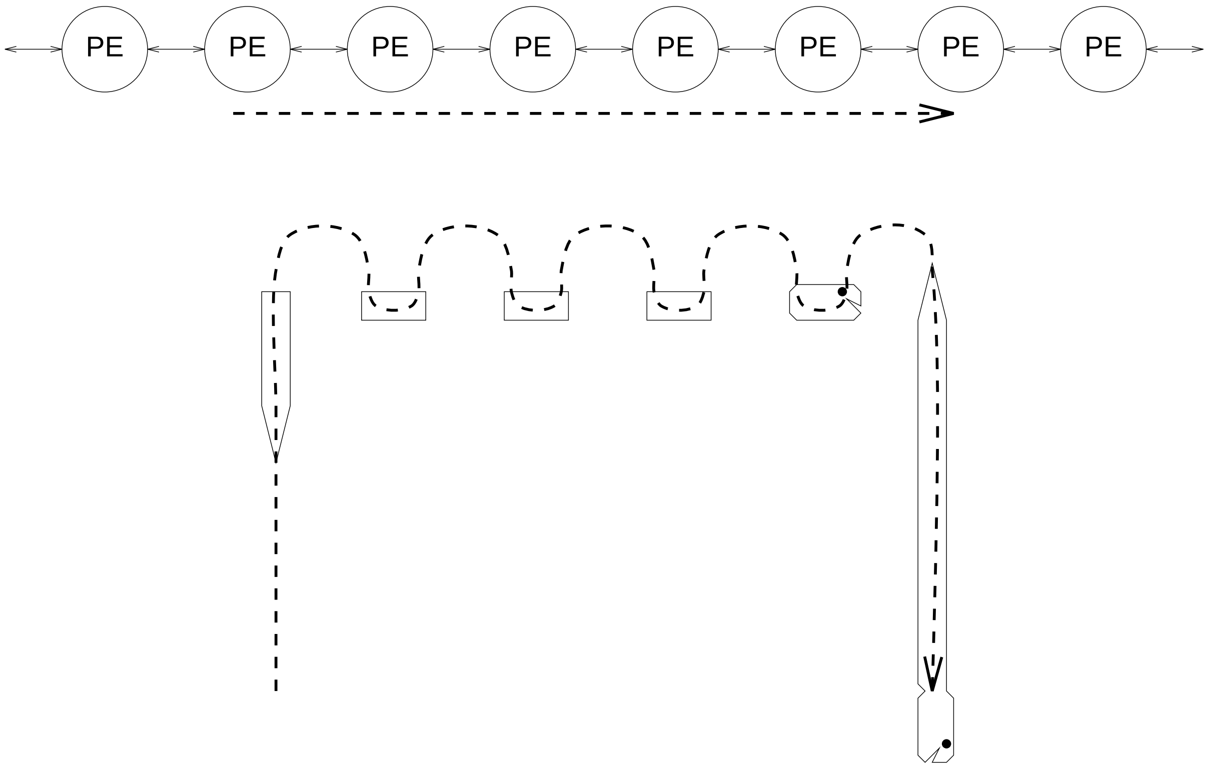
  - n is the number of hops.
  - l is the number of data bits.
  - h is the number of header bits.
  - B is the link bandwidth in bits per second.

  Assumptions

  - There is no overhead for decision making (hardware routing).
  - We have an empty network (packets will be further delayed in a busy network).

Wormhole Routing

Wormhole routing offers improved performance since it makes the routing decision early. The packet may be passed on as soon as the head has been received[1]:



[1]note that wormhole does not set a limit on packet size since the buffer at each node is only as big as the header

Routing Performance – *Wormhole Routing*

Advantages:

- In an empty network

  The time taken for the delivery of a message is;

$$T = \frac{l + (n * h)}{B}$$

  Wormhole exhibits significant advantages over *store & forward*.

Disadvantages:

- In an busy network:

  When a single message is delayed due to a temporary blockage it remains spread out across the network, potentially blocking a large number of other messages.

  It appears that *store & forward* routing may have advantages in busy networks, as a single waiting packet can only block one node.

Virtual Cut-Though Packet Routing

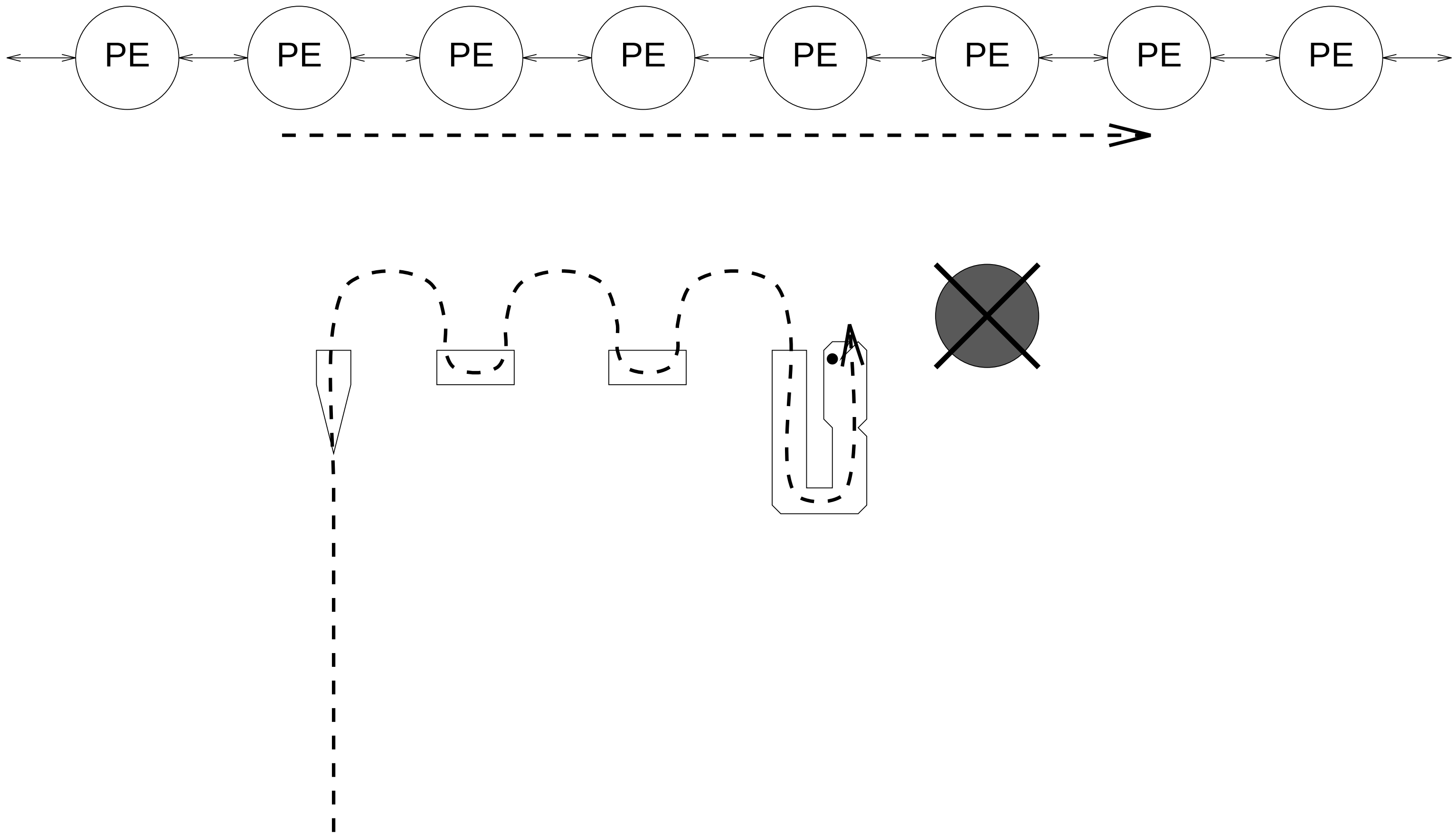
Virtual cut-through packet routing is a hybrid of *wormhole* and *store & forward*.

- In an empty network it behaves like wormhole routing where each packet spreads itself across the network.
- When a packet encounters a blockage, the head stops at the blockage while the tail continues until all of the packet is buffered at a single node.

  When the blockage is gone the packet will again spread out across the network.
- In this way we get the best of both worlds.

Virtual Cut-Though Packet Routing

Buffering of a blocked packet in *virtual cut-through* routing:

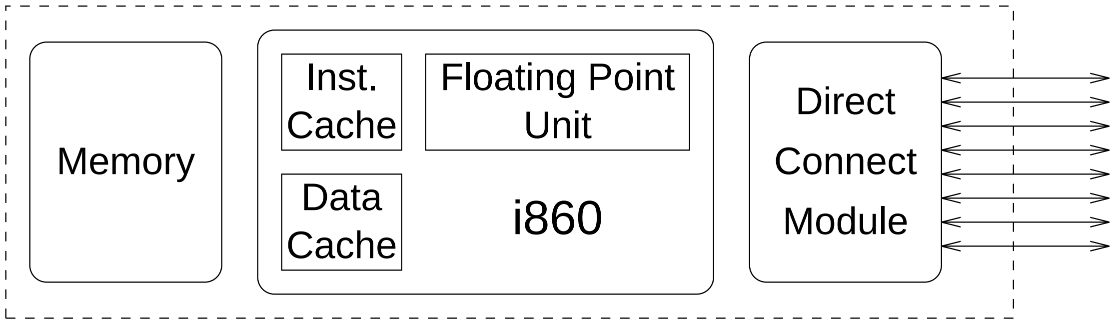


iPSC/860 Hypercube Computer

Intel Personal SuperComputer

- Built by Intel Scientific Computers (iSC).
- Node Architecture:
  - i860 CPU:
    - - - 40 MHz RISC with on chip 64 bit FPU
    - - - Multiple Instructions per cycle with pipelining & instruction caching
    - - - 60 Mflops peak - Only with very careful programming.
  - Direct-Connect Routing Module:
    - - - Hardware Routing co-processor
    - - - 8 bi-directional links @ 2.8 Mbytes/s
    - - - *Wormhole* routing
  - 8-16 Mbytes of memory.

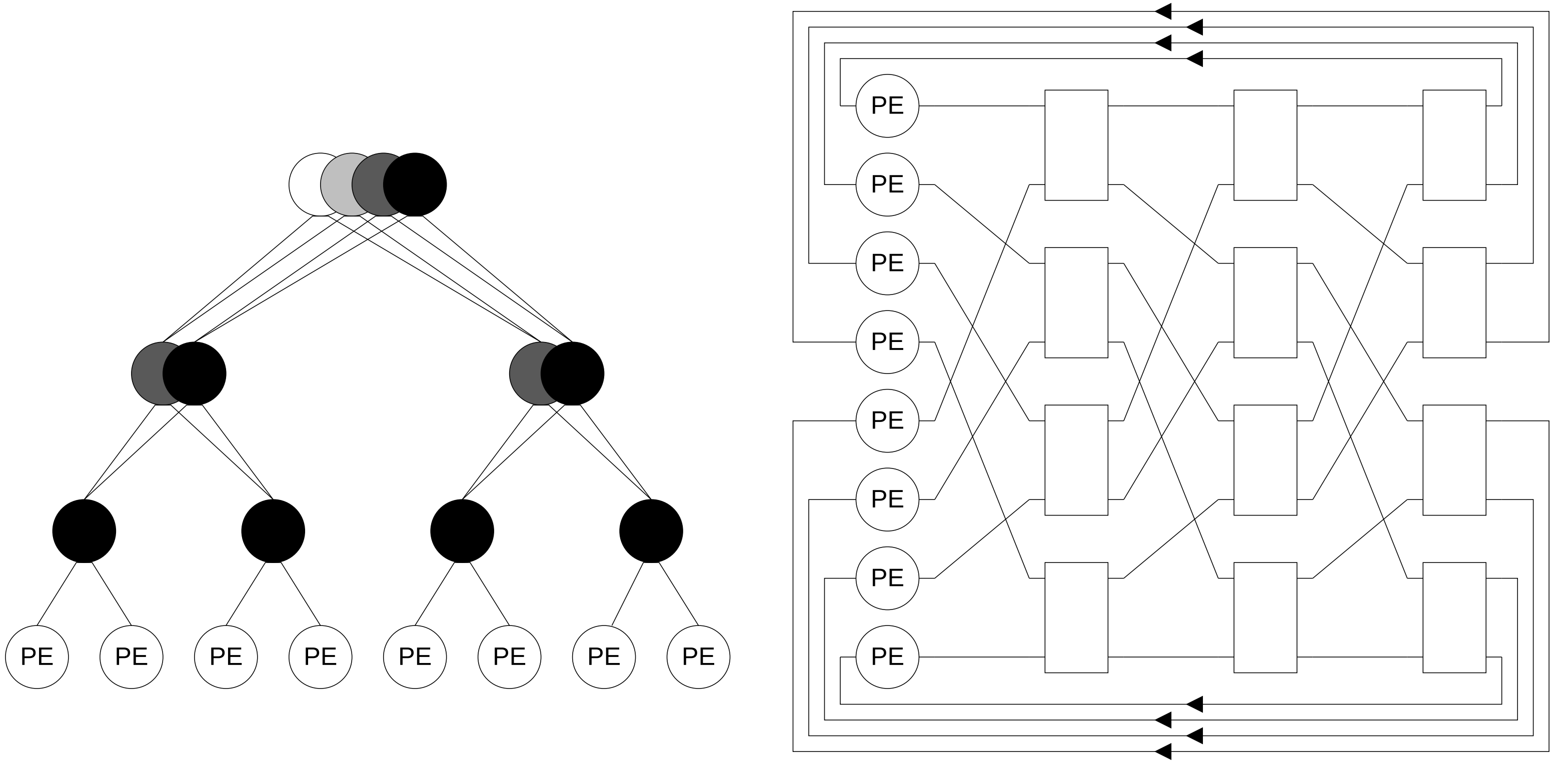
iPSC/860 Hypercube Computer



- System Architecture:
  - 8 to 128 nodes
  - Hypercube structure $3D$ to $7D$
  - Performance:
    - - Theroetical peak performance: 7.6 Gflops
    - - Benchmark performance 2.6 Gflops

Indirect Networks

With separate routing processors we can make use of *indirect networks*[2]:

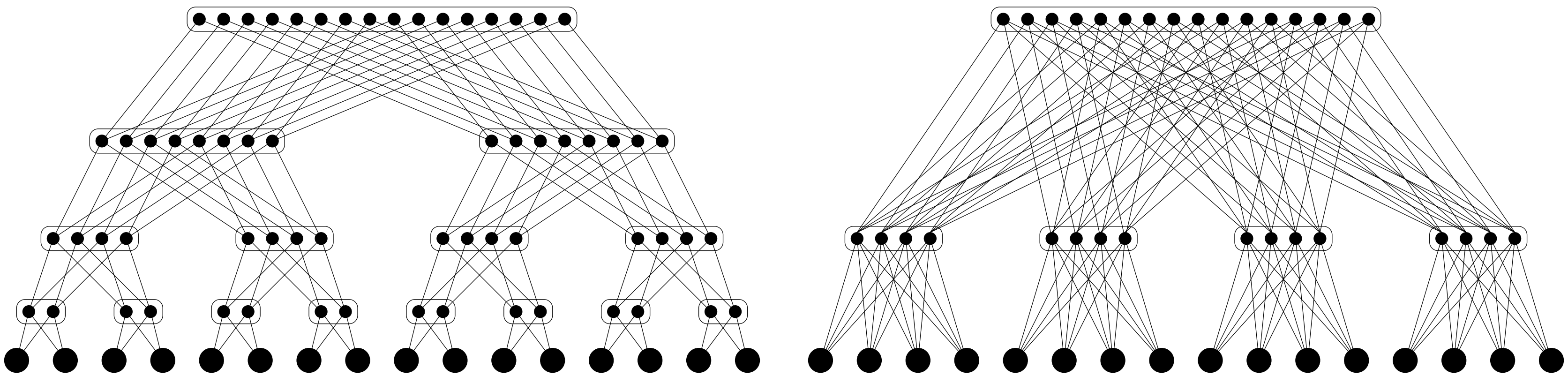


Examples of indirect networks; *Fat tree* and *Omega network*.

[2]a.k.a. switching networks or dynamic networks

Indirect Networks

Fat Trees



- A quaternary fat tree has the same diameter ($log_2 P$) as a hypercube with the same number of PEs.
- Scalability

  In an Nary fat-tree where each node has N parents, the bisection bandwidth increases in proportion to the size of the network.

  This is possible since the proportion of routing nodes is greater in larger networks[3].

[3] c.f. in a hypercube we must increase the number of links per node to achieve the same effect.