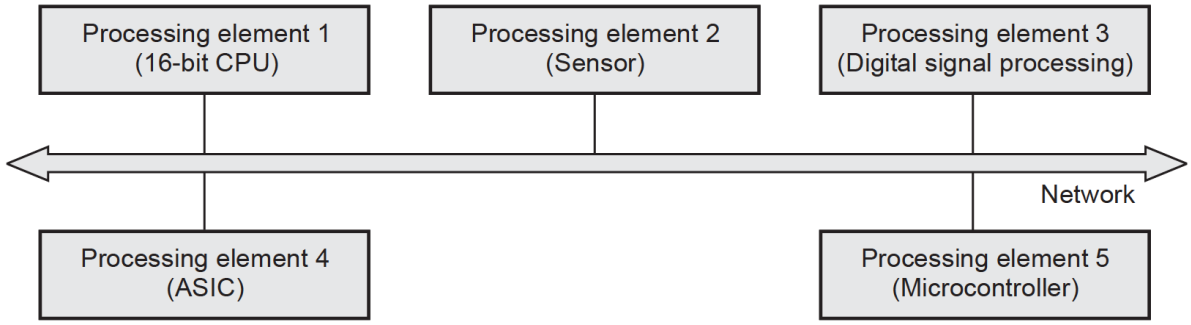
Distributed Embedded Systems

In a distributed embedded system, several Processing Elements (PEs) are connected by a network that allows them to communicate.



Distributed embedded system

Processing elements may includes DSP, CPU or microcontroller. Nonprogrammable unit such as the ASICs is also used to implement as PE.

By using this entire processing element, it forms bus topology. It is also possible to form other topology also. It is also possible that the system can use more than one network, such as when relatively independent functions require relatively little communication among them.

All the processing elements are connected by communication link. The system of PEs and networks forms the hardware platform on which the application runs.

Why Distributed

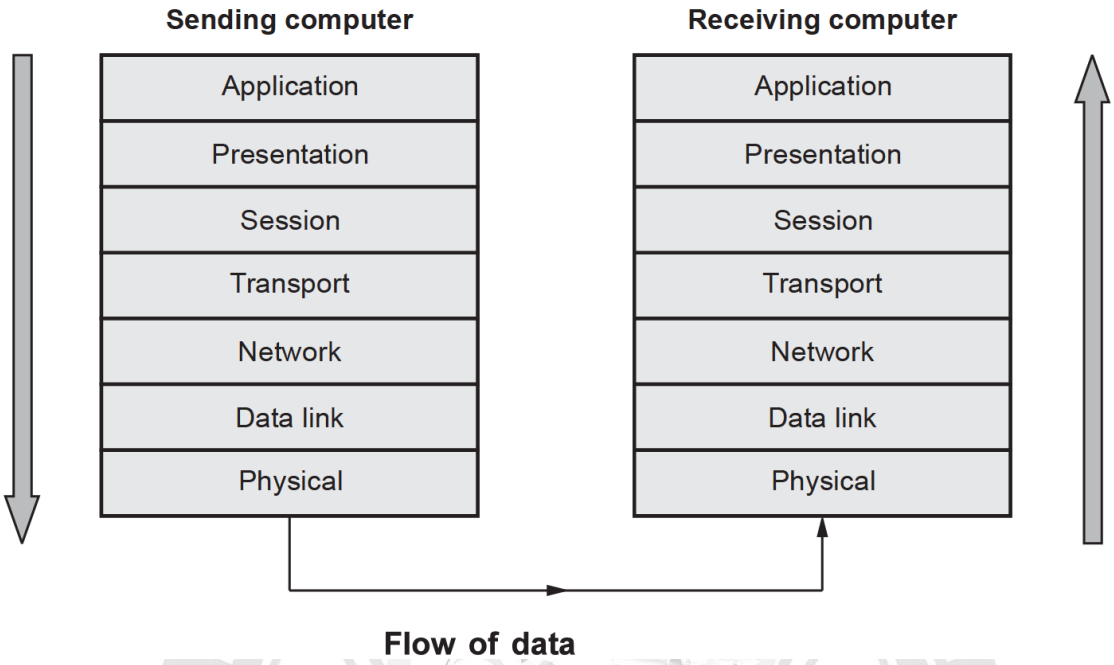
- Higher performance at lower cost.
- Physically distributed activities, i.e. time constants may not allow transmission to central site.
- Improved debugging : use one CPU in network to debug others.
- May buy subsystems that have embedded processors.
- Distributed systems are necessary because the devices that the PEs communicate with are physically separated.

Network Abstractions

- Networks are complex systems. They provide high-level services while hiding many of the details of data transmission from the other components in the system.
- To understand network design, the International Standards Organization has developed a seven-layer model for networks known as Open Systems Interconnection (OSI) models.
- It is based on a common model of network architecture and a suite of protocols used in its implementation. The International Organization for Standardization (ISO) established the Open Systems Interconnection (OSI) Reference Model.
- Each layer deals with a particular aspect of network communication. There are seven layers in the model, hence the name the 7-Layer model. The model acts as a frame of reference in the design of communications and networking products.

7. Application layer
6. Presentation layer
5. Session layer
4. Transport layer
3. Network layer
2. Data link layer
1. Physical layer
OSI layers

The OSI model describes how information or data makes its way from application programmers through a network medium to another application programmer located on another network. The OSI reference model is a hierarchical structure. Changes in one layer should not require changes in other layers.



1. **Physical layer :** The lowest layer of the OSI model. It is concerned with the transmission and reception of the unstructured raw bit stream over a physical medium. It describes the electrical/optical, mechanical and functional interfaces to the physical medium, and carries the signals for the entire higher layer.
2. **Data link layer :** DLL is responsible for the transfer of data over the channel. It groups zeros and ones into frames. A frame is a series of bits that forms a unit of data. The data link layer provides error-free transfer of data frames from one node to another over the physical layer. It contains two sublayers : Medium Access Control (MAC) and Logical Link Control Layer (LLC). DLL divides the bit stream of the physical layer into frames, messages containing data and control information. It handles lost, damaged and duplicate frames.
3. **Network layer :** This is responsible for addressing messages and data so they are sent to the correct destination, and for translating logical addresses and names into physical addresses. This layer is also responsible for finding a path through the network to the destination computer. Lowest layer that deals with host-to-host communication, call this end-to-end

communication. Functions of network layer : Logical addressing, Routing and Frame fragmentation

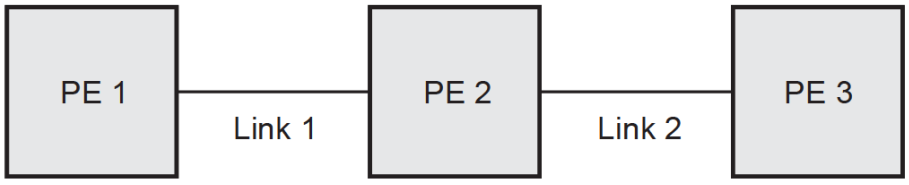
4. **The transport layer** ensures that messages are delivered error-free, in sequence, and with no losses or duplications. It relieves the higher layer protocols from any concern with the transfer of data between them and their peers. It also provides flow control, sequence numbering and message acknowledgement.Function of transport layer : Message segmentation, Message acknowledgment, Session multiplexing
5. The **session layer** adds mechanisms to establish, maintain, synchronize and manage communication between network entities. The session layer allows session establishment between processes running on different stations. Services provided by session layer : Synchronization and dialog management.
6. The **presentation layer** is responsible for data compression, data expansion, data encryption and data decryption.
7. **Application layer** : It contains all services or protocols needed by application software or operating system to communicate on the network. Typical applications include a client/server application, an e-mail and an application to transfer files using FTP or HTTP.

Hardware and Software Architectures

Distributed embedded systems can be organized in many different ways depending upon the needs of the application and cost constraints.

Point-to-point :

Point-to-point link establishes a connection between exactly two PEs. Point-to-point links are simple to design precisely because they deal with only two components.



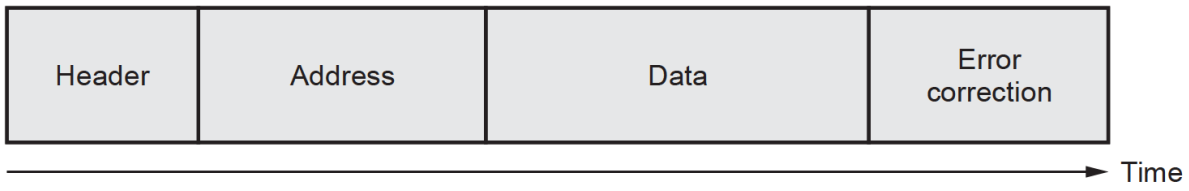
Signal processing system built from print-to-point links

Input device sampled the signal and passed to the first digital filter (Fl) by using point-to-point link. The results of that filter are sent through a second point-to-point link to filter (F2). The results in turn are sent to the output device over a third point-to-point link.

Filters must process their inputs in a timely fashion. It is possible to build full duplex point-to-point distributed system.

A bus is a more general form of network since it allows multiple devices to be connected to it. Like a microprocessor bus, PEs connected to the bus have addresses.

Communication between processing elements takes place by using packets.



Format of packet

Packet contains destination address, user data and error correction codes. Sending data size is not exactly fit into packet but processing elements must take care of packet

The data to be transmitted from one PE to another may not fit exactly into the size of the data payload on the packet. It is the responsibility of the transmitting PE to divide its data into packets; the receiving PE must of course reassemble the complete data message from the packets.

Arbitration scheme

The device that is allowed to initiate transfers on the bus at any given time is called the bus master

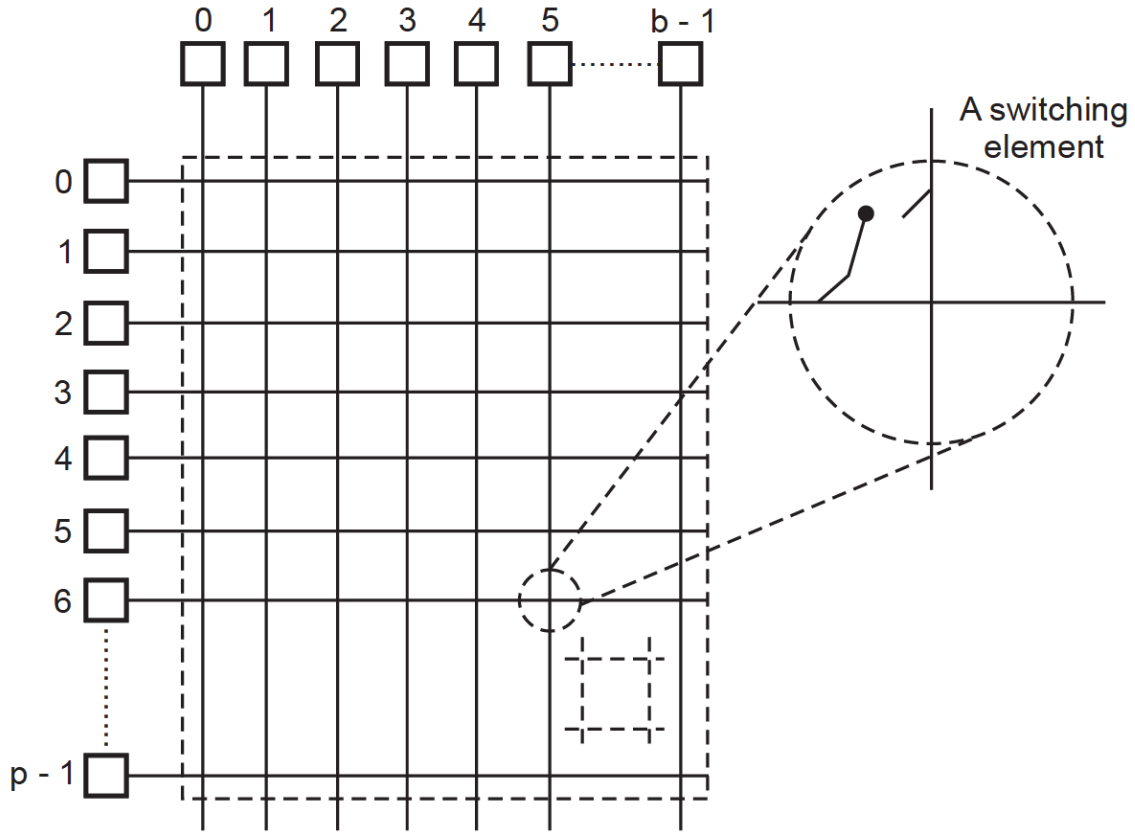
1. **Fixed-priority arbitration :** High priority devices always get chance to transmit data. If low priority device and high priority device are ready to transmit data, then high priority device will transmit data then low priority device will transmit data.
2. **Fair arbitration schemes :** This scheme take care of starvation. Round-robin arbitration is the most commonly used of the fair arbitration schemes. The PCI bus requires that the arbitration scheme used on the bus must be fair, although it does not specify a particular arbitration scheme. Most implementations of PCI use round-robin arbitration.

Crossbar network

A bus topology provides limited available bandwidth. Since all devices connect to the bus, communications can interfere with each other. For reducing communication conflicts, other network topology can be used.

A crossbar network uses an p x m grid of switches to connect p inputs to m outputs in a non-blocking manner. A connection of one processor to a given memory bank does not block a connection of another processor to a different memory bank. There must be p x b switches. It is reasonable to assume that b>p.

A link carries one message; a switch can process up to two messages at the same time. Crossbar is not fault tolerant, failure of any switchbox will disconnect certain pairs.



Direct network crossbar

Message Passing Programming

A message-passing system is a subsystem of distributed operating system that provides a set of message-based IPC protocols, and does so by shielding the details of complex network protocols and multiple heterogeneous platforms from programmers. It enables processes to communicate by exchanging messages and allows programs to be written by using simple communication primitives, such as send and receive.

Transport layer provides message-based programming interface : send_msg (adrs, datal);

Data must be broken into packets at source, reassembled at destination.

Data-push programming : receivers respond to new data.