ARCHITECTURE OF DISTRIBUTED SYSTEMS

Communication:
Components of a distributed system have to communicate in order to interact. This implies support at two levels:

1. Networking infrastructure (interconnections &
network software).
2. Appropriate communication primitives and models
and their implementation:
• communication primitives:
- send
- receive
(Message Passing)
- remote procedure call (RPC)
• communication models
- **client-server communication**: implies a message exchange between two processes: the process which requests a service and the one which provides it;
- group muticast: the target of a message is a set of processes, which are members of a given group.

Performance and Scalability
Several factors are influencing the performance of a distributed system:
The performance of individual workstations.
The speed of the communication infrastructure
Extent to which reliability (fault tolerance) is provided (replication and preservation of coherence imply large overheads).
Flexibility in workload allocation: for example, idle processors (workstations) could be allocated automatically to a user's task.
Scalability
The system should remain efficient even with a significant increase in the number of users and resources connected:
- cost of adding resources should be reasonable;
- performance loss with increased number of users and resources should be controlled;
- software resources should not run out (number of bits allocated to addresses, number of entries in tables, etc.)

Architecture

Architectures for Parallel Programming Parallel computers can be divided into two categories: those that contain physical shared memory, called multiprocessors and those that do not, called

multicomputers .A simple taxonomy is given in Fig.

Hardware and Software Concept:
Even though all distributed system consists of multiple cpus, there are several different ways the hardware can be organized in terms of how they are interconnected and how they communicate. Various classification schemes for multiple CPU computer system have been proposed. Most frequently cited taxonomy is Flynn's although it is fairly rudimentary.

Flynn proposed the following categories of computer systems:
Single instruction single data (SISD) stream: A single processor executes a single instruction stream to operate on data stored in a single memory. All traditional uniprocessor computers (i.e those having only one CPU) fall in this category, from personal computers to mainframes.
Single instruction multiple data (SIMD) stream: This type refers to array processors with one instruction unit that fetches an instruction, and then commands many data units to carry it out in parallel, each with its own data. These machines are useful for computation that repeat the same calculation on many sets of data.Vector and array processors fall into this category.
Multiple instruction single data (MISD) stream: A sequence of data is transmitted to a set of processors, each of which executes a different instruction sequence. This structure has never been implemented. No known computers fit in this model.
Multiple instruction multiple data (MIMD) stream: A set of processors simultaneously execute different instruction sequences on different data sets. Which essentially means a group of independent computers, each with its own program counter, program and data. All distributed system are MIMD.

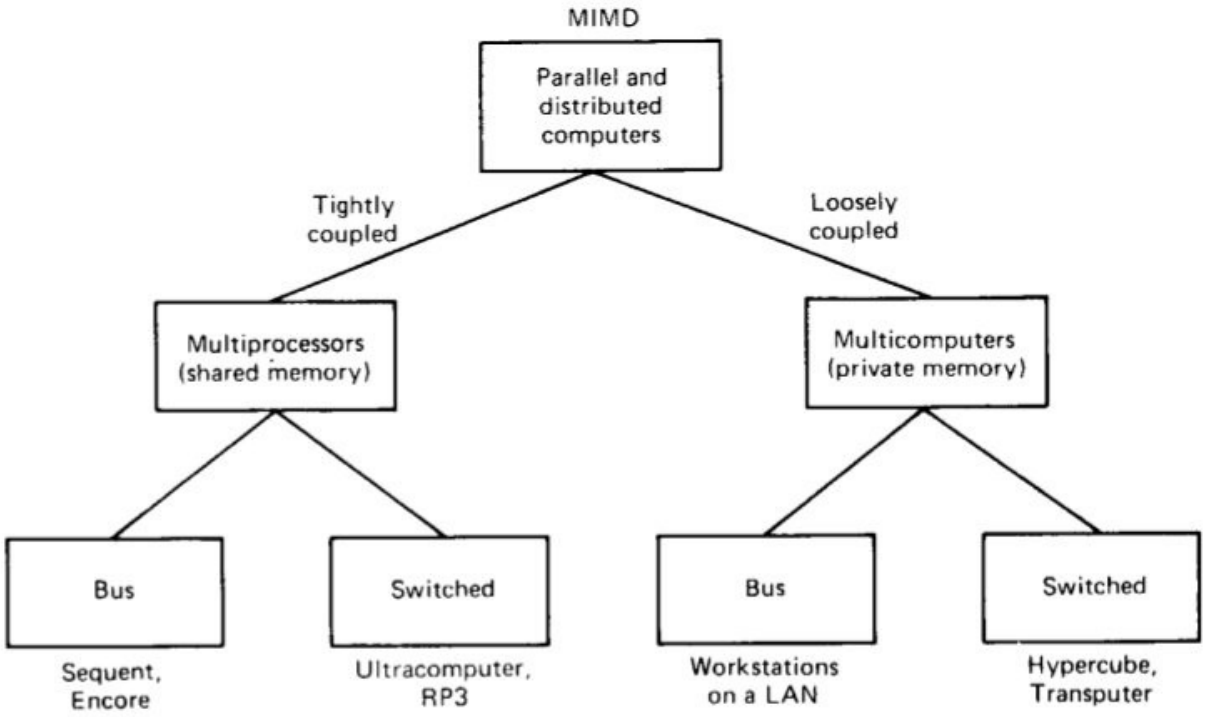

Fig: A taxonomy of parallel and distributed computer systems.

We divide all MIMD computers into two groups: those that have shared memory, usually called **Multiprocessors** and those that do not, sometimes called **Multicomputers.** The essential difference is this: in a multiprocessor, there is a single virtual address space that is shared by all CPUs. If any CPU writes, for example the address value 44 to address 1000, any other CPU subsequently reading from its address 1000 will get the value 44. All the machine share the same memory.

In contrast, in a multicomputer, every machine has its own private memory. If one CPU writes the value 44 to the address 1000, when another CPU reads the address 1000 it will get whatever value was there before. The write of 44 doesn't affect its memory at all. A common example of multicomputers is the collection of personal computers connected by the network.
Each of these category can be further divided into bus and switched based on architecture of the interconnection network.
By bus we mean that there is is single network, backbone, bus, cable, or other medium that connects all machine. Cable television uses a scheme like this, the cable company runs the cable down the street, and all the subscribers have taps running to it from their television sets.
Switched system do not have a single backbone like cable television, instead there are individual wires from machine to machine, with many different wiring patterns in use. Messages move along the wires, with an explicit switching decisions made at each step to route the message along one of the out going wires. The world-wide public telephone system is organized in this way.

In a tightly coupled system, the delay experienced when a message is sent from one computer to other another is short, and the data rate is high; that is the number of bits per second that can be transferred is large. In a loosely coupled system the opposite is true: the inter-machine delay is large and the date rate is low. For example two CPUs chips on the same printed board and connected by wires are likely to be tightly coupled, whereas two computers connected by a 2400 bits/sec modem over the telephone system are certain to be loosely coupled.

Tightly coupled system tend to be used more as parallel systems (working on a single problem) and loosely coupled system tend to be used as distributed system (working on many unrelated problems).