Chapter one

1. Introduction to distributed system

Computer systems are undergoing a revolution, from 1945, when the modern computer era began, until about 1985, computer was large and expensive. Even computers normally cost tens of thousands of dollars each. As a result, most organizations had only a handful of computers and for lack of a way to connect them, these operated independently from one another.

Starting from the mid-1980s, however, two advances in technology began to change the situation. The 1st was the development of powerful microprocessors. Initially, these were 8-bit, but soon 16-, 32-, and even 64-bit CPUs become common. Many of these had the computing power of a decent-sized mainframe (i.e., large) computer, but for a fraction of the price.

The 2nd development was the invention of high-speed computer network. The local area networks or LANs allow dozens, or even hundreds, of machines within a building to be connected in such a way that small amount of information can be moved between machines in a millisecond or so. Larger amount of data can be moved between machines at rates of 10 to 100 million bits/sec and sometimes more. The wide area networks or WANs allows millions of machines all over the earth to be connected at speed of varying from 64Kbps to gigabits per second for some advanced experimental networks.
The result of these technologies is that it not only feasible, but easy to put together computing systems composed of large number of CPUs connected by a high-speed network. They are usually called distributed systems, in contrast to the previous centralized systems (or single processor systems) consisting of a single CPU, its memory, peripherals, and some terminals.

2. What is distributed system?

A distributed system is a collection of independent computers that appear to the users of the system as a single computer. This definition has two aspects. The first one deals with hardware: the machines are autonomous. The second one deals with software: the user think of the system as a single computer. And both are essential.
In order to support heterogeneous computers and networks while offering a single-system view, distributed systems are often organized by means of a layer of software-that is, logically placed between a higher-level layer consisting of users and applications, and a layer underneath consisting of

operating systems and basic communication facilities, as shown in Fig. 1-1 Accordingly, such a distributed system is sometimes called middleware.

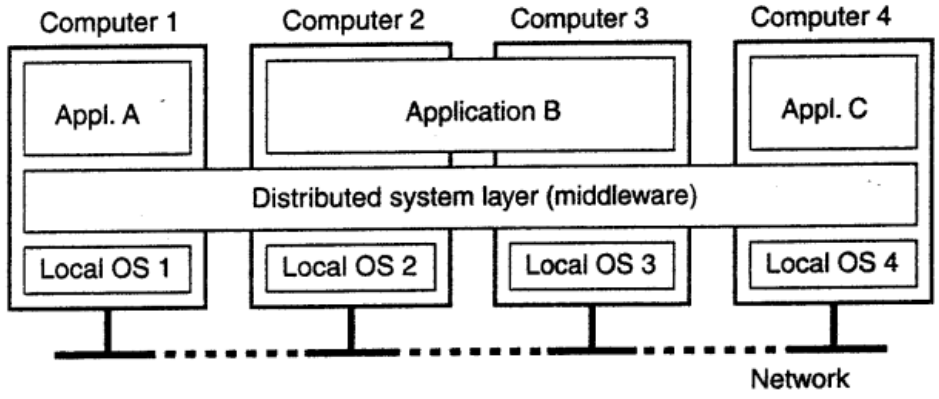

Figure I-I. A distributed system organized as middleware. The middleware layer extends over multiple machines, and offers each application the same interface.

Fig. 1-1 shows four networked computers and three applications, of which application B is distributed across computers 2 and 3. Each application is offered the same interface. The distributed system provides the means for components of a single distributed application to communicate with each other, but also to let different applications communicate. At the same time, it hides, as best and reason-able as possible, the differences in hardware and operating systems from each application.

3. Examples of Distributed System

Rather than going further with definitions, it is probably more helpful to give several examples of distributed systems. As a first example, consider a network of workstations in a university or company department. In addition to each user's personal workstation, there might be a pool of processors in the machine room that are not assigned to specific users but are allocated dynamically as needed. Such a system might have a single file system, with all files accessible from all machines in the same way and using the same path name. Furthermore, when a user typed a command, the system could look for the best place to execute that command, possibly on the user's own workstation, possibly on an idle

workstation belonging to someone else, and possibly on one of the unassigned processors in the machine room. If the system as a whole looked and acted like a classical single-processor timesharing system, it would qualify as a distributed system.

As a second example, consider a factory full of robots, each containing a powerful computer for handling vision, planning, communication, and other tasks. When a robot on the assembly line notices that a part it is supposed to install is defective, it asks another robot in the parts department to bring it a replacement. If all the robots act like peripheral devices attached to the same central computer and the system can be programmed that way, it too counts as a distributed system.

As a final example, think about a large bank with hundreds of branch offices all over the world. Each office has a master computer to store local accounts and handle local transactions. In addition, each computer has the ability to talk to all other branch computers and with a central computer at headquarters. If transactions can be done without regard to where a customer or account is, and the users do not notice any difference between this system and the old centralized mainframe that it replaced, it too would be considered a distributed system.

4. Characteristics of Distributed system

- The way of communication between nodes (an entity with a capability of sending and receiving message in a distributed system) are hidden from the user.
- Users and applications are in a distributed system can interact in a consistent and uniform way
- Relatively easy to expand or scale. Because distributed system is a collection of independent computers
- A distributed system will normally be available although certain may be out of order. But, this status is not noticed to the user. That means:
  - ✓ Parts are replaced or fixed
  - ✓ New parts are added to serve more users or applications

5. Goals of Distributed system

Building a distributed system have four important goals that should be met to worth the effort. A distributed system should make resources easily accessible; it should reasonably hide the fact that resources are distributed across a network; it should be open; and it should be scalable.

5.1. Making Resources Accessible

The main goal of a distributed system is to make it easy for the users (and applications) to access remote resources, and to share them in a controlled and efficient way. Resources can be just about

anything, but typical examples include things like printers, computers, storage facilities, data, files, web pages, and net-works, to name just a few. There are many reasons for wanting to share resources. One obvious reason is that of economics. For example, it is cheaper to let a printer be shared by several users in a small office than having to buy and maintain a separate printer for each user. Likewise, it makes economic sense to share costly re-sources such as supercomputers, high-performance storage systems, imagesetters, and other expensive peripherals.

Connecting users and resources also makes it easier to collaborate and ex-change information, as is clearly illustrated by the success of the Internet with its simple protocols for exchanging files, mail. documents, audio, and video. The connectivity of the Internet is now leading to numerous virtual organizations in which geographically widely-dispersed groups of people work together by means of groupware, that is, software for collaborative editing, teleconferencing, and soon. Likewise, the Internet connectivity has enabled electronic commerce allowing us to buy and sell all kinds of goods without actually having to go to a store or even leave home.

5.2. Distribution Transparency

An important goal of a distributed system is to hide the fact that its processes and resources are physically distributed across multiple computers. A distributed system that is able to present itself to users and applications as if it were only a single computer system is said to be transparent. Let us first take a look at what kinds of transparency exist in distributed systems. After that we will address the more general question whether transparency is always required.

5.2.1. Types of Transparency

The concept of transparency can be applied to several aspects of a distributed system, the most important ones shown in table below:

Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource is replicated
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource

Table 1: Different forms of transparency in a distributed system (ISO, 1995)

- **Access transparency:** deals with hiding differences in data representation and the way that resources can be accessed by users. At a basic level, we wish to hide differences in machine architectures, but more important is that we reach agreement on how data is to be represented by different machines and operating systems. For example, a distributed system may have computer systems that run different operating systems, each having their own file-naming conventions. Differences in naming conventions, as well as how files can be manipulated, should all be hidden from users and applications.
- **Location transparency:** refers to the fact that users cannot tell where a resource is physically located in the system. Naming plays an important role in achieving location transparency. In particular, location transparency can be achieved by assigning only logical names to resources, that is, names in which the location of a resource is not secretly encoded. An example of a such a name is the URL http://www.prenhall.com/index.html. which gives no clue about the location of Prentice Hall's main Web server. The URL also gives no clue as to whether index.html has always been at its current location or was recently moved there.
- **Replication** also plays a very important role in distributed systems. For example, resources may be replicated to increase availability or to improve performance by placing a copy close to the place where it is accessed. Replication transparency deals with hiding the fact that several copies of a resource exist. To hide replication from users, it is necessary that all replicas have the same name. Consequently, a system that supports replication transparency should generally support location transparency as well, because it would otherwise be impossible to refer to replicas at different locations.
- In some cases, it is important that each user does not notice that the other is making use of the same resource. This phenomenon is called **concurrency transparency**. An important issue is that concurrent access to a shared resource leaves that resource in a consistent state. **Consistency** can be achieved through locking mechanisms, by which users are, in turn, given exclusive access to the desired resource.
- And finally, making a distributed system **failure transparent** means that a user does not notice that a resource (he has possibly never heard of) fails to work properly, and that the system subsequently recovers from that failure. Masking failures is one of the hardest issues in distributed systems and is even impossible when certain apparently realistic assumptions are made

5.2.2. An important group of transparency

We can group distributed transparency in to five groups:

Item	Description
Economics	Microprocessors offer a better price/performance than mainframes
Speed	A distributed system may have more total computing power than a mainframe
Inherent distribution	Some applications involve spatially separated machines
Reliability	If one machine crashes, the system as a whole can still survive
Incremental growth	Computing power can be added in small increments

Table 2: Groups of transparency in distributed system

6. Hardware and Software concepts in DS

6.1. Hardware concept

Various classification schemes for multiple CPU computer systems have been proposed over the years, but none of them have really caught on and been widely adopted. Four our purposes, we consider only systems built from a collection of independent computers. Even though all distributed system consists of multiple CPUs, there are several different ways the hardware can be organize, especially in terms of how they are interconnected and how they communicate. So, all computers in DS can be divided into two categories

1. Multi processors: those that have shared memory and
2. Multicomputer: those do not have have shared memory

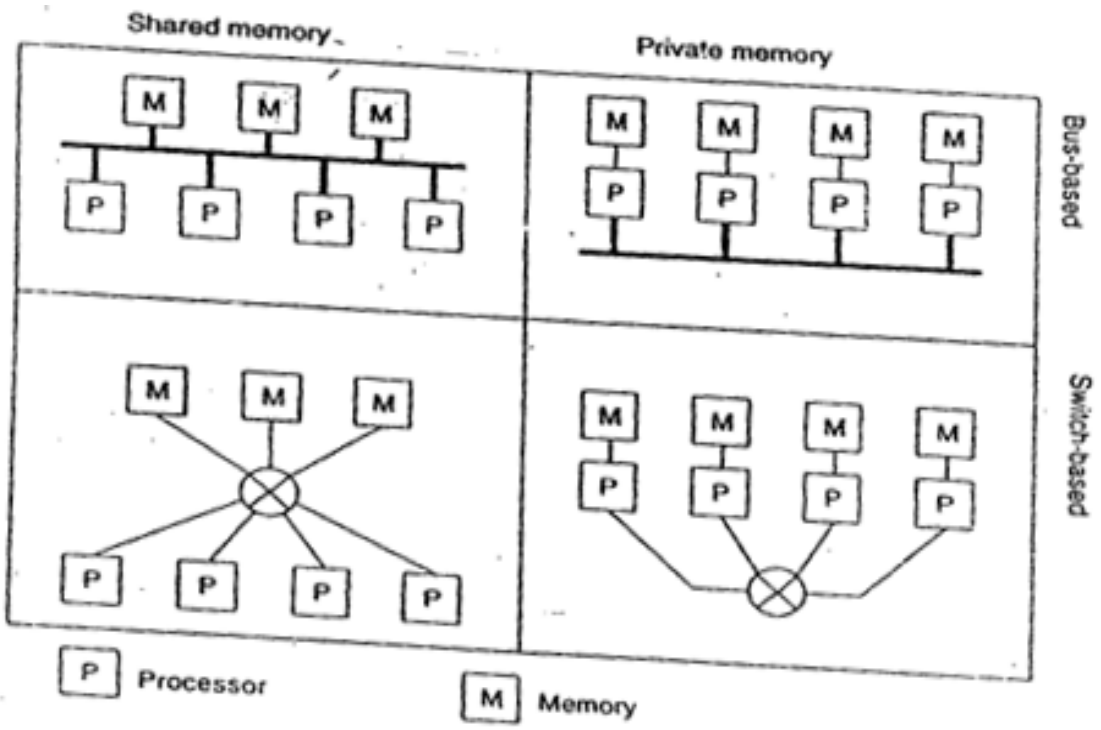

Fig. : Different basic organizations of processors and memories in distributed systems

6.1.1. Homogeneous multicomputer system

In contrast to multiprocessors building multicomputer is relatively easy. Each CPU has a direct connection to its local memory. The only problem is how the CPU communicate with each other. This kind of system referred to as System Area Networks(SANs)

- In a bus-based multi-computer the processors are connected through a shared multiaccess network such as Fast Ethernet.
- And in Switch-Based multicomputer, messages between the processors are routed through an interconnection network instead of broadcast as in bus-based systems.

6.1.2. Heterogeneous Multicomputer System

Most DSs are built on top of heterogeneous multicomputer systems. This means that, the computer that form part are vary widely with respect to each other. E.g. Processor type, memory size and I/O bandwidth.

6.2. Software Concept

H/W for DSs is important, but its s/w that largely determines what a distributed system actually looks like:

- First, they are resource managers for underlying h/w.
- Allowing multiple users and applications to share resources such as CPU, memory, peripheral devices, the network and data.

6.2.1. Operating System

Operating system for distributed system can be roughly divided into two categories:

- Tightly coupled systems and
- Loosely-coupled systems

Tightly coupled systems: The operating system is essentially tries to maintain a single global view of resources it manages and **Loosely-coupled systems** can be thought of as a collection of computers each running their own operating system. However, these operating system work together to make their own services and resources to the other.

- The tightly-coupled systems referred to as a Distributed System (DOS) and it is used to managing multiprocessor and homogenous multicomputer.
- The loosely-coupled Network Operating System (NOS) is used for heterogenous computer systems. To actually come to distributed system, the enhancement to the services of NOS are needed. And this enhancement known as **middleware**.

System	Description	Main goal
DOS	Tightly-coupled operating system for multiprocessors and homogeneous multicomputer	Hide and mange hardware resources
NOS	Loosely-coupled operating system for heterogeneous multicomputer (LAN and WAN)	Offer local services to remote clients
Middleware	Additional layer at top NOS implementing general purpose services	Provide distributed transparency

Table 3: overview of DOS, NOS and Middleware

7. Advantages of distributed system

Item	Description
Data sharing	Allow many users access to a common data base
Device sharing	Allow many users to share expensive peripherals like color printers
Communication	Make human-to-human communication easier, for example, by electronic mail
Flexibility	Spread the workload over the available machines in the most cost effective way

Table 4: Advantages of Distributed system over independent computers

8. Challenges (or limitations) of distributed system

Item	Description
Software	Little software exists at present for distributed systems
Networking	The network can saturate or cause other problems
Security	Easy access also applies to secret data

Table 5: Challenges of distributed system

End of Chapter One