INTRODUCTION TO DISTRIBUTED SYSTEMS

INTRODUCTION

The process of computation was started from working on a single processor. This uni-processor computing can be termed as **centralized computing**. As the demand for the increased processing capability grew high, multiprocessor systems came to existence. The advent of multiprocessor systems, led to the development of distributed systems with high degree of scalability and resource sharing. The modern day parallel computing is a subset of distributed computing

A distributed system is a collection of independent computers, interconnected via a network, capable of collaborating on a task. Distributed computing is computing performed in a distributed system.

A distributed system is a collection of independent entities that cooperate to solve a problem that cannot be individually solved. Distributed computing is widely used due to advancements in machines; faster and cheaper networks. In distributed systems, the entire network will be viewed as a computer. The multiple systems connected to the network will appear as a single system to the user. Thus the distributed systems hide the complexity of the underlying architecture to the user. Distributed computing is a special version of parallel computing where the processors are in different computers and tasks are distributed to computers over a network.

The definition of distributed systems deals with two aspects that:

Deals with hardware: The machines linked in a distributed system are autonomous.

Deals with software: A distributed system gives an impression to the users that they are dealing with a single system.

Features of Distributed Systems:

No common physical clock - This is an important assumption because it introduces the element of "distribution" in the system and gives rise to the inherent asynchrony amongst the processors.

No shared memory - A key feature that requires message-passing for communication. This feature implies the absence of the common physical clock.

Geographical separation – The geographically wider apart that the processors are, the more representative is the system of a distributed system**.**

Autonomy and heterogeneity – Here the processors are "loosely coupled" in that they have different speeds and each can be running a different operating system.

Issues in distributed systems

Heterogeneity

Openness

Security

Scalability

Failure handling

Concurrency

Transparency

Quality of service

QOS parameters

The distributed systems must offer the following QOS:

- Performance
- Reliability
- Availability
- Security

Differences between centralized and distributed systems

Centralized Systems	Distributed Systems
In Centralized Systems, several jobs are done on a particular central processing unit(CPU)	In Distributed Systems, jobs are distributed among several processors. The Processor are interconnected by a computer network
They have shared memory and shared variables.	They have no global state (i.e.) no shared memory and no shared variables.
Clocking is present.	No global clock.

RELATION TO COMPUTER SYSTEM COMPONENTS

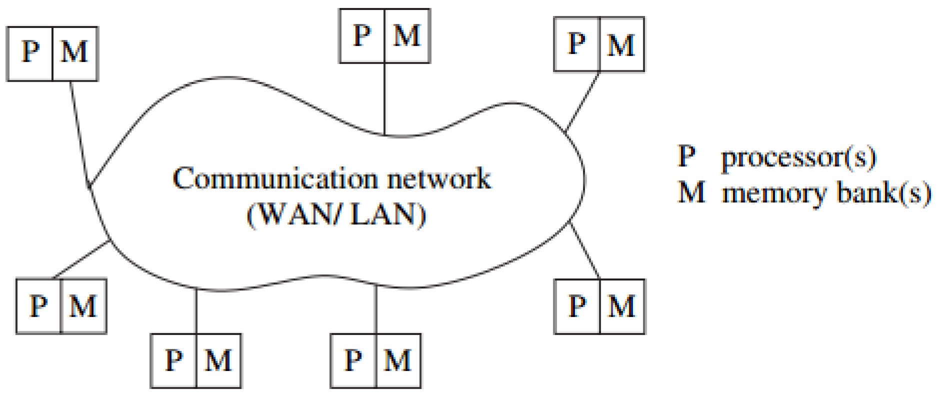

Fig : Example of a Distributed System

As shown in Fig., Each computer has a memory-processing unit and the computers are connected by a communication network. Each system connected to the distributed networks hosts distributed software which is a middleware technology. This drives the Distributed System (DS) at the same time preserves the heterogeneity of the DS. The term **computation or run** in a distributed system is the execution of processes to achieve a common goal.

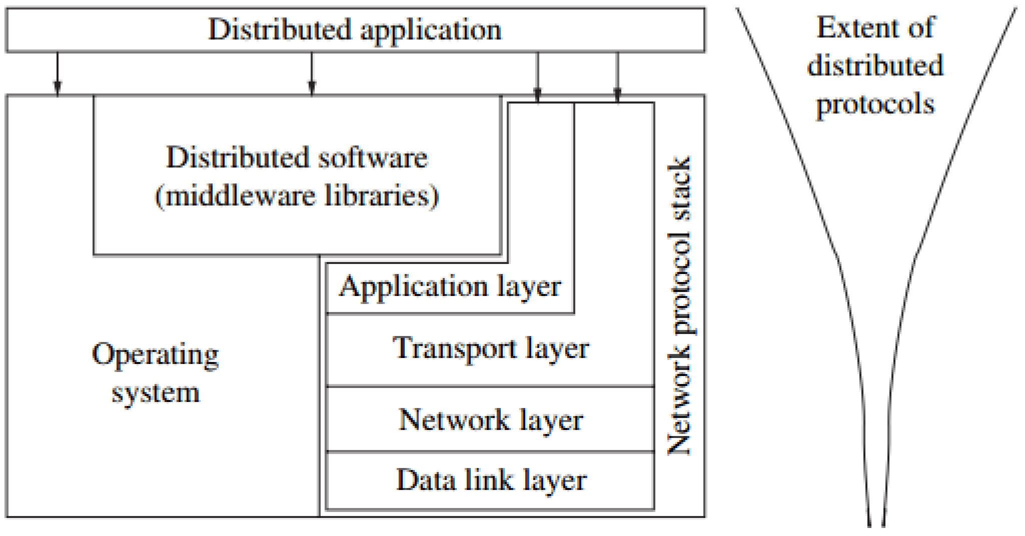

Fig : Interaction of layers of network

The interaction of the layers of the network with the operating system and middleware is shown in Fig. The middleware contains important library functions for facilitating the operations of DS.

The distributed system uses a layered architecture to break down the complexity of system design. The middleware is the distributed software that drives the distributed system, while providing transparency of heterogeneity at the platform level

Examples of middleware: Object Management Group's (OMG), Common Object Request Broker Architecture (CORBA) [36], Remote Procedure Call (RPC), Message Passing Interface (MPI)

MOTIVATION

The following are the keypoints that acts as a driving force behind DS:

Inherently distributed computations: DS can process the computations at geographically remote locations.

Resource sharing: The hardware, databases, special libraries can be shared between systems without owning a dedicated copy or a replica. This is cost effective and reliable.

Access to geographically remote data and resources: As mentioned previously, computations may happen at remote locations. Resources such as centralized servers can also be accessed from distant locations.

Enhanced reliability: DS provides enhanced reliability, since they run on multiple copies of resources. The distribution of resources at distant locations makes them less susceptible for faults.

The term reliability comprises of:

1. **Availability:** the resource/ service provided by the resource should be accessible at all times
2. **Integrity:** the value/state of the resource should be correct and consistent.
3. **Fault-Tolerance:** the ability to recover from system failures

Increased performance/cost ratio: The resource sharing and remote access features of DS naturally increase the performance / cost ratio.

Scalable: The number of systems operating in a distributed environment can be increased as the demand increases.