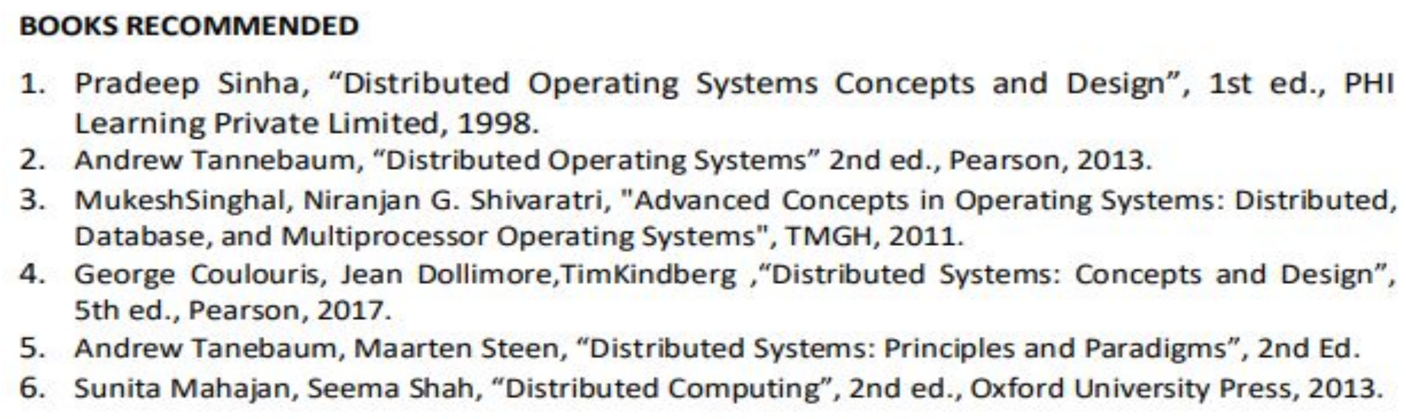
Distributed Systems(CSE604)

M.Tech. I (CSE) Semester – II	L	T	P	C
CSE604: DISTRIBUTED SYSTEMS (CORE -6)	3	0	2	4

Course Objective	
1	To learn the principles, architectures, algorithms and programming models used in distributed systems
2	To understand design and implementations issues in distributed systems.
3	To understand scheduling in distributed operating systems, process management, fault tolerance, real-time distributed systems, and designing of distributed file systems
4	To study the distributed resource and process management components.
5	To study advanced topics related to distributed operating systems.

INTRODUCTION TO DISTRIBUTED SYSTEMS	**(04 Hours)**
Review of Networking Protocols, Point to Point Communication, Operating Systems, Concurrent Programming, Characteristics and Properties of Distributed Systems, Goals of Distributed Systems, Multiprocessor and Multicomputer Systems, Distributed Operating Systems, Network Operating Systems, Middleware Concept, The Client-Server Model, Design Approaches-Kernel Based-Virtual Machine Based, Application Layering.	
COMMUNICATION IN DISTRIBUTED SYSTEM	**(05 Hours)**
Layered Protocols, Message Passing-Remote Procedure Calls(RPC), Remote Method Invocation(RMI), Message Oriented Communication, Stream Oriented Communication, Case Studies.	
SYNCHRONIZATION IN DISTRIBUTED SYSTEM	**(08 Hours)**
Clock Synchronization, Logical Clocks, Global State, Election Algorithms-The Bully algorithm-A Ring algorithm, Mutual Exclusion- Centralized Algorithm, Distributed Algorithm, Token ring Algorithm, Distributed Transactions, Distributed deadlock detection.	
DISTRIBUTED SHARED MEMORY	**(06 Hours)**
Introduction, General architecture of DSM systems, Design and implementation issues of DSM, Granularity, Structure of shared memory space, consistency models, Replacement strategy, Thrashing.	
RESOURCE MANAGEMENT	**(05 Hours)**
Desirable features of scheduling algorithm, Task assignment approach, Load balancing and Load sharing approach.	
PROCESS MANAGEMENT	**(03 Hours)**
Concept of Threads, Process, Processor allocation, Process Migration and Related Issues, Software Agents, Scheduling in Distributed System, Load Balancing and Sharing Approaches, Fault tolerance, Real time Distributed System	
DISTRIBUTED FILE SYSTEM	**(05 Hours)**
Introduction, Architecture, Mechanisms for Building Distributed File Systems-Mounting-Caching-Hints-Bulk Data Transfer-Encryption, Design issues-Naming and Name Resolution-Caches on Disk or Main Memory-Writing Policy-Cache consistency-Availability-Scalability-Semantics, Case Studies, Log Structured File Systems	
ADVANCED TOPICS	**(04 Hours)**
Introduction of Security in Distributed OS, Overview of security techniques, Features, Need, Access Control, Security Management, Micro Services Architecture, Lockless Data Structures, Distributed/Scalable Messaging Architecture, AMQP.	
CASE STUDY	**(02 Hours)**
Amoeba, Mach, Chorus and their comparison.	
Practical assignments will be based on the coverage of above topics.	**(28 Hours)**
(Total Contact Time: 42 Hours + 28 Hours = 70 Hours)	

Textbook & Readings



BOOKS RECOMMENDED

1. Pradeep Sinha, "Distributed Operating Systems Concepts and Design", 1st ed., PHI Learning Private Limited, 1998.
2. Andrew Tannebaum, "Distributed Operating Systems" 2nd ed., Pearson, 2013.
3. MukeshSinghal, Niranjan G. Shivaratri, "Advanced Concepts in Operating Systems: Distributed, Database, and Multiprocessor Operating Systems", TMGH, 2011.
4. George Coulouris, Jean Dollimore,TimKindberg ,"Distributed Systems: Concepts and Design", 5th ed., Pearson, 2017.
5. Andrew Tanebaum, Maarten Steen, "Distributed Systems: Principles and Paradigms", 2nd Ed.
6. Sunita Mahajan, Seema Shah, "Distributed Computing", 2nd ed., Oxford University Press, 2013.

Readings will also include papers.

Assignments:

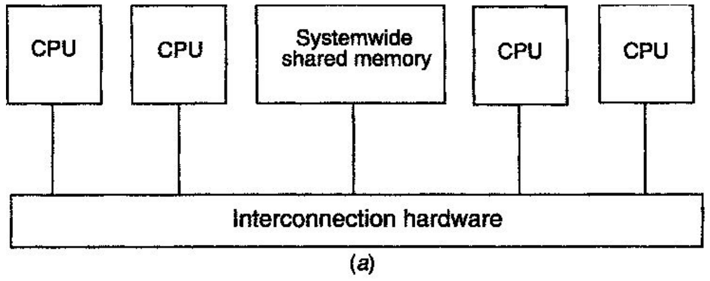
This course will involve listening to lectures, reading papers, writing reviews of papers, participating in class discussions, presenting papers and leading class discussions.

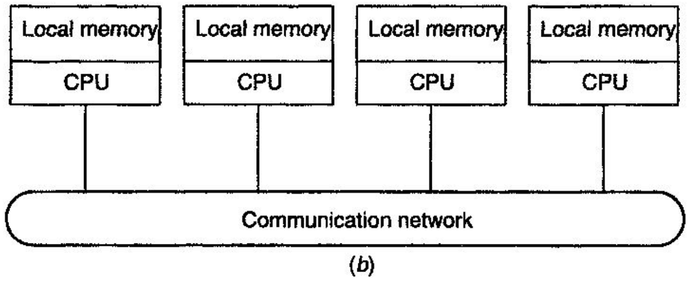
Students will be required to write reviews for papers they read.

Prologue

- Since 1945, computer systems are undergoing a revolution
- Until 1985, computers were large and expensive
- After 1980, two advances in technology began – microprocessors, computer networks
- Result – computer networks/distributed systems

Distributed (Computing) System (DCS): Definition





(a) tightly coupled – parallel processing systems

(b) Loosely coupled – distributed computing systems

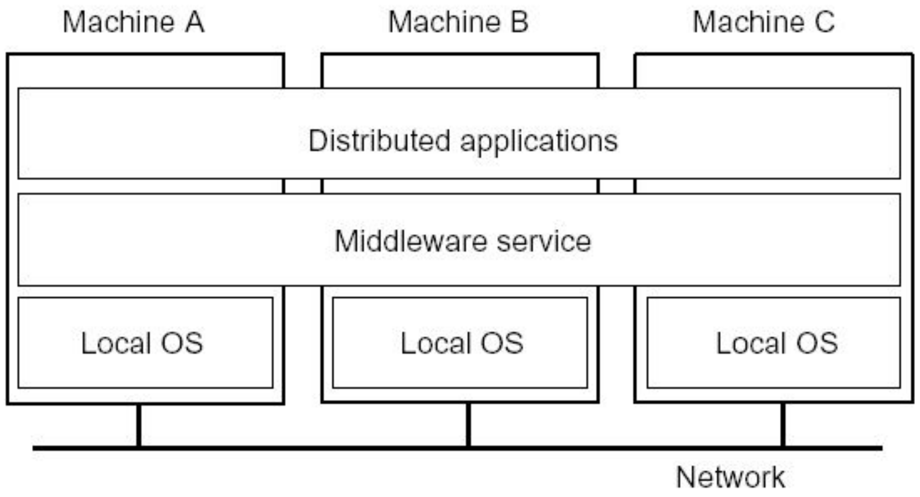
Distributed System (DS): Definition

A distributed system is
- a collection of independent computers that appears to its users as a single coherent system
- one in which components located at networked computers communicate and coordinate their actions only by passing messages

Two aspects:

(1) hardware - autonomous computers

(2) software – users think they are dealing with a single system



Characteristics of DS

- Hidden from the users: differences between various computers and the way they communicate in between
- Users and applications interact with DS in a consistent and uniform way regardless of time and place
- Should be easy to expand and scale
- Continuous available

Why DS gaining popularity?

- Inherently distributed applications
- Information sharing among distributed users
- Resource sharing
- Better price-performance ratio
- Shorter response times and higher throughput
- Higher reliability – degree of tolerance against failures/errors
- Extensibility and incremental growth
- Better flexibility in meeting users’needs

Examples of DS

- Network of workstations (NOW): a group of networked personal workstations connected to one or more server machines.
- The Internet/WWW
- An intranet: a network of computers and workstations within an organization, separated from the Internet via a protective device (a firewall).
- A large-scale distributed system – eBay
- Collection of Web servers: distributed database of hypertext and multimedia documents.
- Distributed file system on a LAN
- Domain Name Service (DNS)
- DBMS
- ATM network

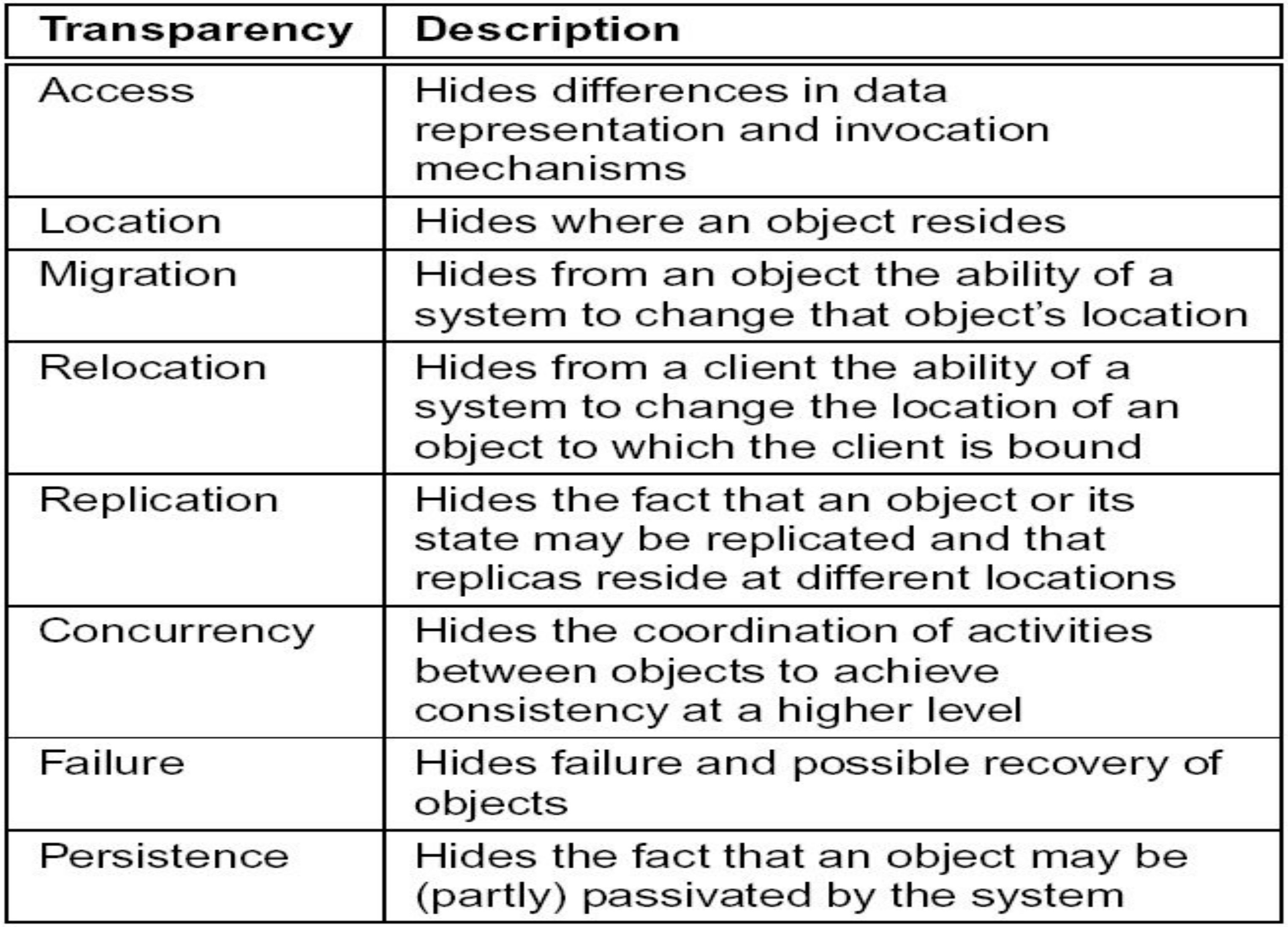
Types of DS

- Distributed Computing Systems – cluster, grid, cloud, etc.
- Distributed Information Systems – TPS, Enterprise application integration
- Distributed Pervasive Systems – Home systems, Electronic health care systems, Sensor networks

Goals of DS

- **Allow users to access and share resources**
- **Transparency**
  - To hide the fact that its processes and resources are physically distributed across multiple computers
  - A DS that is able to present itself to users and applications as if it were only a single computer system is said to be transparent
- **Openness**
  - To offer services according to standard rules that describe the syntax and semantics of those services
  - E.g., specify interfaces using an interface definition language (IDL)
- **Scalability**

Transparency



Transparency	Description
Access	Hides differences in data representation and invocation mechanisms
Location	Hides where an object resides
Migration	Hides from an object the ability of a system to change that object's location
Relocation	Hides from a client the ability of a system to change the location of an object to which the client is bound
Replication	Hides the fact that an object or its state may be replicated and that replicas reside at different locations
Concurrency	Hides the coordination of activities between objects to achieve consistency at a higher level
Failure	Hides failure and possible recovery of objects
Persistence	Hides the fact that an object may be (partly) passivated by the system

Cont..

- **Access Transparency**
- ex. Send an integer from Intel system(little endian) to SPARC(big endian)
- Difference in data representations due to different Oss
- **Location Transparency**
- Can be achieved by assigning only logical names
- Ex. URL framing
- **Replication Transparency**
- It is necessary to have same name to all replicas
- Should support ???? transparency
- **Concurrency Transparency**
- Should leave shared resource in a consistent state
- Possible using locking mechanisms

Cont..

- **Failure Transparency**
- Masking failures – one of the hardest issues in DS
- Inability to differentiate between a dead resource and a painfully slow resource
- Ex – time out response of web server? Really down or busy?
- **Persistence Transparency**
- Ex. Object oriented databases provide facilities for directly invoking methods on stored objects
- User should unaware that server is moving state between primary and secondary memory

Degree of Transparency

- In general transparency is preferred, but not always good to blindly hid all distribution aspects from users
- Ex. Requesting your e-news paper in your mailbox at 7am while you are at the other end of the globe living with different time zone
- Due to computer network/switches/routers/gateways – it is not possible to send message from one process(China) to another(Europe) in predefined time
- Trade-off between high degree of transparency and performance of system
- Ex. Many internet applications try to contact server before finally giving up. Attempting to mask a transient server failure before trying another one may slow down the system as a whole
- Hiding several replicas located on different continents – changes take time due to network and other aspects
- Conclusion – transparency is required but should also consider performance.

Openness of DS

Open distributed system: that offers services according to standard rules that describes the syntax and semantics of those services like in computer networks protocols are implemented.

- Be able to interact with services from other open systems, irrespective of the underlying environment:

- Systems should conform to well-defined **interfaces**
- Systems should support **portability** of applications
- Systems should easily **interoperate**
- Systems should be flexible

Achieving openness: At least make the distributed system independent from **heterogeneity** of the underlying environment:

- Hardware
- Platforms
- Languages

Scalability in DS

- Most important design goals
- It can be measured along at least three different dimensions (Neuman, 1994):
  - Number of users and/or processes **(size scalability)**
  - Maximum distance between nodes **(geographical scalability)**
  - Number of administrative domains **(administrative scalability)**

- Most systems account only, to a certain extent, for size scalability.
- Today, the challenge lies in geographical and administrative scalability.

Techniques for Scaling

1. **Hiding communication latencies**
   - Applicable to geographical scalability
   - Try to avoid waiting for responses to remote service requests as much as possible
   - Possible using asynchronous communication
2. **Distribution**
   - Partition data and computations across multiple machines
   - Move computations to clients (Java applets)
   - Decentralized naming services (DNS)
   - Decentralized information systems (WWW)

Cont…

3. **Replication**

- Make copies of data available at different machines
- Replicated file servers (mainly for fault tolerance)
- Replicated databases
- Mirrored Web sites
- Large-scale distributed shared memory systems
- Helps to balance the load -> better performance

4. **Caching(special form of replication)**

- Allow client processes to access local copies:
- Web caches (browser/Web proxy)
- File caching (at server and client)
- Caching is a decision made by the client of a resource and not by the owner of a resource
- Problem with replication/caching - consistency

Distributed System Architecture

Hardware Architecture:

- Uniprocessor
- Multiprocessor
- Multicomputer

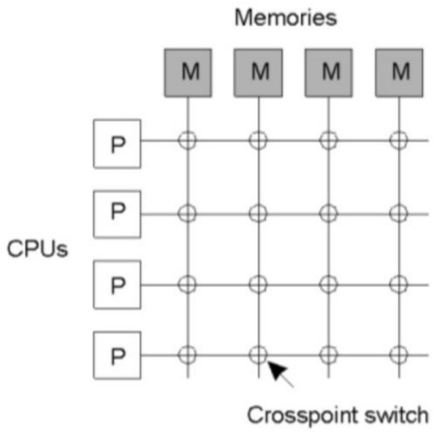
Software Architecture:

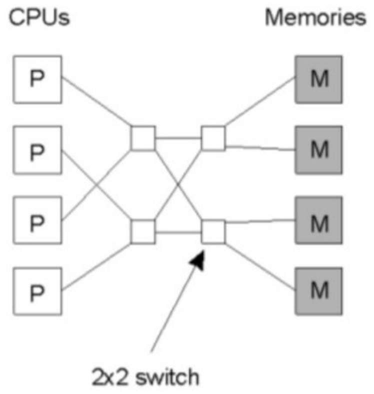
- Uniprocessor OS
- Multiprocessor OS
- Network OS (NOS)
- Distributed OS (DOS)
- Middleware

- A key characteristic of distributed systems that includes
  – a hardware aspect (independent computers)
  – and a software aspect (performing a task and providing a service)

Distributed Systems: Hardware Concepts

- Multiprocessors
  - All the CPUs have direct access to the shared memory
  - bus based – limited scalability
  - crosspoint switch – for large n, nxn switches are required
  - omega switching network – less switches
  - NUMA





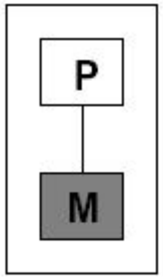
- Multicomputers
  - Homogeneous
  - Heterogeneous

- Networks of Computers
  - High degree of node and network heterogeneity

Uniprocessor

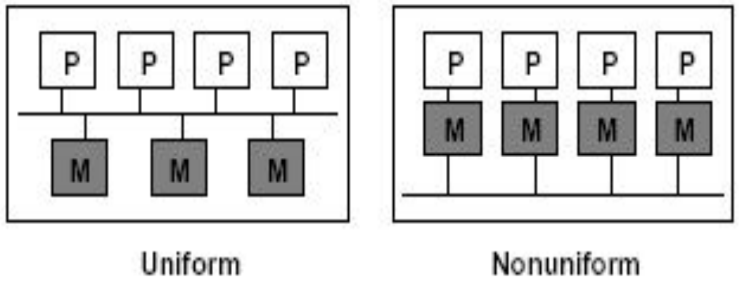
Properties:

1. Single processor
2. Direct memory access



Multiprocessors

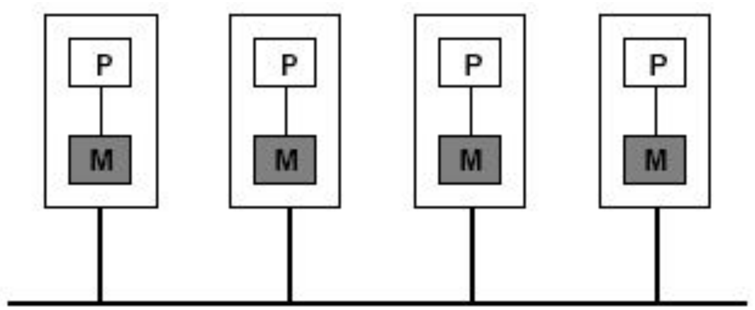
- Those that have shared memory, usually called multiprocessors
- Those that have **NOT** shared memory, usually called Multicomputers.
- Typically, there are 2 major types of Parallel Architectures that are common in the industry: Shared Memory Architecture and Distributed Memory Architecture.
- Shared Memory Architecture, again, is of two types
  - Uniform Memory Access (UMA)
  - Non-Uniform Memory Access (NUMA).



Properties:

1. Multiple process
2. Direct memory ac
3. Uniform memory access -access time to a memory location is independent of which processor makes the request
4. Non-uniform memory access- the memory access time depends on the memory location relative to a processor.

Multicomputers



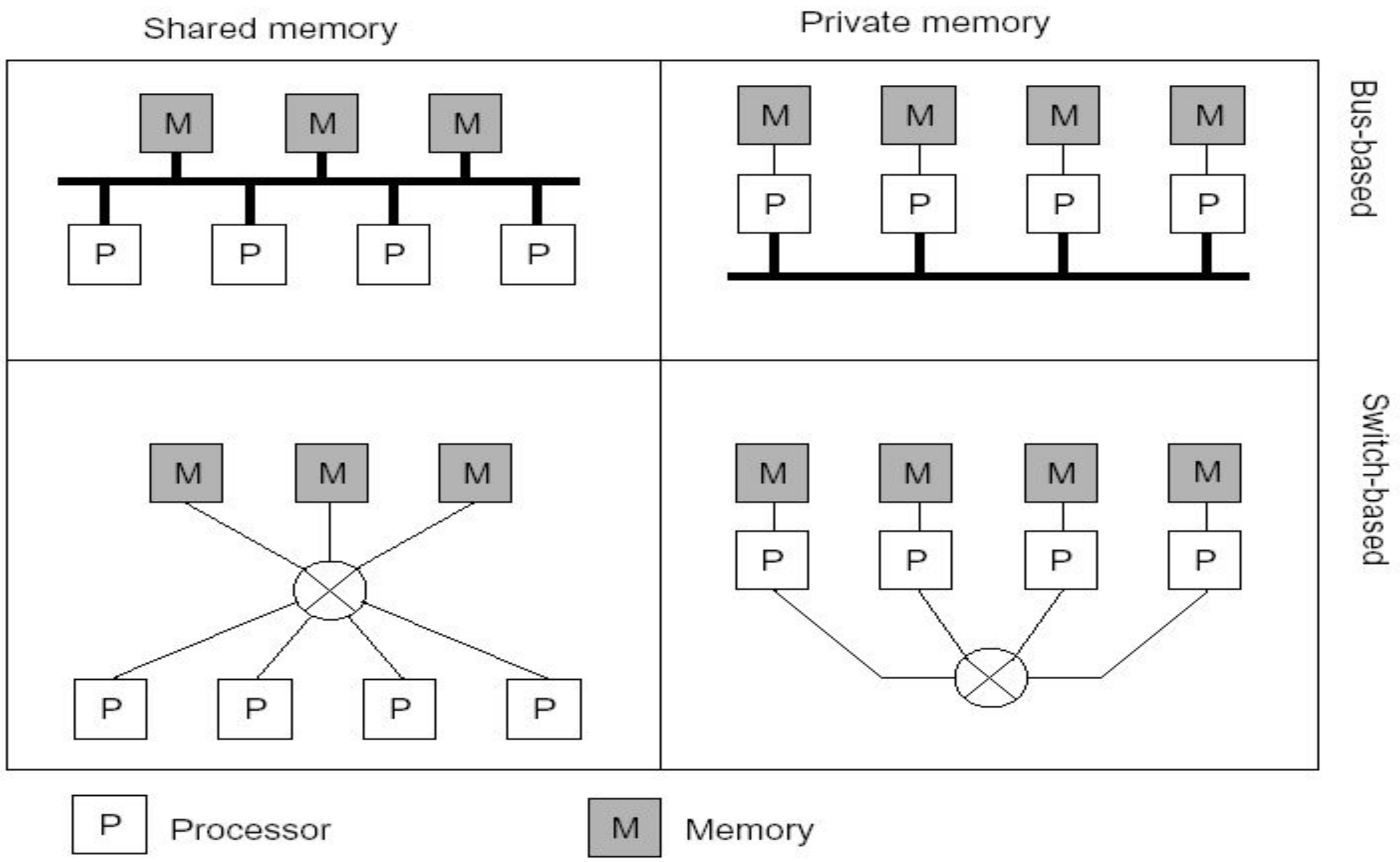
<u>Properties</u>:

1. Multiple computers
2. Network Connection (to support higher bandwidth)
3. Homogeneous (all nodes support same physical architecture – SAN, MPPs, COWs) vs. Heterogeneous (does not support same physical architecture)

Multiprocessors and Multicomputers

Distinguishing features:

- Private versus shared memory
- Bus versus switched interconnection



Distributed Systems : Software Concepts

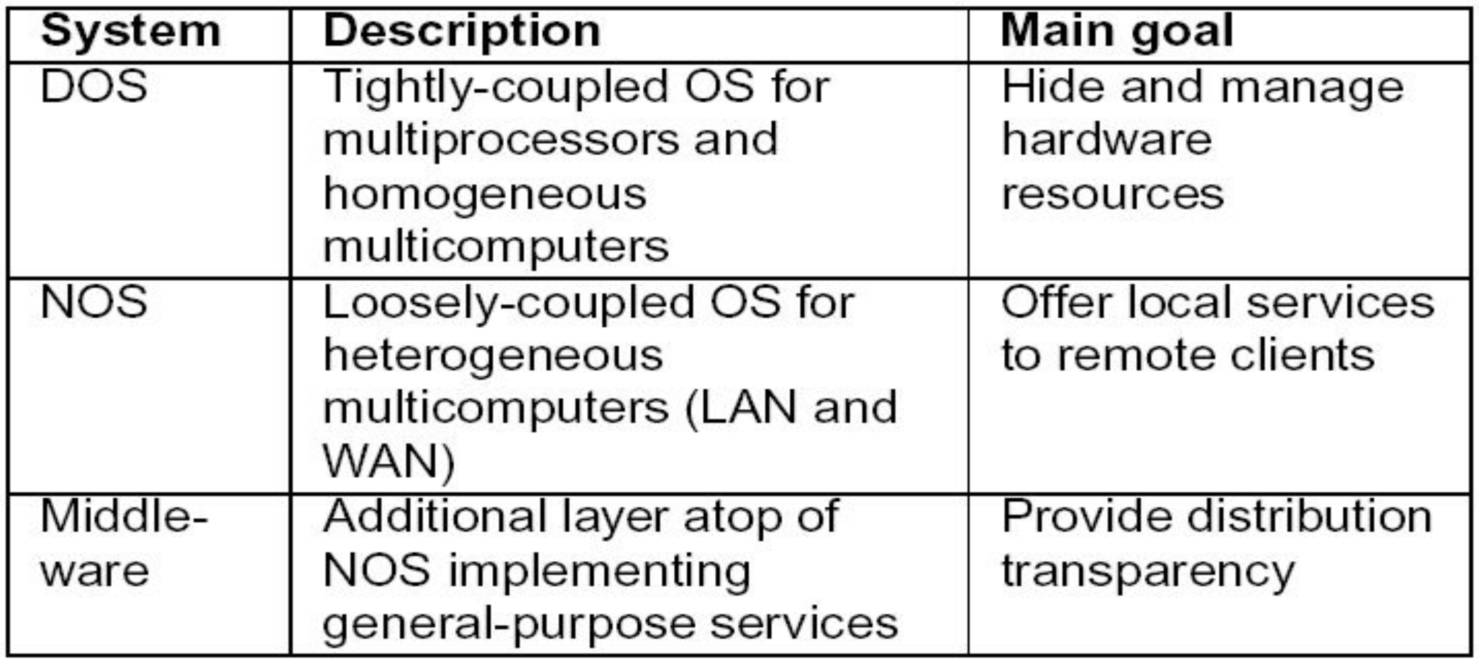
•OS for DS divided into two categories due to hardware
-Loosely coupled
-Tightly coupled

•Tightly coupled – OS tries to maintain a single, global view of the resources it manages
– DOS – manages multiprocessors and homogeneous multi computer system – aim is to hide intricacies

•Loosely coupled – collection of computers and each running their own OS
– NOS – heterogeneous multi computer system –local services available to remote clients
− Middleware – enhanced NOS – better support for distribution transparency – recent concept of DS – eq. cloud computing – running VM on top of hypervisor/middleware

Distributed Systems : Software Concepts

- Distributed Operating system(DOS)
- Network Operating System(NOS)
- Middleware

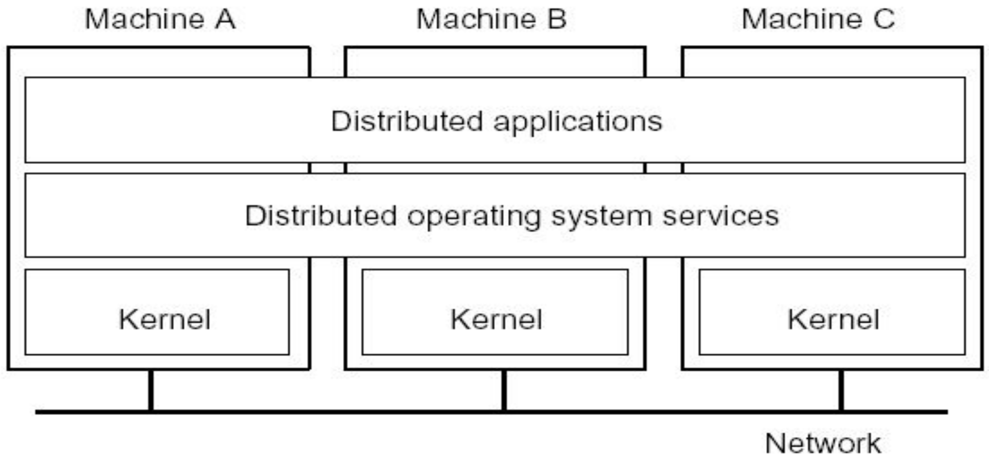


System	Description	Main goal
DOS	Tightly-coupled OS for multiprocessors and homogeneous multicomputers	Hide and manage hardware resources
NOS	Loosely-coupled OS for heterogeneous multicomputers (LAN and WAN)	Offer local services to remote clients
Middle-ware	Additional layer atop of NOS implementing general-purpose services	Provide distribution transparency

Distributed Operating System

Some characteristics:

- OS on each computer knows about the other computers
- OS on different computers generally the same
- Services are generally (transparently) distributed across computers (services may include distributed shared memory, assignment of tasks to processors, distributed storage, interprocess communication, transparent sharing of resources, distributed resource management, etc.)
- High degree of transparency (single system image)
- Homogeneous hardware



Multicomputer Operating System

- **Harder than traditional (multiprocessor) OS:** Because memory is not shared, emphasis shifts to processor communication by message passing:
- Often no simple global communication:
  - Only bus-based multicomputers provide hardware broadcasting
  - Efficient broadcasting may require network interface programming techniques
- No simple system-wide synchronization mechanisms
- Virtual (distributed) shared memory requires OS to maintain global memory map in software
- Inherent distributed resource management: no central point where allocation decisions can be made
- Example: Amoeba

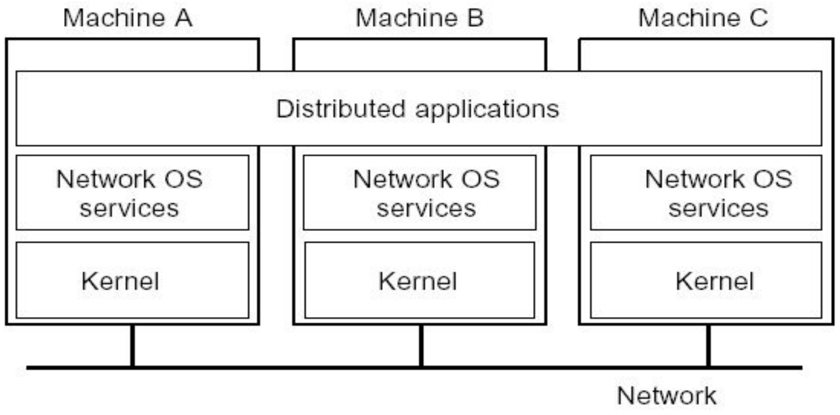
Network Operating System

Some characteristics:

- Each computer has its own operating system with networking facilities
- Computers work independently (i.e., they may even have different operating systems)
- Highly file oriented (basically, processors share only files)
- Services are tied to individual nodes (ftp, telnet,WWW)
- Access to resources of various machines is done explicitly by:

  - Remote logging into the appropriate remote machine (telnet)

  - Transferring data from remote machines to local machines, via the File Transfer Protocol (FTP) mechanism
- Examples: Linux, Windows



Need for Middleware

- Too many networked applications were difficult to integrate:

- Departments are running different NOSs

- Integration and interoperability only at level of primitive NOS Services

- Need for federated information systems:

  – Combining different databases, but providing a single view to applications

  – Setting up enterprise-wide Internet services, making use of existing information systems

  – Allow transactions across different databases

  – Allow extensibility for future services (e.g., mobility, teleworking, collaborative applications)

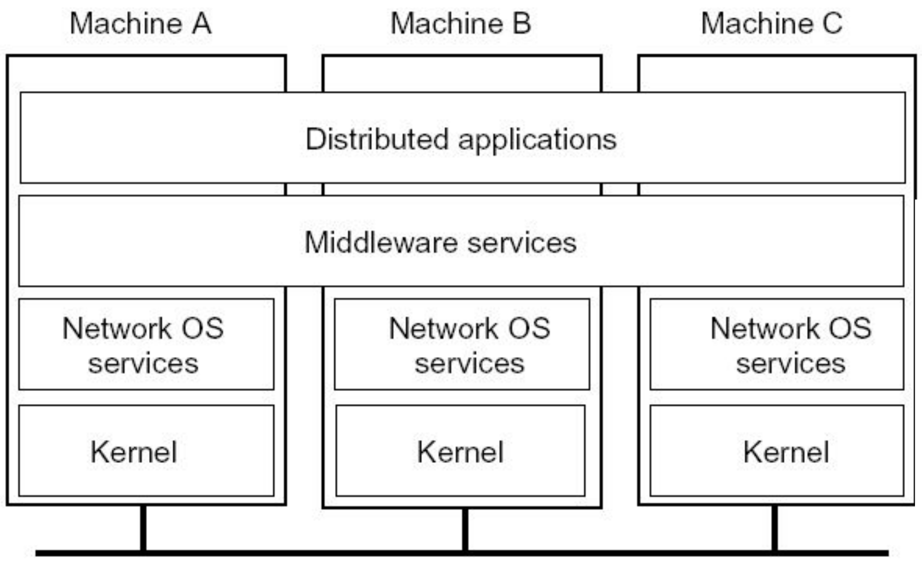
Middleware

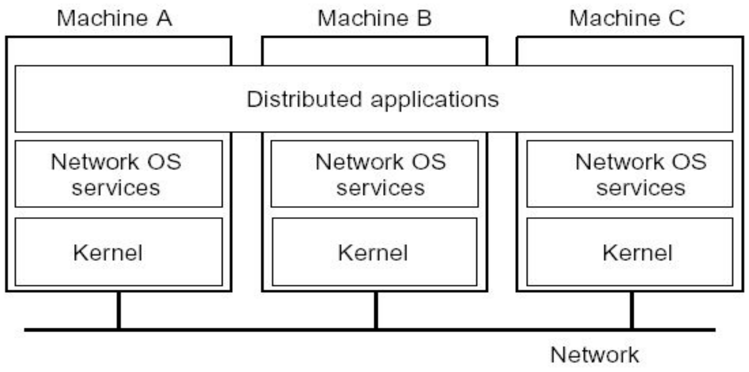
- As per definition, neither DOS or NOS really qualifies as a distributed system
- DOS does not intended to handle a collection of independent computers
- NOS does not provide a view of a single coherent system
- Is it possible to develop distributed system with both worlds:
- Scalability and openness of NOS
- Transparency and ease of use of DOS
- Solution: one more layer – middleware; may be a separate or part of NOS

Middleware

Some characteristics:

- OS on each computer need not know about the other computers
- OS on different computers need not generally be the same
- Services are generally (transparently) distributed across computers
- System independent interface for distributed programming
- Improves transparency (e.g., hides heterogeneity)
- Provides services (e.g., naming service, transactions, etc.)
- Provides programming model (e.g., distributed objects)





Middleware Services

Communication services: Abandon primitive socket-based message passing in favor of(for access transparency):

- Procedure calls across networks
- Remote-object method invocation
- Message-queuing systems
- Advanced communication streams
- Event notification service

Information system services: Services that help manage data in a distributed system:

- Large-scale, system-wide naming services
- Advanced directory services (search engines)
- Location services for tracking mobile objects
- Persistent storage facilities
- Data caching and replication

Middleware Services

Control services: Services giving applications control over when, where, and how they access data:

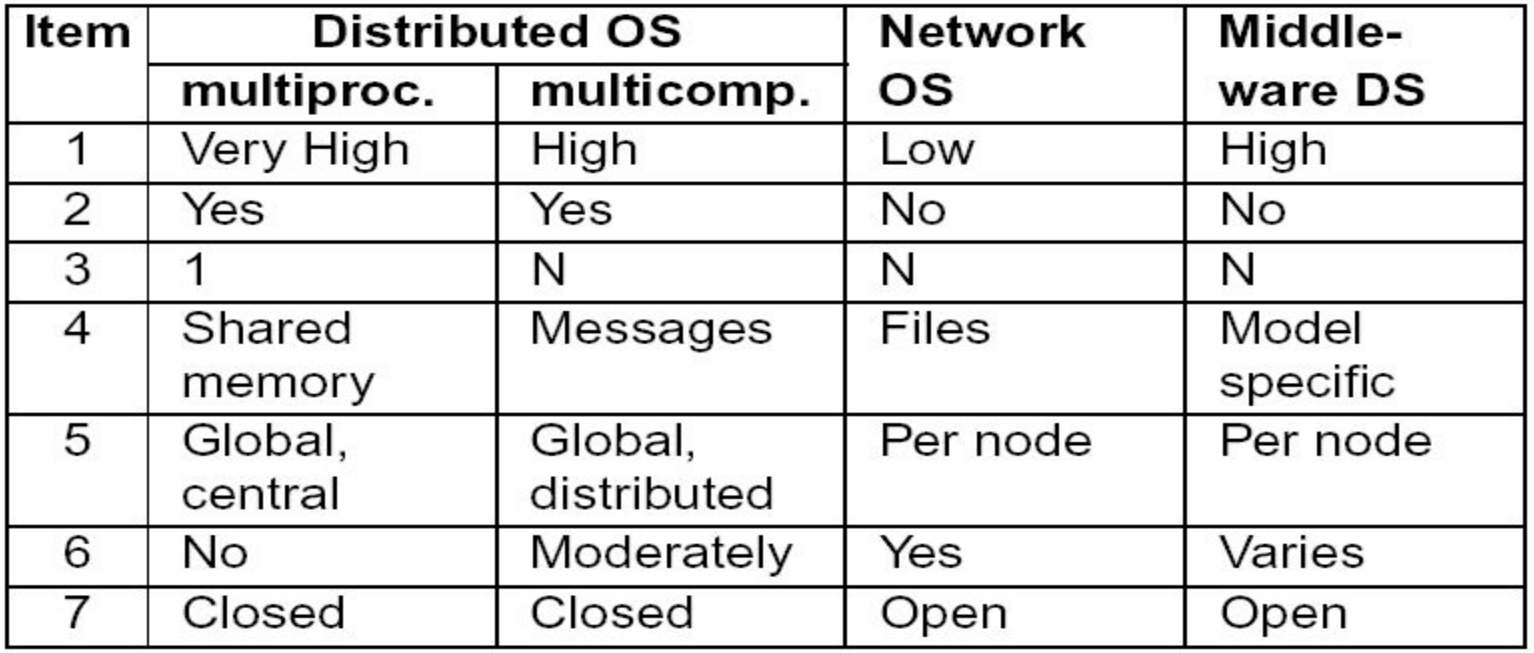
- Distributed transaction processing
- Code migration

Security services: Services for secure processing and communication:

- Authentication and authorization services
- Simple encryption services
- Auditing service

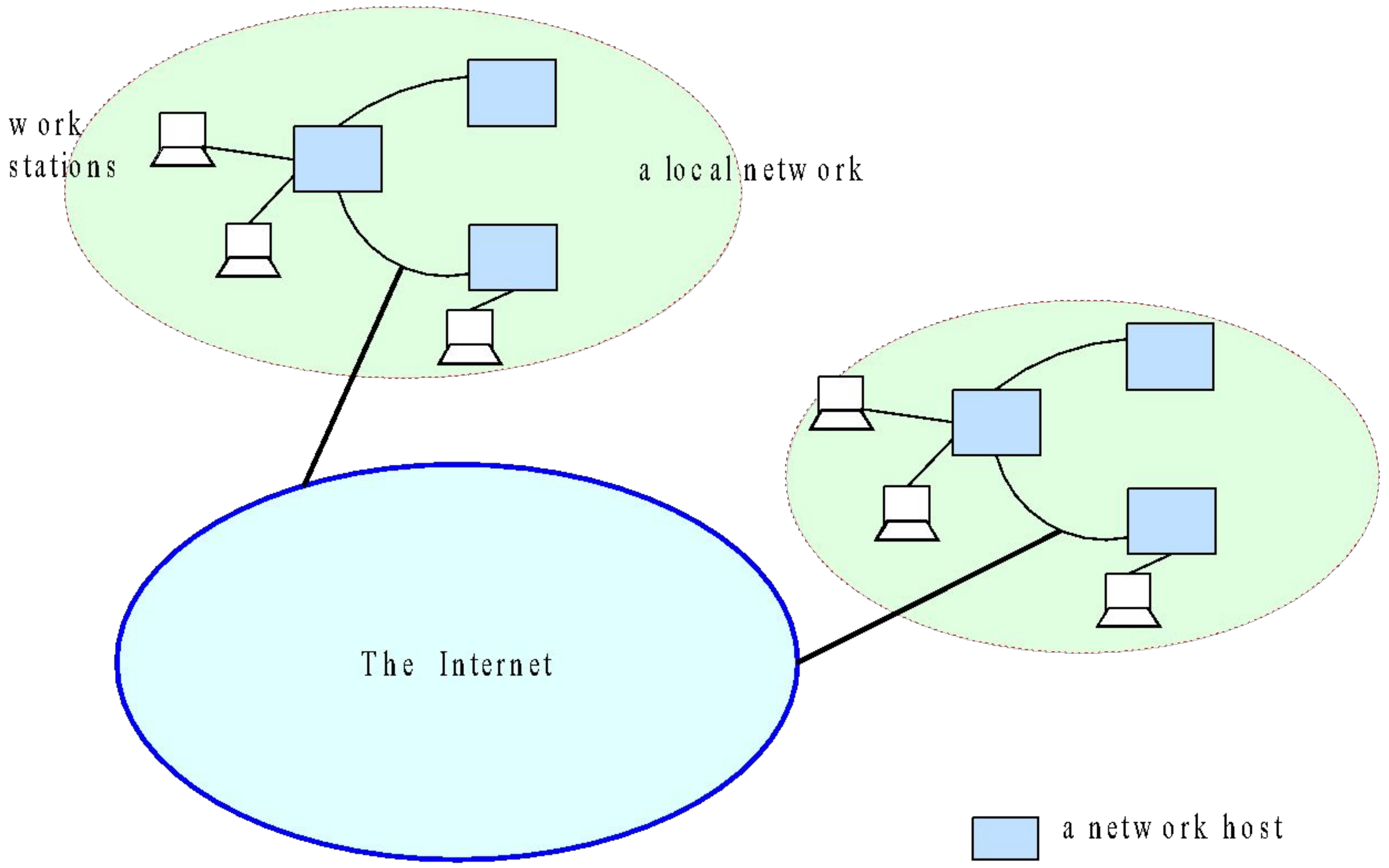
Comparison of DOS, NOS, and Middleware

1: Degree of transparency
2: Same operating system on each node?
3: Number of copies of the operating system
4: Basis for communication
5: How are resources managed?
6: Is the system easy to scale?
7: How open is the system?



Item	Distributed OS		Network OS	Middle-ware DS
	multiproc.	multicomp.		
1	Very High	High	Low	High
2	Yes	Yes	No	No
3	1	N	N	N
4	Shared memory	Messages	Files	Model specific
5	Global, central	Global, distributed	Per node	Per node
6	No	Moderately	Yes	Varies
7	Closed	Closed	Open	Open

Distributed Systems



Why Distributed?

- **Resource and Data Sharing**
  - Printers, databases, multimedia servers etc.
- **Availability, Reliability**
  - The loss of some instances can be hidden
- **Scalability, Extensibility**
  - System grows with demands (e.g. extra servers)
- **Performance**
  - Huge power (CPU, memory etc.) available
  - Horizontal distribution (same logical level is distr.)
- **Inherent distribution, communication**
  - Organizational distribution, e-mail, video conference
  - Vertical distribution (corresponding to org. struct.)

Problems of Distribution

- **Concurrency, Security**
  - Clients must not disturb each other
- **Partial failure**
  - We often do not know, where is the error (e.g. RPC)
- **Location, Migration, Replication**
  - Clients must be able to find their servers
- **Heterogeneity**
  - Hardware, platforms, languages, management
- **Convergence**
  - Between distributed systems and telecommunication

The Advantages Of Distributed Systems

- **Economics:**
- **Performance:** By using the combined processing and storage capacity of many nodes, performance levels can be reached that is out of the scope of centralized machines.
- **Scalability:** Resources such as processing and storage capacity can be increased incrementally.
- **Inherent distribution:** Some applications, such as email and the Web (where users are spread out over the whole world), are naturally distributed. This includes cases where users are geographically dispersed as well as when single resources (e.g., printers, data) need to be shared.
- **Reliability:** By having redundant components, the impact of hardware and software faults on users can be reduced.
- Better **flexibility** in meeting users’ needs.

The Disadvantages Of Distributed Systems

- **Multiple Points of Failures:** the failure of one or more participating computers, or one or more network links, can bring trouble.
- **Security Concerns**: In a distributed system, there are more opportunities for unauthorized attack.
- **Software complexity:** Distributed software is more complex and harder to develop than conventional software.

Distributed systems are hard to build and understand.

Basic Problems And Challenges In Distributed Systems

The distributed nature of a systems gives rise of the following challenges:

- Transparency
- Scalability
- Dependability
- Performance
- Flexibility

Principles

There are several key principles underlying all distributed systems. As such, any distributed system can be described based on how those principles apply to that system.

- System Architecture
- Communication
- Synchronization
- Replication and Consistency
- Fault Tolerance
- Security
- Naming

Paradigms

To make development and integration of distributed applications as simple as possible, most middleware of distributed systems are built based on a particular paradigm (or model) for describing distribution and communication.

- Shared memory
- Distributed objects
- Distributed file system
- Distributed documents
- RPC
- Shared documents
- Distributed coordination
- Agents