CHAPTER 1
Fundamentals of Distributed System

- INTRODUCTION
- DISTRIBUTED COMPUTING MODELS
- SOFTWARE CONCEPTS
- ISSUES IN DESIGNING DISTRIBUTED SYSTEM
- CLIENT – SERVER MODEL

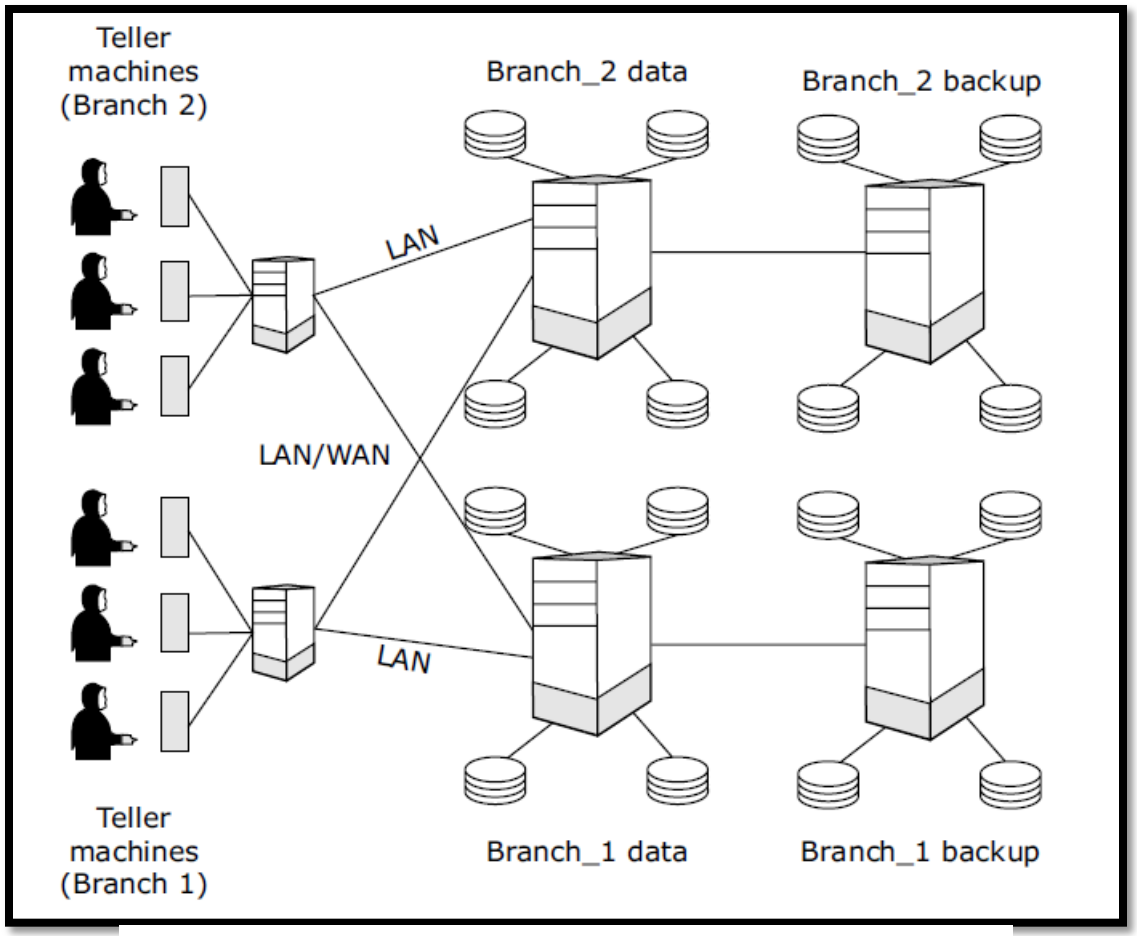
What is a Distributed System?

Tanenbaum's definition of a distributed system:

"A distributed system is a collection of independent computers that appear to the users of the system as a single coherent system."

An Example of a Distributed System

- Nationalized Bank with multiple Branch Offices



Internet connected network representing a bank

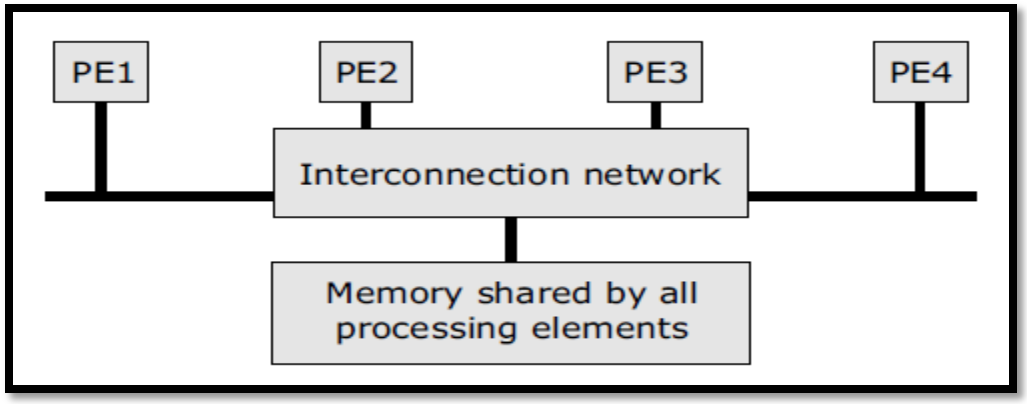
Requirements of Distributed Systems

- Security and reliability
- Consistency of replicated data
- Concurrent transactions (operations which involve accounts in different banks; simultaneous access from several users, etc.)
- Fault tolerance

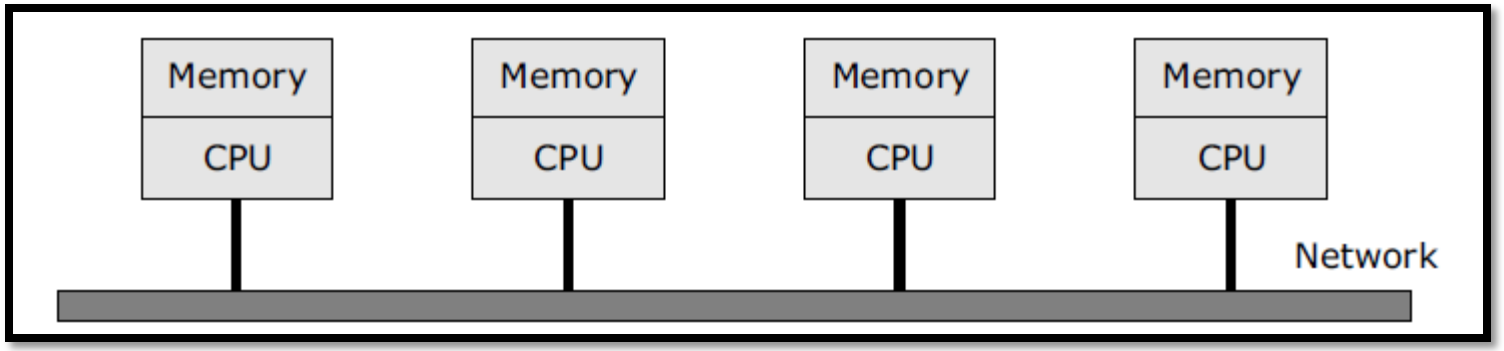
Architectures for Distributed Systems

- Shared memory architectures / Tightly coupled systems
  - ➢easier to program
- Distributed memory architectures / Loosely coupled systems
  - ➢offer a superior price performance ratio and are scalable

Architectures for Distributed Systems



Shared memory architecture



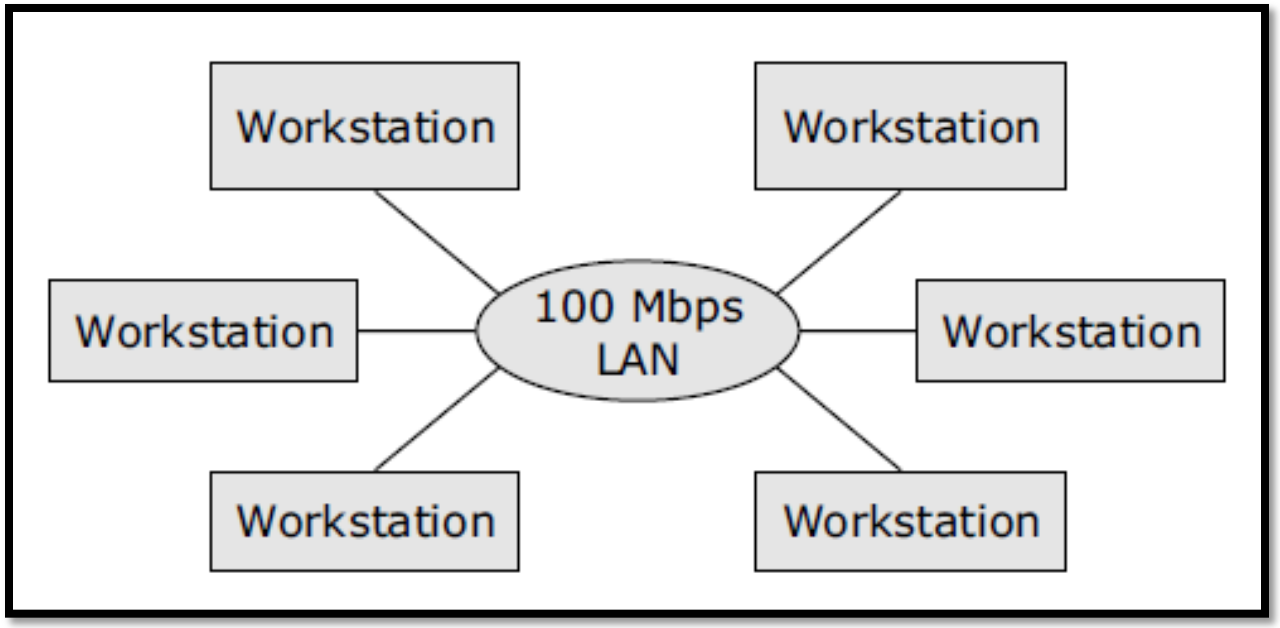
Distributed memory architecture

Distributed Computing Models

- Workstation model
- Workstation–server model
- Processor-pool model

Workstation Model

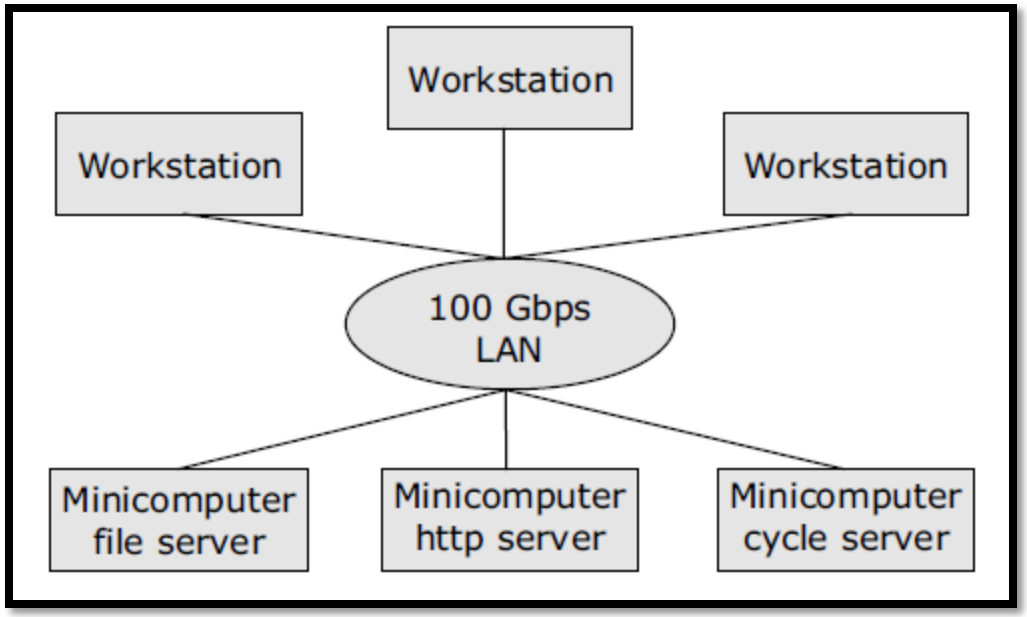
- Consists of network of personal computers
- Each one with its own hard disk and local file system
- Interconnected over the network



Workstation model

Workstation-server Model

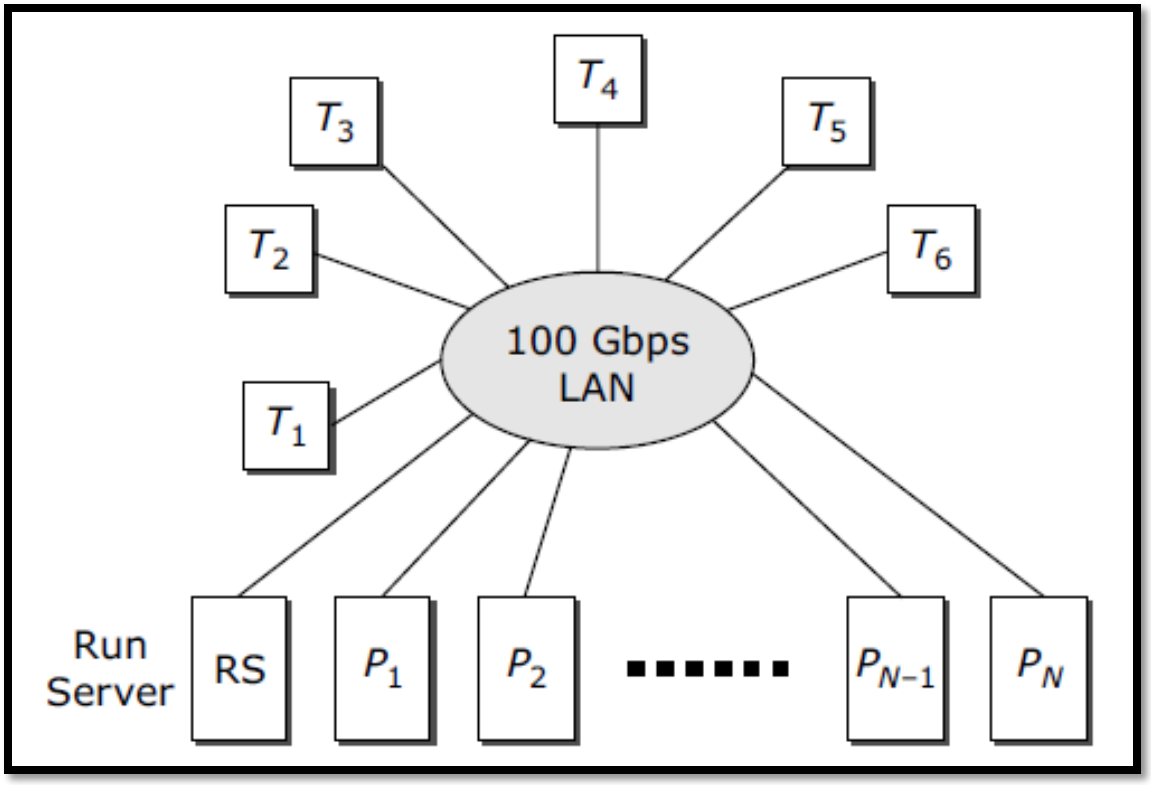
- Consists of multiple workstations coupled with powerful servers with extra hardware to store the file systems and other software like databases



Workstation–server model

Processor-pool Model

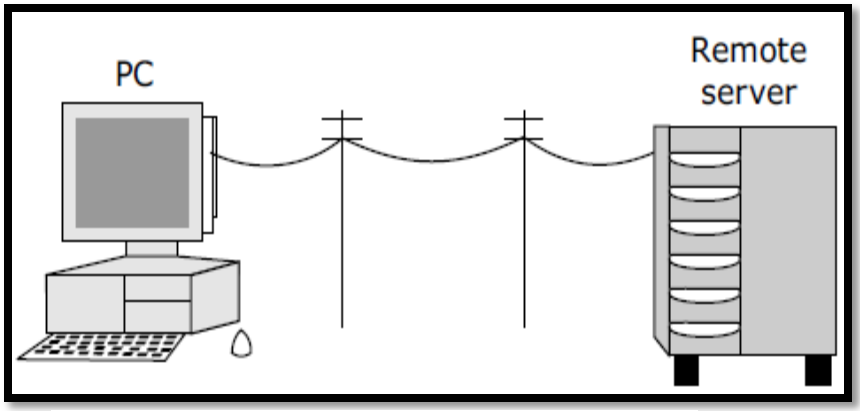
- Consists of multiple processors: a pool of processors and a group of workstations



Processor-pool model

Advantages of Distributed Systems

- Inherently distributed applications
- Information sharing among geographically distributed users
- Resource Sharing
- Better price performance ratio
- Shorter response time & higher throughput
- Higher reliability and availability against component failures
- Extensibility and Incremental Growth
- Better Flexibility



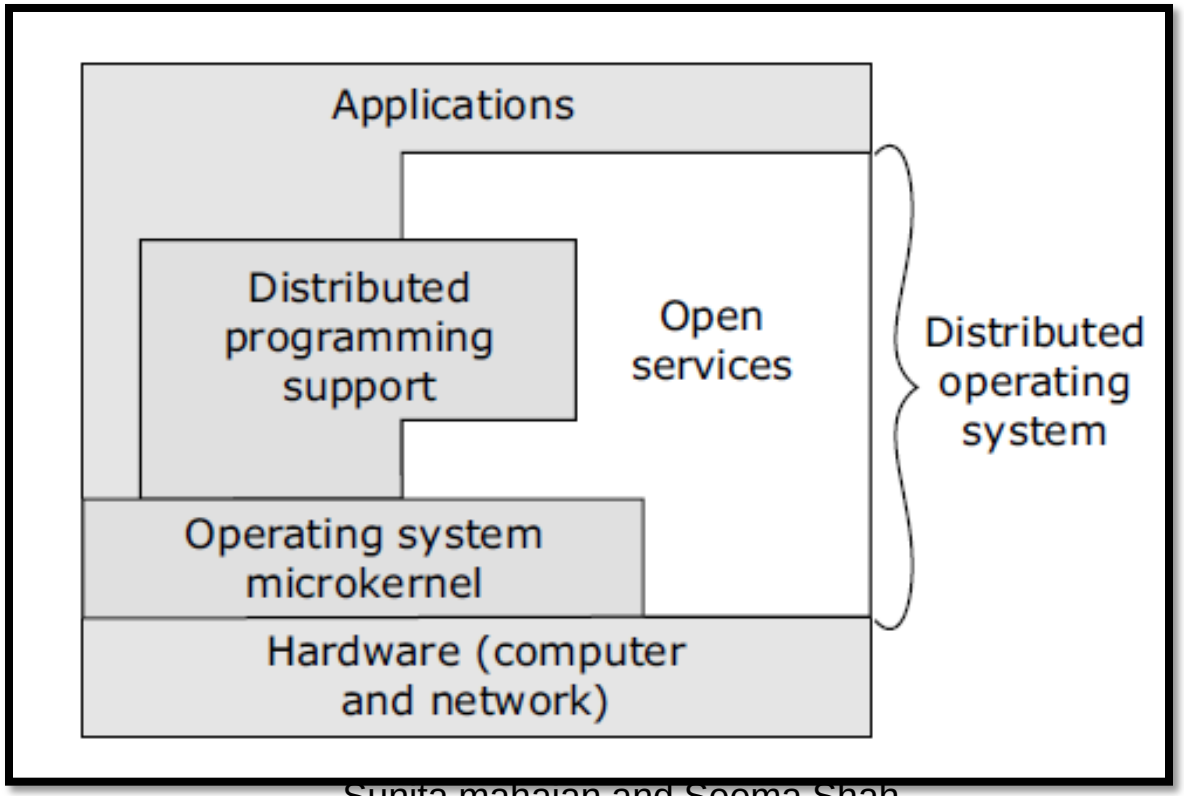
A PC connected to a remote server

Disadvantages of Distributed Systems

- Relevant software does not exist currently
- Security poses a problem due to easy access to all data
- Networking saturation may cause a hurdle in data transfer

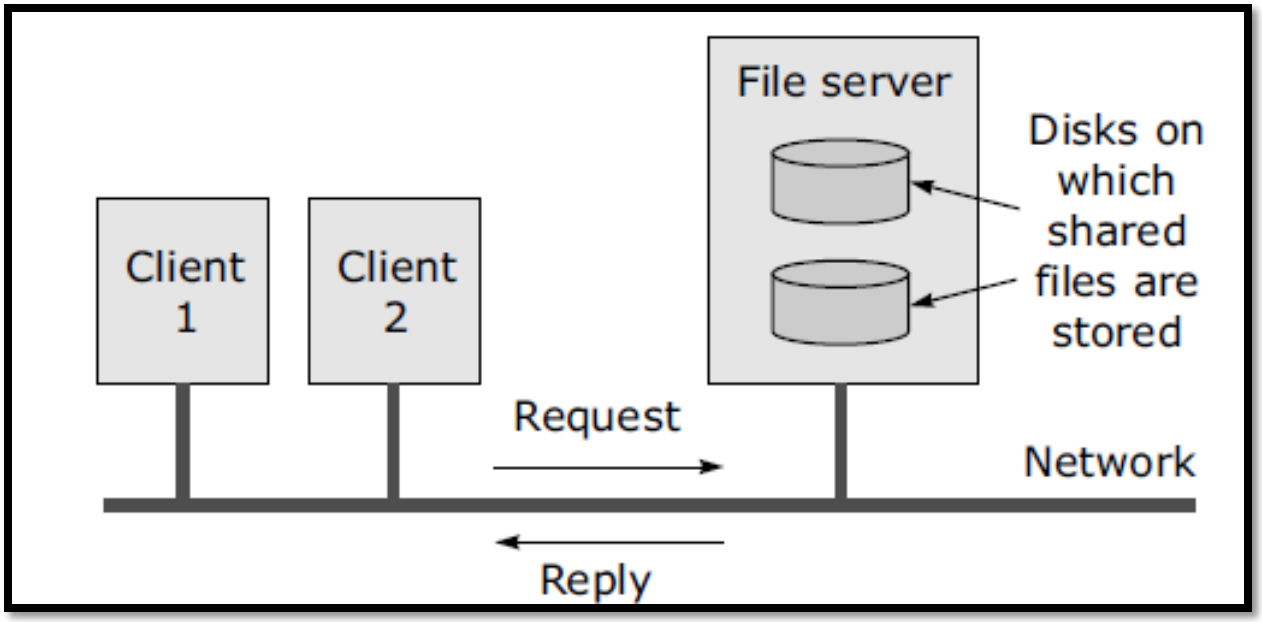
Software Concepts

- Network Operating System (NOS)
- Distributed Operating System (DOS)
- Multiprocessor Time Sharing System



Network Operating System (NOS)

- Build using a distributed system from a network of workstations connected by high speed network.
- Each workstation is an independent computer with its own operating system, memory and other resources like hard disks, file system and databases



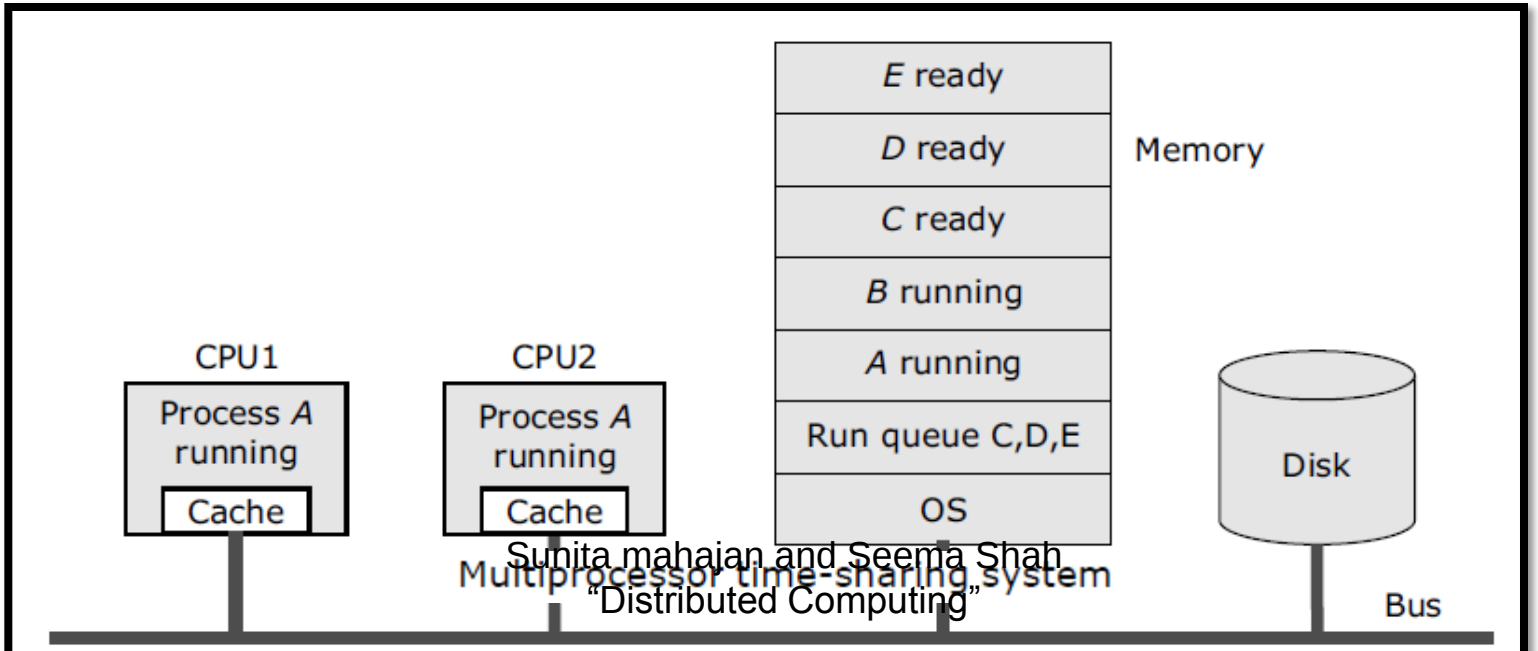
Network operating system

Distributed Operating System (DOS)

- Enables a distributed system to behave like a virtual uniprocessor even though the system operates on a collection of machines.
- Characteristics
  - ✓enabling Inter process communication
  - ✓Uniform process management mechanism
  - ✓Uniform and visible file system
  - ✓Identical kernel implementation
  - ✓Local control of machines
  - ✓handling scheduling issues

Multiprocessor Time Sharing System

- Combination of tightly coupled software and tightly coupled hardware with multiple CPUs projecting a uniprocessor image.
- Tasks are queued in shared memory and are scheduled to be executed in time shared mode on available processors.



Multiprocessor time-sharing system

Comparison of Different Operating Systems

Software Concepts

Criteria	Network OS	Distributed OS	Multiprocessor time-sharing OS
Projects a virtual uniprocessor image	No	Yes	Yes
Runs same operating system	No	Yes	Yes
Copies of operating system	*N*	*N*	1
Access to files	Sharing	Messages in memory	Sharing
Network protocols required	Yes	Yes	No
Single run queue	No	No	Yes
Well-defined file sharing	Usually no	Yes	Yes

Issues in Designing Distributed Systems

- Transparency
- Flexibility
- Reliability
- Performance
- Scalability
- Security

Transparency

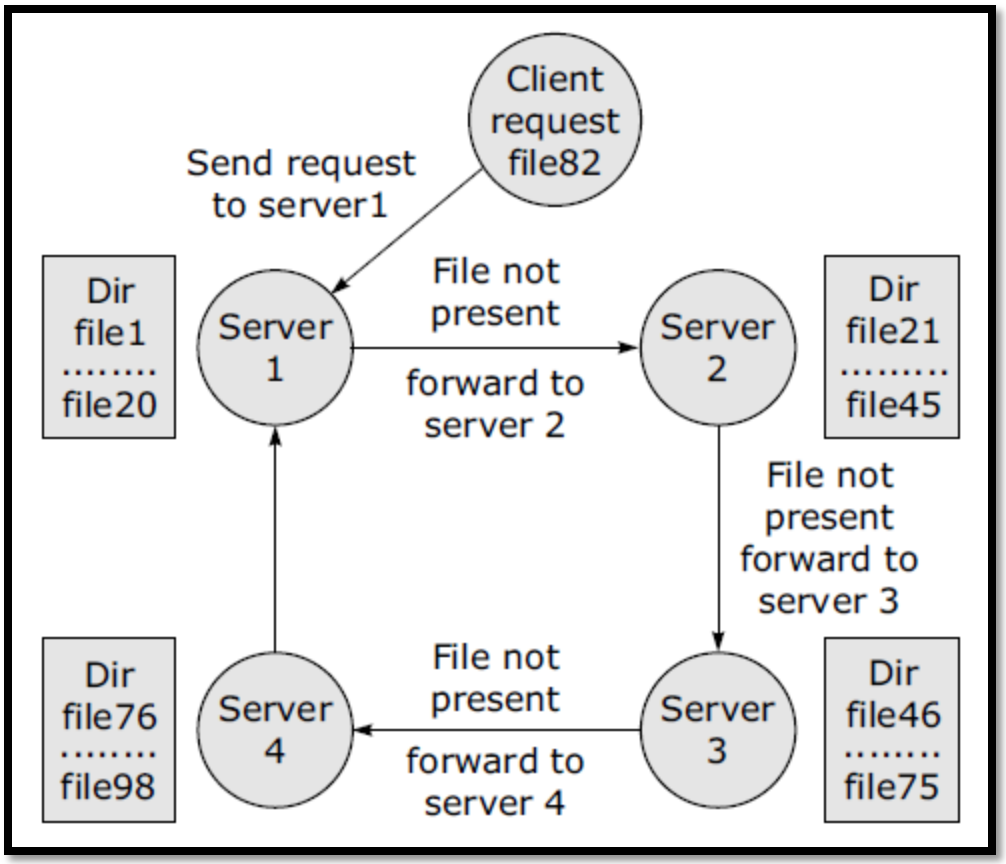
Transparencies required for Distributed Systems

Types of transparencies

Transparency	Description
Access	Hide the differences in data representation and how a resource is accessed
Location	Hide where a resource is physically located
Migration	Hide the movement of a resource to another location
Relocation	Hide the movement of a resource to another location while in use
Replication	Hide the fact that multiple copies of the resource exist without user's knowledge
Concurrency	Hide the fact that a resource may be shared by several users
Failure	Hide the failure and recovery of a resource
Persistence	Hide whether a resource is in memory or on disk

Replication Transparency

Locating Replicated File stored on any server



Location transparency

Flexibility

- Monolithic kernel approach
- Here kernel does all functions and provides facilities at local machine
- There is over Burdon on kernel

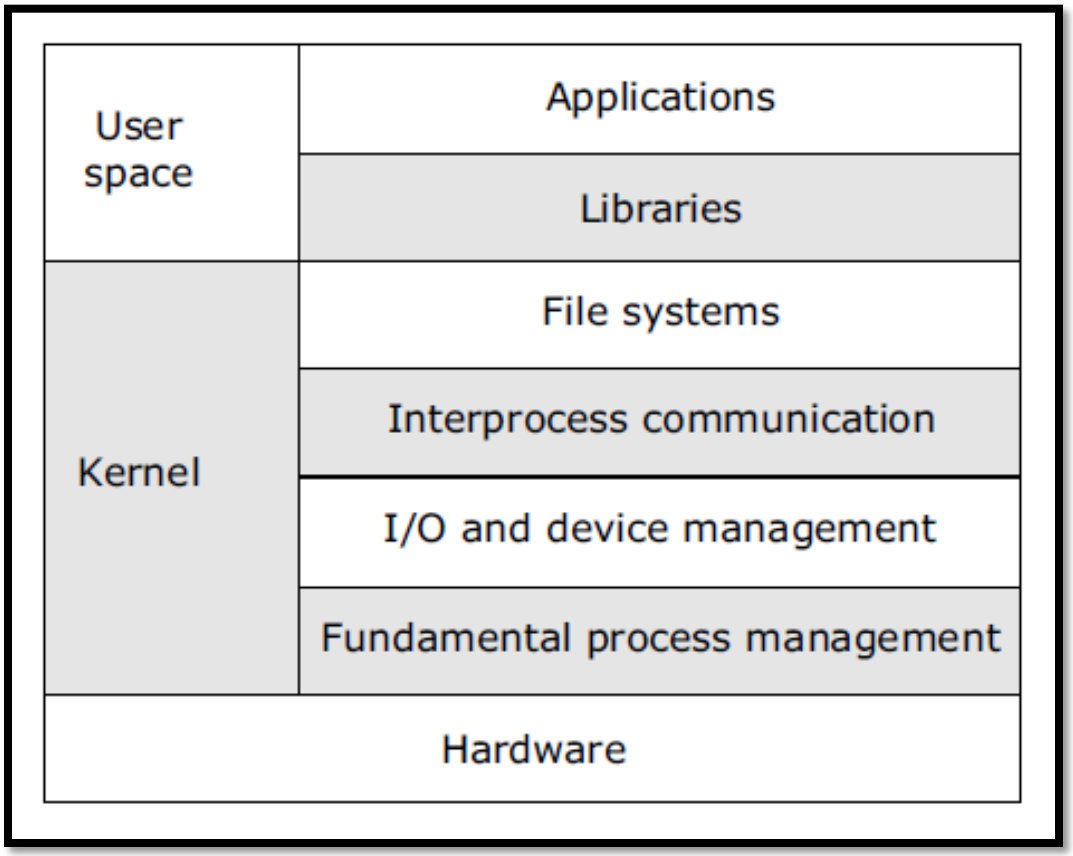
- Microkernel approach
- It takeout as much functionality as possible from kernel And retain only essential functions.
- Provides only few functions in the kernel while uses process server to manage IPC,pager fro MM and fs management.
- Its easy to port maintain and extend

flexiblity

- In initial stages there may be need for modification of platform
- The best way to achieve flexibility is to take a decision where to use monolithic or microkernel on each machine.
- Kernel is the central controller which provides basic system facilities.

Monolithic Kernel Approach

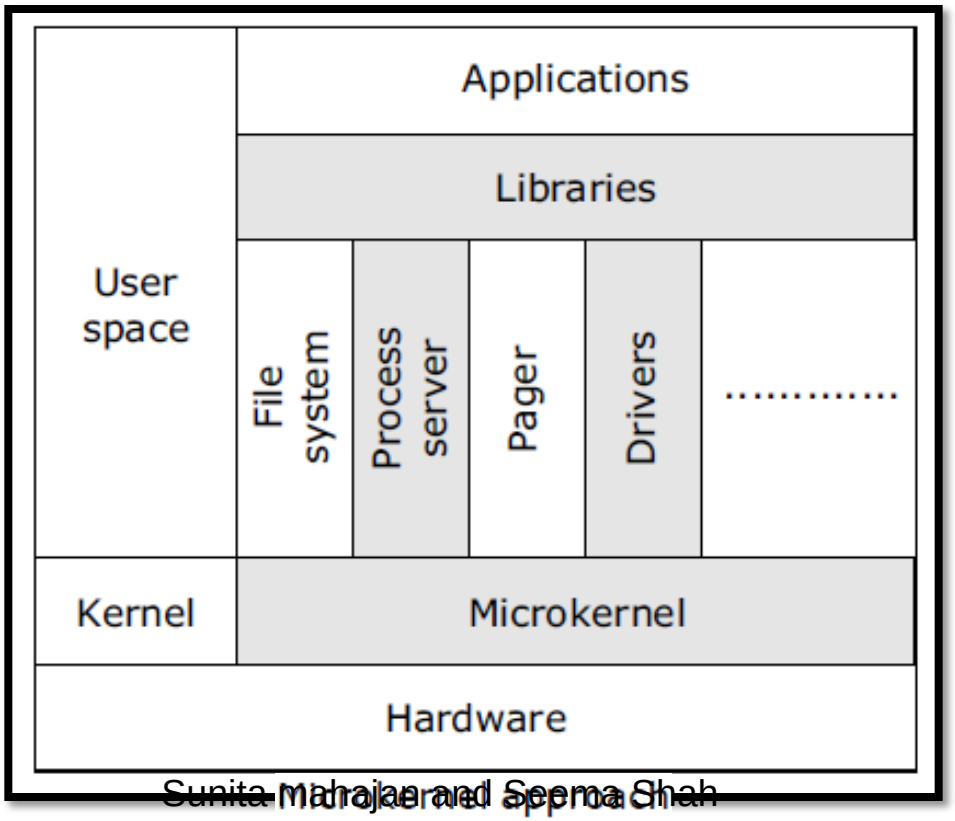
- Uses the minimalist, modular approach with accessibility to other services as needed



Monolithic kernel approach

Microkernel Approach

- Uses the kernel does it all approach with all functionalities provided by the kernel irrespective whether all machines use it or not



Microkernel approach

Monolithic versus Microkernel Approach

Monolithic vs microkernel

Services (file, network)

Kernel code and data

Reliability

- Availability in case of Hardware failure
- Data recovery in case of Data failure
- Maintain consistency in case of replicated data

Performance

Metrics are

- Response time
- Throughput
- System utilization
- Amount of network capacity used

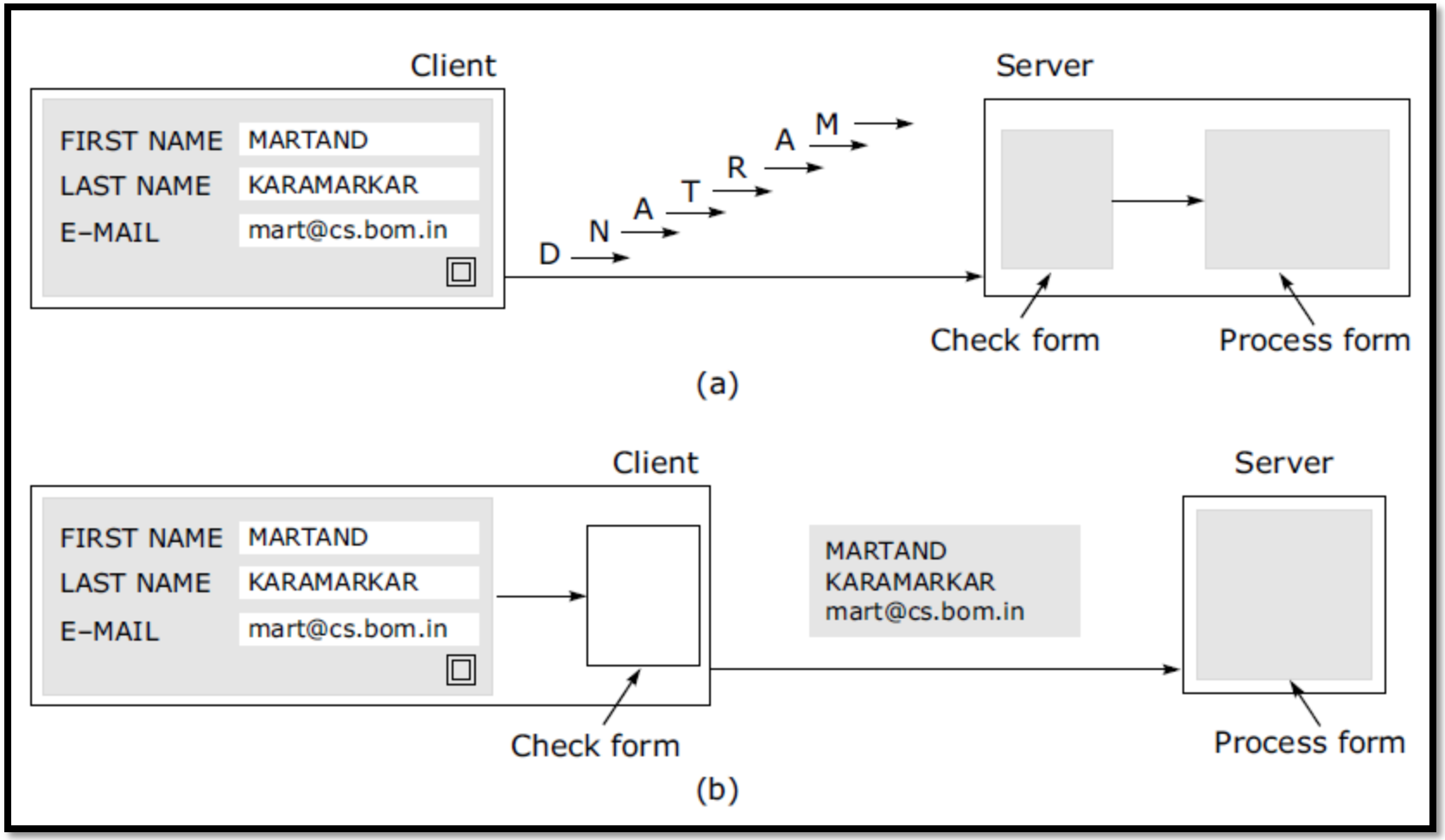
Scalability

- Techniques to handle scalability issues
  - hide communication latencies
  - hide distribution
  - hide replication

Scalability-related issues

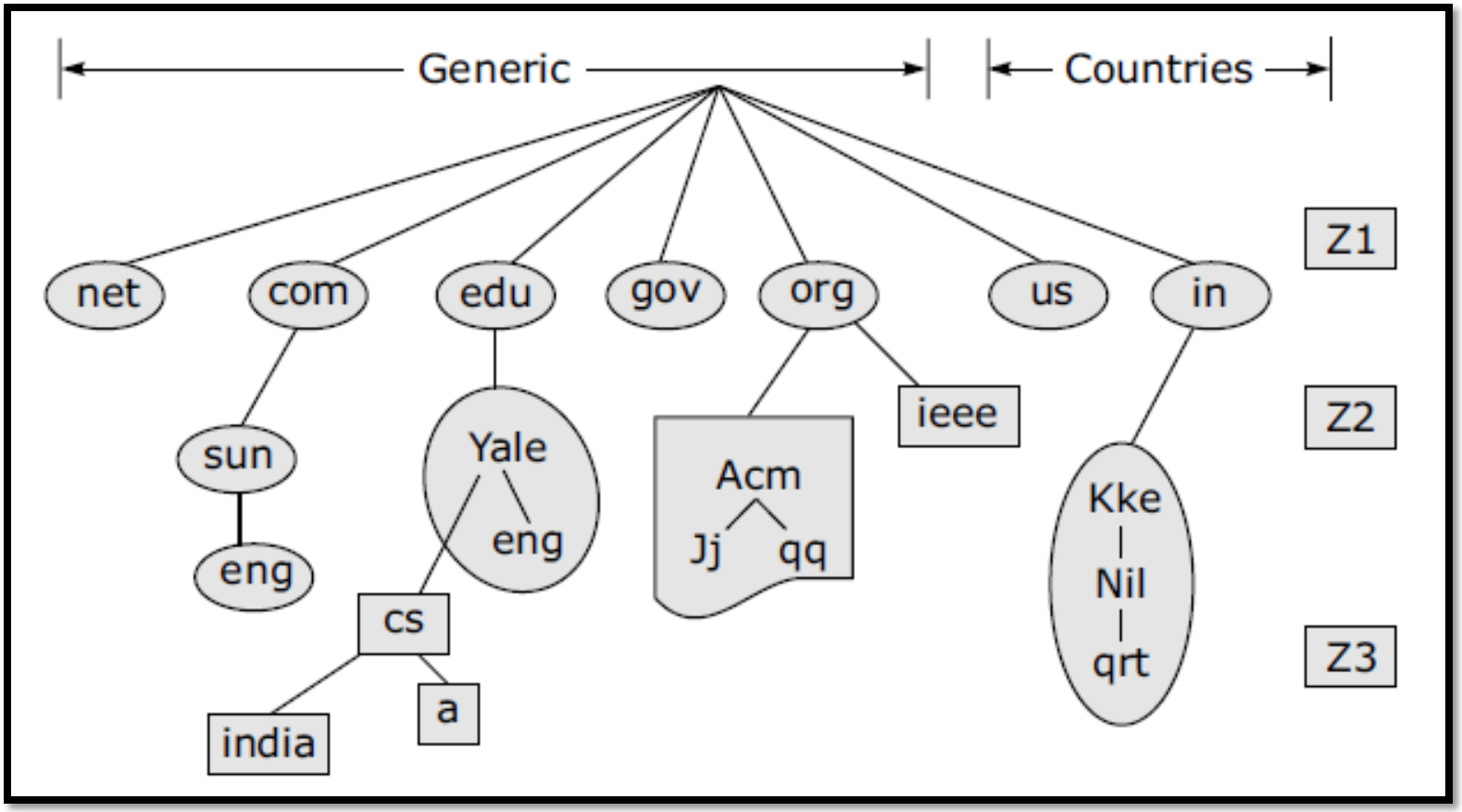
Concept	Example
Centralized services	A single server for all users
Centralized data	A single on-line telephone book
Centralized algorithms	A single algorithm doing routing based on available information

Hide Communication Latencies



Hide communication latencies

Hide Distribution

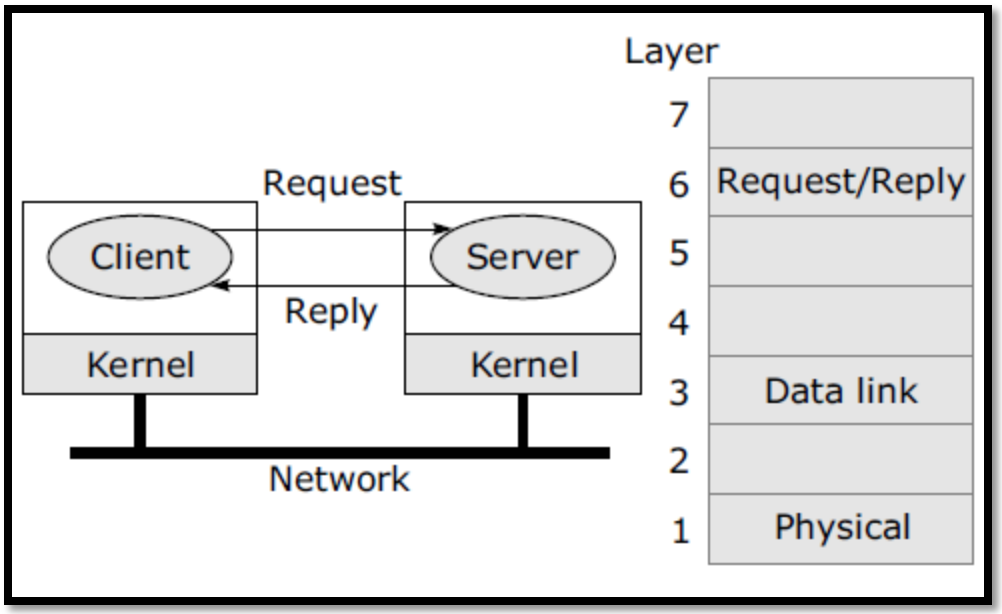


Internet DNS

Security

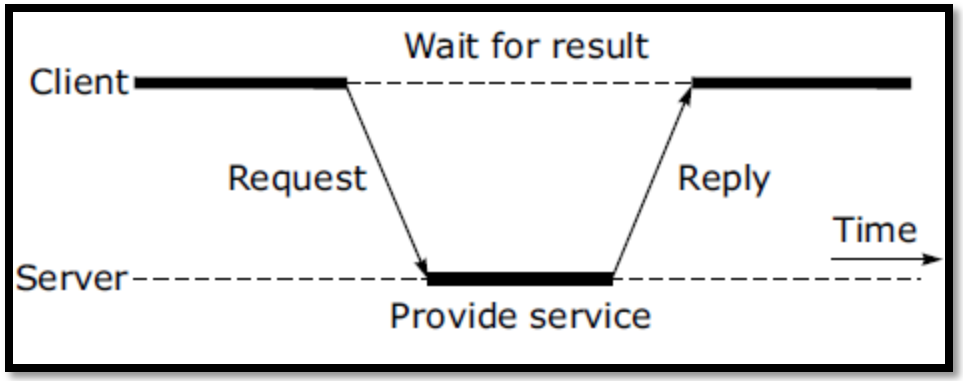
- Confidentiality means protection against unauthorized access
- Integrity implies protection of data against corruption
- Availability means protection against failure always accessible

Client-Server Model



The client–server model

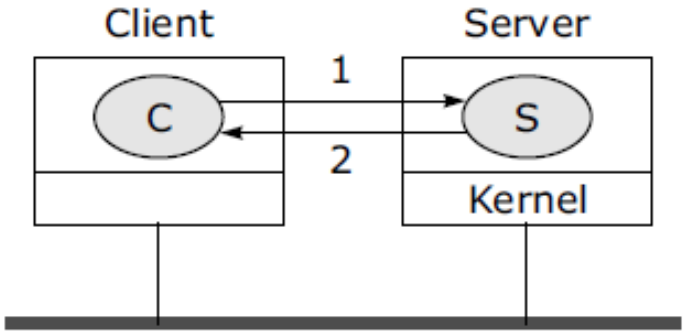
The client–server interaction



Client-Server Addressing Techniques

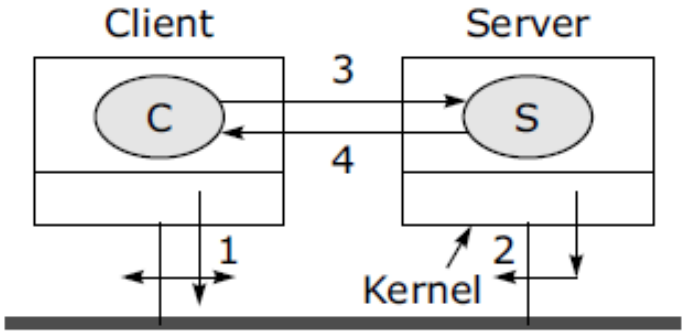
- Machine addressing
- Process addressing
- Name server addressing

Client-Server Addressing Techniques



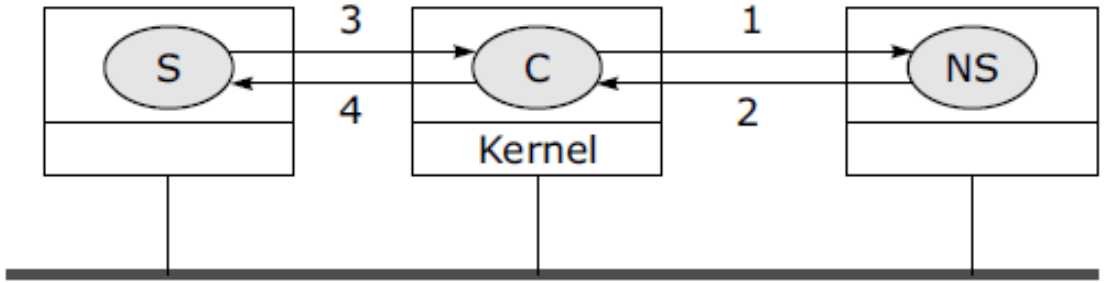
1: Request to client
2: Reply to server

(a) Machine addressing



1: Broadcast 2: Give own location
3: Request 4: Reply

(b) Process addressing



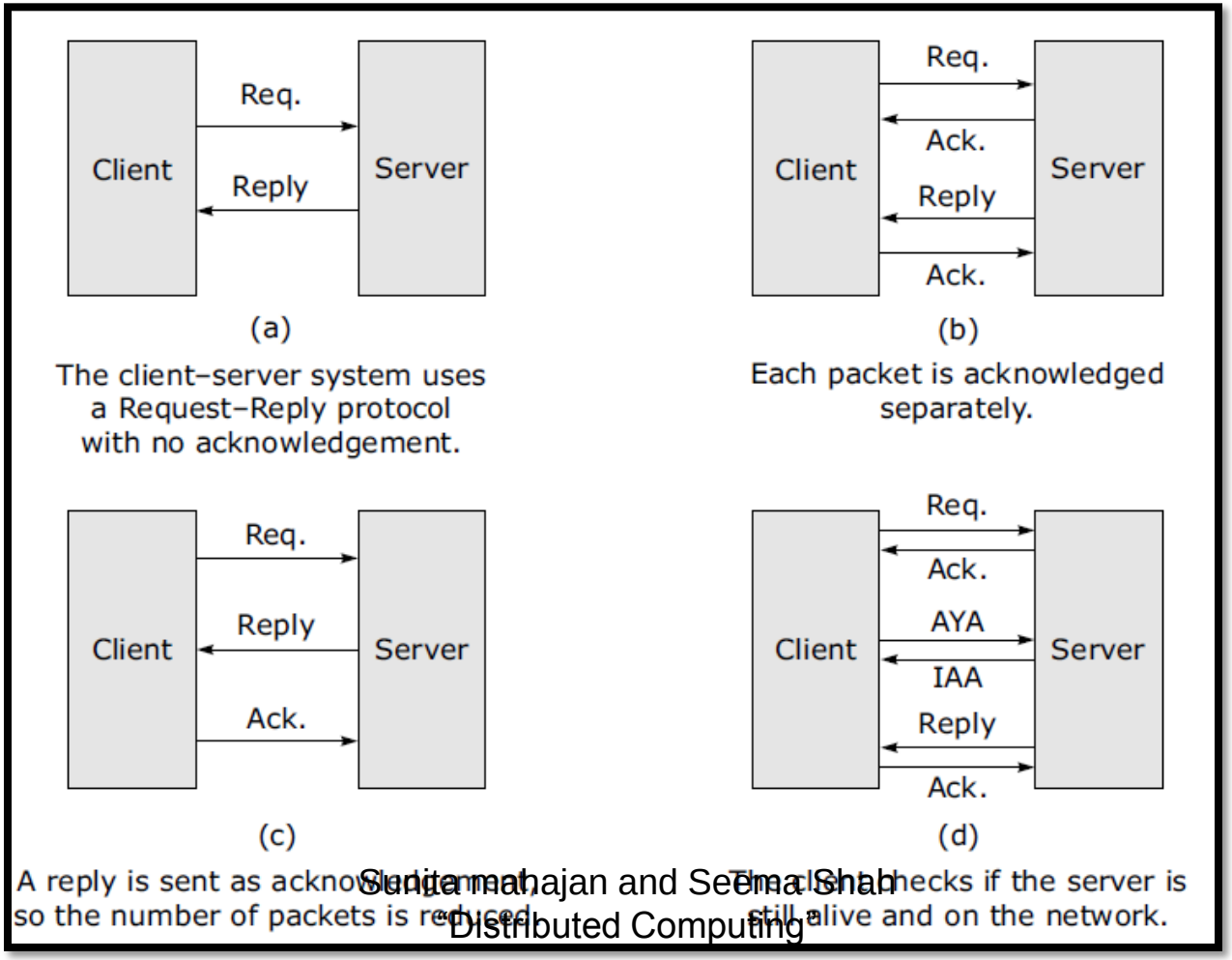
1: Lookup in name server 2: Reply from NS
3: Request 4: Reply

(c) Name server technique

Client-Server Implementation

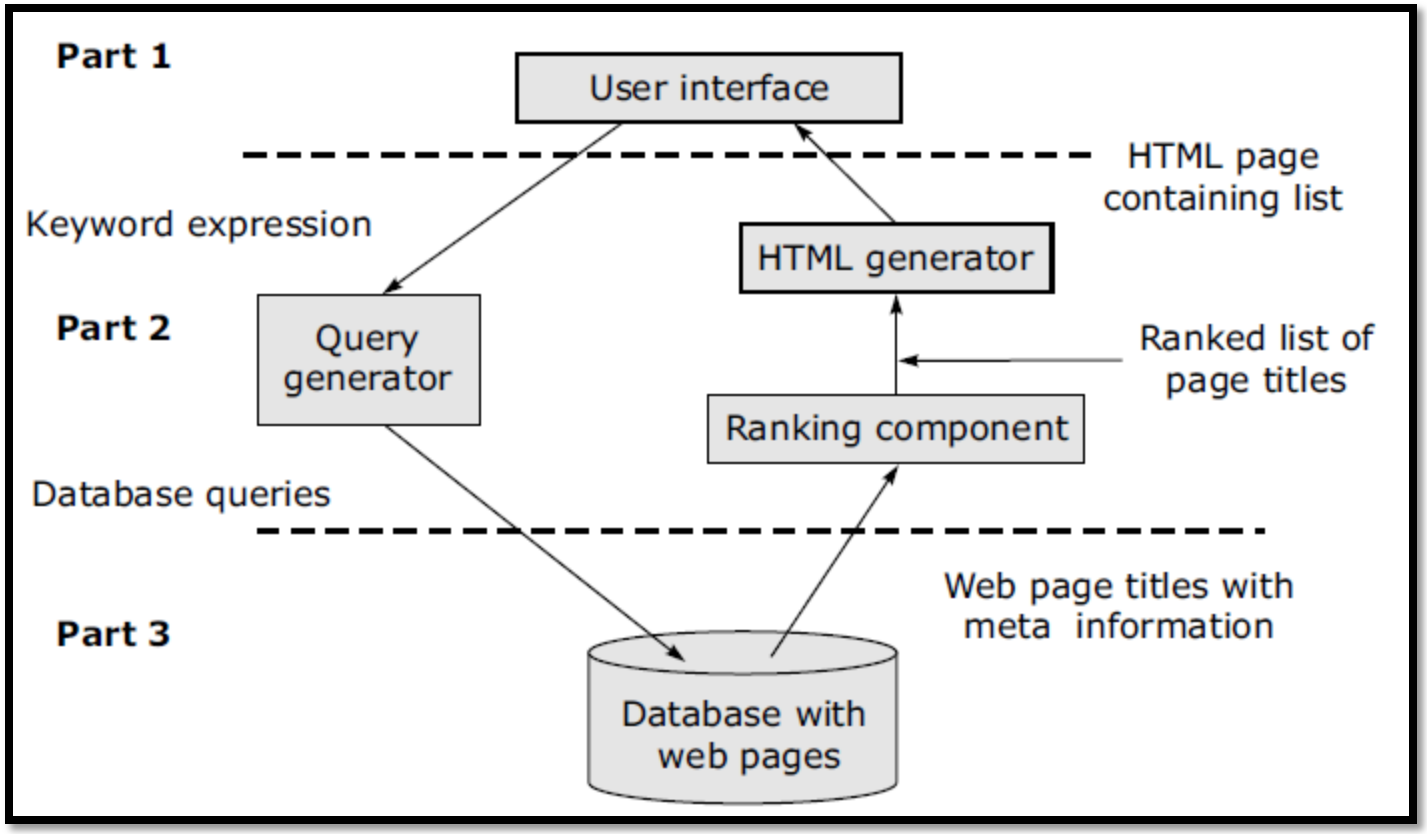
Messages for client server interaction

- Request, Reply, Acknowledge, Are you Alive, I am Alive



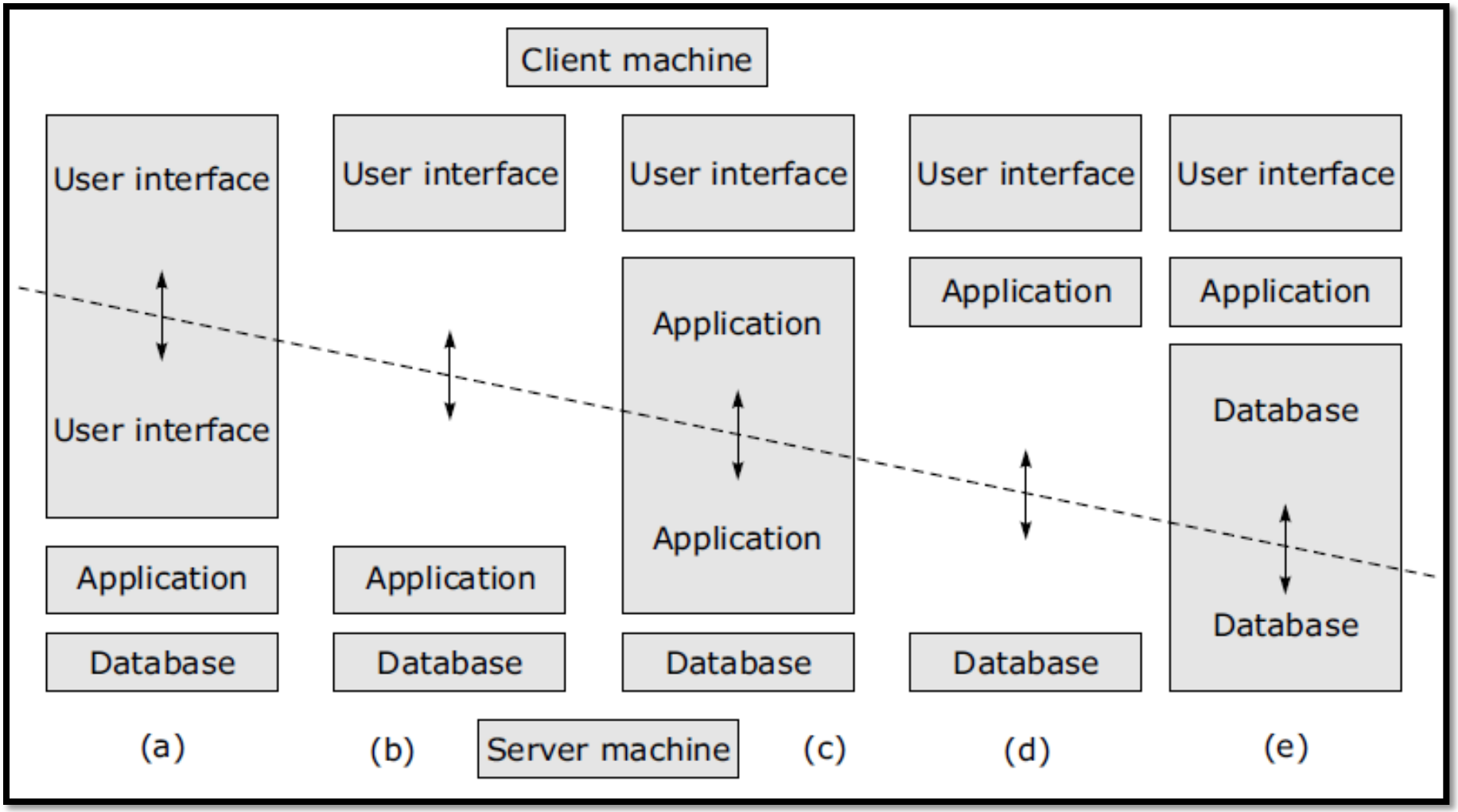
Differentiation between the Client and the Server

- User interface level
- Processing level
- Data level



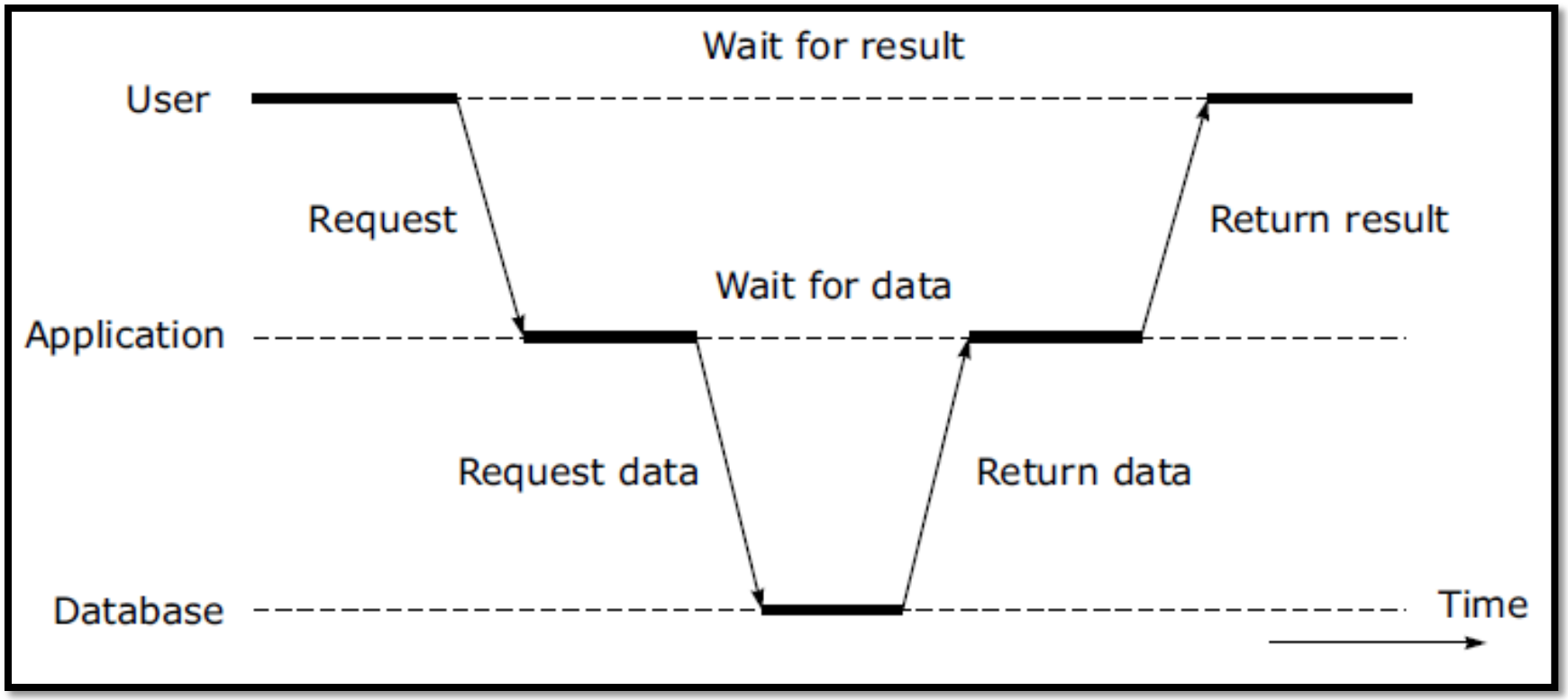
Three-tiered Internet search engine

Client-Server Architecture



Multi-tiered client–server architecture

Client-Server Architecture



Server acts as a client

Thank you

PROCEDURAL PROGRAMMING

Procedural programming is by far the most common form of programming. A program is a series of instructions which operate on variables. It is also known as imperative programming

- Examples of procedural programming languages include FORTRAN, ALGOL, Pascal, C, MODULA2, Ada, BASIC. Despite their differences they all share the common characteristics of procedural programming

Advantages of procedural programming include its relative simplicity, and ease of implementation of compilers and interpreters.

Disadvantages of procedural programming include the difficulties of reasoning about programs and to some degree difficulty of parallelization. Procedural programming tends to be relatively low level compared to some other paradigms, and as a result can be very much less productive.

OBJECT ORIENTED PROGRAMMING

Object oriented programming is characterized by the defining of classes of objects, and their properties. Inheritance of properties is one way of reducing the amount of programming, and provision of class libraries in the programming environment can also reduce the effort required.
The most widely used object oriented language is C++ which provides object extensions to C, but this is rapidly being overtaken by Java.

Features Of Object-Oriented Programming

Data Abstraction and Encapsulation

Operations on the data are considered to be part of the data type
We can understand and use a data type without knowing all of its implementation details Neither how the data is represented nor how the operations are implemented

We just need to know the interface (or method headers) – how to "communicate" with the object Compare to functional abstraction with methods

OBJECT ORIENTED PROGRAMMING

Inheritance

Properties of a data type can be passed down to a sub-type – we can build new types from old ones We can build class hierarchies with many levels of inheritance

Polymorphism

Operations used with a variable are based on the class of the object being accessed, not the class of the variable

Parent type and sub-type objects can be accessed in a consistent way

OOP's world		**Procedural Language world**
Class	|	structure
{	|	{
Objects	|	structure variables
}	|	}

Distributed System and Middleware Concepts

DISTRIBUTED SYSTEM

A distributed system is a collection of independent computers that appears to its users as a single coherent system

Important characteristics of distributed systems

- Differences between the various computers and the ways in which they communicate are hidden from users

- Users and applications can interact with a distributed system in a consistent and uniform way, regardless of where and when interaction takes place

Goals of Distributed System

4 important goals that should be met to make building a distributed system worth the effort they are

1) Easily connect Users to resources, hide the fact that resources are distributed across a network, open, scalable

2) Transparency

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. Which hides whether a Implementation i.e. software resource is in main memory or disk

Forms of Transparency in a Distributed System

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 may be shared by several competitive users
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource
Persistence	Hide whether a (software) resource is in memory or on disk

3) Openness

An open distributed system is a system that offers services according to standard rules.

4) Scalability

Scalability of a system can be measured along at Least three different dimensions

-First, a system can be scalable with respect to its size, we can easily add more users and resources to the system.

-Second, a geographically scalable system users and resources may lie far apart.

-Third, a system can be administratively scalable; it can still be easy to manage even if it spans many independent administrative organizations.

Reference

Sunita Mahajan and Seema Shah

“Distributed Computing”

- ANY QUESTION

T H A N K

Y O U