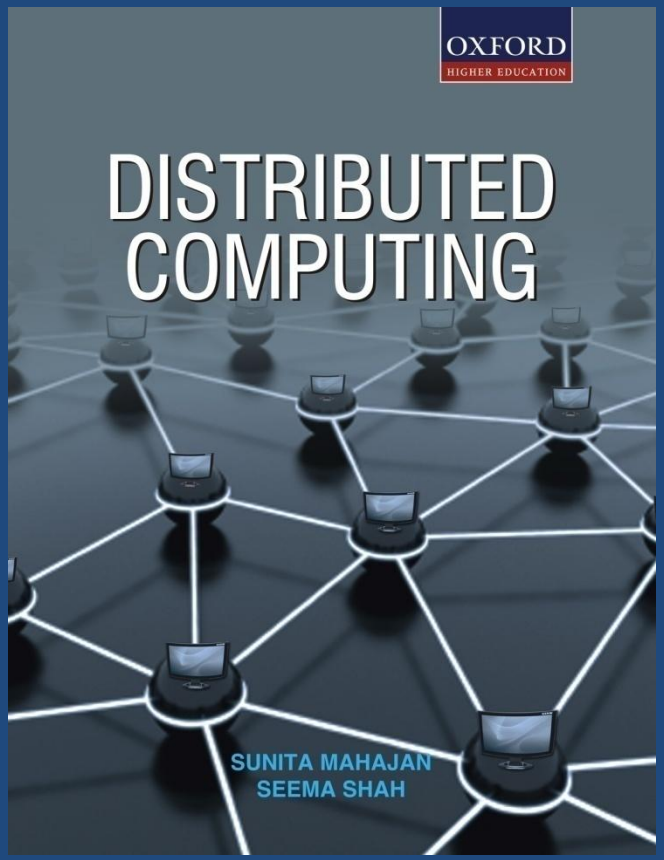

DISTRIBUTED COMPUTING

By
Sunita Mahajan & Seema Shah

Presented By
Prof. S.J. Soni,
Asst. Professor, CE Dept.,
SPCE, Visnagar

CHAPTER-1

BASIC DISTRIBUTED SYSTEM CONCEPTS

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

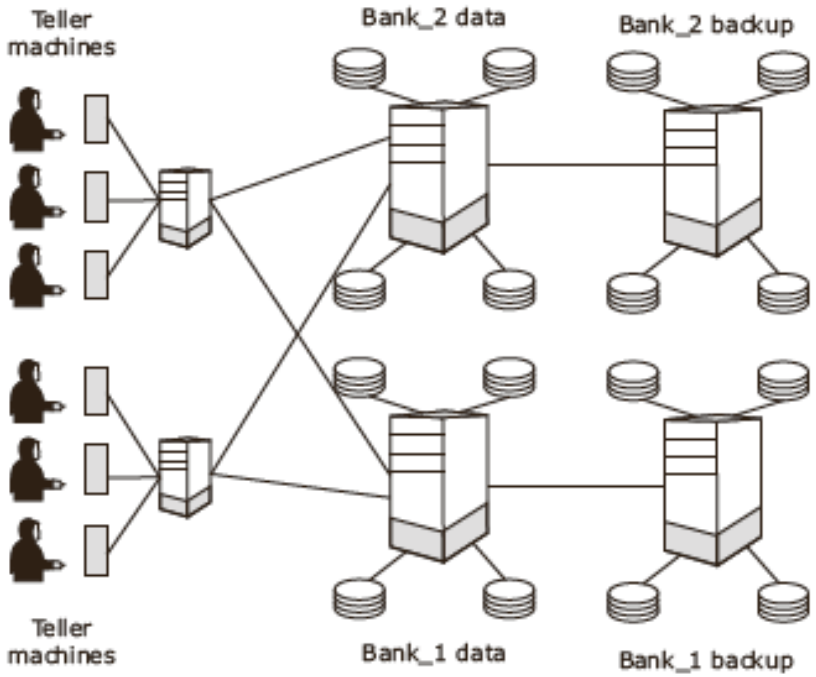

Figure 1-1 Internet connected network representing a bank

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**

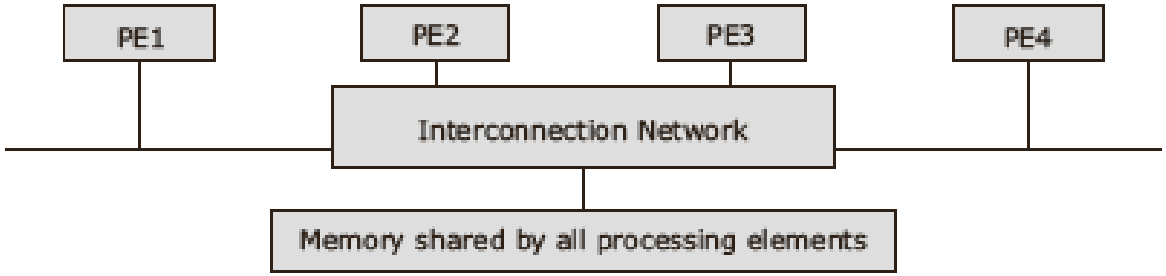

Figure 1-2 Shared memory architecture

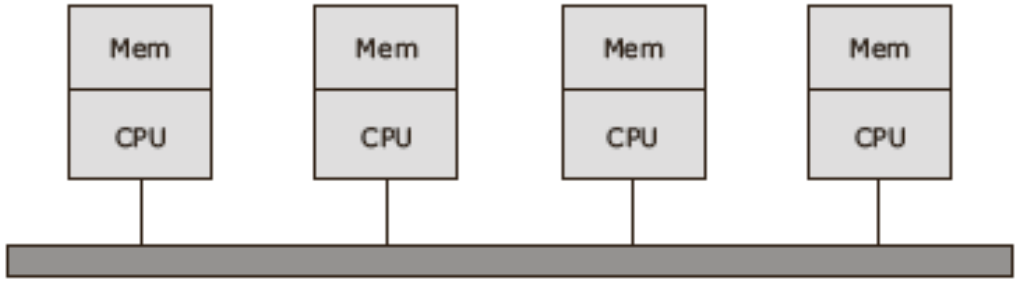

Figure 1-3 Distributed memory architecture

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

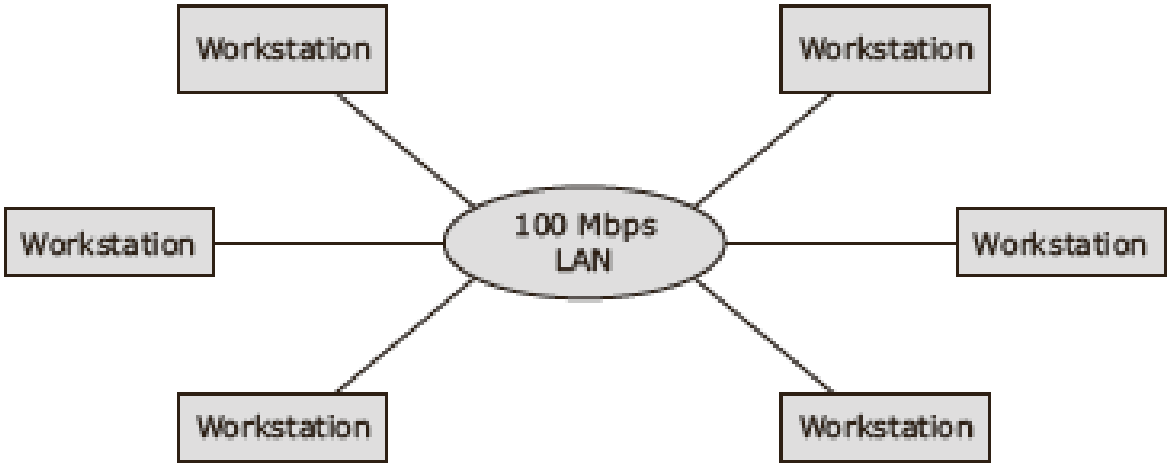

Figure 1-4 Workstation model

Workstation model

- This model is not very easy to implement, since several issues need to be resolved. They are:
  - How to find an idle workstation?
  - How to transfer a process from one workstation to another workstation in a transparent manner?
  - What happens to the remote process when a user logs on to that workstation and a home process is created?

workstation-server model

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

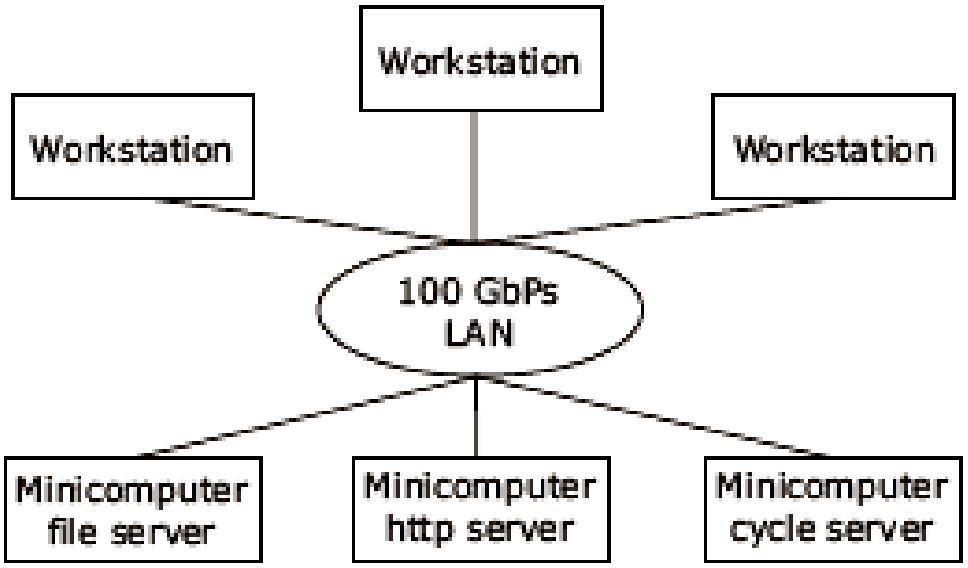

Figure 1-5 Workstation-server model

processor-pool model

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

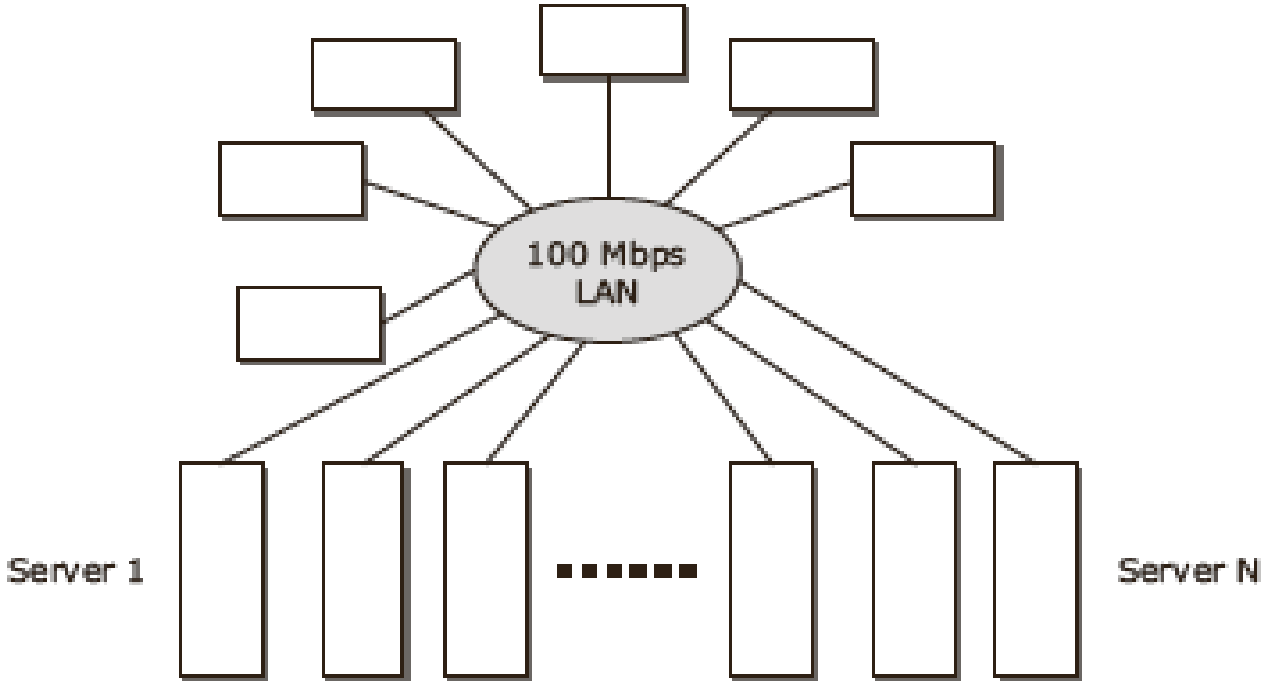

Figure 1-6 Processor pool model

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

Figure 1-7 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

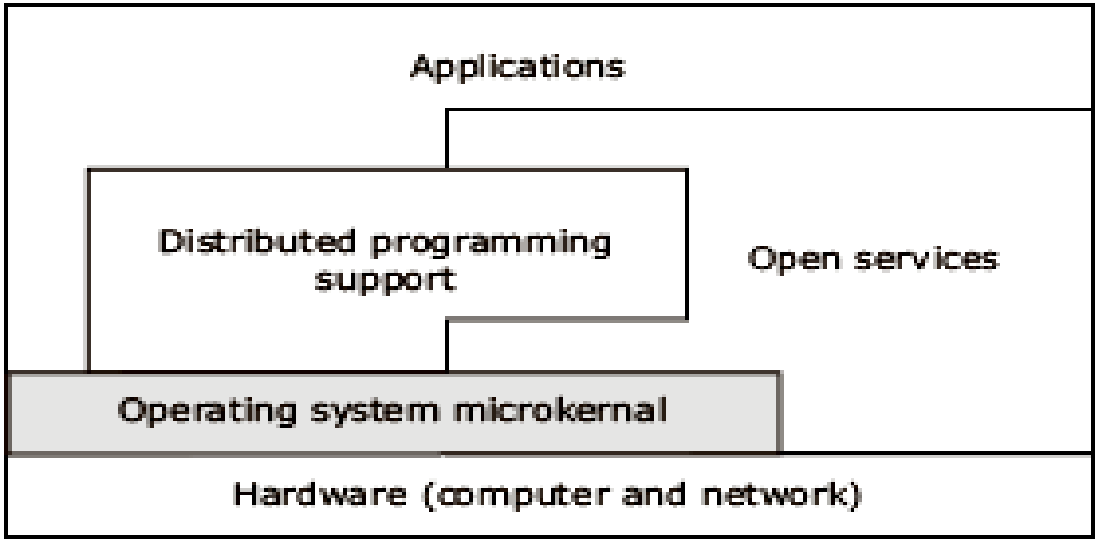

Figure 1-8 Software services

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

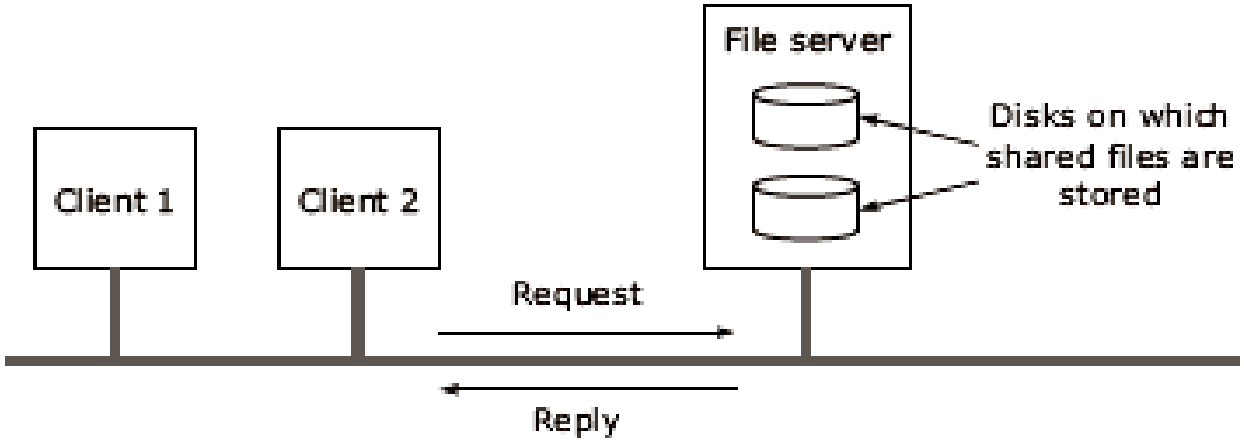

Figure 1-9 Network operating system

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.

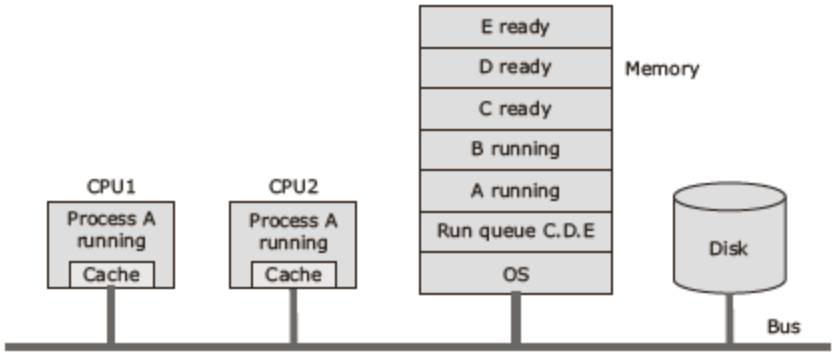

Figure 1-10 Multiprocessor time-sharing system

Comparison of different Operating systems

Software Concepts

Table1-1 Comparison of the three operating systems

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
Communication process of files	Sharing	Messages of 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

Table 1-2 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

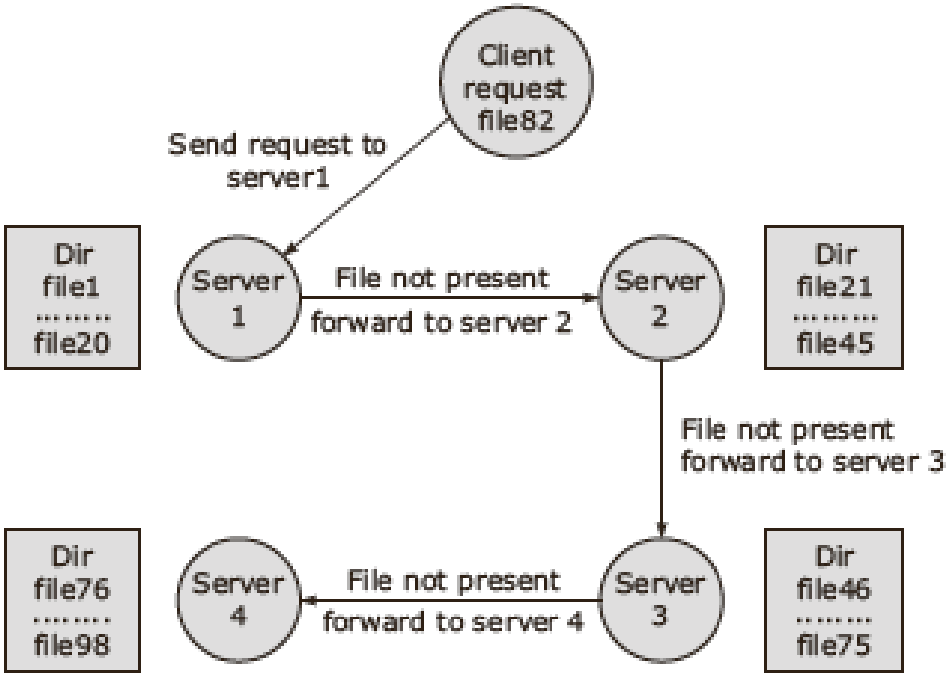

Figure 1-11 Replication transparency

Flexibility

- **Monolithic kernel approach**
- **Microkernel approach**

Monolithic kernel approach

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

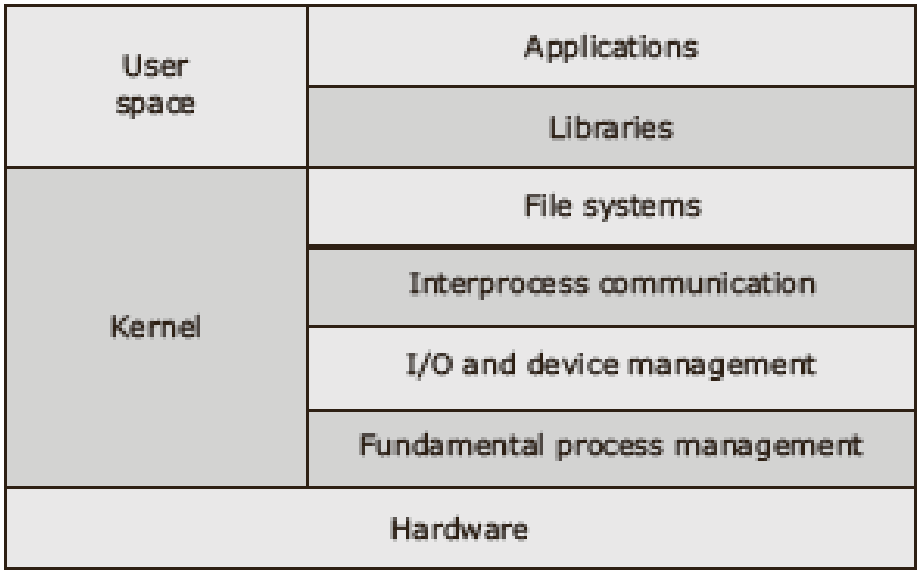

Figure 1-12 Monolithic kernel approach

Microkernel approach

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

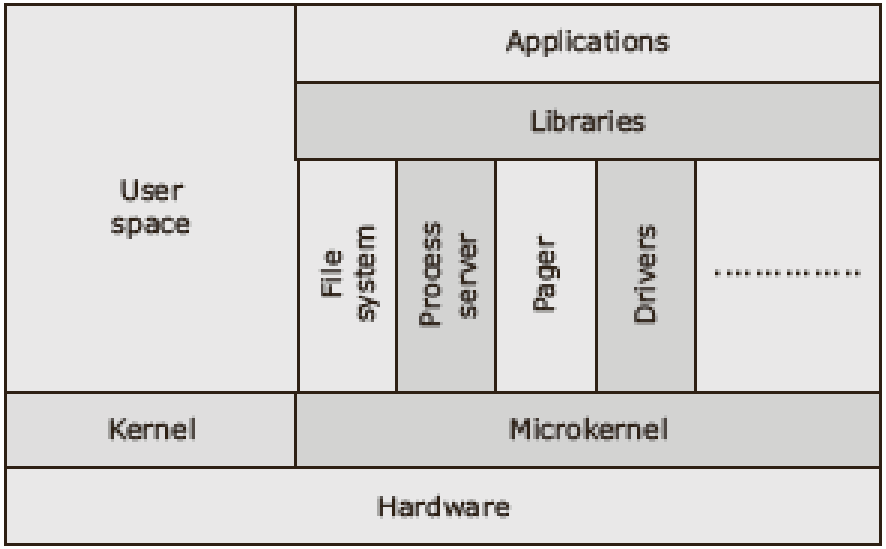

Figure 1-13 Microkernel approach

Monolithic versus Microkernel Approach

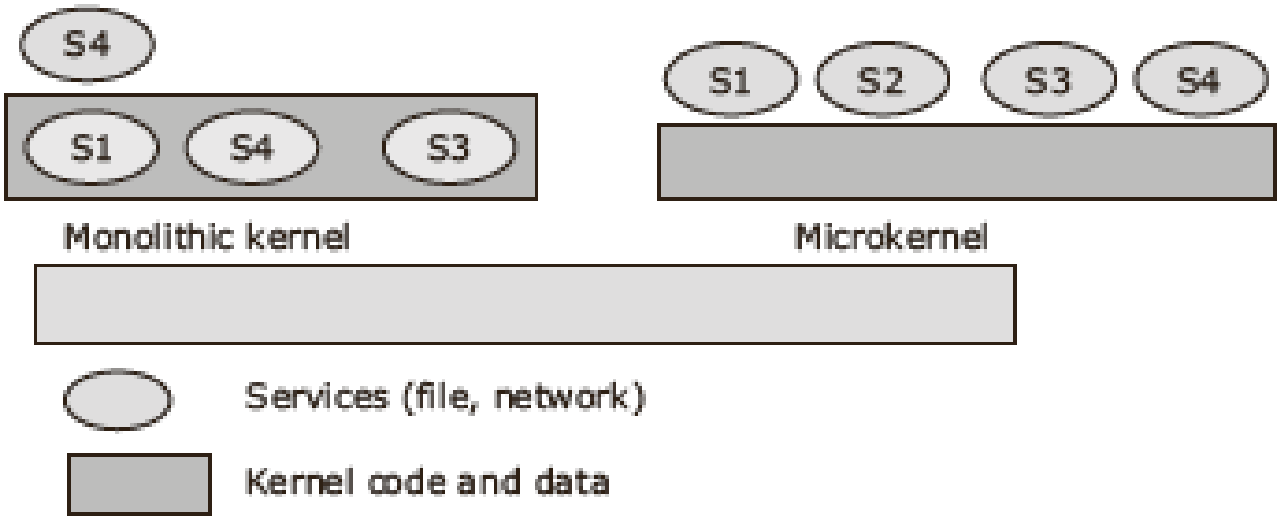

Figure 1-14 Monolithic vs. microkernel

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

Table 1-3 Scalability related issues

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

Hide communication latencies,

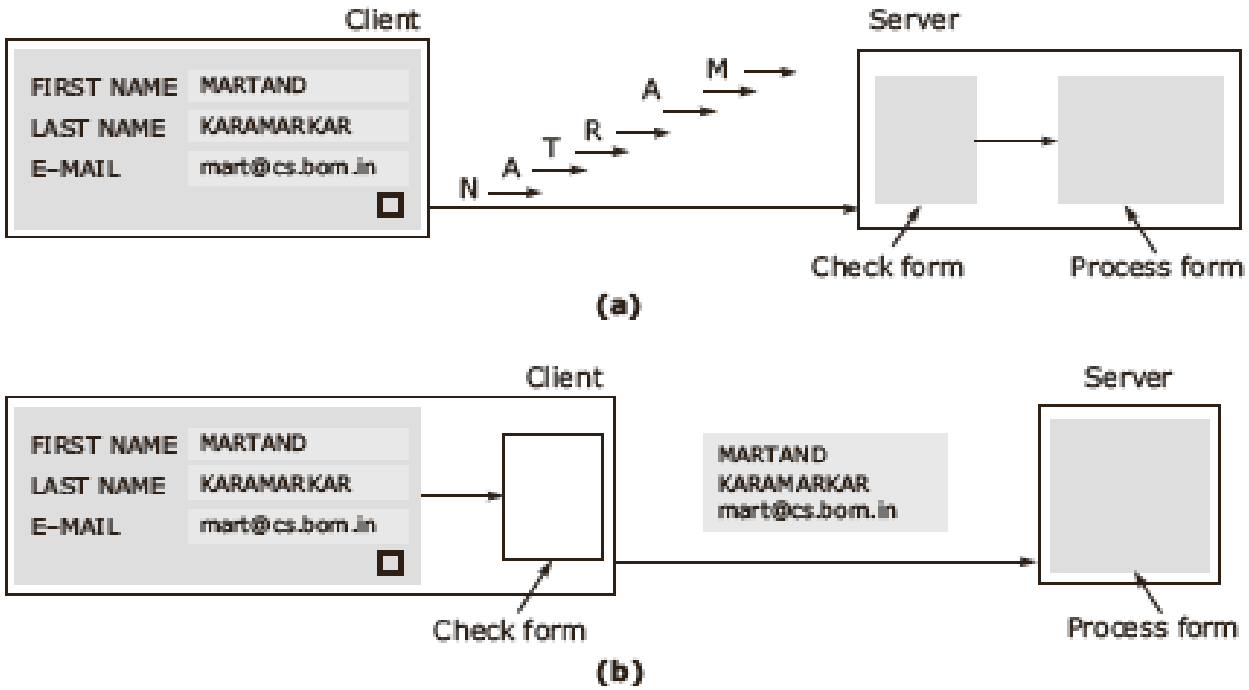

Figure 1-15 Hide communication latencies

Hide distribution

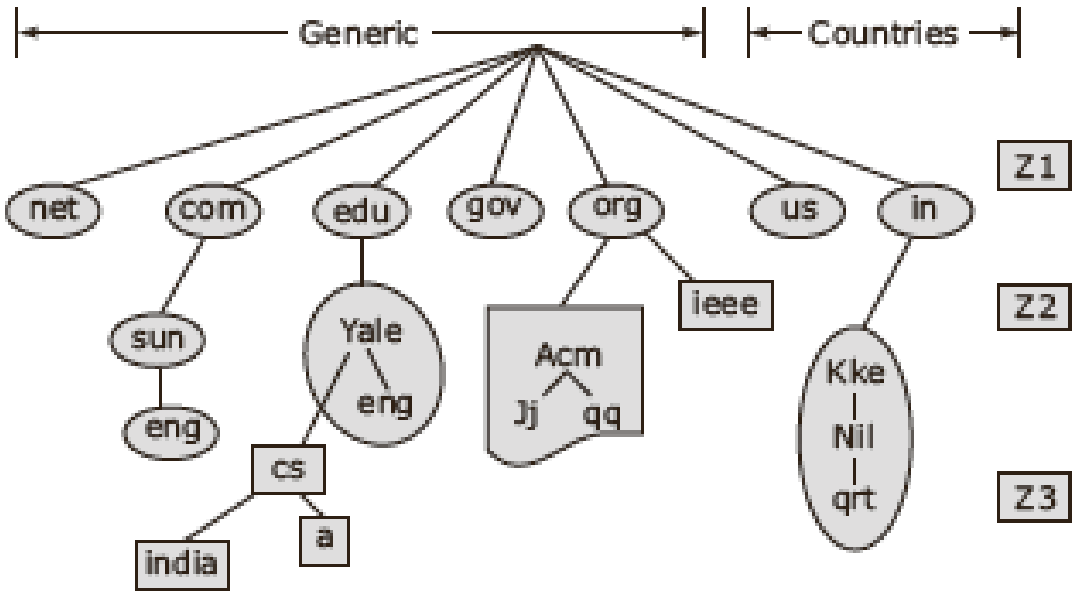

Figure 1-16 Internet DNS

Security

- confidentiality means protection against unauthorized access;
- integrity implies protection of data against corruption
- availability means protection against failure always accessible.

Client Server model

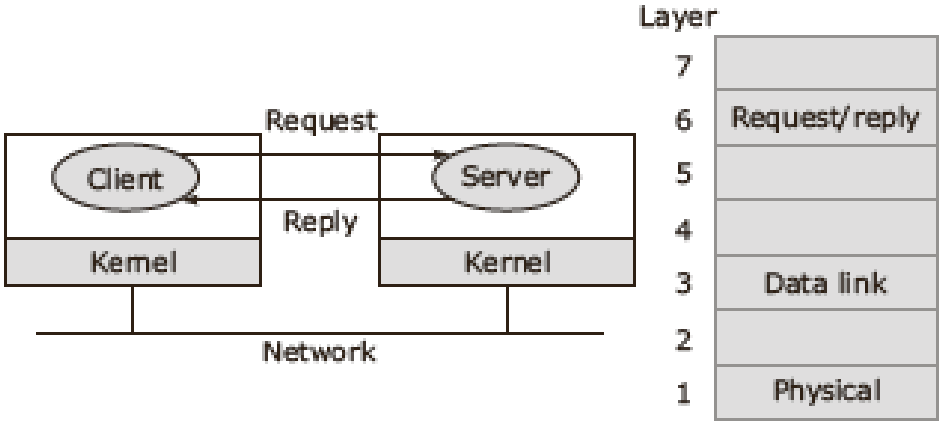

Figure 1-17 The client-server model

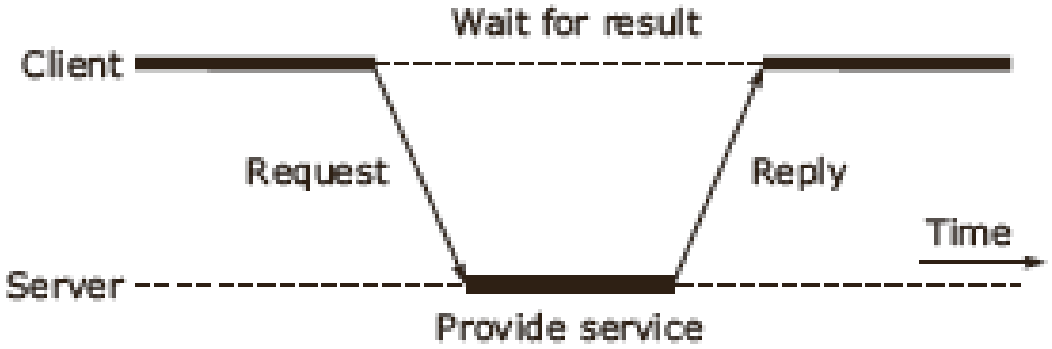

Figure 1-18 The Client Server interaction

Client Server addressing techniques

- Machine addressing,
- process addressing
- Name server addressing

Client Server addressing techniques

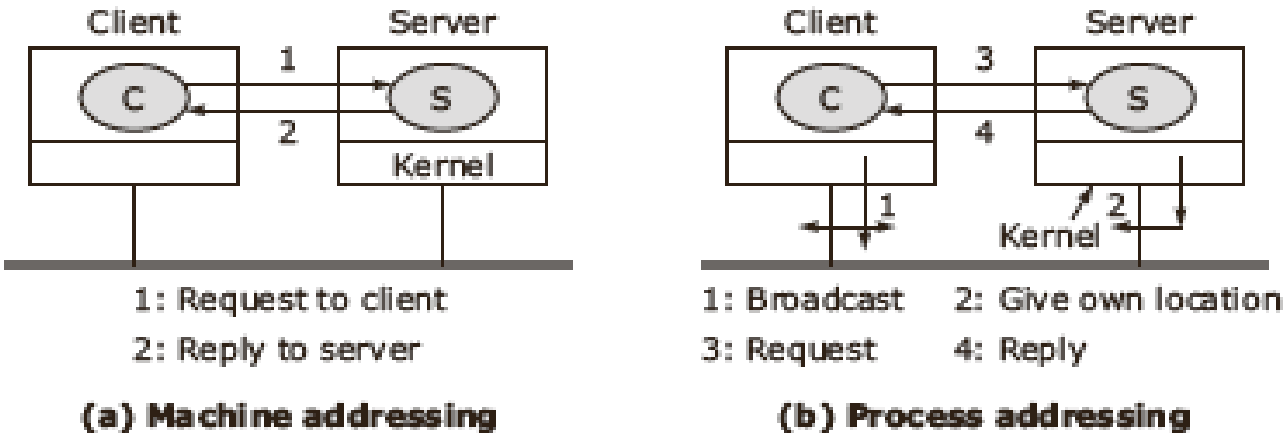

Figure 1-19 Addressing techniques

Client Server implementation

- **Messages for client server interaction**
  - Request, Reply, Acknowledge, Are you Alive, I am Alive.

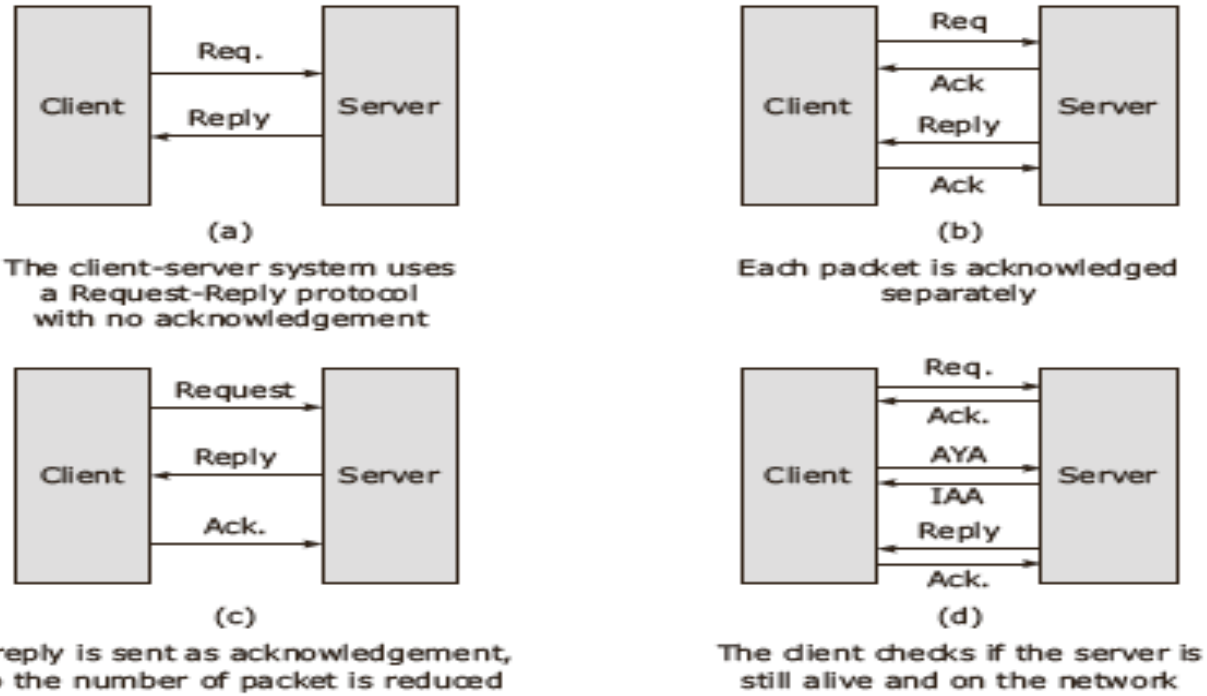

Figure 1-20 Common packet message sequences

differentiation between the client and the server

- User interface level
- Processing level
- data level

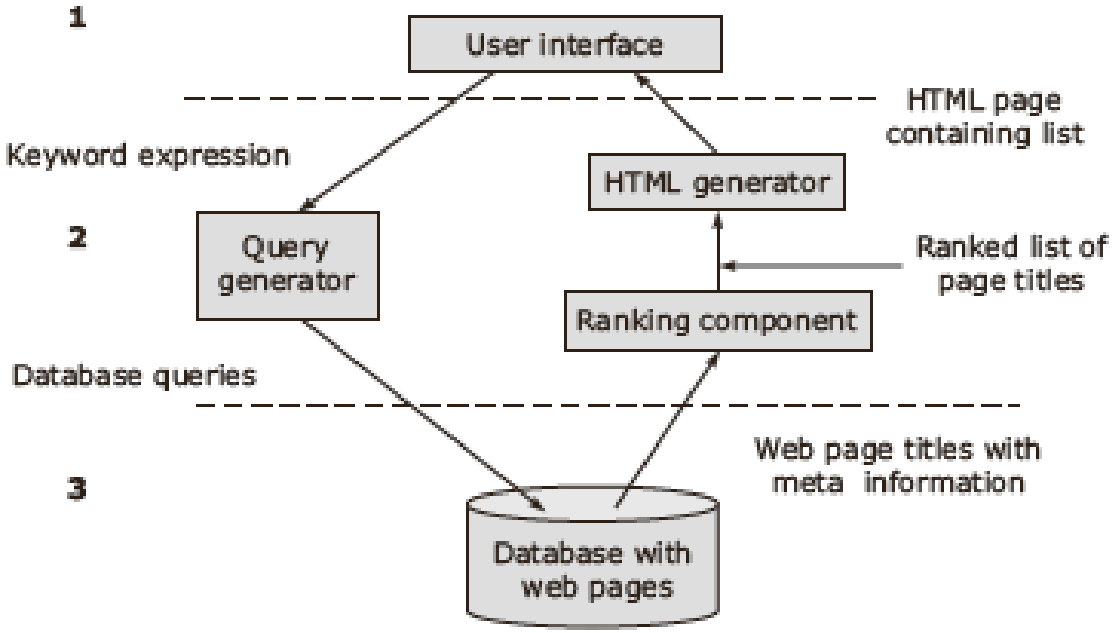

Figure 1-21 Three tiered Internet search engine

Client Server Architecture

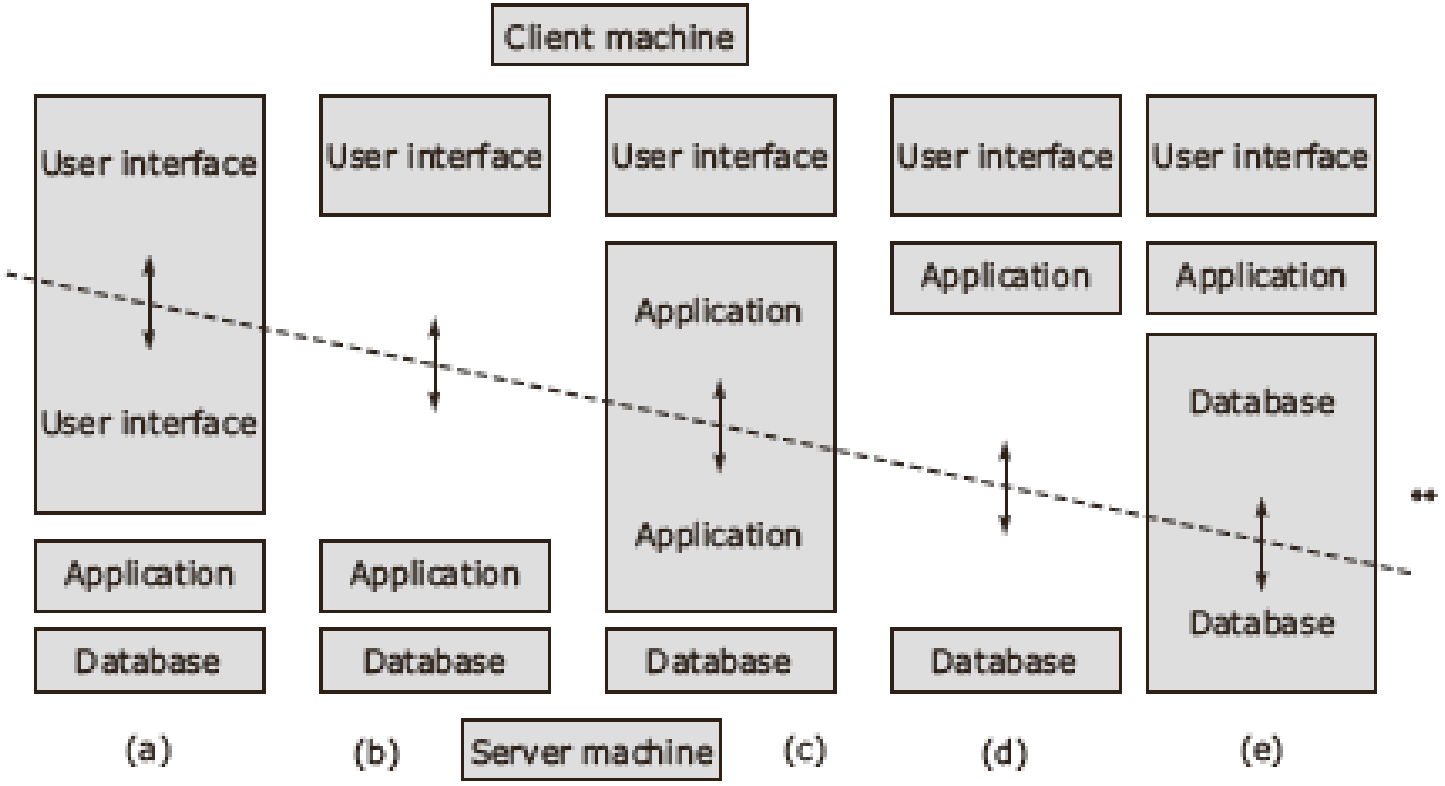

Figure 1-22 Multi-tiered client-server architecture

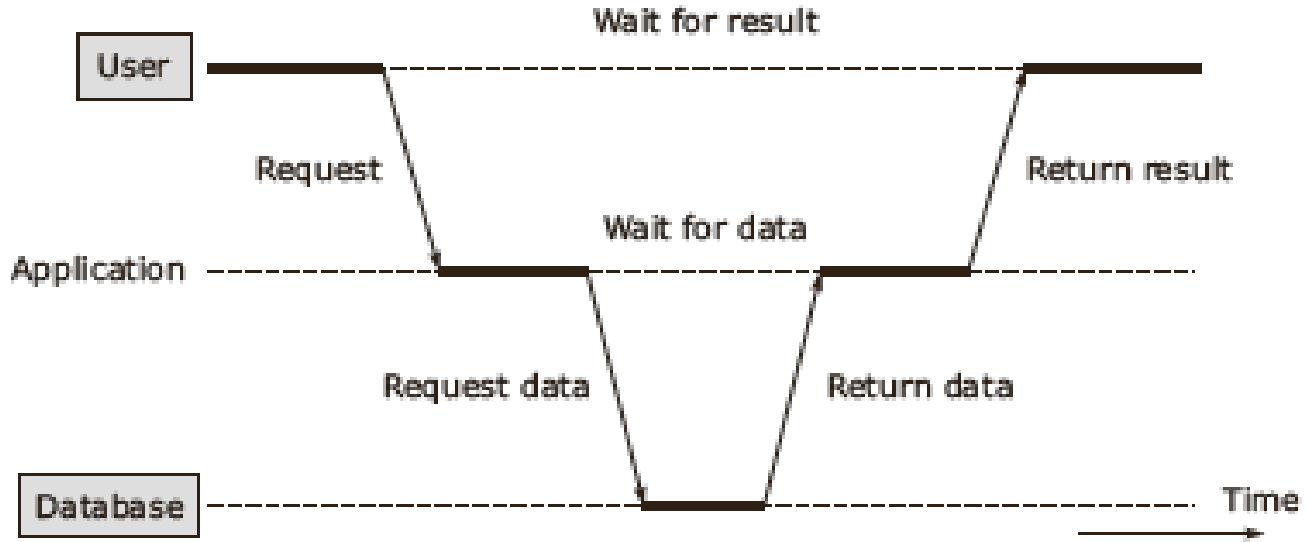

Figure 1-23 Server acts as a client

Case Study: World Wide Web 1.0

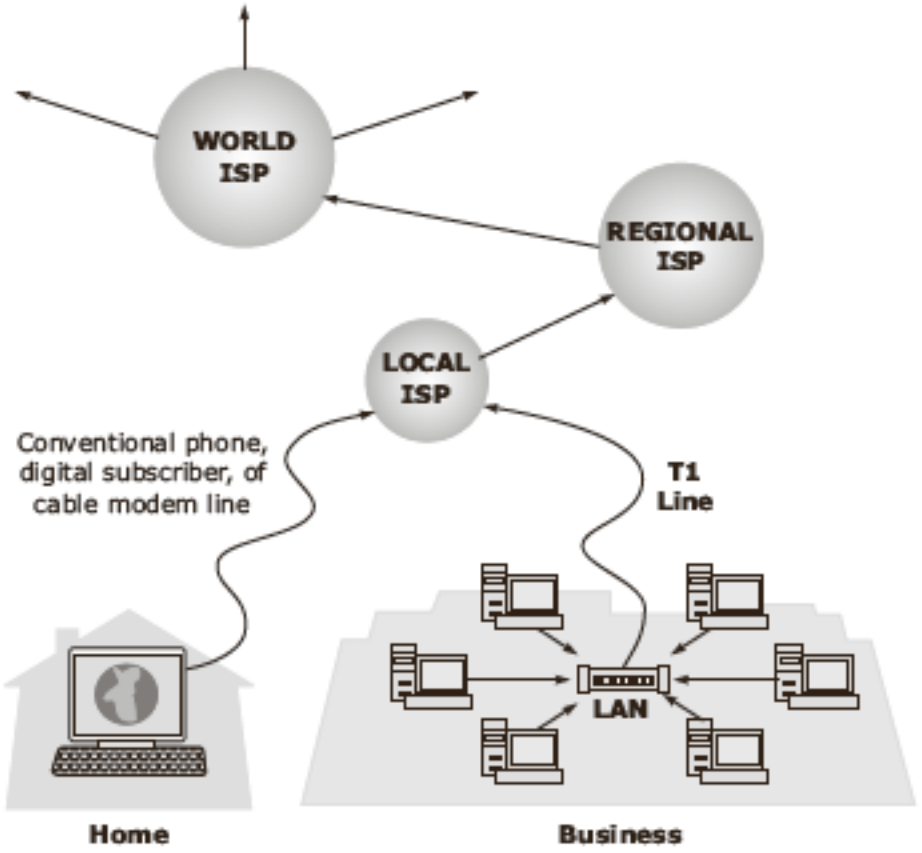

Figure 1-24 WWW architecture

Internet scenario with web servers and web browsers

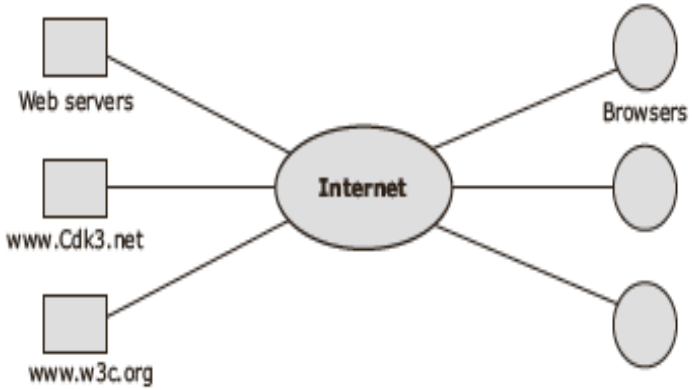

Figure 1-25 Web servers and web browsers

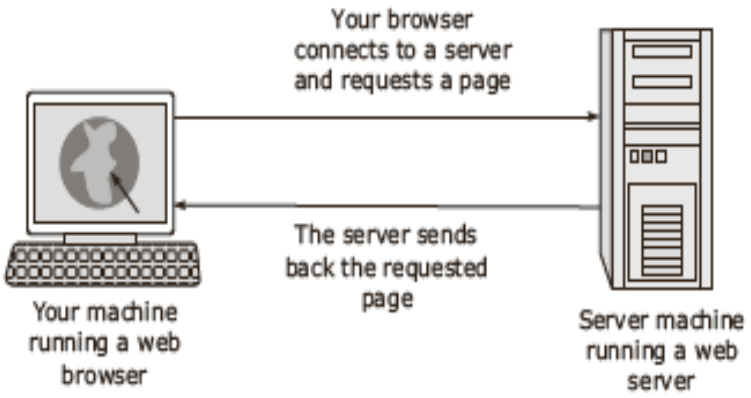

Figure 1-26 Accessing a URL

Case study: **World Wide Web 2.0**

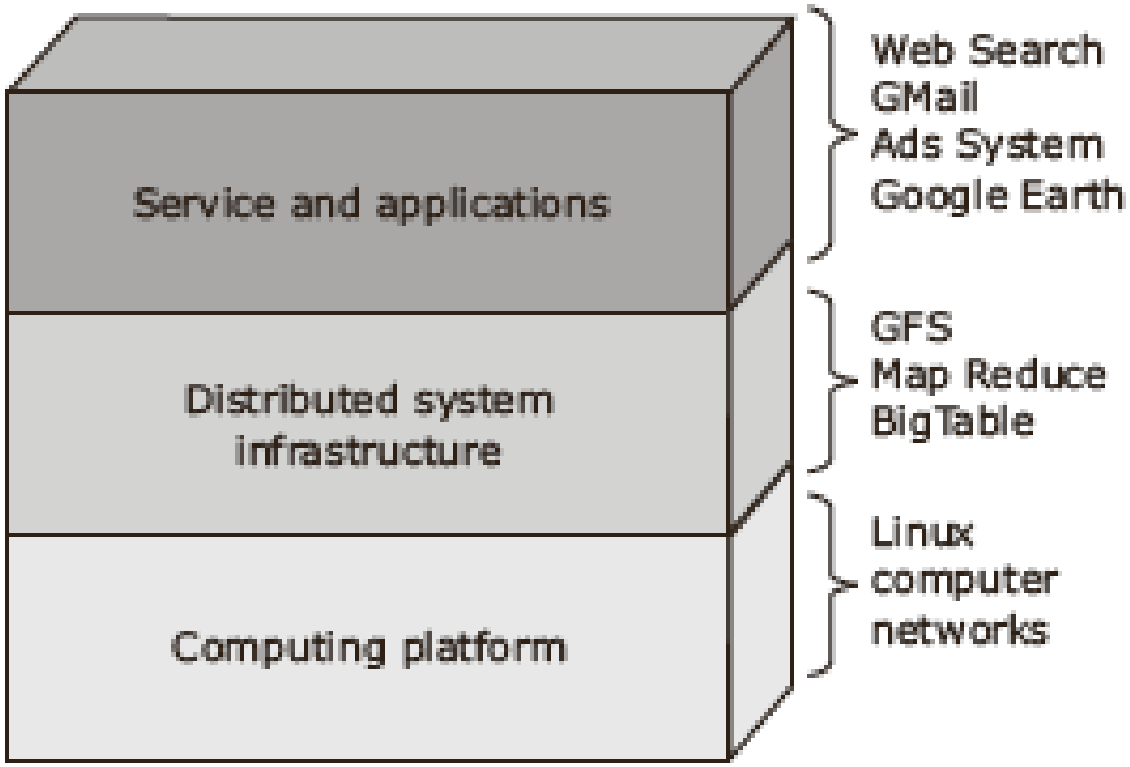

Figure 1-27 Google architecture

Case Study: Google Servers

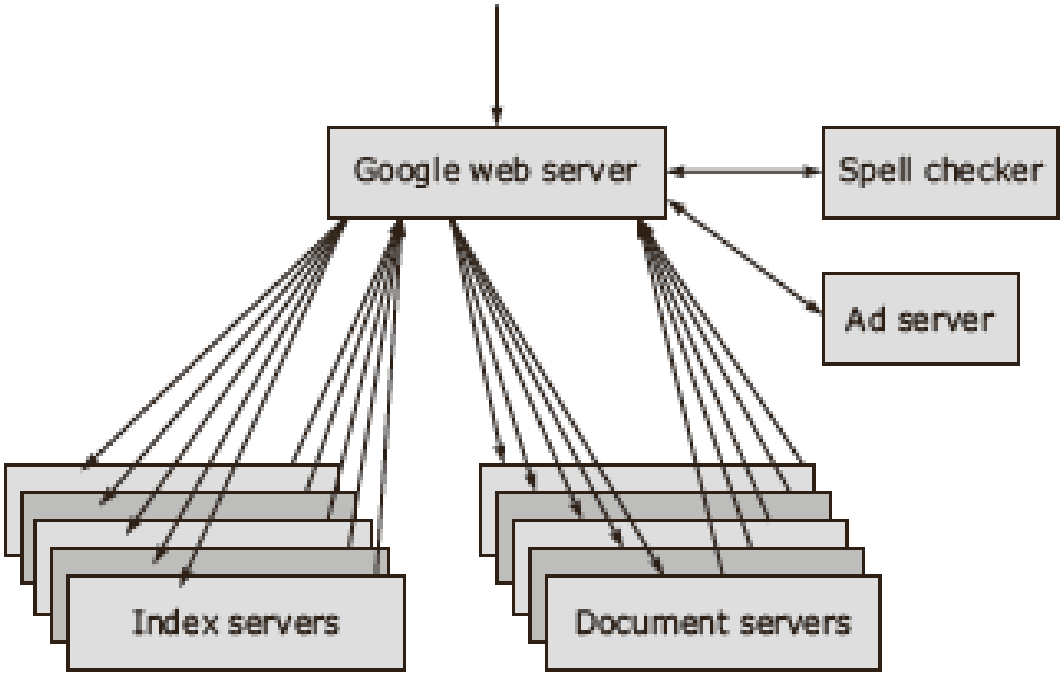

Figure 1-28 Google servers

Summary

- Multiple-interconnected computers can have either shared memory or distributed memory architectures
- Distributed systems offer integration of distributed applications, resource sharing, more reliability, better flexibility
- Network operating system, distributed operating system and multiprocessor timesharing system are different types of distributed systems

- This slide is available on

www.worldsj.wordpress.com