Client – Server & Distributed System

A Basic Introduction

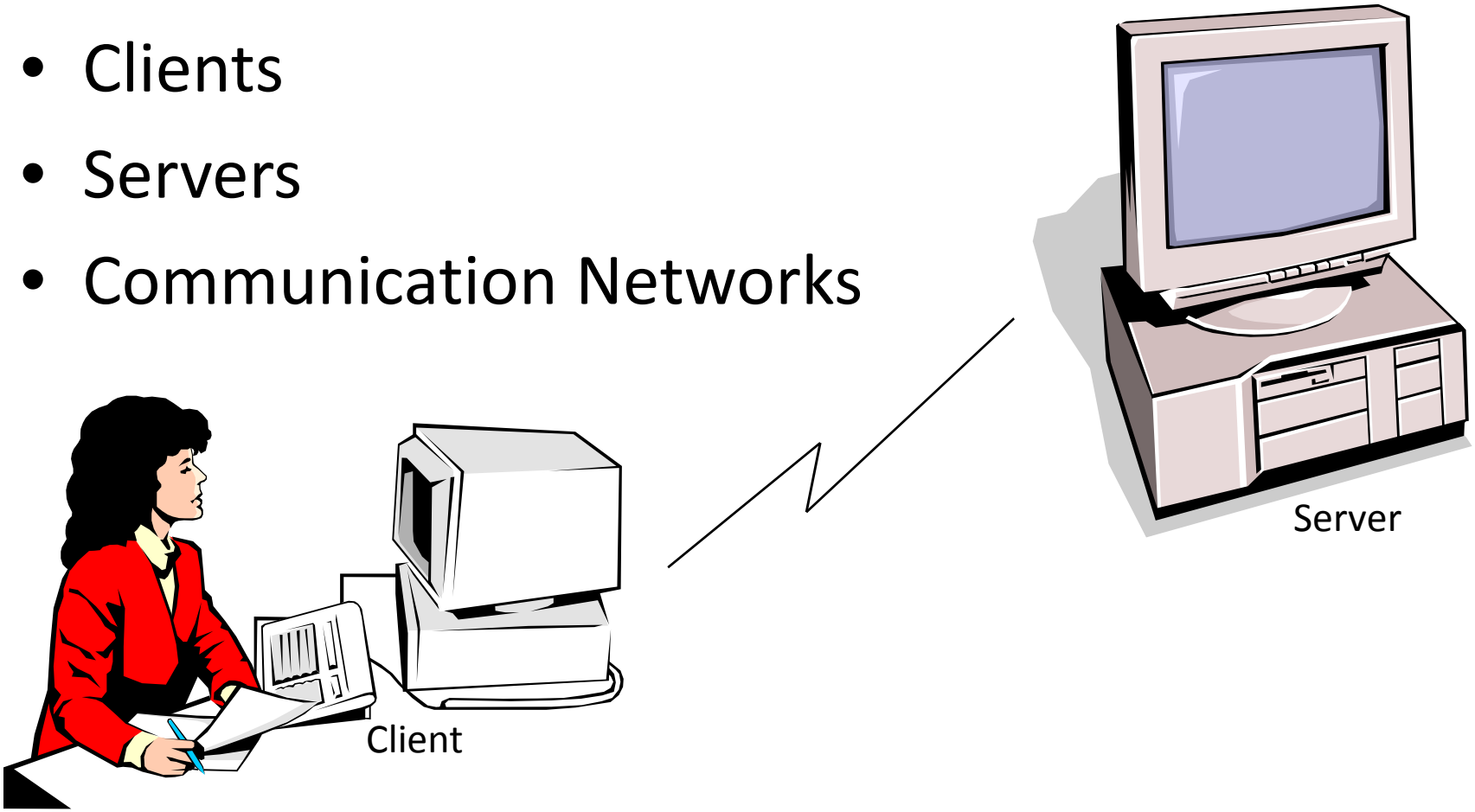
Client Server Architecture

- A network architecture in which each computer or process on the network is either a *client* or a *server*.

Source: http://webopedia.lycos.com

Components

- Clients
- Servers
- Communication Networks



Clients

- Applications that run on computers
- Rely on servers for
  - Files
  - Devices
  - Processing power
- Example: E-mail client
  - An application that enables you to send and receive e-mail

Clients are Applications

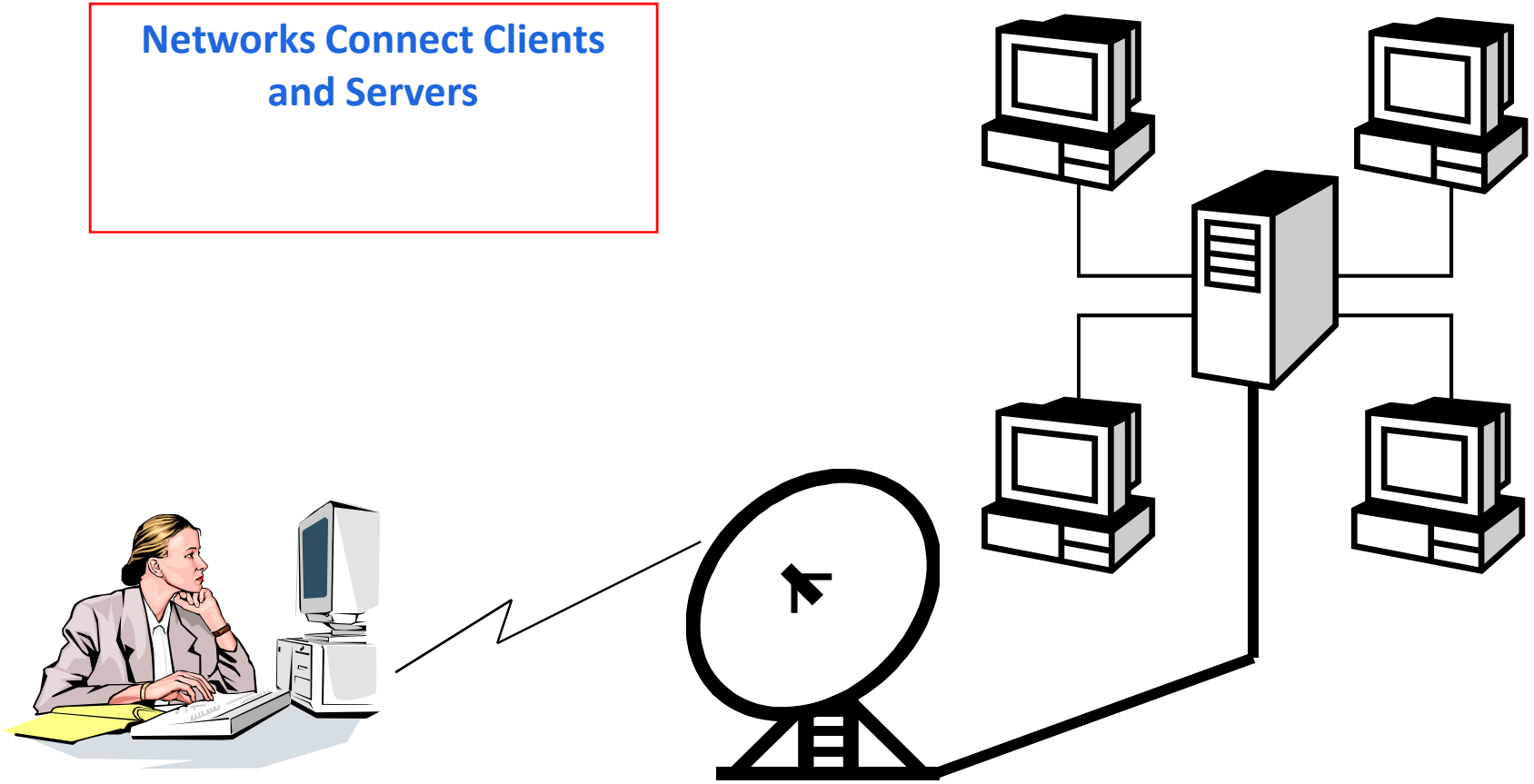
Servers

- Computers or processes that manage network resources
  - Disk drives (file servers)
  - Printers (print servers)
  - Network traffic (network servers)
- Example: Database Server
  - A computer system that processes database queries

Servers Manage Resources

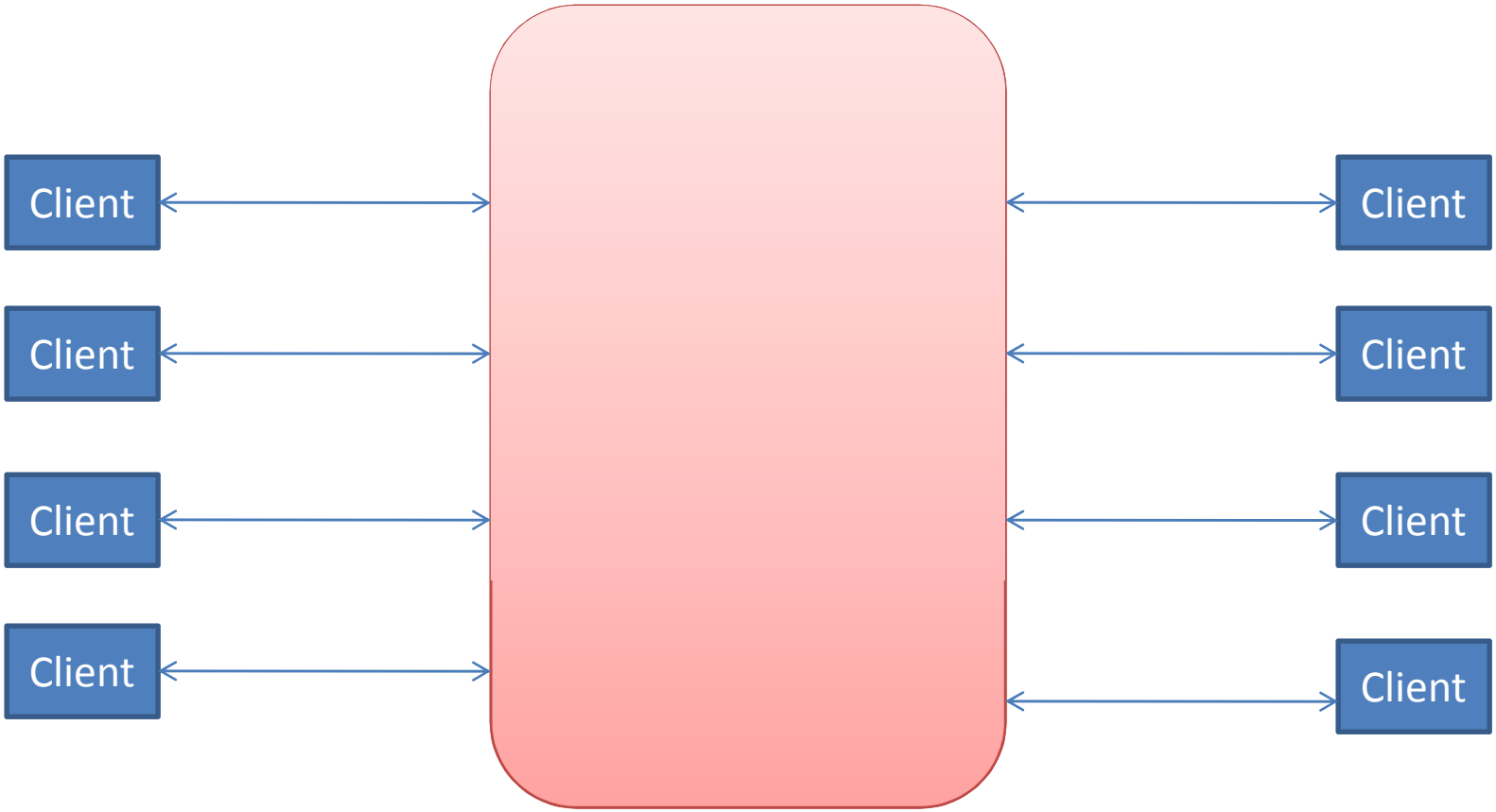
Communication Networks

Networks Connect Clients and Servers



Client–Server Computing

- The computing environment might consists of collection of equally powerful computers having same processor speed and equal amount of memory.
- The equal distribution of resources typically does not provide the best services to users.
- An alternative distribution of resources is to buy at least one machine much more powerful and have the other machines arranged so that users may connect to the more powerful machine when they need.



The client/server design provides users with a means to issue commands which are sent across a network to be received by a server which executes their commands for them. The results are then sent back to the client machine which sent the request in order that the user may see the results.

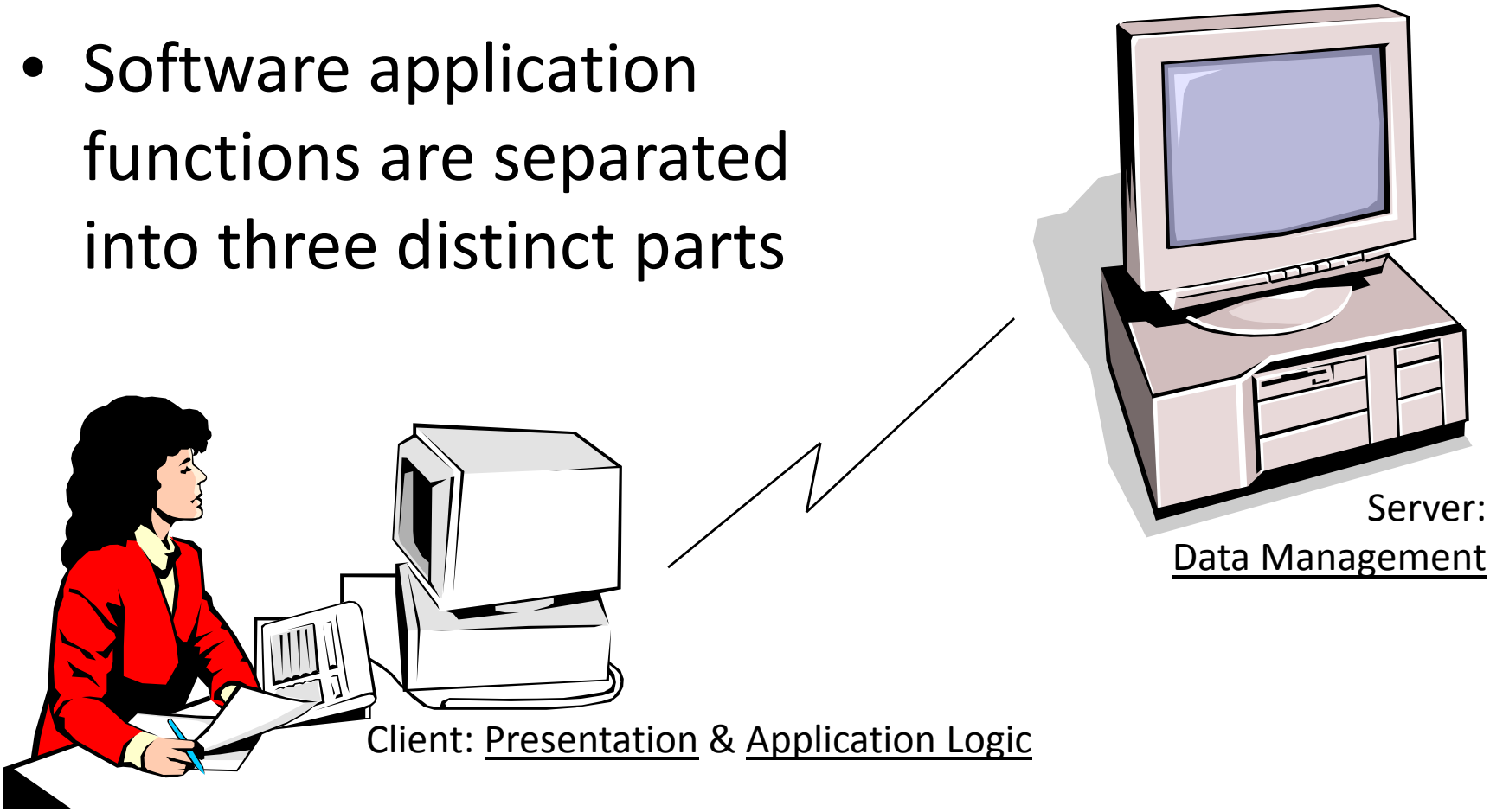
Client–Server Computing

- Process takes place
  - on the server and
  - on the client
- Servers
  - Store and protect data
  - Process requests from clients
- Clients
  - Make requests
  - Format data on the desktop

Client-Server Computing Optimizes Computing Resources

Application Functions

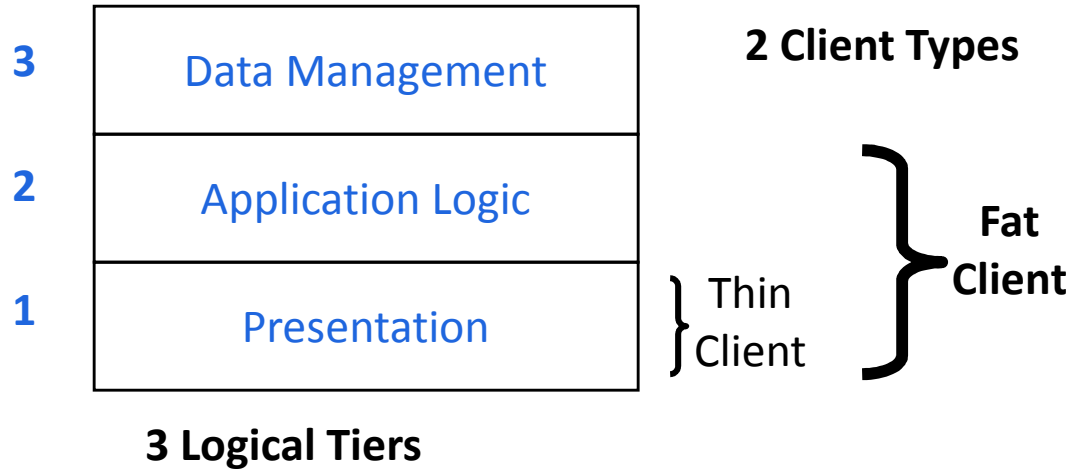
- Software application functions are separated into three distinct parts



Server:
Data Management

Client: Presentation & Application Logic

Application Components

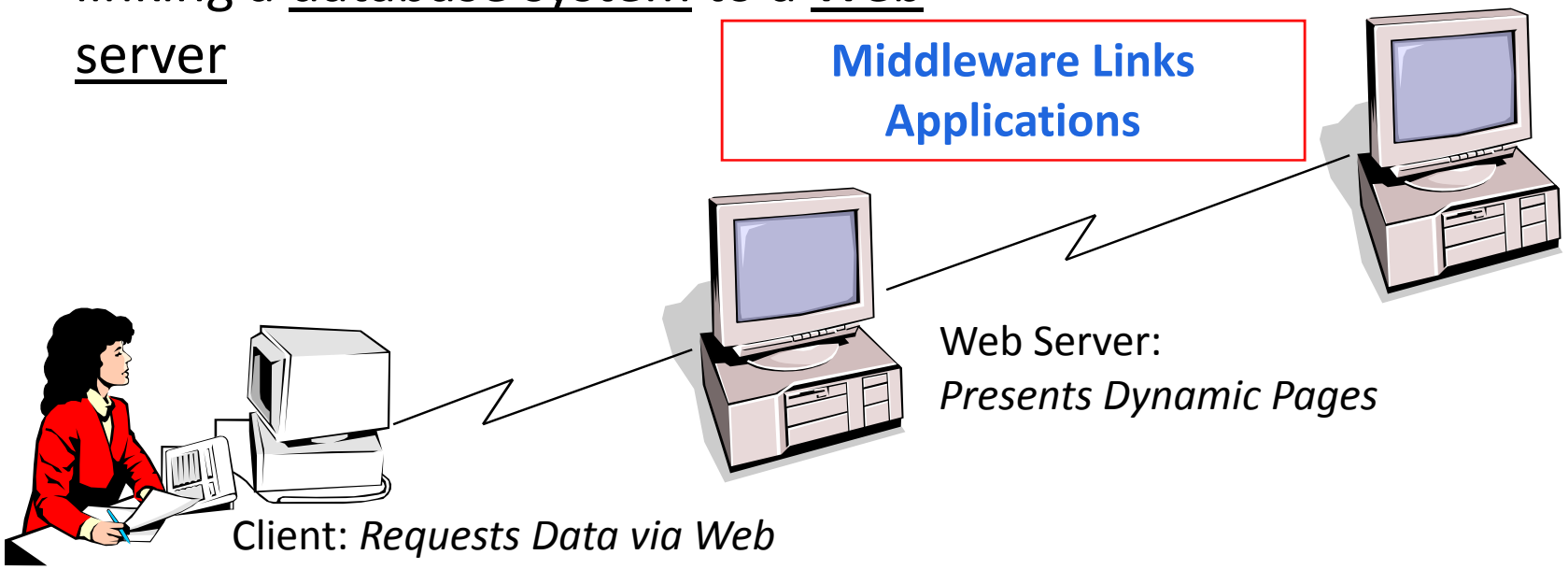


Database Applications:

Most common use of client-server architectures

Middleware

- Software that connects two otherwise separate applications
- Example: Middleware product linking a database system to a Web server



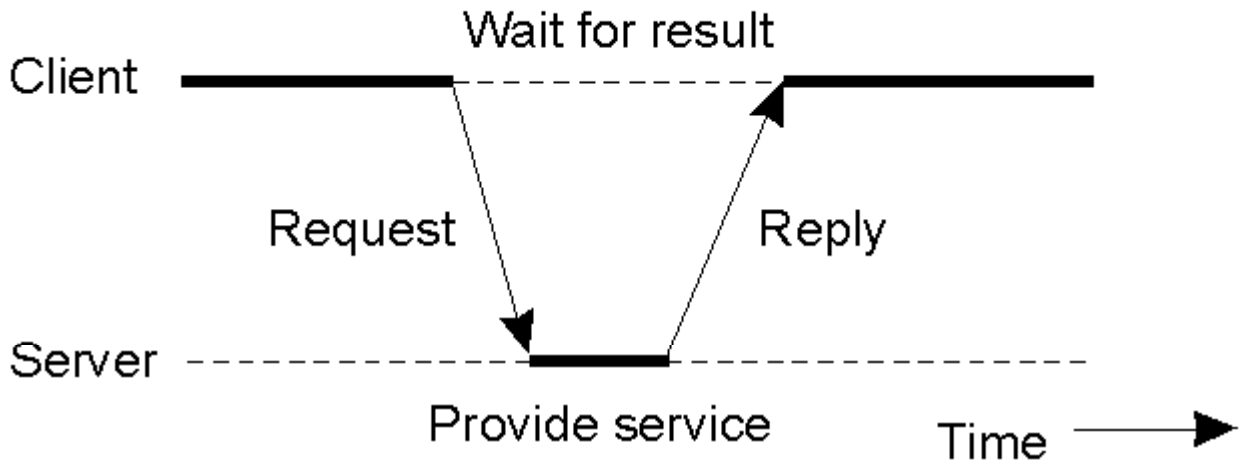
Types of Servers

From A to Z

- Application Servers
- Audio/Video Servers
- Chat Servers
- Fax Servers
- FTP Servers
- Groupware Servers
- IRC Servers
- List Servers
- Mail Servers
- News Servers
- Proxy Servers
- Telnet Servers
- Web Servers

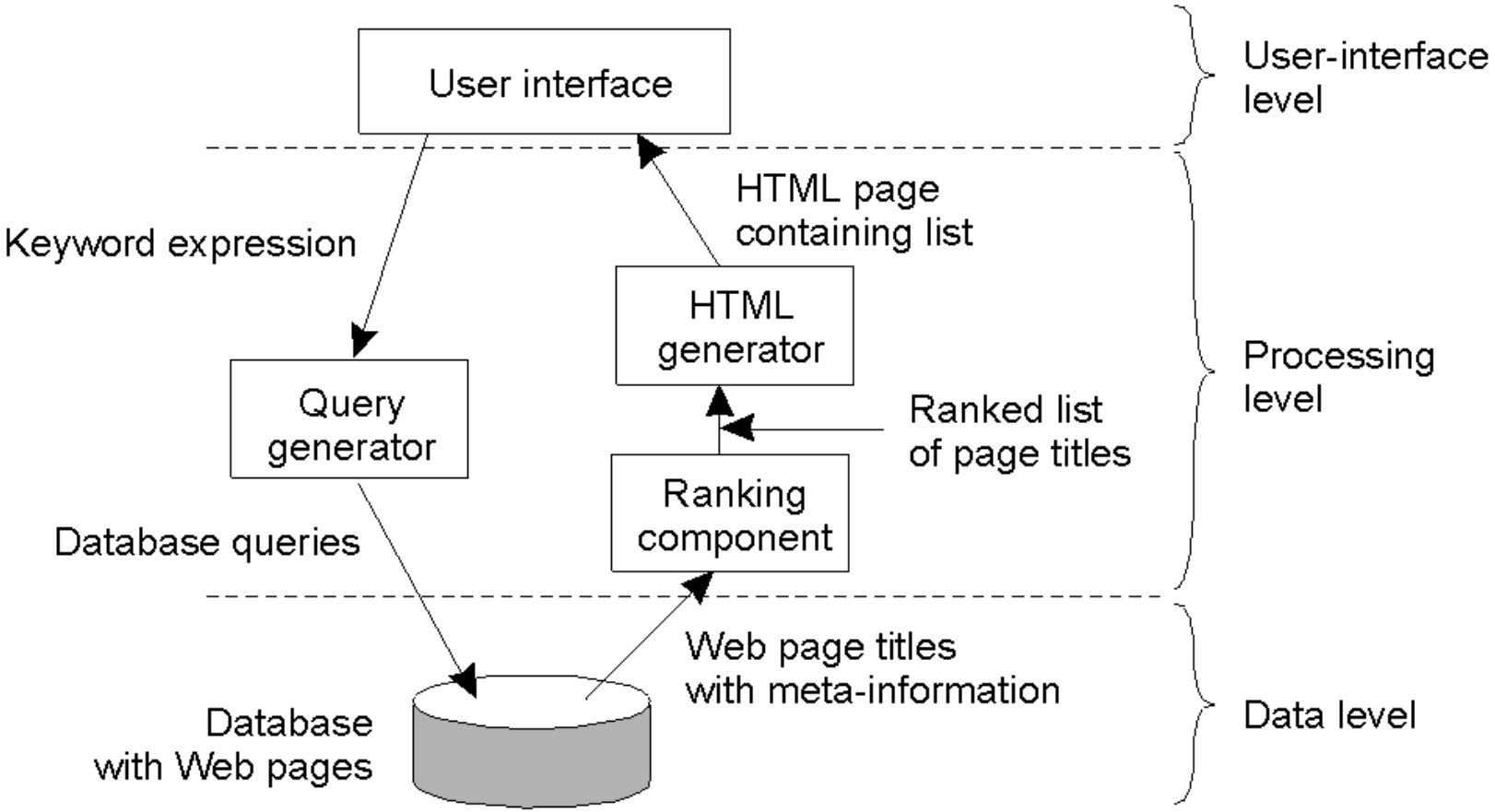
Source: http://webopedia.lycos.com

Client-Server Model

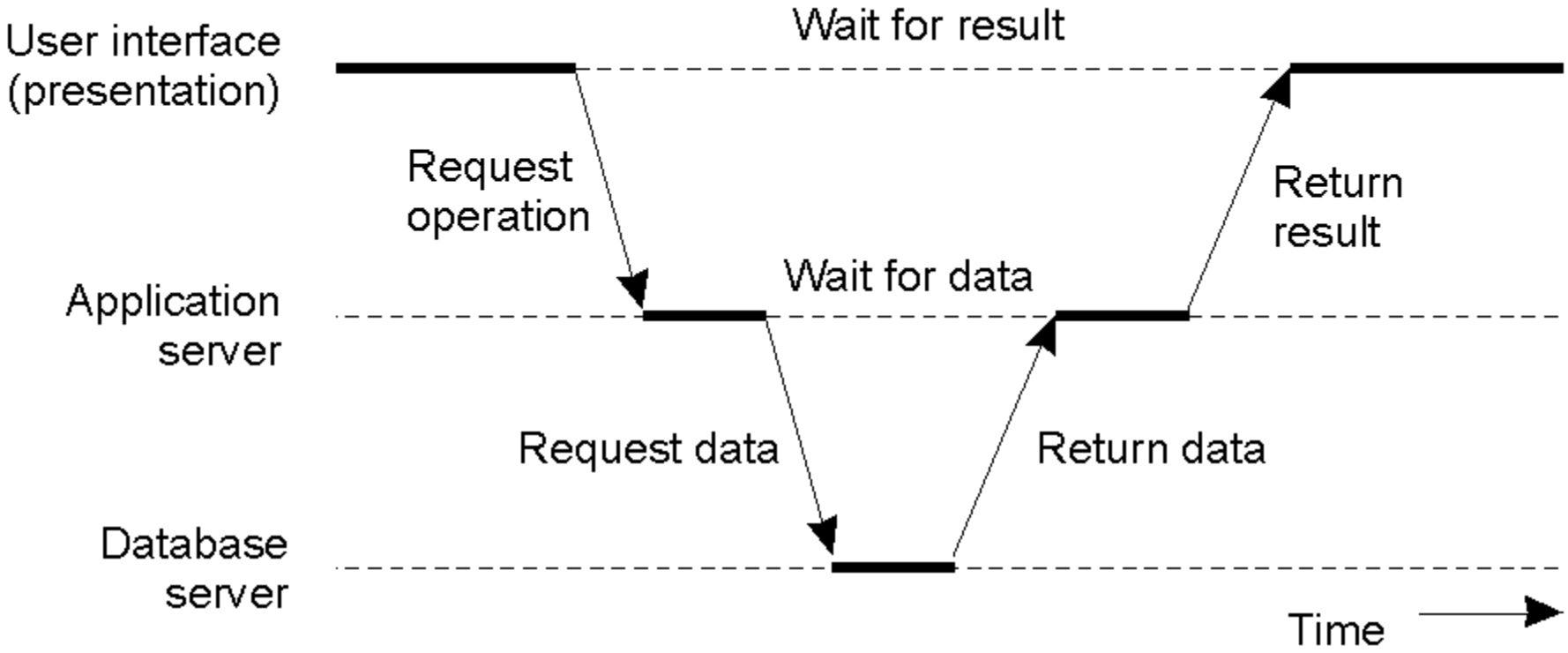


- Use TCP/IP for reliable network connection.
  - This implies the client must establish a connection before sending the first request.

Internet Search Engine

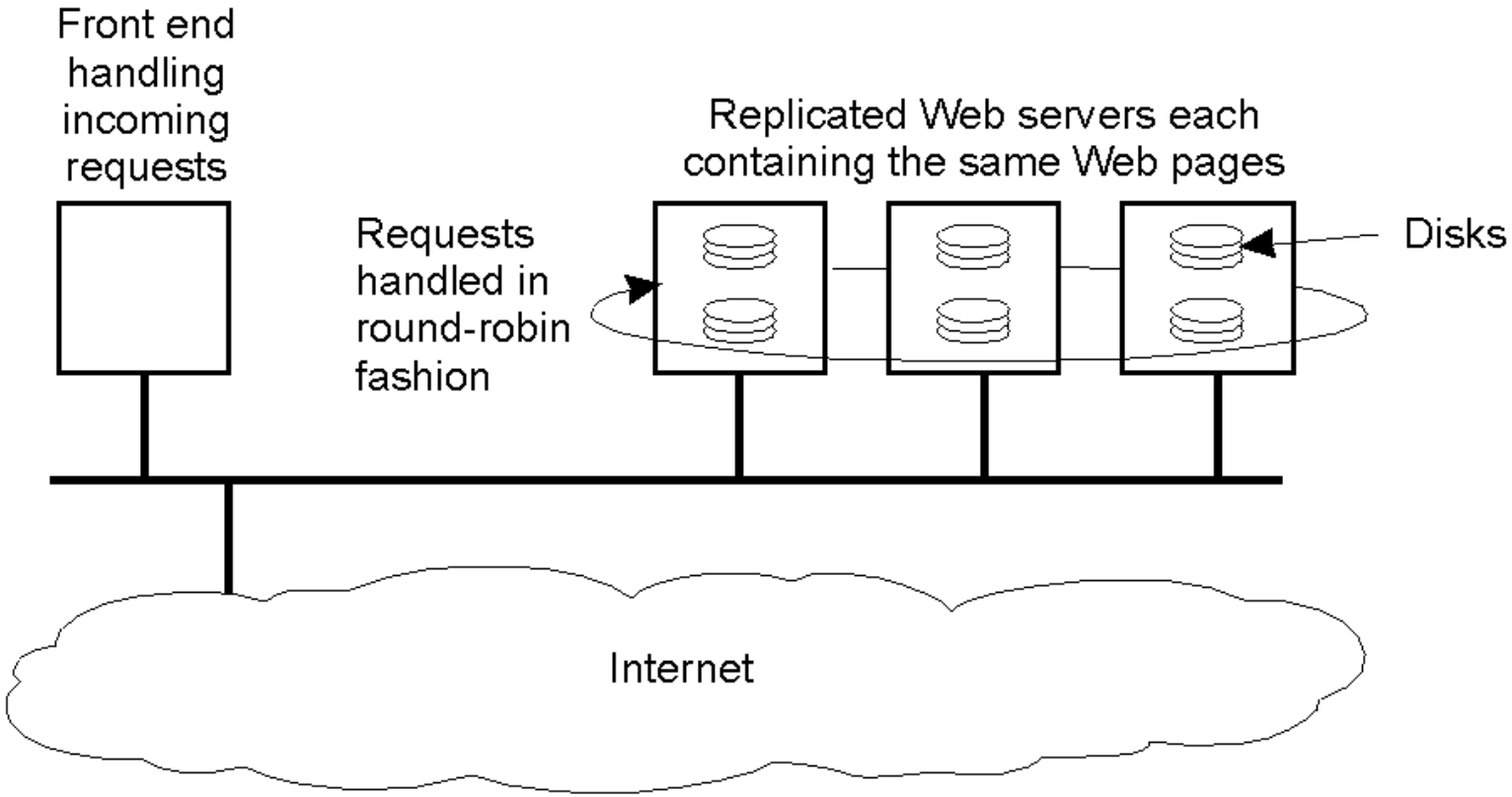


Multitiered Architectures: 3 tiers



- Server may act as a client
  - Example would be transaction monitor across multiple databases

Horizontal Distribution



- Distribute servers across nodes
  - E.g., Web server “farm” for load balancing
- Distribute clients in peer-to-peer systems.

Introduction to Distributed Systems

- Why do we develop distributed systems?
  - availability of powerful yet cheap microprocessors (PCs, workstations), continuing advances in communication technology,

- **What is a distributed system?**
- A distributed system is a collection of independent computers that appear to the users of the system as a single system.
- Examples:
  - Network of workstations
  - Distributed manufacturing system (e.g., automated assembly line)
  - Network of branch office computers

- Definition: a *distributed system* is
  - A collection of independent computers that appears to its users as a single coherent system.

  - one that looks like an ordinary system to its users, but runs on a set of autonomous processing elements (PEs) where each PE has a separate physical memory space and the message transmission delay is not negligible.

  - There is close cooperation among these PEs. The system should support an arbitrary number of processes and dynamic extensions of PEs.

Definition of Distributed System

- Distributed Systems encounter number of terminologies:
  - distributed, network, parallel, concurrent, and decentralized.

- Parallel means lockstep actions on a data set from a single thread of control.

- Distributed means that the cost or performance of a computation is governed by the communication of data and control.

- A system is centralized if its components are restricted to one site, decentralized if its components are at different sites with no or limited or close coordination

- When a decentralized system has no or limited coordination, it is called networked; otherwise, it is termed distributed indicating a close coordination among components at different sites.

Advantages of Distributed Systems over Centralized Systems

- **Economics**: a collection of microprocessors offer a better price/performance than mainframes. Low price/performance ratio: cost effective way to increase computing power.
- **Speed**: a distributed system may have more total computing power than a mainframe. Ex. 10,000 CPU chips, each running at 50 MIPS. Not possible to build 500,000 MIPS single processor since it would require 0.002 nsec instruction cycle. Enhanced performance through load distributing.
- **Inherent distribution**: Some applications are inherently distributed. Ex. a supermarket chain.
- **Reliability**: If one machine crashes, the system as a whole can still survive. Higher availability and improved reliability.
- **Incremental growth**: Computing power can be added in small increments. Modular expandability
- **Another deriving force**: the existence of large number of personal computers, the need for people to collaborate and share information.

Advantages of Distributed Systems over Independent PCs

- **Data sharing**: allow many users to access to a common data base
- **Resource Sharing**: expensive peripherals like color printers
- **Communication**: enhance human-to-human communication, e.g., email, chat
- **Flexibility**: spread the workload over the available machines

Disadvantages of Distributed Systems

- Software: difficult to develop software for distributed systems
- Network: saturation, lossy transmissions
- Security: easy access also applies to secrete data
- Distribution of control
  - Hard to detect faults
  - Administration issues
- Performance
- Interconnect & servers must scale

Goals of D.S.

- Transparency
- Openness
- Reliability
- Performance
- Scalability

Design Challenges of Distributed Systems

- Designers of distributed systems need to take the following challenges into account:
  - Heterogeneity
    - Heterogeneous components must be able to interoperate.
  - Openness
    - Interfaces should allow components to be added or replaced.
  - Security
    - The system should only be used in the way intended.

Design Challenges of Distributed Systems

- Scalability
  - System should work efficiently with an increasing number of users.
  - System performance should increase with inclusion of additional resources.
- Failure handling
  - Failure of a component (partial failure) should not result in failure of the whole system.
- Transparency
  - Distribution should be hidden from the user as much as possible

Transparency

- How to achieve the single-system image, i.e how to make a collection of computers appear as a single computer.

- Hiding all the distribution from the users as well as the application programs can be achieved at two levels:
  - hide the distribution from users
  - At a lower level, make the system look transparent to programs.

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 or is migrated to newer version
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 accessed
Failure	Hide the failure and recovery of a resource
Persistence	Hide whether a (software) resource is in memory or on disk

Openness

- Make it easier to build and change

- The first step in openness is publishing the documentation of software components and interfaces of the components to make them available to software developers

- **Monolithic Kernel**: systems calls are trapped and executed by the kernel. All system calls are served by the kernel, e.g., UNIX.
- **Microkernel**: provides minimal services
  - IPC
  - some memory management
  - some low-level process management and scheduling
  - low-level i/o (E.g. multiple system interfaces.)

Reliability

- Distributed system should be more reliable than a single system.
  - **Availability**: fraction of time the system is usable.
  - **Redundancy** improves it.
  - Need to maintain **consistency**
  - Need to be **secure**
  - **Fault tolerance**: need to mask failures, recover from errors.

Performance

- Performance loss due to communication delays:
  - fine-grain parallelism: high degree of interaction
  - coarse-grain parallelism

  - *(**Granularity** is the extent to which a system is broken down into small parts, either the system itself or its description or observation)*
- Performance loss due to making the system fault tolerant

Scalability

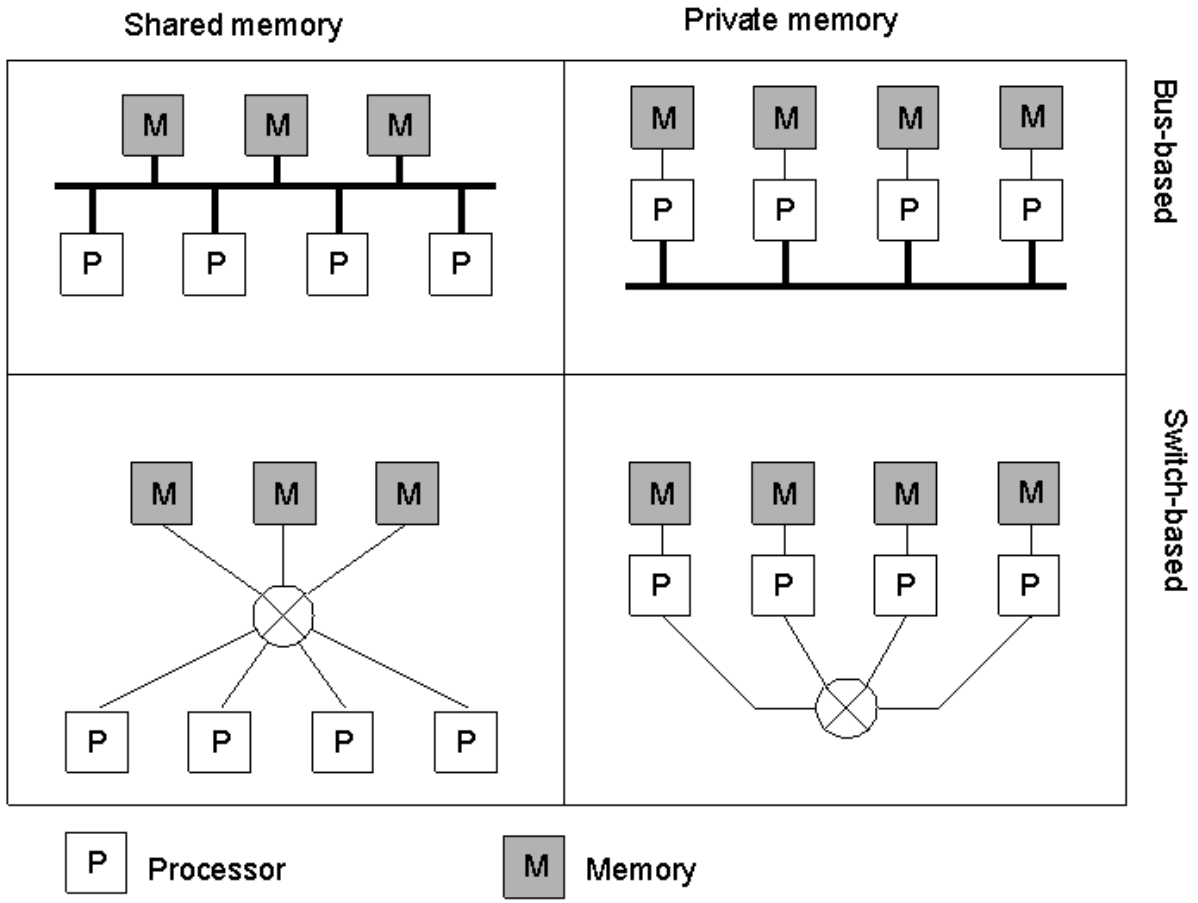
- System should work efficiently with an increasing number of users.
- System performance should increase with inclusion of additional resources
- Techniques that require resources linearly in terms of the size of the system are not scalable. (e.g., broadcast based query won't work for large distributed systems.

Pitfalls when Developing Distributed Systems

- False assumptions made by first time developer:
  - The network is reliable.
  - The network is secure.
  - The network is homogeneous.
  - The topology does not change.
  - Latency is zero.
  - Bandwidth is infinite.
  - Transport cost is zero.
  - There is one administrator.

Hardware Concepts

Basic organizations and memories in distributed computer systems

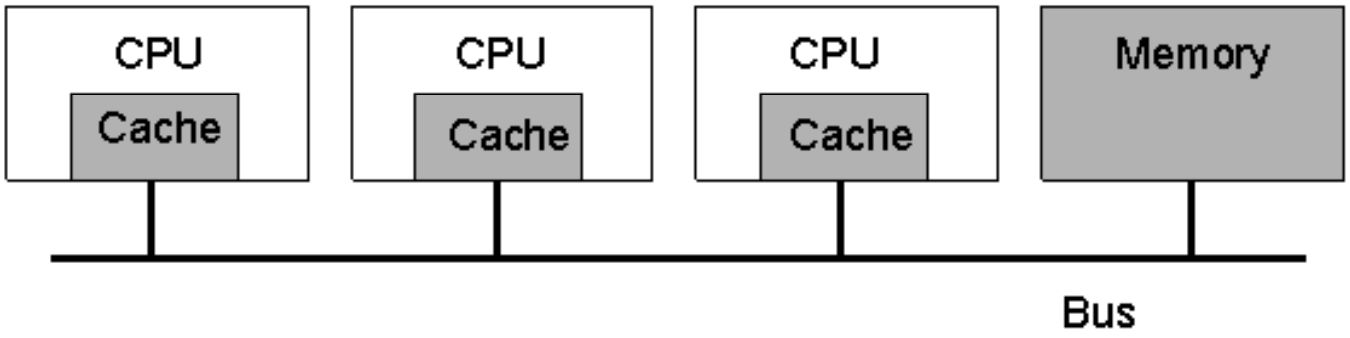


Hardware Considerations

- General Classification:
  - Multiprocessor – a single address space among the processors
  - Multicomputer – each machine has its own private memory.

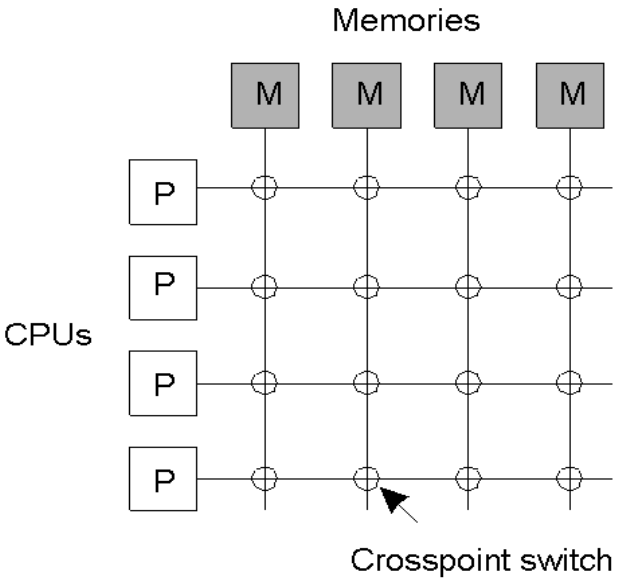
- OS can be developed for either type of environment.

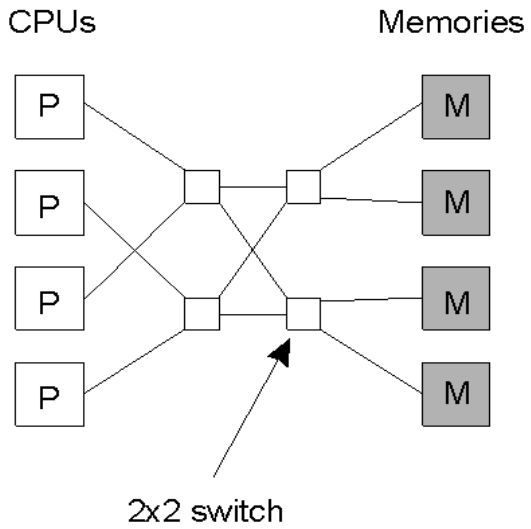
Multiprocessors



A bus-based multiprocessor

Multiprocessors





A crossbar switch

An omega switching network

Enslow's Modle of DS

- Enslow (Scientist) proposed that distributed systems can be examined using three dimensions of hardware, control, and data.

- Distributed system = distributed hardware + distributed control + distributed data

- a system can be classified as a distributed system if all three categories (hardware, control, and data) reach a certain degree of decentralization.

- Several points in the dimension of hardware organization are as follows:
  - H1. A single CPU with one control unit.
  - H2. A single CPU with multiple ALUs (arithmetic and logic units). There is only one control unit

  - H3. Separate specialized functional units, such as one CPU with one floating-point coprocessor.
  - H4. Multiprocessors with multiple CPUs but only one single I/O system and one global memory.
  - H5. Multicomputers with multiple CPUs, multiple I/O systems and local memories.

- Similarly, points in the control dimension in order of increasing decentralization are the following:
  - C1. Single fixed control point. Note that physically the system may or may not have multiple CPUs.

- C2. Single dynamic control point. In multiple CPU cases the controller changes from time to time among CPUs.
- C3. A fixed master/slave structure. For example, in a system with one CPU and one coprocessor, the CPU is a fixed master and the coprocessor is a fixed slave.

- C4. A dynamic master/slave structure. The role of master/slave is modifiable by software.
- C5. Multiple homogeneous control points where copies of the same controller are used.
- C6. Multiple heterogeneous control points where different controllers are used

- The database has two components that can be distributed: files and a directory that keeps track of these files

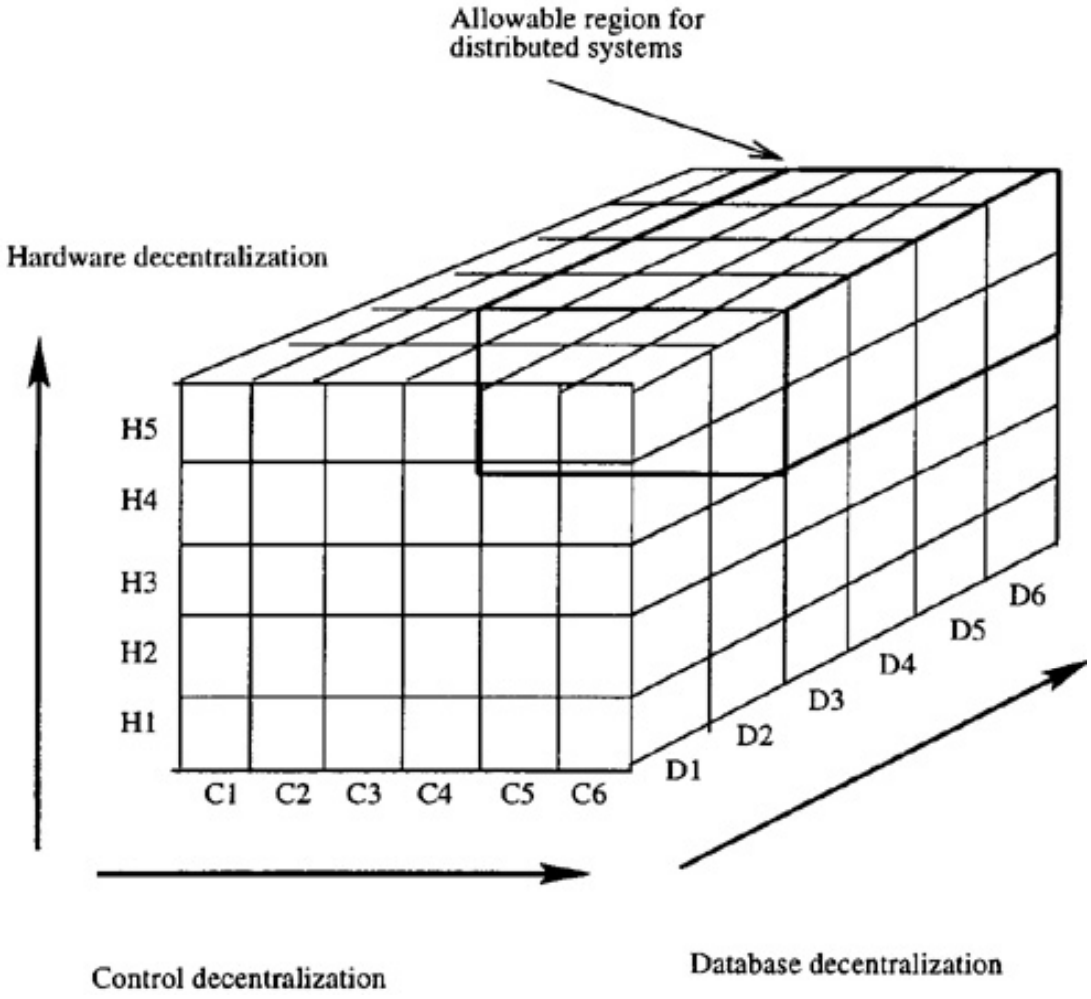
- Distribution can be done in one of two ways, or a combination of both: replication and partition

- A database is partitioned if it is split into sub-databases and then each sub-database is assigned to different sites

- D1. Centralized databases with a single copy of both files and directory.
- D2. Distributed files with a single centralized directory and no local directory.
- D3. Replicated database with a copy of files and a directory at each site.

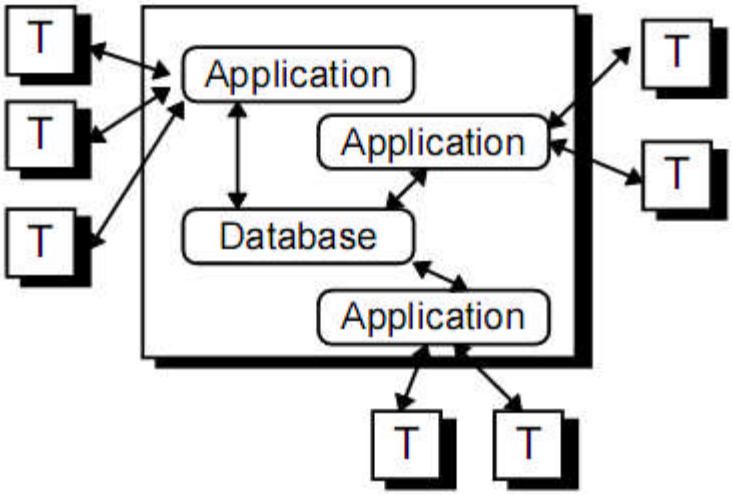
- D4. Partitioned database with a master that keeps a complete duplicate copy of all files.
- D5. Partitioned database with a master that keeps only a complete directory.
- D6. Partitioned database with no master file or directory.

- A system is a distributed one if it has:
  - Multiple processing elements (PEs).
  - Interconnection hardware.
  - Shared states.

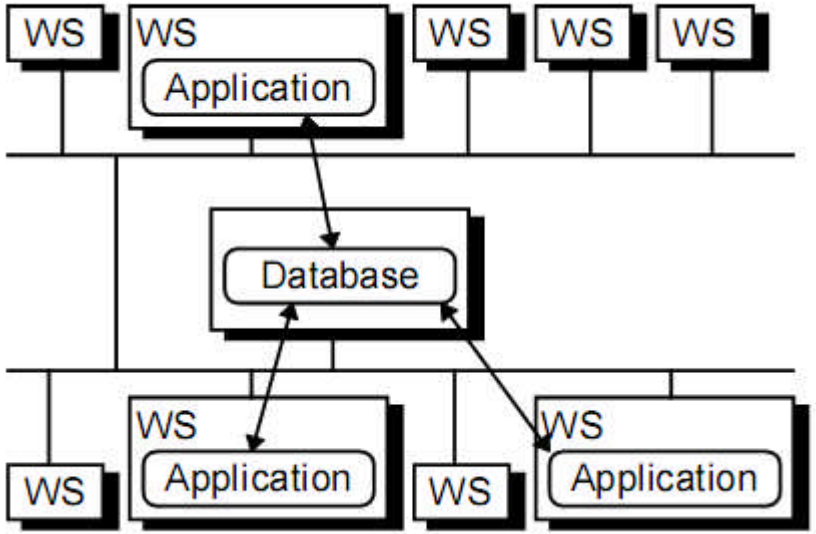


Enslow's model of distributed systems

(Some researchers also considered computer networks and parallel computers as part of distributed system)



Traditional Applications



Distributed Applications

Role of DCE

- Distributing applications requires the creation of a distributed environment in which they can run.

- Many vendors have already solved some of the relevant problems for their proprietary environments.

- OSF DCE provides a vendor-neutral solution
  - It's a platform for building distributed applications
  - It can support a range of commercial applications
  - It builds on work already done by vendors

Requirement of Distributed Env.

- A supporting protocol for distributed Applications
- Mechanisms to exploit the environment's inherent parallelism.
- A way to locate distributed services, i.e., a directory service
- Security services, A mechanism for synchronizing the internal clocks of distributed systems.

Requirement …..

- Support for simple systems:
  - Personal computers
  - Diskless system

- Optionally, some number of distributed applications such as:
  - A distributed file service
  - A network print service
  - Others

Distribution Problems

- What approach should be used to distribute Applications?
  - Remote Procedure Call (RPC)
  - Message Passing TCP/IP
- What directory service(s) should be used?
  - A local directory must be fast and flexible
  - A global directory must be standard and widely supported

Problems ….

- How should security be provided?
  - What services are needed?
  - What mechanisms should be used to provide those services ?
- What protocol should be used to synchronize clocks?
  - There are several possible choices
- How can simple systems be supported?
  - Provide special treatment for PCs and diskless workstations.
  - Alternatively, treat them like any other system in the distributed environment

Problems…

- What distributed applications should be provided?
  - A distributed file service is essential
  - There are many other possibilities

DCE Approach

- Distributing applications
  - Use remote procedure call (RPC)
- Allowing parallelism
  - Support a Threads package
- Directory Services
  - Use a Cell Directory Service for local lookups
  - Provide options for a global directory service

DCE …

- Security
  - Provide authentication, authorization, data integrity, and data privacy

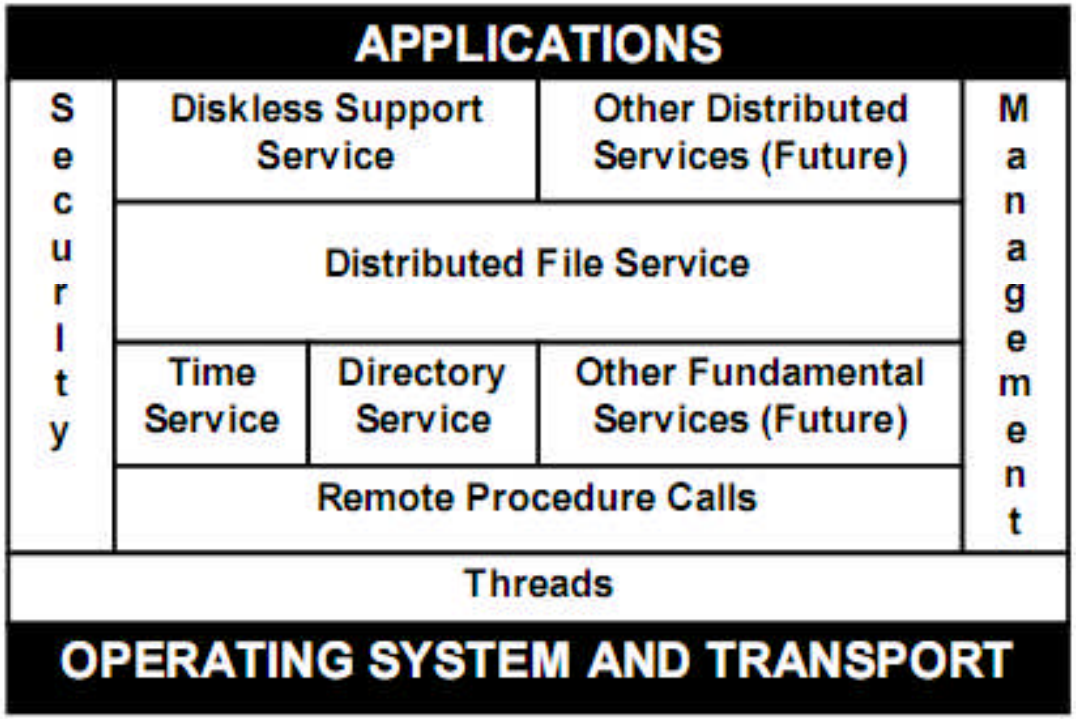
- integrity, and data privacy
  - Use a Distributed Time Service (DTS)
  - Allow some integration with the widely used Network Time Protocol (NTP)

DCE ..

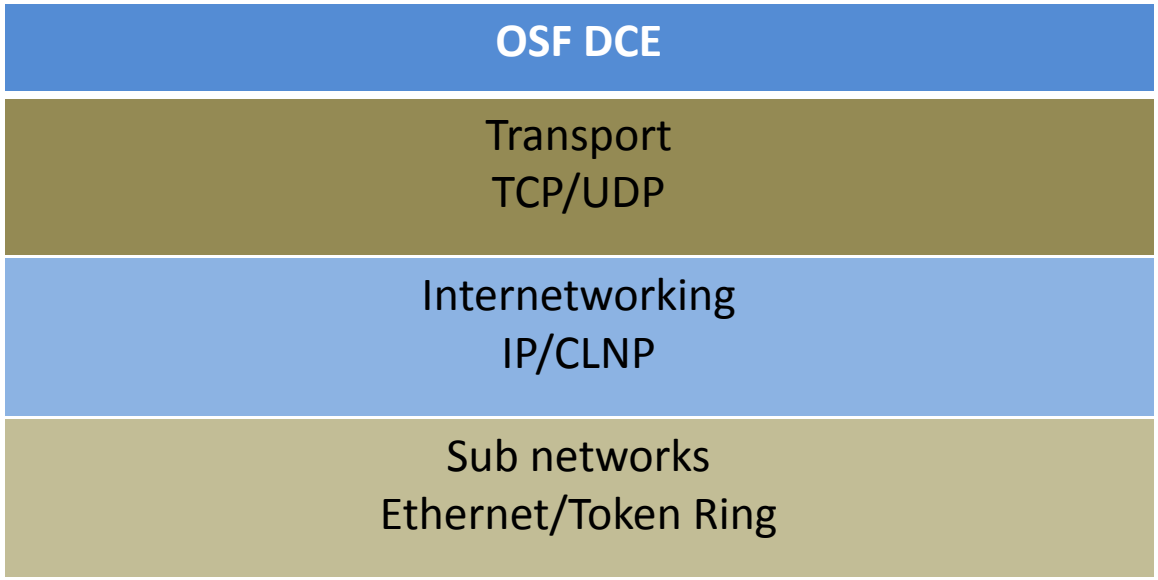
- Simple Systems
  - Provide special services for diskless support
  - Treat PCs like any other system

- Distributed applications
  - Provide a Distributed File Service (DFS)
  - Allow the creation of a common distributed environment to encourage competition among application developers

OSF DCE: A System View



DCE & Distributed Computing



Open Source Foundation (OSF) DCE: A Layered View

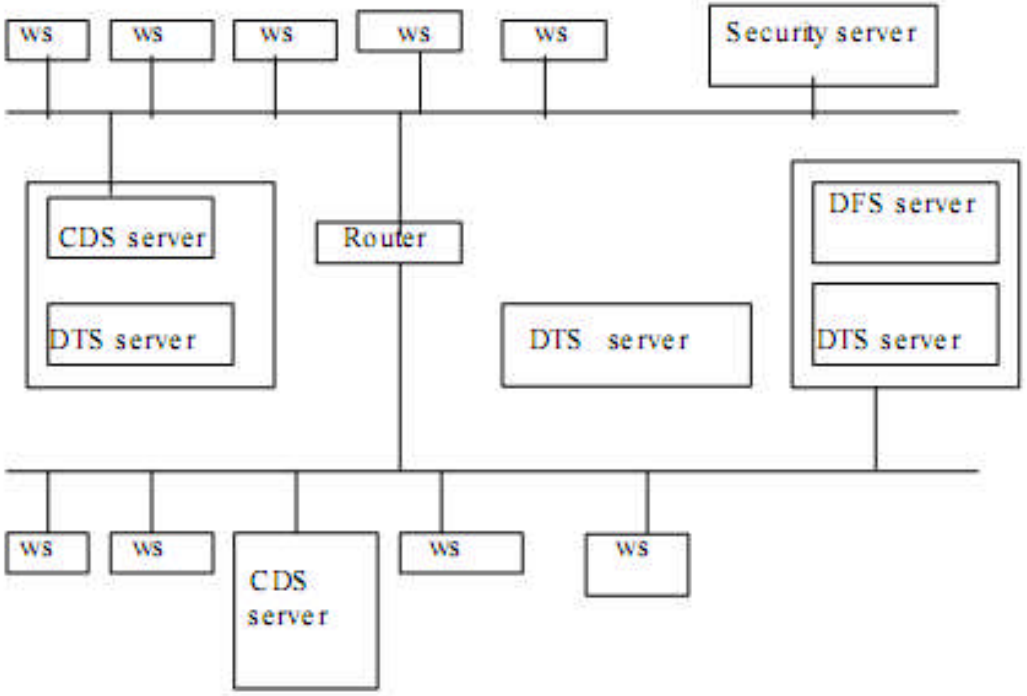
DCE: Clients & Servers

- DCE relies on the notion of clients and servers
- Clients request services
- Servers provide services
- A single machine may support both the clients and servers
- A single process may act as both a client and a server at different times

cell

- Mostly clients perform most of their communication with only a few servers.
- In DCE, clients and servers that communicate mostly with one another are grouped into a cell
- A cell is an administrative unit
- Every machine belongs to one cell
- A cell may consists of two to thousand systems
- DCE optimizes intra-cell communication

cells


ws
ws
ws
ws
ws
Security server
CDS server
DTS server
Router
DTS server
DFS server
DTS server
ws
ws
CDS server
ws
ws

cells