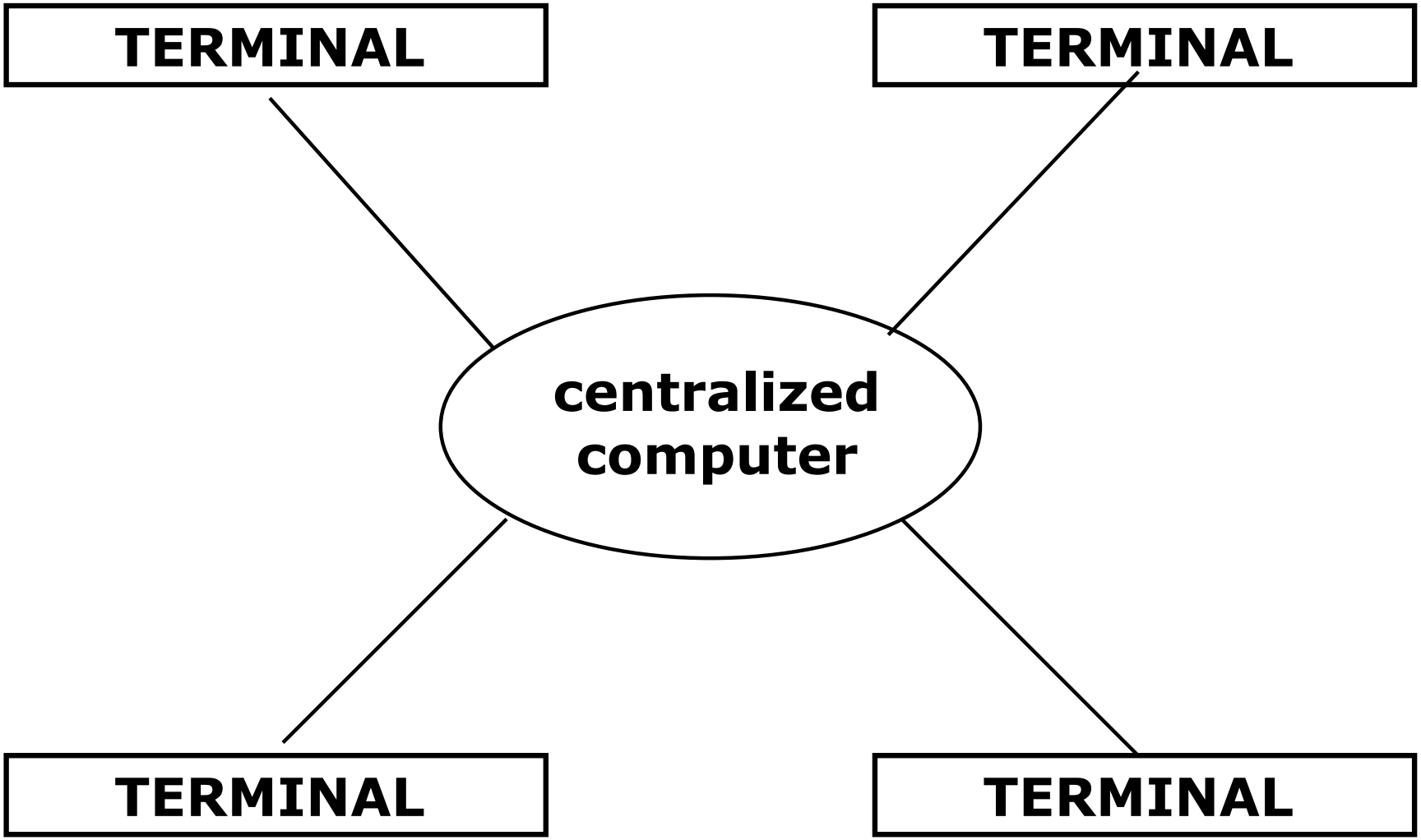
ARCHITETCTURES of COMPUTERS SYSTEMS

Centralized systems

Characteristics

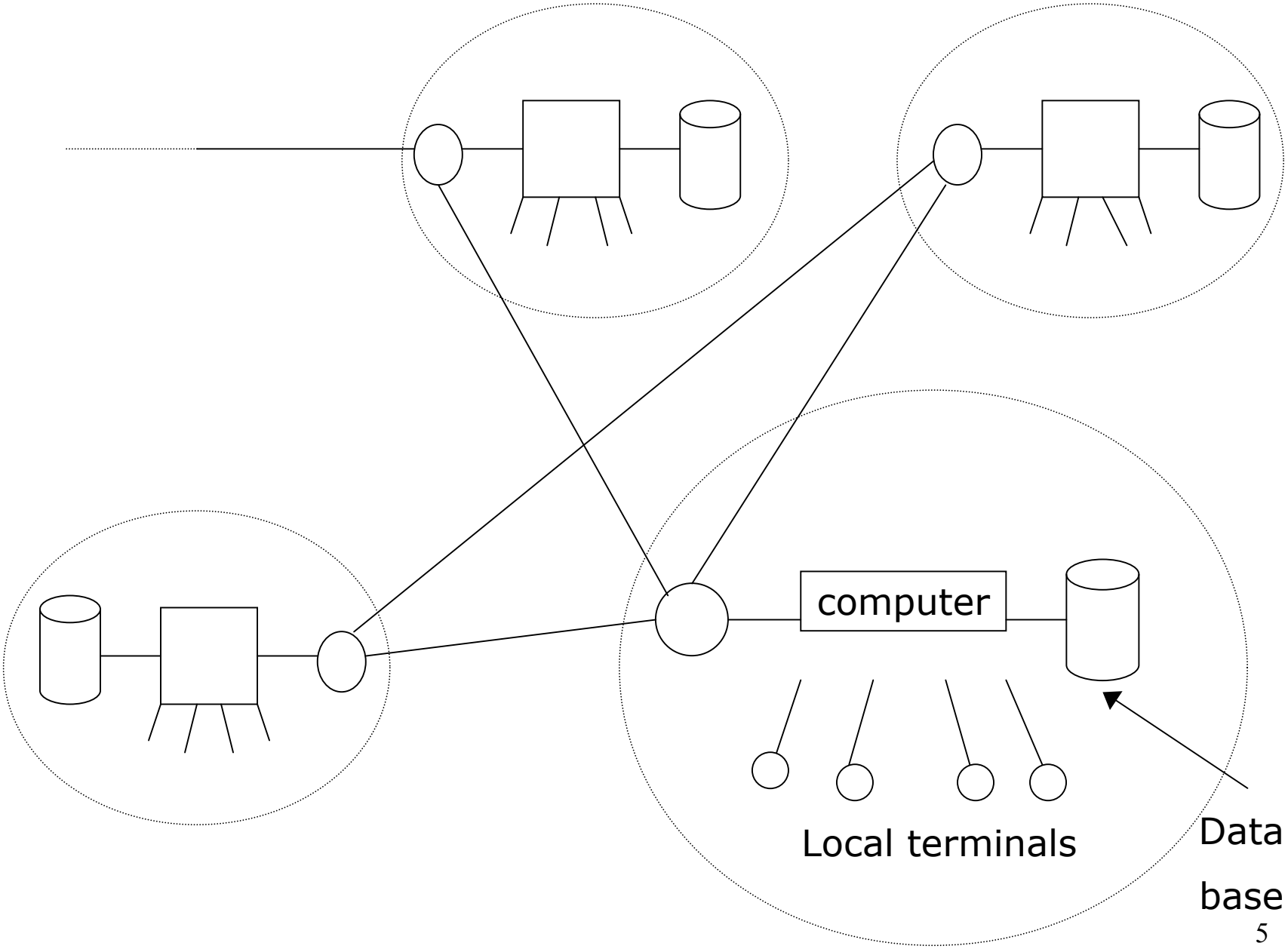
- Proprietary systems
- Limited knowledge of the computer thecnologies
- Backlog

Centralized solution



Distributed Systems

Set of independent computers connected by a communication network in order to execute different functions (administration, management, thecnical problems, logistic, production..) that are present in a complex organization.


computer
Local terminals
Data
base

Advances in thechnology

- personal computer
- local network (LAN)
- High speed networks (LAN, MAN, WAN)

Advantages of Distributed systems over Centralized ones

- servers offer a better price/performance than mainframes
- A large number of applications are distributed (they run on different machines)
- Reliability
- Incremental growth (computing power can be added in small increments)
- Data sharing (allow many users access to a common data base)
- Device sharing (allow many users to share expensive peripherals)
- Flexibility (spread the workload over the available machines in the most cost effective way)

Some definitions

- **Reliability:** the ability of a system to perform and maintain its functions in routine circumstances, as well as hostile or unexpected circumstances.

- **availability:** The ratio of (a) the total time a funcional unit is capable of being used during a given interval to (b) the length of the interval

- **security:** protection against non authorized access

- **fault tolerance:** the ability of a system to respond gracefully to an unexpected hardware or software failure

Open Systems

- Distributed systems consisting of eterogenous hardware and software components from different system vendors.
- Unlike a propritary solution a **open** distributed system can be realized by using components from the different vendors.
- Utilization of *standards (hardware, software)*

Open systems are computer systems that provide some combination of interoperability, portability, and open standard software.

The term was popularized in the early 1980s, mainly to describe systems based on UNIX, especially in contrast to the more complex mainframes and minicomputers in use at that time.

Unlike older **legacy system**, the newer generation of Unix systems featured standardized programming interfaces and peripheral interconnects; third party development of hardware and software was encouraged.

A significant departure from the norm of the time, which saw companies such as Amdhal and Hitachi going to court for the right to sell systems and peripherals that were compatible with IBM's mainframes.

Standard

- A **technical standard** is an established norm or requirement. It is usually a formal document that establishes uniform engineering or technical criteria, methods, processes and practices.

- **Standard de jure**: Hardware or software that is endorsed by a standards organization.

- **Standard de facto**: Hardware or software that is widely used, but not endorsed by a standards organization

- **The standard concept** is relative to the functional aspects of a component; the component may be realized in different ways.

CLIENT-SERVER MODEL

Coordination model in a distributed system.

It defines:

- Which process may begin the interaction
- Which process may answer
- How error conditions may be managed.

• Although an Internet system provides a basic communication service, the protocol software cannot initiate control with, or accept contact from, a remote computer.

• Of course, two application involved in a communication **cannot both** wait for a message to arrive. One application must actively initiate interaction while the other application passively waits.

• Most network applications use a form of communication known as **the client –server paradigm**. A server application waits passively for contact, while a client application initiates communication actively.

Process classification

- **Client process**, the process that requires a service
- **Server process**, the process that provides the required service
- The client requires a service, the server provides the service and makes available the results to the client

Client functions

In generally, client software:

- is an arbitrary application program that becames a client temporarily when remote access is needed, but also performs other computation locally.
- is invoked locally by a user, and executes only for one session
- runs locally on a user personal computer
- actively initiates contact with a server
- can access multiple services as needed, but actively contacts one remote server at a time.
- does not require special hardware or a sophisticated operating system

Server functions

- Is a special purpose, privileged program dedicated to providing one service, but can handle multiple remote clients at the same time.
- Run on a shared computer(i.e. not a user's personal computer).
- Wait passively for contact from arbitrary remote clients
- Accepts contact from arbitrary clients, but offers a single service
- Requires powerfull hardware and a sophisticated operating system

Application server

Kinds of client services:

- *mail server*
- *file server*
- *terminal server*
- *name server*
- *authentication server*
- *gateway server*
- *administration server*
- *……..*

A server must guarantee:

authentication: client identity verification

authorization: verification of the possibility for a client to access to a particular service

data security: garantee that specific data cannot be read and/or modified

SERVER E SERVER CLASS COMPUTERS

- Formally the term **server** refers to a program that waits passively for communication and not to the computer on wich it executes.

- However,when a computer is dedicated to running one or more server programs, the computer itself is sometimes called a **server**.

- **Server class computer** refers to a powerfull computer used to run server software.

- It may offer simultaneosly more services, each of them corresponding to a specific program application

• Information can flow in either or both directions between a client and a server.

• Typically, a client sends a request to a server, and the server returns a response to the client.

• In some cases, a client sends a series of requests and the server issues a series of responses (e.g., a data base server can allow to a user to look up more than one item at a time)

• In other cases, the server provides continous output without any request . As soon as the client contacts the server, the server begins sending data (e.g. local weather server might send continous weather reports with updated temparature and barometric pressure.)

Characteristics of a client-server architecture

- Client and server machines need different amount of hardware and software resources.
- Client and server machines may belong to different vendors.
- *orizzontal scalability* (increase of the client machines) and *vertical scalability* (migration to a more powerful server or to a multiserver solution)
- A client or server application interacts directly with a transport layer protocol to establish communication and to send or receive information.
- The transport protocol then uses lower layer protocols to send or receive individual messages. Thua, a computer needs a complete stack of protocols to run either a client or a server.

• A single server-class computer can offer multiple services at the same time; a separate server program is needed for each service.

•**Identifyng a particular service**
TCP uses 16-bit integer values (protocol port numbers) to identify services, and assign a unique port number to each service.

•A client specifies the protocol port number of the desired service when sending a request.

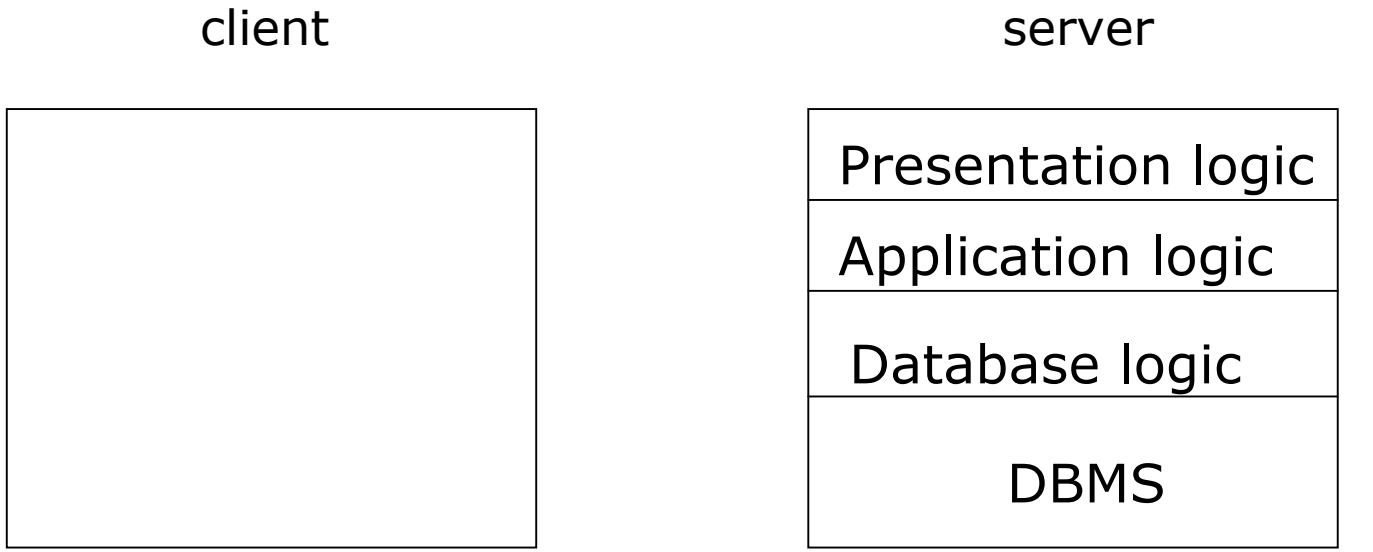
Classification of client server structures

• In a client/server application **three functions** are present:

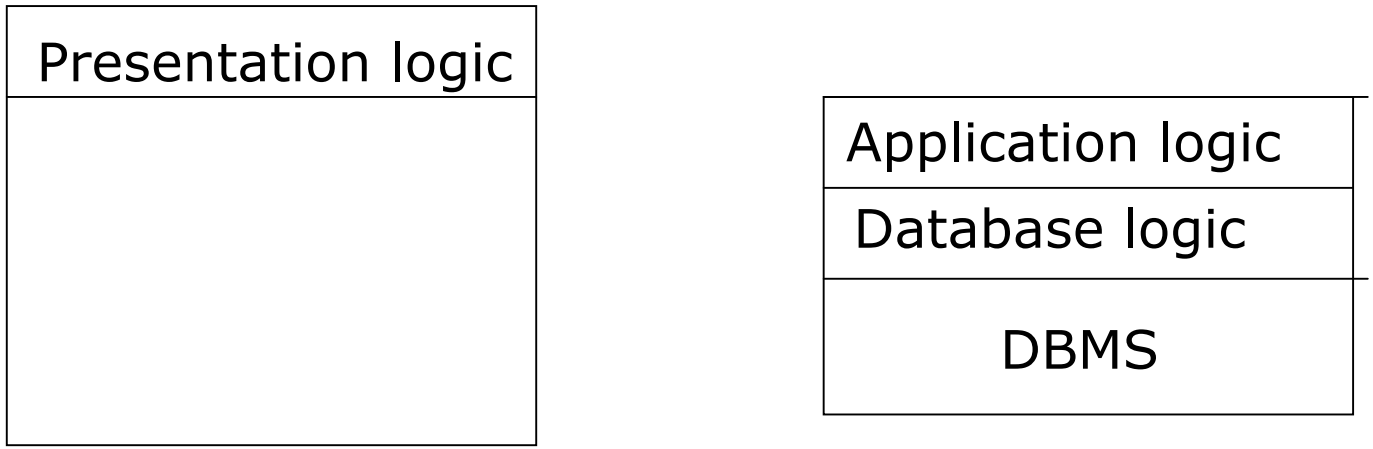
- user interface
- application programs
- data management

• Following the assignment of functions among client and server we have three possible types of structures:

- host-based processing
- server based processing
- cooperative processing
- client based processing



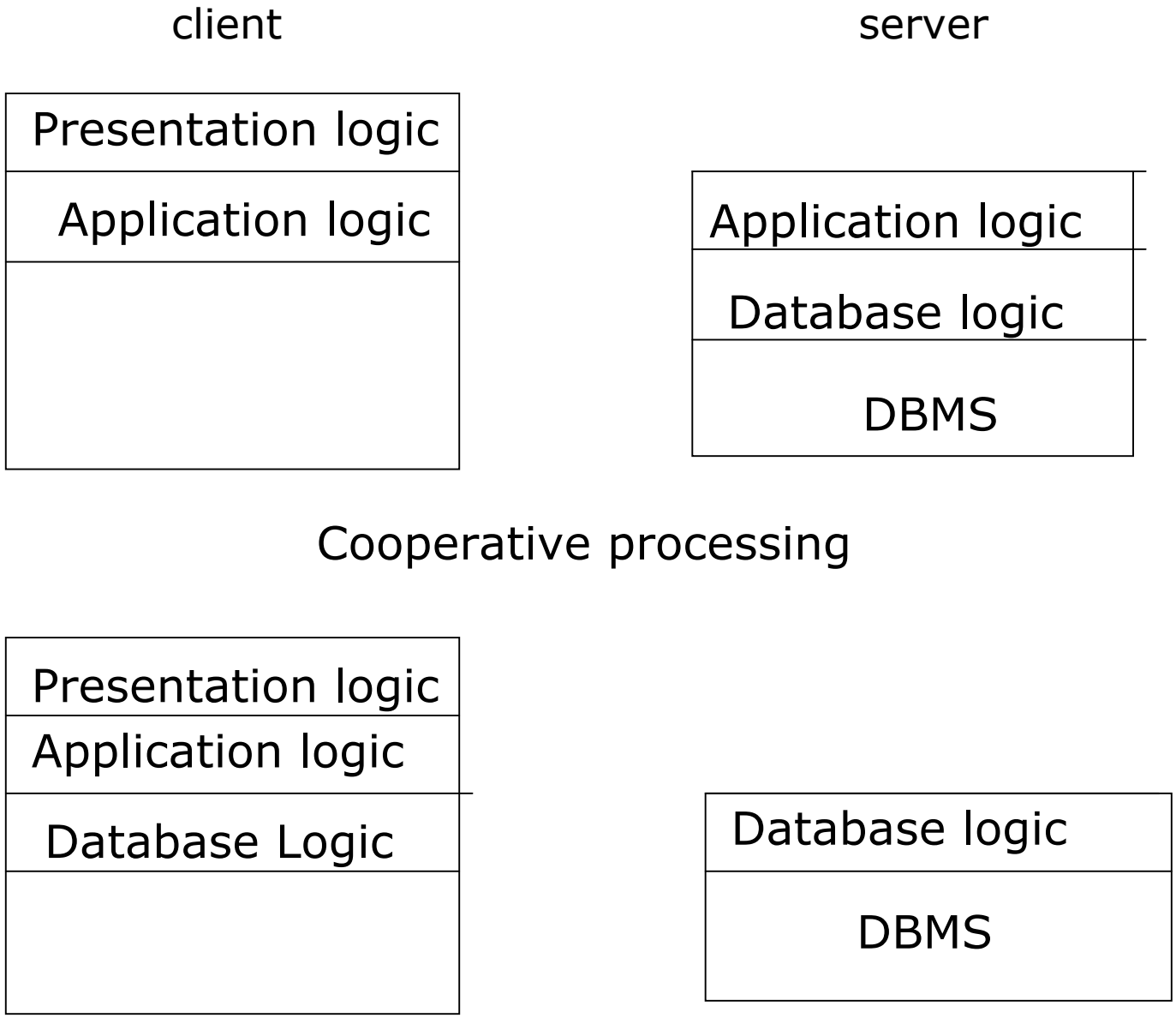
Host-based processing



Server-based processing

• **Host- based processing** is not true client server computing. It refers to the traditional mainframe environment in which all or virtually all of the processing is done on a central host. The user's station is generally limited to the role of a terminal emulator.

Server-based processing. The client is principally responsible for providing a **graphical user interface**, while virtually all the processing is done on the server.


client
server
Presentation logic
Application logic
Application logic
Database logic
DBMS
Cooperative processing
Presentation logic
Application logic
Database Logic
Database logic
DBMS
Client-based processing

Client-based processing. Virtually all application processing may be done at the client, with the exception of database logic functions that are best performed at the server.
This configuration is perahps the most common client server approach in current use.

Cooperative processing. That application processing is performed in an optimized fashion, taking advantage of the strenghts of both client and server machines and of distribution of data.

Three-level Client/Server Architecture

The traditional client/server architecture involves **two levels**, a client level and a server level.

A **three level** architecture is constituted by three types of machines: a user machine, a middle-level server and a back end server.

The user machine (client) is typically a thin client.

The middle-level server is the **application server**.

The back-end server is the data server.

Legacy system

- A “hold” system (mainframe) which has not been removed after the installation of a new system.

Dynamic server creation

• **Concurrency** is fundamental to the client server model of interaction because a concurrent server offers service to multiple clients at the same time, without requiring each client to wait for previous clients to finish.

•The server creates a new thread for each request that arrives.

•The main server thread waits for a request to arrive, then it creates a new service thread to handle each client

Complex Client-Server Interactions

• A client application **is not restricted to accessing a single service**. The client may contact a different server (perhaps on a different computer) for each service.

•. A server for a service can became a client of another server.

STATELESS Vs STATEFULL SERVERS

•**State information**: information that a server maintains about the interaction state with the clients.

•The information state provides an incremental answer to a new request.

•The information state can increase the system efficiency, but problem may arise in the case of duplicated messages , delays and messages arrived out of order.

- File server - remote access by clients . Two kinds of requests : Reading data from the file and writing data on the file.

- **Stateless server** – It does not mantain informations about the transactions. Each request must specify the name of the file, the position in the file from which to extract or to insert information, the number of bytes to insert or extract.

- **Statefull server** – The server mantains the state information about the files that accessed the server

Statefull server

The server sends to the client the identifier of the previously open file. The client utilizes this identifier for the following requests.

identifier	file name	current position
1	test.program.c	0
2	tcp.book.doc	456
3	dept.budget.txt	38
4	tetris.exe	128

Statefull server

- **Advantages**: higher efficiency, incremental operations, lower dimension messages.

- **Problems**: comand duplication, delays, messages out of order, server crash (open files are not correctly closed).

COOKIES

•Information records that the server sends to the client together with the service answer.

•A cookie contains
    user informations
    informations on tha visited pages

•The following requests of the client to the server require the transmission of the relative cookies.

Cookies

The server answer to a request of a client contains:

Set-cookie: user identification number created by the server web
Set-cookie: **1678453.**

The client inserts this information in a *cookie file* . The information includes the host name and the identification number of the client.

In the following requests the client inserts the previous information .

The server does not know the user name, but it is able to identify the client.

The web servers use cookies for the following objectives:

To eliminate the need to require user name and password in order to authenticate the client.

To store the user preferences

To keep track of the objects that a client is buying on the web site.