UNIT: 1

Introduction

Introduction

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

This definition has several important aspects.

- The first one is that a distributed system consists of components (i.e., computers) that are autonomous.
- A second aspect is that users (be they people or programs) think they are dealing with a single system. This means that one way or the other the autonomous components need to collaborate.

Distributed System

- **A distributed system is a collection of independent computers that appears to its users as a single coherent system** – *Andrew Tanenbaum*

- A distributed system is the one that prevents you from working because of the failure of a machine that you had never heard of – *Veríssimo & Rodrigues*

Why Distributed System? – Objectives of DS

1. **Cost**. Better price/performance as long as commodity hardware is used for the component computers.
2. **Reliability**. By having redundant components the impact of hardware and software faults on users can be reduced.
3. **Inherent distribution**. Naturally and physically distributed
4. **Transparency.** The end users can be concealed or hidden from the actual separation of the distributed system so that the user feels that everything is transparent to everyone.
5. **Scalability**. Resources such as processing and storage capacity can be increased significantly.
6. **Dependability.** The dependence of a system on another system can be achieved to solve a particular task jointly.
7. **Performance**. By using the combined processing and storage capacity of many nodes, performance levels can be reached that are beyond the range of centralized machines.
8. **Flexibility**. Easily can be added or removed a node

Distributed systems Principles

A distributed system consists of a collection of autonomous computers, connected through a network and distribution middleware, which enables computers to coordinate their activities and to share the resources of the system, so that users perceive the system as a single, integrated computing facility.

Centralized System Characteristics

- One component with non-autonomous parts
- Component shared by users all the time
- All resources accessible
- Software runs in a single process
- Single Point of control
- Single Point of failure

Distributed System Characteristics

- Multiple autonomous components
- Components are not shared by all users
- Resources may not be accessible
- Software runs in concurrent processes on different processors
- Multiple Points of control
- Multiple Points of failure

Examples of distributed systems and applications of distributed computing include the following:

❖telecommunication networks:

- telephone networks and cellular networks,
- computer networks such as the Internet,
- wireless sensor networks,
- routing algorithms;

Network Applications:

- World wide web and peer-to-peer networks,
- massively multiplayer online games and virtual reality communities,
- distributed databases and distributed database management systems,
- network file systems,
- distributed information processing systems such as banking systems and airline reservation systems;
- real-time process control:
- aircraft control systems,
- industrial control systems;
- parallel computation
- scientific computing

Common Characteristics

Certain common characteristics can be used to assess distributed systems

- Resource Sharing
- Openness
- Concurrency
- Scalability
- Fault Tolerance
- Transparency

Resource Sharing

 Ability to use any hardware, software or data anywhere in the system.

 Resource manager controls access, provides naming scheme and controls concurrency.

 Resource sharing model (e.g. client/server or object-based) describing how resources are provided, they are used and provider and user interact with each other.

Openness

 Openness is concerned with extensions and improvements of distributed systems.

 Detailed interfaces of components need to be published.

 New components have to be integrated with existing components.

 Differences in data representation of interface types on different processors (of different vendors) have to be resolved.

Concurrency

Components in distributed systems are executed in concurrent processes.

 Components access and update shared resources (e.g. variables, databases, device drivers).

 Integrity of the system may be violated if concurrent updates are not coordinated.

- Lost updates
- Inconsistent analysis

Scalability

- Adaption of distributed systems to accommodate more users respond faster (this is the hard one)
- Usually done by adding more and/or faster processors.
- Components should not need to be changed when scale of a system increases.
- Design components to be scalable

Fault Tolerance

Hardware, software and networks fail!

 Distributed systems must maintain availability even at low levels of hardware/software/network reliability.

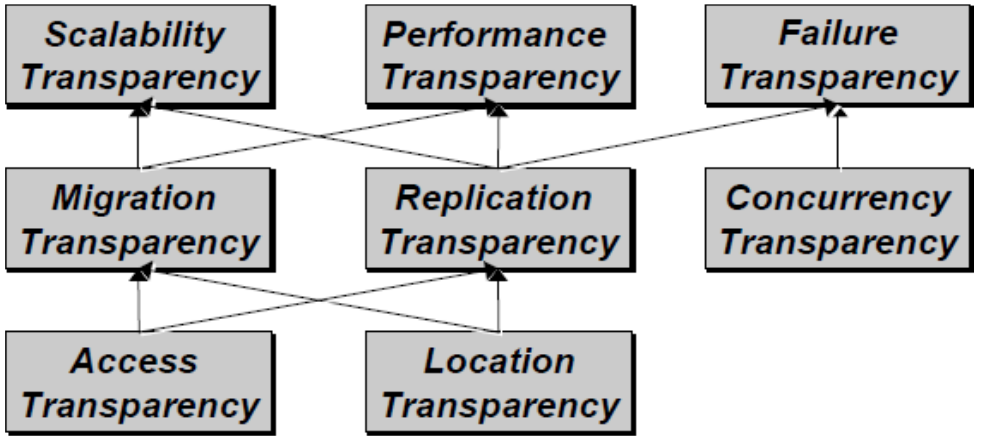
 Fault tolerance is achieved by

- recovery
- redundancy

Transparency

Distributed systems should be perceived by users and application programmers as a whole rather than as a collection of cooperating components.

- Transparency has different dimensions that were identified by ANSA(Advanced **Networking** and Server Administration).
- These represent various properties that distributed systems should have.



Access Transparency

Enables local and remote information objects to be accessed using identical operations.

- Example: File system operations in NFS.
- Example: Navigation in the Web.
- Example: SQL Queries

Location Transparency

Enables information objects to be accessed without knowledge of their location.

- Example: File system operations in NFS
- Example: Pages in the Web
- Example: Tables in distributed databases

Concurrency Transparency

Enables several processes to operate concurrently using shared information objects without interference between them.

- Example: NFS
- Example: Automatic teller machine network
- Example: Database management system

Replication Transparency

Enables multiple instances of information objects to be used to increase reliability and performance without knowledge of the replicas by users or application programs

- Example: Distributed DBMS
- Example: Mirroring Web Pages.

Failure Transparency

- Enables the concealment(hidden) of faults
- Allows users and applications to complete their tasks despite the failure of other components.
- Example: Database Management System

Migration Transparency

Allows the movement of information objects within a system without affecting the operations of users or application programs

- Example: NFS
- Example: Web Pages

Performance Transparency

Allows the system to be reconfigured to improve performance as loads vary.

- Example: Distributed make.

Scaling Transparency

Allows the system and applications to expand in scale without change to the system structure or the application algorithms.

- Example: World-Wide-Web
- Example: Distributed Database

Goals

- A distributed system should make resources easily accessible; it should reasonably hide the fact that resources are distributed across a network; it should be open; and it should be scalable.

Making Resources Accessible

- The main goal of a distributed system is to make it easy for the users (and applications) to access remote resources, and to share them in a controlled and efficient way. Resources can be just about anything, but typical examples include things like printers, computers, storage facilities, data, files, Web pages, and networks, to name just a few.

Distribution Transparency

- An important goal of a distributed system is to hide the fact that its processes and resources are physically distributed across multiple computers. 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.

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 is replicated
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource

- **Openness**
- Another important goal of distributed systems is openness. An **open distributed** system is a system that offers services according to standard rules that describe the syntax and semantics of those services. For example, in computer networks, standard rules govern the format, contents, and meaning of messages sent and received. Such rules are formalized in protocols.
- Benefits of Open Distributed Systems
- Interoperability: components written by different programmers can easily work together
- Portability: applications can be easily ported between different distributed systems that implement the same interfaces
- Extensibility: new services can be easily added and old services can be easily re-implemented

- **Scalability**
- Worldwide connectivity through the Internet is rapidly becoming as common as being able to send a postcard to anyone anywhere around the world. With this in mind, scalability is one of the most important design goals for developers of distributed systems.
- It describes the ability of the system to dynamically adjust its own computing performance by changing available computing resources and scheduling methods. Scalability is divided into two aspects: hardware and software. Scalability in hardware refers to changing workloads by changing hardware resources, such as changing the number of processors, memory, and hard disk capacity. The scalability of software is to meet the changing workload by changing the scheduling method and the degree of parallelism.

Types of Distributed System

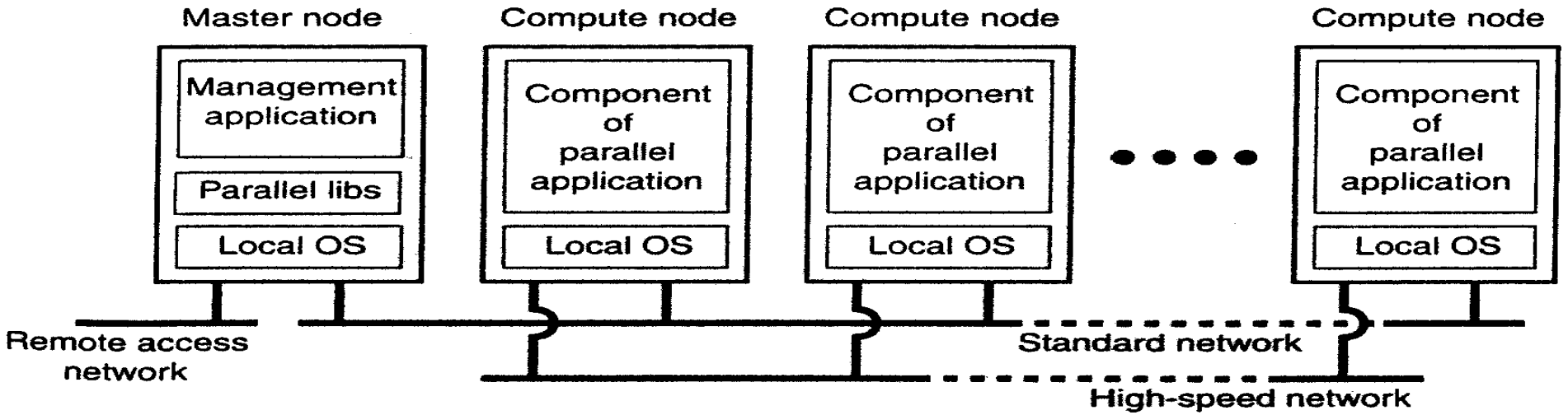
Distributed Computing Systems

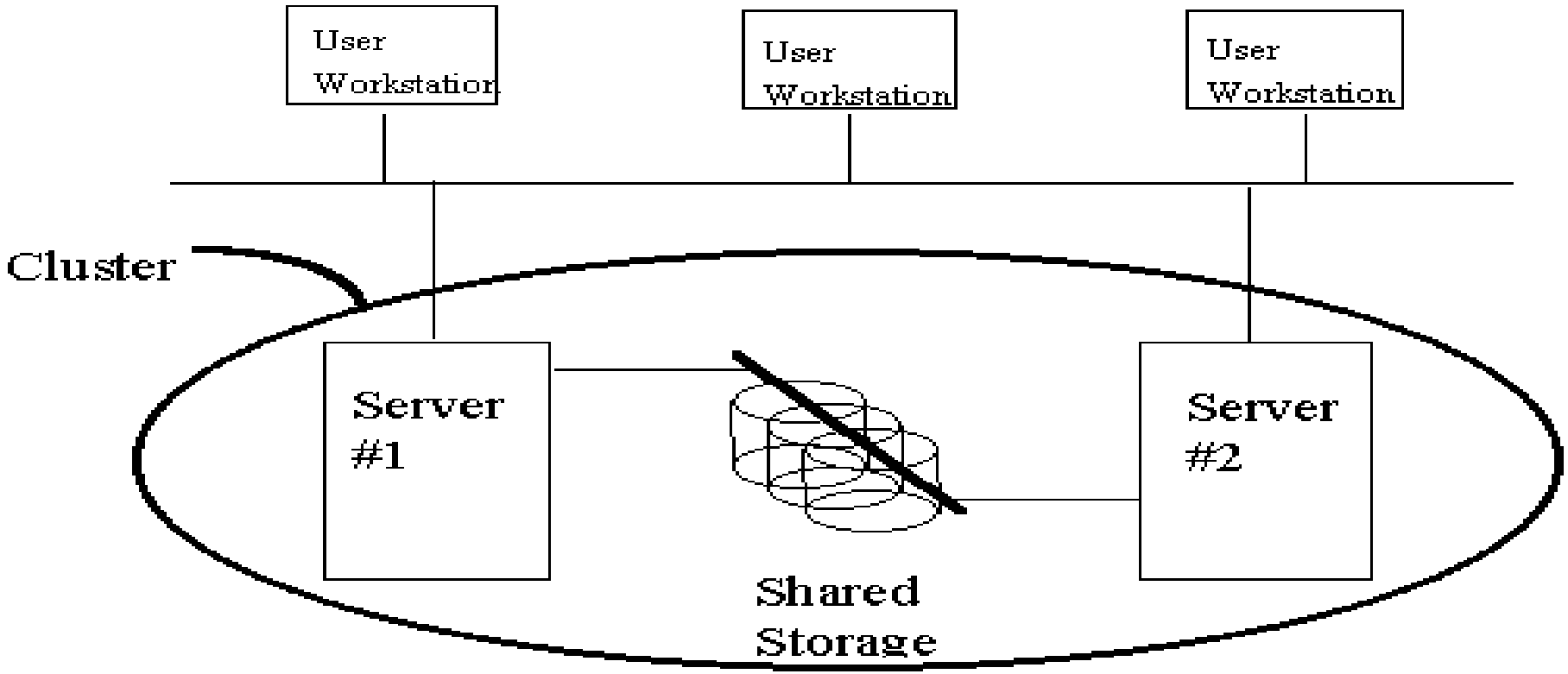
- An important class of distributed systems is the one used for high-performance computing tasks.
- Roughly speaking, one can make a distinction between two subgroups.
    1. Cluster Computing
    2. Grid Computing

Cluster computing system

- It a type of computing in which a group of computers are linked together so that they can act as Single Entity.
- In cluster computing the underlying hardware consists of a collection of similar workstations or PCs, closely connected by means of a highspeed local-area network. In addition, each node runs the same operating system.
- In virtually all cases, cluster computing is used for parallel programming in which a single (compute intensive) program is run in parallel on multiple machines.
- Clusters are typically used for High Availability(HA) for greater reliability or High Performance Computing(HPC) to provide greater power than a single computer can provide.

- Cluster computing systems became popular when the price/performance ratio of personal computers and workstations improved. At a certain point, it became financially and technically attractive to build a supercomputer using off-the-shelf technology by simply hooking up a collection of relatively simple computers in a high-speed network.
- For example, the internet, search engine, Google uses cluster computing to provide reliable and efficient internet search services.



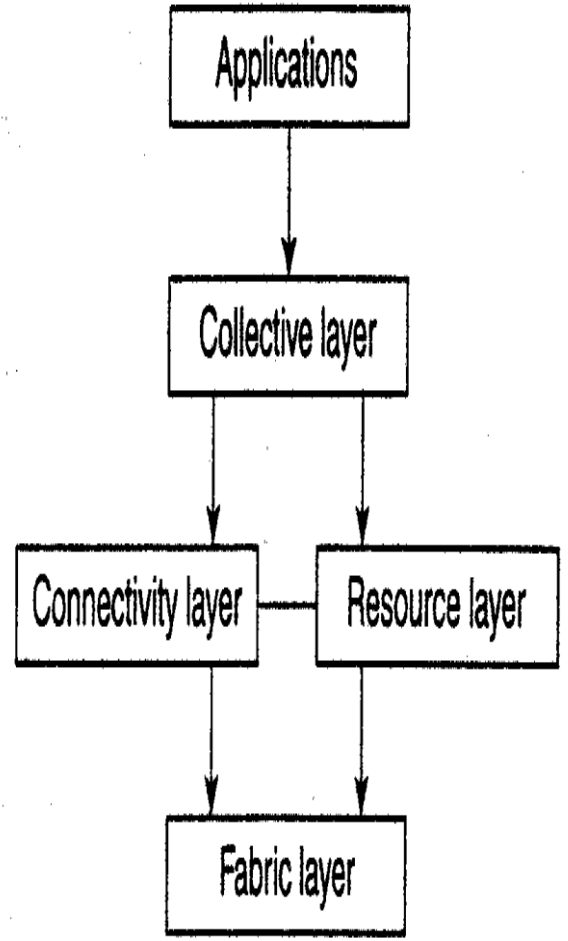

User Workstation
User Workstation
User Workstation
Cluster
Server #1
Server #2
Shared Storage

Grid Computing Systems

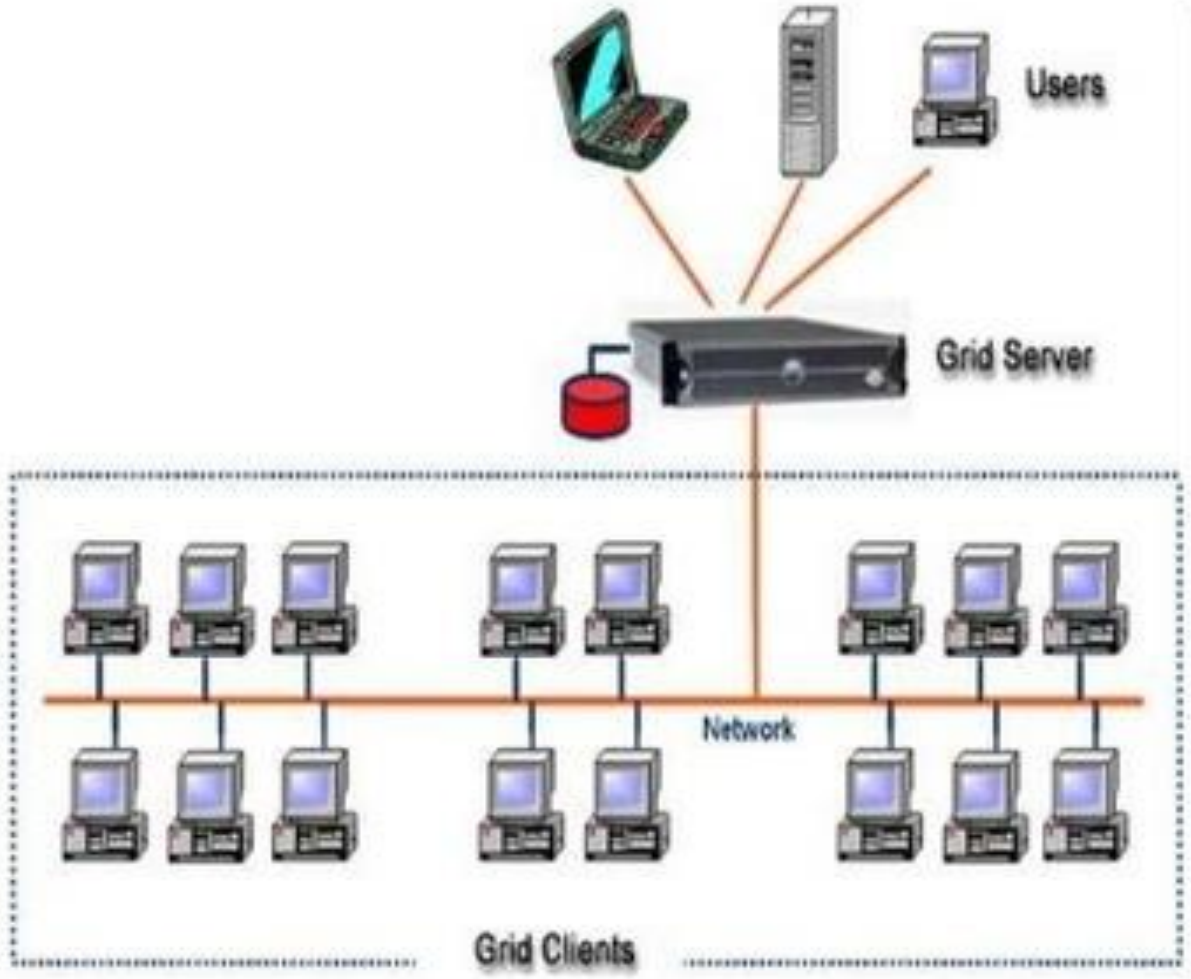
- It is a collection of computer resources from multiple locations to reach a common goal.
- In contrast, grid computing systems have a high degree of heterogeneity: no assumptions are made concerning hardware, operating systems, networks, administrative domains, security policies, etc.
- A key issue in a grid computing system is that resources from different organizations are brought together to allow the collaboration of a group of people or institutions. Such a collaboration is realized in the form of a virtual organization. The people belonging to the same virtual organization have access rights to the resources that are provided to that organization.

The architecture consists of four layers.

- The lowest *fabric layer* provides interfaces to local resources at a specific site.
- The *connectivity layer* consists of communication protocols for supporting grid transactions that span the usage of multiple resources. For example, protocols are needed to transfer data between resources, or to simply access a resource from a remote location.
- The *resource layer* is responsible for managing a single resource. It uses the functions provided by the connectivity layer and calls directly the interfaces made available by the fabric layer
- The next layer in the hierarchy is the *collective layer.* It deals with handling access to multiple resources and typically consists of services for resource discovery, allocation and scheduling of tasks onto multiple resources, data replication, and so on.
- Finally, the *application layer* consists of the applications that operate within a virtual organization and which make use of the grid computing environment.



How Grid computing works ?



In general, a grid computing system requires:

- **At least one computer, usually a server, which handles all the administrative duties for the System**
- **A network of computers running special grid computing network software.**
- **A collection of computer software called middleware**

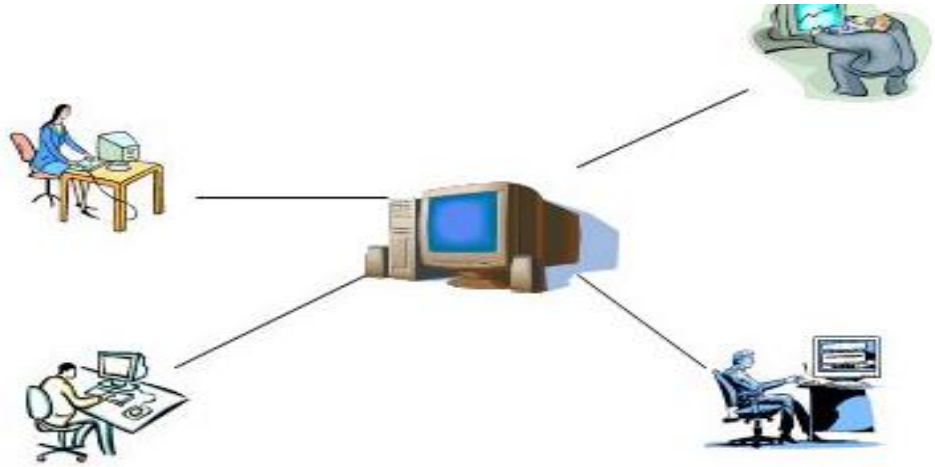
Grid example

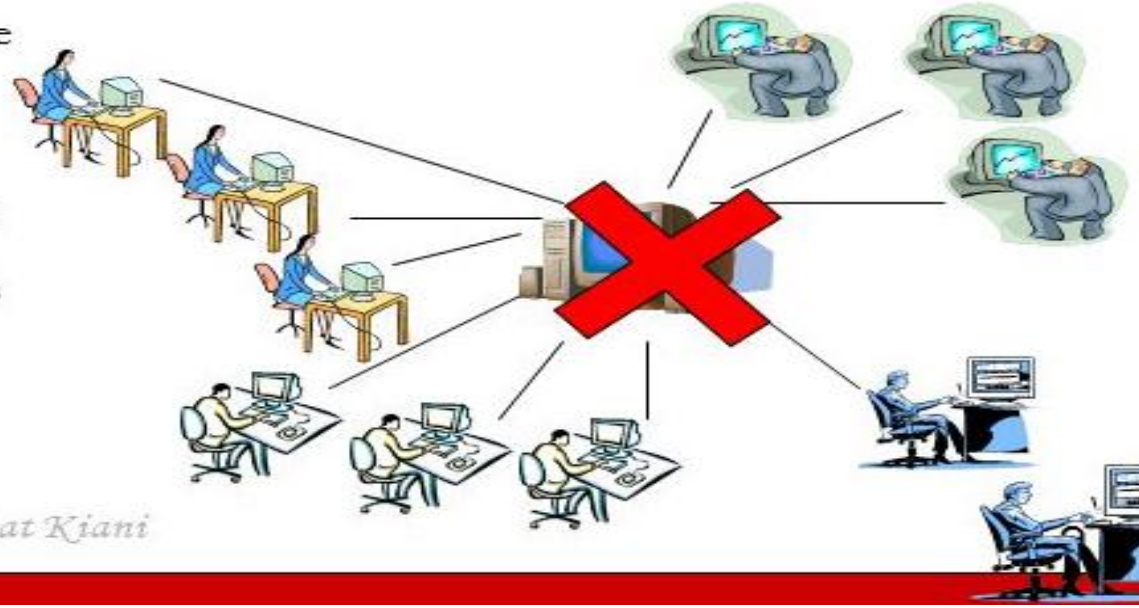
Grid computing for online gaming.

- It illustrates how processing resources can be used on an as-needed basis as the demand for power grows.

The popularity of multi player online games has grown as users buy subscriptions to play online games with other users from around the world.

- The challenge with traditional multi player games is the limitation placed on the number of concurrent players per server. The following scenario can occur:
- A player buys a game and a subscription to play online with others.
- There are thousands other players who buy the same game and want to play online.
- The server handling the game is capable of handling a limited number of players.
- If the power capacity is overcome, the solution has been to acquire additional servers and run additional copies of the gaming engine, but this must be done considering the application capability, that is, the capability of the application runs in a parallel environment.





Saad Liaquat Kiani

Distributed Information Systems

It is a computer service that runs at a single central location is more likely to become unavailable than a service distributed to many sites. Distributed Information System can be explained further as:

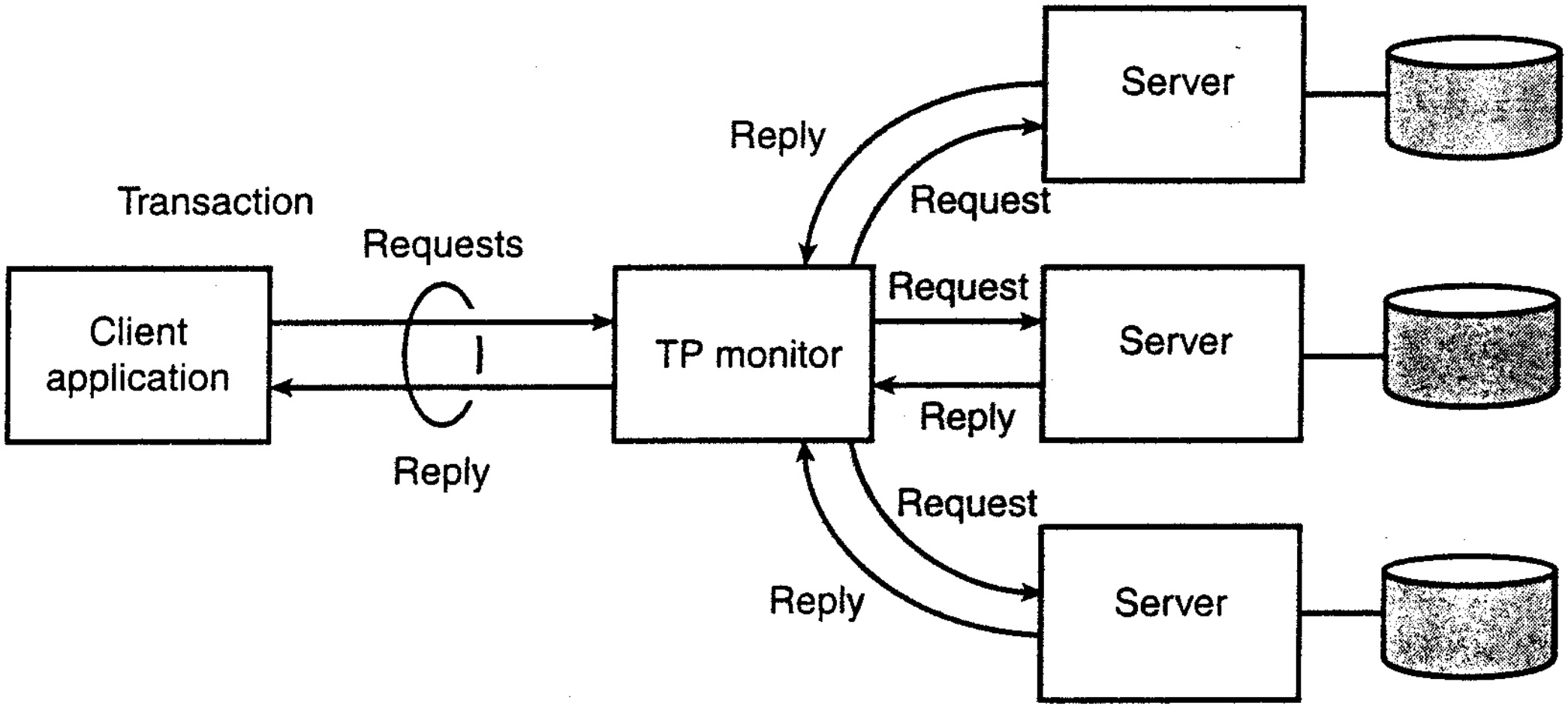
Transaction processing system

Provides a highly structured client-server approach for database applications. More specifically, transactions are:

1. Atomic: To the outside world, the transaction happens indivisibly.
2. Consistent: The transaction does not violate system invariants.
3. Isolated: Concurrent transactions do not interfere with each other.
4. Durable: Once a transaction commits, the changes are permanent.

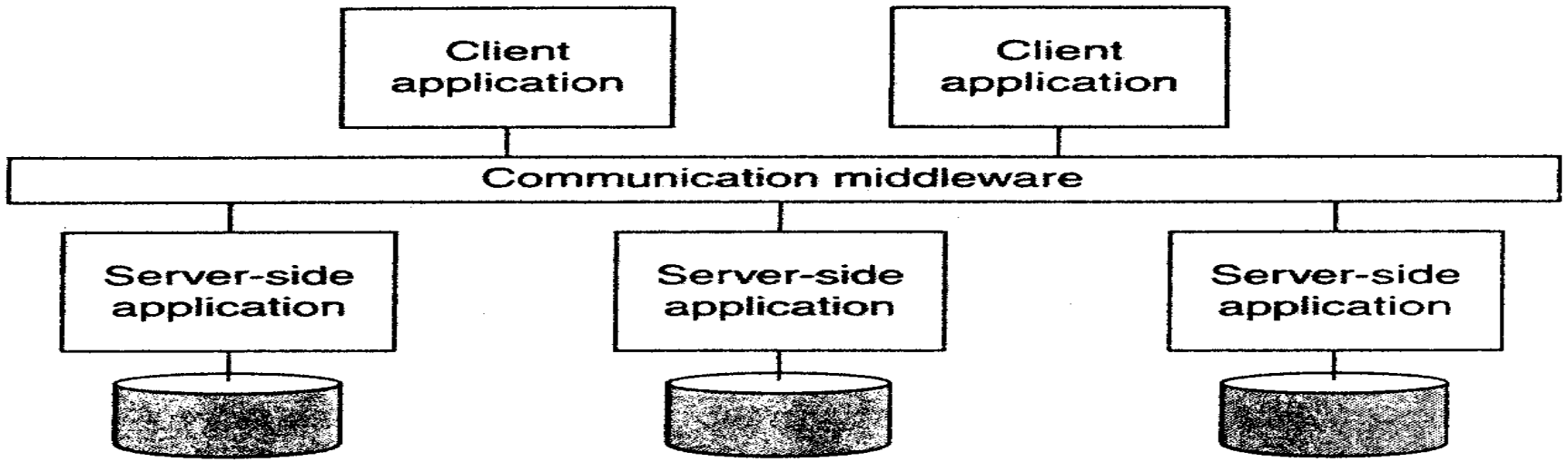
- Transaction processing may be centralized(traditional client/server system) or distributed.
- A distributed database is one in which the data storage is distributed – connected to separate processors.

Nested transaction is a transaction within another transaction(a sub-transaction). For example: a transaction may ask for two things (airlines reservation info + hotel info) which would spawn two nested transactions.


Server
Reply
Request
Transaction
Requests
Client application
TP monitor
Request
Server
Reply
Reply
Request
Reply
Server

Enterprise Application Integration:

- In particular, application components should be able to communicate directly with each other and not merely by means of the request/reply behavior that was supported by transaction processing systems.
- The main idea that existing applications could directly exchange information is shown below



- **Middleware** in the context of distributed applications is software that provides services beyond those provided by the operating system to enable the various components of a distributed system to communicate and manage data.
- Middleware supports and simplifies complex distributed applications.
- It includes web servers, application servers, messaging and similar tools that support application development and delivery.

Distributed Pervasive system

- The devices in these, what we refer to as **distributed** pervasive systems, are often characterized by being small, battery-powered, mobile, and having only a wireless connection, although not all these characteristics apply to all devices.
- As its name suggests, a distributed pervasive system is part of our surroundings (and as such, is generally inherently distributed).

Some concrete examples of pervasive systems.

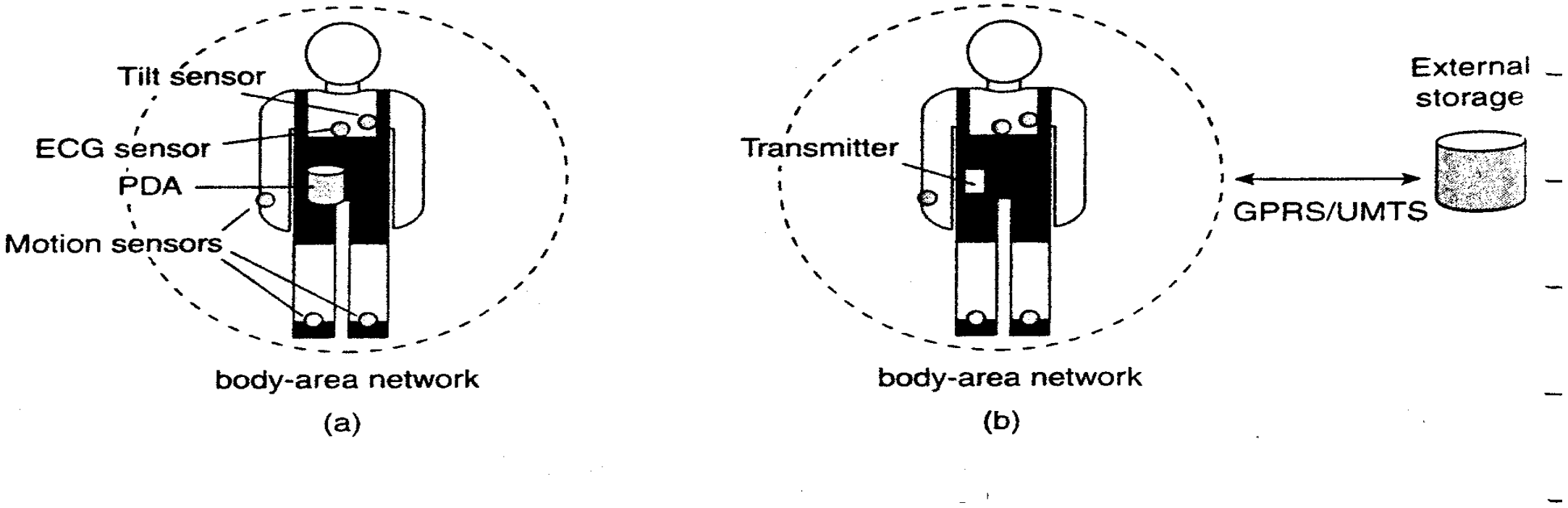
1. Home system
2. Electronic Health Care Systems
3. Sensor network

Home Systems

- An increasingly popular type of pervasive system, but which may perhaps be the least constrained, are systems built around home networks.
- These systems generally consist of one or more personal computers, but more importantly integrate typical consumer electronics such as TVs, audio and video equipment. Gaming devices, (smart) phones, PDAs, and other personal wearables into a single system.
- In addition, we can expect that all kinds of devices such as kitchen appliances, surveillance cameras, clocks, controllers for lighting, and so on, will all be hooked up into a single distributed system.

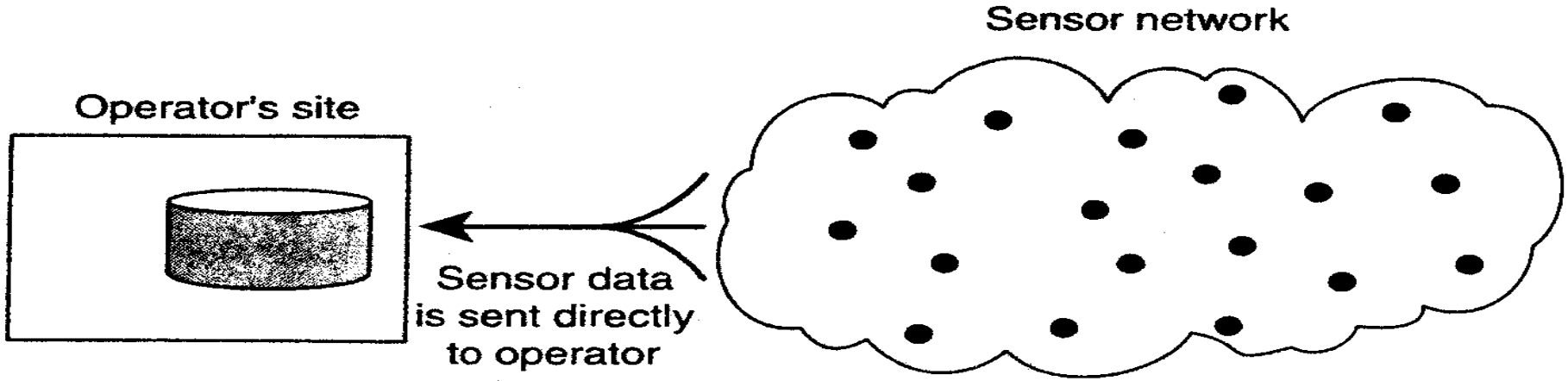
Electronic Health Care Systems

- Another important and upcoming class of pervasive systems are those related to (personal) electronic health care.
- With the increasing cost of medical treatment, new devices are being developed to monitor the well-being of individuals and to automatically contact physicians when needed.
- In many of these systems, a major goal is to prevent people from being hospitalized.

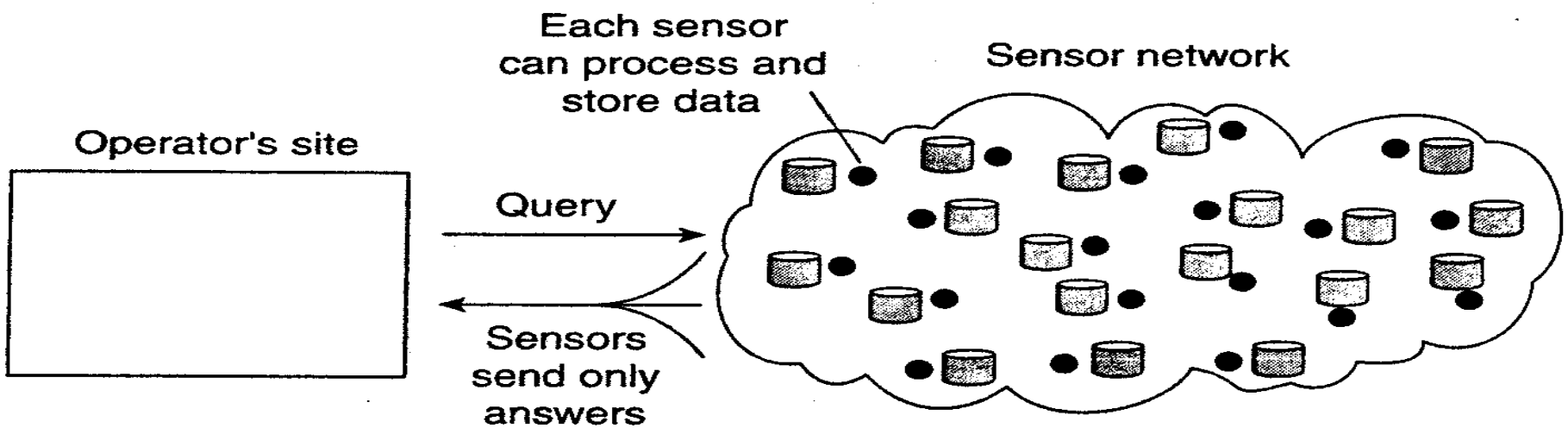

Tilt sensor
ECG sensor
PDA
Motion sensors
body-area network
(a)
Transmitter
External storage
GPRS/UMTS
body-area network
(b)

Sensor Networks

- Our last example of pervasive systems is sensor networks.
- These networks in many cases form part of the enabling technology for pervasiveness and we see that many solutions for sensor networks return in pervasive applications. What makes sensor networks interesting from a distributed system's perspective is that in virtually all cases they are used for processing information.


Sensor network
Operator's site
Sensor data
is sent directly
to operator

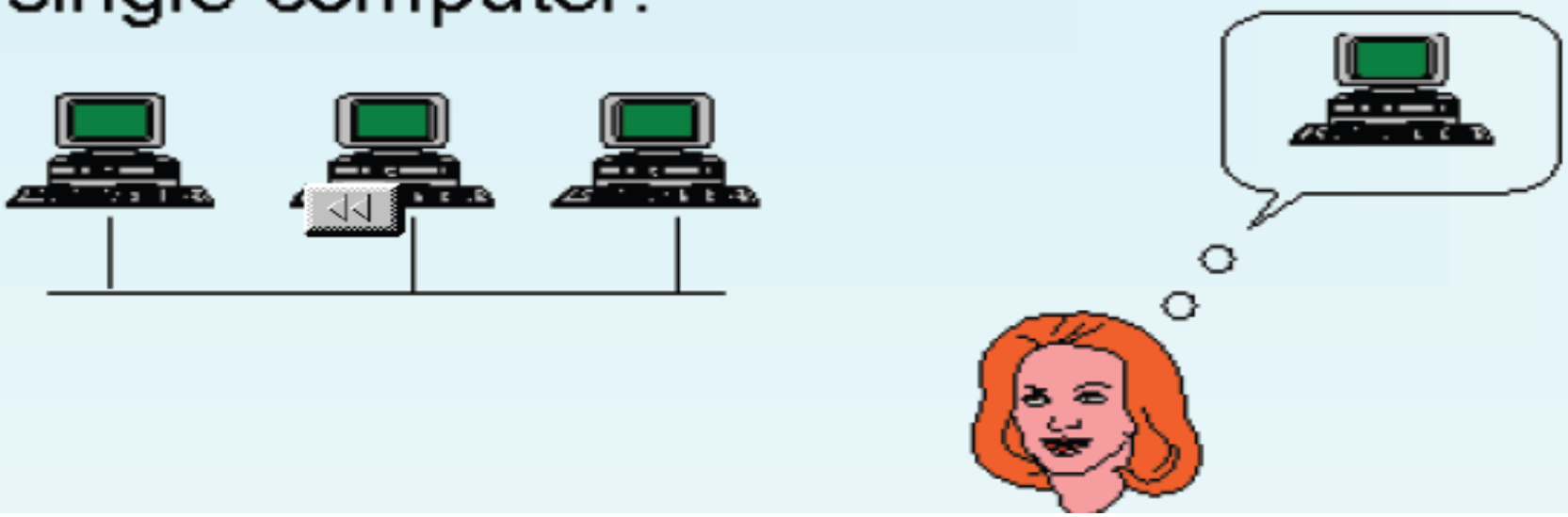
(a)


Each sensor
can process and
store data
Sensor network
Operator's site
Query
Sensors
send only
answers

(b)

Distributed Operating System

Goal: An operating system which manages a collection of independent computers and makes them appear to the users of the system as a single computer.



Distributed Operating System

- A **distributed operating system** is the logical aggregation of operating system software over a collection of independent, networked, communicating, and physically separate computational nodes (a distributed system)

- Individual nodes each hold a specific software subset of the global aggregate operating system.

- Each subset is a composite of two distinct service provisioners.
  - *Kernel* - that directly controls that node's hardware.
  - *System management components* - that coordinate the node's individual and collaborative activities. These components abstract microkernel functions and support user applications.
- The collection of kernel and system management components work together to make the distributed systems to appear as a single system image.

DOS: characteristics (1)

- **Distributed Operating Systems**
  - **Allows a multiprocessor or multicomputer network resources to be integrated as a single system image**
  - **Hide and manage hardware and software resources**
  - **provides transparency support**
  - **provide heterogeneity support**
  - **control network in most effective way**
  - **consists of low level commands + local operating systems + distributed features**
  - **Inter-process communication (IPC)**

DOS: characteristics (2)

- **remote file and device access**
- **global addressing and naming**
- **synchronization and deadlock avoidance**
- **resource allocation and protection**
- **global resource sharing**
- **communication security**
- **Some examples: Windows server 2003,Windows server 2008,Windows server 2012,Ubuntu,Linux (Apache Server),AMOEBA,etc.**

OS vs DOS

OS

- Manages the local resources to serve the local user

DOS

- Manages the global resources (systems connected in a distributed network) and integrates them in such a way that all the systems acts as a single coherent system.

Multiprocessors

- **Multiprocessor** is the use of two or more central processing units(CPUs) within a single computer system – ***MIMD***
- The term also refers to the ability of a system to support more than one processor and/or the ability to allocate tasks between them
- The term **multiprocessing** may be sometimes used an analogous to *multitasking*, or *multiprogramming*

Multiprocessing Systems

- Generally accepted definition of a multiprocessing/multicomputer system:
  - Multiple processors, each with its own CPU and Memory.
  - Interconnection hardware
  - Processors fail independently
  - There exists a shared state
  - Appears to users as single system

Multicomputer

- A *multicomputer consists of separate computing nodes connected to each other over a network*

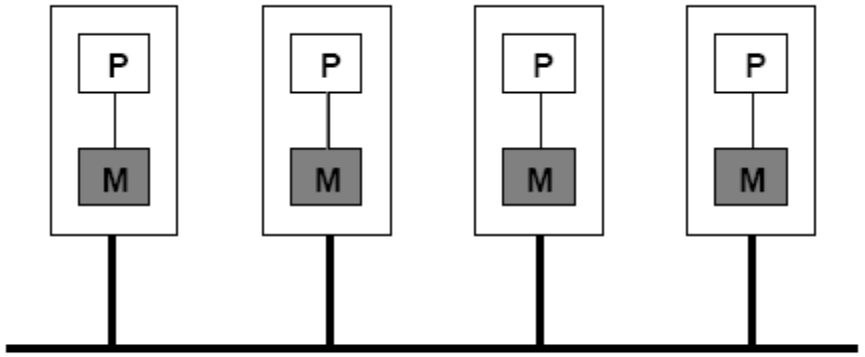


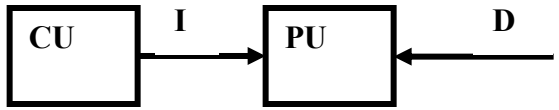
Figure 1: A multicomputer.

- Multicomputer generally differ from each other in three ways:
  - Node resources: processor, memory
  - Network connection: star, ring etc.
  - Homogeneity: same nodes with same physical architecture (processor, system bus, memory)

Flynn's Classification – Processor architectures

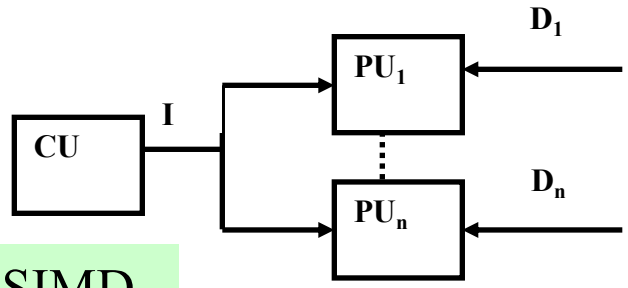
- Flynn, 1966+1972 classification of computer systems in terms of instruction and data stream organizations
- Based on Von-Neumann model (separate processor and memory units)
- 4 machine organizations
  - SISD - Single Instruction, Single Data
  - SIMD - Single Instruction, Multiple Data
  - MISD - Multiple Instruction, Single Data
  - MIMD - Multiple Instruction, Multiple Data

Flynn Architectures (1)



SISD

Serial Processor (8080,M6800,i8080)

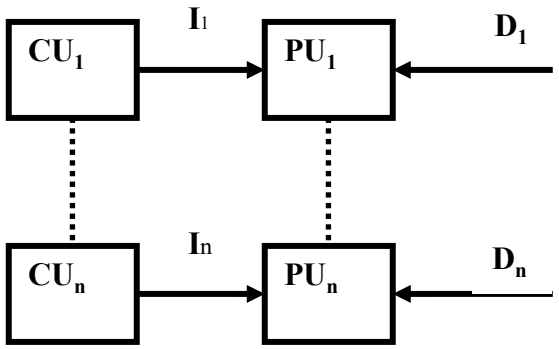


SIMD

Array Processor (Supercomputers, Thinking Machine,etc.)

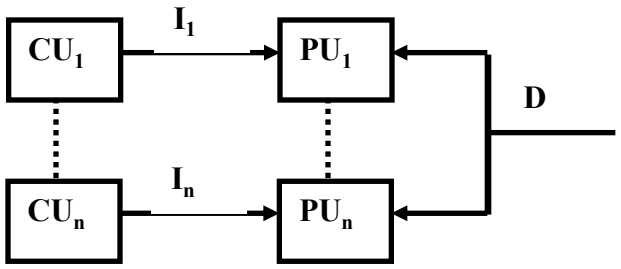
CU – control unit
PU – processor unit
I – instruction stream
D – data stream

Flynn Architectures (2)



MIMD

Multiprocesor (All Distributed systems)



MISD

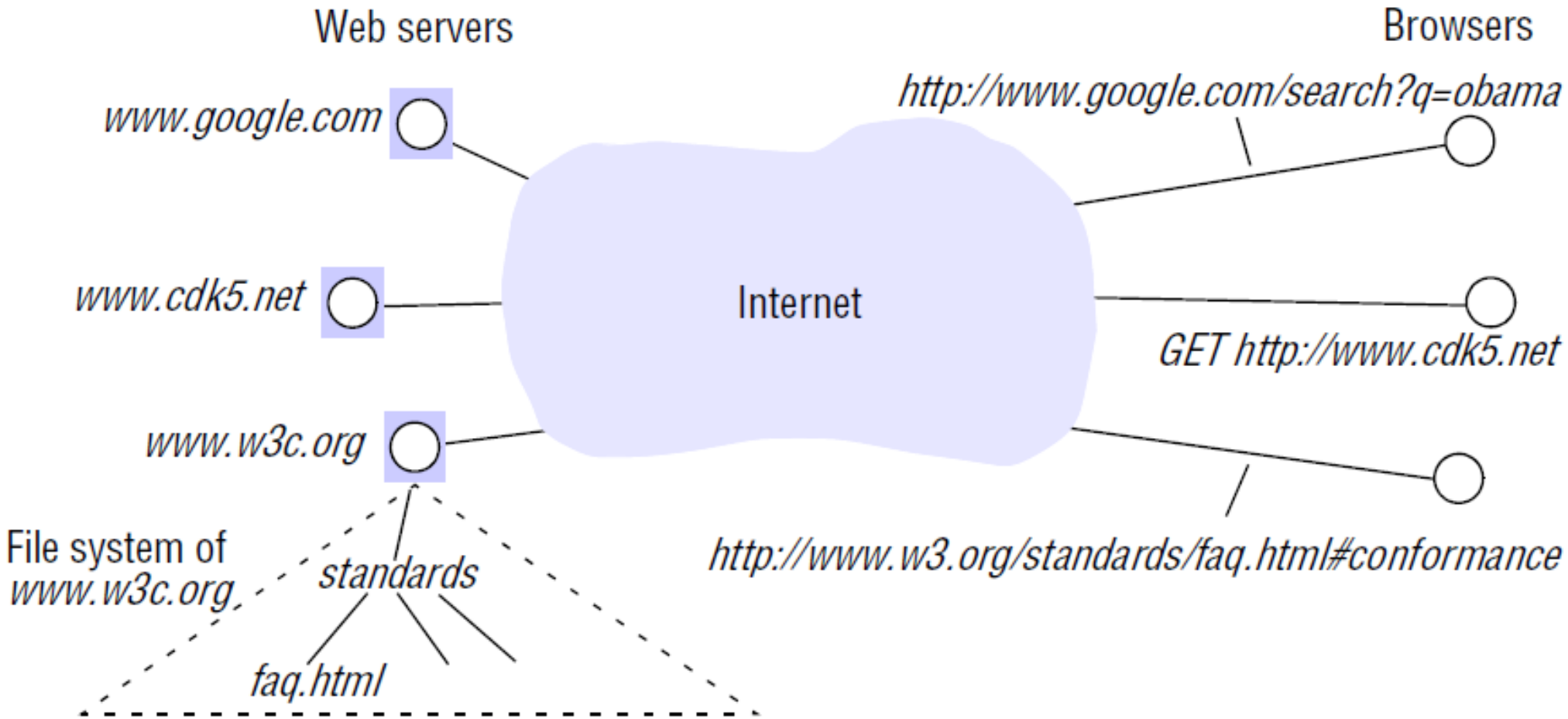
No real examples
- possibly some pipeline architectures (Cray-1, CDC cyber 205, PIC18)

CASE STUDY: The World Wide Web

- The World Wide Web [www.w3.org I, Berners-Lee 1991] is an evolving system for publishing and accessing resources and services across the Internet. Through commonly available web browsers, users retrieve and view documents of many types, listen to audio streams and view video streams, and interact with an unlimited set of services.
- The Web began life at the European centre for nuclear research (CERN), Switzerland, in 1989 as a vehicle for exchanging documents between a community of physicists connected by the Internet [Berners-Lee 1999]. A key feature of the Web is that it provides a *hypertext* structure among the documents that it stores, reflecting the users' requirement to organize their knowledge. This means that documents contain *links* (or *hyperlinks*) – references to other documents and resources that are also stored in the Web.
- The Web is an *open* system: it can be extended and implemented in new ways without disturbing its existing functionality. First, its operation is based on communication standards and document or content standards that are freely published and widely implemented. Second, the Web is open with respect to the types of resource that can be published and shared on it.
- WWW is the huge collection of pages of information linked to each other around one globe.
- Web page is the collection of text, images, video, audio, animation and hyperlink.

- The Web has moved beyond these simple data resources to encompass services, such as electronic purchasing of goods. It has evolved without changing its basic architecture. The Web is based on three main standard technological components:
- the HyperText Markup Language (HTML), a language for specifying the contents and layout of pages as they are displayed by web browsers;
- Uniform Resource Locators (URLs), also known as Uniform Resource Identifiers (URIs), which identify documents and other resources stored as part of the Web;
- a client-server system architecture, with standard rules for interaction (the HyperText Transfer Protocol – HTTP) by which browsers and other clients fetch documents and other resources from web servers. Figure below shows some web servers, and browsers making requests to them. It is an important feature that users may locate and manage their own web servers anywhere on the Internet.

Web servers and web browsers



- **HTML** • The HyperText Markup Language [www.w3.org II] is used to specify the text and images that make up the contents of a web page, and to specify how they are laid out and formatted for presentation to the user. A web page contains such structured items as headings, paragraphs, tables and images. HTML is also used to specify links and which resources are associated with them.
- Users may produce HTML by hand, using a standard text editor, but they more commonly use an HTML-aware 'wysiwyg' editor that generates HTML from a layout that they create graphically. A typical piece of HTML text follows:
- *<IMG SRC = "http://www.cdk5.net/WebExample/Images/earth.jpg"> 1*
- *<P> 2*
- *Welcome to Earth! Visitors may also be interested in taking a look at the 3*
- *<A HREF = "http://www.cdk5.net/WebExample/moon.html">Moon</A>. 4*
- *</P> 5*

- **URLs** • The purpose of a Uniform Resource Locator [www.w3.org III] is to identify a resource. Indeed, the term used in web architecture documents is Uniform Resource *Identifier* (URI), but in this book the better-known term URL will be used when no confusion can arise. Browsers examine URLs in order to access the corresponding resources.
- Sometimes the user types a URL into the browser. More commonly, the browser looks up the corresponding URL when the user clicks on a link or selects one of their 'bookmarks'; or when the browser fetches a resource embedded in a web page, such as an image.
- URL is the global address of web documents on www.
- URL are unique in nature may not have any copy.
- There are two types of URL
- 1. Protocol
- 2. Resource name

http:\\ www.google.com
Protocol resource name

- An HTTP URL has two main jobs: to identify which web server maintains the resource, and to identify which of the resources at that server is required.
- Figure above shows three browsers issuing requests for resources managed by three web servers.
- The topmost browser is issuing a query to a search engine.
- The middle browser requires the default page of another web site.
- The bottommost browser requires a web page that is specified in full, including a path name relative to the server.

- Publishing a resource: While the Web has a clearly defined model for accessing a resource from its URL, the exact methods for publishing resources on the Web are dependent upon the web server implementation. In terms of low-level mechanisms, the simplest method of publishing a resource on the Web is to place the corresponding file in a directory that the web server can access.
- The types of websites are:

1. Static
2. Dynamic

- Web application:

Web applications are run into browser through URL.

Types :

i. Service Oriented : it is used to implement web services. It is coded into CGI, JSP, ASP, etc.

JSP stands for Java Server Pages, which helps developers to create dynamically web pages based on HTML, XML, or other types.

ASP stands for Active Server Pages, which is used in web development to implement dynamic web pages.

CGI stands for Common Gateway Interface. It is a technology that enables a web browser to submit forms and connect to programs over a web server. It is the best way for a web server to send forms and connect to programs on the server.

ii. Presentation oriented:

- It provides client side services. They are coded using HTML, XML, Javascript, etc.

Web architecture:

- WWW follows the 2 tier architecture.
- It is combination of webserver and web client.
- Webserver produce and deliver the information.
- Web client retrieve and display information.