INTRODUCTION to DISTRIBUTED SYSTEMS

References [Col94], [Gos91], [Tan95]

D1STRIBUTED SYSTEMS

- **Definition of the distributed system**

STRUCTURE of DISTRIBUTED SYSTEMS

- **HARDWARE ARCHITECTURES**
  - **Tightly-coupled systems**
  - **Loosely-coupled systems**
- **SOFTWARE ARCHITECTURES**
  - **Network operating system (NTOS)**
  - **Multiprocessor operating system (MTOS)**
  - **Distributed operating system (DTOS)**
  - **Co-operative autonomous system**
- **MODELS of DISTRIBUTED SYSTEMS**
  - **Minicomputer model**
  - **Workstation model**
  - **Workstations-server model**
  - **Processor-pool model**
  - **Hybrid model**

PROPERTIES of DISTRIBUTED SYSTEMS

- **FEATURES of DISTRIBUTED SYSTEMS**
- **KEY CHARACTERISTICS of DISTRIBUTED SYSTEMS**
  - **Resource sharing**
  - **Autonomy & interdependence**
  - **Heterogeneity & openness**
  - **Concurrency and parallelism**
  - **Scalability**
  - **Compatibility**
  - **Transparency**
  - **Reliability**
  - **State inconsistency**
  - **Timing**
- **DESIGN GOALS & ISSUES of DISTRIBUTED SYSTEMS**
  - **Design goals**
  - **Design problems**
  - **Design issues of distributed systems**
- **ADVANTAGES and DISADVANTAGES of DISTRIBUTED SYSTEMS**
  - **Advantages of distributed systems**
  - **Disadvantages of distributed systems**

DISTRIBUTED SYSTEMS

Definition of the distributed system

- distributed systems is a set of computers that appears to the users as a single computer but it runs on multiple independent computers,

- the user views the computer system as a virtual uniprocessor and not as a set of computers,

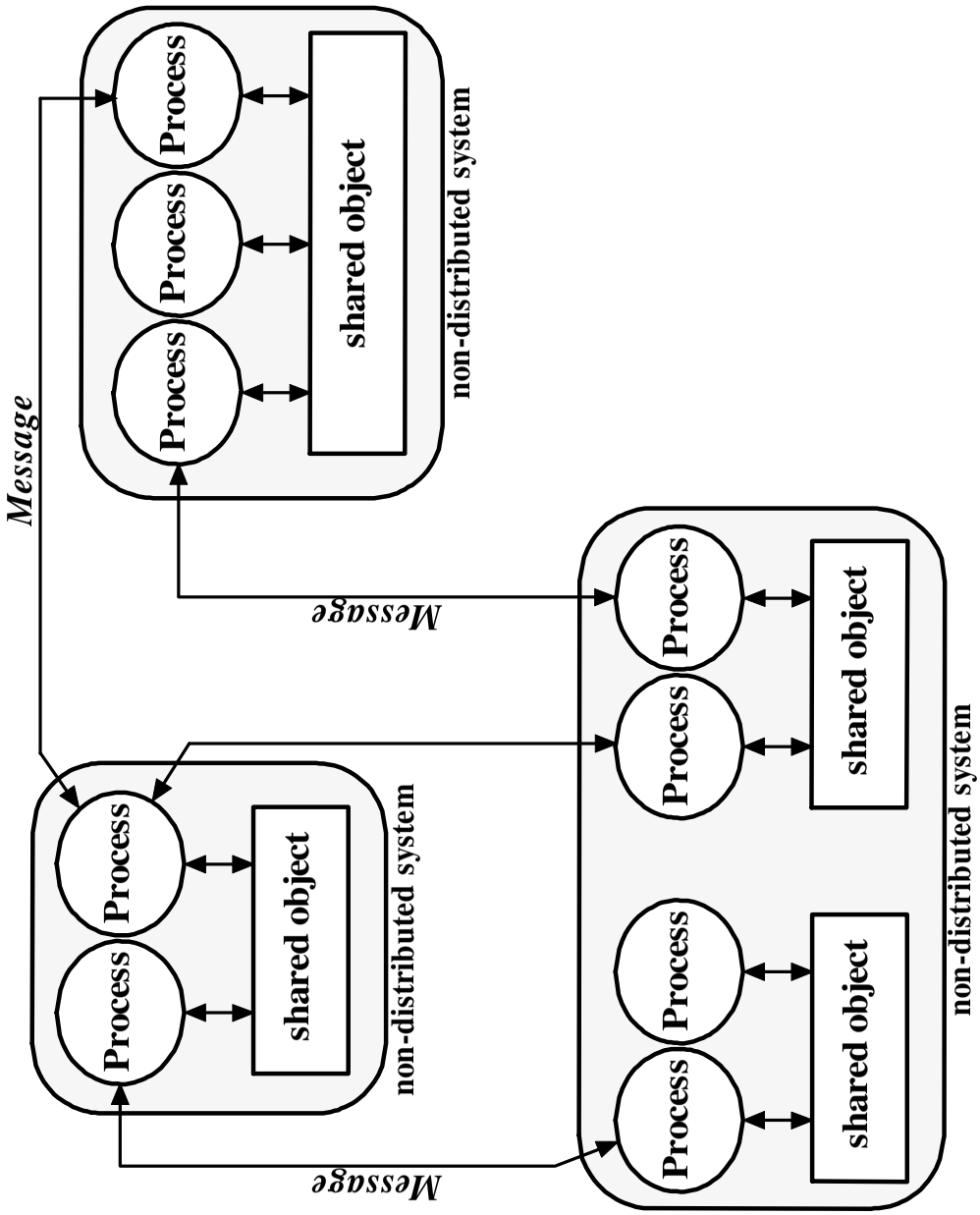
- the user does not know on what computer the job or task was performed and on what computer data was stored and how the communication and synchronisation among computers was carried out,

- the use of multiple computers, the access to data at different sites and the communication among computers should be invisible or transparent for the users,

- to provide the transparency, the same copy of the operating systems, the distributed operating system, runs at every computer,

- the distributed operating system hides implementation details from the users,

- many applications are inherently distributed applications, they require that some processing power be available at the distributed locations for collecting, pre-processing and accessing data.



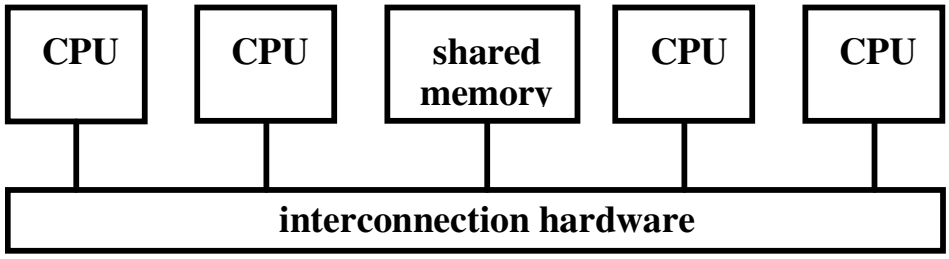
Distributed system which comprises several non-distributed systems.

STRUCTURE of DISTRIBUTED SYSTEMS

HARDWARE ARCHITECTURES

Tightly-coupled systems

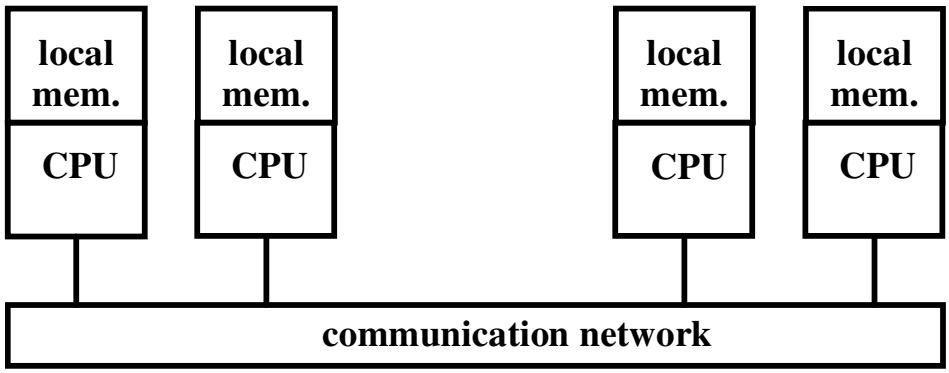
- *parallel systems* - address space is shared by all the processors



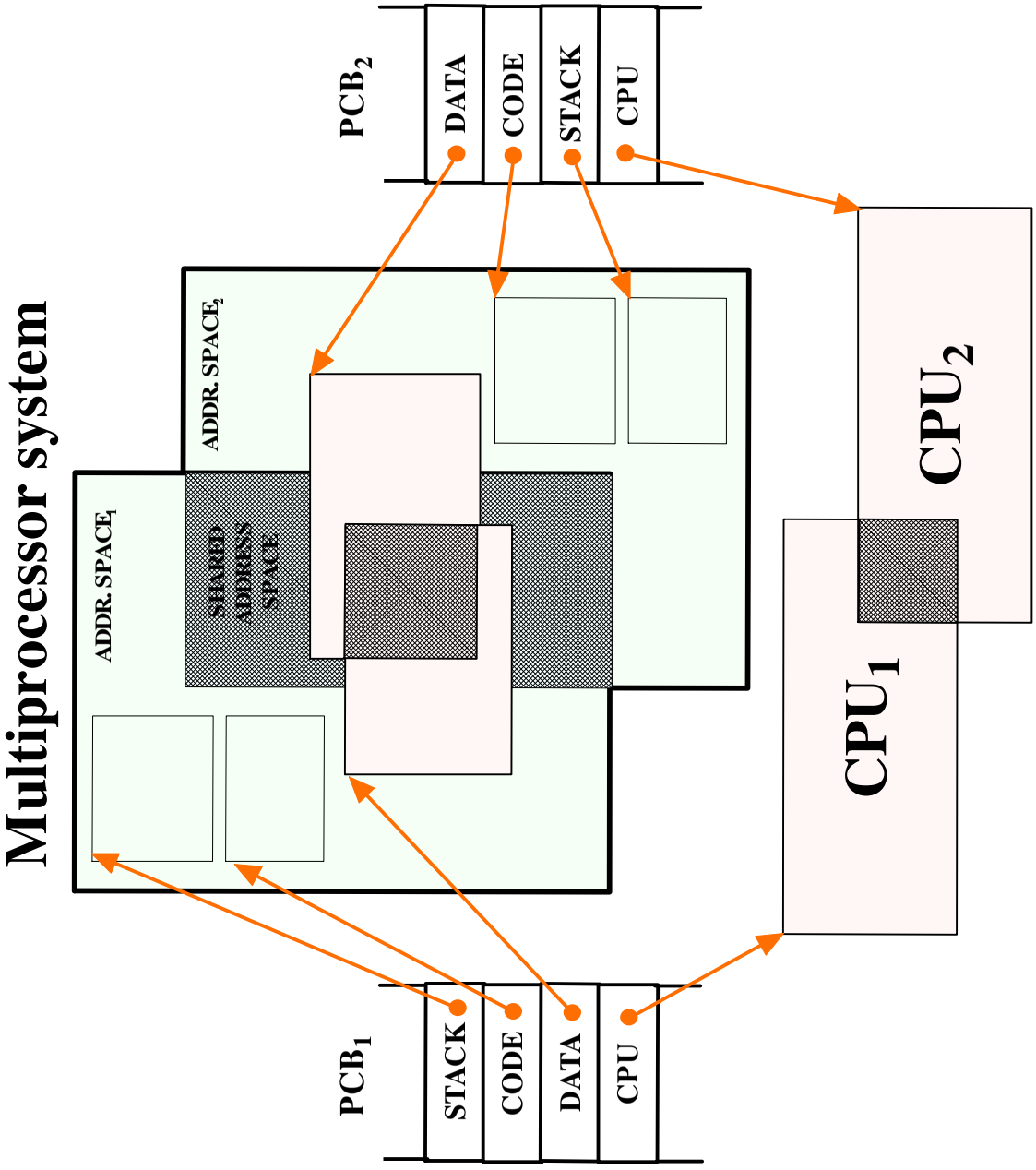
tightly-coupled systems

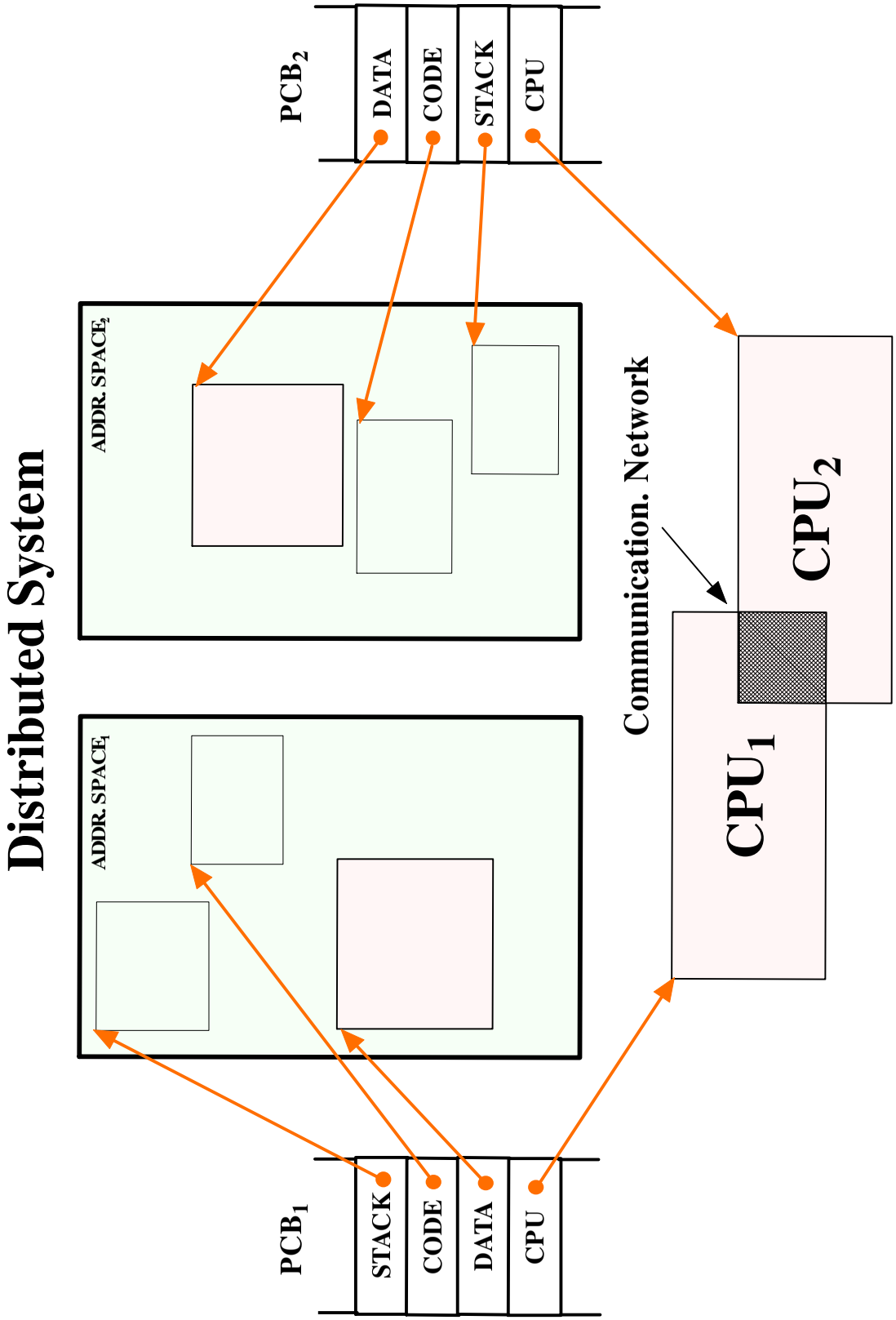
Loosely-coupled systems

- *distributed systems* - each processor has its own local memory



loosely-coupled systems


Multiprocessor system
ADDR. SPACE1
ADDR. SPACE2
SHARED ADDRESS SPACE
PCB1
STACK
CODE
DATA
CPU
PCB2
DATA
CODE
STACK
CPU
CPU1
CPU2


Distributed System
PCB2
DATA
CODE
STACK
CPU
ADDR. SPACE2
Communication. Network
CPU2
CPU1
ADDR. SPACE1
PCB1
STACK
CODE
DATA
CPU

SOFTWARE ARCHITECTURES

Network operating system (NTOS)

- computers are loosely-coupled by a communication network and have loosely-coupled OS i.e. each runs its own OS that may be different from others,

- OS manages computers and communication between users and servers.

- each computer has its own OS and its own user whose tasks are performed by the computer,

- each computer can do whatever it wants as long as it obeys standard protocols

- no direct hw or sw control of one computer to another exists in the network, and the communication overhead among computers is much higher than the internal data transfer within each computer,

- the computer system may contain servers for example file and print server,

Multiprocessor operating system (MTOS)

- computers are tightly-coupled by a bus system and have tightly-coupled OS

- there is a list of all processes, stored in the shared memory, that are ready to run,

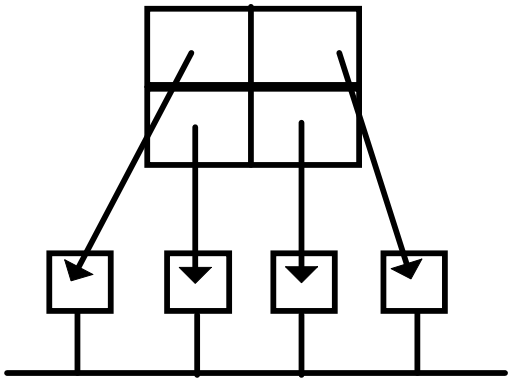
- none of the processors have local memory and all programs are stored in the global shared memory

Distributed operating system (DTOS)

- computers are loosely-coupled by a communication network and have tightly-coupled OS i.e. each computer has the same kernel that has considerable control over its own local resources,

- process management must be the same and there must be a global file system in the computer system,

- it supports a global interprocess communication mechanism i.e. there is the same mechanisms for local and remote communication,



decomposition in distributed systems

- distributed systems are characterised by service decomposition among several servers,

Co-operative autonomous system

- co-operative autonomous system is a high-level service-oriented sw system that requires the support of communication,

- the key concept of the co-operative autonomous systems is the integration of services to form co-operative activities

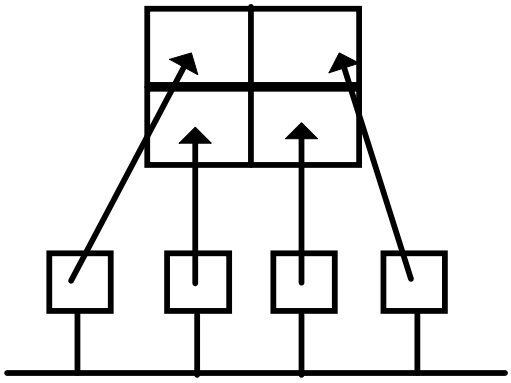
- co-operative autonomous systems use intelligent trader and broker services to facilitate interaction of co-operative autonomous systems,

- in co-operative autonomous systems both hw and sw are completely de-coupled and decentralised,

- in co-operative autonomous systems each user operates autonomously by exporting and requesting services,

- group activities can be co-ordinated by exchanging services and requests where high-level services can be composed from low-level services,

- any sw system can be formed without any limitations by integrating various services using some structures,



composition in co-operative systems

- there are several development in co-operative autonomous systems for example:

ODP = Open Distributed Processing

- **its supports distribution, interoperability and portability for heterogeneous distributed processing both within and among autonomous organisations**

CORBA = Common Object Request Broker Architecture

- **it uses the object-oriented model to implement transparent services request across an interconnected heterogeneous multiple object distributed systems**

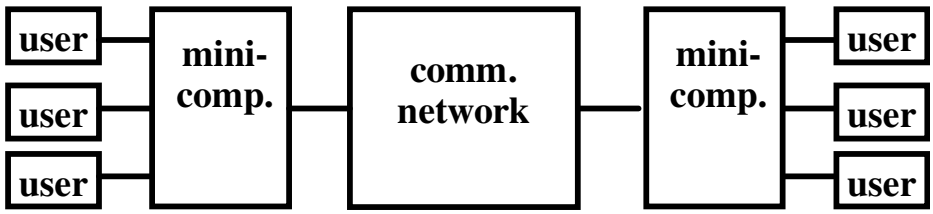
MODELS of DISTRIBUTED SYSTEMS

Minicomputer model

- several interactive terminals are connected to each minicomputer that is connected through the communication network to other minicomputers

- each computer supports multiple users and provides access to remote resources,

- the ratio of the number of processors to the number of users is normally less than one



minicomputer model

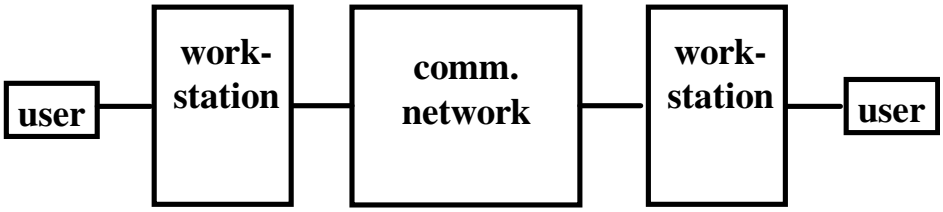
Workstation model

- several workstations are connected by a communication network,

- each user has a separate workstation, where all of the user's work is executed,

- there is a distributed file system that provides access to data for users,

- the ratio of the number of processors to the number of users is one,



workstation model

Workstations-server model

- this model is based on multiple workstations, which are used by users, and several servers, which provides services for users,

- the workstation is a stand-alone computer that provides local processing capability and access to the communication network for users,

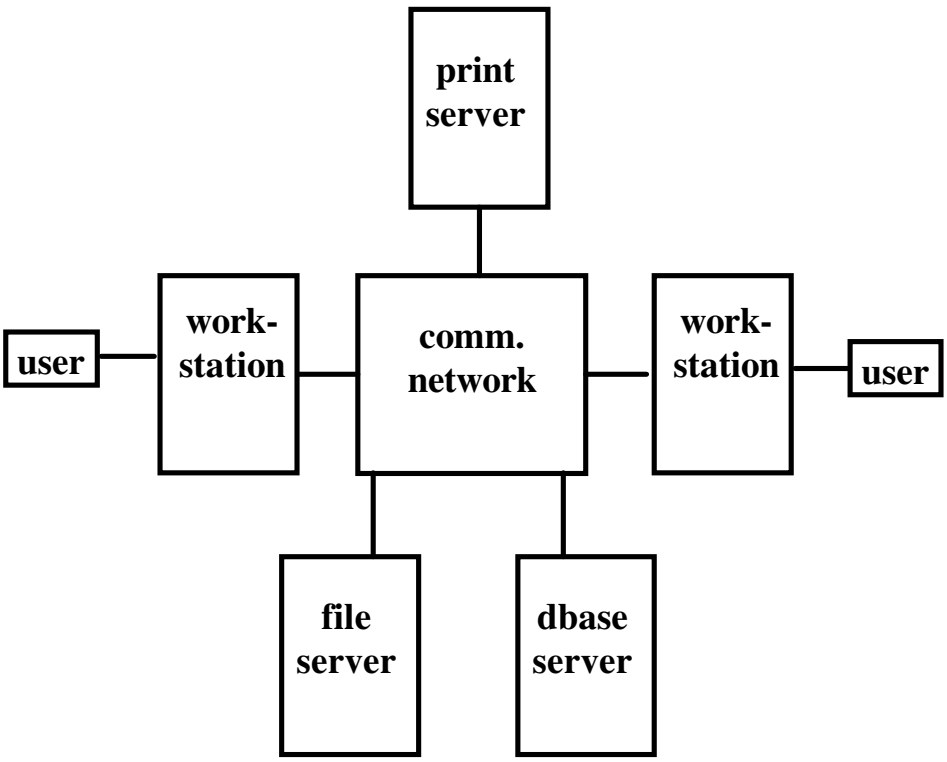
- workstations and servers connected by a communication network that provides access to services of servers,

- servers are computers equipped with fast and/or high-power hw and sw resources and they provide services for workstations using these resources

client = user that requires services provided by servers,

server = provides a particular type of service to a priori unknown client,

service = a sw entity running on one or more servers



workstations-server model

- advantages of the workstation-server model:

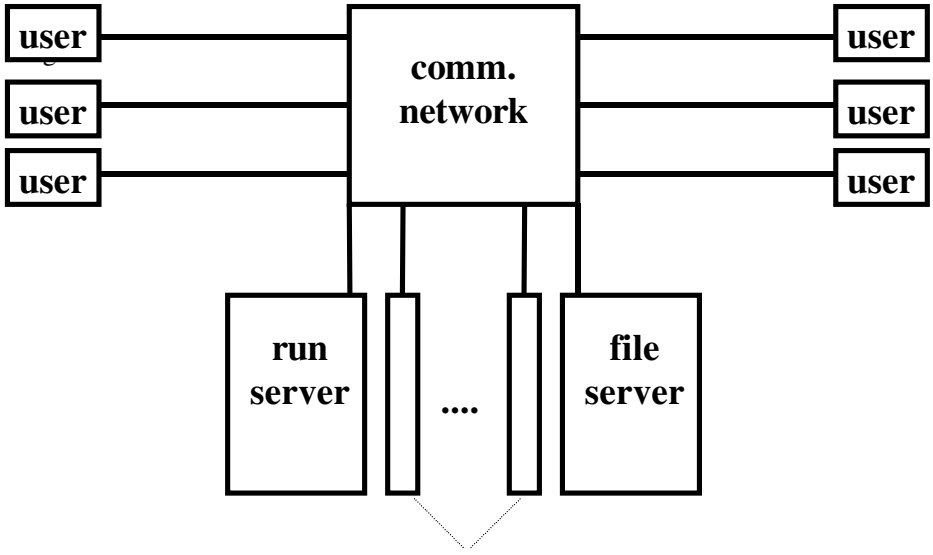
- **much cheaper to use fast and high-power resources by all users than to provide dedicated resources for each user for example servers with large and fast speed disks than workstations with small and low speed disks,**

- **it does not need a process migration facility because it uses request-response protocol**

- **user has a guaranteed response time because workstations are not used for executing remote processes**

Processor-pool model

- the processors are pooled together to be shared by the users as needed, where each processor in the pool has its own memory to load and run a system program or an application pr
- ogram,



pool of processors

processor-pool model

- special processor - run server - manages and allocates the processors in the pool to different users on a demand basis,

- this model allocates one or more processors to a user taking into account the user's needs,

- one or more of the processors assigned to a user first, they execute their tasks next, they return to the pool after completing their tasks and await a new assignment,

- the ratio of the number of processors to the number of users is normally greater than one,

- advantages of the processor-pool model:

- **provides better utilisation of the available processing power of the system,**
- **the system's services can be easily expanded without the need to install more computers**

- disadvantages of the processor-pool model:

- **it is unsuitable for high-performance interactive applications using graphics or window systems**

Hybrid model

- it is based on the workstation-model but with the addition of a pool of processors

PROPERTIES of DISTRIBUTED SYSTEMS

- tasks of operating systems:

- **to present users with a virtual machine that is easier to program than the underlying hardware**
- **to manage the various resources of the system**

FEATURES of DISTRIBUTED SYSTEMS

1. performance

DTOS

- **provides rapid response time and higher system throughput due to the fact that many task can be concurrently executed at different computers,**
- **employs a load distributing approach to improve response time,**

2. system image

DTOS

- **it hides the existence of multiple computers from the users,**
- **a user of a DTOS need not keep track of the locations of various resources for accessing them, and**
- **the same set of system calls is used for accessing both local and remote resources,**

NTOS

- **collection of distinct computers connected by a communication network i.e. users are aware of the fact that multiple computers are being used**
- **a user has to know the location of a source to access it, and different sets of system calls have to be used for accessing local and remote calls**

3. autonomy

DTOS

- **a system-wide operating system controls the whole system and each computer runs a part of this global operating system,**
- **it tightly interconnects all the computers of the distributed system,**
- **processes and resources are managed globally by a single set of system calls available on all computers,**
- **identical kernels are run on all the computers and they co-operate with each other to make global decisions,**

NTOS

- **it is built on a set of existing operating systems and handles the interfacing and co-ordination of remote operations and communication among these operating systems,**
- **each computer has its own local OS, and there is no co-ordination at all among the computers at OS level,**
- **each computer functions independently of other computers,**

4. availability and reliability

DTOS

- **provides higher availability and reliability because the system has a small number of components that can fail without affecting the availability of the rest of the system,**

5. fault tolerance

DTOS

- **most of the users are normally unaffected by the failed machines and can continue their work,**

NTOS

- **provides little or no fault tolerance capability if the communication network fails,**

KEY CHARACTERISTICS of DISTRIBUTED SYSTEMS

Resource sharing

- resources are physically encapsulated within one computer and can be accessed from other computers by communication,

- users of DTOS should be able to access remote resources as easily as they can access local resources

- resource sharing requires:

- **naming scheme for each class of resources to be accessed from any location,**
- **mapping of resource names to communication addresses,**
- **co-ordination of concurrent accesses that change the state of shared resources,**

- shared resources are made available for users in the following way:

- **data migration**
  - ♦ **data is brought to the location of the computation that needs access to it by the DTOS**
  - ♦ **the data may be a file stored either locally or remotely, or contents of a physical memory of either a local or a remote computer,**
- **computation migration**
  - ♦ **the computation migrates to another location**

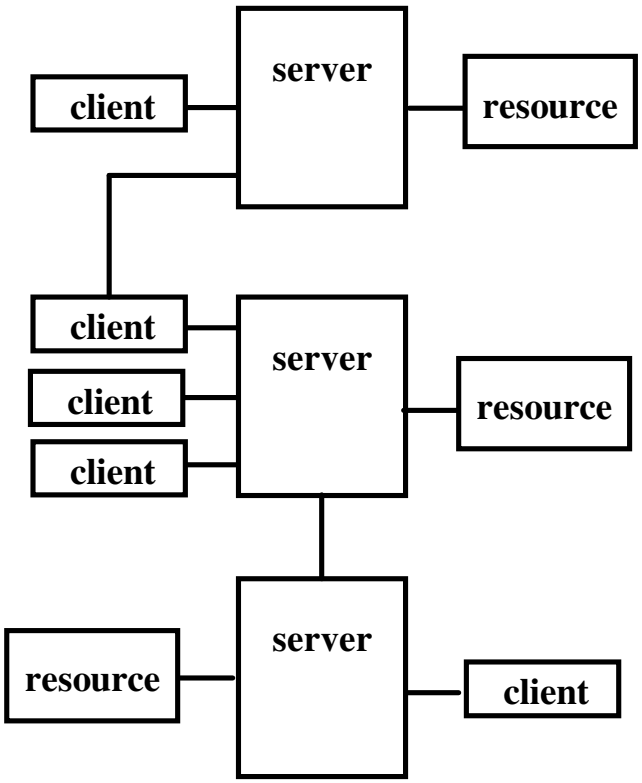
- **distributed scheduling**

  - **processes are transferred from one computer to another by the DTOS**

  - **process relocation is recommended if the computer where a process originated is overloaded or it does not have the necessary resources,**

- resource sharing models:

client-server model

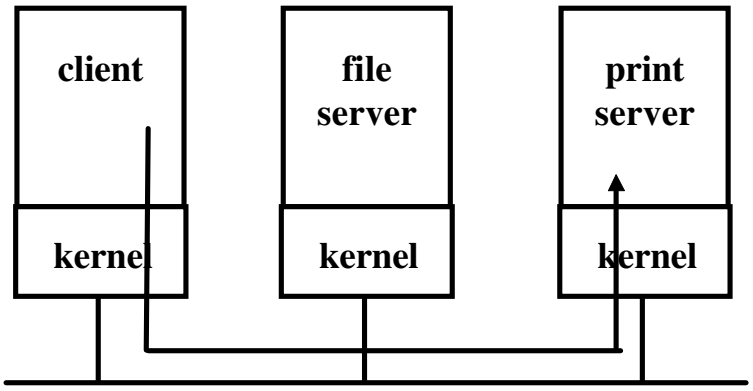


client-server model

- **client-server model is a process model based on a message passing among processes that provides co-operation and interaction,**

- **set of server processes, each acting as a resource manager for collection of resources and a collection of client processes each performing a task that requires access to some shared hw and sw resources,**

- **client processes issue requests to servers whenever they want to access one of resources, then the server performs the requested action and sends a reply to the client process,**



client-server model in a distributed system

problems of client server model:

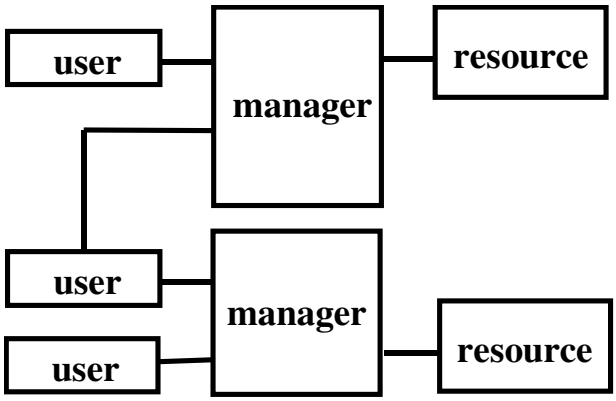
- **the control of resources is centralised in a single server, thus if the computer supporting the server fails, then the server also fails, thus each single server is a potential bottleneck,**

- **to improve performance multiple servers are used, but this increases total costs of distributed systems,**

<u>object-based model</u>

- **every entity in a running program is viewed as an object with a message-handling interface providing access to its operations,**

- **each shared resource is viewed as an object i.e. collection compromising of a data structure and a set of operations defined on this data structure, and operations or services are provided by objects**



object-based model

- **objects communicates via communication primitives implemented on by messages,**

- **synchronisation and control are performed via management and allocation of capabilities,**

- **whenever a program using a resource requires access to a resource it sends a message containing a request to the corresponding object, the message is dispatched to the appropriate process which performs the requested operation and sends a reply message to the requesting process if required,**

pipe model

- **pipe is a communication facility implemented using the file system which allows:**

  - **transfer of data between processes based on a FIFO strategy,**

  - **synchronisation of process execution,**

- **communication through pipes allows processes to exchange messages even though they do not know what processes are on the other end of pipe,**

- **pipes are used in communication as a remote procedure, but pipes do not block the callers and do not return the results**

- **types of pipes:**

  - **named pipe "open" system call in UNIX**

  - **unnamed pipe "pipe" system call in UNIX**

Autonomy & interdependence

- DTOS provides autonomy for the following reasons:

- **to provide policy freedom by giving a user a choice of where, how and when to compute,**

- **to provide robustness i.e. a system remains operational despite temporarily failures of resources, and**

- **to control co-operation between mutually suspicious users and to make**

a distributed system working correctly in the face of different problems,

- DTOS minimises interdependency by using the following methods:

- **services which are not needed should be not used,**
- **to avoid communication overhead use a workstation on which a requested service can be provided locally; and**
- **resources should be replicated if a service which utilises it is crucial.**

Heterogeneity & openness

- there are the following levels of heterogeneity in DSs:

- **computer heterogeneity,**
- **network heterogeneity,**
- **operating system heterogeneity.**

- open distributed systems may be extended at the hw level by the addition of computers to the computer system and at the sw level by the introduction of new services, enabling application programs to share resources because

- **their key interfaces are published,**
- **they are based on an uniform interprocess communication.**

Concurrency and parallelism

- in distributed systems concurrency and parallelism arise from separate activities of users, the independence of resources and the location of server processes in separate computers because

- **many users simultaneously invoke commands or interact with application programs,**
- **many server processes run concurrently, each responding to different requests from client processes,**

Scalability

- computer systems grow with time that should not result in unavailability or degraded performance,

- the system hw and the sw should not need to change when the scale of the system increases,

- in distributed systems no single resources - neither hw nor sw - is assumed to be in restricted supply, thus as the demand for a resource grows, it should be possible to extend the system to meet it,

Compatibility

- compatibility is interoperability among different resources in a system,

- there are three different levels of compatibility:

- **binary level compatibility**
  - ♦ **all processors execute the same binary instruction set, even though the processors may differ in performance and in input-output**
- **execution level compatibility**
  - ♦ **the same source code can be compiled and executed properly on any computer in the system**
- **protocol level compatibility**
  - ♦ **achieves interoperability by requiring all system components to support a common set of protocols**
  - ♦ **computers can run different OSs while not sacrificing their interoperability**

Transparency

- concealment from the user the separation of hw and sw components and all irrelevant system dependent details from the user whenever possible in DSs,

- DTOS ensures that users view a DSs as a virtual uniprocessor, not a collection of distinct computers connected by communication subsystem,

- development and maintenance of sw for DSs is far more difficult and expensive than centralised system sw because:

- **accessing of remote resources is different from local resource access,**
- **the possible number of errors in DSs is greater than in centralised systems, and**
- **the transmission delays influence operations in DSs,**

- the transparency makes the network invisible to most users and applications

- types of transparency:

- ***access transparency***
  - ♦ **enables local and remote information to be accessed using identical operation,**
  - ♦ **the physical separation of system objects is concealed from the users**
- ***concurrency transparency***
  - ♦ **enables several processes to operate concurrently using shared information without interference between them, as a result user does not notice the existence of other users,**
- ***parallel transparency***
  - ♦ **permits parallel activities without users knowing how, where and when these activities are carried out by the systems**

- ***replication transparency***

♦ **enables multiple instances of information to be used without knowledge of the replicas by users or application programs,**

♦ **it is closely related to concurrency transparency but is treated separately since files and data are special objects**

- ***scaling transparency***

♦ **allows the system and applications to expand in scale without change to the system structure or the application algorithms,**

- ***location transparency***

♦ **enables information to be accessed without knowledge of their location,**

♦ **objects are mapped and referred to by logical names thus it is also called as name transparency,**

- ***migration transparency***

♦ **allows the movement of information and resources within a system without affecting the operation of users or application program,**

♦ **object is not only referred to by its logical name but can also be moved to a different physical location without changing the name**

- ***execution transparency***

♦ **allows load balancing, moving programs and data, interprocess communication between processes in different processors,**

- ***performance transparency***

♦ **allows the system to be reconfigured to improve performance as loads vary,**

- ***fault transparency***

♦ **enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hw or sw components**

♦ **transforms failures into graceful system performance degradation rather than disruptions and damage to the users is minimised**

Reliability

- resources must be available for users at any time, the availability is provided by either usage of critical components or redundancy,

- reliability also includes that resources must be protected from unauthorised usage,

- failure modes are independent because components of distributed systems may fail independently of each other,

- fault tolerance increases reliability by two approaches in distributed systems:

- ***hw redundancy*** **- use of redundant component**
- ***sw recovery*** **- capability of recovering from faults**

State inconsistency

- concurrency, communication delays and failures lead to a non-consistent state of distributed systems,

Timing

- each computer of the distributed system has a local clock, but they do not guarantee the same time at all computers as a result of clock drift,

DESIGN GOALS & ISSUES of DISTRIBUTED SYSTEMS

Design goals

- high performance

-

- efficiency

- **applications should be structured by assigning distributed processes such that computation and communication can be balanced and overlapped whenever possible**

- **computation speed and system throughput can be enhanced through distributed processing and load sharing if the communication system is carefully designed**

- scalability

-

- flexibility

- **it implies the ease of use of the system interface and the ability to relate the computation process,**

- **it is the system's ability to evolve and migrate that includes modularity, scalability, portability and interoperability**

- consistency

- **the lack of global information, potential replication and partitioning of data, the possibility of component failures, complexity of interaction among modules contribute to inconsistency in the system**

- reliability,

-

- security

- **DSs are vulnerable to security threats because of their openness in the operating environment,**
- **security is provided authentication, which includes authentication of clients, messages and servers, and authorisation, which means access control across the communication network**

- robustness

- **it means that the system is capable of reinitialising itself to a state where the integrity of the system is preserved with only some possible degradation of its performance**

- fault tolerance

- **failures can be handled if there is redundancy in the system,**
- **redundancy is an inherent property in a DS since data and resources can be replicated**
- **to provide fault tolerance state information of the system must be kept in DSs,**

Design problems

- resources are physically separated,

- delivery of messages is delayed and messages could be lost,

- no complete information is available

- no common clock among the multiple processors,

Design issues of distributed systems

1. naming

- object in DSs are processes, files, memory, I/O devices, networks and processors,

- each object can be associated with some well-defined access operations to the object and it is represented by the allowable operations on the object,

- the process that manages the object is the object server i.e. objects are encapsulated in servers and the only visible entities in the system are servers

- names assigned to resources or objects must have global meanings, and they must be supported by a name interpretation system that can translate names in order to enable programs to access resources

design goals = naming schemes that will scale to an appropriate degree, in which names are translated efficiently to meet appropriate goals of performance,

design issues = the choice of an appropriate name space for each type of resource,

- resource names must be resolvable to communication and server identifiers,

2. communication

- the most essential issue in any DS since everything relies on it,

- at the lowest level, message passing is the only means of communication in DSs using message passing primitives,

- at the highest level logical communication methods hide the physical details of message passing

design goals = to optimise the implementation of communication while retaining a high-level programming model,

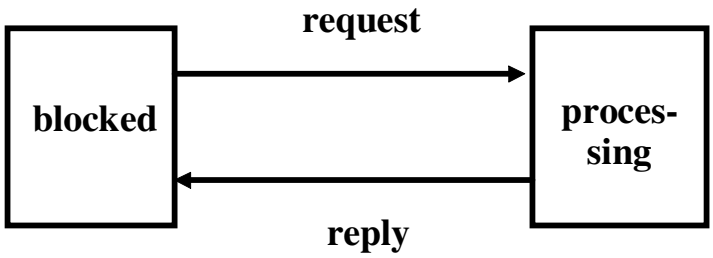
design issues = data transfer between processes by message-passing,

- synchronisation of sending and receiving activity by communication primitives,

- communication models used in distributed systems:

1. client-server communication model

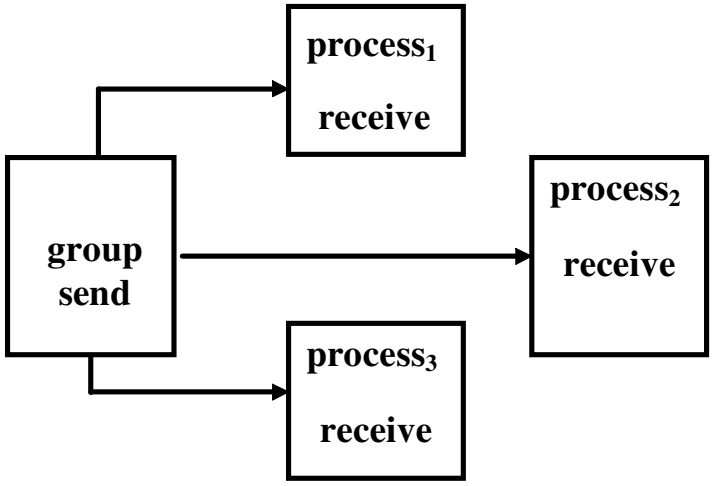
- **a client process sends a request to a server process and the client process blocks until receiving reply from the server process,**
- **the server process executes the request,**
- **the server process sends a reply to the client process**



client-server communication model

2. group communication model

- **processes interact by message-passing, but the target is not a single process but a group of processes**



group communication model

3. synchronisation & co-ordination

- interacting concurrent processes require co-ordination to achieve synchronisation

- there are three types of synchronisation and co-ordination:

a. barrier synchronisation

- set of processes must reach a common synchronisation point before they can continue

b. condition co-ordination

- a process must wait for a condition that will be set asynchronously by other interacting processes to maintain some ordering of execution

c. mutual exclusion

- concurrent processes must have mutual exclusion when accessing a critical shared resource

- synchronisation is difficult to perform because complete state information is difficult to obtain in DSs

- state info is obtained by sending messages but it is in accurate or incomplete by the time messages are received due to the message transfer time delay in the network

<u>design goals</u> = to specify and control of the joint activity of co-operating processes

- serialisation of concurrent access to shared objects by multiple processes

<u>design issues</u> = event ordering in order to allow only one process to be in the critical section,

- message-based resolution protocols or distributed co-ordination provides synchronisation or co-ordination in DSs,

4. software structure

<u>design goals</u> = to structure a system so that new services can be introduced that will co-operate fully with existing services without duplicating existing service elements,

<u>design issues</u> = to restrict the role of the kernel to basic resource management,

- to provide all other services by open services to all other shared resources and services,

- basic types of kernel:

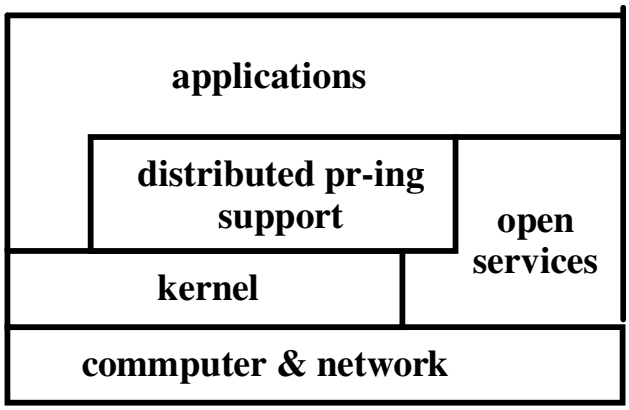
<u>1. monolithic kernel:</u>

- **the monolithic kernel would consist of all the services provided by the OS,**
- **system calls are made by trap to the kernel that provides the requested service returning the result to the user process,**
- **in DSs it is not worth for every computer to run a huge monolithic OS when not every computer could use every service provided by the OS**

<u>2. microkernel:</u>

- **the OS is structured as a collection of processes that are largely independent of each other,**
- **provides the most basic set of resources and services while protecting the basic hw & sw components by independent processes,**
- **the microkernel supports the interaction between processes and some *basic services* that are essential to every computer for example:**
  - ♦ **memory allocation & protection,**
  - ♦ **some process management & scheduling,**
  - ♦ **interprocess communication,**

♦ **low level input/output handling**



structure of the microkernel

- **microkernel does not provide services for the file system, most of the systems calls and full process management are implemented by user level servers,**

- **flexibility is provided by adding or changing services so-called *open services* that does not require stopping the system and booting a new kernel,**

♦ **open services bring the programming facilities of a distributed system up to level required for application programming for example file service, electronic mail delivery service**

- **run-time support for language facilities to allow programs written in conventional languages to work together for example remote procedure calling, group operation**

5. workload allocation

<u>design goals</u> = to deploy the processing, communication and resources in a network to optimum effect in the processing of changing workload,

<u>design issues</u> = use of idle workstation - to use idle or under-utilised workstations as a pool of extra computers that have spare processing capacity that can be used by users who do not have sufficient capacity,

- **approaches to decrease workload in distributed systems:**

1. processor pool model

- ♦ **processor pool consists of a collection of computers each of them has network connection,**
- ♦ **processors are allocated to processes for their lifetimes and user is able to access more computing power than a single computer can offer**

2. processors with shared-memory

- ♦ **processes communicate with each other via interprocess communication using the shared memory**

6. transparency

design goals = to ensure that users view a DS as a virtual uniprocessor, not as a collection of distinct computers,

design issues = to provide easier sw development,

- to support incremental changes,

- to provide potential for increased reliability,

- to have simpler user model of DS

7. consistency maintenance

design goals = to optimise the performance of distributed systems

design issues = all copies of replicated information should be identical, (to solve problem of "multiple copy update")

a. *update consistency* - several processes access and update date concurrently

solution - to exclude mutual access to data that can be updated by several processes,

b. *replication consistency* - data that are derived from a single source have been copied to several computers and subsequently modified at

more of them

<u>solution</u> - each modification should be transferred to all other computers

c. *<u>cache consistency</u>* - data values that have been cached by one client are updated by another one,

<u>solution</u> - to exclude mutual updating to cache that can be accessed by several processes

d. *<u>failure consistency</u>* - when one computer fails, the others, including any with which it was co-operating, will normally continue to run, but if they depend on the continued operation of the failed computer then they may fail later

<u>solution</u> - recovery procedures are used to roll back the computer that failed to a known state after the failure

e. *<u>clock consistency</u>* - clocks of computers can only be synchronised by communication via network, but sending message through a network normally takes an unpredictable amount of time

<u>solution</u> - logical clocks that are used to define an order on events without measuring the physical time at which they occurred

ADVANTAGES and DISADVANTAGES of DISTRIBUTED SYSTEMS

Advantages of distributed systems

1. information sharing among distributed users:

- **information generated by the users working at one node can be shared by users working at other nodes,**

2. resource sharing

- **hw and sw resources can be shared among all the computers and the users within the distributed computing system.**

3. good performance-price ratio as a result of the

- **increasing power and reduction in the price of microprocessors,**
- **increasing speed of communication network,**
- **resource sharing among multiple computers,**
- **computers & workstations offer a better performance/price ration than mainframes**

4. short response time & high throughput as a result of the

- **multiple processors among that computation can be partitioned into number of processes or tasks,**
- **load distribution among the multiple processors by moving processes or jobs from currently overloaded processors to lightly loaded ones,**

5. high reliability

- **as a result of the**
  - **multiple processors and storage devices allow the maintenance of multiple copies of critical information within the system,**
  - **the execution of important processes redundantly protects them against failures,**
  - **if one computer fails the system as a whole can further work distributing the workload of the failed computer among other computers,**

6. extensibility & incremental growth

- **distributed systems could be gradually extended by adding additional resources in order to increase their power and functionality,**

7. flexibility to meet users' needs

- **distributed system may have a pool of different types of computers, thus the most appropriate one can be selected for processing a user's job depending on the nature of the job,**

Disadvantages of distributed systems

1. software support

- **not too much experience in designing, implementing and using distributed software,**

2. communication network

- **it can loses messages and can be overloaded,**
- **it can be failed and the distributed system can be out of work,**

3. security

- **access of data all over the system is not regulated**