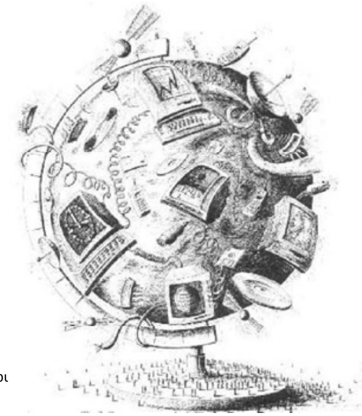
Lecture (01)
Introduction

Dr. Ahmed ElShafee

Agenda

- Distributed System
- Design Considerations
- Protocol Layers
- Port and Socket
- Connection Models
- Distributed Computing Models
- Middleware



Distributed System

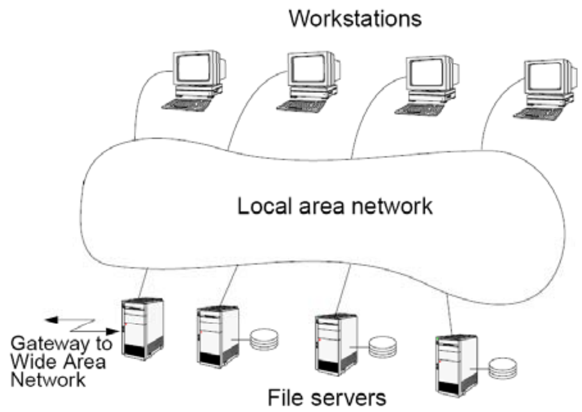
Definition:

- Distributed system can be defined as a combination of several computers with separate memory, linked over a network, and on which it is possible to run a distributed applications.

- Characteristics:
  1) capable of communicating over a network
  2) the network is usually stable
  3) fail-safe
  4) each device has a permanent identification within the network

Distributed System 2

- So, it is a collection of independent computers, interconnected via a network, capable of collaborating on a task.



Distributed Application

- A distributed application consist of several parts of a program communicating with each other, which cooperate to carry out a common task.
- For example, client server application.
- Typically, but not necessarily, the parts of the application are distributed across several computers
- The distribution can also be simulated on one computer.
- In this case, however, information is not transmitted via a common memory or address space, but with the aid of techniques of remote communication, over a network.

Distributed Programming

- Distributed programming is a model in which processing occurs in many different places (or nodes) around a network.

- Characteristics:
  1) processing can occur whenever it makes the most sense
  2) carried out on a distributed system
  3) making calls to other address spaces possibly on different machines
  4) tasks are handled in parallel

Why Distributed Programming?

1) balance resource loading
2) lower cost of development since clients can access remote codes for services
3) separation of concerns (each machine responsible of different function)
4) Platform independence

Design Considerations

In general, three aspects need to be put into consideration:

1. Transparency
2. Communication
3. Performance
4. scalability
5. Heterogeneity
6. Openness
7. Reliability
8. Fault tolerance
9. Security

Design Considerations 2

Transparency

☞ How to achieve the single system image?

☞ How to "fool" everyone into thinking that the collection of machines is a "simple“ computer?

Design Considerations 3

1.1 Access transparency

Local and remote resources are accessed using identical operations.

1.2 Location transparency

Users cannot tell where hardware and software resources (CPUs, files, data bases) are located.

The name of the resource shouldn't encode the location of the resource.

Design Considerations 4

1.3 Migration (mobility) transparency

Resources should be free to move from one location to another without having their names changed.

1.4 Replication transparency

The system is free to make additional copies of files and other resources (for purpose of performance and/or reliability), without the users noticing.

Example: several copies of a file; at a certain request that copy is accessed which is the closest to the client.

Design Considerations 5

1.5 Concurrency transparency

The users will not notice the existence of other users in the system (even if they access the same resources).

1.6 Failure transparency

Applications should be able to complete their task even if failures occurred in certain components of the system.

Design Considerations 6

1.7 Performance transparency

Load variation should not lead to performance degradation.

This could be achieved by automatic reconfiguration as response to changes of the load;

* it is difficult to achieve.

Design Considerations 7

2. Communication

☞ Components of a distributed system have to communicate in order to interact.

This implies support at two levels:

1. ***Networking infrastructure*** (interconnections & network software).
2. Appropriate ***communication primitives*** and ***models*** and their implementation:

Design Considerations 8

3. Performance

Several factors are influencing the ***performance*** of a distributed system:

- The performance of individual workstations.
- The speed of the communication infrastructure.
- Extent to which reliability (fault tolerance) is provided (replication and preservation of coherence imply large overheads).
- Flexibility in workload allocation: for example, idle processors (workstations) could be allocated automatically to a user's task.

Design Considerations 9

4. Scalability

The system should remain efficient even with a significant increase in the number of users and resources connected:

- Cost of adding resources should be **reasonable**;
- Performance loss with increased number of users and resources should be **controlled**;
- Software resources should not **run out** (number of bits allocated to addresses, number of entries in tables, etc.)

Design Considerations 10

5. Heterogeneity

Distributed applications are typically heterogeneous:

Different hardware: mainframes, workstations, PCs, servers, etc.;

- Different software: UNIX, MSWindows, IBM OS/2, Real-time OSs, etc.;
- Unconventional devices: teller machines, telephone switches, robots, manufacturing systems, etc.;
- Diverse networks and protocols: Ethernet, FDDI, ATM, TCP/IP, Novell Netware, etc.

Design Considerations 11

6. Openness

One of the important features of distributed systems

is openness and flexibility:

- Every service is equally accessible to every client (local or remote);
- It is easy to implement, install and debug new services;
- Users can write and install their own services.

Design Considerations 12

7. ***Reliability***

One of the main goals of building distributed systems is improvement of reliability.

Availability**:** If machines go down, the system should work with the reduced amount of resources.

Design Considerations 13

8. ***Fault-tolerance***

Is a main issue related to reliability: the system has to detect faults and act in a reasonable way:

- ***mask the fault****:* continue to work with possibly reduced performance but without loss of data/ information.
- ***fail gracefully****:* react to the fault in a predictable way and possibly stop functionality for a short period, but without loss of data/information.

Design Considerations 14

9. Security

Security of information resources:

1. ***Confidentiality***

Protection against disclosure to unauthorized person.

2. ***Integrity***

Protection against alteration and corruption.

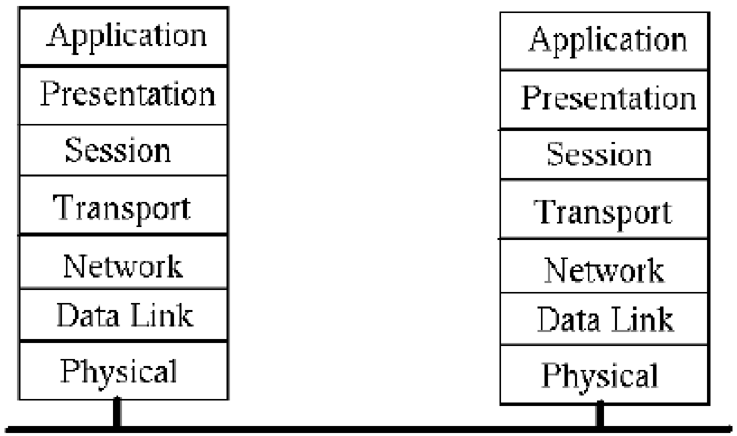
3. ***Availability***

Keep the resource accessible.

Protocol Layers

1) Communications betweens processes takes place using agreed conventions - protocols
2) Network communications requires protocols to cover high-level application communication all the way down to wire communication
3) Complexity handled by encapsulation in protocol layers

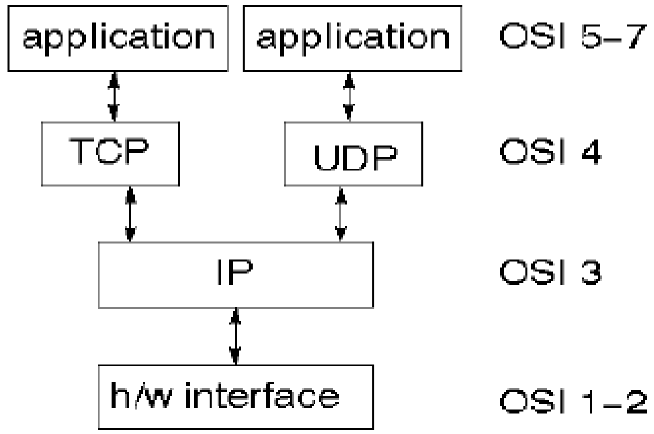
ISO OSI Protocol



OSI layers

1) Network layer provides switching and routing technologies
2) Transport layer provides transparent transfer of data between end systems and is responsible for end-to-end error recovery and flow control
3) Session layer establishes, manages and terminates connections between applications.
4) Presentation layer provides independence from differences in data representation (e.g. encryption)
5) Application layer supports application and end-user processes

TCP/IP Protocol



Port and Socket

1) port

a) conduit into a computer through which information flows and assigned a unique number

b) usually port numbers 0 to 1023 are reserved for special purposes (e.g. HTTP – 80, FTP – 21, SMTP – 25

c) TCP/IP-based computer is identified by a pair of IP address and Port number

Port and Socket 2

2) socket

a) a socket is one end of a process that an application is using to communicate

b) defined by two addresses: the IP address of the host computer; and the port address of the application or process running on the host

Connection Models

There are two types of connection models:

1) Connection oriented

2) Connectionless

- Connection oriented transports may be established on top of connectionless ones –TCP over IP
- Connectionless transports my be established on top of connection oriented ones – HTTP over TCP

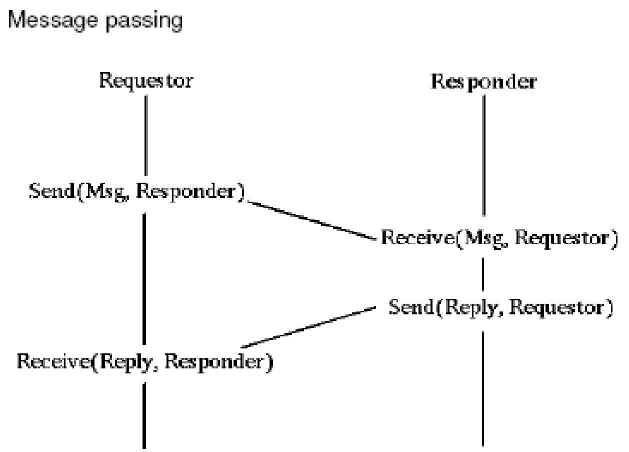
Connection oriented

1) A single connection is established for the session
2) Two-way communications flow along the connection
3) When the session is over, the connection is broken
4) The analogy is to a phone conversation
5) An example is TCP

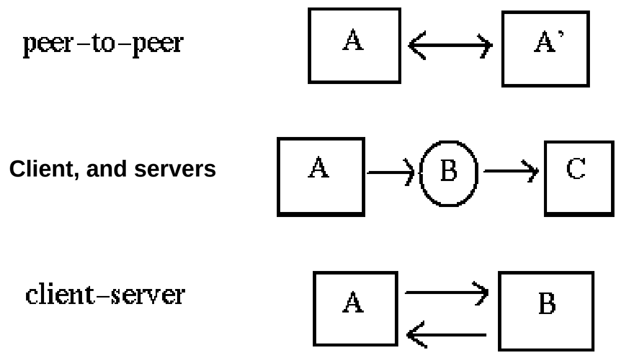
Connectionless

1) In a connectionless system, messages are sent independent of each other
2) Ordinary mail is the analogy
3) Connectionless messages may arrive out of order
4) An example is the IP protocol

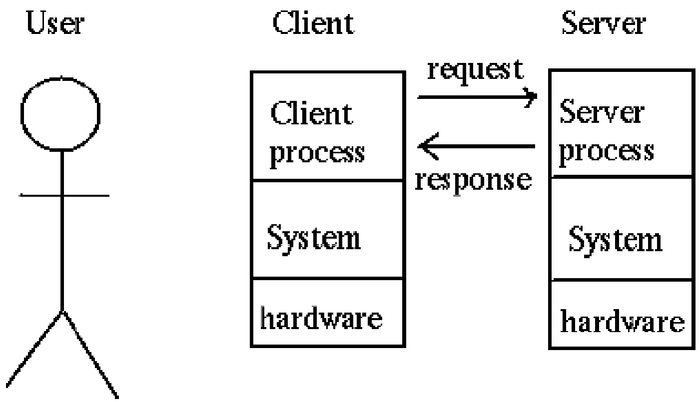
Communications Model



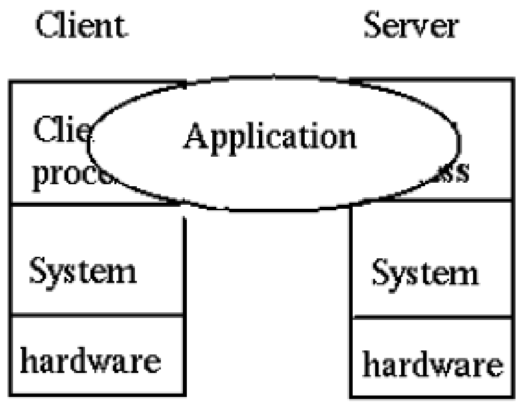
Distributed Computing Models



Client/Server System



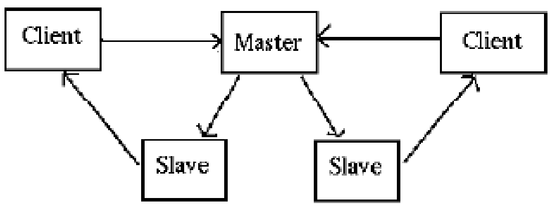
Client/Server Application



Server Distribution 1

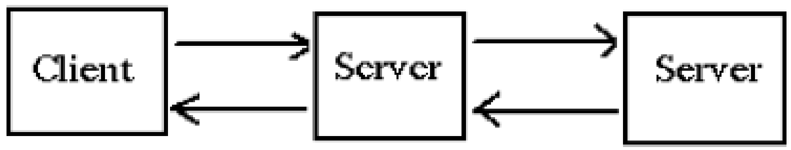
Single client, single server

multiple clients, single server



Server Distribution 2

single client, multiple servers



multiple clients, multiple servers

Middleware

1) intermediate layers between client and server

2) what exactly is it?

   a)a vague term that covers all the distributed software needed to support interactions between client and server

3) where does the middleware start and where does it end?

   a)It starts with the API set on the client side that is used to invoke a service, and it covers the transmission of the request over the network and the resulting response"

Middleware 2

1) The network services include things like TCP/IP

2) The middleware layer is application-independent s/w using the network services

3) Examples of middleware are: DCE, RPC, Corba

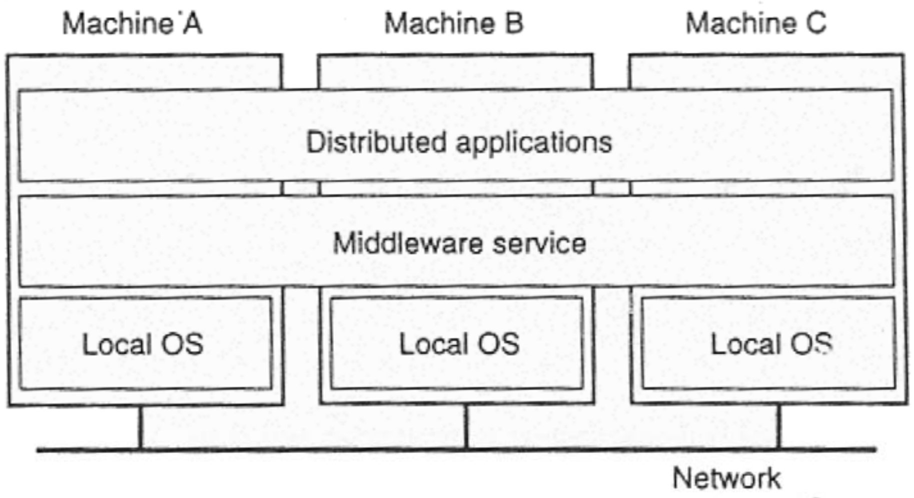
4) Middleware may only perform one function (such as RPC) or a many (such as DCE, Network OS)

DCE: Distributed Computing Environment

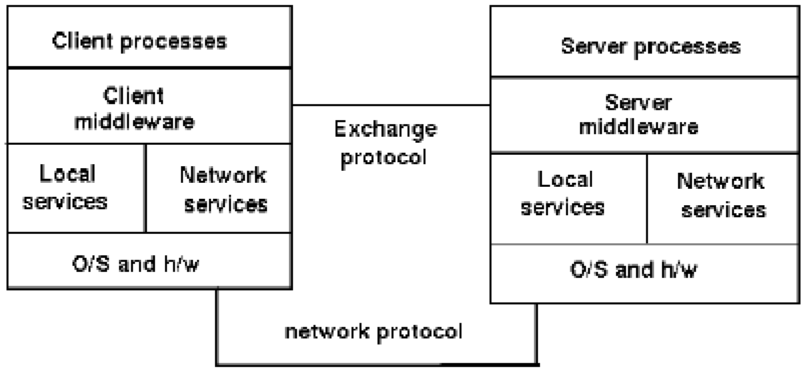
RPC: Remote Procedure Call

CORBA: Common Object Request Broker Architecture

Middleware 3



Middleware Model



Example: Middleware

1) Primitive services such as terminal emulators, file transfer, email, or RPC
2) Integrated services such as DCE, Network O/S
   - Distributed object services such as CORBA, OLE/ActiveX
   - Mobile object services such as RMI (Remote Message Invocator)
   - World Wide Web

Middleware Functions

1) Initiation of processes at different computers
   - Session management
2) Directory services to allow clients to locate servers
   - remote data access
   - Concurrency control to allow servers to handle multiple clients
   - Security and integrity
   - Monitoring, and maintaining sessions
   - Termination of processes both local and remote

Thanks,
See you next Week, isA