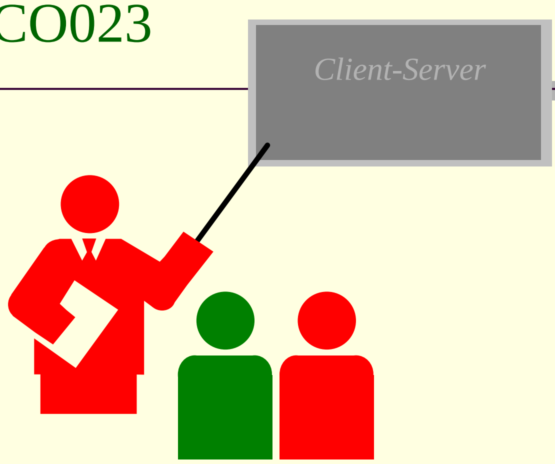
Modulo II – Introdução Sistemas Distribuídos

Prof. Ismael H F Santos

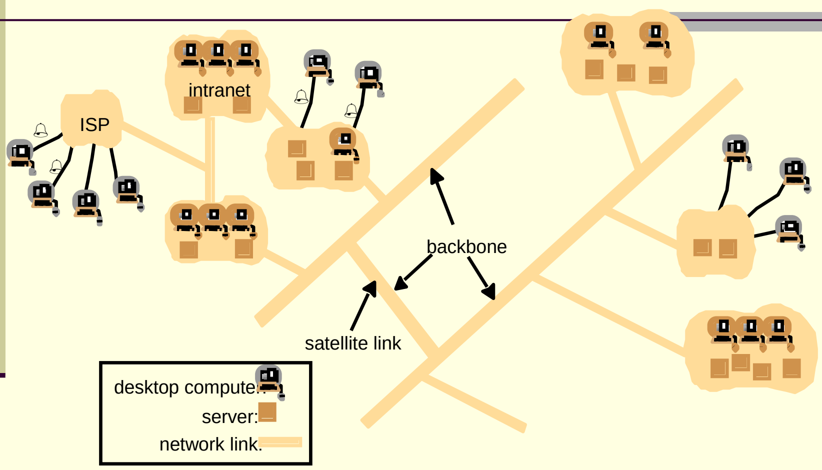
Ementa

- Sistemas Distribuídos
  - *Cliente-Servidor*

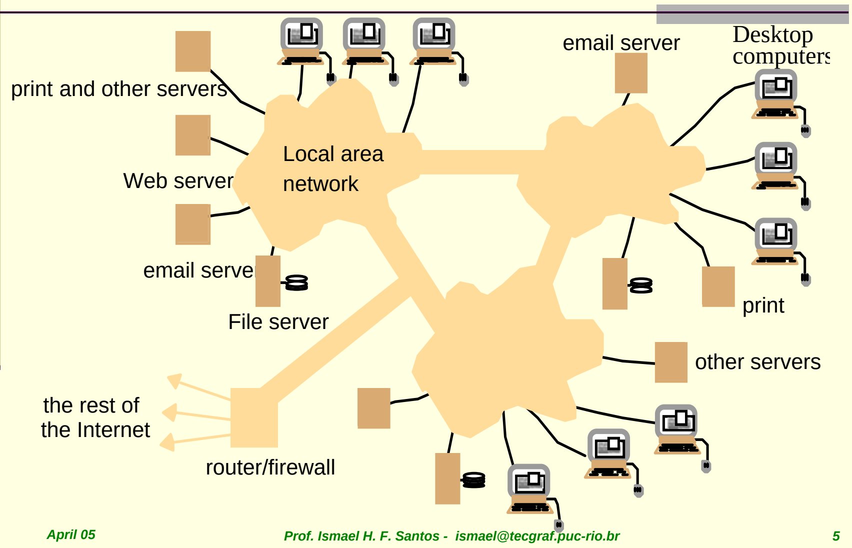
SCD – CO023



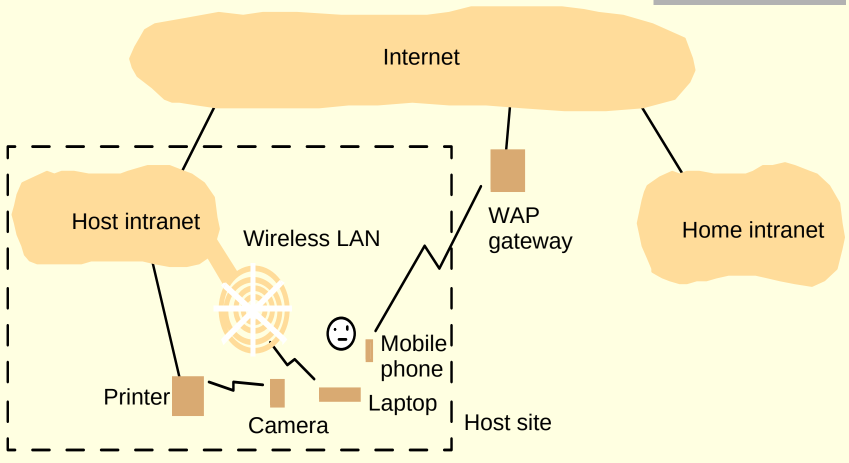
A typical portion of the Internet



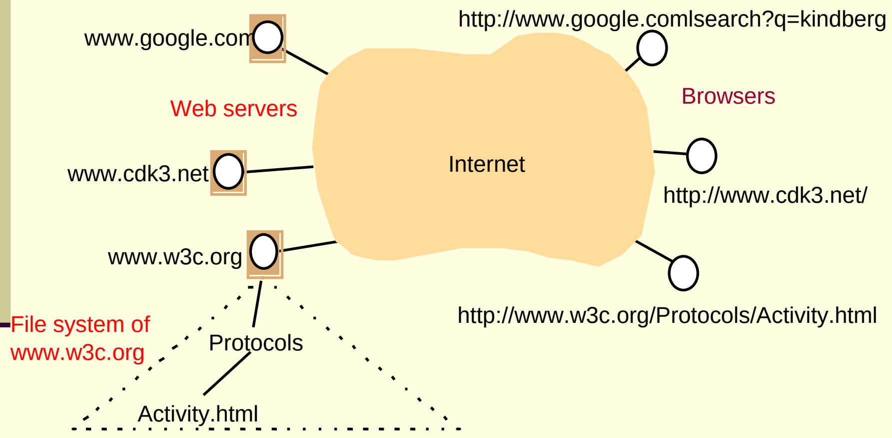
A typical intranet



Portable and handheld devices in a distributed system



Web servers and web browsers



Computers in the Internet

Date	*Computers*	*Web servers*
1979, Dec.	188	0
1989, July	130,000	0
1999, July	56,218,000	5,560,866

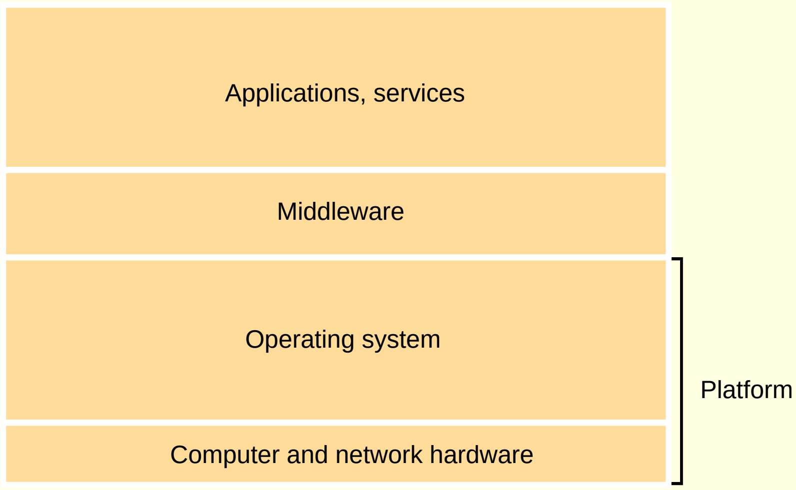
Computers vs. Web servers in the Internet

Date	*Computers*	*Web servers*	*Percentage*
1993, July	1,776,000	130	0.008
1995, July	6,642,000	23,500	0.4
1997, July	19,540,000	1,203,096	6
1999, July	56,218,000	6,598,697	12

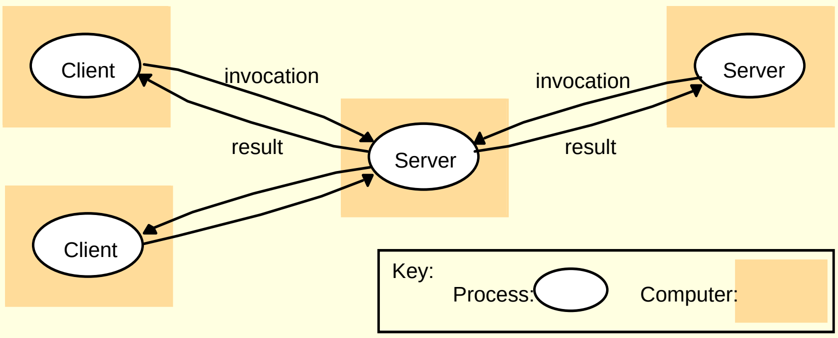
Transparencies

Access transparency: enables local and remote resources to be accessed using identical operations.
Location transparency: enables resources to be accessed without knowledge of their location.
Concurrency transparency: enables several processes to operate concurrently using shared resources without interference between them.
Replication transparency: enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers.
Failure transparency: enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.
Mobility transparency: allows the movement of resources and clients within a system without affecting the operation of users or programs.
Performance transparency: allows the system to be reconfigured to improve performance as loads vary.
Scaling transparency: allows the system and applications to expand in scale without change to the system structure or the application algorithms.

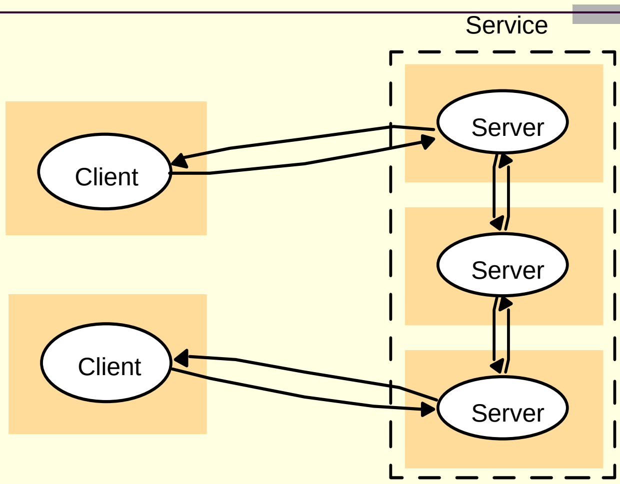
Software and hardware service layers in distributed systems



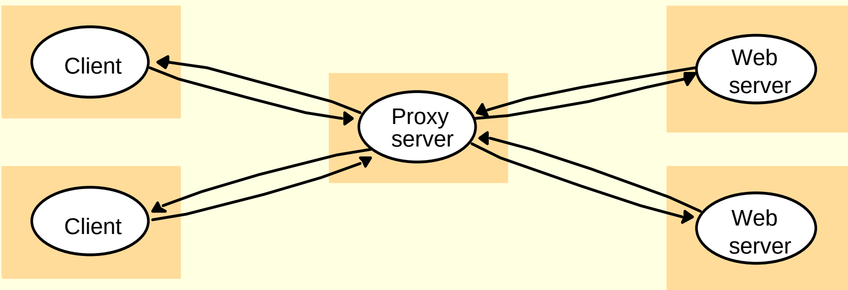
Clients invoke individual servers



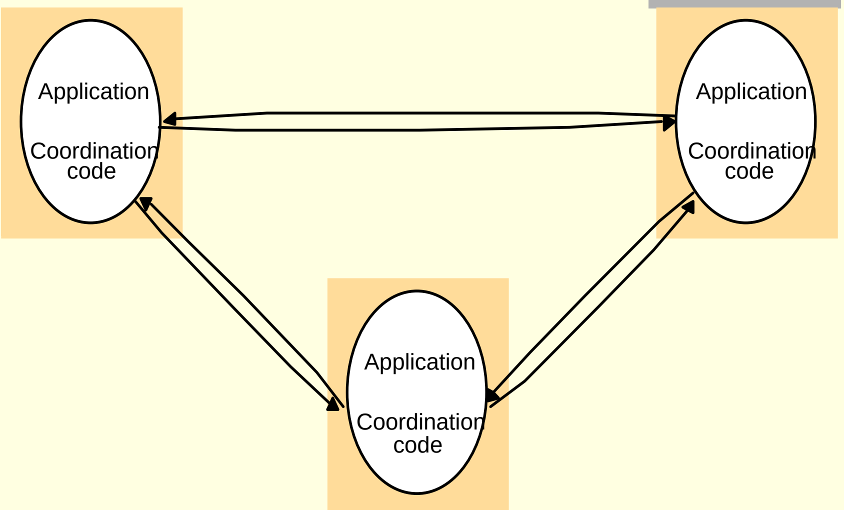
A service provided by multiple servers



Web proxy server

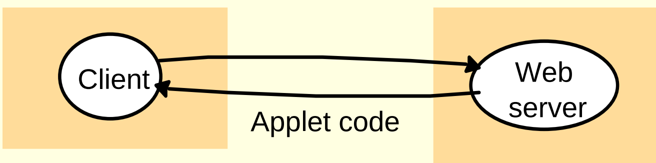


A distributed application based on peer processes



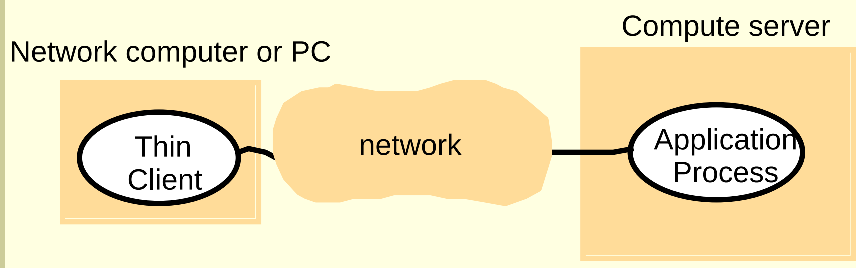
Web applets

a) client request results in the downloading of applet code

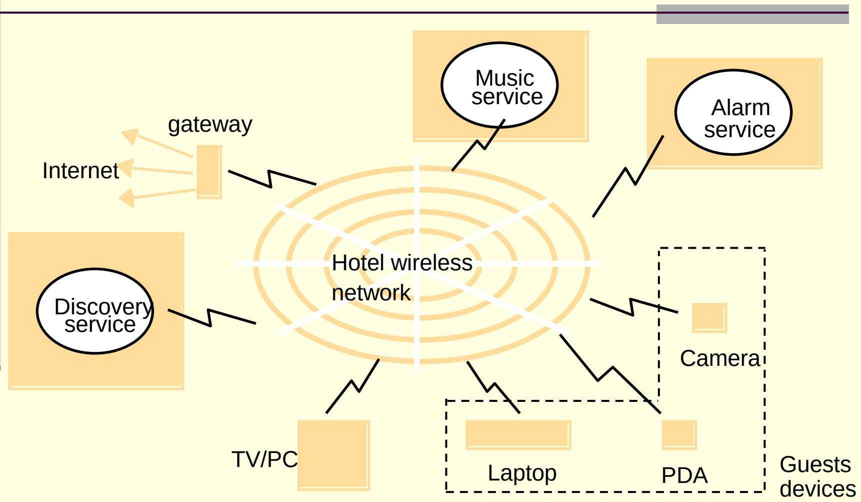


b) client interacts with the applet

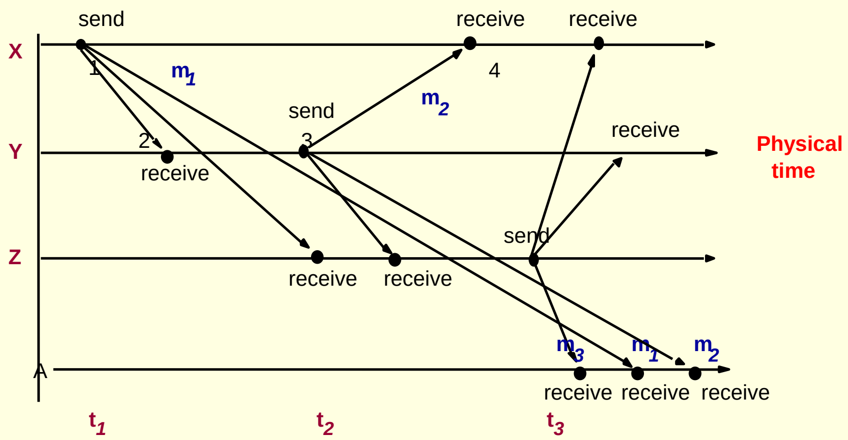
Thin clients and compute servers



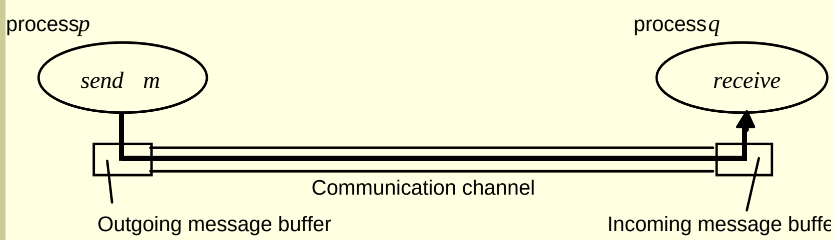
Spontaneous networking in a hotel



Real-time ordering of events



Processes and channels



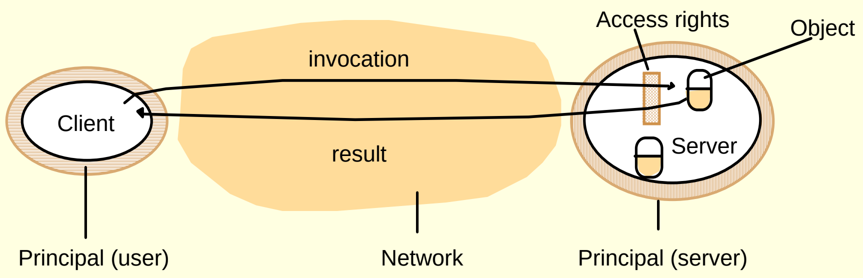
Omission and arbitrary failures

Class of failure	*Affects*	*Description*
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Process	A process completes a *send*, but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process or channel	Process/channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.

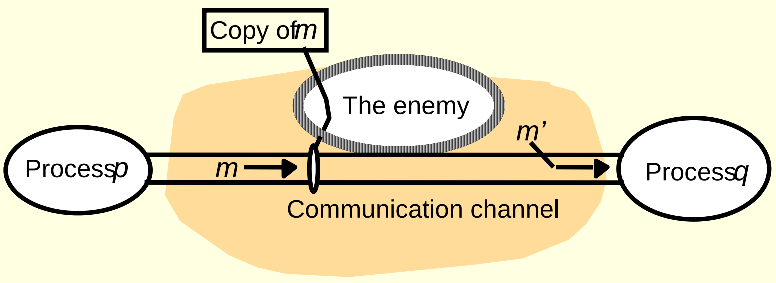
Timing failures

Class of Failure	*Affects*	*Description*
Clock	Process	Process's local clock exceeds the bounds on its rate of drift from real time.
Performance	Process	Process exceeds the bounds on the interval between two steps.
Performance	Channel	A message's transmission takes longer than the stated bound.

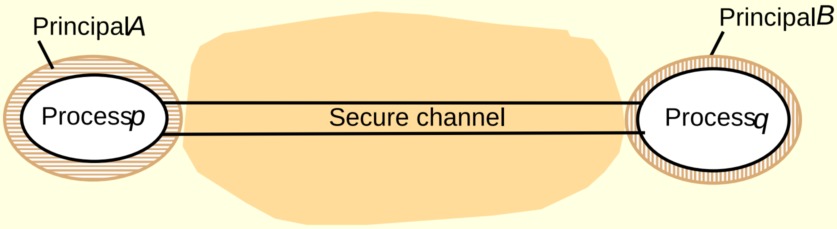
Objects and principals



The enemy

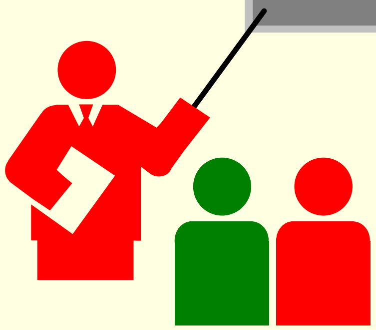


Secure channels

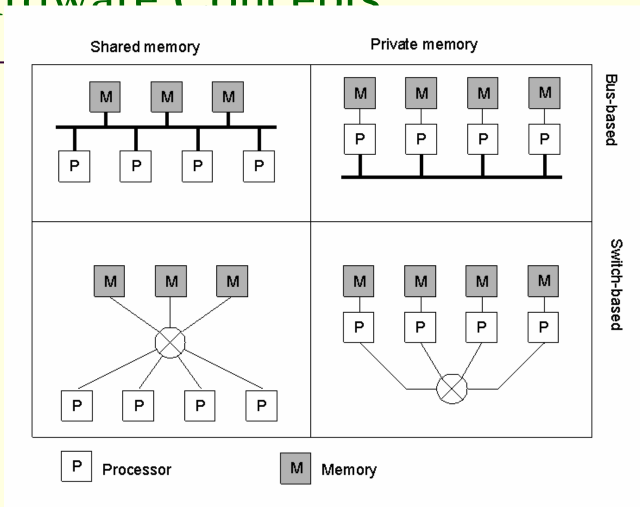


SCD – CO023

Hardware Concepts

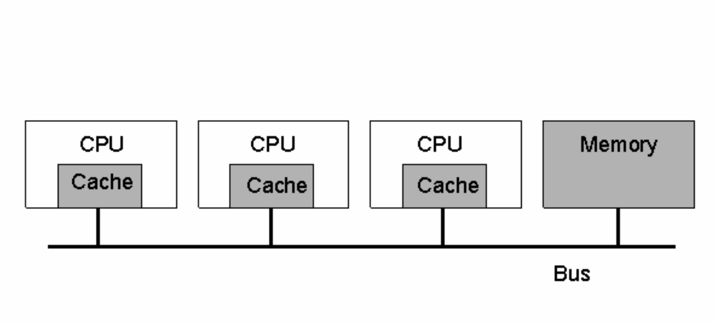


Hardware Concepts

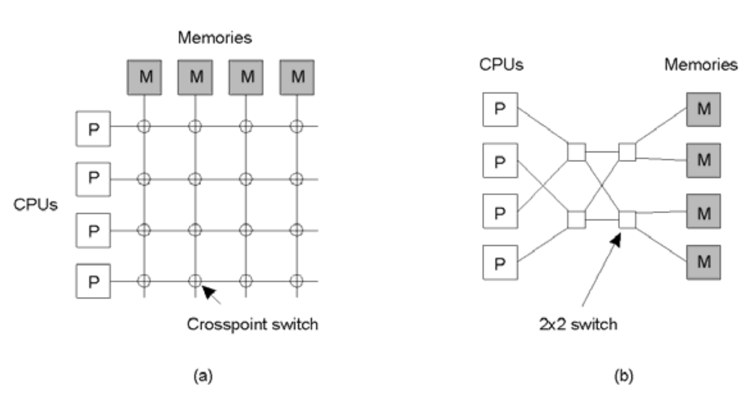


Different basic organizations and memories in distributed computer systems

Multiprocessors (1)

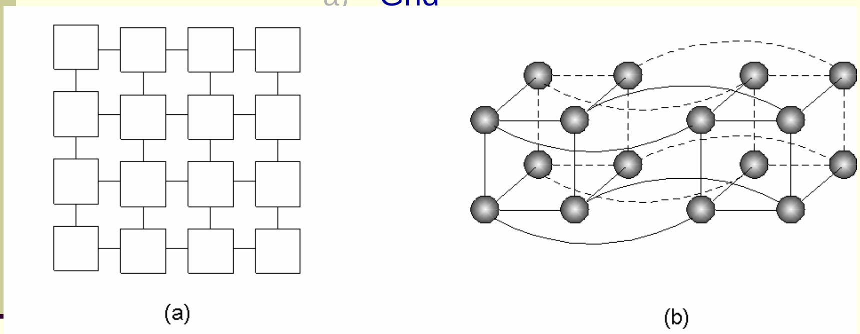


Multiprocessors (2)



Homogeneous Multicomputer Systems

a) Grid

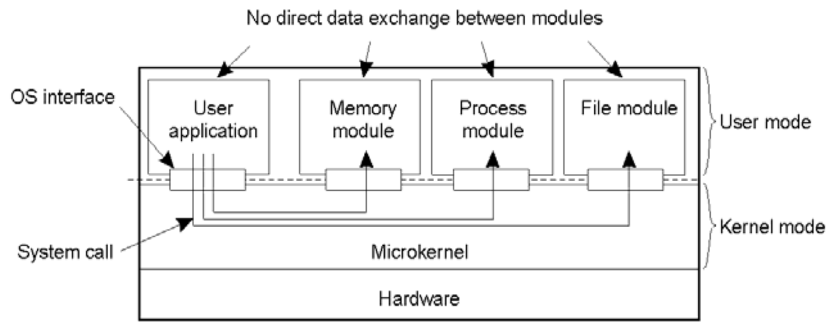


Software Concepts

System	Description	Main Goal
DOS	Tightly-coupled operating system for multi-processors and homogeneous multicomputers	Hide and manage hardware resources
NOS	Loosely-coupled operating system for heterogeneous multicomputers (LAN and WAN)	Offer local services to remote clients
Middleware	Additional layer atop of NOS implementing general-purpose services	Provide distribution transparency

- An overview of
- DOS (Distributed Operating Systems)
- NOS (Network Operating Systems)
- Middleware

Uniprocessor Operating Systems



Multiprocessor Operating Systems (1)

- A monitor to protect an integer against concurrent access.

```
monitor Counter {
private:
 int count = 0;
public:
  int value() { return count;}
  void incr () { count = count + 1;}
  void decr() { count = count – 1;}
}
```

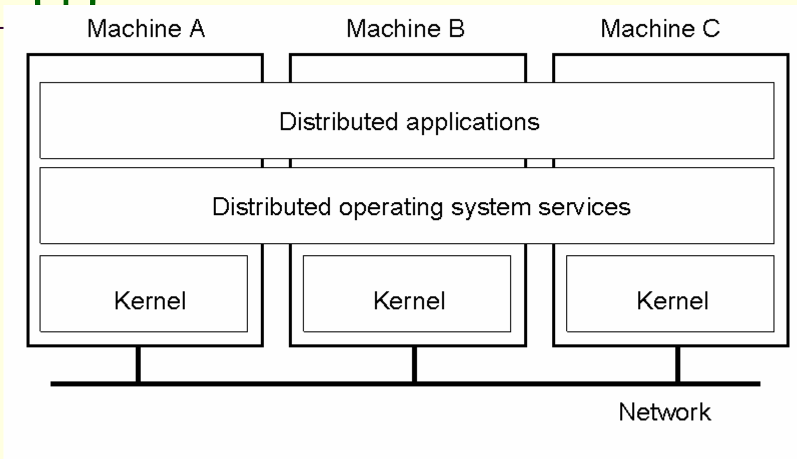

Multiprocessor Operating Systems (2)

- A monitor to protect an integer against concurrent access, but blocking a process.

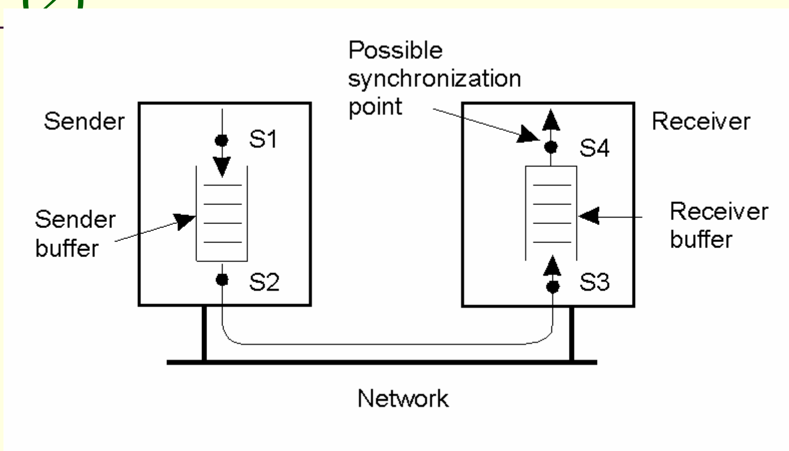
```
monitor Counter {
private:
  int count = 0;
  int blocked_procs = 0;
  condition unblocked;
public:
  int value () { return count;}
  void incr () {
     if (blocked_procs == 0)
      count = count + 1;
     else
      signal (unblocked);
}
```

```
void decr() {
if (count ==0) {
  blocked_procs = blocked_procs + 1;
  wait (unblocked);
  blocked_procs = blocked_procs – 1;
  }
  else
   count = count – 1;
 }
}
```

Multicomputer Operating Systems (1)



Multicomputer Operating Systems (2)



Multicomputer Operating Systems (3)

Synchronization point	Send buffer	Reliable comm. guaranteed?
Block sender until buffer not full	Yes	Not necessary
Block sender until message sent	No	Not necessary
Block sender until message received	No	Necessary
Block sender until message delivered	No	Necessary

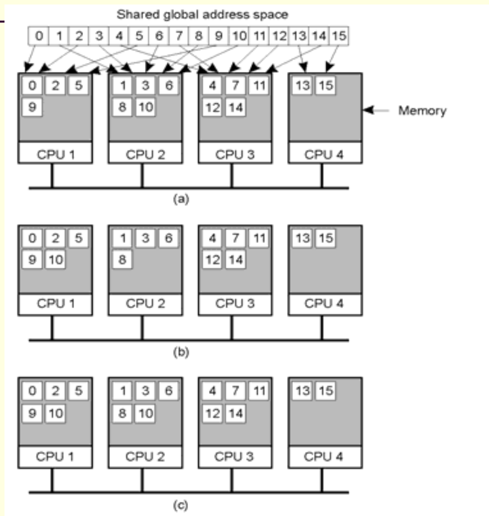
- Relation between blocking, buffering, and reliable communications.

Distributed Shared Memory Systems (1)

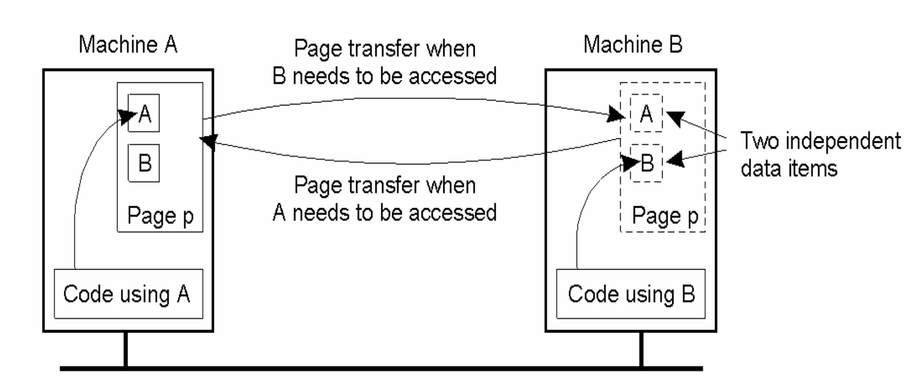
a) Pages of address space distributed among four machines

b) Situation after CPU 1 references page 10

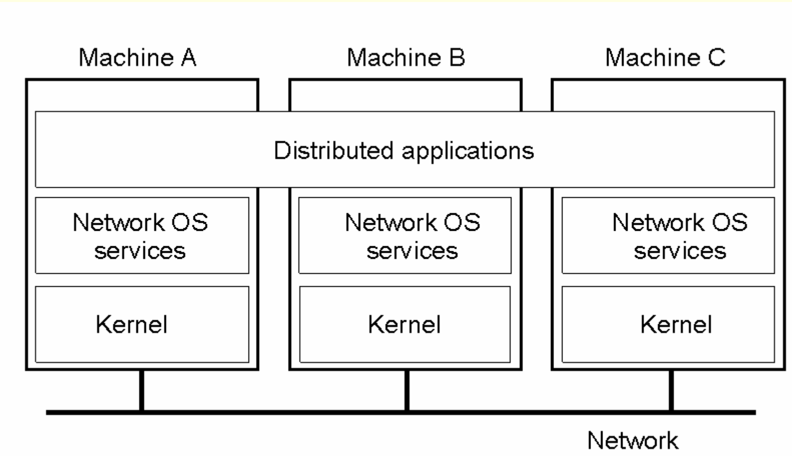
c) Situation if page 10 is read only and replication is used



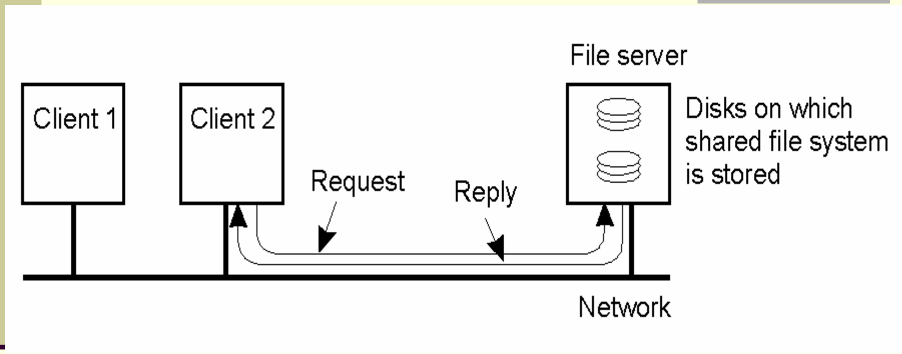
Distributed Shared Memory Systems (2)



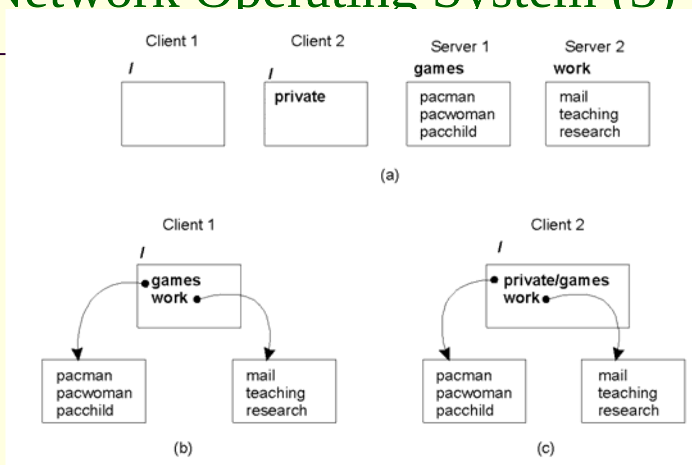
Network Operating System (1)



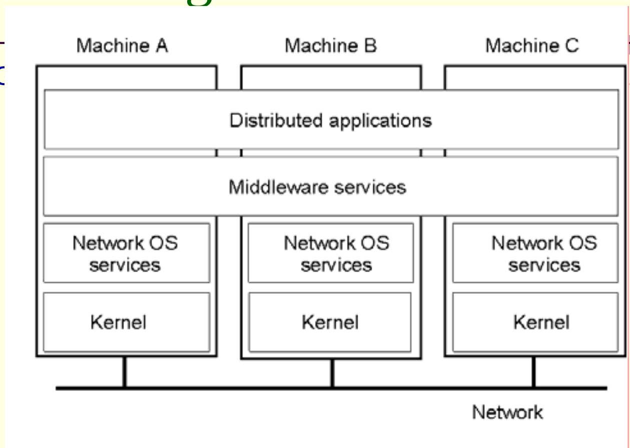
Network Operating System (2)



Network Operating System (3)

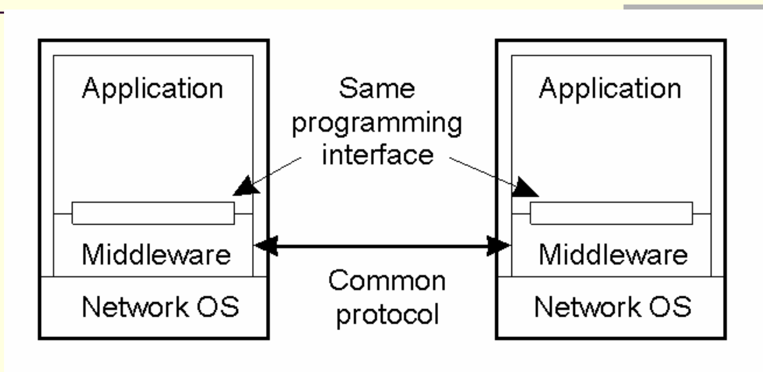


Positioning Middleware



- C re.

Middleware and Openness



- In an open middleware-based distributed system, the protocols used by each middleware layer should be the same, as well as the interfaces they offer to applications.

Comparison between Systems

- A comparison between multiprocessor operating systems, multicomputer operating systems, network operating systems, and middleware based distributed systems.

Item	Distributed OS		Network OS	Middleware-based OS
	Multiproc.	Multicomp.		
Degree of transparency	Very High	High	Low	High
Same OS on all nodes	Yes	Yes	No	No
Number of copies of OS	1	N	N	N
Basis for communication	Shared memory	Messages	Files	Model specific
Resource management	Global, central	Global, distributed	Per node	Per node
Scalability	No	Moderately	Yes	Varies
Openness	Closed	Closed	Open	Open

An Example Client and Server (1)

- The

```
/* Definitions needed by clients and servers.                    */
#define TRUE              1
#define MAX_PATH          255   /* maximum length of file name        */
#define BUF_SIZE          1024  /* how much data to transfer at once  */
#define FILE_SERVER       243   /* file server's network address      */

/* Definitions of the allowed operations */
#define CREATE            1     /* create a new file                  */
#define READ              2     /* read data from a file and return it */
#define WRITE             3     /* write data to a file               */
#define DELETE            4     /* delete an existing file            */

/* Error codes. */
#define OK                0     /* operation performed correctly      */
#define E_BAD_OPCODE      -1    /* unknown operation requested        */
#define E_BAD_PARAM       -2    /* error in a parameter               */
#define E_IO              -3    /* disk error or other I/O error      */

/* Definition of the message format. */
struct message {
    long source;                /* sender's identity                  */
    long dest;                  /* receiver's identity                */
    long opcode;                /* requested operation                */
    long count;                 /* number of bytes to transfer        */
    long offset;                /* position in file to start I/O      */
    long result;                /* result of the operation            */
    char name[MAX_PATH];        /* name of file being operated on     */
    char data[BUF_SIZE];        /* data to be read or written         */
};
```

An Example Client and Server (2)

```
#include <header.h>
void main(void) {
    struct message ml, m2;                /* incoming and outgoing messages    */
    int r;                                /* result code                       */

    while(TRUE) {                         /* server runs forever               */
        receive(FILE_SERVER, &ml);        /* block waiting for a message       */
        switch(ml.opcode) {               /* dispatch on type of request       */
            case CREATE:   r = do_create(&ml, &m2); break;
            case READ:     r = do_read(&ml, &m2); break;
            case WRITE:    r = do_write(&ml, &m2); break;
            case DELETE:   r = do_delete(&ml, &m2); break;
            default:       r = E_BAD_OPCODE;
        }
        m2.result = r;                    /* return result to client           */
        send(ml.source, &m2);             /* send reply                        */
    }
}
```

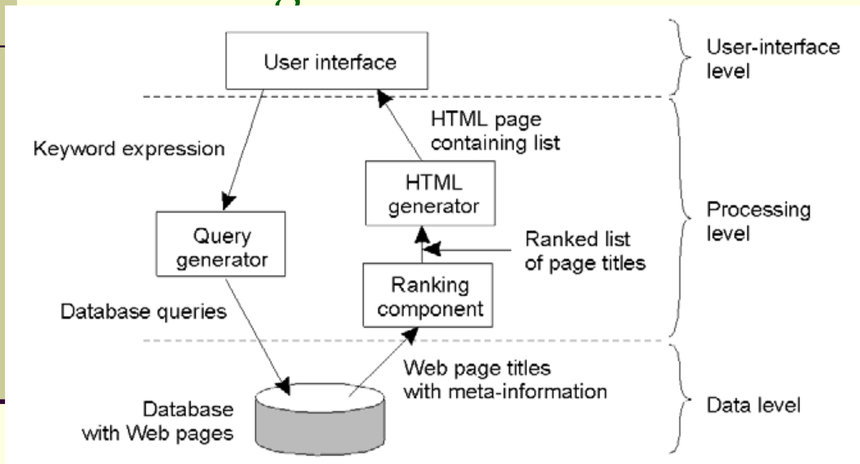
- A sample server.

An Example Client and Server (3)

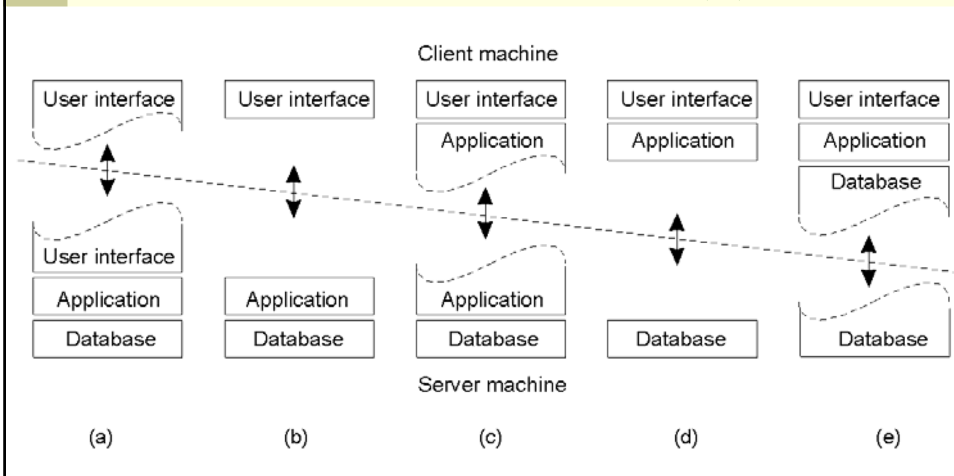
```
#include <header.h>
int copy(char *src, char *dst){          /* procedure to copy file using the server   */
    struct message ml;                   /* message buffer                            */
    long position;                       /* current file position                     */
    long client = 110;                   /* client's address                          */

    initialize( );                       /* prepare for execution                     */
    position = 0;
    do {
        ml.opcode = READ;                /* operation is a read                       */
        ml.offset = position;            /* current position in the file              */
        ml.count  = BUF_SIZE;                                                      /* how many bytes to read*/
        strcpy(&ml.name, src);           /* copy name of file to be read to message   */
        send(FILESERVER, &ml);           /* send the message to the file server       */
        receive(client, &ml);            /* block waiting for the reply               */

        /* Write the data just received to the destination file.                      */
        ml.opcode = WRITE;               /* operation is a write                      */
        ml.offset = position;            /* current position in the file              */
        ml.count  = ml.result;           /* how many bytes to write                   */
        strcpy(&ml.name, dst);           /* copy name of file to be written to buf    */
        send(FILE_SERVER, &ml);          /* send the message to the file server       */
        receive(client, &ml);            /* block waiting for the reply               */
        position += ml.result;           /* ml.result is number of bytes written      */
    } while( ml.result > 0 );            /* iterate until done                        */
    return(ml.result >= 0 ? OK : ml result);  /* return OK or error code              */
}
```

- A client using the server to copy a file.

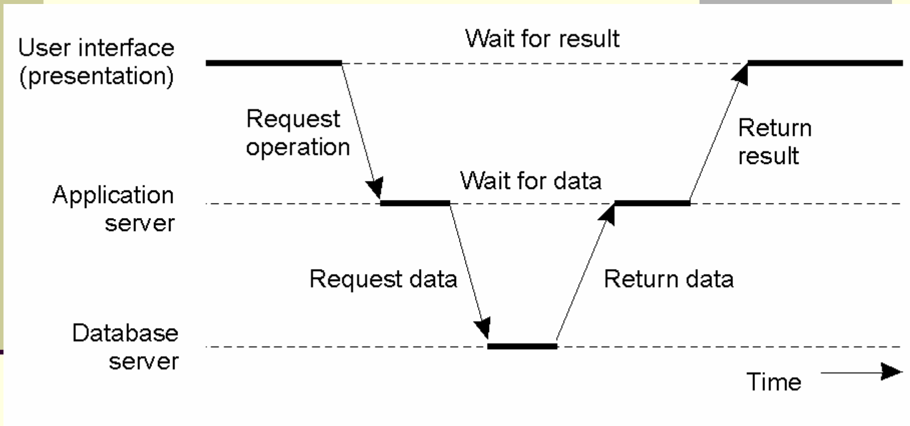
Processing Level



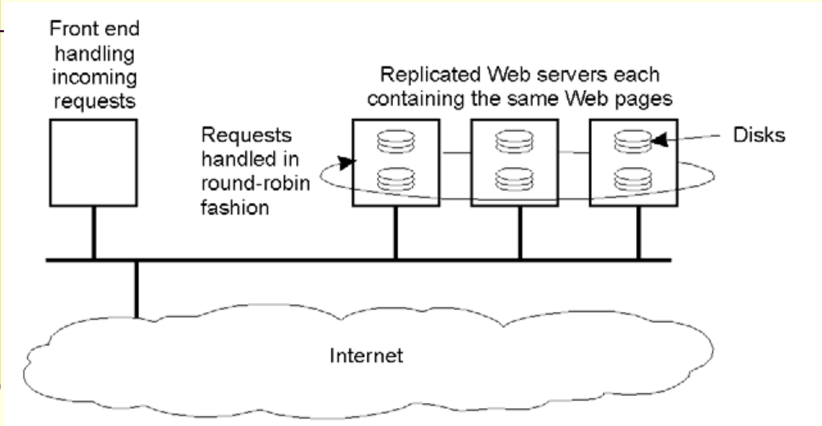
Multitiered Architectures (1)



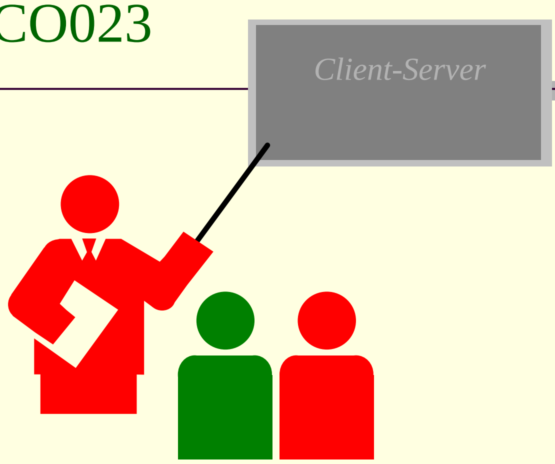
Multitiered Architectures (2)



Modern Architectures



SCD – CO023



Module 16: Network Structures

- Motivation
- Types of Distributed Operating Systems
- Network Structure
- Network Topology
- Communication Structure
- Communication Protocols
- Robustness
- Design Issues
- An Example: Networking
- Design Strategies

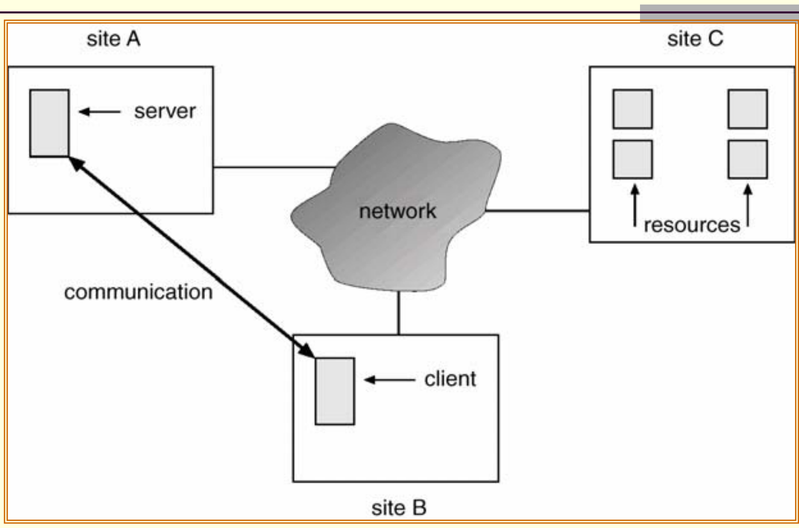
Chapter Objectives

- To provide a high-level overview of distributed systems and the networks that interconnect them
- To discuss the general structure of distributed operating systems

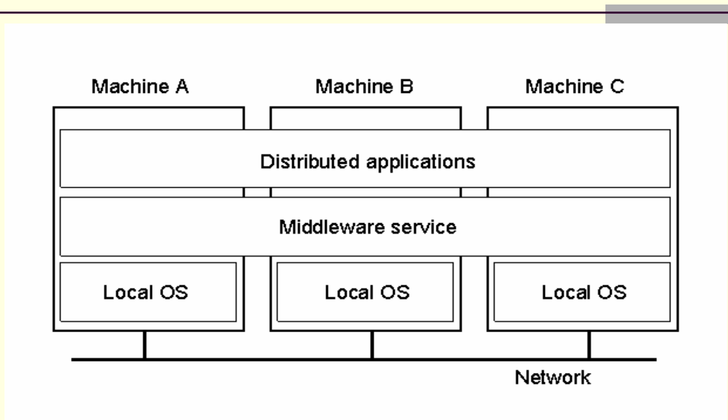
Motivation

- **Distributed system** is collection of loosely coupled processors interconnected by a communications network
- Processors variously called *nodes, computers, machines, hosts*
  - *Site* is location of the processor
- Reasons for distributed systems
  - Resource sharing
    - sharing and printing files at remote sites
    - processing information in a distributed database
    - using remote specialized hardware devices
  - Computation speedup – **load sharing**
  - Reliability – detect and recover from site failure, function transfer, reintegrate failed site
  - Communication – message passing

A Distributed System



Definition of a Distributed System (2)



A distributed system organized as middleware.
Note that the middleware layer extends over multiple machines.

Scalability Problems

Concept	Example
Centralized services	A single server for all users
Centralized data	A single on-line telephone book
Centralized algorithms	Doing routing based on complete information

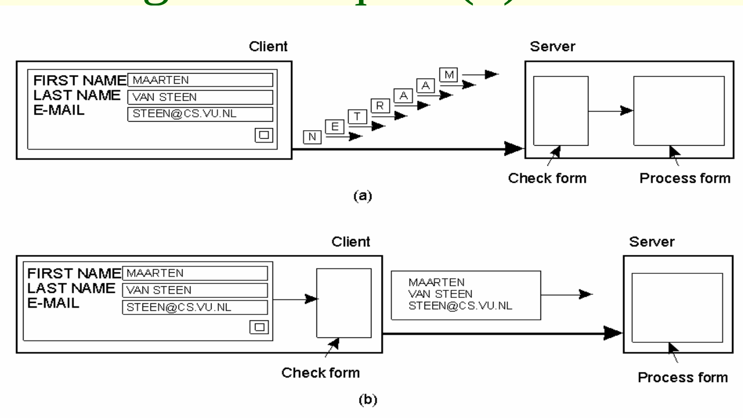
Examples of scalability limitations.

Transparency in a Distributed System

Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource may be shared by several competitive users
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource
Persistence	Hide whether a (software) resource is in memory or on disk

Different forms of transparency in a distributed system.

Scaling Techniques (1)

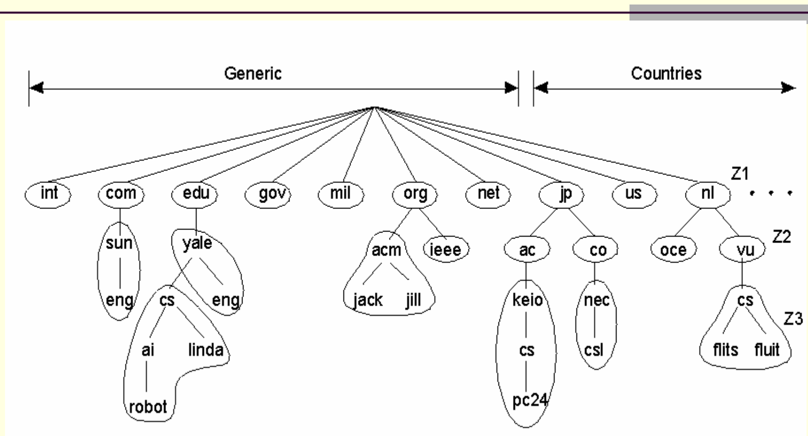


The difference between letting:

a) a server or

b) a client check forms as they are being filled

Scaling Techniques (2)



An example of dividing the DNS name space into zones.

Types of Distributed Operating Systems

- Network Operating Systems
- Distributed Operating Systems

Network-Operating Systems

- Users are aware of multiplicity of machines. Access to resources of various machines is done explicitly by:
  - Remote logging into the appropriate remote machine (telnet, ssh)
  - Transferring data from remote machines to local machines, via the File Transfer Protocol (FTP) mechanism

Distributed-Operating Systems

- Users not aware of multiplicity of machines
  - Access to remote resources similar to access to local resources
- Data Migration – transfer data by transferring entire file, or transferring only those portions of the file necessary for the immediate task
- Computation Migration – transfer the computation, rather than the data, across the system

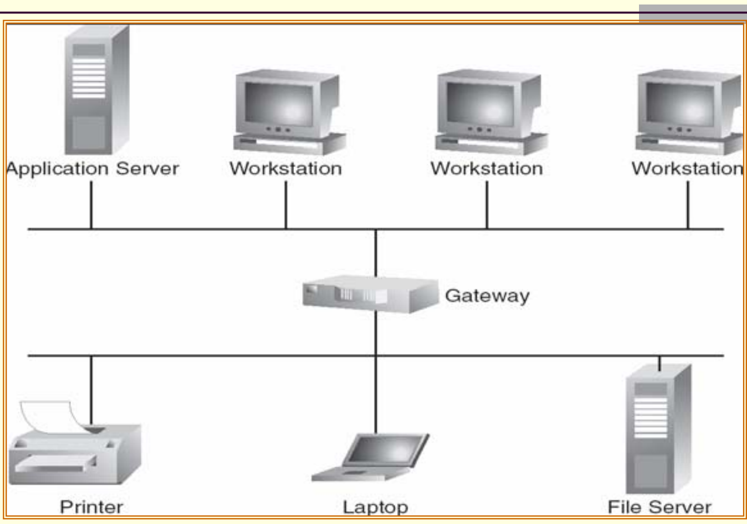
Distributed-Operating Systems (Cont.)

- Process Migration – execute an entire process, or parts of it, at different sites
  - Load balancing – distribute processes across network to even the workload
  - Computation speedup – subprocesses can run concurrently on different sites
  - Hardware preference – process execution may require specialized processor
  - Software preference – required software may be available at only a particular site
  - Data access – run process remotely, rather than transfer all data locally

Network Structure

- Local-Area Network (LAN) – designed to cover small geographical area.
  - Multiaccess bus, ring, or star network
  - Speed ≈ 10 megabits/second, or higher
  - Broadcast is fast and cheap
  - Nodes:
    - usually workstations and/or personal computers
    - a few (usually one or two) mainframes

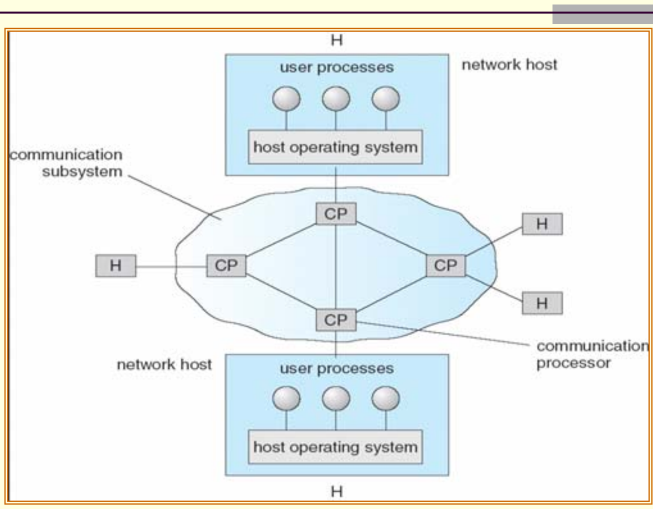
Depiction of typical LAN



Network Types (Cont.)

- Wide-Area Network (WAN) – links geographically separated sites
  - Point-to-point connections over long-haul lines (often leased from a phone company)
  - Speed ≈ 100 kilobits/second
  - Broadcast usually requires multiple messages
  - Nodes:
    - usually a high percentage of mainframes

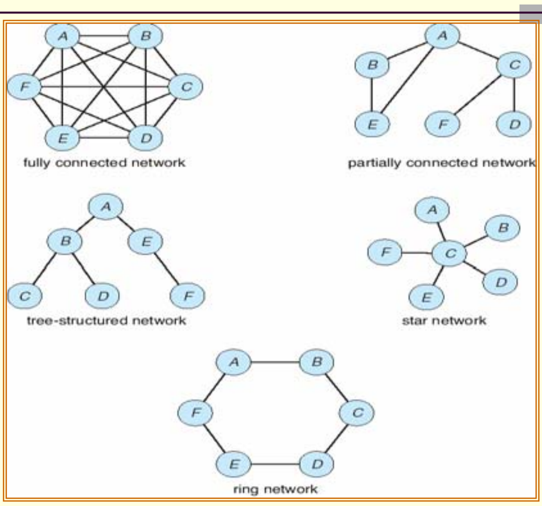
Communication Processors in a Wide-Area Network



Network Topology

- Sites in the system can be physically connected in a variety of ways; they are compared with respect to the following criteria:
  - **Basic cost** - How expensive is it to link the various sites in the system?
  - **Communication cost** - How long does it take to send a message from site *A* to site *B*?
  - **Reliability** - If a link or a site in the system fails, can the remaining sites still communicate with each other?
- The various topologies are depicted as graphs whose nodes correspond to sites
  - An edge from node *A* to node *B* corresponds to a direct connection between the two sites
- The following six items depict various network topologies

Network Topology



Communication Structure

- The design of a communication network must address four basic issues:
  - **Naming and name resolution** - How do two processes locate each other to communicate?
  - **Routing strategies** - How are messages sent through the network?
  - **Connection strategies** - How do two processes send a sequence of messages?
  - **Contention -** The network is a shared resource, so how do we resolve conflicting demands for its use?

Naming and Name Resolution

- Name systems in the network
- Address messages with the process-id
- Identify processes on remote systems by

  <host-name, identifier> pair
- *Domain name service* (DNS) – specifies the naming structure of the hosts, as well as name to address resolution (Internet)

Routing Strategies

- **Fixed routing** - A path from *A* to *B* is specified in advance; path changes only if a hardware failure disables it
  - Since the shortest path is usually chosen, communication costs are minimized
  - Fixed routing cannot adapt to load changes
  - Ensures that messages will be delivered in the order in which they were sent
- **Virtual circuit** - A path from *A* to *B* is fixed for the duration of one session. Different sessions involving messages from *A* to *B* may have different paths
  - Partial remedy to adapting to load changes
  - Ensures that messages will be delivered in the order in which they were sent

Routing Strategies (Cont.)

- **Dynamic routing** - The path used to send a message form site *A* to site *B* is chosen only when a message is sent
  - Usually a site sends a message to another site on the link least used at that particular time
  - Adapts to load changes by avoiding routing messages on heavily used path
  - Messages may arrive out of order
    - This problem can be remedied by appending a sequence number to each message

Connection Strategies

- **Circuit switching** - A permanent physical link is established for the duration of the communication (i.e., telephone system)
- **Message switching** - A temporary link is established for the duration of one message transfer (i.e., post-office mailing system)
- **Packet switching** - Messages of variable length are divided into fixed-length packets which are sent to the destination
  - Each packet may take a different path through the network
  - The packets must be reassembled into messages as they arrive
- Circuit switching requires setup time, but incurs less

Contention

- Several sites may want to transmit information over a link simultaneously. Techniques to avoid repeated collisions include:
  - **CSMA/CD** - Carrier sense with multiple access (CSMA); collision detection (CD)
    - A site determines whether another message is currently being transmitted over that link. If two or more sites begin transmitting at exactly the same time, then they will register a CD and will stop transmitting
    - When the system is very busy, many collisions may occur, and thus performance may be degraded
    - CSMA/CD is used successfully in the Ethernet

Contention (Cont.)

- **Token passing** - A unique message type, known as a token, continuously circulates in the system (usually a ring structure)
  - A site that wants to transmit information must wait until the token arrives
  - When the site completes its round of message passing, it retransmits the token
  - A token-passing scheme is used by some IBM and HP/Apollo systems
- **Message slots** - A number of fixed-length message slots continuously circulate in the system (usually a ring structure)
  - Since a slot can contain only fixed-sized messages, a single logical message may have to be broken down into a number of smaller packets, each of

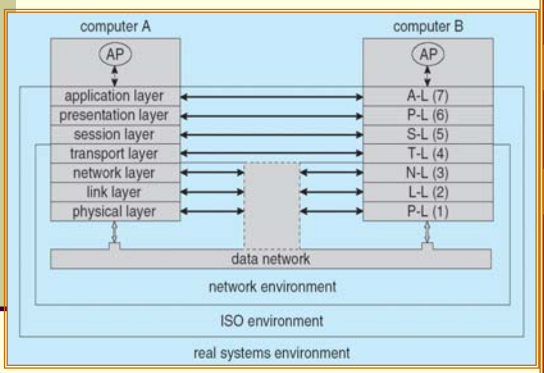
Communication Protocol

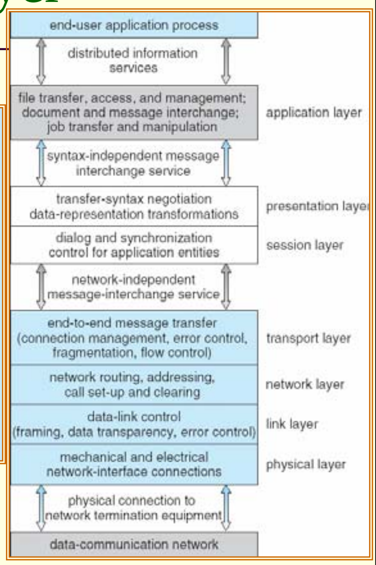
- The communication network is partitioned into the following multiple layers:
  - **Physical layer** – handles the mechanical and electrical details of the physical transmission of a bit stream
  - **Data-link layer** – handles the *frames*, or fixed-length parts of packets, including any error detection and recovery that occurred in the physical layer
  - **Network layer** – provides connections and routes packets in the communication network, including handling the address of outgoing

Communication Protocol (Cont.)

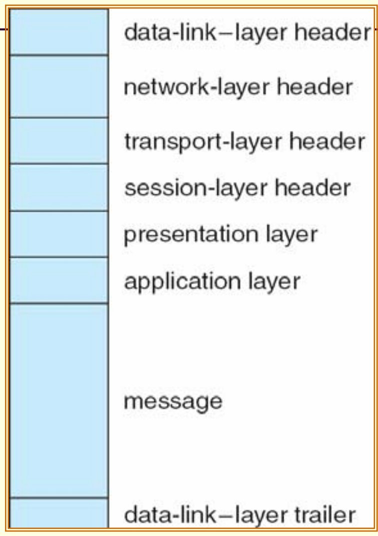
- **Transport layer** – responsible for low-level network access and for message transfer between clients, including partitioning messages into packets, maintaining packet order, controlling flow, and generating physical addresses
- **Session layer** – implements sessions, or process-to-process communications protocols
- **Presentation layer** – resolves the differences in formats among the various sites in the network, including character conversions, and half duplex/full duplex (echoing)
- **Application layer** – interacts directly with the

The ISO Protocol Layer

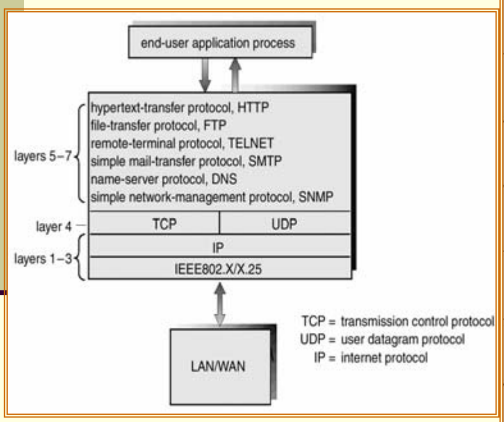




The ISO Network Message



The TCP/IP Protocol Layers



ISO	TCP/IP
Application	HTTP, DNS, Telnet SMTP, FTP
Presentation	Not Defined
Session	Not Defined
Transport	TCP-UDP
Network	IP
Data Link	Not Defined
Physical	Not Defined

Robustness

- Failure detection
- Reconfiguration

Failure Detection

- Detecting hardware failure is difficult
- To detect a link failure, a handshaking protocol can be used
- Assume Site A and Site B have established a link
  - At fixed intervals, each site will exchange an *I-am-up* message indicating that they are up and running
- If Site A does not receive a message within the fixed interval, it assumes either (a) the other site is not up or (b) the message was lost
- Site A can now send an *Are-you-up?* message to Site B
- If Site A does not receive a reply, it can repeat the message or try an alternate route to Site B

Failure Detection (cont)

- If Site A does not ultimately receive a reply from Site B, it concludes some type of failure has occurred
- Types of failures
  - Site B is down
  - The direct link between A and B is down
  - The alternate link from A to B is down
  - The message has been lost
- However, Site A cannot determine exactly **why** the failure has occurred

Reconfiguration

- When Site A determines a failure has occurred, it must reconfigure the system:
  1. If the link from A to B has failed, this must be broadcast to every site in the system
  2. If a site has failed, every other site must also be notified indicating that the services offered by the failed site are no longer available
- When the link or the site becomes available again, this information must again be broadcast to all other sites

Design Issues

- **Transparency** – the distributed system should appear as a conventional, centralized system to the user
- **Fault tolerance** – the distributed system should continue to function in the face of failure
- **Scalability** – as demands increase, the system should easily accept the addition of new resources to accommodate the increased demand
- **Clusters** – a collection of semi-autonomous machines that acts as a single system

Example: Networking

- The transmission of a network packet between hosts on an Ethernet network
- Every host has a unique IP address and a corresponding Ethernet (MAC) address
- Communication requires both addresses
- Domain Name Service (DNS) can be used to acquire IP addresses
- Address Resolution Protocol (ARP) is used to map MAC addresses to IP addresses
- If the hosts are on the same network, ARP can be used

An Ethernet Packet

bytes		
7	preamble—start of packet	each byte pattern 10101010
1	start of frame delimiter	pattern 10101011
2 or 6	destination address	Ethernet address or broadcast
2 or 6	source address	Ethernet address
2	length of data section	length in bytes
0–1500	data	message data
0–46	pad (optional)	message must be > 63 bytes long
4	frame checksum	for error detection

Design Strategies

- The communication network is partitioned into the following multiple layers
  - **Physical layer** – handles the mechanical and electrical details of the physical transmission of a bit stream.
  - **Data-link layer** – handles the *frames*, or fixed-length parts of packets, including any error detection and recovery that occurred in the physical layer.
  - **Network layer** – provides connections and routes packets in the communication network, including handling the address of outgoing packets, decoding the address of incoming packets, and

Design Strategies (Cont.)

- **Transport layer** – responsible for low-level network access and for message transfer between clients, including partitioning messages into packets, maintaining packet order, controlling flow, and generating physical addresses.
- **Session layer** – implements sessions, or process-to-process communications protocols.
- **Presentation layer** – resolves the differences in formats among the various sites in the network, including character conversions, and half duplex/full duplex (echoing).
- **Application layer** – interacts directly with the users' deals with file transfer, remote-login protocols and electronic mail, as well as schemas for distributed **databases.**