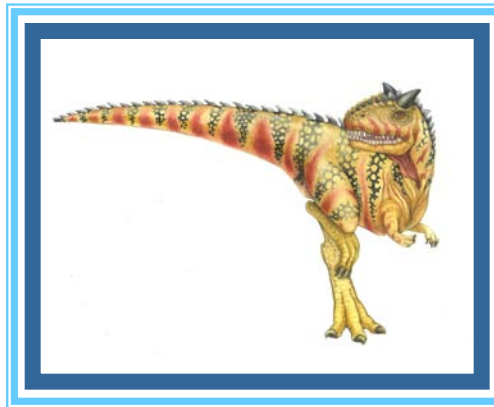# Module 17: Distributed Systems

Chapter 17: Distributed Systems

- Motivation

- Types of Network-Based Operating Systems

- Network Structure

- Communication Structure

- Communication Protocols

- TCP/IP

- Robustness

- Design Issues

- Distributed File System

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

- To explain the naming mechanism that provides location transparency and independence

- To describe the various methods for accessing distributed files

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
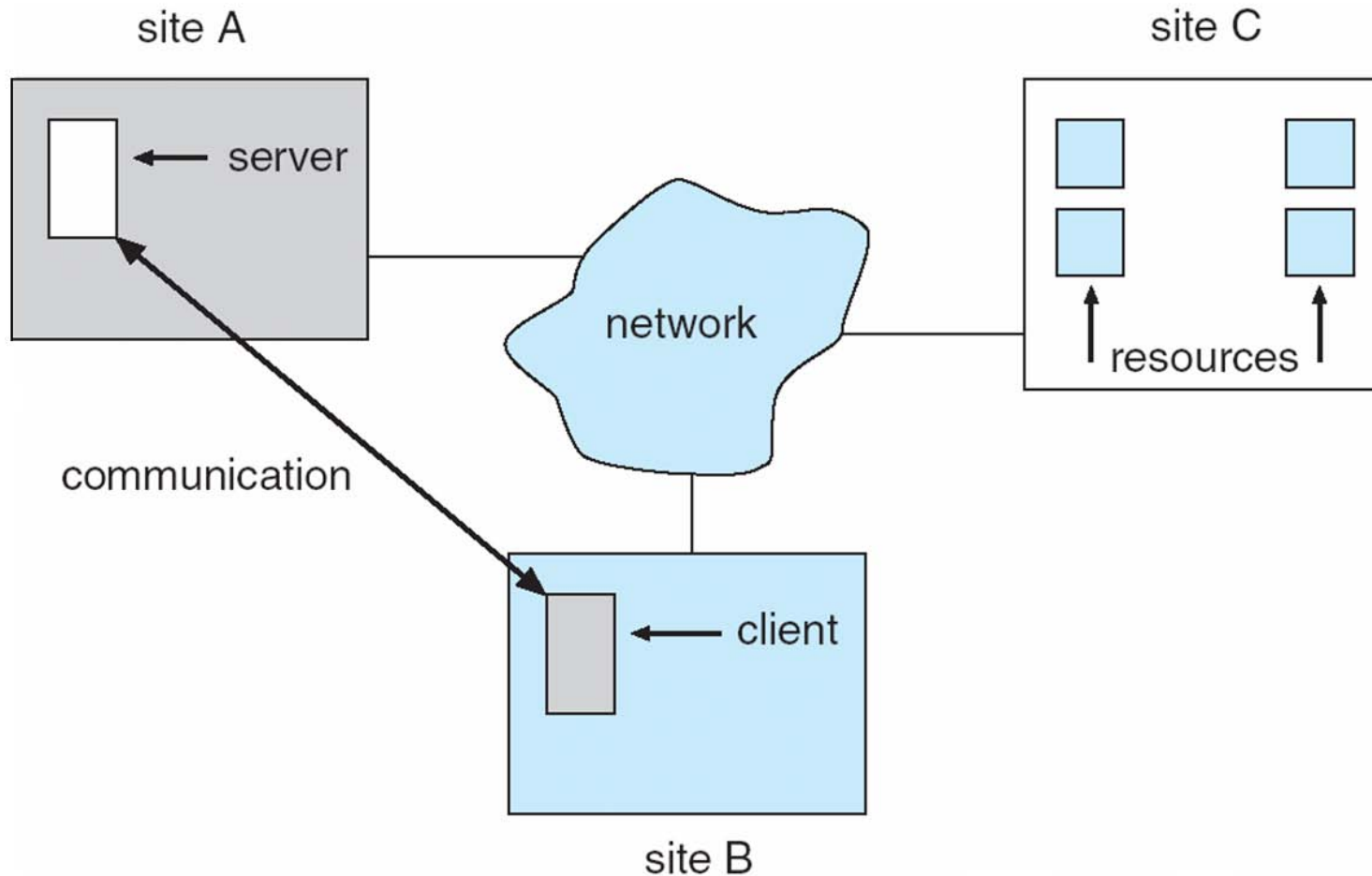
Types of Distributed Operating Systems

- 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)

  - Remote Desktop (Microsoft Windows)

  - 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
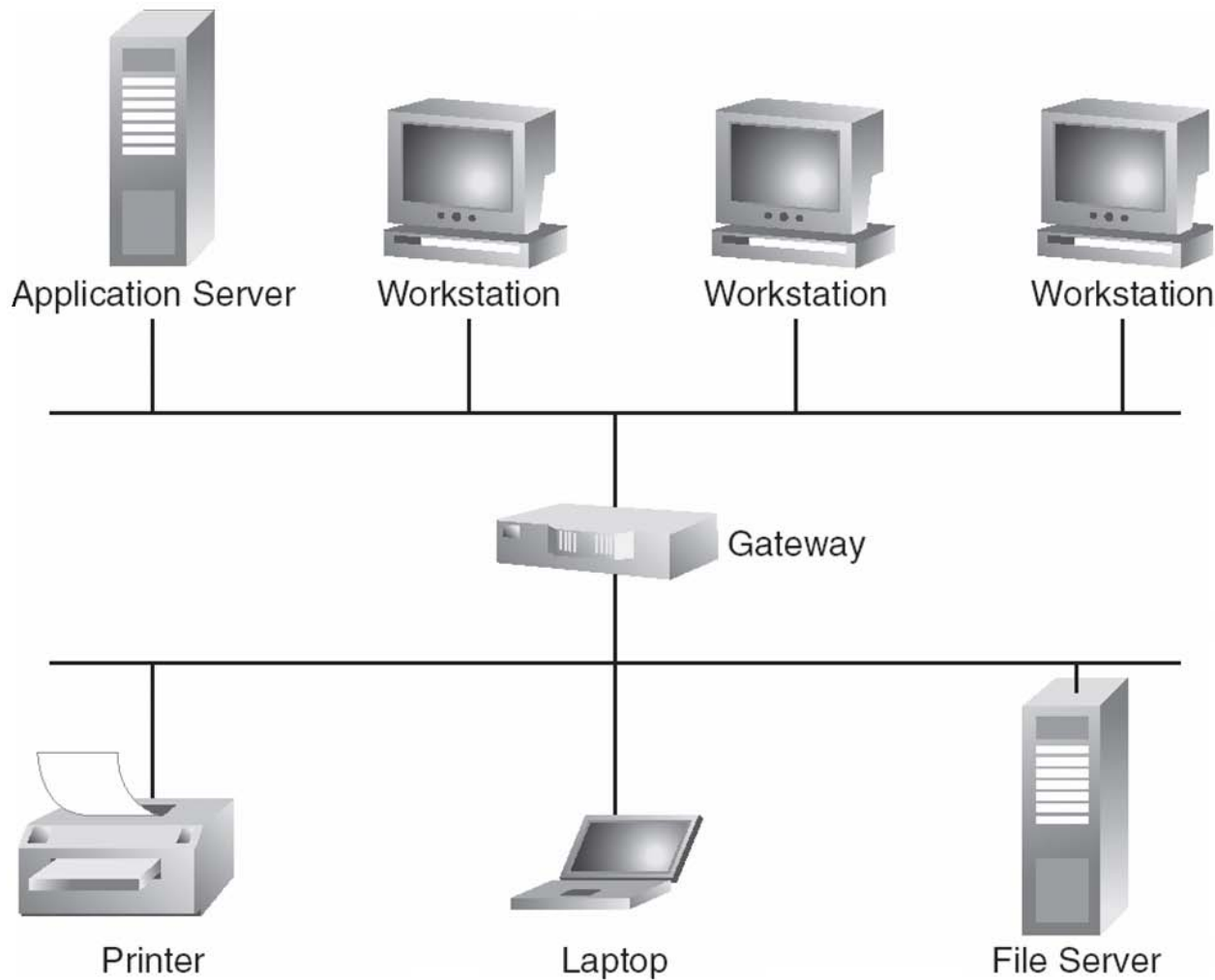
Network Structure

- **Local-Area Network** (**LAN**) – designed to cover small geographical area.

  - Multiaccess bus, ring, or star network

  - Speed ≈ 10 – 100 megabits/second

  - Broadcast is fast and cheap

  - Nodes:

    - usually workstations and/or personal computers

    - a few (usually one or two) mainframes

Depiction of typical LAN
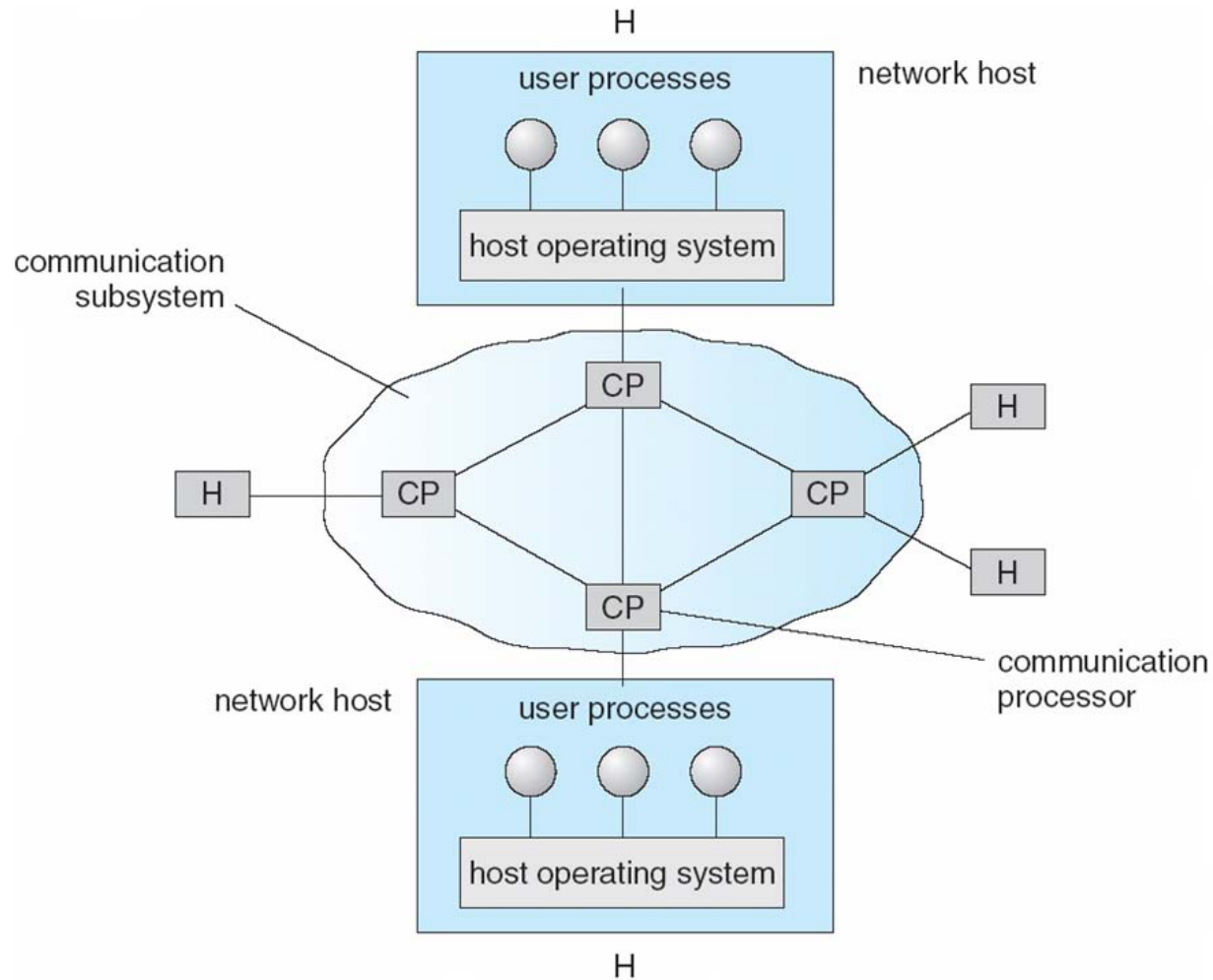
Network Types (Cont.)

- **Wide-Area Network** (**WAN**) – links geographically separated sites

  - Point-to-point connections over long-haul lines (often leased from a phone company)

  - Speed $\approx$ 1.544 – 45 megbits/second

  - Broadcast usually requires multiple messages

  - Nodes:

    ‣ usually a high percentage of mainframes

Communication Processors in a Wide-Area Network

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 overhead for shipping each message, and may waste network bandwidth
    - Message and packet switching require less setup time, but incur more overhead per message

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 system, the most common network system

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 which is sent in a separate slot
  - This scheme has been adopted in the experimental Cambridge Digital Communication Ring

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 packets, decoding the address of incoming packets, and maintaining routing information for proper response to changing load levels
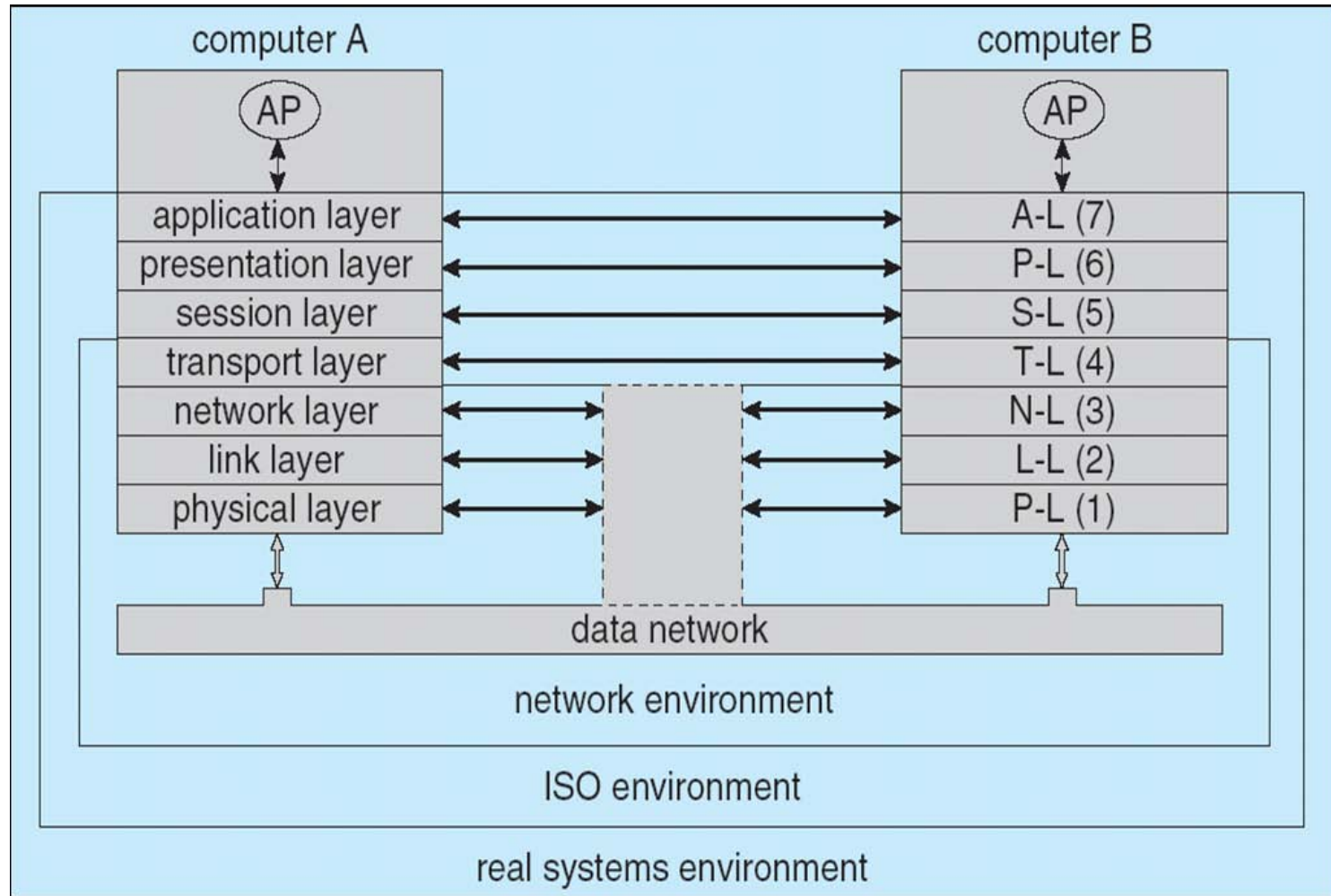
Communication Protocol (Cont.)

- **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 users' deals with file transfer, remote-login protocols and electronic mail, as well as schemas for distributed databases
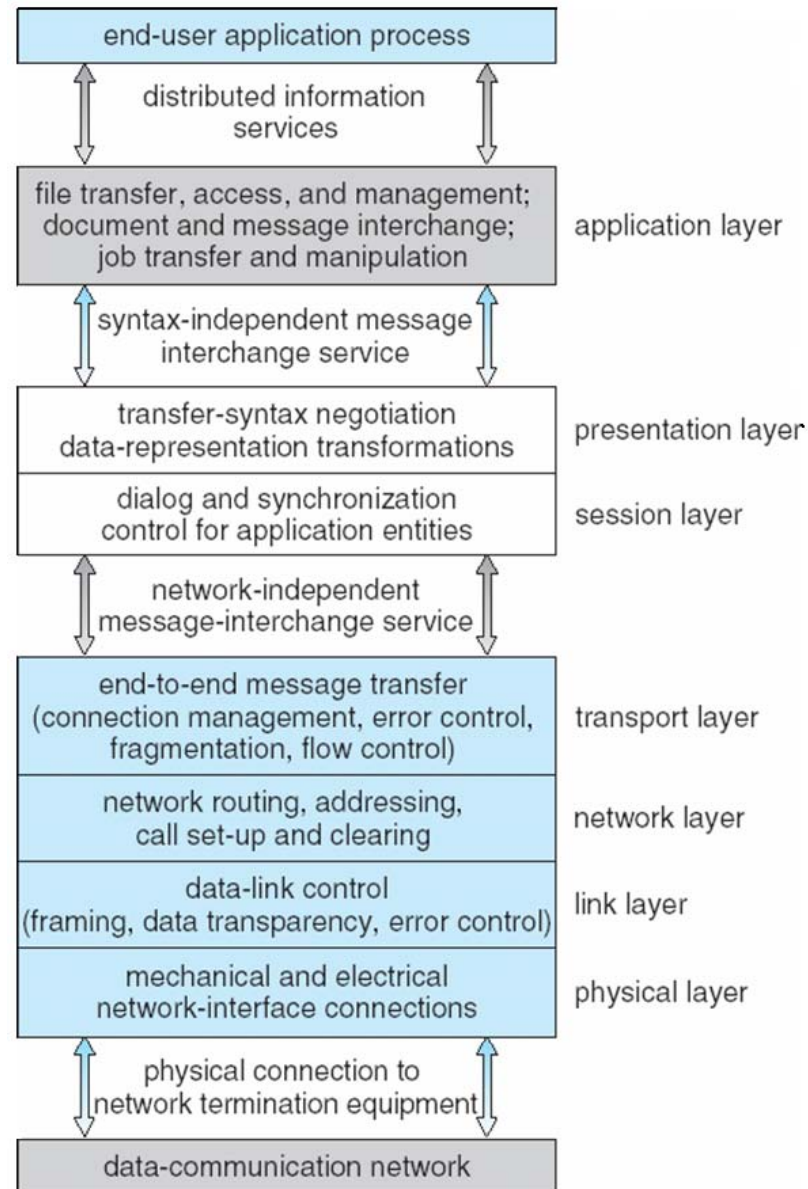
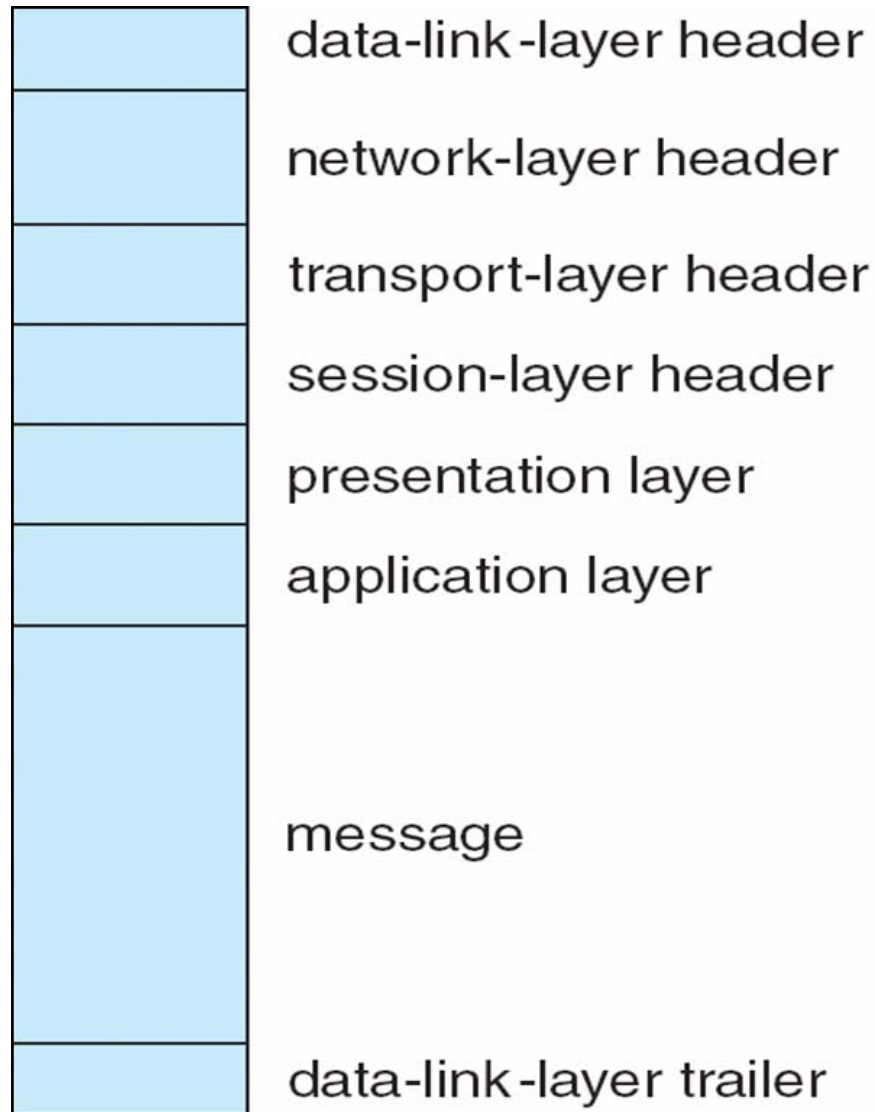# Communication Via ISO Network Model

The ISO Protocol Layer

end-user application process	
↕ distributed information services ↕	
file transfer, access, and management; document and message interchange; job transfer and manipulation	application layer
↕ syntax-independent message interchange service ↕	
transfer-syntax negotiation data-representation transformations	presentation layer
dialog and synchronization control for application entities	session layer
↕ network-independent message-interchange service ↕	
end-to-end message transfer (connection management, error control, fragmentation, flow control)	transport layer
network routing, addressing, call set-up and clearing	network layer
data-link control (framing, data transparency, error control)	link layer
mechanical and electrical network-interface connections	physical layer
↕ physical connection to network termination equipment ↕	
data-communication network	

The ISO Network Message

data-link-layer header

network-layer header

transport-layer header

session-layer header
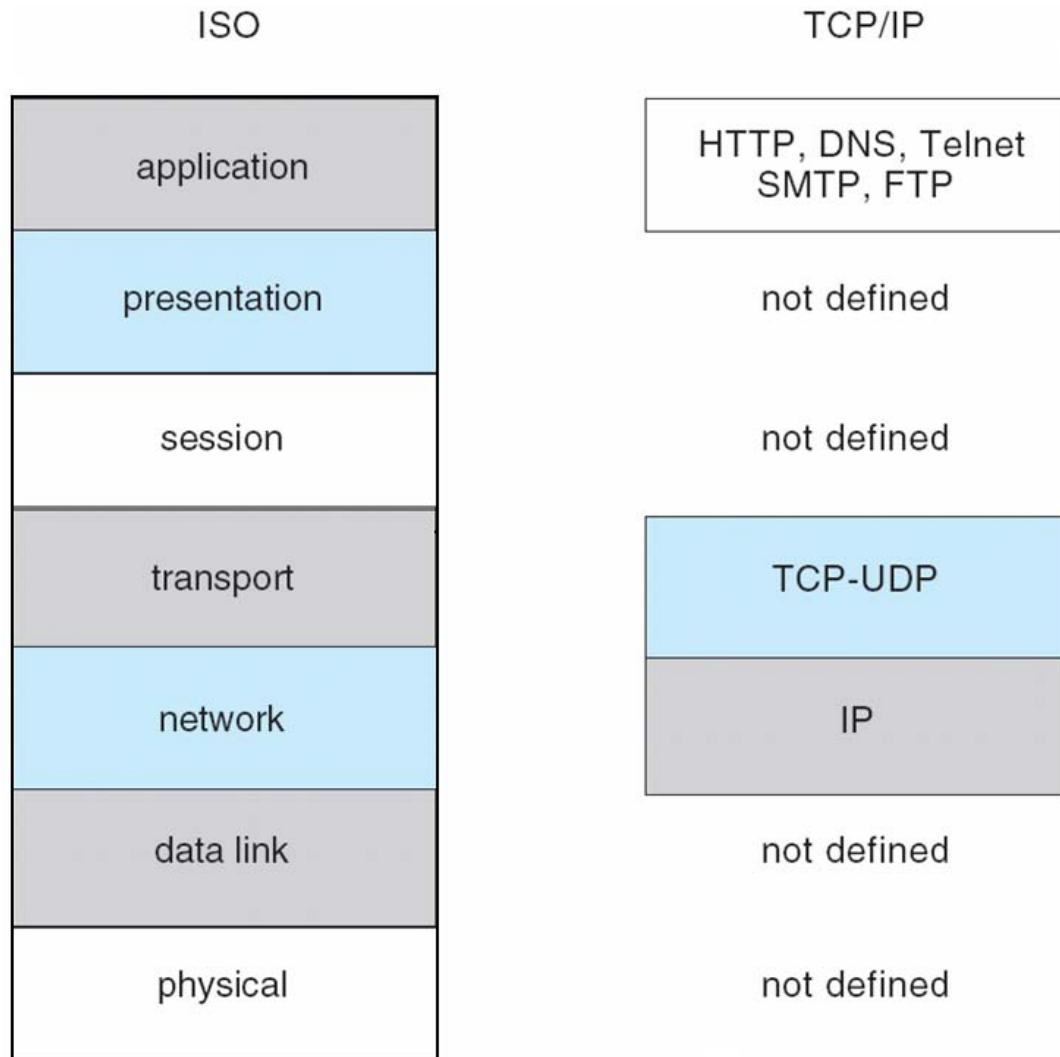
presentation layer

application layer

message

data-link-layer trailer

The TCP/IP Protocol Layers

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
  - If the hosts are on different networks, the sending host will send the packet to a *router* which routes the packet to the destination network

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

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

Distributed Fole system

- Naming and Transparency
- Remote File Access

Background

- **Distributed file system** (**DFS**) – a distributed implementation of the classical time-sharing model of a file system, where multiple users share files and storage resources

- A DFS manages set of dispersed storage devices

- Overall storage space managed by a DFS is composed of different, remotely located, smaller storage spaces

- There is usually a correspondence between constituent storage spaces and sets of files

DFS Structure

- **Service** – software entity running on one or more machines and providing a particular type of function to a priori unknown clients

- **Server** – service software running on a single machine

- **Client** – process that can invoke a service using a set of operations that forms its client interface

- A client interface for a file service is formed by a set of primitive file operations (create, delete, read, write)

- Client interface of a DFS should be transparent, i.e., not distinguish between local and remote files

Naming and Transparency

- **Naming** – mapping between logical and physical objects

- **Multilevel mapping** – abstraction of a file that hides the details of how and where on the disk the file is actually stored

- A **transparent** DFS hides the location where in the network the file is stored

- For a file being replicated in several sites, the mapping returns a set of the locations of this file's replicas; both the existence of multiple copies and their location are hidden

Naming Structures

- **Location transparency** – file name does not reveal the file's physical storage location

- **Location independence** – file name does not need to be changed when the file's physical storage location changes

Naming Schemes — Three Main Approaches

- Files named by combination of their host name and local name; guarantees a unique system-wide name

- Attach remote directories to local directories, giving the appearance of a coherent directory tree; only previously mounted remote directories can be accessed transparently

- Total integration of the component file systems
  - A single global name structure spans all the files in the system
  - If a server is unavailable, some arbitrary set of directories on different machines also becomes unavailable

Remote File Access

- **Remote-service mechanism** is one transfer approach

- Reduce network traffic by retaining recently accessed disk blocks in a cache, so that repeated accesses to the same information can be handled locally

  - If needed data not already cached, a copy of data is brought from the server to the user

  - Accesses are performed on the cached copy

  - Files identified with one master copy residing at the server machine, but copies of (parts of) the file are scattered in different caches

  - **Cache-consistency problem** – keeping the cached copies consistent with the master file

    ‣ Could be called **network virtual memory**

Cache Location – Disk vs. Main Memory

- Advantages of disk caches

  - More reliable

  - Cached data kept on disk are still there during recovery and don't need to be fetched again

- Advantages of main-memory caches:

  - Permit workstations to be diskless

  - Data can be accessed more quickly

  - Performance speedup in bigger memories

  - Server caches (used to speed up disk I/O) are in main memory regardless of where user caches are located; using main-memory caches on the user machine permits a single caching mechanism for servers and users

Cache Update Policy

- **Write-through** – write data through to disk as soon as they are placed on any cache
  - Reliable, but poor performance

- **Delayed-write** – modifications written to the cache and then written through to the server later
  - Write accesses complete quickly; some data may be overwritten before they are written back, and so need never be written at all
  - Poor reliability; unwritten data will be lost whenever a user machine crashes
  - Variation – scan cache at regular intervals and flush blocks that have been modified since the last scan
  - Variation – **write-on-close**, writes data back to the server when the file is closed
    - ▸ Best for files that are open for long periods and frequently modified

CacheFS and its Use of Caching
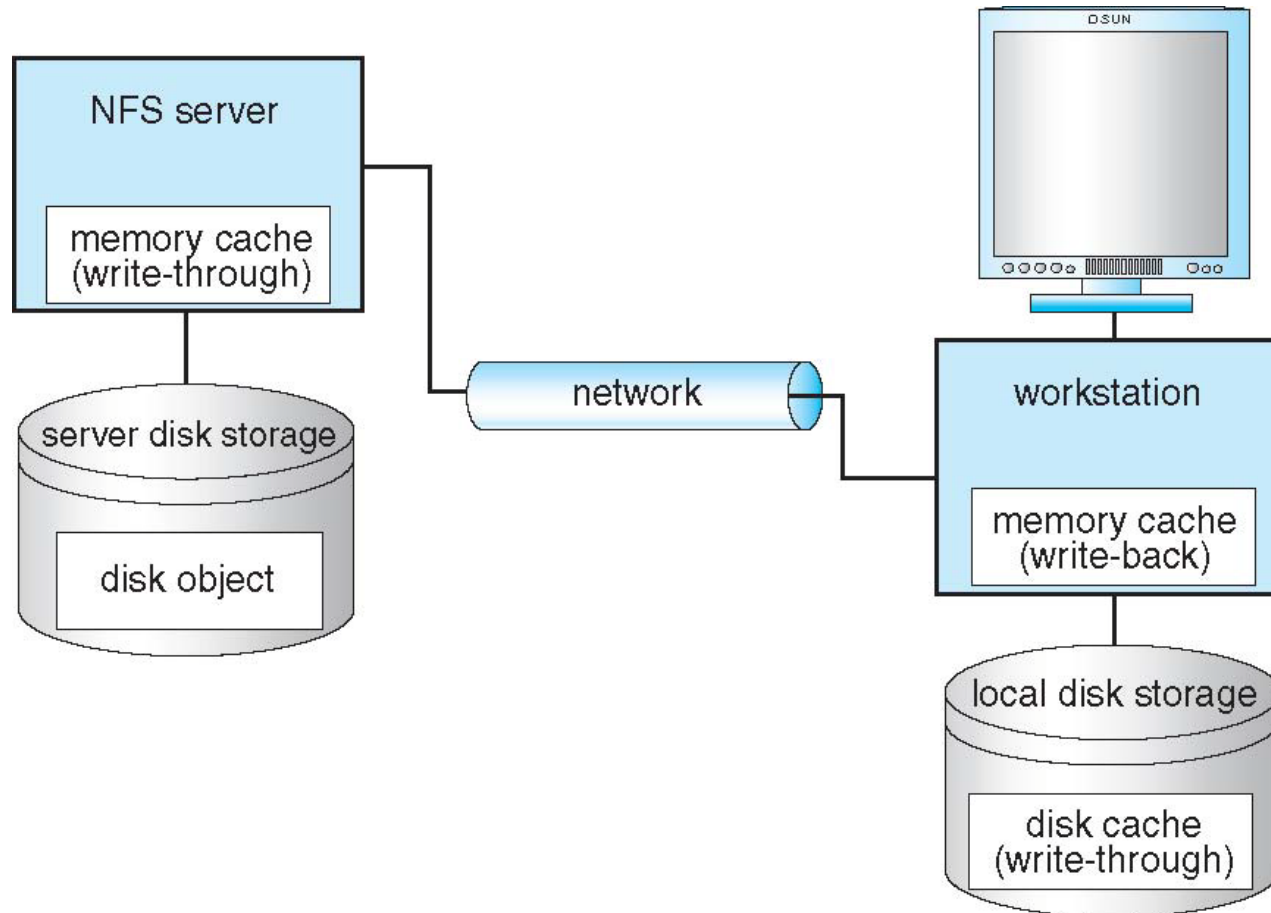
Consistency

- Is locally cached copy of the data consistent with the master copy?

- **Client-initiated approach**
  - Client initiates a validity check
  - Server checks whether the local data are consistent with the master copy

- **Server-initiated approach**
  - Server records, for each client, the (parts of) files it caches
  - When server detects a potential inconsistency, it must react

Comparing Caching and Remote Service

- In caching, many remote accesses handled efficiently by the local cache; most remote accesses will be served as fast as local ones

- Servers are contracted only occasionally in caching (rather than for each access)
    - Reduces server load and network traffic
    - Enhances potential for scalability

- Remote server method handles every remote access across the network; penalty in network traffic, server load, and performance

- Total network overhead in transmitting big chunks of data (caching) is lower than a series of responses to specific requests (remote-service)

Caching and Remote Service (Cont.)

- Caching is superior in access patterns with infrequent writes
    - With frequent writes, substantial overhead incurred to overcome cache-consistency problem

- Benefit from caching when execution carried out on machines with either local disks or large main memories

- Remote access on diskless, small-memory-capacity machines should be done through remote-service method

- In caching, the lower intermachine interface is different form the upper user interface

- In remote-service, the intermachine interface mirrors the local user-file-system interface

End of Chapter 17