Distributed System Structures

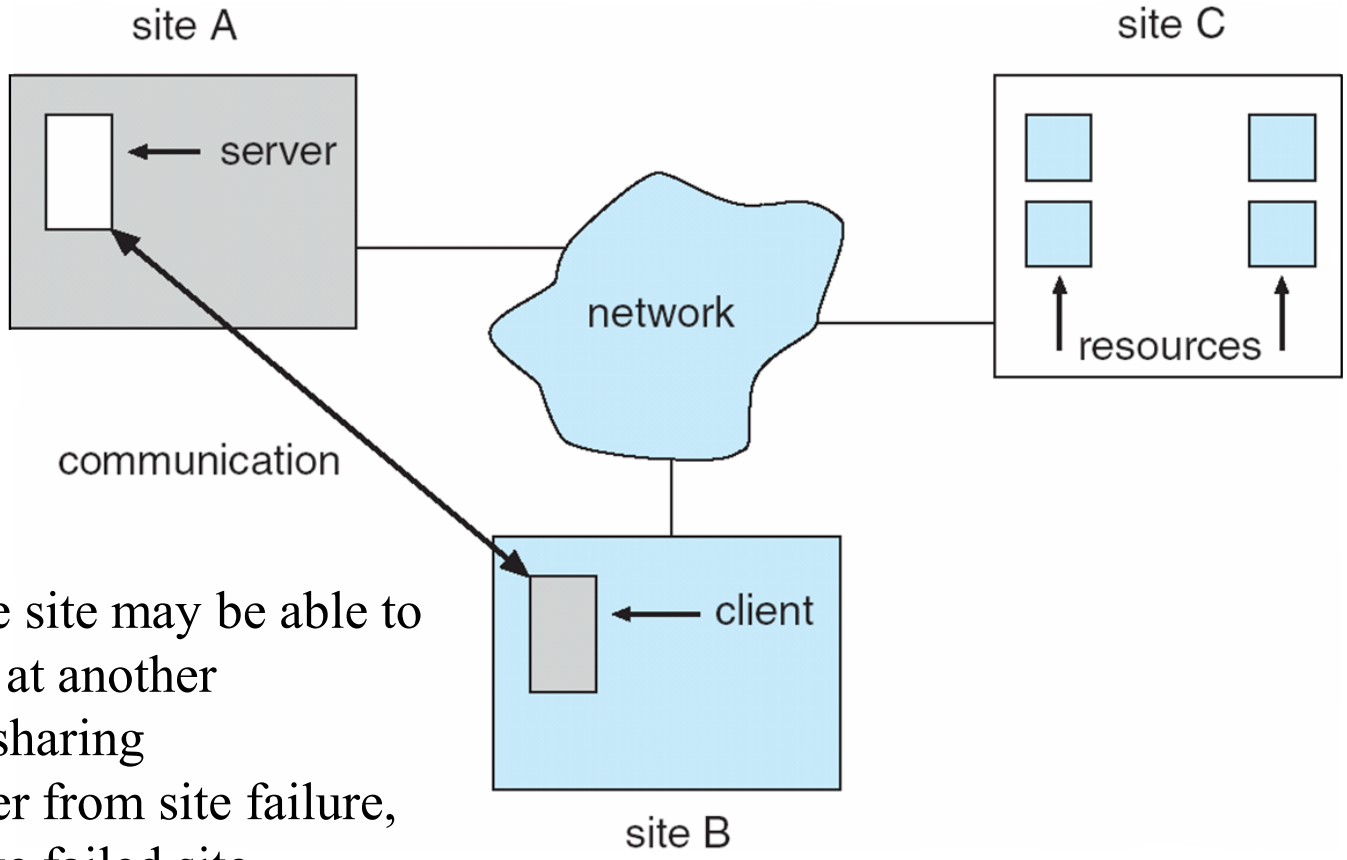
Reading: Chapter16 of Textbook

What is the general structure of distributed operating systems?
What are the design principles of distributed system?

Distributed System

A distributed system is a collection of loosely coupled processors interconnected by a communication network.

Generally, one host at one site, the *server,* has a resource that another host at another site, the *client* (or user), would like to use



A general structure of a distributed system

Reasons for distributed systems

Resource sharing - user at one site may be able to use the resources available at another

Computation speedup – load sharing

Reliability – detect and recover from site failure, function transfer, reintegrate failed site

Communication – message passing

Types of Distributed Operating Systems

Two general categories of network-oriented operating systems: network operating systems and 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) or
- Transferring data from remote machines to local machines, via the File Transfer Protocol (FTP) mechanism
- Simple to implement but difficult to user to access and utilize.

Types of Distributed Operating Systems (Contd…)

Distributed-Operating Systems

Users not aware of multiplicity of machines

Access to remote resources similar to access to local resources

Data and process migration from one site to another is under control of OS

Data Migration – transfer data by transferring entire file (Andrew file system), or transferring only those portions of the file necessary for the immediate task (NFS protocol or Microsoft SMB protocol)

Computation Migration – transfer the computation, rather than the data, across the system

Process Migration – Logical extension of computation migration, execute an entire process, or parts of it, at different sites. Used for:

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

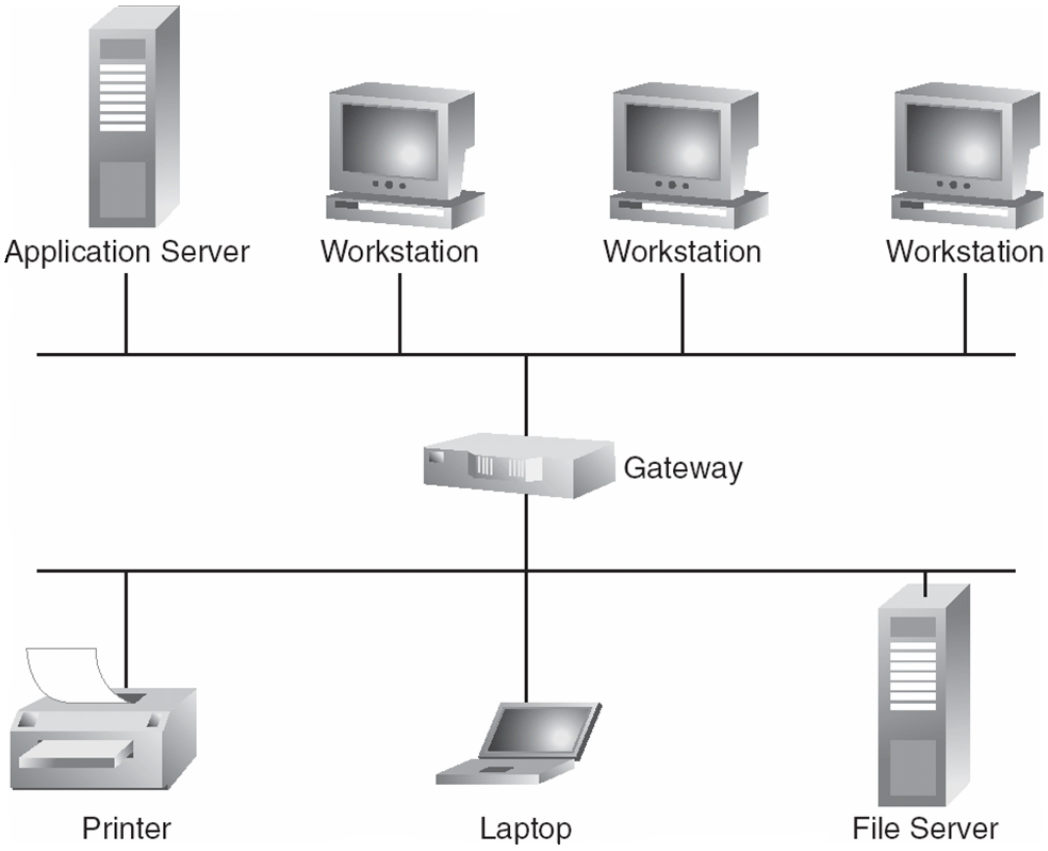
Network Structure

There are basically two types of networks: *Local-Area-Networks (LAN)* and *Wide-Area-Networks (WAN)*

Local-Area-Networks (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

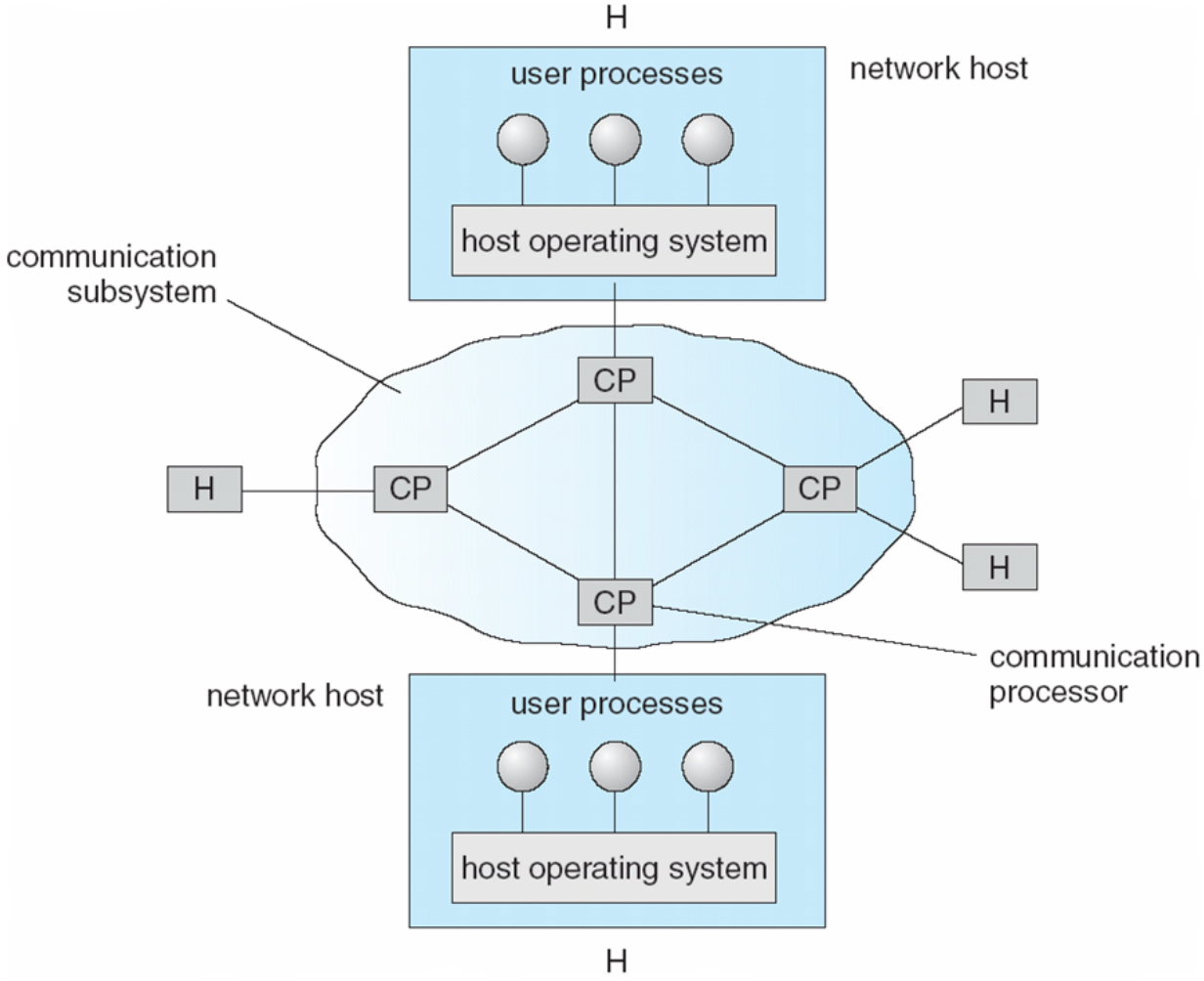


Network Structure

Wide-Area Network (WAN)

Links geographically separated sites

- Point-to-point connections over long-haul lines (often leased from a phone company)
- Speed ≈ 1.544 – 45 megbits/second
- Broadcast usually requires multiple messages
- Nodes:
  - usually a high percentage of mainframes



Network Topology

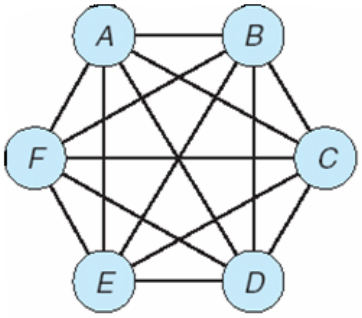
Sites in the system can be physically connected in a variety of ways; they are compared with respect to the following criteria:

- **Installation 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*?
- **Availability** - If a link or a site in the system fails, can the remaining sites still communicate with each other?

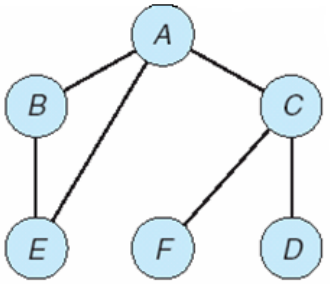
The various topologies are depicted as graphs whose nodes correspond to sites

- An edge from node *A* to node *B* corresponds to a direct connection between the two sites

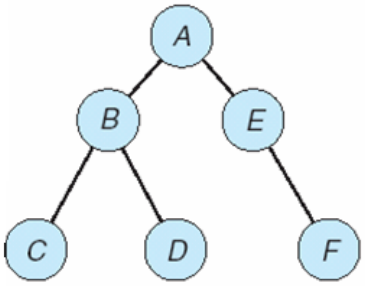
The six items, shown in figure, depict various network topologies



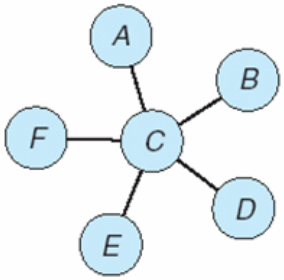
fully connected network



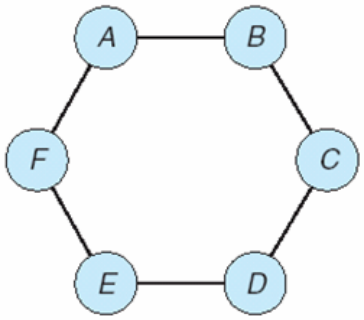
partially connected network



tree-structured network



star network



ring 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

The naming of the systems in the network

Address messages with the process-id

Identify processes on remote systems by <host-name, identifier> pair, where host-name is a name unique within network and identifier is pid

Domain name service (**DNS**) – specifies the naming structure of the hosts, as well as name to address resolution (Internet). Names progress from most specific to most general e.g cdcsit.tu.edu.np. The system resolves the addresses by examining the host-name component in reverse order.

Routing Strategies

When a process at site A wants to communicate with a process at site B, how is the message sent, if multiple paths are exist?

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
- Can not adopt in link failure or load changes

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

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

Pairs of processes that want to communicate over the network can be connected in a number of ways

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. Start transmitting if link is free. If two or more sites begin transmitting at exactly the same time, then they will register a CD and will stop transmitting and try after some random time. 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

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

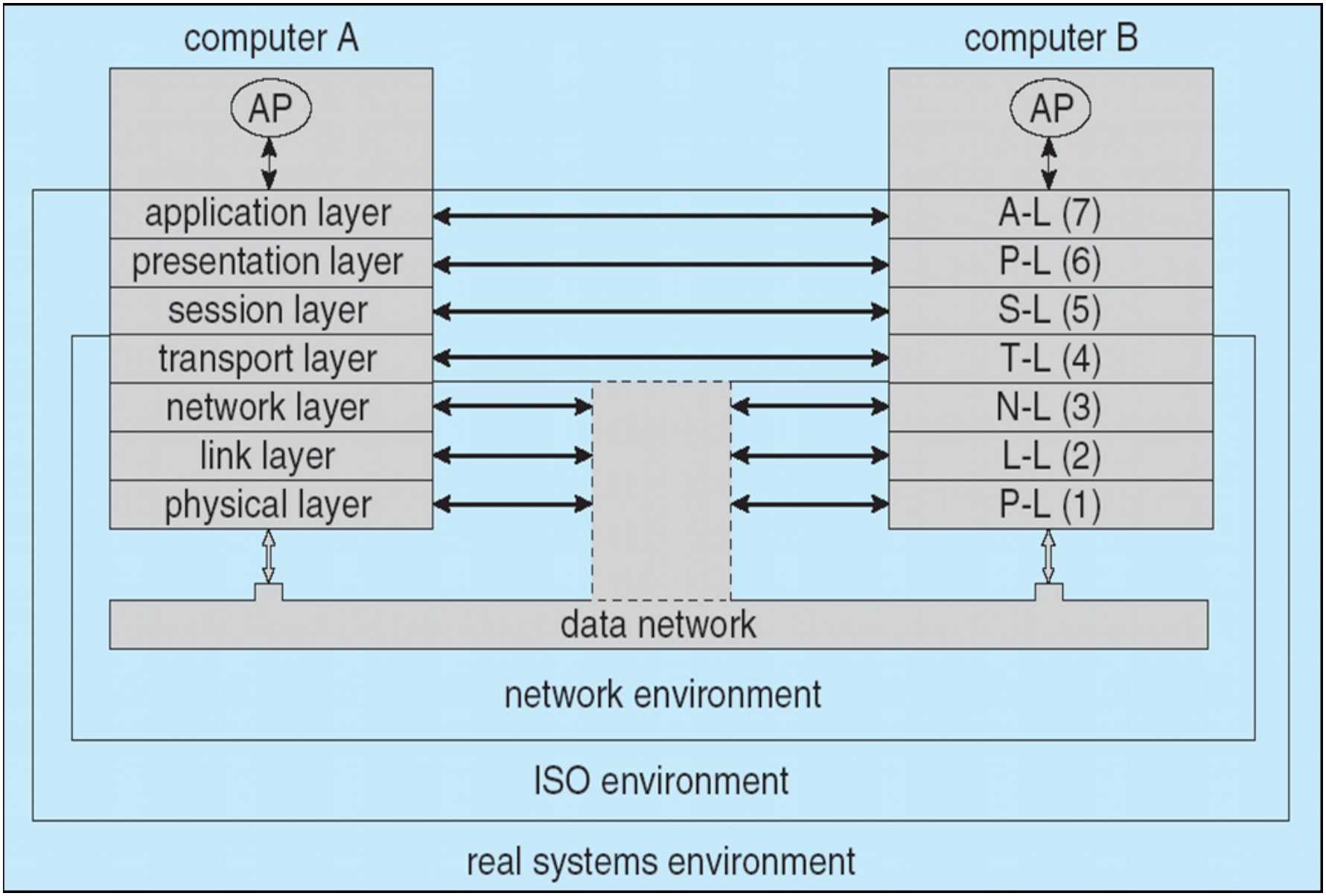
This scheme has been adopted in the experimental Cambridge Digital Communication Ring

Communication Protocol

Simplifying design by partitioning problem into layer structure, the ISO layers are:

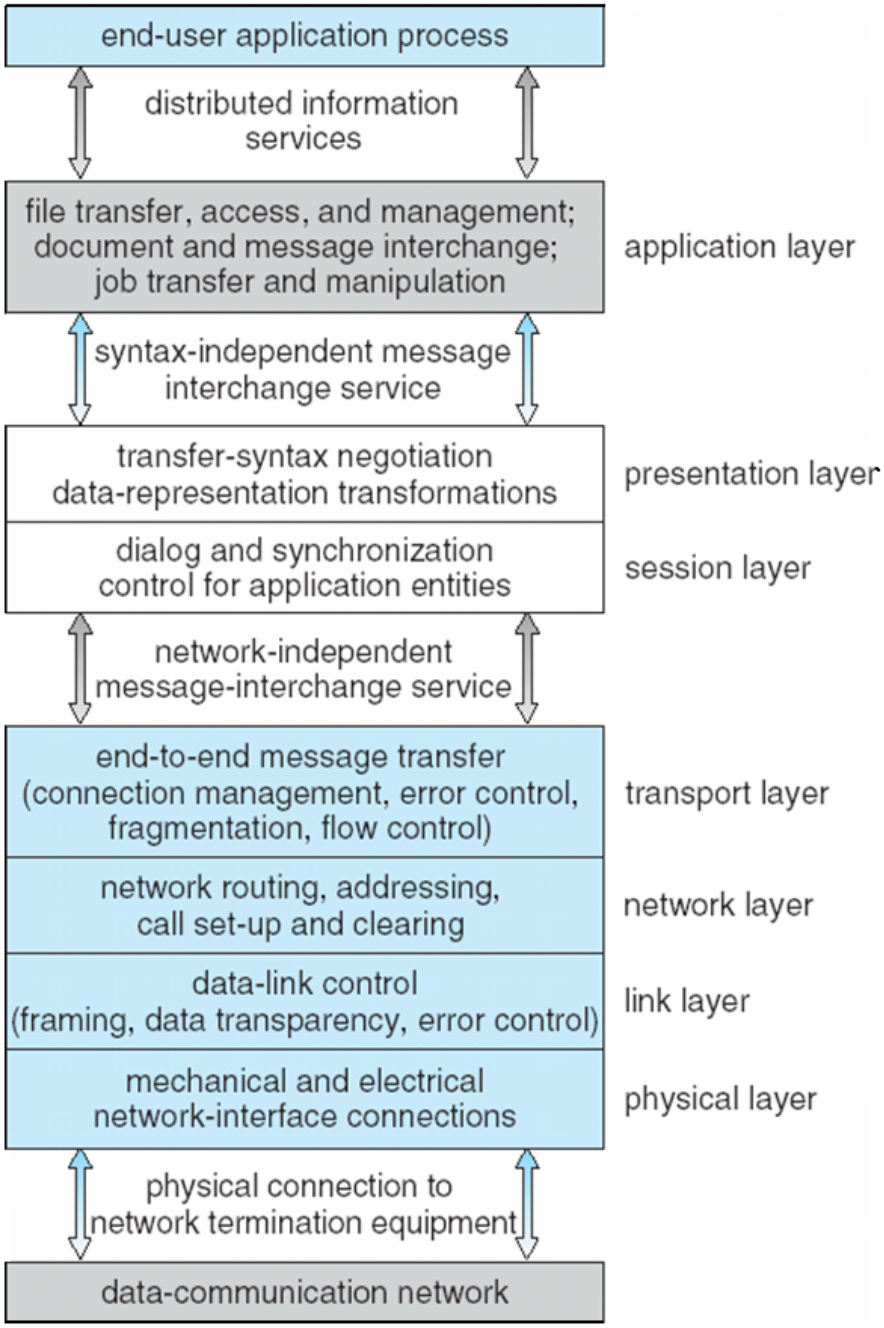
Physical layer – handles the mechanical and electrical details of the physical transmission of a bit stream. Hardware of the networking.
Data-link layer – handles the *frames*, or fixed-length parts of packets, including any error detection and recovery that occurred in the physical layer.
Network layer – 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
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



Logical communication between two computers

The ISO Protocol Layer



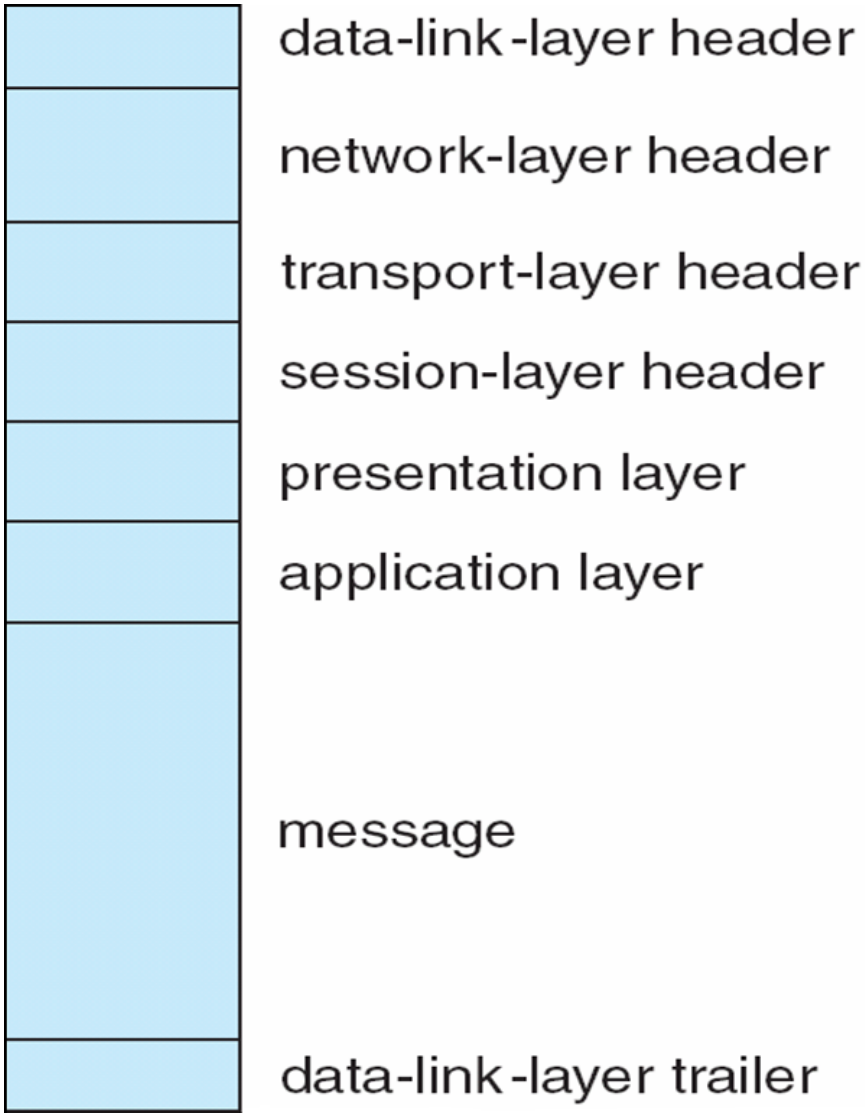
logically each layer of a protocol stack communicates with the equivalent layer on other systems

Each layer may modify the message and include message-header data for the equivalent layer on the receiving side

The message reaches the data-network layer and is transferred as one or more packets.

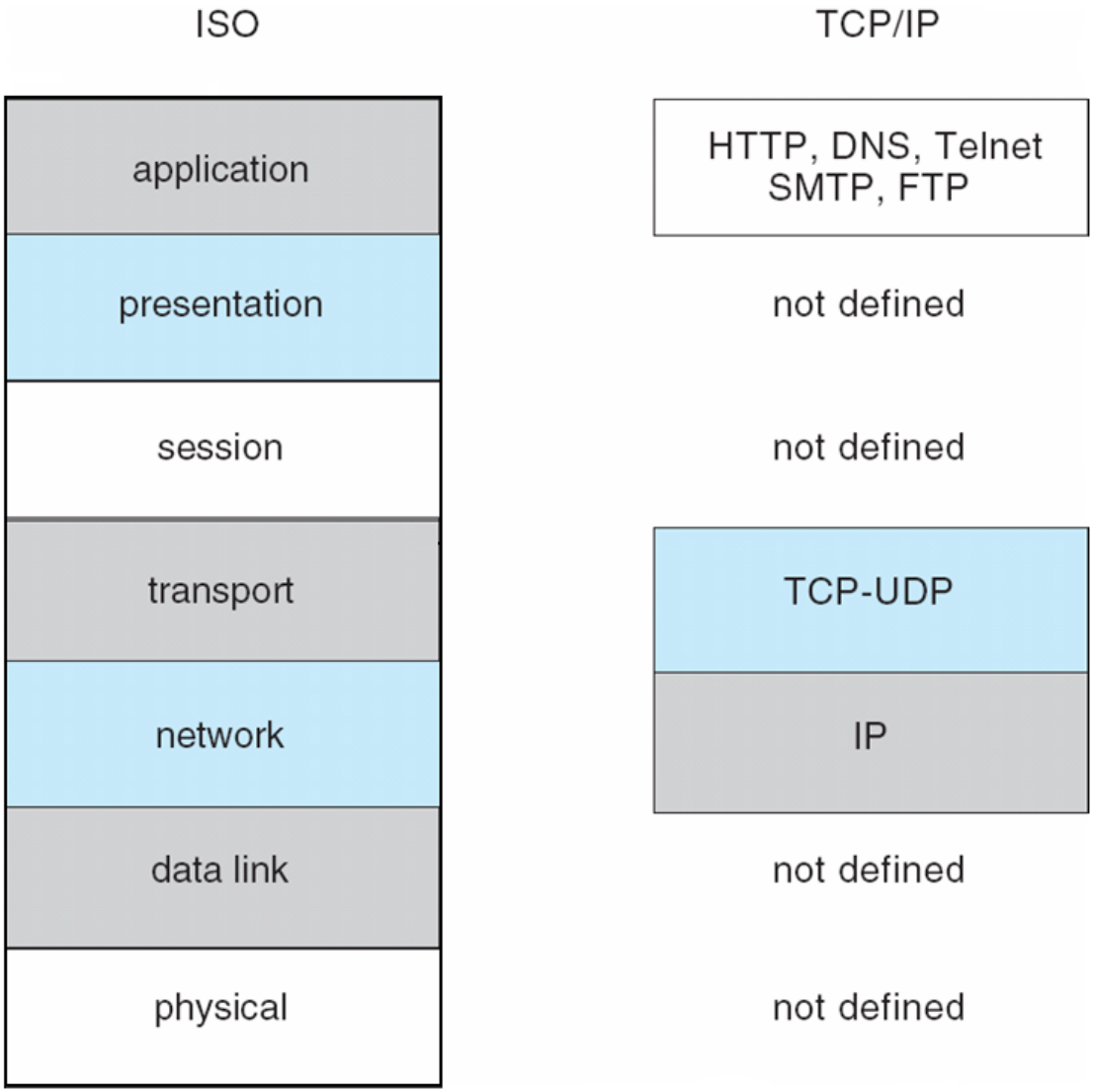
The data-link layer of the target system receives these data, and the message is moved up through the protocol stack; it is analyzed, modified, and stripped of headers as it progresses. It finally reaches the application layer for use by the receiving process

The ISO Network Message



The ISO vs TCP/IP Protocol Layers

Widely adopted to-day is the TCP/IP protocol, used in internet



Robustness

To ensure that the system is robust, we must detect any of failures (link, hardware etc), reconfigure the system so that computation can continue, and recover when a site or a link is repaired

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

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

Robustness

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, so that the various routing tables can be updated accordingly

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, so that they will no longer attempt to use the services of the failed site

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. User should be able to access remote resources as local resources and allow to any machine in the system.

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

Home Works

HW #13:
Question 16.3, 16.4, 16.6, 16.9, 16.10, 16.12, 16.13, 16.16 & 16.20 from book

Distributed File System

Reading: Chapter 17 of Textbook

How multiple user share file and storage resources when file are physically dispersed among the distributed system?

How distributed file design and implemented?

Distributed file system (DFS)

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, user manipulate logical data objects where as system manipulate physical block of data

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 and Transparency

Naming Structures – two notions

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. Dynamic mapping, can map same file name in different location

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, request for access are delivered to the server, server perform the accesses and results are forwarded back to the user.

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

Where should the cashed data be stored – on disk or in 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 (Write-Back) – 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

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 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

Stateful vs Stateless File Service

Stateful File Service

Mechanism:

Client opens a file
Server fetches information about the file from its disk, stores it in its memory, and gives the client a connection identifier unique to the client and the open file
Identifier is used for subsequent accesses until the session ends
Server must reclaim the main-memory space used by clients who are no longer active

Increased performance

Fewer disk accesses
Stateful server knows if a file was opened for sequential access and can thus read ahead the next blocks

Stateless File Server

Avoids state information by making each request self-contained
Each request identifies the file and position in the file
No need to establish and terminate a connection by open and close operations

Distinctions Between Stateful and Stateless Service

- Failure Recovery
  - A stateful server loses all its volatile state in a crash
    - Restore state by recovery protocol based on a dialog with clients, or abort operations that were underway when the crash occurred
    - Server needs to be aware of client failures in order to reclaim space allocated to record the state of crashed client processes (orphan detection and elimination)
  - With stateless server, the effects of server failure sand recovery are almost unnoticeable
    - A newly reincarnated server can respond to a self-contained request without any difficulty
- Penalties for using the robust stateless service:
  - longer request messages
  - slower request processing
  - additional constraints imposed on DFS design
- Some environments require stateful service
  - A server employing server-initiated cache validation cannot provide stateless service, since it maintains a record of which files are cached by which clients
  - UNIX use of file descriptors and implicit offsets is inherently stateful; servers must maintain tables to map the file descriptors to inodes, and store the current offset within a file

File Replication

- Replicas of the same file reside on failure-independent machines
- Improves availability and can shorten service time
- Naming scheme maps a replicated file name to a particular replica
  - Existence of replicas should be invisible to higher levels
  - Replicas must be distinguished from one another by different lower-level names
- Updates – replicas of a file denote the same logical entity, and thus an update to any replica must be reflected on all other replicas
- Demand replication – reading a nonlocal replica causes it to be cached locally, thereby generating a new nonprimary replica

Home Works

HW #14:
Question No 17.2, 17.5, 17.6, 17.8 & 17.9 from book