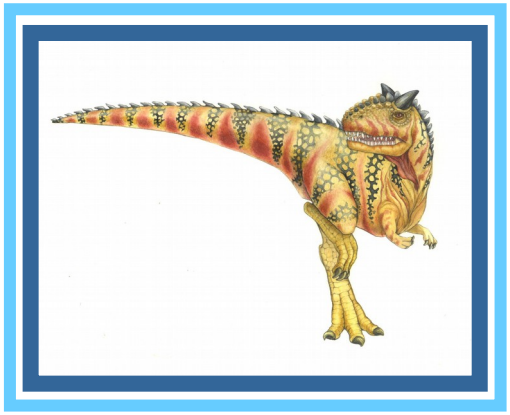
Chapter 19: Distributed Systems



Chapter 19: Distributed Systems

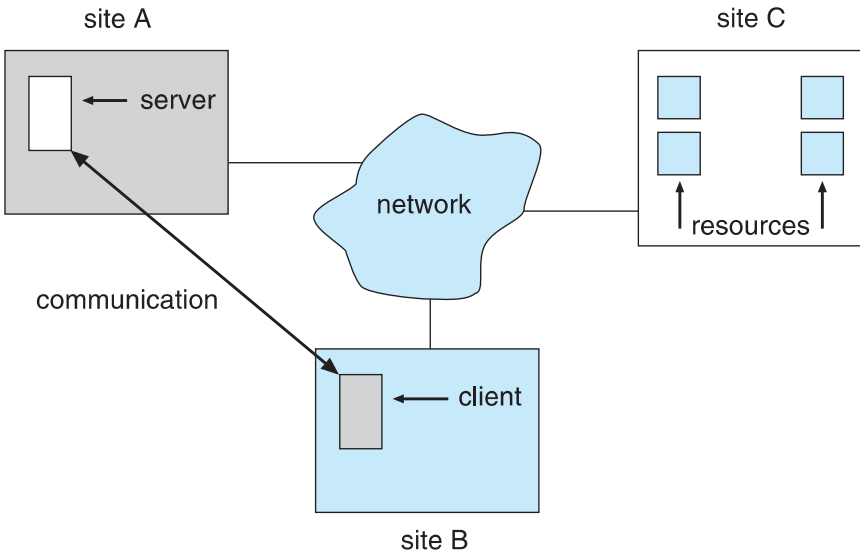
- Advantages of Distributed Systems
- Types of Network-Based Operating Systems
- Network Structure
- Communication Structure
- Communication Protocols
- An Example: TCP/IP
- Robustness
- Design Issues
- Distributed File System

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 general communication structure and communication protocols
- To describe issues concerning the design of distributed systems

Overview

- **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
  - Generally a ***server*** has a resource a ***client*** node at a different site wants to use



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** or **job migration**
- Reliability – detect and recover from site failure, function transfer, reintegrate failed site
- Communication – **message** passing
  - All higher-level functions of a standalone system can be expanded to encompass a distributed system
- Computers can be downsized, more flexibility, better user interfaces and easier maintenance by moving from large system to multiple smaller systems performing distributed computing

Types of Distributed Operating Systems

- Network Operating Systems
  - Users are aware of multiplicity of machines
- Distributed Operating Systems
  - Users not aware of multiplicity of machines

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 (ssh)
  - Remote Desktop (Microsoft Windows)
  - Transferring data from remote machines to local machines, via the File Transfer Protocol (FTP) mechanism
- Users must change paradigms – establish a **session**, give network-based commands
  - More difficult for users

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
  - Via remote procedure calls (RPCs)
  - Via messaging system

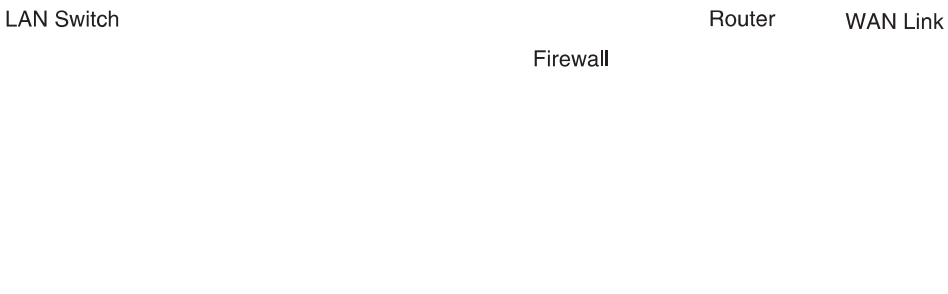
Distributed-Operating Systems (Cont.)

- **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
- Consider the World Wide Web

Network Structure

- **Local-Area Network** (**LAN**) – designed to cover small geographical area
  - Multiple topologies like star or ring
  - Speeds from 1Mb per second (Appletalk, bluetooth) to 40 Gbps for fastest Ethernet over twisted pair copper or optical fibre
  - Consists of multiple computers (mainframes through mobile devices), peripherals (printers, storage arrays), routers (specialized network communication processors) providing access to other networks
  - Ethernet most common way to construct LANs
    - Multiaccess bus-based
    - Defined by standard IEEE 802.3
  - Wireless spectrum (**WiFi**) increasingly used for networking
    - I.e. IEEE 802.11g standard implemented at 54 Mbps

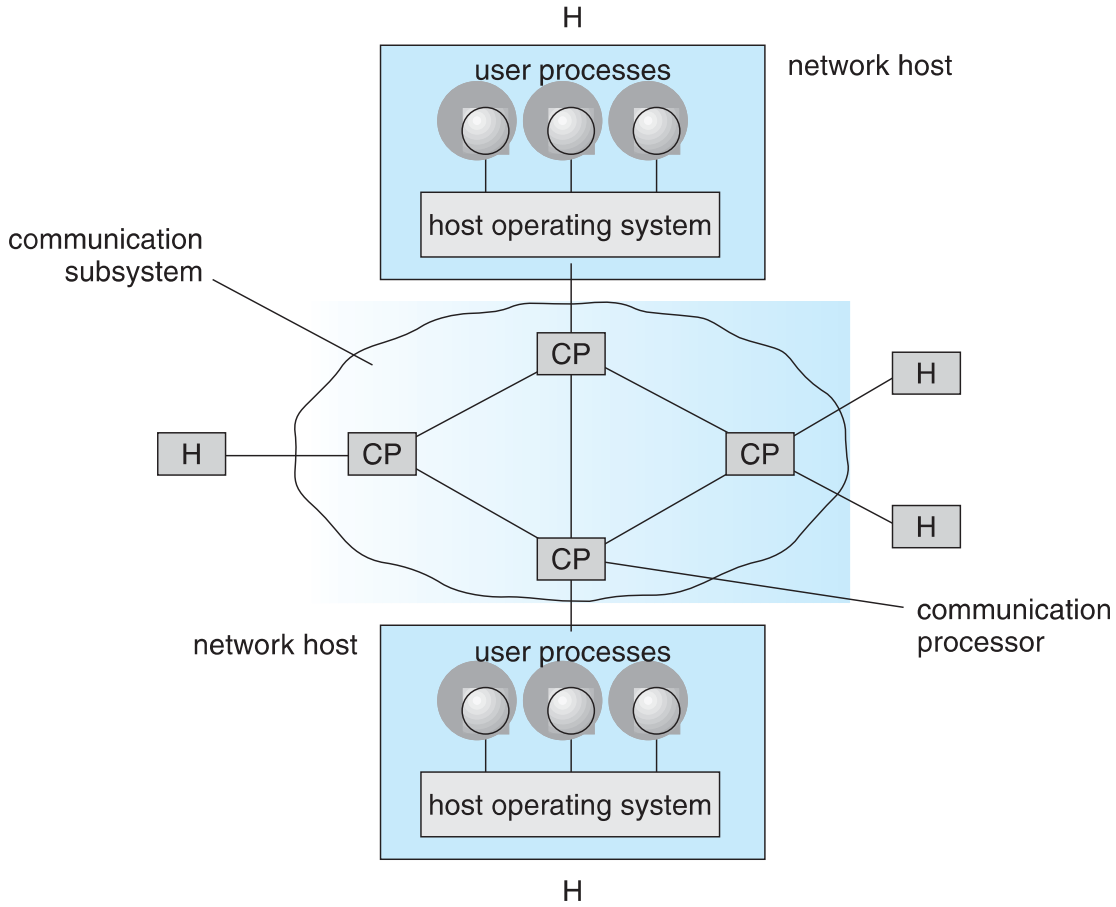
Local-area Network



Network Structure (Cont.)

- **Wide-Area Network** (**WAN**) – links geographically separated sites
  - Point-to-point connections over long-haul lines (often leased from a phone company)
    - Implemented via **connection processors** known as **routers**
  - Internet WAN enables hosts world wide to communicate
    - Hosts differ in all dimensions but WAN allows communications
  - Speeds
    - T1 link is 1.544 Megabits per second
    - T3 is 28 x T1s = 45 Mbps
    - OC-12 is 622 Mbps
  - WANs and LANs interconnect, similar to cell phone network:
    - Cell phones use radio waves to cell towers
    - Towers connect to other towers and hubs

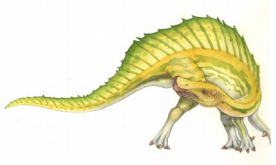
Communication Processors in a WAN



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

- Each computer system in the network has a unique name
- Each process in a given system has a unique name (process-id)
- Identify processes on remote systems by

  <**host-name**, **identifier**> pair
- **Domain name system** (**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 routing**- 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
  - Most complex to set up
- Tradeoffs mean all methods are used
  - UNIX provides ability to mix fixed and dynamic
  - Hosts may have fixed routes and **gateways** connecting networks together may have dynamic routes

Routers

- **Router** -- a communication processor responsible for routing messages
- Must have at least 2 network connections
- Maybe special purpose or just a function running on host
- Checks its tables to determine where destination host is; where to send messages
- Routing methods:
  - Static routing – table only changed manually
  - Dynamic routing – table changed via **routing protocol**

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

Communication Protocol

The communication network is partitioned into the following multiple layers:

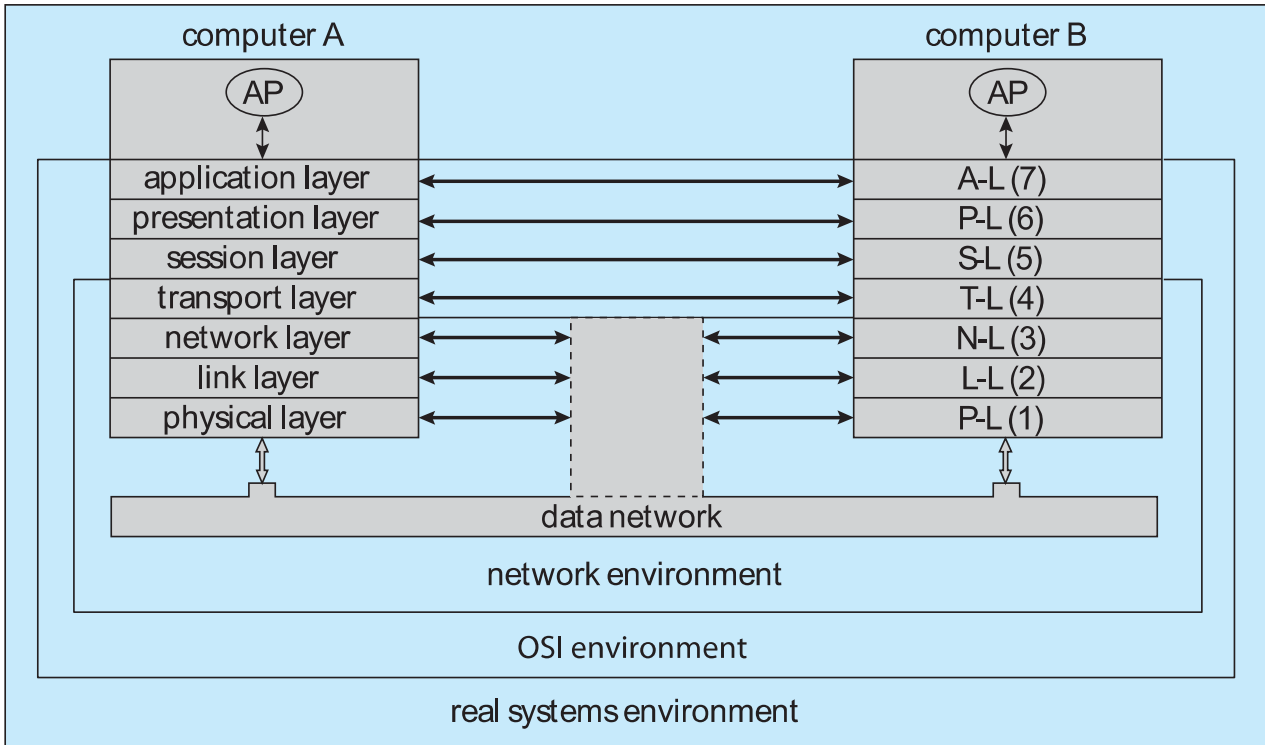
- **Layer 1: Physical layer** – handles the mechanical and electrical details of the physical transmission of a bit stream
- **Layer 2: Data-link layer** – handles the *frames*, or fixed-length parts of packets, including any error detection and recovery that occurred in the physical layer
- **Layer 3: 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.)

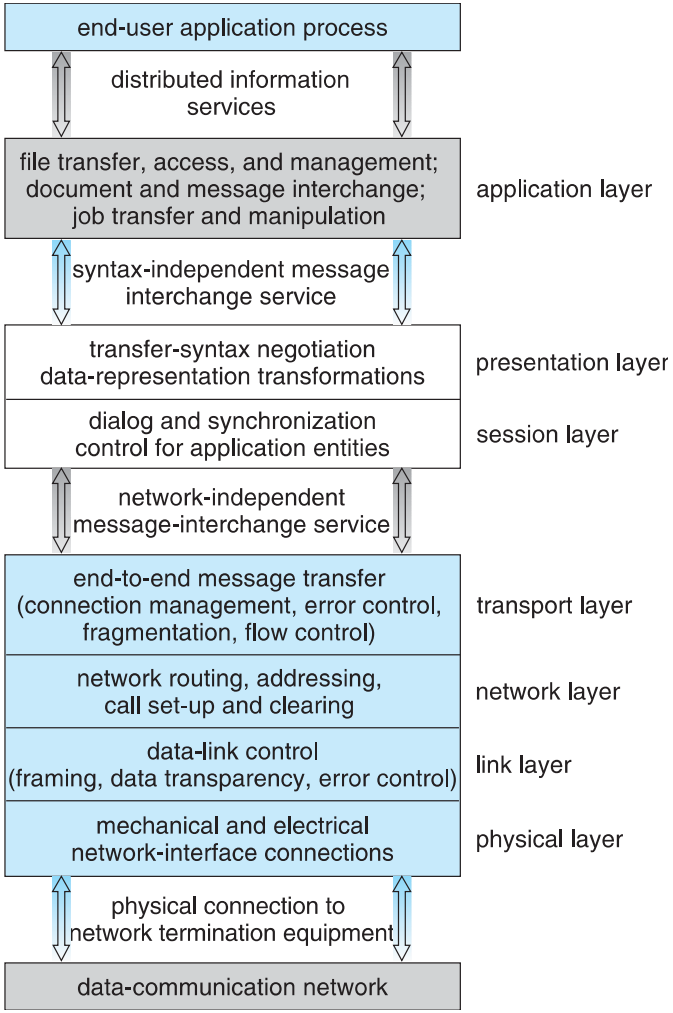
- **Layer 4: Transport layer** – responsible for low-level network access and for message transfer between clients, including partitioning messages into packets, maintaining packet order, controlling flow, and generating physical addresses
- **Layer 5: Session layer** – implements sessions, or process-to-process communications protocols
- **Layer 6: Presentation layer** – resolves the differences in formats among the various sites in the network, including character conversions, and half duplex/full duplex (echoing)
- **Layer 7: 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 OSI Network Model

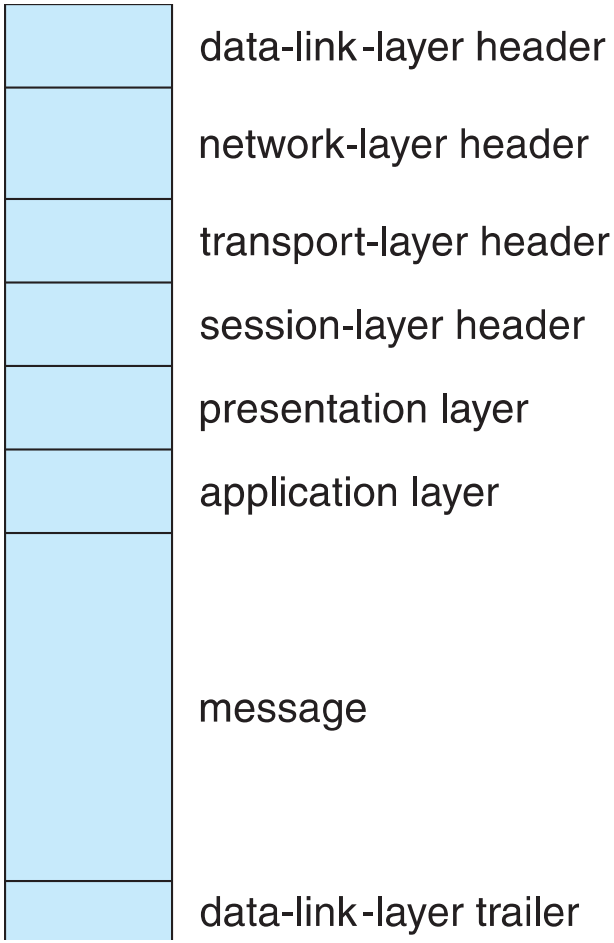
Logical communication between two computers, with the three lowest-level layers implemented in hardware.



The OSI Protocol Stack



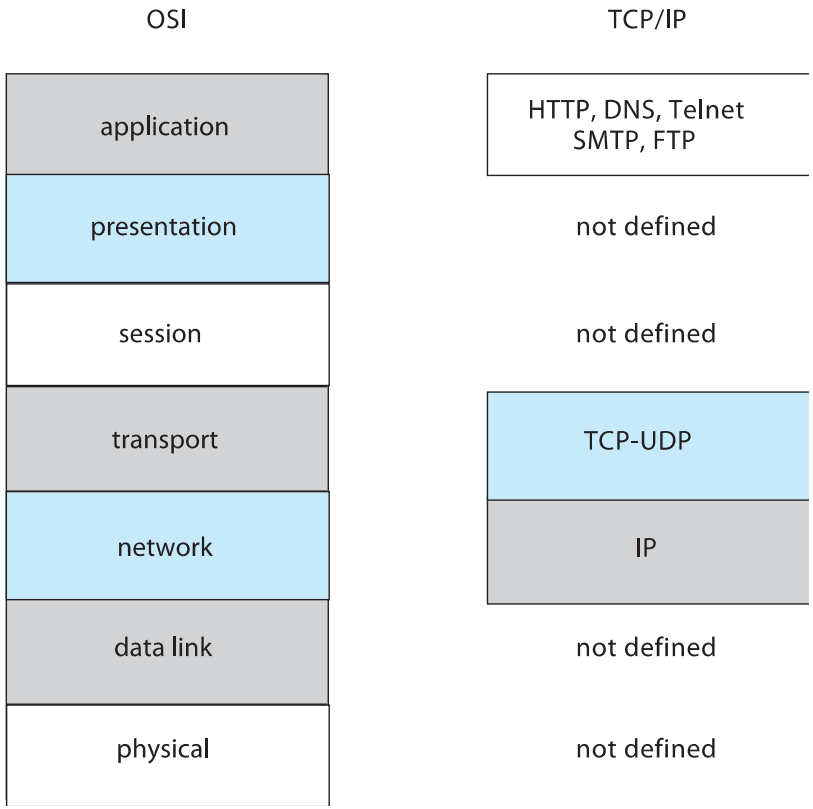
The OSI Network Message



The OSI model

- The OSI model formalizes some of the earlier work done in network protocols but was developed in the late 1970s and is currently not in widespread use.
- The most widely adopted protocol stack is the TCP/IP model, which has been adopted by virtually all Internet sites.
- The TCP/IP protocol stack has fewer layers than the OSI model. Theoretically, because it combines several functions in each layer, it is more difficult to implement but more efficient than OSI networking.
- The relationship between the OSI and TCP/IP models is shown in the next slide.

The OSI and TCP/IP Protocol Stacks



Robustness

- Failure detection
- Reconfiguration

Failure Detection

- Detecting hardware failure is difficult
- To detect a link failure, a **heartbeat** 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:
  - If the link from A to B has failed, this must be broadcast to every site in the system
  - 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
  - Consider **Hadoop** open source programming framework for processing large datasets in distributed environments (based on Google search indexing)
- **Clusters** – a collection of semi-autonomous machines that acts as a single system

Distributed File System

- **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
- Challenges include:
  - Naming and Transparency
  - Remote File Access

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
- Sometimes lower level **inter-machine** interface need for cross-machine interaction

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

- Files named by combination of their host name and local name; guarantees a unique system-wide name. This naming scheme is neither location transparent nor location independent.
- NFS based. Attach remote directories to local directories, giving the appearance of a coherent directory tree; only previously mounted remote directories can be accessed transparently

Naming Schemes (Cont.)

- 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
- In practice most DFSs use static, location-transparent mapping for user-level names
  - Some support file migration
  - Hadoop supports file migration but without following POSIX standards

Remote File Access

- Consider a user who requests access to a remote file. The server storing the file has been located by the naming scheme, and now the actual data transfer must take place.
- **Remote-service mechanism** is one transfer approach. A requests for accesses are delivered to the server, the server machine performs the accesses, and their results are forwarded back to the user. One of the most common ways of implementing remote service is the RPC paradigm, which we discussed

Remote File Access (Cont.)

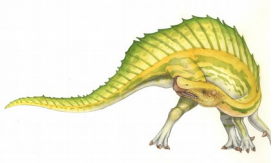
- Reduce network traffic by retaining recently accessed disk blocks in a cache, so that repeated accesses to the same information can be handled locally
  - If needed data not already cached, a copy of data is brought from the server to the user
  - Accesses are performed on the cached copy
  - Files 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** (**write-back**) – modifications are 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

End of Chapter 19