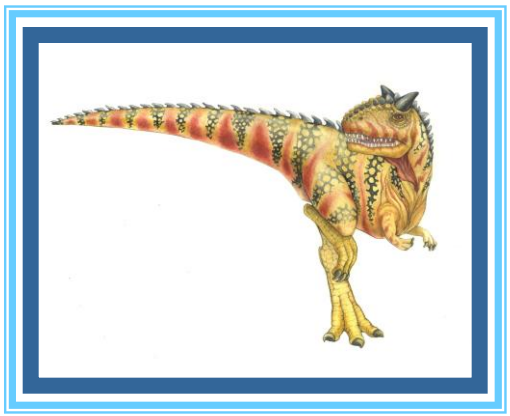
Chapter 19: Network and Distributed Systems

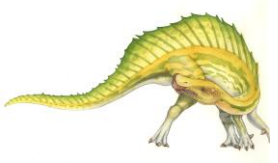


Silberschatz, Galvin and Gagne ©2018



Chapter 19: Distributed Systems

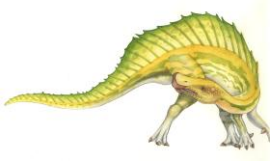
- Advantages of Distributed Systems
- Network Structure
- Communication Structure
- Network and Distributed Operating Systems
- Design Issues of Distributed Systems
- Distributed File Systems





Chapter Objectives

- Explain the advantages of networked and distributed systems
- Provide a high-level overview of the networks that interconnect distributed systems
- Define the roles and types of distributed systems in use today
- Discuss issues concerning the design of distributed file systems

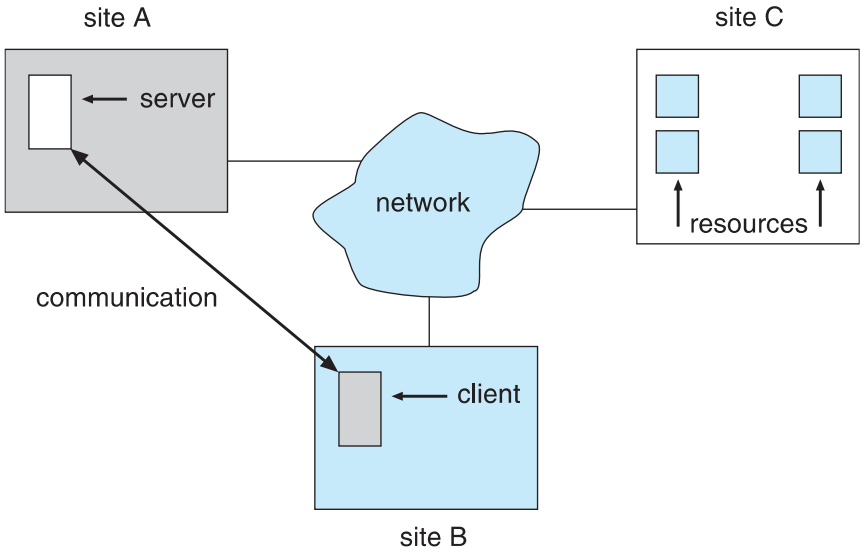


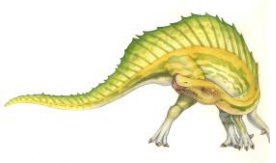
Troy University Resources



Overview

- A **distributed system** is a collection of loosely coupled nodes interconnected by a communications network
- Nodes variously called ***processors, computers, machines, hosts***
  - ***Site*** is location of the machine, ***node*** refers to specific system
  - Generally a ***server*** has a resource a ***client*** node at a different site wants to use

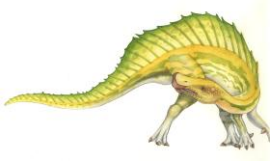






Overview (Cont.)

- Nodes may exist in a ***client-server***, ***peer-to-peer***, or ***hybrid*** configuration.
  - In client-server configuration, server has a resource that a client would like to use
  - In peer-to-peer configuration, each node shares equal responsibilities and can act as both clients and servers
- Communication over a network occurs through **message passing**
  - All higher-level functions of a standalone system can be expanded to encompass a distributed system

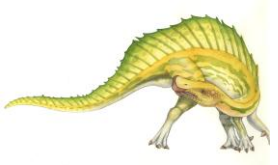


Troy University Resources



Reasons for Distributed Systems

- **Resource sharing**
  - Sharing files or printing at remote sites
  - Processing information in a distributed database
  - Using remote specialized hardware devices such as ***graphics processing units*** (GPUs)
- **Computation speedup**
  - Distribute subcomputations among various sites to run concurrently
  - **Load balancing** – moving jobs to more lightly-loaded sites
- **Reliability**
  - Detect and recover from site failure, function transfer, reintegrate failed site





Network Structure

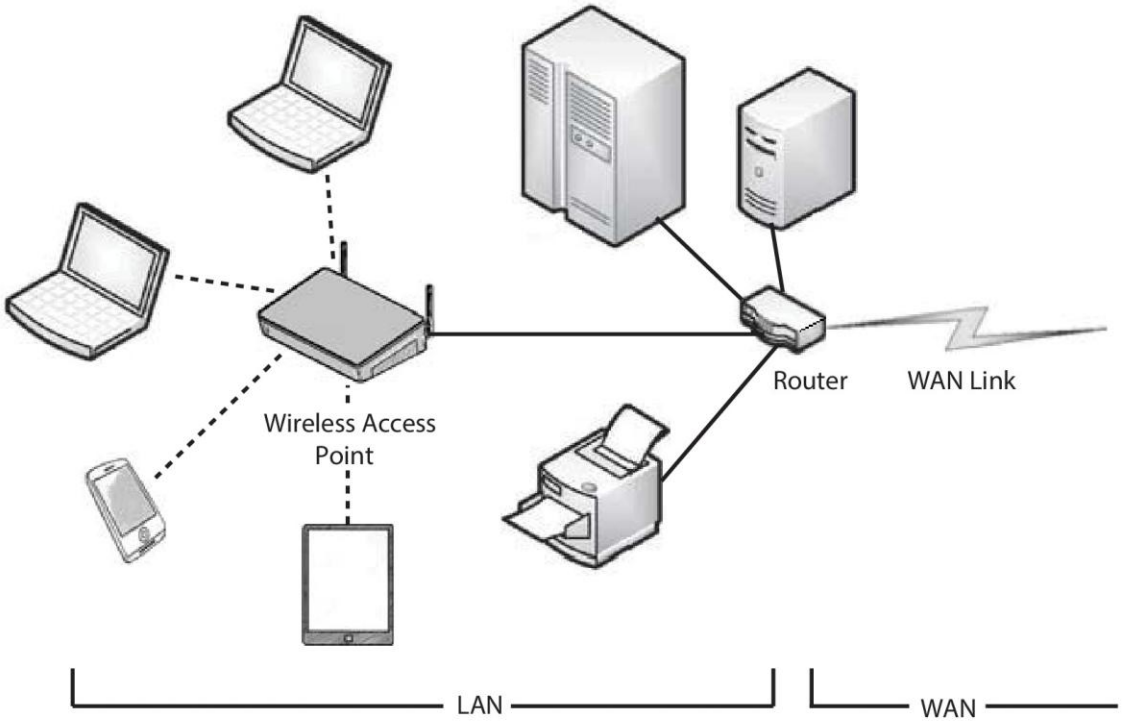
- **Local-Area Network** (**LAN**) – designed to cover small geographical area
  - Consists of multiple computers (workstations, laptops, mobile devices), peripherals (printers, storage arrays), and routers providing access to other networks
  - Ethernet and/or Wireless (**WiFi**) most common way to construct LANs
    - Ethernet is defined by IEEE 802.3. The speeds shown here are historical examples, from 10Mbps to over 10Gbps
    - WiFi is defined by IEEE 802.11. Earlier examples range from 11Mbps to over 400Mbps
    - Both standards constantly evolving



Troy University Resources



Local-Area Network (LAN)







Network Structure (Cont.)

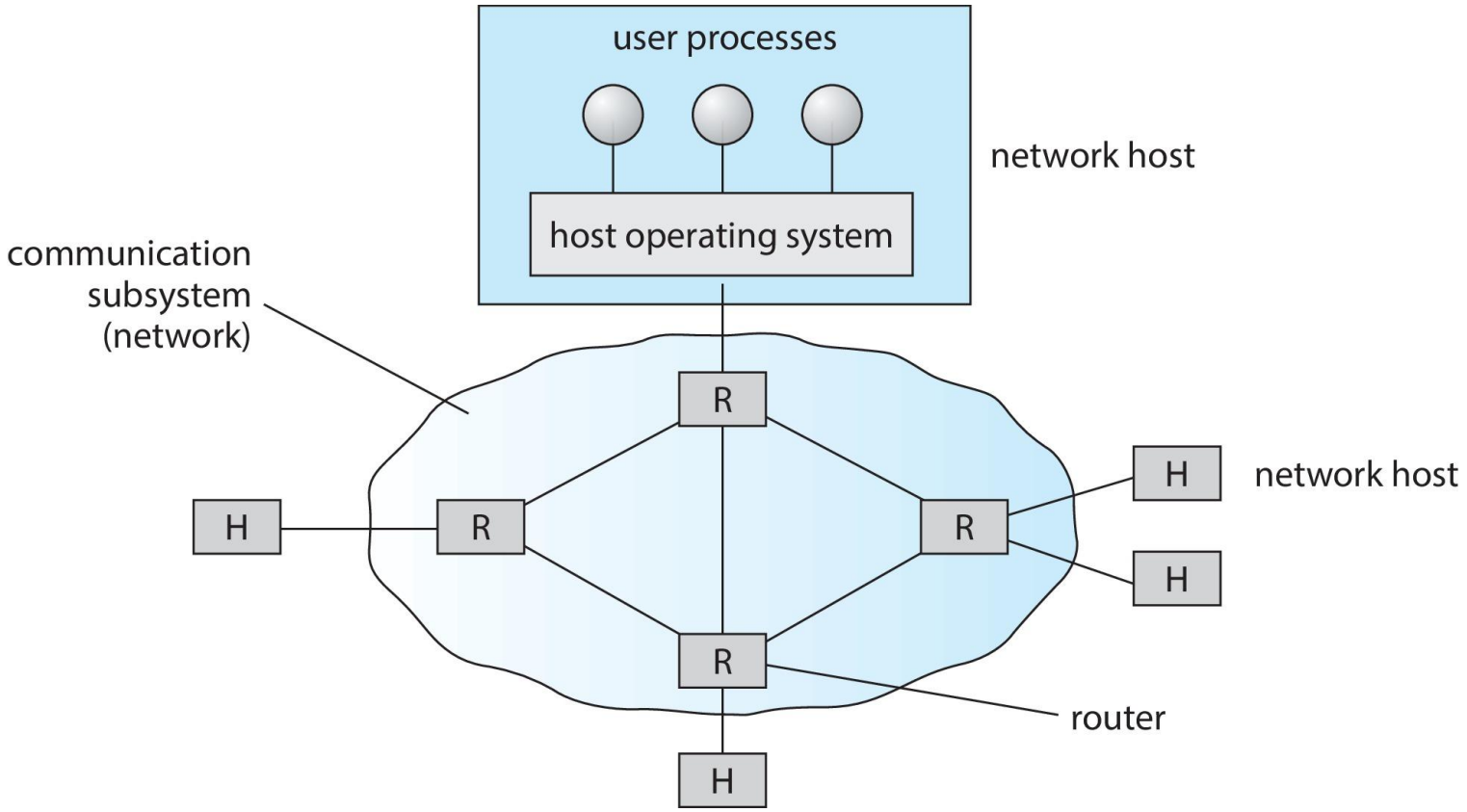
- **Wide-Area Network** (**WAN**) – links geographically separated sites
  - Point-to-point connections via links
    - Telephone lines, leased (dedicated data) lines, optical cable, microwave links, radio waves, and satellite channels
  - Implemented via **routers** to direct traffic from one network to another
  - The Internet connects networks worldwide. The World Wide Web is one service that uses it
  - Speeds vary
    - Many backbone providers have speeds at 40-100Gbps
    - Local **Internet Service Providers** (**ISPs**) may be slower
    - WAN links constantly being upgraded
  - WANs and LANs interconnect, similar to cell phone network:
    - Cell phones use radio waves to cell towers
    - Towers connect to other towers and hubs

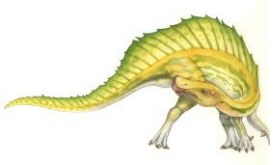


Troy University Resources



Wide-Area Network (WAN)







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**, **process-id**> pair
- **Domain name system** (**DNS**) – specifies the naming structure of the hosts, as well as name to address **resolution** (Internet)

```
/**
 * Usage: java DNSLookUp <IP name>
 * i.e. java DNSLookUp www.wiley.com
 */
public class DNSLookUp {
   public static void main(String[] args) {
      InetAddress hostAddress;

      try {
         hostAddress = InetAddress.getByName(args[0]);
         System.out.println(hostAddress.getHostAddress());
      }
      catch (UnknownHostException uhe) {
         System.err.println("Unknown host: " + args[0]);
      }
    }
}
```

Figure 19.4 Java program illustrating a DNS lookup.

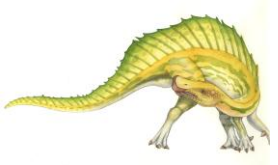




Communication Protocol

The communication network is partitioned into the following multiple layers:

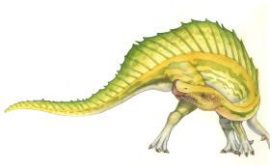
- **Layer 1: Physical layer** – handles the mechanical and electrical details of the physical transmission of a bit stream
- **Layer 2: Data-link layer** – transfers *frames* over a link, with framing and error detection. Frame sizes depend on the link protocol
- **Layer 3: Network layer** – provides 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** – provides end-to-end communication between applications using ports. Depending on the protocol, it may provide ordering, reliability, and flow control
- **Layer 5: Session layer** – implements sessions, or process-to-process communications protocols
- **Layer 6: Presentation layer** – handles data representation, such as character encoding, format conversion, compression, and encryption
- **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

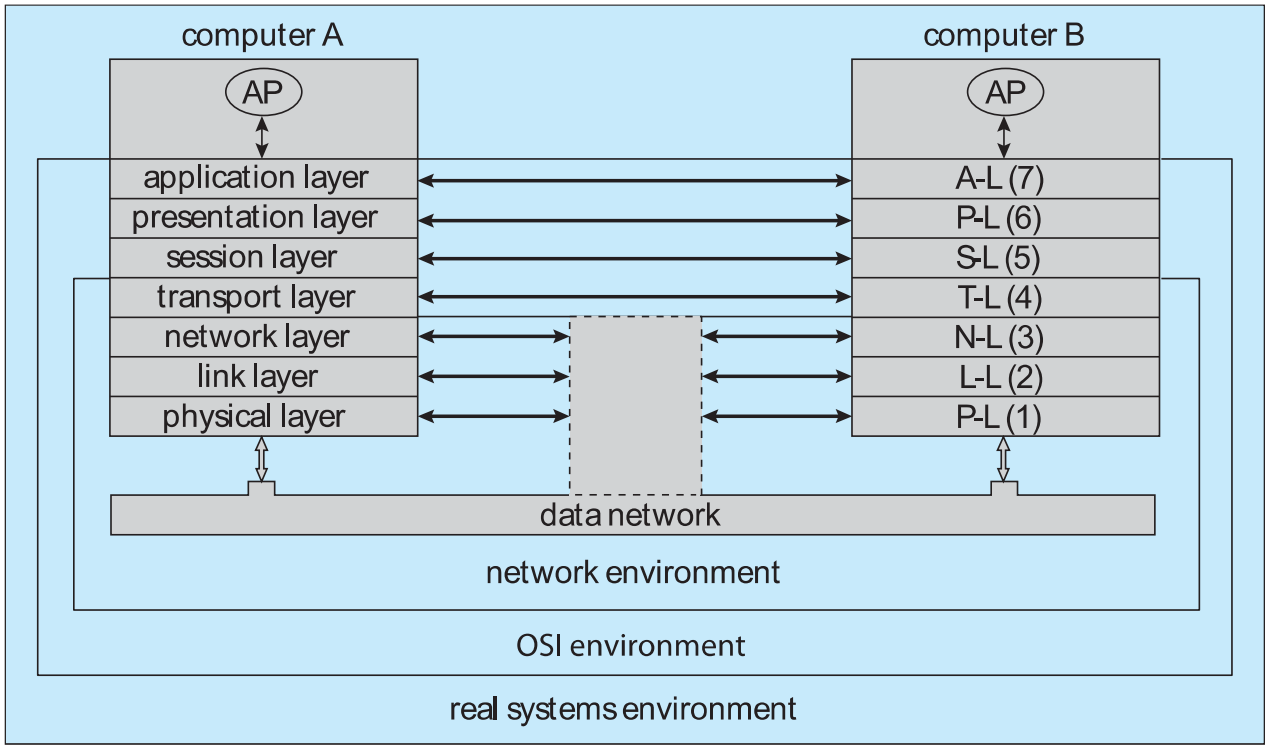


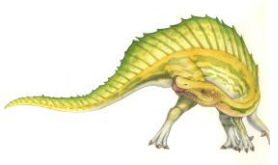
Troy University Resources



OSI Network Model

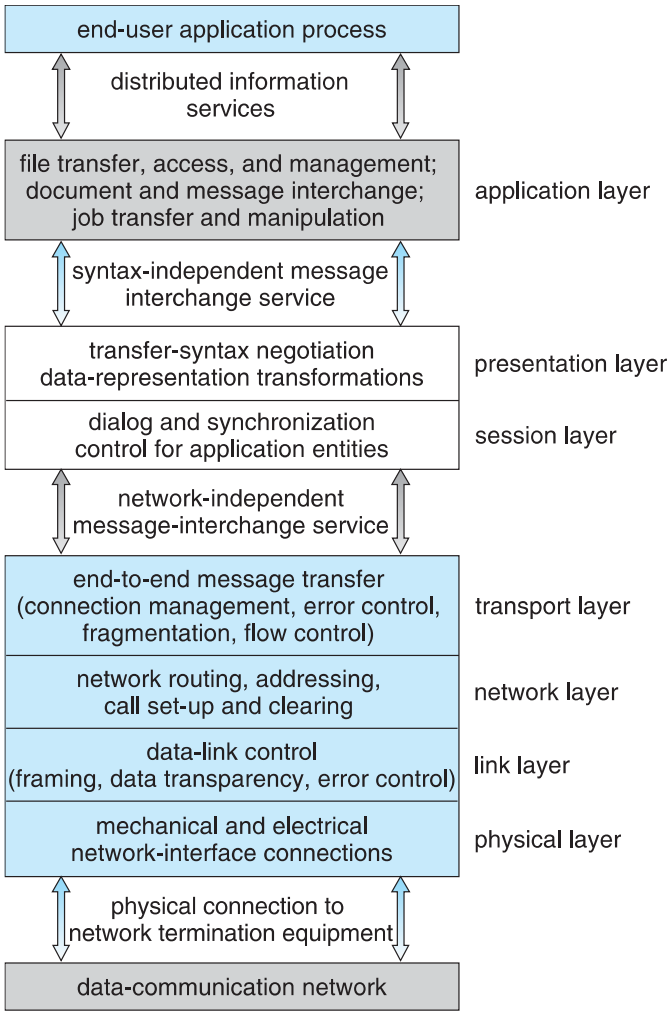
Logical communication between two computers. Lower layers can use a combination of hardware, firmware, and software

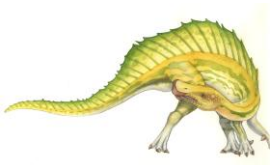






OSI Protocol Stack

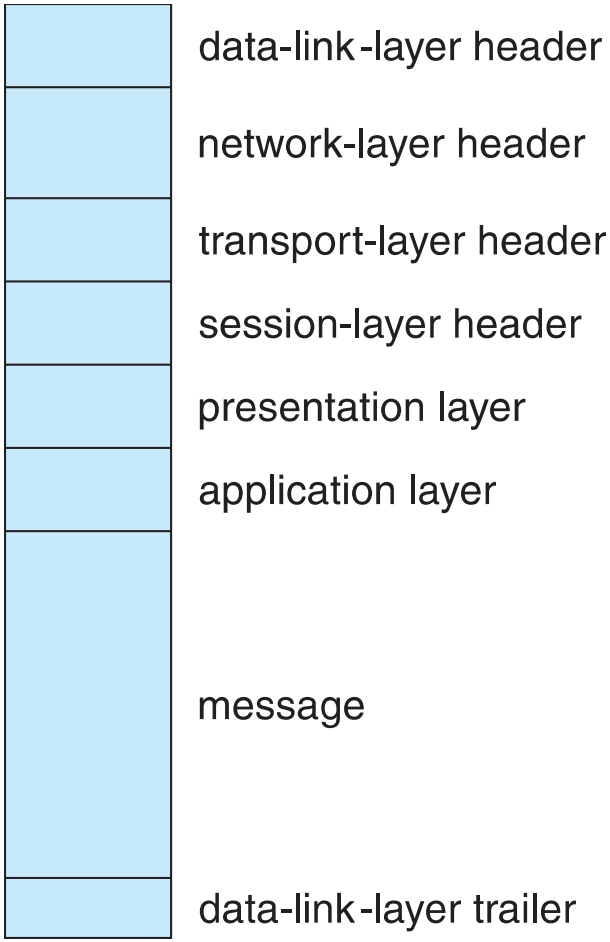


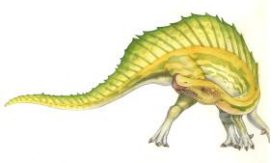


Troy University Resources



OSI Network Message

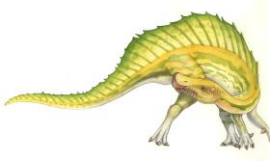






The OSI model

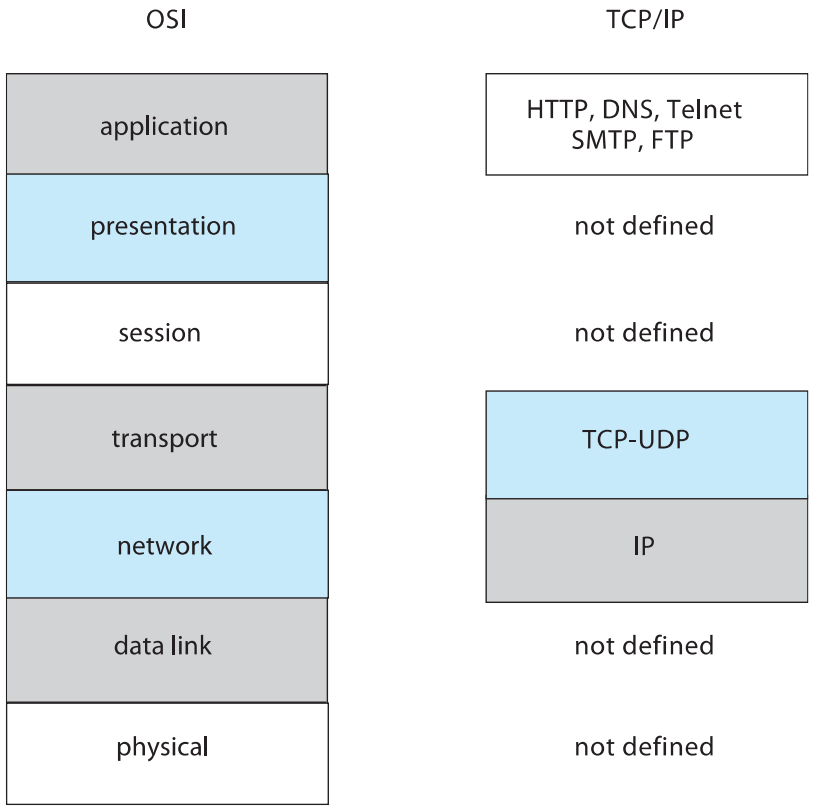
- The OSI model is a widely used conceptual reference. The OSI protocol suite saw limited deployment
- The most widely adopted protocol stack is the TCP/IP model, which has been adopted by virtually all Internet sites
- TCP/IP combines functions of several OSI layers. Fewer layers alone do not guarantee greater efficiency
- The relationship between the OSI and TCP/IP models is shown in the next slide



Troy University Resources



The OSI and TCP/IP Protocol Stacks







TCP/IP Example

- A host can have multiple IP addresses and names, associated **with its network interfaces**
  - Hierarchical and segmented
- The sender uses its routing table to choose direct delivery or a next-hop router
- Router uses segmented network part of host-id to determine where to transfer packet
  - This may repeat among multiple routers
- Destination system receives the packet
  - Packet may be complete message, or it may need to be reassembled into larger message spanning multiple packets

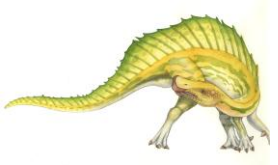


Troy University Resources



TCP/IP Example (Cont.)

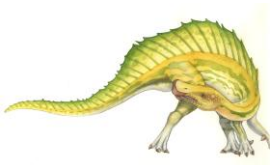
- Within a network, how does a packet move from sender (host or router) to receiver?
  - Every Ethernet/WiFi device has a **Media Access Control** (MAC) address
  - Two devices on same LAN communicate via MAC address
  - If a system needs to send data to another system, it needs to discover the IP to MAC address mapping
    - IPv4 uses ARP. IPv6 uses Neighbor Discovery
  - A broadcast uses a special network address to signal that all hosts should receive and process the packet
    - Not forwarded by routers to different networks





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

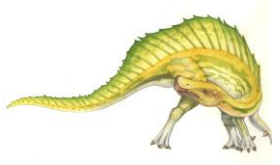


Troy University Resources



Transport Protocols UDP and TCP

- Once a host with a specific IP address receives a packet, it must somehow pass it to the correct waiting process
- Transport protocols TCP and UDP identify receiving and sending processes through the use of a port number
  - Allows host with single IP address to have multiple server/client processes sending/receiving packets
  - ***Well-known*** port numbers are used for many services
    - FTP – TCP port 21 for control
    - ssh – port 22
    - SMTP – port 25
    - HTTP – port 80
- Transport protocol can be simple or can add reliability to network packet stream





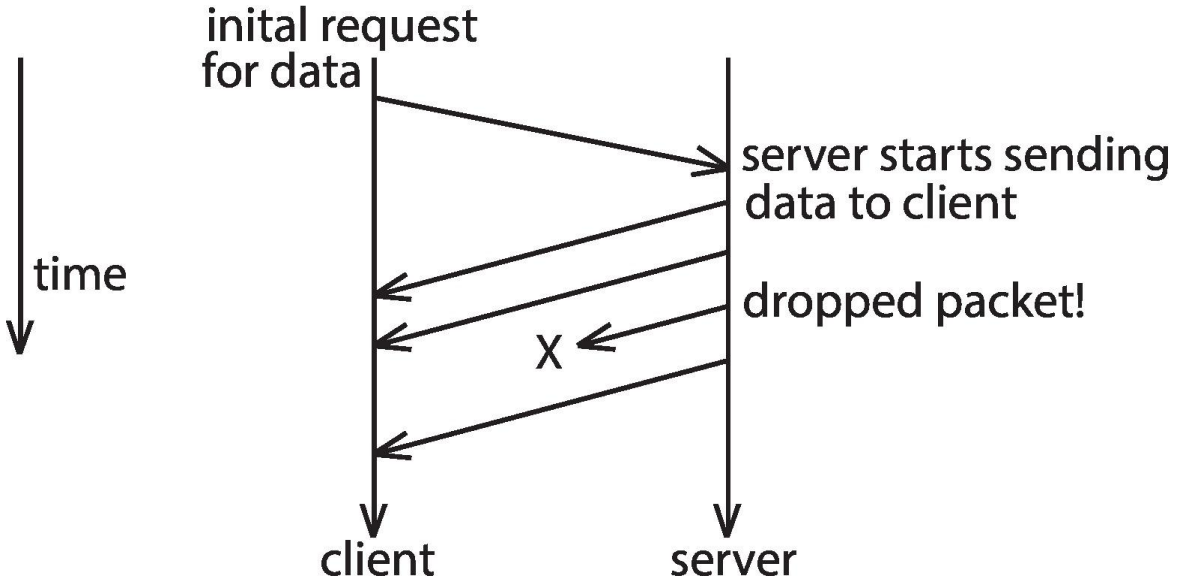
User Datagram Protocol

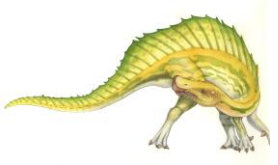
- UDP is ***unreliable*** – bare-bones extension to IP with addition of port number
  - Since there are no guarantees of delivery in the lower network (IP) layer, packets may become lost
  - Packets may also be received out-out-order
- UDP is also ***connectionless*** – no connection setup at the beginning of the transmission to set up state
  - Also no connection tear-down at the end of transmission
- UDP packets are also called **datagrams**





UDP Dropped Packet Example







Transmission Control Protocol

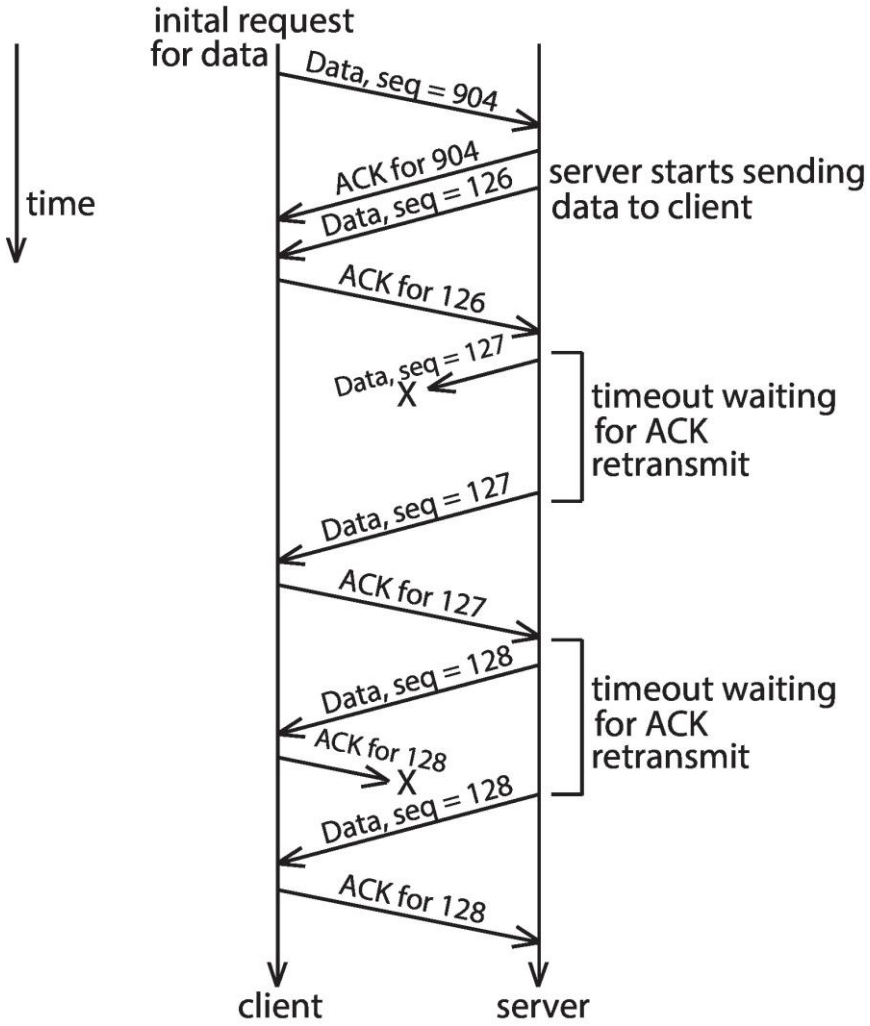
- TCP is both ***reliable*** and ***connection-oriented***
- TCP provides a reliable, ordered ***byte stream*** over an unreliable network
  - **Acknowledgements and retransmission detect and recover from loss. ACKs can be cumulative or delayed, rather than one per packet**
  - **Sequence numbers** in TCP header allow receiver to put packets in order and notice missing packets
  - Connections are initiated with series of control packets called a ***three-way handshake***
    - Connections also closed with series of control packets



Troy University Resources



TCP Data Transfer Scenario







Transmission Control Protocol (Cont.)

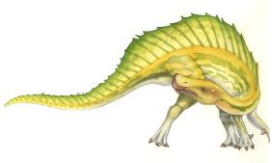
- Receiver can send a **cumulative ACK** to acknowledge series of packets
  - Server can also send multiple packets before waiting for ACKs
  - Takes advantage of network throughput
- Flow of packets regulated through **flow control** and **congestion control**
  - ***Flow control*** – prevents sender from overrunning capacity of receiver
  - ***Congestion control*** – approximates congestion of the network to slow down or speed up packet sending rate





Network-oriented Operating Systems

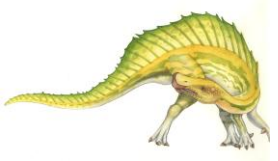
- Two main types
- **Network Operating Systems**
  - Users are aware of multiplicity of machines
- **Distributed Operating Systems**
  - Users not aware of multiplicity of machines





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)
    - `ssh kristen.cs.yale.edu`
  - Transferring data from remote machines to local machines, via the File Transfer Protocol (FTP) mechanism
  - Upload, download, access, or share files through cloud storage
- Users must change paradigms – establish a **session**, give network-based commands, use a web browser
  - More difficult for users

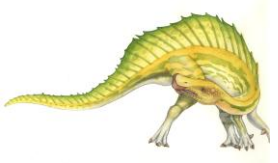


Troy University Resources



Distributed Operating Systems

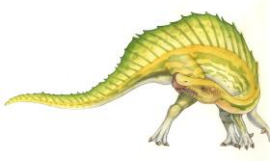
- Users not aware of multiplicity of machines
  - Access to remote resources similar to access to local resources
- **Data Migration** – transfer data by transferring entire file, or transferring only those portions of the file necessary for the immediate task
- **Computation Migration** – transfer the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





Distributed OSs (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

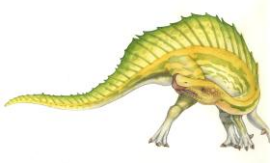


Troy University Resources



Design Issues of Distributed Systems

- We investigate three design questions:
  - ***Robustness*** – Can the distributed system withstand failures?
  - ***Transparency*** – Can the distributed system be transparent to the user both in terms of where files are stored and user mobility?
  - ***Scalability*** – Can the distributed system be scalable to allow addition of more computation power, storage, or users?





Robustness

- Hardware failures can include failure of a link, failure of a site, and loss of a message.
- A **fault-tolerant system** can tolerate a certain level of failure
  - Degree of fault tolerance depends on design of system and the specific fault
  - The more fault tolerance, the better!
- Involves ***failure detection***, ***reconfiguration***, and ***recovery***

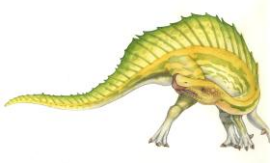


Troy University Resources



Failure Detection

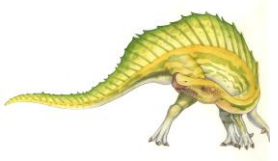
- Detecting hardware failure is difficult
- To detect a link failure, a **heartbeat** protocol can be used
- Assume Site A and Site B have established a link
  - At fixed intervals, each site will exchange an *I-am-up* message indicating that they are up and running
- If Site A does not receive a message 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 receives no reply from Site B before a timeout, it suspects a failure
- 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
- Site A cannot distinguish these failures from a delayed response or an overloaded site using the timeout alone

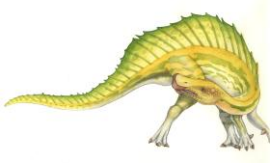


Troy University Resources



Reconfiguration and Recovery

- 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





Transparency

- The distributed system should appear as a conventional, centralized system to the user
  - User interface should not distinguish between local and remote resources
    - Example: Network File system (NFS)
  - User mobility allows users to log into any machine in the environment and see his/her environment
    - Example: Lightweight Directory Access Protocol (LDAP) plus desktop virtualization

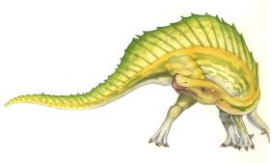


Troy University Resources



Scalability

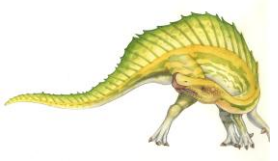
- As demands increase, the system should easily accept the addition of new resources to accommodate the increased demand
  - Reacts gracefully to increased load
  - Adding more resources may generate additional indirect load on other resources if not careful
  - Data **compression** or **deduplication** can cut down on storage and network resources used





Distributed File System

- **Distributed file system** (**DFS**) – a file system whose clients, servers, and storage devices are dispersed among the machines of a distributed system
  - Should appear to its clients as a conventional, centralized file system
- Key distinguishing feature is management of dispersed storage devices

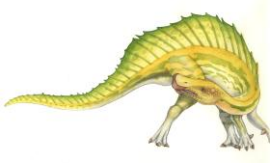


Troy University Resources



Distributed File System (Cont.)

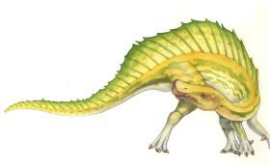
- **Service** – software entity running on one or more machines and providing a particular type of function to a priori unknown clients
- **Server** – service software running on a single machine
- **Client** – process that can invoke a service using a set of operations that forms its client interface
- 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





Distributed File System (Cont.)

- Two widely-used architectural models include **client-server model** and **cluster-based model**
- Challenges include:
  - Naming and transparency
  - Remote file access
  - Caching and cache consistency

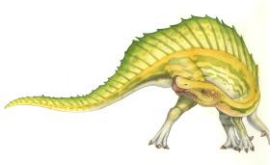


Troy University Resources



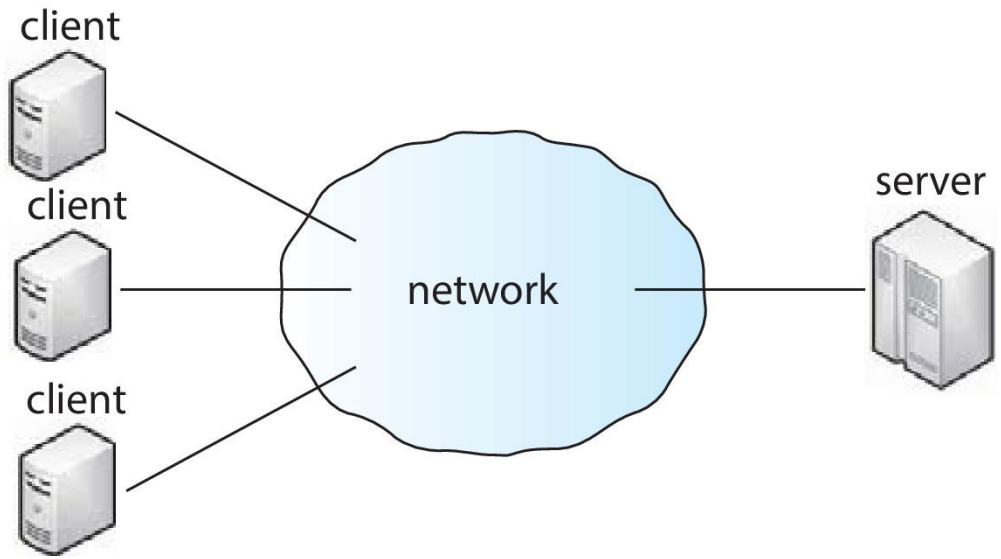
Client-Server DFS Model

- Server(s) store both files and metadata on attached storage
  - Clients contact the server to request files
  - Sever responsible for authentication, checking file permissions, and delivering the file
  - Changes client makes to file must be propagated back to the server
- Popular examples include ***NFS*** and ***OpenAFS***
- A single unreplicated server is a point of failure
- Server presents a bottleneck for all requests of data and metadata
  - Could pose problems with scalability and bandwidth





Client-Server DFS Model (Cont.)



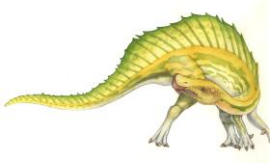


Troy University Resources



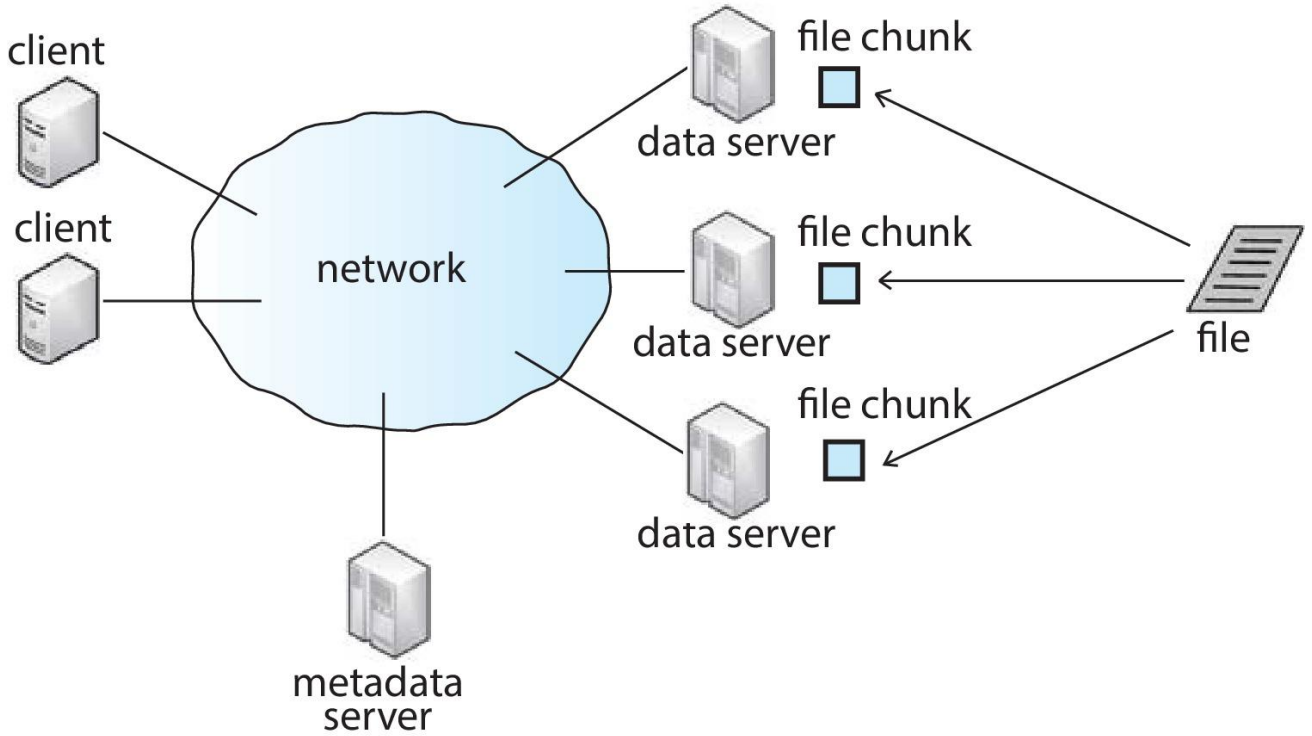
Cluster-based DFS Model

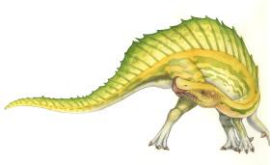
- Built to be more fault-tolerant and scalable than client-server DFS
- Examples include the **Google File System** (**GFS**) and **Hadoop Distributed File System** (**HDFS**)
  - Clients connected to master metadata server and several data servers that hold “chunks” (portions) of files
  - Metadata server keeps mapping of which data servers hold chunks of which files
    - As well as hierarchical mapping of directories and files
  - File chunks replicated *n* times





Cluster-based DFS Model (Cont.)



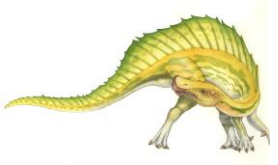


Troy University Resources



Cluster-based DFS Model (Cont.)

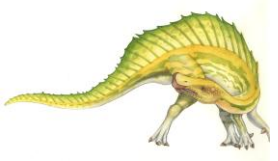
- GFS design was influenced by following observations:
  - Hardware component failures are the norm rather than the exception and should be routinely expected.
  - Files stored on such a system are very large.
  - Most files are changed by appending new data to the end rather than overwriting existing data.
  - Redesigning the applications and file system API increases system flexibility
    - Requires applications to be programmed specially with new API
- Modularized software layer **MapReduce** can sit on top of GFS to carry out large-scale parallel computations while utilizing benefits of GFS
  - **Hadoop** framework also stackable and modularized





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

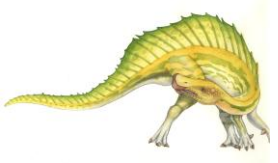


Troy University Resources



Naming Structures

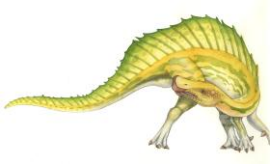
- **Location transparency** – file name does not reveal the file's physical storage location
- **Location independence** – file name does not need to be changed when the file's physical storage location changes
- In practice most DFSs use static, location-transparent mapping for user-level names
  - Some support file migration (e.g. OpenAFS)
  - Hadoop supports file migration but without following POSIX standards; hides information from clients
  - Amazon S3 provides object storage through APIs, with physical placement managed by the service





Naming Schemes

- Three approaches:
  - 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.
  - Attach remote directories to local directories, giving the appearance of a coherent directory tree; only previously mounted remote directories can be accessed transparently
  - Single global name structures spans all files in the system. If a server is unavailable, some arbitrary set of directories on different machines also becomes unavailable

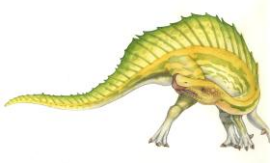


Troy University Resources



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





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**

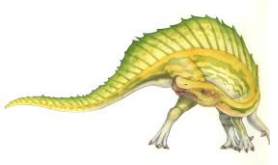


Troy University Resources



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** – propagate changes to the server before acknowledging the write. Durability depends on the server's commit policy
  - 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
  - With a volatile cache, a client crash can lose changes not yet written back
  - 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

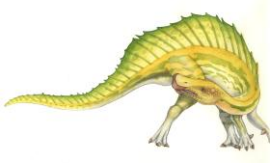


Troy University Resources



Consistency

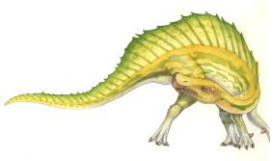
- Is locally cached copy of the data consistent with the master copy?
- **Client-initiated approach**
  - Client initiates a validity check
  - Server checks whether the local data are consistent with the master copy
- **Server-initiated approach**
  - Server records, for each client, the (parts of) files it caches
  - When server detects a potential inconsistency, it must react





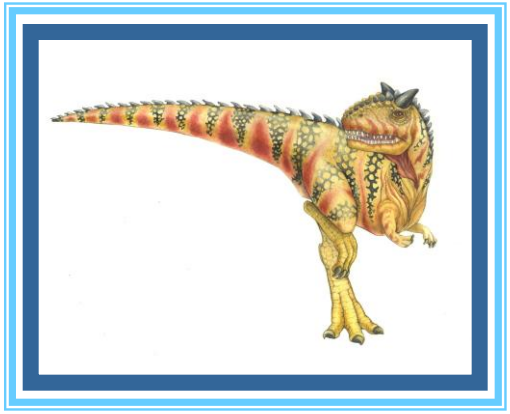
Consistency (Cont.)

- In cluster-based DFS, cache-consistency issue more complicated due to presence of metadata server and replicated file data chunks
  - HDFS allows *append-only* write operations (no random writes) and a *single* file writer
  - GFS allows *random* writes with *concurrent* writers
- Complicates write consistency guarantees for GFS while simplifying it for HDFS



Troy University Resources

End of Chapter 19



Silberschatz, Galvin and Gagne ©2018