Chapter 19: Network and Distributed Systems

Silberschatz, Galvin and Gagne ©2018

Overview

- A **distributed system** is a collection of loosely coupled nodes interconnected by a communication network
- Nodes variously called ***processors, computers, machines, hosts***
  - ***Site*** is location of the machine, ***node*** refers to a specific system at a site

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**

Reasons for Distributed Systems

- **Resource sharing**
  - Accessing files at remote sites
  - Retrieving 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**
  - If one site fails, the remaining sites can continue operating
  - System must be able to detect and recover from site failure, transfer function of failed site to anther site, and reintegrate failed site when it recovers or is repaired

Network Structure

- **Local-Area Network** (**LAN**) – designed to cover small geographical area (e.g., a building)
  - Consists of multiple computers (workstations, laptops, mobile devices), peripherals (printers, storage arrays), and routers providing access to other networks
  - Ethernet and WiFi most common way to construct LANs
    - Ethernet defined by standard IEEE 802.3 with speeds typically varying from 10Mbps to over 10Gbps
    - WiFi defined by standard IEEE 802.11 with speeds typically varying from 11Mbps to over 400Mbps.

Network Structure (Cont.)

- **Wide-Area Network** (**WAN**) – links sites that are physically distributed over a large geographical area
  - Typical links are telephone lines, leased (dedicated data) lines, optical cable, radio waves, and satellite channels
  - Implemented via **routers** to direct traffic from source to destination

The Internet

- The Internet is a WAN that enables hosts world wide to communicate
- The Internet is a **network of networks**
  - Internet consists of interconnected Internet Service Provider (ISP) networks
  - **Access ISPs** (e.g., residential ISPs, university ISPs) provide Internet access to hosts
  - Access ISPs are interconnected through regional/national/global ISPs
- Speeds vary
  - Many backbone providers have speeds at 40-100Gbps
  - Access ISPs have slower speeds
  - WAN links constantly being upgraded

Network Layers

- The communication network is partitioned into multiple layers
- The International Standards Organization created the Open Systems Interconnection (OSI) model, which consists of 7 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 *frames*, 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, handles addressing, and maintaining routing tables

Communication Protocol (Cont.)

- **Layer 4: Transport layer** – responsible for message transfer between application processes, including partitioning messages into packets, maintaining packet order, and controlling flow to avoid congestion
- **Layer 5: Session layer** – implements communication sessions
- **Layer 6: Presentation layer** – resolves the differences in formats among the various sites in the network
- **Layer 7: Application layer** – interacts directly with the users, deals with file transfer, remote-login, email, etc.

OSI Network Model

Logical communications between two computers

Physical Flow of Data

OSI Protocol Stack

OSI network message

The OSI and TCP/IP Protocol Stacks

- The OSI model was developed in the late 1970s and is not in use today
- The Internet adopts the TCP/IP model
- The TCP/IP protocol stack has fewer layers than the OSI model
  - It combines functions of the presentation layer in the application layer
  - It combines functions of the session layer in the transport layer

TCP/IP Example

- Every Internet host has a **name** (e.g., www.google.com) and an associated **IP address** (e.g., 216.58.192.196)
- Sending host sends packet to nearest router
- Router uses destination IP address field in packet to determine where to forward packet
  - This may repeat among multiple routers
- Destination host receives the packet
  - Packet may be complete message, or it may need to be reassembled into larger message spanning multiple packets

Transport Protocols UDP and TCP

- Once a host with a specific IP address receives a packet, it must pass it to the correct destination process
- Transport protocols TCP and UDP identify sending and receiving processes through the use of port numbers
  - 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 – port 21
    - SSH – port 22
    - SMTP – port 25
    - HTTP – port 80

User Datagram Protocol (UDP)

- UDP is ***unreliable***
  - 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
  - Also no connection tear-down at the end of transmission

UDP dropped packet example

Transmission Control Protocol (TCP)

- TCP is both ***reliable*** and ***connection-oriented***
- TCP provides in-order, no loss delivery of packets across an unreliable network
  - Whenever host sends packet, the receiver must send an **acknowledgement packet** (ACK). If ACK not received before a timer expires, sender will resend.
  - **Sequence numbers** in TCP header allow receiver to put packets in order and notice missing packets
- Flow of packets regulated through **flow control** and **congestion control**
  - ***Flow control*** – prevents sender from overrunning capacity of receiver
  - ***Congestion control*** – slow down or speed up packet sending rate based on congestion level in network

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

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

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***

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 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: NFS
  - User mobility allows users to log into any machine in the environment and see his/her environment
    - Example: LDAP plus desktop virtualization

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

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

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***
- Design suffers from single point of failure if server crashes
- Server presents a bottleneck for all requests of data and metadata
  - Could pose problems with scalability and bandwidth

Client-Server DFS Model (Cont.)

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.)

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