DCS

Unit - 1

Goals of Distributed Systems

- Resource availability
  - Printers, files, computers, webpages, etc.
  - Make it easy for end-users to access remote resources and share them in a competent way
- Transparency (Clearness)
  - Should practically hide the fact that resources and processes are scattered across a network
  - Should present itself to end-users and applications as if it were only a single automated system
- Openness (Directness)
  - Offers services according to standard rules that describe syntax and semantics of those services
- Scalability
  - Measured along 3 dimensions: size, geographical distance, administration

Transparency

Types:
- Access
  - Hiding differences in data or facts representation and method with which resources can be retrieved by users
- Location
  - Users cannot tell where a resource is physically located in the system
- Migration
  - Resources can be moved from one location to another without affecting how they can be accessed
- Relocation
  - Resource may be moved from one location to another while in use
- Replication
  - Hides that several replicas of a resource exist
- Concurrency
  - Hides that resource may be shared by several competitive users
- Failure
  - Hides failure and recovery of a resource

Note: Highest degree of transparency may lead to loss of performance that users may not accept. Hence, it is not always a good aim.

Scalability

Issues:
- Synchronous communication
- Inconsistent data
- Speed
- Location

Resolution Techniques:
- Replication
  - Creates numerous duplicates of resources
  - Two important issues:
    - Placement of replicas
    - Improves dependability/consistency and handiness of services
  - Mechanisms by which they are kept consistent
    - If change is made to an object, should be clear to everyone in system
    - Timestamps used to differentiate copies if inconsistent updates received
- Distribution
  - Used to manage scale in distributed systems, allows maintained information to be extended across several servers
  - Minimises database size
  - Spreads load across servers, reducing number of requests handled by each
- Caching
  - Minimises load on servers and thus network
  - Cached data can be accessed faster if a new request is made
  - Cached is short-term data; different than replication

Design Issues

- Transparency
- Communication
  - Networking infrastructure (interconnections network software)
  - Appropriate communication primitives and models and their implementation
  - Communication primitives:
    - Send
    - Receive
    - Remote Procedure Call (RPC)
  - Communication models:
    - Client-server communication: message exchange between two processes, requester and server
    - Group multicast: target of a message is a set of processes
- Performance
  - Speed of communication infrastructure
  - Extent to which fault tolerance provided
  - Flexibility in workload allocation
- Scalability
  - Performance loss with high number of users
  - Reasonable cost of adding resources
  - Software resources should not run out
- Heterogeneity
  - Different hardware
  - Different software
  - Unconventional devices
  - Diverse networks and protocols
- Openness
  - Every service is equally accessible to every client
  - Easy to implement, install and debug new services
  - Users can write and install own services
- Reliability and Fault Tolerance
  - Mask the fault: work with reduced performance without loss of data
  - Fail gracefully: react to fault in a predictable way and stop functionality for a short period without data loss
- Security
  - Confidentiality: protection against disclosure to unauthorised entity
  - Integrity: protection against corruption and alteration
  - Availability: keep resource handy or accessible

Middleware

- Additional layer on top of network OS that implements general purpose services
- Goal is to provide distribution transparency
- Services:
  - Communication: abandon socket-based message passing in favour of-
    - Procedure calls across networks
    - Remote-object method invocation
    - Message-queueing systems
    - Event notification service
  - Information system: help manage data
    - Large scale, system wide naming services
    - Advanced directory services (search engines)
    - Location services to track mobile objects
    - Persistent storage facilities
    - Data caching and replication
  - Control: giving applications control over when, where and how they access data
    - Distributed transaction processing
    - Code migration
  - Security: secure processing and communication
    - Authentication and authorisation
    - Encryption
    - Auditing

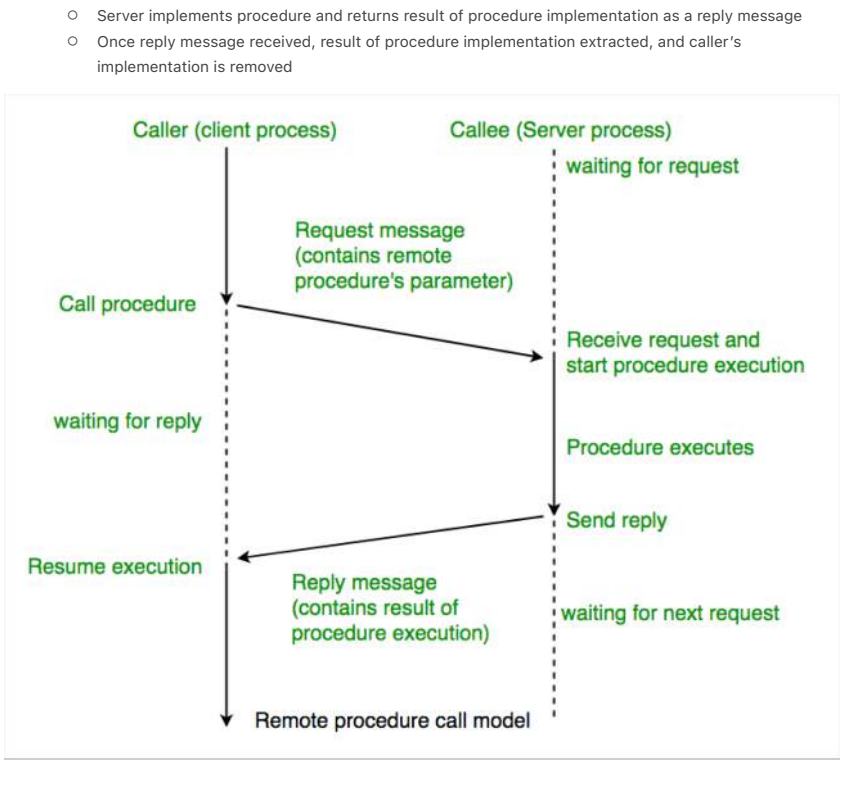
System Architectures

- Client Server
  - Server process provides a service
  - Client sends a request, processed at server and reply returned to client
  - Implementation varies as to how parts are separated over client and server roles
  - Decomposed into three logical parts: interface, application logic, data
- Vertical Distribution (Multi-Tier)
  - Focuses on splitting up a server's functionality over multiple computers
  - Client request directed to first server
  - Server requests services of next server, until final server reached
  - Each server responsible for different tier in success of original client request
  - Flexibility allowed by internal functionality of each server to be modified as long as interfaces provided remain same
- Horizontal Distribution
  - Replicating a server's functionality over multiple computers
  - Used to improve scalability (by reducing load on individual server) and reliability (by providing redundancy)
  - Possible to combine vertical and horizontal distribution
- Peer-to-Peer
  - Assumes each machine has somewhat equivalent capabilities
  - No machine consigned to a specific set of services
- Hybrid

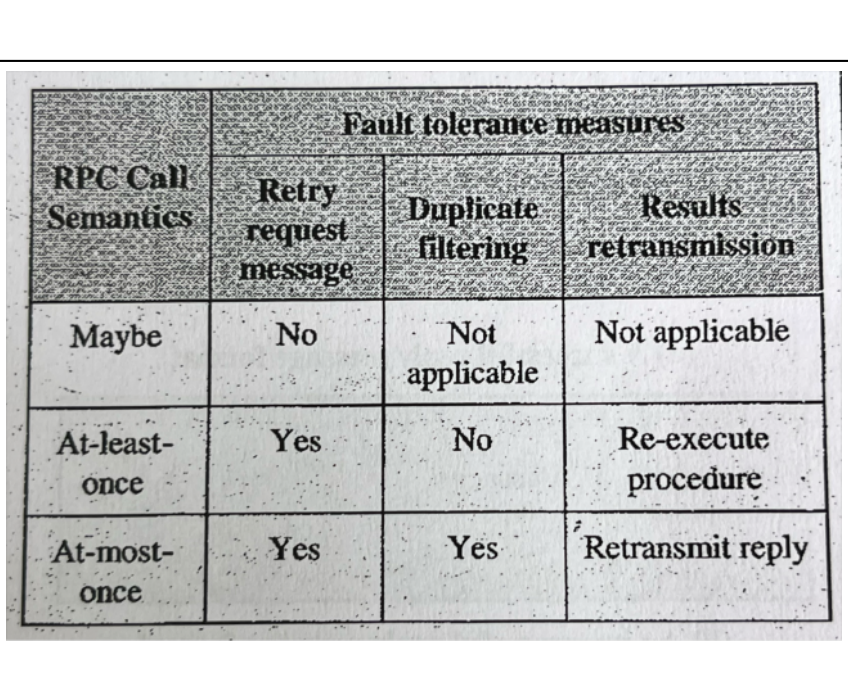
Unit - 2

RPC

- IPC plays an important role in distributed systems. RPC emerged out of this requirement.
- RPC is extension of procedure call mechanism; allows a call to be made to a procedure that does not exist in address space of calling process
- RPC uses a message passing mechanism for information exchange between caller and callee process
- Called procedure called remote procedure; may be on same or different computer as that of calling process
- When RPC is made:
  - Caller (client) sends a call (request message) to callee (server)
  - Server implements procedure and returns result of procedure implementation as a reply message
  - Once reply message received, result of procedure implementation extracted, and caller's implementation is removed


Caller (client process)
Callee (Server process)
waiting for request
Request message (contains remote procedure's parameter)
Call procedure
Receive request and start procedure execution
waiting for reply
Procedure executes
Send reply
Resume execution
Reply message (contains result of procedure execution)
waiting for next request
Remote procedure call model

RPC Semantics



RPC Call Semantics	Fault tolerance measures		
	Retry request message	Duplicate filtering	Results retransmission
Maybe	No	Not applicable	Not applicable
At-least-once	Yes	No	Re-execute procedure
At-most-once	Yes	Yes	Retransmit reply

RMI

- Process of activating a method on a remotely running object
- Communication between client and server handled by two intermediate objects: Stub object (on client side) and Skeleton object (on server-side)
- **Stub Object:** builds an information block and sends this information to the server. The block contains:
  - Identifier of remote object to be used
  - Method name to be invoked
  - Parameters to remote JVM
- **Skeleton Object:** passes the request from stub to remote object
  - It calls the desired method on the real object present on server
  - It forwards the parameters received from the stub object to the method

RMI vs RPC

RPC	RMI
Remote Procedure Call	Remote Method Invocation
Call processes which will run on different computer than user	Invoke processes which will run on different computer than user
Invokes functions	Invokes methods
Not object oriented	Object oriented
Calls specific subroutines; does not deal with objects	Uses OO paradigm; user needs to know object and method to invoke
Old protocol based on C	Developed by Java, uses JVM

Socket and Connection-Oriented Communication

- Communication end point from which data can be read / to which data can be written
- In practise, an IP address along with a port is known as a socket
- Communication-oriented communication has the following sequence of operations:
  - Establishing Connection – requires a session connection to be established before any data is transported with a direct physical connection among sessions
  - Transferring Data or Message – When session connection is established, data is transferred
  - Releasing the Connection – After transferring data, we release the connection

Cristian's Algorithm for Clock Synchronisation

- Used to synchronise time with a time server by client processes
- Works well with low-latency networks where round trip time is short
- Algorithm:
  - process on client machine sends request to fetch clock time (time at the server) to the Clock Server at time T0
  - Clock Server listens to request made by client process, returns response in form of clock server time
  - Client process fetches response from the Clock Server at time T1 and calculates synchronised client clock time using: **Tclient = Tserver + (T1 - T0) / 2**
    - Tclient = synchronised clock time
    - Tserver = clock time returned by server
    - T0 = time at which request was sent by client
    - T1 = time at which response was received by client
- Time at the client-side differs from actual time by at most **(T1 - T0) / 2** seconds. Hence, error in synchronisation can be at most **(T1 - T0) / 2** seconds

Centralised Mutual Exclusion Algorithm

- One process from several available processes elected as coordinator
- Every time a process wants to access a pooled resource, it directs a request message to coordinator process
- If no other process currently retrieving that resource, coordinator sends response granting permission
- Rigorous technique used to reject permission is system dependent
- Coordinator abstains from replying, blocks new process B waiting for a reply; can also send "Permission denied"
- Process A sends message to coordinator, freeing its limited access of pooled resource
- Coordinator takes B off queue of delayed requests and forwards with grant message
- Algorithm assures mutual exclusion
- No starvation problem
- Easy to implement, requires 3 messages per use of resource: request, grant and release
- Coordinator is single point of failure
- Single coordinator can become a bottleneck in a large system

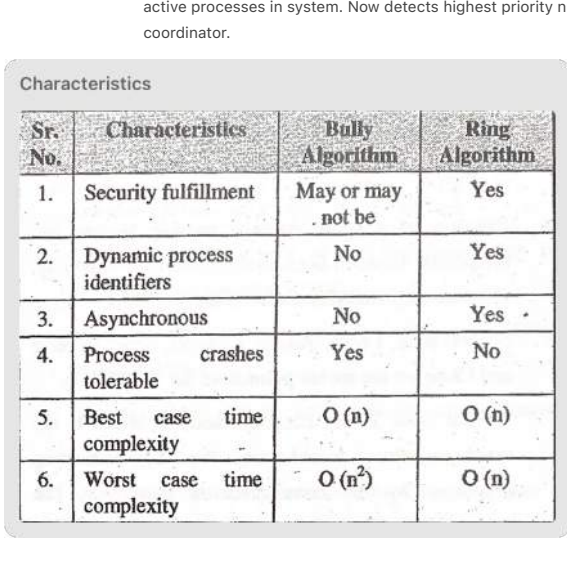
Election Algorithms

Bully Algorithm

- Hector Garcia Molina, 1982
- Applies to system where every process can send a message to every other process in the system
- 3 types of messages used to elect a coordinator: election, coordinator, OK
- Steps:
  - Process sends an election message to all other processes having higher process number wrt sending process
  - If sending process receives OK, it gives up and waits for receipt of coordinator message from other processes
  - New process sends an election message to all processes having process number wrt new process
  - All processes involved give up election except for a process having higher process number amongst all
  - Process having highest process number elected as new coordinator and transfers a coordinator message to remaining processes to broadcast itself as newly elected coordinator

Ring Algorithm

- applies to systems organized as a ring(logically or physically)
- Assume that link between processes is unidirectional and every process can message the process on its right only
- Data structure used is **active list** (list with priority number of all active processes in system)
- Steps:
  - If process P1 detects coordinator failure, creates new active list (initially empty), sends election message to neighbour on right and adds number 1 to its active list
  - If P2 receives message elect from processes on left, responds in three ways:
    - If message received does not contain 1 in active list, P1 adds 2 to its active list and forwards the message
    - If this is the first election message it has received/sent, P1 creates new active list with numbers 1 and 2. Then sends election message 1 followed by 2
    - If P1 receives its own election message 1 then active list for P1 now contains numbers of all active processes in system. Now detects highest priority number from list and elects it as new coordinator.

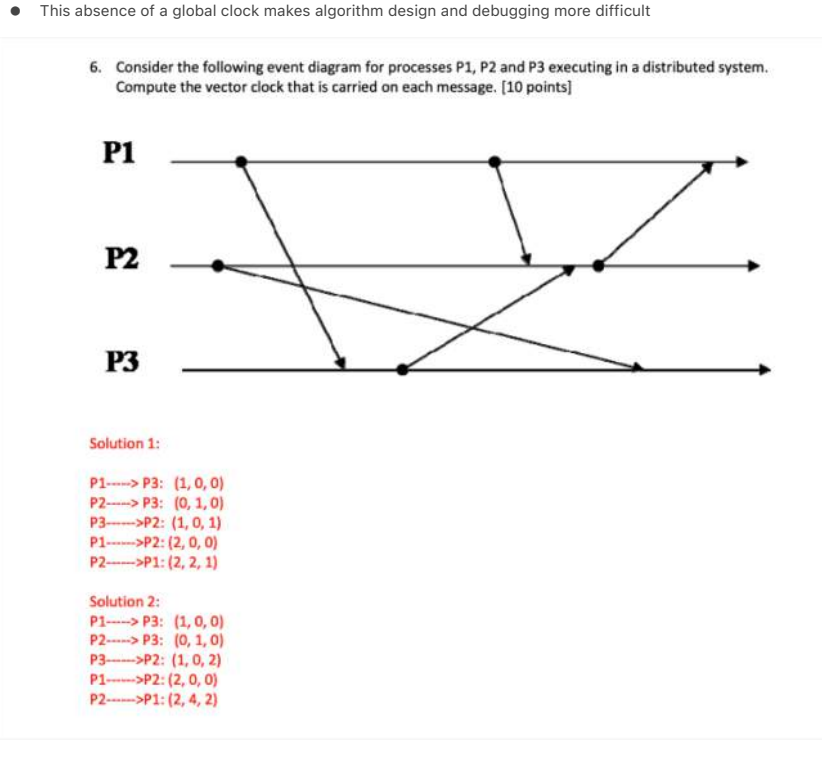


Characteristics

Sr. No.	Characteristics	Bully Algorithm	Ring Algorithm
1.	Security fulfillment	May or may not be	Yes
2.	Dynamic process identifiers	No	Yes
3.	Asynchronous	No	Yes
4.	Process crashes tolerable	Yes	No
5.	Best case time complexity	O (n)	O (n)
6.	Worst case time complexity	O (n²)	O (n)

Precision Time and Global Clock

- Precision time is needed for:
  - Temporal ordering of events
  - Collecting up-to-date information on the state of the integrated system
  - Scheduling of processes
- In a DS, there are a lot of systems and each system has its own clock, running at a different rate/granularity thus asynchronous
- After one local clock cycle, clocks are out of sync and no clock has the exact time
- This absence of a global clock makes algorithm design and debugging more difficult



6. Consider the following event diagram for processes P1, P2 and P3 executing in a distributed system. Compute the vector clock that is carried on each message. [10 points]

Solution 1:

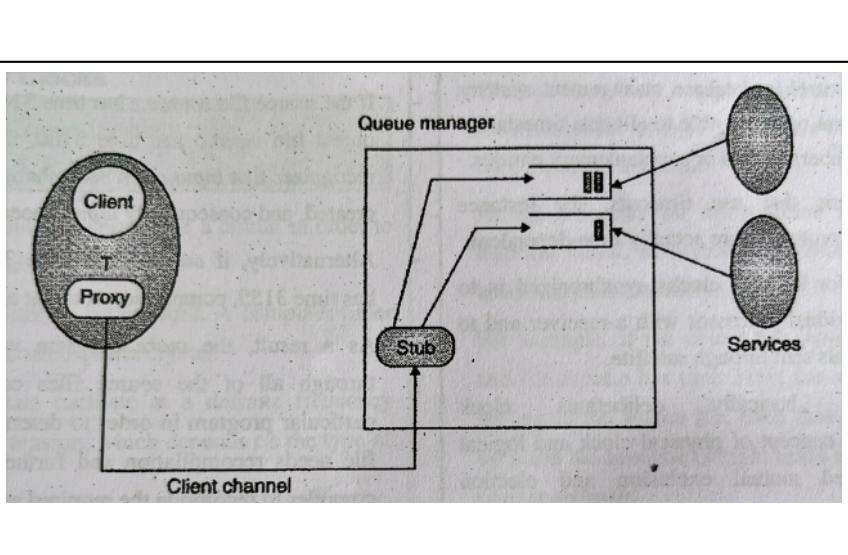
P1-----> P3: (1, 0, 0)
P2-----> P3: (0, 1, 0)
P3------>P2: (1, 0, 1)
P1------>P2: (2, 0, 0)
P2------>P1: (2, 2, 1)

Solution 2:

P1-----> P3: (1, 0, 0)
P2-----> P3: (0, 1, 0)
P3------>P2: (1, 0, 2)
P1------>P2: (2, 0, 0)
P2------>P1: (2, 4, 2)

IBM's Websphere Message-Queuing System

- point-to-point type of indirect communication system
- sender puts message into queue, then removed by a particular process
- Three kinds of receive:
  - blocking - blocks queue until suitable message is obtainable
  - non-blocking - checks queue status, then returns a message if available else returns not available indication
  - notify - gives event notification if message available in related queue
- queue manager manages a set of queues in Websphere MQ
- queue managers at a particular node permite apps to access queues by Message Queue Interface (MQI)
- MQI allows apps to perform MQPUT/MQGET or MQCONN/MQDISC



Unit - 3

Replication	Caching
Deals with a server	Associated with a client
More persistent, secure, complete, accurate and available	Dependent on a replica. Can be helpful only by periodic revalidation wrt a replica
Long-term storage	Short-term storage

Active and Passive Replication

- Active
  - Client request goes to all replicas.
  - Every replica receives client request in same order else system will get inconsistent.
  - No need for coordination because each copy processes same request in same sequence.
  - All replicas respond to client's request.
- Passive
  - Client request goes to primary/main replica
  - More replicas act as backup for primary replica.
  - Primary replica informs all other backup replicas about any modification done.
  - Response is returned to client by a primary replica.
  - Periodically primary replica sends some signal to backup replicas to let them know that it is working perfectly fine.
  - In case of failure of a primary replica, a backup replica becomes the primary replica.

Client-centric Consistency Model

Provides an efficient but weaker form of consistency. Types:
- Eventual Consistency - all replicas should eventually converge on a final value
  - Updates are propagated infrequently in eventually consistent data stores
  - Acceptable and efficient if replicas in data store rarely exchange updates
  - e.g.: webpage updates, web search, crawling, updates to DNS server
  - Write-write conflicts are rare, can be handled by mutual exclusion
  - Read-write conflicts are more frequent, design has to focus on efficiently resolving such conflicts
- Client Consistency Guarantees - each client process should be guaranteed some level of consistency while accessing data value from different replicas
  - Monotonic Reads
    - Provides guarantees on successive reads
    - If a client process reads value of data item X, any successive read by that process should return same or more recent value of X
  - Monotonic Writes
    - Assures monotonic writes
    - Write operation on a data item X is completed before any successive write operation on X by same process
    - New write on a replica should wait for all old writes on any replica
  - Read Your Writes
    - Effect of a write on data X will always be seen by a successive read on X by same process
  - Write Follow Reads
    - Write on a data item X following a previous read on X by same process is guaranteed to take place on the same or a more recent value of X that was read

Type of Failure	Server Response
Crash failure	Halts, but works correctly until it halts
Omission failure	Fails to respond to incoming requests
Receive omission	Fails to receive incoming messages
Send omission	Fails to send messages
Timing failure	Lies outside specified time interval
Response failure	Incorrect response
Value failure	Value of response is wrong
State transaction failure	Deviates from correct flow of control
Arbitrary failure	May produce arbitrary responses at arbitrary times

Unit - 4

Design of DFS wrt HDFS

- HDFS is primary data storage system used by Hadoop apps
- Employs a NameNode and DataNode architecture to implement a DFS
- Closely coupled with MapReduce: a data processing framework that filters and divides work among nodes in a cluster; then organizes and condenses the results into a cohesive answer to a query
- Data is written on server once, and read and reused numerous times after that
- Primary NameNode keeps track of where file data is kept in cluster and manages file access: CRUD
- Multiple DataNodes (one in a cluster) generally organized within the same rack in a data center
- Data broken down into blocks and distributed among nodes
- Blocks are replicated across nodes
- NameNode always knows which DataNode contains which blocks and where DataNode resides within machine cluster
- If NameNode realises that one DataNode isn't working properly, it can reassign that DataNode's task to another containing same data block
- Supports a traditional hierarchical file organization
- NameNode records changes to file system namespace / properties.
- Features:
  - Data replication
  - Fault tolerance and reliability
  - High availability
  - Scalability
  - High throughput
  - Data locality

Design Goals of Sun NFS

- Machine and OS Independence
  - Protocols must be self-governing/independent of UNIX
  - NFS-server can source files to multiple diverse clients
  - Protocols must be modest enough to be implemented on low-end machines
- Crash Recovery
  - Clients and servers should be able to recover easily from machine crashes and network problems
- Reasonable Performance
  - People will not opt for a remote file system if it is not faster than the existing network utilities
- UNIX Semantics retained on UNIX client
  - For clear access to work on UNIX machines, UNIX file semantics have to be maintained or preserved for remote files
- Clear (Open) Access
  - Programs must not be able to articulate whether a file is local or remote

Name Services

- Name
  - refers to an entity (computer, service, remote object, file, user)
  - any character string; thus meaningful names used
- Address
  - Name that refers to an access point of an entity
  - An entity may have multiple access points and addresses
  - An entity may change its access points
- Identifiers
  - uniquely identify an entity
  - Properties:
    - refers to at most one entity
    - each entity referred to by at most one identifier
    - always refers to same entity; never reused
- Namespace
  - distributed system file names organized into namespaces
  - represented as a labelled directed graph with two types of nodes:
    - leaf node - represents a file, stores its information
    - directory node - represents a file directory, stores directory table of (edge label, node ID) pairs

DNS

- Naming service whose main naming database is used throughout the Internet via a logical network of servers
- Internet DNS name space segregated according to geography and organization
- Internet DNS basically used for simple host name resolution
- DNS client aka resolver; implemented as a library software
- Resolver receives queries, arranges into messages, forwards to DNS servers for satisfying those queries

Resource Management for QoS

- Resources should be arranged and scheduled (on a priority basis) properly to make them available to a particular decentralized app
- Resource scheduler responsible for scheduling resources
- Several real-time scheduling algorithms used
- Since both continuous and discrete data have to be handled, it is a bit of a challenge
- Scheduling categories - fair, real-time
- If multiple streams require same resource, it is necessary to consider the fairness of processes and avoid misuses of bandwidth through defective processes
- Technique given by Demers (bit-by-bit) offers more fairness wrt packet-by-packet tactic -> fair queuing
- Earliest-deadline first (EDF) scheduler uses deadline allotted to each task to regulate and determine subsequent item to be managed
- EDF necessitates one scheduling decision per message
- Rate-mnemonic (RM) scheduling is a protruding mechanism for real-time scheduling of periodic processes

BitTorrent

- Intended predominantly for downloading (not streaming) large data files
- P2P distributed file sharing protocol
- Largest file divided into fixed-sized data files made available throughout the P2P network
- Clients may download these files in parallel from different sites -> diminishes overall load on a solitary site
- .torrent file holds metadata:
  - name of data file
  - length of data file
  - location of a tracker
  - checksum value linked to every data chunk generated by SHA-1
- Trackers keep track on download status allied with a particular data file
- To comprehend information held by tracker, it is essential to deliberate lifecycle of a given file
- Any peer with a comprehensive version of a file is called a seeder
- Peer that creates the file delivers initial seed for file distribution
- Peers that want to download the file are called leechers
- As soon as leecher downloads all involved data chunks, it becomes a seeder for succeeding file downloads
- Tracker server preserves information regarding current state of downloads of a file in terms of related leechers and seeders
- When a peer wants to download a file, it contacts tracker and is given a limited view of the torrent in terms of collection of peers that might support the downloading

Unit - 5

Architecture of Web Services

- Similar to that of a traditional distributed system
- Typically structured as a traditional client-server system
- We can refer a document through a link known as URL (Uniform Resource Locator)
- URL identifies where the document is placed by using DNS which is linked with a specific server and a file name with which a server can search the document in a local file system
- Communication and coordination amongst web server and browser managed by HTTP
- WWW is most protruding fragment of Internet
- WWW introduced by Tim Berners-Lee in 1989 at CERN
- WWW offers a hypertext structure for data storage in documents that comprise of hyperlinks dedicated to reference other resources/documents stored on the web
- Components of web:
  - HTML
    - postulates layout and overall content of web page
    - used for specification of images and text contents of web pages
  - URLs
    - unique identification of resources stored on web
    - Uniform Resource Locators/Identifiers
    - consists of :
      - scheme - indicates type of URL
      - scheme specific identifier - appropriate identification of resource
  - HTTP
    - offers standard rules for proper communication and coordination among clients and servers with the help of client-server architecture
    - request-reply kind of protocol
    - Web1.0 - read only web
    - Web2.0 - read-write web
    - Web3.0 - semantic web
    - Web4.0 - symbiotic web / read-write-execution-concurrency

Principle of a Web Service

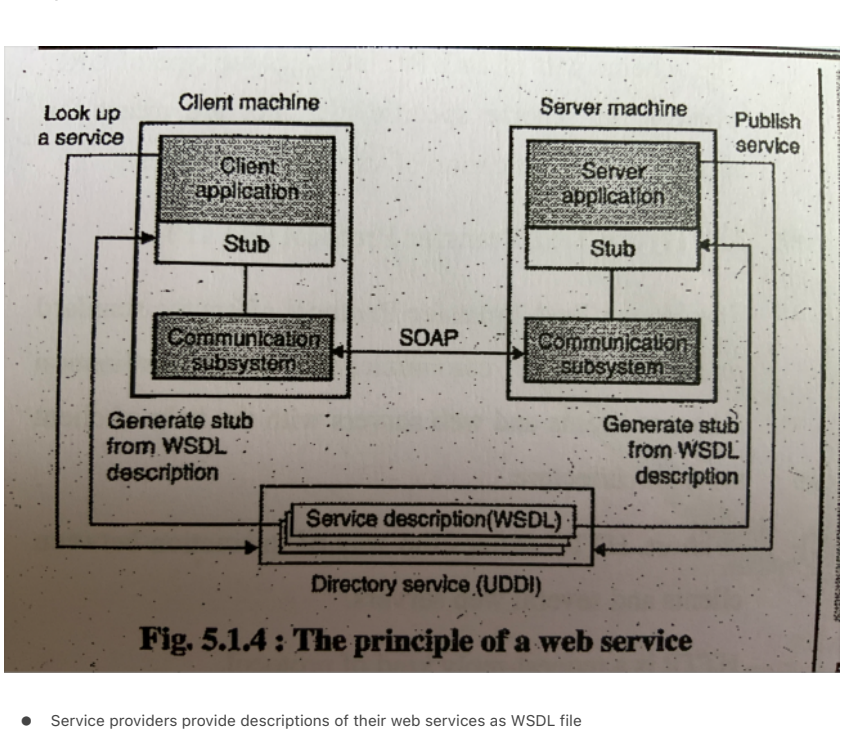
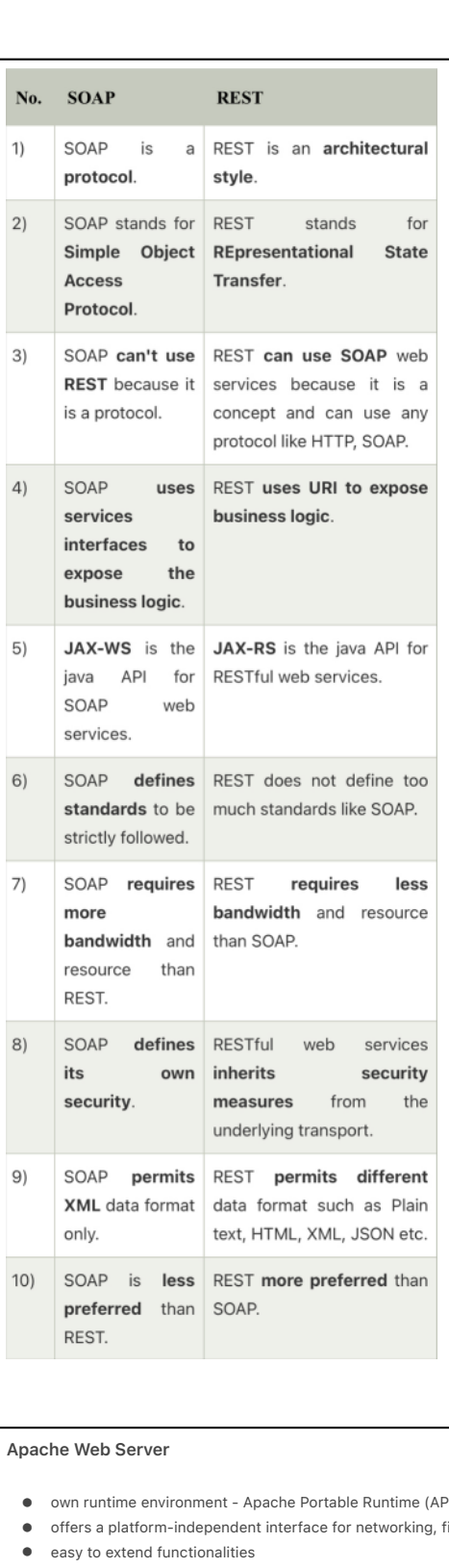


Fig. 5.1.4 : The principle of a web service

- Service providers provide descriptions of their web services as WSDL file
- WSDL file describes operations, data and exchange protocols offered by web service - machine readable information
- Consumers create a SOAP request to access the service's operations
- Service processes SOAP request and returns result as SOAP response
- In case of UDDI, UDDI acts as a service broker and permits consumers to find services
- The consumer searches UDDI and selects required service
- Then consumer is bound dynamically to service provider
- Finally, consumer generates a SOAP request (on the basis of WSDL file) to access methods offered by service
- Service then processes SOAP request and returns result as SOAP response

SOAP and REST



No.	SOAP	REST
1)	SOAP is a **protocol**.	REST is an **architectural style**.
2)	SOAP stands for **Simple Object Access Protocol**.	REST stands for **REpresentational State Transfer**.
3)	SOAP **can't use REST** because it is a protocol.	REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP.
4)	SOAP **uses services interfaces to expose the business logic**.	REST **uses URI to expose business logic**.
5)	**JAX-WS** is the java API for SOAP web services.	**JAX-RS** is the java API for RESTful web services.
6)	SOAP **defines standards** to be strictly followed.	REST does not define too much standards like SOAP.
7)	SOAP **requires more bandwidth** and resource than REST.	REST **requires less bandwidth** and resource than SOAP.
8)	SOAP **defines its own security**.	RESTful web services **inherits security measures** from the underlying transport.
9)	SOAP **permits XML** data format only.	REST **permits different** data format such as Plain text, HTML, XML, JSON etc.
10)	SOAP is **less preferred** than REST.	REST **more preferred** than SOAP.

Apache Web Server

- own runtime environment - Apache Portable Runtime (APR)
- offers a platform-independent interface for networking, file handling, etc.
- easy to extend functionalities
- All inward request queries follow TCP-based communication because web browsers use HTTP (which is executed on top of TCP)
- Hook - placeholder for a definite group of functions, is followed
- Apache core believes that inward queries are handled in a number of phases and each phase contains a few hooks
- Each hook signifies a group of similar actions to be executed as part of processing an inward request
- Hooks normally processed in a predetermined order
- Functions allied with a hook are offered by separate modules
- Modules are considered to be self-determining, so functions in same hook will be accomplished in a haphazard order

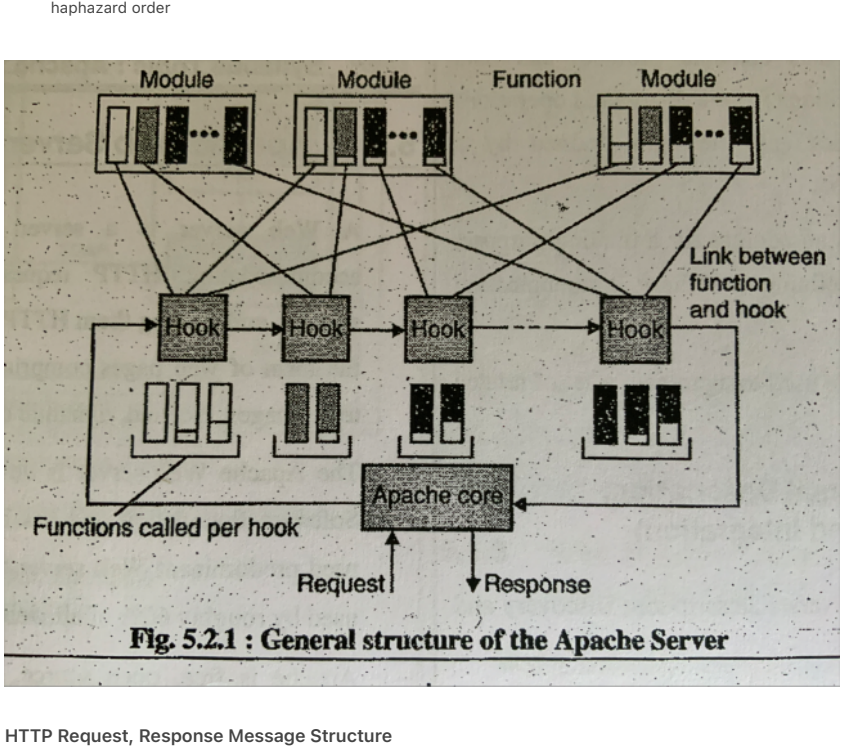


Fig. 5.2.1 : General structure of the Apache Server

HTTP Request, Response Message Structure

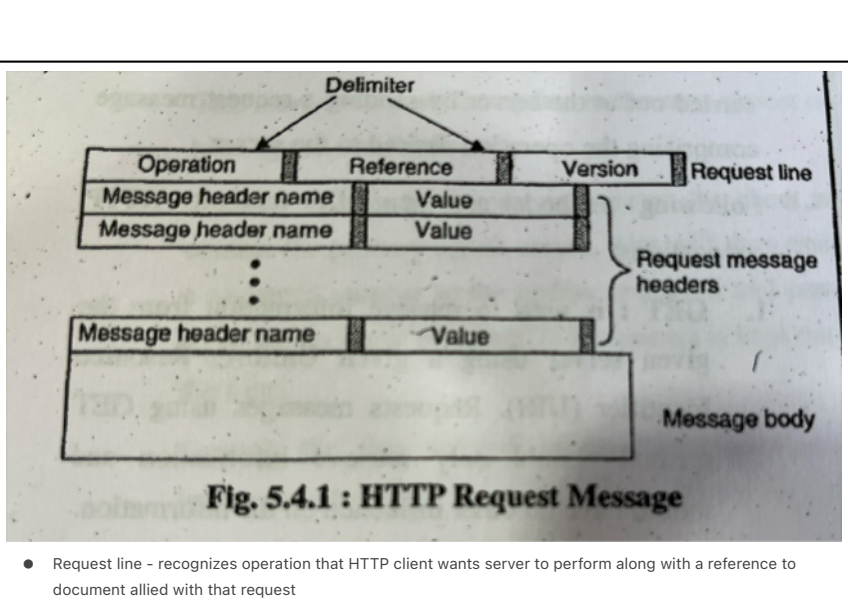


Fig. 5.4.1 : HTTP Request Message

- Request line - recognizes operation that HTTP client wants server to perform along with a reference to document allied with that request
- Header fields - offer essential information about request or response messages
- Types of headers:
  - General-header: general applicability for both request and response messages
  - Request-header: applicability only for request messages
  - Response-header: applicability only for response messages
  - Entity-header: define meta-information about entity-body; if body absent, defines resource identified by request
- Request-header fields let HTTP client pass additional information about request and HTTP client to server

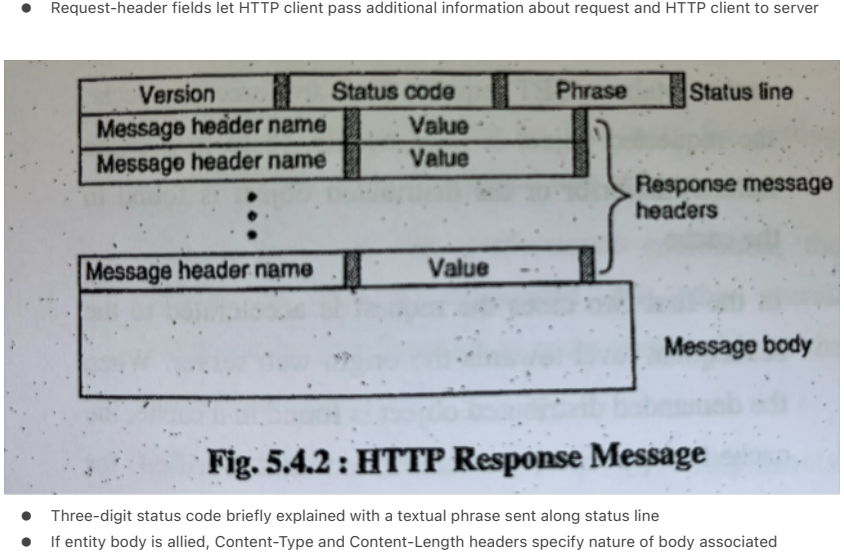


Fig. 5.4.2 : HTTP Response Message

- Three-digit status code briefly explained with a textual phrase sent along status line
- If entity body is allied, Content-Type and Content-Length headers specify nature of body associated
- Message body carries actual HTTP request and response data from server
- Response-header fields allow server to pass additional information about response that cannot be placed in status line

Hierarchical Caches and Cooperative Caching

- Drawback: can potentially incur a higher latency compared to nonhierarchical schemes
- This higher latency is caused by the need for client to check multiple caches rather than just one in the nonhierarchical scheme
- Solution: distributed/cooperative caching
- Whenever a cache miss occurs at a web proxy, it first checks a number of neighbouring proxies to see if they contain the requested document
- If check fails, proxy forwards the request to the Web server responsible for the document
- This scheme is primarily deployed with Web caches belonging to same organization that are colocated in same LAN

Unit - 6

Security Threats

- Modification
  - unauthorized access to data and damage to it
  - e.g.: hacking
- Interruption
  - data or distributed computing service becomes unavailable
  - e.g.: DoS attack
- Interception
  - unauthorized user gets access to data
  - e.g.: packet sniffing
- Fabrication
  - distributed activities/data are produced that would usually not be existent
  - e.g.: spoofing attack

Security Mechanisms

- Data integrity
- Digital signature
  - based on asymmetric cryptosystems and promises data integrity
- Encipherment
  - several security algorithms can be applied to data to make original data unreadable
- Authentication exchange
  - entity that communicates and coordinates data transfer activity within a DCS is authenticated by this security mechanism
- Notarization
  - trusted third party assures properties of data transfer
- Traffic padding
  - addition of data bits into gaps originated in normal data streams to diminish attacks by means of traffic analysis

Kerberos Authentication

- User has to provide username and password on client machine which cryptographically hashed to create a secret key for the client
- After verification done with authentication server, AS replies with:
  - TGS (Ticket Granting Service) session key Kt, encrypted using clients secret key Kc (now stored in authentication server)
  - TGT (Ticket Granting Ticket) encrypted using secret key of TGS. TGT includes client TGS session key Kt and its validity period
- Client now decrypts TGS session key Kt using his secret Kc. To request a service, client sends two messages to TGS:
  - TGT and name of service Sr that client wants to request
  - Authentication token which includes client ID and timestamp, encrypted using client TGS session key Kt
- Upon receiving all details from client TGS decrypts TGT using Kt, thus retrieving client TGS session key Kt and validity of TGT. If it is valid, TGS sends following messages to client:
  - New client server session key Ksc, encrypted using TGS session key Kt
  - Client to server ticket, encrypted using specific services key Ks, known to TGS only. (client to server ticket contains client ID, network address, validity period and client server session key Ksc)
- Upon retrieving all details from TGS, client decrypts client server session key Ksc and authenticates to service Sr by sending following messages:
  - Client server ticket sent by TGS in previous step
  - Client ID and timestamp encrypted using client server session key Ksc
- Service provider decrypts client to server ticket using secret key Ks and obtains client server session key Ksc
- Using client server session key Ksc, service provider decrypts client ID and timestamp information. To prove final identity service providers increment timestamp by 1 and sends it back to client
- Client decrypts and verifies response using client to server session key Ksc. Once this verification succeeds, client-server can start.

Firewall Configurations

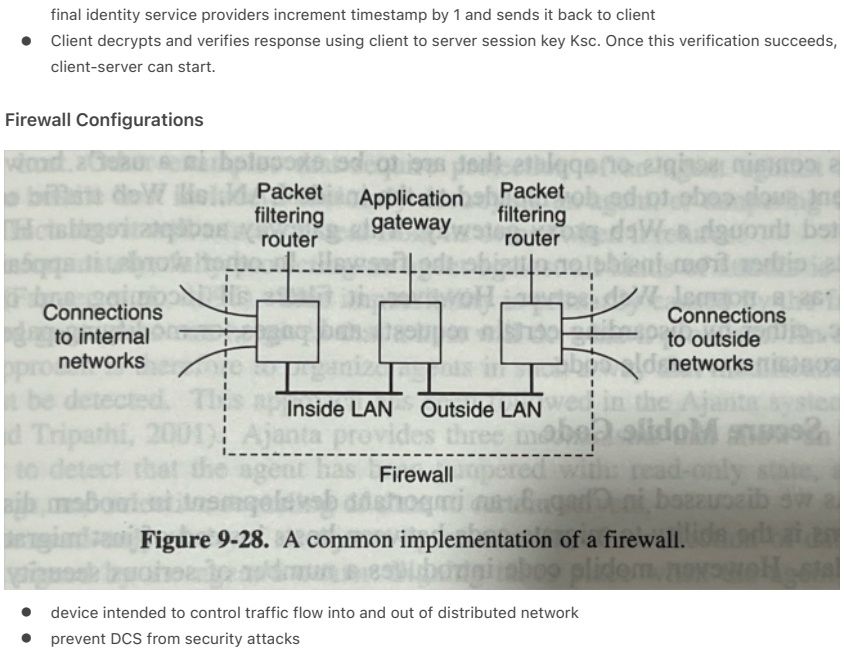


Figure 9-28. A common implementation of a firewall.

- device intended to control traffic flow into and out of distributed network
- prevent DCS from security attacks
- can be a hardware device or software program
- filters traffic flow on basis of TCP, UDP or IP information
- Configurations:
  - Packet-filtering gateway
    - operates as a router
    - decides whether or not to pass a network packet based on source and destination address contained in packet header
  - Application-level gateway
    - inspects content of an incoming/outgoing message
  - Proxy gateway
    - special kind of application-level gateway
    - works as a frontend to a specific application
    - ensures only those messages are passed that meet a certain criteria
    - accepts regular HTTP requests but filters all traffic; either by discarding certain requests and pages or modifying pages when they contain executable code

Secure Mobile Code

- security mechanism to balance communication and coordination within a DCS
- mobile codes should be protected in contradiction of malicious hosts
- when securing a mobile code from a malicious host, code mobility indicates that program will run under complete control of host
- To protect DCS from integrity violation, mobile code segments and data segments must be protected first
- Agent systems comprise of:
  - Runtime ENV for host protection (typically JVM)
  - Code signing to prove that agent has not been tampered with
  - Host authentication to prove that agent is about to move to intended host
  - Establishment of a secure cryptographic communication channel over which agent can migrate
- Host can be protected from a mobile code by simple verification of mobile code semantics and infrastructure that is gradually enhanced with security mechanisms

Key Management - Secret, Public Key Distribution

- Distribution of Public Keys:
  - Public announcement
    - Any user can send their key to another user or broadcast it to a group
    - Drawback: any user can pretend to be someone else; pretender can read all encrypted messages
  - Publicly available directory
    - maintenance and distribution controlled by a trusted entity
    - trusted entity maintains directory for each user as <name, public key>
    - each user has to register a public key with directory
    - user can replace existing key with new one anytime
    - drawback: hacker can obtain pvt key of directory or tamper with directory contents
  - Public key authority
    - P sends timestamped request to authority for public key of Q
    - A (authority) sends encrypted (with A's pvt key, so P can decrypt with A's public key) message to P containing Q's public key
    - P uses Q's public key to send encrypted message to Q
    - Q retrieves P's public key from A and decrypts message
    - Now P and Q start communication
    - Drawbacks: authority is tamperable, creates bottleneck
  - Public key certificates
    - P sends public key to certificate authority
    - Authority sends encrypted certificate using its private key
    - Certificate consists of timestamp, IDp, public key of P
    - Q retrieves certificate from authority
    - P publishes its certificate Cp to anyone who needs P's public key
    - Q also published its certificate Cq to anyone who needs Q's public key
    - P and Q begin communication
- Distribution of Secret Keys:
  - Simple Secret Key Distribution
  - Secret Key Distribution with Confidentiality and Authentication
    - P sends encrypted message using Q's public key, consists of IDp and nonce N1
    - Q sends encrypted message using P's public key, consists of P's nonce N1 and Q's nonce N2
    - P returns N2, encrypted using Q's public key to Q
    - P selects a secret key Ks
    - Q decrypts it to recover secret key
  - Hybrid Schemes
    - retains use of KDC (key distribution center) that shares a secret master key with each user and distributes secret session keys encrypted with master key
    - public key used to distribute master key
  - Key Generation, Distribution, Storage and Usage

Authorization Management

- Managing security is also concerned with managing access rights
- In non-DCS, when a new user is added to system, he/she is given initial rights and a complete account for a user is set up for one specific machine in which all rights have been specified in advance by sysadmins
- In a DCS, capabilities are used.
- Capability = unforgeable data structure for a specific resource, specifying exact access rights that holder of capability has wrt that resource
- Generalization of capabilities used in modern DCS is called attribute certificate; used to list certain (attribute, value) pairs that apply to an identified entity
- Attribute certificates handed out by attribute certificate authorities
- Delegation of access rights is done by passing certain access rights from one process to another
- Delegation avoids overhead as protection can be handled locally
- Proxy (in the context of security in computer systems) is a token that allows its owner to operate with same or restricted rights as the subject that granted the token

Digital Signature

- digital signature = digital key that is unique for individual message and party involved in communication
- represented as a string of binary digits
- computer uses an algorithm to identify the person signing the document
- defined as signature generated electronically from digital computer to ensure identity of sender and contents of message cannot be modifier during transmission
- achieves authenticity, integrity and non-repudiation of data over Internet
- sender uses signing (private) key to sign a message and send message and its digital signature to a receiver
- receiver uses a verification (public) key of sender to verify message origin and integrity
- digital signature = hash value of message encrypted with private key
- example of asymmetric key cryptography