UNIT-1

SMALL ANSWER QUESTIONS

1.**What is meant distributed system**?

1. We define a distributed system as a collection of autonomous computers linked by a network, with software designed to produce an integrated computing facility.

2. A system in which hardware or software components located at networked computers communicate and coordinate their actions only by message passing.

3. A collection of two or more independent computers which coordinate their processing through the exchange of synchronous or asynchronous message passing.

4. A collection of independent computers that appear to the users of the system as a single computers.

2. **What are the significance of distributed system?**
   a. Concurrency of computers.
   b. No global clock.
   c. Independent failures.
3. **Why we do you need distributed system?**

   a. **Functional distribution**: Computers have different functional capabilities (i.e., sharing of resources with specific functionalities).

   b. **Load distribution/balancing**: Assign tasks to processors such that the overall system performance is optimized.

   c. **Replication of processing power**: Independent processors working on the same task.

   d. Distributed system consisting of collections of microcomputers may have processing powers that no supercomputer will ever achieve.

   e. **Physical separation**: Systems that rely on the fact that computers are physically separated (e.g., to satisfy reliability requirements).

   f. **Economics**: Collections of microprocessors offer a better price/performance ratio than large mainframes.mainframes:10 times faster, 1000 times as expensive.

4**. Examples of distributed system?**

   a. Internet
   b. Intranet
   c. Mobile and ubiquitous computing.

5. **What is meant by location aware computing?**

Mobile computing is the performance of computing tasks while the users are on the move and away from their residence intranet but still provided with access to resources via the devices they carry with them. They can continue to access the intranet, they can continue to access resources in their home intranet, and there is increasing provision for users to utilize resources such as printers that are conveniently nearby as they move around. This is known as location aware computing.

6. **What are the two type of resource sharing?**

a. Hardware sharing: Printers. plotters and large disks and other peripherals are shared to reduce costs.

b. Data sharing is important in many applications:

1. Software developers working in a team need to access each other's code and share the same development tools.

2. Many commercial applications enable users to access shared data objects in a single active database.

3. The rapidly growing area of group-ware tools enables users to cooperate with in a network.

7. **List the importance of data sharing?**

- Software developers working in a team need to access each other's code and share the same development tools.
- Many commercial applications enable users to access shared data objects in a single active database.
- The rapidly growing area of group- ware tools enables users to cooperate with in a network.

8. Write the technological components of web?

- HTML
- HTTP-request-reply protocol
- URL's

9. List the distributed systems challenges?

a. Heterogeneity: standards and protocols; middleware; virtual machine;
b. Openness: publication of services; notification of interfaces;
c: Security: firewalls; encryption;
d. Scalability: replication; caching; multiple servers;
e. Failure Handling. failure tolerance; recover/roll-back; redundancy;
f. Concurrency. concurrency control to ensure data consistency.
g. Transparency. Middleware; location transparent naming; anonymity

10. What are the three components of security?

Security for information resources has three components:

- **Confidentiality:** production against disclosure to unauthorized individuals.
- **Integrity:** production against or corruption.
- **Availability:** production against interference with the means to access the resources.

11. What is the use of firewall?

A firewall can be used to form a barrier around an intranet to protect it from outside users but does not deal with ensuring the appropriate use of resources by users within the intranet.

12. What are the security challenges? List them.

a. **Denial of service attacks:** Another security problem is that the user may wish to disrupt a service for some reason. This can be achieved by bombarding the service with such a large number of pointless requests that the serious users are unable to use it. This is called a denial of service attack and there are many on well known web services.

b. **Security of mobile code:** Mobile codes needed to be handled with care. PC users sometimes send executable files as email attachments to be run by the recipient, but a recipient will not be able to run it.

13. List the challenges to be considered for designing scalable distributed system?

- Controlling the cost of physical resources
- Controlling the performance loss
- Preventing software resources running out
- Avoiding performance bottlenecks.

14. What are the types of transparencies?

a. **Access transparency:** enables local and remote resources to be accessed using identical operations.

b. **Location transparency:** enables resources to be accessed without knowledge of their location.

c. **Concurrency transparency:** enables several processes to operate concurrently using shared resources without interference between them.

d. **Replication transparency:** enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers. `

e. **Failure transparency:** enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.

f. **Mobility transparency:** allows the movement of resources and clients within a system without affecting the operation of users or programs.

g. **Performance transparency:** allows the system to be reconfigures to implement performance as loads vary.

h. **Scaling transparency:** allows the system and applications to expand in scale without change to the system structure or the application algorithms.

i. Access and location transparency together provide network transparency.

15. What are the failures detected in DS?

Masking failures: Some detected failures can be hidden or made less severe. Examples of hiding failures:

1. Messages can be retransmitted when they fail to arrive
2. File data can be written to a pair of disks that if one is corrupted, the other may still be correct.

Tolerating failures: Most of the services in the Internet do exhibit failures. It would not be practical for them to detect and hide all the failures occur in such network. Their clients are designed to tolerate failures, which generally involve the users in that.

Recovery from failures: involves the design of software so that the state permanent data can be rolled back after a server has crashed.

16. List the key design goals of DS?

a. Performance

b. Reliability
c. Scalability
d. Consistency
e. Security

17. List the technical design goals of DS?

a. Naming
b. Communication
c. Software structure

d. Workload allocation

e. Maintenance of consistency.

18 .What is the use of multicast?

a. **Locating an object:** A process multicasts a message containing a name of a resource to a group of server processes. Only the process that holds the resource responds to the message.

b. **Fault tolerance:** A process multicasts its request to a group of identical server processes. The group of servers can continue to provide their service even if one of its members fails.

c. **Multiple update:** Used for example in video conferencing with multiple participants.

19. Write the models used in workload allocation?

The following four models for workload allocation are presented here.

- Processor pool model
- Shared memory multiprocessors
- Parallel virtual machines
- Distributed shared memory.

20.What is meant by PVM?

PVM is an integrated set of software tools and libraries that emulates a general-purpose, flexible heterogeneous concurrent computing framework on interconnected computers of varied architecture. The overall objective of the PVM system is to enable a collection of computers to be used cooperatively for concurrent or parallel computation.

21.What is meant by DSM?

Distributed shared memory (DSM): DSM provides a global shared address space across the different machines on a cluster. The shared address space distinguishes it from packages such as PVM that provide a message passing interface between machines. There is a growing consensus in the parallel computing community that a shared memory interface is more desirable from the application programmer's viewpoint, allowing him to focus on algorithmic development rather than on managing communication.

22.List the types of consistencies in DS?

- Update consistency
- Replication consistency
- Cache consistency
- Failure consistency
- Clock consistency

23.List the user requirements used in design of DS?

- **Functionality:** what should the system do for the users.
- **Quality of service:** issue containing performance, reliability and security.
- **Reconfigurability:** the need to accommodate changes without causing disruption of the actual service.

24.List the main types of architectural model?

a. **Software architecture.**
b. **System architecture.**
    - Client server model
    - Services provided by multiple servers.
    - Proxy servers and cache.
    - Peer processes.

25. Enumerate the factors to be considered for variations in client server model?

The factors considered for several variations on the client server model.

- The use of multiple servers and caches to increase performance and flexibility.
- The use of mobile code and mobile agents.
- User's need for low cost components with limited hardware resources that are simple to manage.
- The requirements to add remove mobile devices in a convenient manner.

26.What is meant by thin clients?

- This refers to software layer that supports a window based user interface on a computer that is local to the user while executing application programs on a remote computer.
- It has the low management and hardware costs, but it runs the application code of user's computer in the computer server, which is the powerful computer that has the capacity to run large numbers of applications simultaneously. It can be multiprocessor or cluster computer running a multiprocessor version of OS such as UNIX or windows NT.

27.What is meant by x-11 window system?

a) The X-11 window system is a process that manages the display and interactive input devices (keyboard, mouse) of the computer on which it runs. It provides an extensive library of procedures (the X-11 protocol) for displaying and modifying graphical objects in windows as well as the creation and manipulation of windows.

b) The X-11 system is referred to as a window server process. The clients of the X-11 server are the application programs that the user is currently interacting with.

c) The client programs communicate with the server by invoking operation in the X-11 protocol, these include operations to draw text and graphical objects in wind

---

28.Enumerate the key features of spontaneous networking?

a. **Easy connection to a local network:** Wireless links avoid the need for pre-installed cabling and avoid the inconvenience and reliability issues surrounding plugs and sockets.

b. **Easy integration with local services:** Devices are able to find themselves inserted into existing networks of devices discover automatically what services are provided there, with no special configuration actions by the user.

29.List the design issues to be considered for spontaneous networking?

a. **Limited connectivity:** Users are not always connected as they move around. They are irregularly disconnected from wireless network as they travel through tunnels by train. They may also be totally disconnected for longer periods of time in regions where wireless connectivity ceases or it is too expensive to remain connected.

b. **Security and privacy:** Many vulnerable security issues arises due to the attempt of wireless connections in unsupervised way. Some systems track the physical locations of users as they move around and this may threaten the user's privacy. This facility enables users to access their home intranet while on the move may expose data that is supposed to remain behind the intranet firewall or it may open up the intranet to attacks from outside.

c. **Discovery services:** Spontaneous networking requires client processes running on portable devices and other appliances to access services on the networks to which they are connected. Here the clients discover what services are available in the network to which they are connected and to investigate their properties. The purpose of a discovery service is to accept and store details of services that are available on the network and to respond to queries from clients about them.

30.What is the purpose solved by fundamental model?

- In general, such a fundamental model should contain only the essential ingredients that we need to consider understanding and reasoning about some aspects of a system's behaviour. The purpose of such a model is:
- To make explicit all the relevant assumptions about the system we are modelling.
- To make generalization concerning what is possible or impossible, given those assumptions. The guarantees are our assumptions clear and explicit, we can hope to prove system properties using mathematical techniques. These properties will then hold for any system meeting our assumptions.

31.How the fundamental models are categorized?

a. Interaction

b. Failure

c. Security

32.List out the characteristics of performance of DS?

The following performance characteristics relating to latency, bandwidth and jitter.

a. **Latency:** The delay between the start of a message's transmission from one process and the beginning of its receipt by another is referred to as latency.

b. **Bandwidth:** The bandwidth of a computer network is the total amount of information that can be transmitted over it in a given time. When large number of Communication channels are using the same network, they have to share the available bandwidth.

c. **Jitter.** Jitter is the variation in the time taken to deliver a serious of messages. Jitter is relevant to multimedia data. For example, if consecutive samples of audio data are played with differing time intervals, the sound will be badly distorted.

33. What is synchronous DS?

1) The time to execute each step of a process has known lower and upper bounds.

2) Each message transmitted over a channel is received within a known bounded time.

3) Each process has a local clock whose drift rate from real time has a known bound.

4) It is possible to suggest likely upper and lower bounds for process execution time, message delay and clock drift rates in a distributed system, but it is difficult to arrive at realistic values and to provide guarantees of the chosen values.

5) In a synchronous system it is possible to use timeouts, for example to detect the failure of a process.

34.What is asynchronous DS?

1. Many distributed systems, such as the Intranet, qualify as asynchronous system.
2. An asynchronous distributed system is one in which there are no bounds on:

1. **Process execution speeds-**for example, one process step may take only a picoseconds and another a century; all that can be said is that each step may take an arbitrarily long time.

2. **Message transmission delays-**for example, one message from process A to process B may be delivered in negligible time and another may take several years. In other words, a message may be received after an arbitrarily long time.

3. **Clock drift rates-** again, the drift rate of a clock is arbitrary.

35. What is omission failure?

The faults classified as omission failures refer to cases when a process or communication channel fails to perform actions that it is supposed to do.

36.What is meant by arbitrary failure?

1) The term arbitrary or Byzantine failure is used to describe the worst possible failure semantics, in which any type of error may occur. For example, a process

may set wrong values in its data items, or it may return a wrong value in response to an invocation.

2) An arbitrary failure of a process is one in which in arbitrarily omits intended processing steps to takes unintended processing steps. Arbitrary failures in processes cannot be detected by seeing whether the process responds to invocations, because it might arbitrarily omit to reply.

37.List out the characteristics of networks hidden by stream abstraction?

a) **Message sizes:** The application can choose how much data it writes to a stream or reads from it. It may deal in very small or very large sets of data. The underlying implementation of a TCP stream decides how much data to collect before transmitting it as one or more IP packets.

b) **Lost messages:** The TCP protocol uses an acknowledgement scheme. As an example of a simple scheme (which is not used in TCP), the sending end keeps a record of each IP packet sent and receiving end acknowledges all the arrivals. If the sender does not receive an acknowledgement within a timeout, it retransmits the message.

c) **Flow control:** The TCP protocol attempts to match the speeds of the processes that read from and write to a stream. If the writer is too fast the reader, then it is blocked until the reader has consumed sufficient data.

d) **Message duplication and ordering:** Message identifiers are associated with each IP packet, which enables the recipient to detect and reject duplicates, or to reorder message that do not arrive in sender order.

e) **Message destinations:** A pair of communication processes establishes a connection before they can communicate over a stream. Once a connection is established, the processes simply read from and write to the stream without needing to use Internet addresses and ports. Establishing a connection involves a connect request from client to server followed by an accept request from server to client before any communication can take place.

38.List the issues related to stream communication?

a. **Matching of data items:** Two communicating processes need to agree as to the contents of the data transmitted over a stream. For example, if one process writes an int followed by a double to a stream, then the reader at the other end must read an int followed by a double. When a pair of processes does not cooperate correctly in their use of a stream, the reading process may experience errors when interpreting the data or may block due to insufficient data in the stream.

b. **Blocking:** The data written to a stream is kept in a queue at the destination socket. When a process attempts to read data from an input channel, it will get data from the queue or it will block until data becomes available. The process that writes data to a stream may be blocked by the TCP flow-control mechanism if the socket at the other end is queuing as much data as the protocol allows.

c. **Threads:** When a server accepts a connection, it generally creates a new thread in which to communicate with the new client. The advantage of using a separate

thread for each client is that the sever can block when waiting for input without delaying other clients.

d. **Failure model:** To satisfy the integrity to property of reliable communication, TCP streams use checksums to detect and reject corrupt packets and sequence numbers to detect and reject duplicate packets. For the deal with lost packets. Therefore, messages are guaranteed to be delivered ever when some of the underlying packets are lost.

39. What is marshalling and unmarshalling?

- An agreed standard for the representation of data structures and primitive values is called an external and data representation

- **Marshalling** is the process of taking a collection of data items and assembling them into a form suitable for transmission in a message.

- **Unmarshalling** is the process of disassembling them on arrival to produce an equivalent collection of data items at the destination. Thus marshalling consists of the translation of structured data items and primitives values into an external data representation. Similarly, unmarshalling consists of the generation of primitive values from their external data representation and the rebuilding of the data structures.

40. What are the approaches used in data representation?

a. **CORBA's common data representation,** which is concerned with an external representation for the structured and primitive types that can be passed as the arguments and results of remote method invocations in CORBA. It can be used by a variety of programming languages.

b. **Java's object serialization,** which is concerned with the flattering and external data representation of any single object or tree of objects that may need to be transmitted in a message or stored on a disk. It is for use only by java.

c. **XML (Extensible Markup Language),** which defines a textual format for representing structured data. It was originally intended for documents containing textual self-describing structured data-for example documents accessible on the Web- but it is now also used to represent the data sent in message exchanged by clients and servers in web services.

ESSAY QUESTIONS

1. Explain about the various examples of distributed systems?

Various examples of distributed systems are

- Internet
- Intranets
- Mobile networks

The Internet

- The Internet is a vast interconnected collection of computer networks of many different types.
- Multimedia services are available in the Internet enabling users to access audio and video data including music, radio, TV channels, phone, and video Conferencing.

General Examples of Distributed Systems

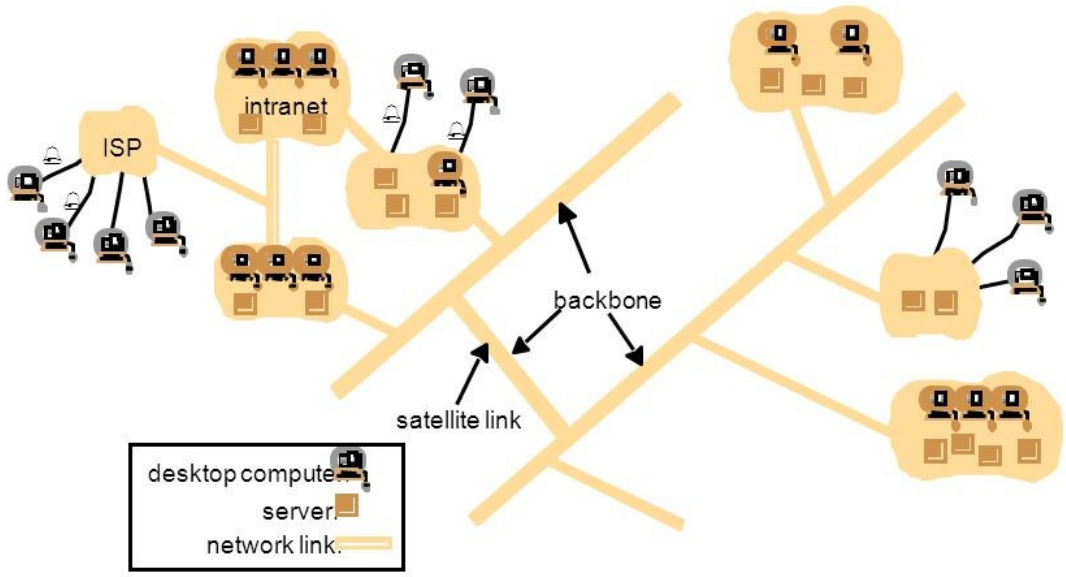


Figure 1. A typical portion of the Internet

- **Intranet**
  - An intranet is a portion of the Internet that is separately administered and has a boundary that can be configured to enforce local security policies.

- **Mobile networks**
  - Technological advances in device miniatorization and wireless networking have

led increasingly to the integration of small and portable computing devices into distributed systems.

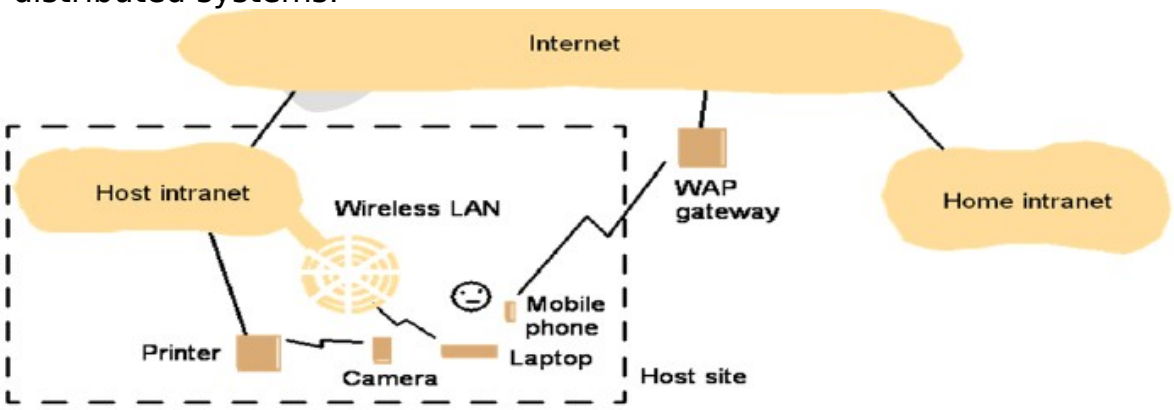


Figure 3. Portable and handheld devices in a distributed system.

These devices include:
Laptop computers
Handheld devices
Personal digital assistants(PDAs)
Mobile phones
Pagers
Video cameras
Digital cameras
Wearable devices
Smart watches with functionality similar to a PDA
Devices embedded in appliances
Washing machines
Hi-fi systems
Cars
Refrigerators

2. Explain about resource sharing?

Note that the users are so accustomed to the benefits of resource sharing that they may Easily overlook their significance. We routinely share hardware resources such as Printers, data resources such as files, and resources with more specific functionality such as search engines.

When we look at from the point of view of hardware provision, we share equipment such as printers and disks to reduce costs. But of far greater significance to users is the sharing of the higher-level resources that play a part in their applications and in their everyday Work and social activities. For example, users are concerned with sharing data in the form of a shared database or a set of web pages. Similarly, users think in terms of shared Resources such as a search engine or a currency converter, without regard for the server or Servers that provide these. We use the term *service* for a distinct part of a computer system that manages a collection of related resources and presents their functionality to users and applications. The only access we have to the service is via the set of operations that it exports.

The fact that services restrict resource access to a well-defined set of operations is in part standard software engineering practice. For effective sharing, each resource must be managed by a program that offers a communication interface enabling the resource to be accessed and updated reliably and consistently.

The term *server* refers to a running program (a *process*) on a networked computer that accepts requests from programs running on other computers to perform a service and responds appropriately. The requesting processes are referred to as *clients,* and the overall approach is known as *client-server computing*. In this approach, requests are sent in messages from clients to a server and replies are sent in messages from the server to the clients. A complete interaction between a client and a server, from the point when the client sends its request to when it receives the server‘s response, is called a *remote invocation*.

Many, but certainly not all, distributed systems can be constructed entirely in the form of interacting clients and servers. The World Wide Web, email and networked printers all fit this model.

There are two types of resources: they are hardware resources (e.g., disks and printers) and software resources (e.g., files, windows, and data objects).
Hardware sharing is used for convenience and reduction of cost. Where as Data sharing (shared usage of information) is used for consistency (compilers and libraries), exchange of information (database), and cooperative work (groupware).
Service resources are used as search engines, computer-supported cooperative working and so on.

Resource Manager: Software module that manages a set of resources. Each resource requires its own management policies and methods.

Client server model: server processes act as resource managers for a set of resources anda set of clients.
Object based model: resources are objects that can move. Object manager is movable. Request for a task on an object is sent to the current manager. Manager must be colocated with object.

3. Explain about various levels of transperancy?

Transparency is defined as the hiding of the separation of components in distributed
systems from the user and the application programmer.
With transparency the system is perceived as a whole rather than a collection of
independent components.
Forms of transparencies:

Access transparency
Enables local and remote resources to be accessed using identical operations.

Location transparency
Enables resources to be accessed without knowledge of their physical or network location (for example, which building or IP address).

Concurrency transparency
Enables several processes to operate concurrently using shared resources without interference between them.

Replication transparency
Enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers.

Failure transparency
Enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.

Mobility transparency
Allows the movement of resources and clients within a system without affecting the operation of users or programs.

Performance transparency
Allows the system to be reconfigured to improve performance as loads vary.

Scaling transparency
Allows the system and applications to expand in scale without change to the system structure or the application algorithms.

The two most important transparencies are access and location transparency referred
to together as network transparency.
Presence or absence of network transparency most strongly affects the utilization of
distributed resources.

4. Explain about fundamental models?

Fundamental Models are concerned with a more formal description of the properties that are common in all of the architectural models.

All architectural models are composed of processes that communicate with each
other by sending messages over a computer networks.

Aspects of distributed systems that are discussed in fundamental models are:

Interaction model
. Computation occurs within processes.
. The processes interact by passing messages, resulting in:Communication (information flow)
Coordination (synchronization and ordering of activities)between processes
Interaction model reflects the facts that communication takes
place with delays.

Failure model

- Failure model defines and classifies the faults.

Security model

- Security model defines and classifies the forms of attacks.
  It provides a basis for analysis of threats to a system .It is used to design of systems that are able to resist threats.

5. **Expalin about the cliet server and Peer- to- peer architecture?**

Client-server: This is the architecture that is most often cited when distributed systems are discussed. It is historically the most important and remains the most widely employed. The Figure 1.2.1 illustrates the simple structure in which processes take on the roles of being clients or servers. In particular, client processes interact with individual server processes in potentially separate host computers in order to access the shared resources that they manage.

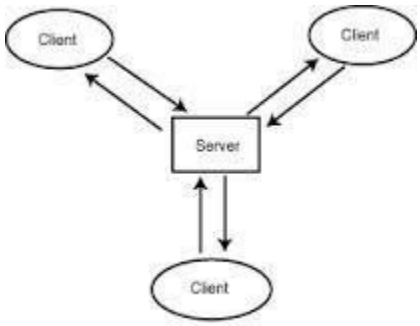


Figure 1 :Client Server Model

Peer-to-peer: In this architecture all of the processes involved in a task or activity play similar roles, interacting cooperatively as peers without any distinction between client and server processes or the computers on which they run. In practical terms, all participating processes run the same program and offer the same set of interfaces to each other. While the client-server model offers a direct and relatively simple approach to the sharing of data and other resources, it scales poorly. The centralization of service provision and management implied by placing a service at a single address does not scale well beyond the capacity of the computer that hosts the service and the bandwidth of its network connections. The Figure 1.2.2 shows Peer-to-Peer model.

MUTIPLE CHOICE QUESTIONS

1.In distributed system each processor has its own
a) local memory
b) clock
c) both (a) and (b)
d) none of the mentioned

Answer:c

2. If one site fails in distributed system
a) the remaining sites can continue operating
b) all the sites will stop working
c) directly connected sites will stop working
d) none of the mentioned

Answer:a

3. Network operating system runs on
a) server
b) every system in the network
c) both (a) and (b)
d) none of the mentioned

Answer:a

4. Which technique is based on compile-time program transformation for accessing remote data in a distributed-memory parallel system.
a) cache coherence scheme
b) computation migration
c) remote procedure call
d) message passing

Answer:b

5. Logical extension of computation migration is
a) process migration
b) system migration

c) thread migration
d) data migration

Answer:a

6. Processes on the remote systems are identified by
a) host ID
b) host name and identifier
c) identifier
d) process ID

Answer:b

7. Which routing technique is used in distributed system?
a) fixed routing
b) virtual routing
c) dynamic routing
d) all of the mentioned

Answer:d

8. In distributed systems, link and site failure is detected by
a) polling
b) handshaking
c) token passing
d) none of the mentioned

Answer:b

9. The capability of a system to adapt the increased service load is called
a) scalability
b) tolerance
c) capacity
d) none of the mentioned

Answer:a

10. Internet provides _____ for remote login.
a) telnet
b) http
c) ftp
d) RPC

Answer:a

11. The file once created can not be changed is called
a) immutable file
b) mutex file
c) mutable file
d) none of the mentioned

Answer:a

12. _____ of the distributed file system are dispersed among various machines of distributed system.
a) Clients
b) Servers
c) Storage devices
d) all of the mentioned

Answer:d

13. _____ is not possible in distributed file system.
a) File replication
b) Migration
c) Client interface
d) Remote access

Answer:b

14. Which one of the following hides the location where in the network the file is stored?
a) transparent distributed file system
b) hidden distributed file system
c) escaped distribution file system
d) spy distributed file system

Answer:a

15. In distributed file system, when a file's physical storage location changes
a) file name need to be changed
b) file name need not to be changed
c) file's host name need to be changed
d) file's local name need to be changed

Answer:b

16. In distributed file system, _____ is mapping between logical and physical objects.
a) client interfacing
b) naming
c) migration
d) heterogeneity

Answer:b

17. In distributed file system, a file is uniquely identified by
a) host name
b) local name
c) the combination of host name and local name
d) none of the mentioned

Answer:c

18. There is no need to establish and terminate a connection through open and close operation in
a) stateless file service
b) stateful file service
c) both (a) and (b)
d) none of the mentioned

Answer:a

19. In distributed file system, file name does not reveal the file's
a) local name
b) physical storage location
c) both (a) and (b)
d) none of the mentioned

Answer:b

20. Which one of the following is a distributed file system?
a) andrew file system
b) network file system
c) novel network
d) all of the mentioned

Answer:d

21.In distributed systems, a logical clock is associated with
a) each instruction
b) each process
c) each register
d) none of the mentioned

Answer:b

22. If timestamps of two events are same, then the events are
a) concurrent
b) non-concurrent
c) monotonic
d) non-monotonic

Answer:a

23. If a process is executing in its critical section
a) any other process can also execute in its critical section
b) no other process can execute in its critical section
c) one more process can execute in its critical section
d) none of the mentioned

Answer:b

24. A process can enter into its critical section
a) anytime

b) when it recieves a reply message from its parent process
c) when it recieves a reply message from all other processes in the system
d) none of the mentioned

Answer:c

25. For proper synchronization in distributed systems
a) prevention from the deadlock is must
b) prevention from the starvation is must
c) both (a) and (b)
d) none of the mentioned

Answer:c

26. In the token passing approach of distributed systems, processes are organized in a ring structure
a) logically
b) physically
c) both (a) and (b)
d) none of the mentioned

Answer:a

27. In distributed systems, transaction coordinator
a) starts the execution of transaction
b) breaks the transaction into number of sub transactions
c) coordinates the termination of the transaction
d) all of the mentioned

Answer:d

28. In case of failure, a new transaction coordinator can be elected by
a) bully algorithm
b) ring algorithm
c) both (a) and (b)
d) none of the mentioned

Answer:c

29. In distributed systems, election algorithms assumes that
a) a unique priority number is associated with each active process in system
b) there is no priority number associated with any process
c) priority of the processes is not required
d) none of the mentioned

Answer:a

30. According to the ring algorithm, links between processes are
a) bidirectional
b) unidirectional

c) both (a) and (b)
d) none of the mentioned

Answer:b

31) What is not true about distributed system ?
a) It is a collection of processor
b) All processors are synchronized
c) They do not share memory
d) None of these

Answer : b

32) What are characteristics of processor in distributed system ?
a) They vary in size and function
b) They are same in size and function
c) They are manufactured with single purpose
d) They are real-time devices

Answer : a

33) What are characteristics of distributed file system ? (Choose two)
a) Its users,servers and storage devices are dispersed
b) Service activity is carried out across the network
c) They have single centralized data repository
d) There are multiple dependent storage devices

Answer : a & b

34) What is not a major reason for building distributed systems ?
a) Resource sharing
b) Computation speedup
c) Reliability
d) Simplicity

Answer : d

35) What are two types of distributed operating system ? (Choose two)
a) Network Operating system
b) Zone based Operating system
c) Level based Operating system
d) Distributed Operating system

Answer : a & d

36) What are characteristic of Network Operating Systems ? (Choose two)
a) Users are aware of multiplicity of machines
b) They are transparent
c) They are simple to use
d) They are not transparent

Answer : a & d

37) How are access to resources of various machines is done ? (Choose two)
a) Remote logging using ssh or telnet
b) Remote Desktop
c) FTP is not used
d) Zone are configured for automatic access

Answer : a & b

38) What are characteristics of Distributed Operating system ? (Choose two)
a) Users are not aware of multiplicity of machines
b) Access is done like local resources
c) Users are aware of multiplicity of machines
d) They have multiple zones to access files

Answer : a & b

39) What are characteristics of data migration ?
a) transfer data by entire file or immediate portion required
b) transfer the computation rather than the data
c) execute an entire process or parts of it at different sites
d) None of these

Answer : a

40) What are characteristics of computation migration ?
a) transfer data by entire file or immediate portion required
b) transfer the computation rather than the data
c) execute an entire process or parts of it at different sites
d) None of these

Answer : b

UNIT-2

SMALL ANSWER QUESTIONS

2 What is meant by hardware and software clock?

Clock devices can be programmed to generate interrupts at regular intervals in orders that, for example, time slicing can be implemented.The operating system reads the node's

hardware clock value, H(t) , scales it and adds an offset so as to produce software clock C

(t)=$\alpha H_i(t)+\beta$ that approximately measures real ,physical time t for process pi.

3 What is clock resolution?

Note that successive events will correspond to different timestamps only if the clock resolution-the period between updates of the clock-value-is smaller than the time interval betw4een successive events. The rate at which events occur depends on such factors as the length of the processor instruction cycle.

4 What is clock drift?

Clock drift, which means that they count time at different rates and so diverge. The underling oscillators are subject to physical variations, with the consequence that their frequencies of oscillation differ. Moreover ,even the same clock's frequency varies with temperature. Designs exist that attempt to compensate for this variation, but they cannot eliminate it. A clock's drift rate is the change in the offset (difference in reading) between the clock and a nominal perfect reference clock per unit of time measured by the reference clock.

5 What is IAT?

Computer clocks can be synchronized to external sources of highly accurate time. The most accurate physical clocks use atomic oscillators, whose drift rate is about one part in 1013.the most accurate physical clocks use atomic oscillators, whose drift rate is about

one part in 1013. The output of these atomic clocks is used as the standard for elapsed real a time, known as International Atomic Time.

6 What is CUT?

Coordinated Universal Time-abbreviated as UTC (From the French equivalent)-is as international standard for timekeeping. It is based on atomic time, but a so-called 'leap second' is inserted-or, more rarely, deleted-occasionally to keep it in step with astronomical time.

7. What is meant by external synchronization?

In order to know at what time of day events occur at the processes in our distributed system –it is necessary to synchronize the processes' clocks, C , with an authoritative, external source of time. This is external synchronization.

8. What is internal synchronization?

And if the clocks C are synchronized with one another to known degree of accuracy, then we can measure the interval between two events occurring at different computers by appealing to their local clocks, even though they are not necessarily synchronized to an external source of time. This is internal synchronization.

a. For a synchronization bound $D > 0$ and for a source S of UTC times, $|S(t)-C_i(t)| < D$, for all real times t in I.

b. Clocks C agree with in the bound D.

9. Define NTP and its design aims.

Cristian's method and the Berkeley algorithm are intended primarily for use within intranets. The Network Time Protocol (NTP) [Mills1995] defines an architecture for a time service and a protocol to distribute time information over the Internet.

NTP's chief design aims and features are as follows:

a. To provide a service enabling clients across the Internet to be synchronized accurately to UTC:

b. To provide a reliable service that can survive lengthy losses of connectivity:

c. To enable clients to resynchronize sufficiently frequently to offset the rates of drift found in most computers: To provide protection against interference with the time service, whatever malicious or accidental.

10. What is strata?

The NTP service is provided by a network of servers located across the Internet. Primary servers are connected directly to a time source such as a radio clock receiving UTC; secondary servers are synchronized, ultimately, with primary servers. The servers are connected in a logical hierarchy called a synchronization subnet whose levels are called strata.

11. Enumerate the mode of synchronization in NTP servers.

a. NTP servers synchronize with one another in one of three: multicast, procedure-call and symmetric mode

b. ***Multicast mode*** is intended for use on a high-speed LAN. One or more servers periodically multicasts the time to the servers running in other computers connected by the LAN, Which set their clocks assuming a small delay. This mode can achieve only relatively low accuracies, but ones that nonetheless are considered sufficient for many purposes.

c. ***Procedure-call mode*** is similar to the operation of Cristian's algorithm. In this mode, one server accepts requests from other computers, which it processes by replying with its timestamp (current clock reading). This mode is suitable when higher accuracies are required than can be achieved with multicast, or where multicast is not supported in hardware.

d. ***In, symmetric mode*** is intended for use by the servers that supply time information in LANs and by the higher levels of the synchronization subnet, where the highest accuracies are to be achieved.

12. What is filter dispersion?

NTP servers apply a data filtering algorithm to successive pairs which estimates the offset o and calculates the quality of this estimates as a statistical quantity called the filter dispersion.

13. What is synchronization dispersion?

Peers with lower stratum numbers are more favoured than those in higher strata because they are 'closer' to the primary time sources. Also, those with the lowest synchronization dispersion are relatively favoured. This is the sum of the filter dispersions measured between the server and the root of the synchronization subnet.

14. What is meant by HB relation?

a. Lamport called the partial ordering obtained by generalizing these two relationships the happened-before relation. It is also sometimes known as the relation of causal ordering or potential causal ordering.

b. We can define the happened-before relation, denoted -> by as follow:

HB1: If process p_i:e->$_i$e' , then e->e'

HB2: For any message m, send (m) receive (m)

HB3: IF e, e' and e" are events such that e->e' then e->e"

15. What is logical clock?

α. Lamport [1978] invented a simple mechanism by which the happened before ordering can be captured numerically, called a logical clock.

β. A Lamport logical clock is a monotonically increasing software counter, whose value need bear no particular relationship to any physical clock.

χ. Each process p keeps its own logical clock, L , which it uses to apply so-called Lamport timestamps to a events.

δ. We denote the timestamp of event e at pi by L (e) , and by L(e) we denote the timestamp of event e at whatever process it occurred at.

16. Define Vector clock

Vector clocks for a system of N processes is an array of N integers

• Shortcoming of Lamport clocks:

$L(e) < L(e')$ doesn't imply $e \rightarrow e'$

6. Vector clock: an array of N integers for a system of N processes

   – Each process keeps its own vector clock *Vi* to timestamp local events

   – Piggyback vector timestamps on messages

7. Rules for updating vector clocks:

   – *Vi*[*i*]] is the number of events that *pi* has timestamped

   – *Vij**i*] ($j \neq i$) is the number of events at *pj* that *pi* has been affected by

VC1: Initially, *Vi*[*j*] := 0 for *pi*, *j*=1.. *N* (*N* processes)

VC2: before *pi* timestamps an event, *Vi*[*i*] := *Vi*[*i*]+1

VC3: *pi* piggybacks *t* = *Vi* on every message it sends

C4: when *pi* receives a timestamp *t*, it sets *Vi*[*j*] := max(*Vi*[*j*] , *t*[*j*]) for *j*=1..*N* (merge

---

operation)

17. What do you meant by distributed garbage

An object is considered to be garbage if there are no longer any reference to it anywhere in the distributed system. The memory taken up by that object can be reclaimed once it is known as to be garbage.

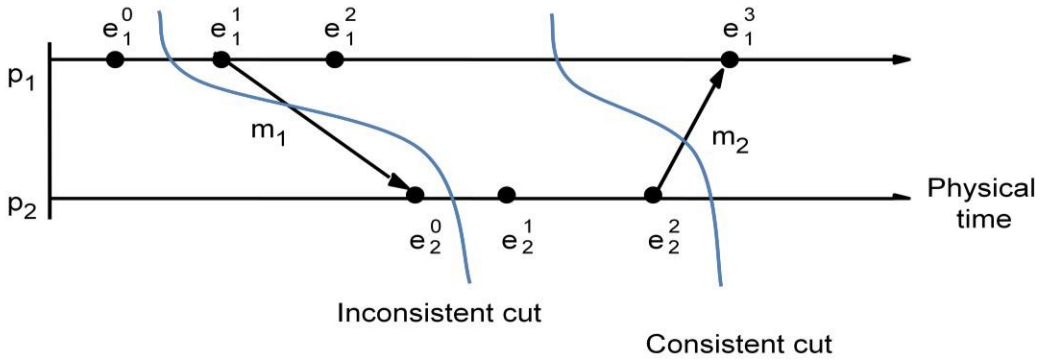
18. Define Global History

Let us return to our general system p of N processes p_i(i=1,2,3,…..N)

Here a series of events occurs at each process, and that we may characterize the execution of each process by its history

19. What is meant by cut?

Consider the events occurring at processes p_1 and p_2 shown in figure



19. Define Global state predicate

10. A Global State Predicate is a function that maps from the set of global process states to True or False.

11. Detecting a condition like deadlock or termination requires evaluating a Global State Predicate.

12. A Global State Predicate is stable: once a system enters a state where it is true, such as deadlock or termination, it remains true in all future states reachable from that state.

13. However, when we monitor or debug an application, we are interested in non stable predicates.

20. List the assumption considered in snapshot algorithm

- Neither channels nor processes fail

- Reliable communications ensure every message sent is received exactly once

- Channels are unidirectional
- Messages are received in FIFO order
- There is a path between any two processes
- Any process may initiate a global snapshot at any time
- Processes may continue to function normally during a snapshot

21. Define Failure detector.

A failure detector is a service that processes queries about whether a particular process has failed .It is often implemented by an object local to each process that runs failure detection algorithms in conjunction with its counterparts at the other processes.

22. List the properties of failure detector

A failure detector is not necessarily accurate. Most fails into the category of ***unreliable failure detectors.***

α. A result of unsuspected

β. A result of Suspected

23. Define critical section problem

The application – level protocol for executing a critical section is as follows

- enter() - enter critical section – block if necessary
- resurceAccesses() - access shared resources in critical section
- exit() - leave critical section other processes may now enter.

24. What is meant by election

Election: choosing a unique process for a particular role is called an election

– All the processes agree on the ***unique*** choice

– For example, server in dist. Mutex

25. List the famous mutual exclusion algorithms

α. Center server algorithm

β. Ring- Based algorithms

χ. Mutual Exclusion using multicast and Logical Clocks

- Maekawa's Voting algorithms
- Mutual Exclusion algorithms comparison

26. What do you meant by bully algorithms and types of messages

Assumption: Each process knows which processes have higher identifiers, and that it can communicate with all such processes

•Compare with ring-based election

– Processes can crash and be detected by timeouts

14. synchronous

15. timeout $T = 2T_{transmitting}$ (max transmission delay) + $T_{processing}$ (max processing delay)

Three types of messages

- Election: announce an election
- Answer: in response to Election
- Coordinator: announce the identity of the elected process

27. What are the types of ordering in multicast

Three types of message ordering

FIFO (First-in, first-out) ordering: if a correct process delivers a message before another, every correct process will deliver the first message before the other

Casual ordering: any correct process that delivers the second message will deliver the previous message first

Total ordering: if a correct process delivers a message before another, any other correct process that delivers the second message will deliver the first message first

Define Consensus

Consensus more precisely and relates it to three related Problems of agreement.For processes to agree on a value (consensus) after one or more of the processes has proposed what that value should be Covered topics: ***byzantine generals, interactive consistency, totally ordered multicast***

16. The byzantine generals problem: a decision whether multiple armies should attack or retreat, assuming that united action will be more successful than some attacking and some retreating
17. Another example might be space ship controllers deciding whether to proceed or abort. Failure handling during consensus is a key concern

29. What are the requirements of interactive consistency?

Three requirements of a consensus algorithm

3. ***Termination***: Eventually every correct process sets its decision variable
4. ***Agreement***: The decision value of all correct processes is the same: if pi and pj are correct and have entered the *decided* state, then di=dj for all (i,j=1,2, …, N)

5. ***Integrity***: If the correct processes all proposed the same value, then any correct process in the *decided* state has chosen that value

ESSAY QUESTIONS

MULTIPLE CHOICE QUESTIONS

1)The systems which allows only one process execution at a time, are called
a) uniprogramming systems
b) uniprocessing systems
c) unitasking systems
d) none of the mentioned

Ans A

2. In operating system, each process has its own
a) address space and global variables
b) open files
c) pending alarms, signals and signal handlers
d) all of the mentioned

Ans D

3. In Unix, Which system call creates the new process?
a) fork
b) create
c) new
d) none of the mentioned

Ans A

4) 4. A process can be terminated due to
a) normal exit
b) fatal error
c) killed by another process
d) all of the mentioned

Ans D

5). A process stack does not contain
a) function parameters
b) local variables
c) return addresses
d) PID of child process

Ans D

6) The address of the next instruction to be executed by the current process is provided by the
a) CPU registers
b) program counter
c) process stack
d) pipe

Ans B

7) Which system call returns the process identifier of a terminated child?
a) wait
b) exit
c) fork
d) get

Ans A

8) Which process can be affected by other processes executing in the system?
a) cooperating process
b) child process
c) parent process
d) init process

Ans A

9) If a process is executing in its critical section, then no other processes can be executing in their critical section. This condition is called
a) mutual exclusion
b) critical exclusion
c) synchronous exclusion
d) asynchronous exclusion

Ans A

10) Which one of the following is a synchronization tool?
a) thread
b) pipe
c) semaphore
d) socket

Ans C

11) Mutual exclusion can be provided by the
a) mutex locks
b) binary semaphores
c) both (a) and (b)
d) none of the mentioned

Ans C

13) Process synchronization can be done on
a) hardware level
b) software level
c) both (a) and (b)
d) none of the mentioned

Ans C

12) Inter process communication :
a) allows processes to communicate and synchronize their actions when using the same address space.
b) allows processes to communicate and synchronize their actions without using the same address space.
c) allows the processes to only synchronize their actions without communication.
d) None of these

Ans B

13) Message passing system allows processes to :
a) communicate with one another without resorting to shared data.
b) communicate with one another by resorting to shared data.
c) share data
d) name the recipient or sender of the message

Ans A

14) Messages sent by a process :
a) have to be of a fixed size
b) have to be a variable size
c) can be fixed or variable sized
d) None of these

Ans C

15) The link between two processes P and Q to send and receive messages is called :
a) communication link
b) message-passing link
c) synchronization link
d) All of these

Ans A

16)Which of the following are TRUE for direct communication :(choose two)
a) A communication link can be associated with N number of process(N = max. number of processes supported by system)
b) A communication link can be associated with exactly two processes
c) Exactly N/2 links exist between each pair of processes(N = max. number of processes supported by system)
d) Exactly one link exists between each pair of processes

Ans Band D

17) 7) In indirect communication between processes P and Q :
a) there is another process R to handle and pass on the messages between P and Q
b) there is another machine between the two processes to help communication
c) there is a mailbox to help communication between P and Q
d) None of these

Ans C

18) In the non blocking send :
a) the sending process keeps sending until the message is received
b) the sending process sends the message and resumes operation
c) the sending process keeps sending until it receives a message
d) None of these

Ans B

19) Bounded capacity and Unbounded capacity queues are referred to as :
a) Programmed buffering
b) Automatic buffering
c) User defined buffering
d) No buffering

Ans B

20) The Zero Capacity queue :
a) is referred to as a message system with buffering

b) is referred to as a message system with no buffering
c) is referred to as a link
d) None of these

Ans B

21. In distributed systems, a logical clock is associated with
a) each instruction
b) each process
c) each register
d) none of the mentioned

Answer:b

22. If timestamps of two events are same, then the events are
a) concurrent
b) non-concurrent
c) monotonic
d) non-monotonic

Answer:a

23. If a process is executing in its critical section
a) any other process can also execute in its critical section
b) no other process can execute in its critical section
c) one more process can execute in its critical section
d) none of the mentioned

Answer:b

24. A process can enter into its critical section
a) anytime
b) when it recieves a reply message from its parent process
c) when it recieves a reply message from all other processes in the system
d) none of the mentioned

Answer:c

25. For proper synchronization in distributed systems
a) prevention from the deadlock is must
b) prevention from the starvation is must
c) both (a) and (b)
d) none of the mentioned

Answer:c

26. In the token passing approach of distributed systems, processes are organized in a ring structure
a) logically
b) physically

c) both (a) and (b)
d) none of the mentioned

Answer:a

27. In distributed systems, transaction coordinator
a) starts the execution of transaction
b) breaks the transaction into number of sub transactions
c) coordinates the termination of the transaction
d) all of the mentioned

Answer:d

28. In case of failure, a new transaction coordinator can be elected by
a) bully algorithm
b) ring algorithm
c) both (a) and (b)
d) none of the mentioned

Answer:c

29. In distributed systems, election algorithms assumes that
a) a unique priority number is associated with each active process in system
b) there is no priority number associated with any process
c) priority of the processes is not required
d) none of the mentioned

Answer:a

30. According to the ring algorithm, links between processes are
a) bidirectional
b) unidirectional
c) both (a) and (b)
d) none of the mentioned

Answer:b

31) What are the characteristics of tightly coupled system ? (Choose three)
a) Same clock, usually shared memory
b) Communication is via this shared memory
c) Multiprocessors
d) Different clock

Answer : a,b & c

32) What are the characteristics of tightly coupled system ? (Choose three)
a) Different clock
b) Use communication links
c) Same clock
d) Distributed systems

Answer : b,c & d

33) What are the characteristics of mutual exclusion using centralized approach ? (Choose three)
a) One processor as coordinator which handles all requests
b) It requires request,reply and release per critical section entry
c) The method is free from starvation
d) When responses are received from all processes, then process can enter its Critical Section

Answer : a,b & c

34) What are the characteristics of fully distributed approach ? (Choose two)
a) When responses are received from all processes, then process can enter its Critical Section
b) When process exits its critical section, the process sends reply messages to all its deferred requests.
c) It requires request,reply and release per critical section entry
d) One processor as coordinator which handles all requests

Answer : a & b

35) What are the advantages of token(with rings) passing approach ? (Choose three)
a) One processor as coordinator which handles all requests
b) No starvation if the ring is unidirectional
c) There are many messages passed per section entered if few users want to get in section
d) One processor as coordinator which handles all requests
e) Only one message/entry if everyone wants to get in

Answer : a,b & d

36) What is the characteristics of atomicity ?
a) All operations associated are executed to completion or none are performed
b) One processor as coordinator which handles all requests
c) When responses are received from all processes, then process can enter its Critical Section
d) Use communication links

Answer : a

37) What things are transaction coordinator is responsible for ? (Choose three)
a) Starting the execution of the transaction
b) Breaking transaction into a number of subtransactions
c) Coordinating the termination of the transaction
d) Syncronization of the parties

Answer : a,b & c

38) Single coordinator approach has the following advantages : (Choose two)
a) Simple implementation
b) Simple deadlock handling
c) Speed of operation
d) No bottleneck

Answer : a & b

39) Single coordinator approach has the following disadvantages : (Choose two)
a) Bottleneck
b) Slow response
c) Vulnerability
d) One request per second

Answer : a & c

40) What are the two parts of global unique identifier ? (Choose two)
a) Local unique time stamp
b) Remote time stamp
c) Site identifier
d) Clock number

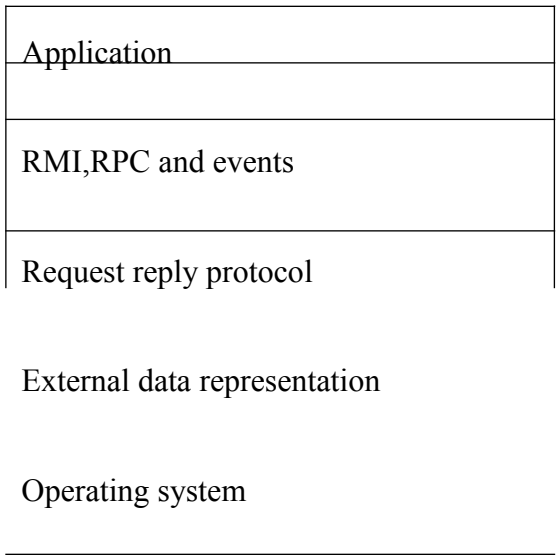
Answer : a & c

UNIT-III

SMALL ANSWER QUESTIONS

UNIT II

1. Draw the Middleware Architecture.



3. What are the benefits of programming with interface in DS?

- As with any form of modular programming, programmers are concerned only with the abstraction offered by the service interface and need not be aware of implementation details.

    i. In potentially heterogeneous distributed systems , programmers also do not need to know the programming language or underlying platform used to implementation service.

    ii. This approach provides natural support for software evolution in this implementation can change as long as the interface remains the same.

4. Define IDL.

Interface Definition Languages (IDLs) are designed to allow procedures implemented in different languages to invoke one another. An IDL provides a notation for defining interfaces in which each of the parameters of an operation may be described as for input or output in addition to having its type specified.

5. List the used of IDL in web services.

The concept of an IDL was initially developed for RPC systems but applies equally to RMI and also web service. Some of them are:

6. Sun XDR as an example of an IDL for RPC

II.CORBA IDL as an example of an IDL for RMI

III. The web service Description Language (WSDL),which is designed for an Internet wide RPC supporting web service.

7. What is meant by action in object model?

Action is an object oriented program is initiated by an object invoking a method in another object. An invocation can include additional information needed to carry out the method. The receiver executes the appropriate method and then returns control to the invoking objects, sometimes supplying a result .An invocation of a method can have three effects:

a The state of the receiver may be changed.

1. A new object may be instantiated, for example, by using a constructor in java or C++.

a. Further invocation on methods in other objects may take place.

8. Define object reference.

Objects can be accessed via object reference. For example in java a variable that appears to hold an object actually holds a reference to that object. To invoke a method in an object the object reference and method name are given together with any necessary arguments. The object whose method name is invoked is sometimes called the target and sometimes the

receiver. Object reference are first class values, meaning that they may be assigned to variables, passed as arguments and returned as results of methods.

7.What is meant by garbage collection?

It is necessary to provide a means of freeing the space occupied by objects when they are no longer needed. A language such as java, that can detect automatically when an object is no longer accessible recovers the space and makes it available for allocation to other objects. This process is called garbage collection; the programmer has to cope with the freeing of space allocated to objects. This can be a major source of errors.

8.List the heart of distributed object model.

Remote object references: Other objects can invoke the methods of a remote object if they have access to its remote object reference. For example a remote object reference for B must be available to A.

The notation of object reference is extended to allow any object that can receive an RMI to have a remote object reference. A remote object reference is an identifier that can be used throughout a distributed system to refer to a particular unique remote object. Its representation which is generally different from that of a local object reference. Remote object references are analogous to local one in that:

2 The remote object to receive a remote method invocation is specified by the invoker as a remote object reference.

3 Remote object references may be passed as arguments and results of remote methods invocations.

Remote interfaces: Every object has a remote interface that specifies which of its methods can be invoked remotely..

The class of a remote object implements the methods of its remote interface, for example as public instance methods in java.Object in other processes can invoke only the methods that belong to its remote interface.

9.Define RMI .

Each process contains a collection of objects, some of which can receive both local and remote invocations whereas the other objects can receive only local invocations as shown in figure.

Method invocation between objects in different processes, whether in the same computer or not, are known as remote method invocations. Method invocation between objects in the same process is local method invocation. We refer to objects that can receive remote invocation as remote objects.

10. What are the main choices to be considered in design of RMI?

RMI invocation semantics

9. Retry-reply protocols, where we showed that doOperation can be implemented in different ways to provide different guarantees.
10. The main choices are:
    1. **Retry request message: C**ontrols whether to retransmit the request message until either a reply is received or the server is assumed to have failed.
    2. **Duplicate filtering:** Controls when retransmissions are used and whether to filter out duplicate requests at the server.

Iii.Retrasmission of results: Controls whether to keep a history of result message to enable lost results to be retransmitted without re-executing the operations at the server.

11. List the choices of RPC invocation semantics.

The choices of RPC invocation semantics are defined as follows:

a **Maybe semantics:** With maybe semantics, the remote procedure call may be executed once or not at all. Maybe semantics is useful only for applications in which occasional failed calls are acceptable. Maybe semantics arises when no fault-tolerance measures are applied and suffer from the following types of failure;

11. Omission failures if the request message is lost;

12. Crash failures when the server containing the remote operation fails.

b**. At-least-once semantics :**With at-least-once semantics ,the invoker receives either a result ,in which case the invoker knows that the procedure was executed at least once ,or an exception informing it that no result was received .At-least-once semantics can be achieved by the retransmission of request message . Which masks the omission failures of the request message? At-least-once semantics can suffer from the following types of failure:

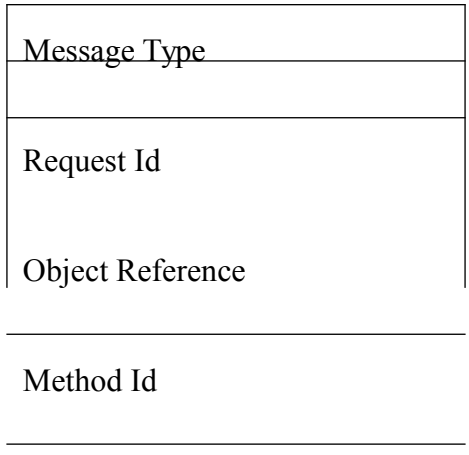
16. crash failure when the sever containing the remote procedure fails;

17. arbitrary failures – incase when the request message is retransmitted, the remote server may

Receive it and execute the procedure more than once, possible causing wrong values to be returned.

c. At-most-once semantics : With at-most-once semantics , the caller receives either a Result , in which case the caller knows that the procedure was executed exactly once, or an Exception informing if that no r using result was received, in which case the procedure will have been executed either once or not at all. At-most-once semantics can be achieved by all, At-most-once semantics can be achieved by using all of the fault-tolerance.

12. Sketch the RMI reply-request message structure.



arguments

13. List the action of remote reference module.

The action of the remote module is as follows:

When a remote object is to be passed as an argument or a result for the first time, the remote reference module is asked to create a remote object reference which it adds to its table.

When a remote object reference arrives in a request or reply message, the remote reference module is asked for the corresponding local object reference, which may refer either to a proxy or to a remote object.

This module is called by components of the RMI software when they are marshalling and unmarshalling remote object reference.

List the function of activator.

processes that start server processes to host remote objects are called activators, for the following reasons.

- A remote object is described as active when it is available for invocation within a running process wherever it is called passive if it is not currently active but can be made active
- A passive object consists of two parts:
  - the implementation of its methods.
  - its state in the marshaled form.

An activator is responsible for:

- Registering passive objects that are available for activation which involves recording the names of servers against the URLs or file names of the

  Corresponding passive objects.

ii Starting named server processes and activating remote objects in them

iii. keeping track of the locations of the servers for the remote objects that it has already activated.

15.What is persistent object store?

An object that is guaranteed to live between activations of processes is called a persistet object. Persistent objects are generally managed by persistent object stores, which store their state in a marshaled from on disk. In generally a persistent object store will manage very large numbers of persistent object which are stored in a disk or in a database until they are needed.

16. How we decide whether the object is persistent or not.

There are two approaches to deciding whether an object is persistent are not:

18. The persistent object store maintains some persistent roots and any object that is reachable from a persistent root is defined to be persistent. This approach is used by persistent java, java Data Objects and perDis. They make use of a garbage collector to dispose of objects that are no longer reachable from the persistent roots.

19. The persistent objects store provides some classes on which persistent is based objects belong to their subclasses.

17. Define RPC.

The software components required to implement RPC are

The client that accesses a service includes one stub procedure for each procedure the service interface. The stub procedure behaves like a local procedure to the client but instead of executing the call it marshals the procedure identifier and the arguments into a request message which it sends via its communication module to the server. When the reply message arrives it unmarshals the results.

The server process contains a dispatcher together with one server stub procedure and one service procedure for each procedure in the service interface. The dispatcher selects one of the server stub procedures according to the procedure identifier in the request message.

18. What is event notification?

The distributed event based system extend the local event model by allowing multiple object at different location to be notified of events takes place at an object. They use the publish subscribe paradigm. A publish subscribe system is a system where publishers publish structured events to an event service and subscriber express interest in particular events through subscriptions which can be arbitrary patterns over the structure events.

19. List the example of publish subscribe system.

Publish-subscribe system is used in a wide variety of application domains particularly those related to a large scale dissemination of events.

2 Financial information systems.

3 Other area with live feeds of real time data(including RSS feeds)

---

1 Supports for cooperative working where a number of participants need to be informed of events of shared interest

2 Support for computing including the management of events from the infrastructure

17. List the characteristics of publish-subscribe system.

Heterogeneity: When event notifications are used as a means of communication components in a distributed system that was not designed to interoperate can be made to work together. All that is required is that event generating objects publish the types of events they offer and that other objects subscribe to patterns of events and provide an interface for receiving and dealing with the resultant notification.

21. Define callbacks.

The general idea behind callbacks is that instead of clients polling the server to find out whether some event has occurred, the server should inform its clients whenever that event occurs. The term callback is used to refer to a server's action of notifying clients about an event.

22. How callback is implemented in RMI.

Callback can be implemented in RMI as follows;

2 The client creates a remote object that implements an interface that contains method for the server to call. We refer to that as a callback object.

3 The server provides an operation allowing interest clients to inform it of the remote object references of their callback objects. It records these in a list.

4 Whenever an event of interest occurs, the server calls the interested clients, For example, the whiteboard server would call its clients whenever a graphical object is added.

23. List the responsibilities of core OS.

The core OS components and their responsibilities are :

Process manager: Creation of and operations upon process. A process is a unit of resource management, including an address space and one or more threads.

Thread manager: Thread creation, synchronization and scheduling. Threads are schedulable activities attached to processes.

Communication manager: Communication between threads attached to different processes on the same computer some kernels also support communication between threads in remort

Processes. Other kernels have no notion of other computers built into them, and an additional service is required for external communication.

Memory manager: Management of physical and virtual memory. It describes the utilization of memory management techniques for efficient data copying and sharing.

Supervisor: Dispatching of interrupts, system call traps and other exceptions; control of memory management unit and hardware caches; processor and floating-point unit register manipulation. This is known as the Hardware Abstraction Layer in Windows.

24. Define process.

A process consists of an execution environment together with one or more threads.

25. Define thread.

A thread is the operating system abstraction of an activity (the term derives from the phrase

'thread of execution'). An execution environment is the unit of resource management: a collection of local kernel managed resources to which its threads have access.

26. Define Unix address space.

This representation of an address space as a sparse set of disjoint regions is a generalization of the UNIX address space, which has three regions: a fixed, unmodifiable text region containing program code; a heap, part of which is initialized by values stored in the program's binary file , and which is extensible towards higher virtual addresses ; and a stack, which is extensible towards lower virtual addresses.

27. List the uses of shared region.

The uses of shared regions include the following:

Libraries: Library code can be very large and would waste considerable memory if it was loaded separately into every process that used it.

Kernel: Often the kernel code and data are mapped into every address space at the same location. Data sharing and

Communication: Two processes, or a process and the kernel, might need to share data in order to cooperate on some task. It can be considerably more efficient for the data to be shared by being mapped as regions in both address spaces than by being passed in messages between them.

18. List the architecture of multi threaded server.

α. Working pool Architecture

β. Thread-per-request Architecture;

Thread-per-connection Architecture

Thread-per-object Architecturre:

d. Compare process and threads.

- Creation a new thread within an existing process is cheaper than creating a process.
    - More importantly switching to a different thread within the same process is cheaper than switching between threads belonging to different processes.
    - Threads within a process may share data and other resources conveniently and efficiently compared with separate processes.
- But by the same token threads within processes are not protected from one another.

30. Explain thread lifetime.

A new thread is created on the same Java Virtual machine (JVM) as its creator in the SUSPENDED state. After it is made RUNNABLE with the start() method, it execute in the run() method of an object designated in its constructor, The JVM and the threads on top of it all execute in a process on top of the underlying operating system. Threads can be assigned a priority so that a java

implementation that supports priorities will run particular threads in preference to any thread with lower priority.

ESSAY QUESTIONS

1. Explain about the API for internet protocols in IPC?

The characteristics of interprocess communication

Message passing between a pair of processes can be supported by two message communication operations, *send* and *receive*, defined in terms of destinations and messages. To communicate, one process sends a message (a sequence of bytes) to a destination and another process at the destination receives the message. This activity involves the communication of data from the sending process to the receiving process and may involve the synchronization of the two processes.

Synchronous and asynchronous communication: A queue is associated with eachmessage destination. Sending processes cause messages to be added to remote queues and receiving processes remove messages from local queues. Communication between the sending and receiving processes may be either synchronous or asynchronous. In the *synchronous* form of communication, the sending and receiving processes synchronize at

every message. In this case, both *send* and *receive* are *blocking* operations. Whenever a *send* is issued the sending process (or thread) is blocked until the corresponding *receive* isissued. Whenever a *receive* is issued by a process (or thread), it blocks until a message arrives.

In the *asynchronous* form of communication, the use of the *send* operation is *nonblocking* in that the sending process is allowed to proceed as soon as the message has been copied to a local buffer, and the transmission of the message proceeds in parallel with the sending process. The *receive* operation can have blocking and non-blocking variants. In the non-blocking variant, the receiving process proceeds with its program after issuing a *receive* operation, which provides a buffer to be filled in the background, but it mustseparately receive notification that its buffer has been filled, by polling or interrupt.

In a system environment such as Java, which supports multiple threads in a single process, the blocking *receive* has no disadvantages, for it can be issued by one thread while other threads in the

process remain active, and the simplicity of synchronizing the receiving threads with the incoming message is a substantial advantage. Non-blocking communication appears to be more efficient, but it involves extra complexity in the receiving process associated with the need to acquire the incoming message out of its flow of control. For these reasons, today‘s systems do not generally provide the nonblocking form of *receive*.

Message destinations: In the Internet protocols, messages are sent to (*Internet address*,*local port*) pairs. A local port is a message destination within a computer, specified as aninteger. A port has exactly one receiver (multicast ports are an exception) but can have many senders. Processes may use multiple ports to receive messages. Any process that knows the number of a port can send a message to it. Servers generally publicize their port numbers for use by clients. If the client uses a fixed Internet address to refer to a service, then that service must always run on the same computer for its address to remain valid. This can be avoided by using the following approach to providing location transparency:

- Client programs refer to services by name and use a name server or binder totranslate their names into server locations at runtime. This allows services to be relocated but not to migrate – that is, to be moved while the system is running.

Reliability: Reliable communication in defined terms of validity and integrity. As far asthe validity property is concerned, a point-to-point message service can be described as reliable if messages are guaranteed to be delivered despite a ‗reasonable‘ number of packets being dropped or lost. In contrast, a point-to-point message service can be described as unreliable if messages are not guaranteed to be delivered in the face of even a single packet dropped or lost. For integrity, messages must arrive uncorrupted and without duplication.

Ordering: Some applications require that messages be delivered in*sender order*–thatis, the order in which they were transmitted by the sender. The delivery of messages out of sender order is regarded as a failure by such applications.

Sockets

Both forms of communication (UDP and TCP) use the *socket* abstraction, which provides an endpoint for communication between processes. Sockets originate from BSD UNIX but are also present in most other versions of UNIX, including Linux as well as Windows and the Macintosh OS.

2. Explain UDP communication in detail?

A datagram sent by UDP is transmitted from a sending process to a receiving process without acknowledgement or retries. If a failure occurs, the message may not arrive. A datagram is transmitted between processes when one process *sends* it and another *receives* it. To send or receive messages a

process must first create a socket bound to an Internet address of the local host and a local port. A server will bind its socket to a *server port* –one that it makes known to clients so that they can send messages to it. A client binds its socket to any free local port. The *receive* method returns the Internet address and port of the sender, in addition to the message, allowing the recipient to send a reply.

The following are some issues relating to datagram communication:

Message size: The receiving process needs to specify an array of bytes of a particular sizein which to receive a message. If the message is too big for the array, it is truncated on arrival. The underlying IP protocol allows packet lengths of up to 216 bytes, which includes the headers as well as the message. However, most environments impose a size restriction of 8 kilobytes. Any application requiring messages larger than the maximum must fragment them into chunks of that size. Generally, an application, for example DNS, will decide on a size that is not excessively large but is adequate for its intended use.

Blocking: Sockets normally provide non-blocking *sends* and blocking *receives* fordatagram communication (a non-blocking *receive* is an option in some implementations). The *send* operation returns when it has handed the message to the underlying UDP and IP protocols, which are responsible for transmitting it to its destination. On arrival, the message is placed in a queue for the socket that is bound to the destination port. The message can be collected from the queue by an outstanding or future invocation of *receive* on that socket. Messages are discarded at the destination if no process already has a socket bound to the destination port.

The method *receives* blocks until a datagram is received, unless a timeout has been set on the socket. If the process that invokes the *receive* method has other work to do while waiting for the message, it should arrange to use a separate thread. For example, when a server receives a message from a client, the message may specify work to do, in which case the server will use separate threads to do the work and to wait for messages from other clients.

Timeouts: The *receive* that blocks forever is suitable for use by a server that is waiting toreceive requests from its clients. But in some programs, it is not appropriate that a process that has invoked a *receive* operation should wait indefinitely in situations where the sending process may have crashed or the expected message may have been lost. To allow for such requirements, timeouts can be set on sockets. Choosing an appropriate

Timeout interval is difficult, but it should be fairly large in comparison with the time required to transmit a message.

Receive from any: The *receive* method does not specify an origin for messages. Instead, an invocation of *receive* gets a message addressed to its socket from any origin. The *receive* method returns the Internet address and local port of the sender, allowing the recipient to check where the message came from. It is possible to connect a datagram socket to a particular remote port and Internet address, in which case the socket is only able to send messages to and receive messages from that address.

Failure model for UDP datagram's: Reliable communication is defined in terms of twoproperties: integrity and validity. The integrity property requires that messages should not be corrupted or duplicated. The use of a checksum ensures that there is a negligible probability that any message received is corrupted. UDP datagram's suffer from the following failures:

Omission failures: Messages may be dropped occasionally, either because of a checksum error or because no buffer space is available at the source or destination. To simplify the discussion, we regard send-omission and receive-omission failures as omission failures in the communication channel.

Ordering: Messages can sometimes be delivered out of sender order. Applications usingUDP datagram's are left to provide their own checks to achieve the quality of reliable communication they require. A reliable delivery service may be constructed from one that suffers from omission failures by the use of acknowledgements.

Use of UDP: For some applications, it is acceptable to use a service that is liable tooccasional omission failures. For example, the Domain Name System, which looks up DNS names in the Internet, is implemented over UDP. Voice over IP (VOIP) also runs over UDP. UDP datagrams are sometimes an attractive choice because they do not suffer

from the overheads associated with guaranteed message delivery. There are three main sources of overhead:

3. Explain TCP communication in detail?

The API to the TCP protocol, which originates from BSD 4.x UNIX, provides the abstraction of a stream of bytes to which data may be written and from which data may be read. The following characteristics of the network are hidden by the stream abstraction:

Message sizes: The application can choose how much data it writes to a stream or readsfrom it. It may deal in very small or very large sets of data. The underlying implementation of a TCP stream decides

how much data to collect before transmitting it as one or more IP packets. On arrival, the data is handed to the application as requested. Applications can, if necessary, force data to be sent immediately.

***Lost messages*:** The TCP protocol uses an acknowledgement scheme. As an example of asimple scheme (which is not used in TCP), the sending end keeps a record of each IP packet sent and the receiving end acknowledges all the arrivals. If the sender does not receive an acknowledgement within a timeout, it retransmits the message. The more sophisticated sliding window scheme cuts down on the number of acknowledgement messages required.

***Flow control*:** The TCP protocol attempts to match the speeds of the processes that readfrom and write to a stream. If the writer is too fast for the reader, then it is blocked until the reader has consumed sufficient data.

***Message duplication and ordering*:** Message identifiers are associated with each I packet, which enables the recipient to detect and reject duplicates, or to reorder messages that do not arrive in sender order.

***Message destinations*:** A pair of communicating processes establishes a connection before they can communicate over a stream. Once a connection is established, the processes simply read from and write to the stream without needing to use Internet addresses and ports. Establishing a connection involves a *connect* request from client to server followed by an *accept* request from server to client before any communication can take place. This could be a considerable overhead for a single client-server request and reply.

The API for stream communication assumes that when a pair of processes are establishing a connection, one of them plays the client role and the other plays the server role, but thereafter they could be peers. The client role involves creating a stream socket bound to any port and then making a *connect* request asking for a connection to a server at its server port. The server role involves creating a listening socket bound to a server port and waiting for clients to request connections. The listening socket maintains a queue of incoming connection requests. In the socket model, when the server *accept*s a

connection, a new stream socket is created for the server to communicate with a client, meanwhile retaining its socket at the server port for listening for *connect* requests from other clients.

The pair of sockets in the client and server are connected by a pair of streams, one in each direction. Thus each socket has an input stream and an output stream. One of the pair of processes can send information to the other by writing to its output stream, and the other process obtains the information

by reading from its input stream. When an application *closes* a socket, this indicates that it will not write any more data to its output stream. Any data in the output buffer is sent to the other end of the stream and put in the queue at the destination socket, with an indication that the stream is broken. The process at the destination can read the data in the queue, but any further reads after the queue is empty will result in an indication of end of stream. When a process exits or fails, all of its sockets are eventually closed and any process attempting to communicate with it will discover that its connection has been broken.

The following are some outstanding issues related to stream communication:

***Matching of data items*:** Two communicating processes need to agree as to the contents of the data transmitted over a stream. For example, if one process writes an *int* followed by a *double* to a stream, then the reader at the other end must read an *int* followed by a *double.* When a pair of processes does not cooperate correctly in their use of a stream, thereading process may experience errors when interpreting the data or may block due to insufficient data in the stream.

***Blocking*:** The data written to a stream is kept in a queue at the destination socket. Whena process attempts to read data from an input channel, it will get data from the queue or it will block until data becomes available. The process that writes data to a stream may be blocked by the TCP flow-control mechanism if the socket at the other end is queuing as much data as the protocol allows.

***Threads*:**When a server accepts a connection, it generally creates a new thread in whichto communicate with the new client. The advantage of using a separate thread for each client is that the server can block when waiting for input without delaying other clients. In an environment in which threads are not provided, an alternative is to test whether input is available from a stream before attempting to read it; for example, in a UNIX environment the *select* system call may be used for this purpose.

Failure model: To satisfy the integrity property of reliable communication, TCP streamsuse checksums to detect and reject corrupt packets and sequence numbers to detect and reject duplicate packets. For the sake of the validity property, TCP streams use timeouts and retransmissions to deal with lost packets. Therefore, messages are guaranteed to be delivered even when some of the underlying packets are lost. But if the packet loss over a connection passes some limit or the network connecting a pair of communicating processes is severed or becomes severely congested, the TCP software responsible for sending messages will receive no acknowledgements and after a time will declare the connection to be broken. Thus TCP does not provide reliable communication, because it does not guarantee to deliver messages in the face of all possible difficulties.

4. Explain in detail about External data Representation and marshaling?

One of the following methods can be used to enable any two computers to exchange binary data values:

1 The values are converted to an agreed external format before transmission and converted to the local form on receipt; if the two computers are known to be the same type, the conversion to external format can be omitted.

2 The values are transmitted in the sender‘s format, together with an indication of the format used, and the recipient converts the values if necessary.

Note, however, that bytes themselves are never altered during transmission. To support RMI or RPC, any data type that can be passed as an argument or returned as a result must be able to be flattened and the individual primitive data values represented in an agreed format. An agreed standard for the representation of data structures and primitive values is called an *external data representation*.

Marshalling is the process of taking a collection of data items and assembling them into aform suitable for transmission in a message. *Unmarshalling* is the process of disassembling them on arrival to produce an equivalent collection of data items at the destination. Thus marshalling consists of the translation of structured data items and primitive values into an external data representation. Similarly, unmarshalling consists of the generation of primitive values from their external data representation and the rebuilding of the data structures.

Three alternative approaches to external data representation and marshalling are discussed:

3 CORBA‘s common data representation, which is concerned with an external representation for the structured and primitive types that can be passed as the arguments and results of remote method invocations in CORBA. It can be used by a variety of programming languages.

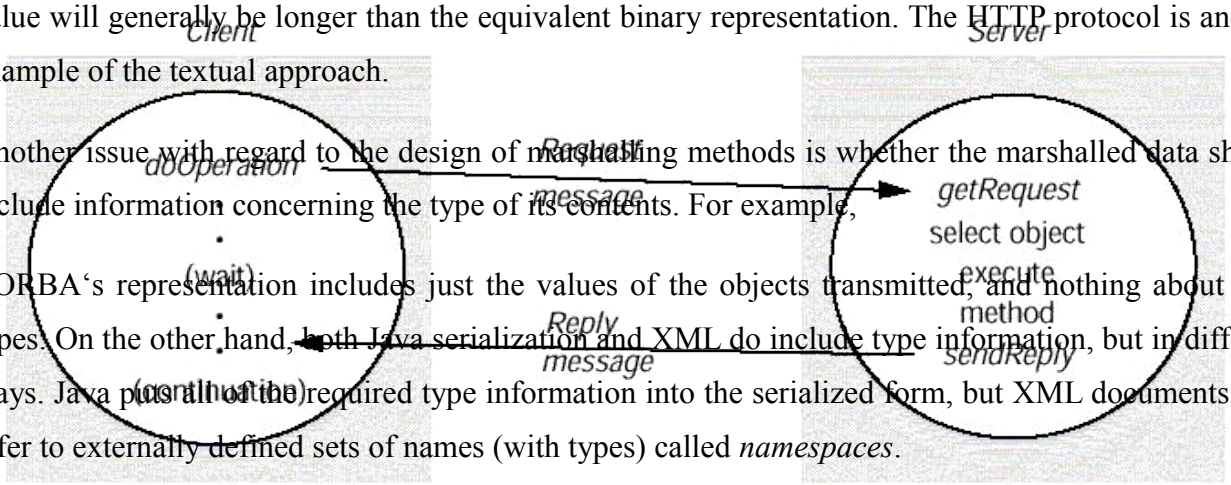
4 Java‘s object serialization, which is concerned with the flattening and external data representation of any single object or tree of objects that may need to be transmitted in a message or stored on a disk. It is for use only by Java.

5 XML (Extensible Markup Language), which defines a textual format for representing structured data. It was originally intended for documents containing textual self-describing

structured data – for example documents accessible on the Web – but it is now also used to represent the data sent in messages exchanged by clients and servers in web services.

In the first two cases, the marshalling and unmarshalling activities are intended to be carried out by a middleware layer without any involvement on the part of the application programmer. Even in the case of XML, which is textual and therefore more accessible to hand-encoding, software for marshalling and unmarshalling is available for all commonly used platforms and programming environments. Because marshalling requires the consideration of all the finest details of the representation of the primitive components of composite objects, the process is likely to be error-prone if carried out by hand. Compactness is another issue that can be addressed in the design of automatically generated marshalling procedures.

In the first two approaches, the primitive data types are marshalled into a binary form. In the third approach (XML), the primitive data types are represented textually. The textual representation of a data value will generally be longer than the equivalent binary representation. The HTTP protocol is another example of the textual approach.



Another issue with regard to the design of marshalling methods is whether the marshalled data should include information concerning the type of its contents. For example,

CORBA's representation includes just the values of the objects transmitted, and nothing about their types. On the other hand, both Java serialization and XML do include type information, but in different ways. Java puts all of the required type information into the serialized form, but XML documents may refer to externally defined sets of names (with types) called *namespaces*.

5. Explain about client server communication in detail?

The client-server communication is designed to support the roles and message exchanges in typical client-server interactions. In the normal case, request-reply communication is synchronous because the client process blocks until the reply arrives from the server. Asynchronous request-reply communication is an alternative that is useful where clients can afford to retrieve replies later. It is often built over UDP datagrams. Client-server protocol consists of request/response pairs, hence no acknowledgements at transport layer are necessary. So, it has no connection establishment overhead. There is no need for flow control due to small amounts of data being transferred. The request-reply protocol was based on a trio of communication primitives: do Operation, get Request, and send Reply shown in Figure

messageType	*int (0=Request, 1= Reply)*
requestId	*int*
objectReference	*RemoteObjectRef*
methodId	*int or Method*
arguments	*// array of bytes*

The designed request-reply protocol matches requests to replies. If UDP datagrams are used, the delivery guarantees must be provided by the request-reply protocol, which may use the server reply message as an acknowledgement of the client request message. Figure 2.2.5 outlines the three communication primitives.

public byte[] doOperation (RemoteObjectRef o, int methodId, byte[] arguments)
sends a request message to the remote object and returns the reply.
The arguments specify the remote object, the method to be invoked and the arguments of that method.

public byte[] getRequest ();
acquires a client request via the server port.

public void sendReply (byte[] reply, InetAddress clientHost, int clientPort);
sends the reply message *reply* to the client at its Internet address and port.

The information to be transmitted in a request message or a reply message is shown in Figure

In a protocol message:

6 The first field indicates whether the message is a request or a reply message.

7 The second field request id contains a message identifier.

8 The third field is a remote object reference.

9 The forth field is an identifier for the method to be invoked.

Message identifier: A message identifier consists of two parts:

10 A requestId, which is taken from an increasing sequence of integers by the sending process

11 An identifier for the sender process, for example its port and Internet address. Failure model of the request-reply protocol: If the three primitive doOperation, getRequest, and sendReply are implemented over UDP datagram, they have the same communication failures.

12 Omission failure

13 Messages are not guaranteed to be delivered in sender order.

6. explain briefly about RMI?

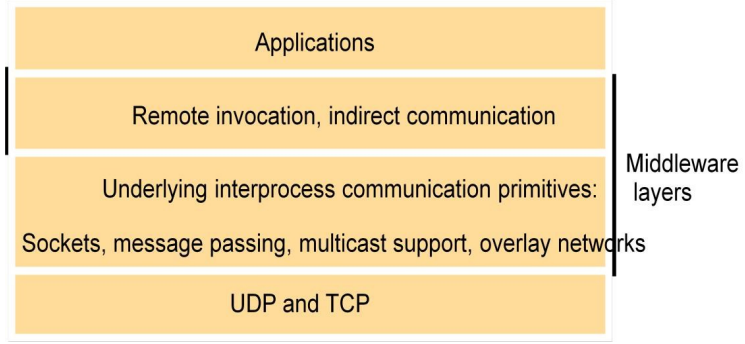
Most current distributed systems software is written in object-oriented languages and RPC can be understood in relation to RMI.

Therefore this unit concentrates on the RMI and event paradigms, each of which applies to distributed objects and communication between distributed objects.

Middleware: It is software that provides a programming model about the basic building blocks of processes and messages passing. The middleware layer uses protocols based on messages between processes to provide its higher-level abstractions such as remote invocations and events as shown in Figure 2.3.1.

An important aspect of middleware is the provision of location transparency and independence from the details of communication protocols, operating systems and computer hardware. Some forms of middleware allow the separate components to be written in different programming languages.

Location transparency: In RPC, the client that calls a procedure cannot tell whether theprocedure runs in the same process or in a different process, possibly on a different computer, nor does the client need to know the location of the server. Similarly, in RMI the objects making the invocation cannot tell whether the object it invokes is local or not and does not need to know its location. Also in distributed event-based programs, the objects generating events and the objects that receive notifications of those events need not be aware of one another‘s' locations.



Communication protocols: The protocols that support the middleware abstractions areindependent of the underlying transport protocols. For example, the request-reply protocol can be implemented over either LTDP or TCP.

Computer hardware: Two agreed standards for external data representation aredescribed in Unit 2 of Module 2. These are used when marshalling and unmarshalling messages. They hide the differences due to hardware architectures, such as byte ordering.

Operating systems: The higher-level abstractions provided by the middleware layer areindependent of the underlying operating systems.

Use of several programming languages: Some middleware is designed to allow distributed applications to use more than one programming language. In particular, CORBA allows clients written

in one language to invoke methods in objects that live in server programs written in another language. This is achieved by using an interface definition language or IDL to define interfaces.

Interfaces

Most modern programming languages provide a means of organizing a program as a set of modules that can communicate with one another. Communication between modules can be by means of procedure calls between modules or by direct access to the variables in another module. In order to control the possible interactions between modules, an explicit interface is defined for each module. The interface of a module specifies the procedures and the variables that can be accessed from other modules. Modules are implemented so as to hide all the information about them except that which is available through its interface. So long as its interface remains the same, the implementation may be changed without affecting the users of the module.

Interfaces in distributed systems: In a distributed program, the modules can run inseparate processes. It is not possible for a module running in one process to access the variables in a module in another process. Therefore, the interface of a module that is intended for RPC or RMI cannot specify direct access to variables. Note that CORBA IDL interfaces can specify attributes, which seems to break this rule. However, the attributes are not accessed directly but by means of some getter and setter procedures added automatically to the interface.

The parameter-passing mechanisms (for example call by value and call by reference) used in local procedure call are not suitable when the caller and procedure are in different processes. The specification of a procedure or method in the interface of a module in a distributed program describes the parameters as input or output or sometimes both. Input parameters are passed to the remote module by sending the values of the arguments in the request message and then supplying them as arguments to the operation to be executed in the server. Output parameters are returned in the reply message and are used as the result of the call or to replace the values of the corresponding variables in the calling environment. When a parameter is used for both input and output, the value must be transmitted in both the request and reply messages. Another difference between local and remote modules is that pointers in one process are not valid in another remote one. Therefore, pointers cannot be passed as arguments or returned as results of calls to remote modules.

Now we discuss the interfaces used in the original client-server model for RPC and in the distributed object model for RMI.

Service interfaces: In the client-server model, each server provides a set of procedures that are available for use by clients. For example, a file server would provide procedures for reading and writing files. The term service interface is used to refer to the specification of the procedures offered by a server, defining the types of the input and output arguments of each of the procedures.

Remote interfaces: In the distributed object model, a remote interface specifies the methods of an object that are available for invocation by objects in other processes, defining the types of the input and output arguments of each of them. However, the big difference is that the methods in remote interfaces can pass objects as arguments and results of methods. In addition, references to remote objects may also be passed — these should not be confused with pointers, which refer to specific memory locations.

Neither service interfaces nor remote interfaces may specify direct access to variables. In the latter case, this prohibits direct access to the instance variables of an object.

Interface definition languages: An RMI mechanism can be integrated with a particular programming language if it includes an adequate notation for defining interfaces, allowing input and output parameters to be mapped onto the language's normal use of parameters.

7. Explain about COMMUNICATION BETWEEN DISTRIBUTED OBJECTS

The object-based model for a distributed system extends the model supported by object-oriented programming languages to make it apply to distributed objects. Here we address communication between distributed objects by means of RMI. The material is presented under the following headings:

The object model: A brief review of the relevant aspects of the object model, suitable forthe reader with a basic knowledge of an object-oriented programming language, for example Java or C++.

Distributed objects: A presentation of object-based distributed systems, which arguesthat the object model is very appropriate for distributed systems.

The distributed object model: A discussion of the extensions to the object modelnecessary for it to support distributed objects.

Design issues: A set of arguments about the design alternatives:

14 Local invocations are executed exactly once, but what suitable semantics is possible for remote invocations?

15 How can RMI semantics be made similar to those of local method invocation and

What differences cannot be eliminated?

Implementation: An explanation as to how a layer of middleware above the request-reply protocol may be designed to support RMI between application-level distributed objects.

Distributed garbage collection: A presentation of an algorithm for distributed garbage collection that is suitable for use with the RMI implementation.

The object model

An object-oriented program, for example in Java or C++, consists of a collection of interacting objects, each of which consists of a set of data and a set of methods. An object

Communicates with other objects by invoking their methods, generally passing arguments and receiving results.

Objects can encapsulate their data and the code of their methods. Some languages, for example Java and C++, allow programmers to define objects whose instance variables can be accessed directly. But for use in a distributed object system, an object's data should be accessible only via its methods.

Object references: Objects can be accessed via object references. For example, in Java, a variable that appears to hold an object actually holds a reference to that object. To invoke a method in an object, the object reference and method name are given, together with any necessary arguments. The object whose method is invoked is sometimes called the target and sometimes the receiver. Object references are first-class values, meaning that they may, for example, be assigned to variables, passed as arguments and returned as results of methods.

Interfaces: An interface provides a definition of the signatures of a set of methods (that is, the types of their arguments, return values and exceptions) without specifying their implementation. An object will provide a particular interface if its class contains code that implements the methods of that interface. In Java, a class may implement several interfaces, and the methods of an interface may be implemented by any class. An interface also defines a type that can be used to declare the type of variables or of the parameters and return values of methods. Note that interfaces do not have constructors.

Actions: Action in an object-oriented program is initiated by an object invoking amethod in another object. An invocation can include additional information (arguments) needed to carry out the method. The receiver executes the appropriate method and then returns control to the invoking object, sometimes supplying a result. An invocation of a method can have two effects:

16 The state of the receiver may be changed, and

17 Further invocations on methods in other objects may take place.

As an invocation can lead to further invocations of methods in other objects, an action is a chain of related method invocations, each of which eventually returns. This explanation does not take account of exceptions.

Exceptions: Programs can encounter many sorts of errors and unexpected conditions ofvarying seriousness. During the execution of a method, many different problems may be discovered: for example, inconsistent values in the object's variables, or failure in attempts to read or write to files or network sockets.

When programmers need to insert tests in their code to deal with all possible unusual or erroneous cases, this detracts from the clarity of the normal case. Exceptions provide a clean way to deal with error conditions without complicating the code. In addition, each method heading explicitly lists as exceptions the error conditions it might encounter, allowing users of the method to deal with them. A block of code may he defined to throw an exception whenever particular unexpected conditions or errors arise. This means that control passes to another block of code that catches the exception. Control does not return to the place where the exception was thrown.

Garbage collection: It is necessary to provide a means of freeing the space occupied byobjects when they are no longer needed. A language, for example Java, that can detect automatically when an object is no longer accessible recovers the space and makes it available for allocation to other objects. This process is called garbage collection. When a language (for example C++) does not support garbage collection, the programmer has to cope with the freeing of space allocated to objects. This can be a major source of errors.

8. Explain about distributed objects?

The state of an object consists of the values of its instance variables. In the object-based paradigm the state of a program is partitioned into separate parts, each of which is associated with an object. Since object-based programs are logically partitioned, the physical distribution of objects into different processes or computers in a distributed system is a natural extension. Distributed object systems may

adopt the client-server architecture. In this case, objects are managed by servers and their clients invoke their methods using remote method invocation. In RMI, the client's request to invoke a method of an object is sent in a message to the server managing the object. The invocation is carried out by executing a method of the object at the server and the result is returned to the client in another message. To allow for chains of related invocations, objects in servers are allowed to become clients of objects in other servers. Distributed objects can assume the other architectural models. For example, objects can be replicated in order to obtain the usual benefits of fault tolerance and enhanced performance, and objects can be migrated with a view to enhancing their performance and availability. Having client and server objects in different processes enforces encapsulation. That is, the state of an object can be accessed only by the methods of the object, which means that it is not possible for unauthorized methods to act on the state. For example, the possibility of concurrent RMIs from objects in different computers implies that an object may be accessed concurrently. Therefore, the possibility of conflicting accesses arises.

However, the fact that the data of an object is accessed only by its own methods allows objects to provide methods for protecting themselves against incorrect accesses. For example, they may use synchronization primitives such as condition variables to protect access to their instance variables.

Another advantage of treating the shared state of a distributed program as a collection of objects is that an object may be accessed via RMI or it may be copied into a local cache and accessed directly provided that the class implementation is available locally.

The fact that objects are accessed only via their methods gives another advantage for heterogeneous systems in that different data formats may be used at different sites — these formats will be unnoticed by clients that use RMI to access the methods of the objects.

MUTIPLE CHOICE QUESTIONS

1) Inter process communication :
a) allows processes to communicate and synchronize their actions when using the same address space.
b) Allows processes to communicate and synchronize their actions without using the same address space.
c) allows the processes to only synchronize their actions without communication.
d) None of these

Answer: b

2) Message passing system allows processes to :
a) communicate with one another without resorting to shared data.
b) communicate with one another by resorting to shared data.
c) share data
d) name the recipient or sender of the message

Answer: a

3) An IPC facility provides atleast two operations : (choose two)
a) write message
b) delete message
c) send message
d) receive message

Answer: c and d

4) Messages sent by a process :
a) have to be of a fixed size
b) have to be a variable size
c) can be fixed or variable sized
d) None of these

Answer: c

5) The link between two processes P and Q to send and receive messages is called
a) communication link
b) message-passing link
c) synchronization link
d) All of these

Answer: a

6) Which of the following are TRUE for direct communication :(choose two)
a) A communication link can be associated with N number of process(N = max. number of processes supported by system)
b) A communication link can be associated with exactly two processes
c) Exactly N/2 links exist between each pair of processes(N = max. number of processes supported by system)
d) Exactly one link exists between each pair of processes

Answer: b and d

7) In indirect communication between processes P and Q :
a) there is another process R to handle and pass on the messages between P and Q
b) there is another machine between the two processes to help communication
c) there is a mailbox to help communication between P and Q
d) None of these

Answer: c

8) In the non blocking send :
a) the sending process keeps sending until the message is received
b) the sending process sends the message and resumes operation
c) the sending process keeps sending until it receives a message
d) None of these

Answer: b

9) In the Zero capacity queue : (choose two)
a) the queue has zero capacity
b) the sender blocks until the receiver receives the message
c) the sender keeps sending and the messages dont wait in the queue
d) the queue can store atleast one message

Answer: a and b

10) The Zero Capacity queue :
a) is referred to as a message system with buffering
b) is referred to as a message system with no buffering
c) is referred to as a link
d) None of these

Answer: b

11) Bounded capacity and Unbounded capacity queues are referred to as :
a) Programmed buffering
b) Automatic buffering
c) User defined buffering
d) No buffering

Answer: b

12)Thread is also called
a) Light Weight Process(LWP)
b) Heavy Weight Process(HWP)
c) Process
d) None of these

Answer: a

13) thread shares its resources(like data section, code section, open files, signals) with :
a) other process similar to the one that the thread belongs to
b) other threads that belong to similar processes
c) other threads that belong to the same process
d) All of these

Answer: c

14) A heavy weight process :
a) has multiple threads of execution

b) has a single thread of execution
c) can have multiple or a single thread for execution
d) None of these

Answer: b

15) 15) UDP and TCP are both ..... layer protocols.

A. data link

B. network

C. transport

D. interface

Ans C

16)..... is a process-to-process protocol that adds only port addresses, checksum error control, and length information to the data from upper layer.

A. TCP

B. UDP

C. IP

D. ARP

Ans B

17) Which of the following functions does UDP perform?

A. Process-to-process communication

B. Host-to-host communication

C. End-to-end reliable data delivery

D. Interface-to-interface communication.

Ans A

18) A port address in TCP/IP is .....bits long.

A. 32

B. 48

C. 16

D. 64

Ans C

19) When the IP layer of a receiving host receives a datagram, ...

A. delivery is complete

B. a transport layer protocol takes over

C. a header is added

D. a session layer protocol takes over

Ans B

20. In RPC, while a server is processing the call, the client is blocked
a) unless the client sends an asynchronous request to the server
b) unless the call processing is complete
c) for the complete duration of the connection
d) none of the mentioned

Answer:a

21. Remote procedure calls is
a) inter-process communication
b) a single process
c) a single thread
d) none of the mentioned

Answer:a

22. RPC allows a computer program to cause a subroutine to execute in
a) its own address space
b) another address space
c) both (a) and (b)
d) none of the mentioned

Answer:b

23 RPC works between two processes. These processes must be
a) on the same computer
b) on different computers connected with a network
c) both (a) and (b)
d) none of the mentioned

Answer:c

24. A remote procedure is uniquely identified by
a) program number
b) version number
c) procedure number
d) all of the mentioned
Answer:d

25 An RPC application requires
a) specific protocol for client server communication
b) a client program
c) a server program
d) all of the mentioned

Answer:d
26. RPC is used to
a) establish a server on remote machine that can respond to queries
b) retrieve information by calling a query
c) both (a) and (b)
d) none of the mentioned

Answer:c

27.RPC is a
a) synchronous operation
b) asynchronous operation
c) time independent operation
d) none of the mentioned

Answer:a

28.The local operating system on the server machine passes the incoming packets to the
a) server stub
b) client stub
c) client operating system
d) none of the mentioned

Answer:a

29)Ethernet frame consists of
a) MAC address
b) IP address
c) both (a) and (b)
d) none of the mentioned

Answer:a

30) 1) TCP is a ..... protocol.

A. stream-oriented

B. message-oriented

C. block-oriented

D. packet-oriented

Ans A

31) Which of the following is not the layer of TCP/IP protocol.

A. Physical layer

B. link layer

C. network layer

D. transport layer.

Ans A

32) TCP groups a number of bytes together into a packet called a ....

A. user datagram

B. segment

C. datagram

D. packet

Ans B

33) The ..... of TCP/IP protocol is responsible for figuring out how to get data to its destination.

A. application layer

B. link layer

C. network layer

D. transport layer.

Ans C

34) TCP is a(n) ..... Transport protocol.

A. protocol delivery
B. reliable

C. best-effort delivery

D. effortless delivery

Ans B

35) ..... is the protocol that hides the underlying physical network by creating a virtual network view.

A. Internet Protocol (IP)

B. Internet Control Message Protocol(ICMP)

C. Address Resolution Protocol(ARP)

D. Bootstrap Protocol(BOOTP)

Ans A

36) To use the services of UDP, we need ..... socket addresses.

A. four

B. two

C. three

D. four

Ans B

37) Which of the following is not the name of Regional Internet Registries (RIR) to administer the network number portion of IP address?

A. American Registry for Internet Numbers (ARIN)

B. Reseaux IP Europeans(RIPE)

C. Europeans Registry for Internet Numbers(ERIN)

D. Asia Pacific Network Information Center(APNIC)

Ans C

38) UDP packets are called.....

A. user datagram's

B. segments

C. frames

D. packets

Ans A

39) ..... addresses use 21 bits for the and 8 bits for the portion of the IP address for TCP/IP network.

A. Class A

B. Class B

C. Class C

D. Class D

Ans C

40) UDP packets have fixed-size header of ..... bytes.

A. 16

B. 8

C. 32

D. 64

Ans B

UNIT-IV
SMALL ANSWER QUESTIONS

1. Enumerate the properties of storage system?

	Sharing	Persistent	Distributed cache/replicas	Consistency maintenance	Example
Main memory	No	No	No	1	RAM
File system n	No	Yes	No	1	UNIX file system
Distributed file system	Yes	Yes	Yes	Yes	Sun NFS
web	Yes	Yes	Yes	No	Web server
Distributed shared memory	Yes	No	Yes	Yes	Ivy(DSM)
Remote objects(RMI/ORB)	Yes	No	No	1	CORBA
Persistent object store	Yes	Yes	No	1	CORBA persistent state service

Peer to peer storage system	Yes	Yes	Yes	2	Ocean Store

2. List out file system modules.

Directory module:	Relates file names to file IDs
File module:	Relates file IDs to particular files
Access control module:	Checks permission for operation requested
File access module:	Read or writes file data or attributes
Block module:	Accesses and allocates disk blocks
Device module:	Disk I/O and buffering

3. Sketch the file attributes and record structure.

File Length
Creation Time stamp
Read Timestamp
Write time stamp
Attribute time stamp
Reference count
Owner
File Type
Access control List

4. List out the UNIX file system Operations:

fields=open(name, mode)

fields=create(name, mode)

status=close(fieldes)

count=read(fieldes,buffer,n)

count=write(fieldes,buufer,n)

pos=Iseek(filedes,offset,whence)

status=unlink(nmae)

status=link(name1,nmae2)

status=stat(name,buffer)

13. List out the transparencies in file system.

1. Access transparency
2. Location transparency
3. Mobility transparency
4. Performance transparency
5. Scaling transparency

14. What is meant by concurrency control?

Changes to a file by one client should not interfere with the operation of other clients simultaneously accessing or changing the same file. This is well-known issue of concurrency control .The need for concurrency control for accss to shared data in many applications Is widely accepted and techniques are known for its implementation ,but they are costly .Most current file services follow morden UNIX standards in providing advisery or mandatoryfile or record-level locking.

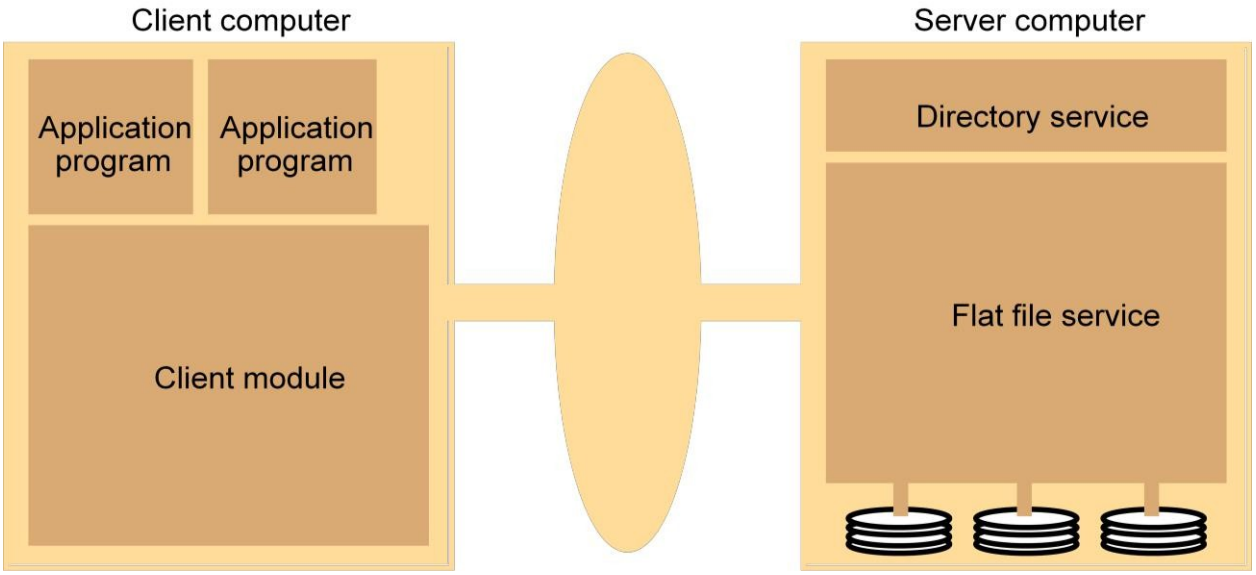
7. What is file replication?

In a file service that supports replication, a file may be represented by several copies of its contents at different locations. This has two benefits-its enables multiple servers to share the load of providing a service to clients accessing the same set of files, enhancing the scalability of the service, and it enhances fault tolerance by enabling clients to locate another server that holds a copy of the file when one has failed. Few file services support replication fully, but most support the catching of files or portions of files locally, a limited form of replication.

8. What is meant by directory services?

The directory services provide a mapping between text names for files and their UFIDs. Client may obtain the UFIDs of a file by quoting its text name to the directory services. The directory services provides the function needed to generate directories, to add new file name to directories and to obtain UFIDs from directories. It is client of the flat file services; its directory is stored in files of the flat services. When a hierarchic file-naming scheme is adopted as in UNIX, directories hold references to other directories.

9.Sketch the file service architecture?



10. List the flat file service operation.

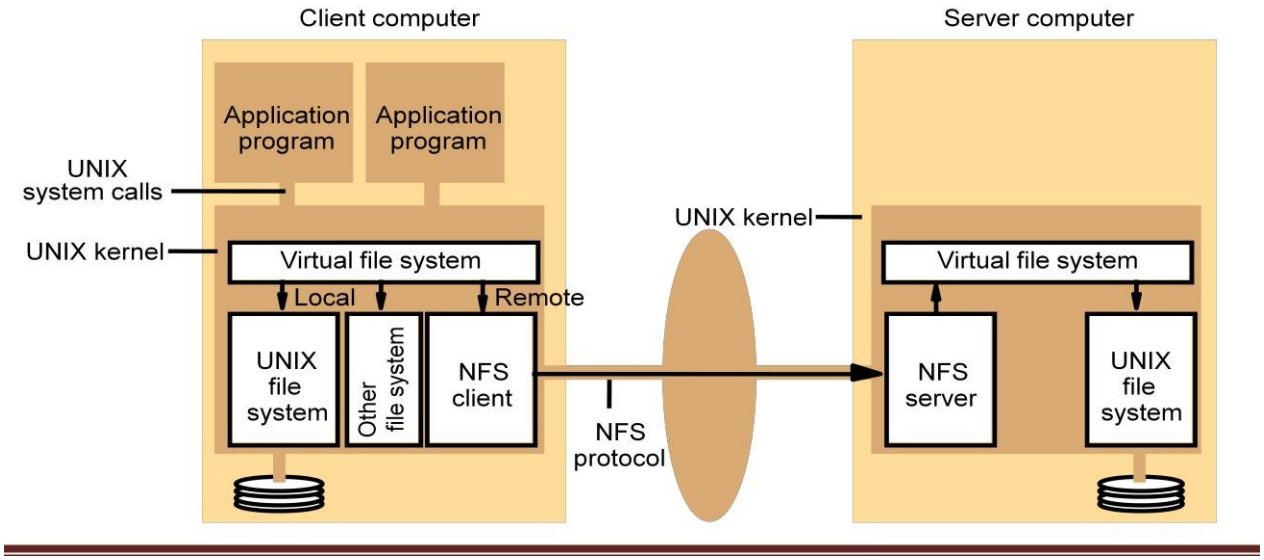
Read (file/d,I,N)>data-throws bad position - if $1 \leq 1 \leq$ length(file):reads a sequence of up to

	N items From a file starting at item/and returns it in data
Write(File/D,I,Ddata)-throws bad position	- if 1 ≤ 1 ≤ length(file)+1: writes a sequence of data to a File, starting at item 1,extending the file if necessary
Create()->FileID	-creates a new file of length 0 and delivers a UFID for it
Delete(FileID)	-removes the file from the file store
GetAttributes(FileID)->>	-returns the file attributes for the file
SetAttributes(FileID)	-sets the file attributes(only those attributes that not Shaded in)

11.List the directory service operation.

Lookup(Dir,Name)→FileID –throws not found	Locate the text name in the directory and returns the relevant UFID.If name is not in the directory, throws an exception
AddName(Dir,Name,File)	If the nameis not in the directory, adds(Nmae,File) to the directory and updates the file's Attribute record.If name is already in the Directory;throws an exception.
UnName (Dir,Name)—throws not found	If name is in the directory: The entry containing name is removed from the directory. If name is not in the directory;throws an exception.
GetName(Dir,Pattern)→Names	Returns all the text names in the directory that match the regular expression pattern.

12. Sketch NFS architecture.



13. List the NFS file server operation.

Lookup(DirFH,Name)→FH,Attr	Returns File handles and attributes for the File Name in the directory DirFH.
Create(DirFH,Name,Attr)→NewFH,Attr	Creates a new file in directory DirFH with Attributes Attr and Returns the new File handle and Attributes.
Remove(DirFH,Name) Status	Removes file name from directory DirFH.
GetAttr(FH)→Attr	Returns the file attributes of file FH.(similar to UNIX stat system call)
Read(FH,Offset,count)→Attr,Dir	Returns up to count bytes of data from the file starting at the offset ,also return the attributes of the file.
Write(FH,Offset,count,Data)→Attr	Writes count bytes of to a file starting at offset. Returns the attributes of the file after write has taken place.

What are the timestamps in called caching?

1. Tc is the time when the cache entry was last validated.
2. Tm is the when the block was last modified at the server.
3. A cache entry is valid at time T if T-Tc is less than a freshness interval t,or if the value for Tm recorded t the client matches the value of Tm at the server (that is,the data has not been modified at the server since the cache entry was made).

15. What is condition used to validate caching?

(T-Tc<t) v (Tm client=Tm server)

16.write the measures to be considered to reduce traffic in getter.

Whenever a new value of Tmserver is received at a client, it is applied to all cache entries derived from the relevant file.

The current attribute values are sent 'piggybacked' with the result of every operation on a file, and if the value of Tmserver has changed the client uses it to update the cache entries relating to the file.

The adaptive algorithm for setting freshness interval t outlined above reduces the traffic considerably for most files.

15. When the name is resolved?

The name is resolved when it is translated into data about the named resource or object, often in order to invoke an action upon it. The association between a name and an object is called a binding. In general, names are bound to attributes of the named objects, rather than the implementation of the objects themselves. An attribute is the value of a property associated with an object.

18. What is meant by URI?

URI-Uniform Resource Identifiers came about from the need to identify resources on the web, and other internet resources such as electronic mailboxes. An important goal was to identify resources in a coherent way, so that they could all be processed by common software such as browser. URIs is 'uniform' in that their syntax incorporates that of indefinitely many individual types of resource identifier(i.e URI schemas),and there are procedures for managing the global namespace of schemas. The advantage of uniformity is that eases the process of introducing new types of identifier, as well as using existing types of identifier in new contexts without disrupting existing usage.

19. What is mean by URN?

Uniform Resource Names are URIs that are used as pure resource names rather than locators.

For example, the URI:

Mid:0E4FC272-5C02-11D9-B115-000A95B55BC8@hpl.hp.com

Is a URN that identifies the email message containing it in its 'message-id' field. The URI distinguishes that message from any other email message. But it does not provide the message's address in any store, so a lookup operation is needed to find it.

20.What is global name services?

The Global name Service developed at the Digital Equipment corporation systems, Research Center,is a descentdant of Grapevine with ambitious goals, including:

16. com- Commercial organization.
17. edu- universities and other educational institutions.
18. gov- US governmental agencies
19. mil- US military organization
20. net- major network support centers
21. org- organizations not mentioned above
    - int – International oragnisations.

8. What is meant by navigation?

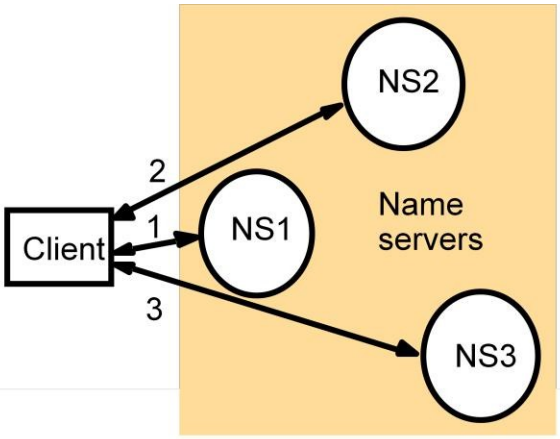
The process of locating naming data from than more than one name server in order to resolve a name is called navigation. The client name resolution software carries out navigation on behalf of the client. It communicates with name servers as necessary to resolve a name.

22. What is multicast navigation?

In multicast navigation, a client multicast the name to be resolved and required object type to the group of name servers. Only the server that holds the named attributes responds to the request. Unfortunately, however, if the name proves to be unbound, the request is greeted with silence.

23. What is iterative navigation?

One navigation model that DNS supports is known as iterative navigation. To resolve a name, a client present the name to the local name server, which attempts to resolve it. If the local name server has the name, it returns the result immediately. If it does not it will suggest another server that will ba able to help. Resolution proceeds at the new server, with further navigation as necessary until the name is located or is discovered to be unbounded.

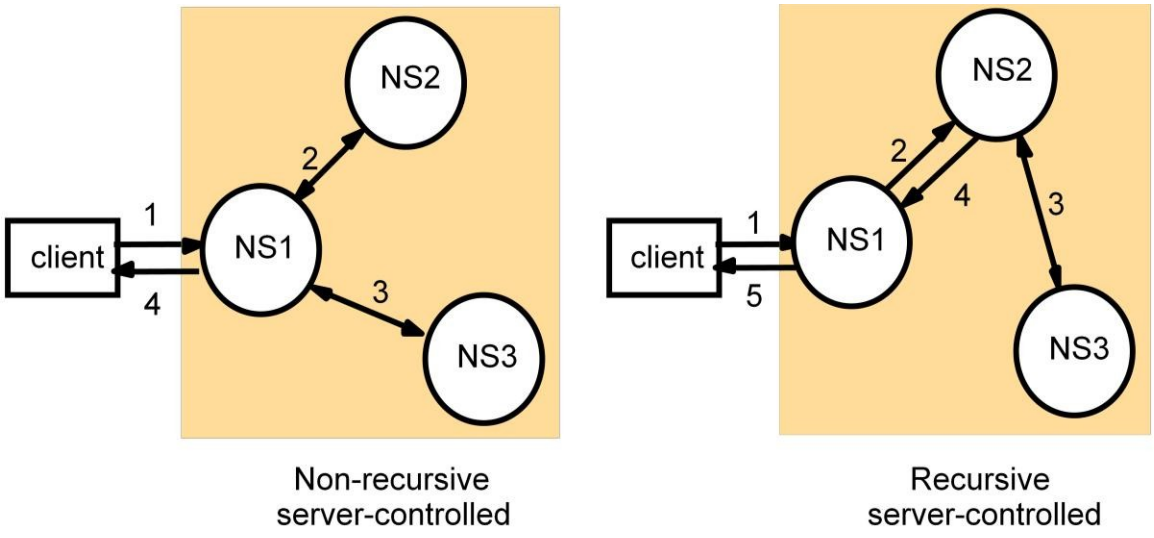


A client iteratively contacts name servers NS1–NS3 in order to resolve a name

24. What is meant by recursive and non recursive navigation?

In the non recursive and non recursive server controlled navigation. Under non recursive server controlled navigation ,any name server may be chosen by the client. This server communicates by multicast or iteratively with its peer in the style described above, as through it were a client. Under recursive server-controlled navigation ,the client once more contacts a single server. If this server doesnot store the name,the server contains a peer to storing a prefix of the

name,which in turn attempts to resolve it. This procedure continues recursively until the name resolved.



A name server NS1 communicates with other name servers on behalf of a client

25. Write the disadvantage of existing name service?

This original scheme was soon seen to suffer from three major shortcomings:

1. It did not scale to large numbers of computers.

2. Local organization wished to administer their own naming system.

3. A general name service was needed- not one that serves only for looking up computers address.

22. What is meant by DNS?

The domain name system is a name service design whose main naming database is used across the internet. It was devised principally by mockapetris and specified in RFC 1034 and RFC 1035. DNS replaced the original internet naming scheme in which all host names and address were held in a single central master file and downloaded by FTP to all computer that required them.

Domain Names: The DNS is designed for use in multiple implementations, each of which may have its own name space. In practice, however, only one is in widespread use, and that is one used for naming across the internet. The internet DNS name space is partitioned both organizationally and according to geography. The names are written with the highest-level domain on the right. The original top-level organizational domains in use across the internet were:

27. What is meant by zone?

The DNS naming data are divided into zones. A zone contains the following data:

β. Attributes data for names in a domain, less any subdomains administrates by lower level authorities.

χ. The names and address of at least two name servers that provide authoritative data for the zone. These are versions of zone data that can be relied upon as being reasonably up to date.

δ. The names of name servers that hold authoritative data for delegated sub domains; and

'glue' data giving the IP address of these servers.

ε. Zone-management parameters, such as those governing the catching and replication of zone data.

14. What is lookup operation?

15. List out options of NFS write operation?

Data in write operation received from client is stored in the memory cache at the server and written to disk before a reply is sent to the client. This is called writethrough caching. The client can be sure that is data stored persistently as soon as reply has been received.

Data in write operation is stored only in the memory cache. It will be written to disk when a commit operation is received for the relevant file. The client can be sure that the data is persistent stored only when a reply to a commit operation for the relevant file has been received. Standard NFS clients use this mode of operation, issuing a commit whenever a file that was open for writing is closed.

30. What is BIND implementation of DNS?

The Berkeley internet name domain is an implementation of the DNS for computers running UNIX. Client programs link in library software as the resolver. DNS name server computers run the named deamon. BIND allows for three categories of name server: primary server, secondary yserver, caching only servers. The named program implements just one of these types, according to the content of a configuration file. Caching only servers read in form a configuration file sufficient names and address of authoritative servers to resolve any name. Thereafter, they only store this data and data they learn by resolving names for clients.

ESSAY QUESTIONS

Explain about file system in Distributed System

A file system is responsible for the organization, storage, retrieval, naming, sharing, and protection of files. File systems provide directory services, which convert a file name (possibly a hierarchical one) into an internal identifier (e.g. inode, FAT index). They contain a representation of the file data itself and methods for accessing it (read/write). The file system is responsible for controlling access to the data and for performing low-level operations such as buffering frequently used data and issuing disk I/O requests.

A distributed file system is to present certain degrees of transparency to the user and the system: **Access transparency:** Clients are unaware that files are distributed and can access them in the same way as local files are accessed.

Location transparency: A consistent name space exists encompassing local as well as remote files. The name of a file does not give it location.

Concurrency transparency: All clients have the same view of the state of the file system. This means that if one process is modifying a file, any other processes on the same system or remote systems that are accessing the files will see the modifications in a coherent manner.

Failure transparency: The client and client programs should operate correctly after a server failure.

Heterogeneity: File service should be provided across different hardware and operating system platforms.

Scalability: The file system should work well in small environments (1 machine, a dozen machines) and also scale gracefully to huge ones (hundreds through tens of thousands of systems).

Replication transparency: To support scalability, we may wish to replicate files across multiple servers. Clients should be unaware of this.

Migration transparency: Files should be able to move around without the client's knowledge. Support fine-grained distribution of data: To optimize performance, we may wish to locate individual objects near the processes that use them.

Tolerance for network partitioning: The entire network or certain segments of it may be unavailable to a client during certain periods (e.g. disconnected operation of a laptop). The file system should be tolerant of this.

File system were originally developed for centralized computer systems and desktop computers.

File system was as an operating system facility providing a convenient programming interface to disk storage.

Distributed file systems support the sharing of information in the form of files and hardware resources.

With the advent of distributed object systems (CORBA, Java) and the web, the picture has become more complex.

Figure 1 provides an overview of types of storage system.

Figure 2 shows a typical layered module structure for the implementation of a non-distributed file system in a conventional operating system.

Directory module:	relates file names to file IDs
File module:	relates file IDs to particular files
Access control module:	checks permission for operation requested
File access module:	reads or writes file data or attributes
Block module:	accesses and allocates disk blocks
Device module:	disk I/O and buffering

File systems are responsible for the organization, storage, retrieval, naming, sharing and protection of files.

Files contain both data and attributes.

A typical attribute record structure is illustrated in Figure 3.

Figure 3. File attributes record structure

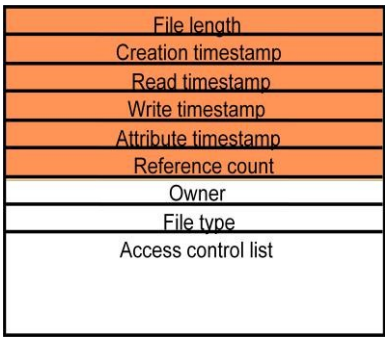


Figure 4 summarizes the main operations on files that are available to applications in UNIX systems.

Figure 4. UNIX file system operations

filedes = open(name, mode) *filedes = creat(name, mode)*	Opens an existing file with the given *name*. Creates a new file with the given *name*. Both operations deliver a file descriptor referencing the open file. The *mode* is *read*, *write* or both.
status = close(filedes)	Closes the open file *filedes*.
count = read(filedes, buffer, n)	Transfers *n* bytes from the file referenced by *filedes* to *buffer*.
count = write(filedes, buffer, n)	Transfers *n* bytes to the file referenced by *filedes* from buffer. Both operations deliver the number of bytes actually transferred and advance the read-write pointer.
pos = lseek(filedes, offset, whence)	Moves the read-write pointer to offset (relative or absolute, depending on *whence*).
status = unlink(name)	Removes the file *name* from the directory structure. If the file has no other names, it is deleted.
status = link(name1, name2)	Adds a new name (*name2*) for a file (*name1*).
status = stat(name, buffer)	Gets the file attributes for file *name* into *buffer*.

write about distributed system requirements

Distributed File system requirements

Related requirements in distributed file systems are:

Transparency

Concurrency

Replication

Heterogeneity

Fault tolerance

Consistency

Security

Efficiency

Write about types of file systems

File service architecture • This is an abstract architectural model that underpins both NFS and AFS. It is based upon a division of responsibilities between three modules – a client module that emulates a conventional file system interface for application programs, and server modules, that perform operations

for clients on directories and on files. The architecture is designed to enable a *stateless* implementation of the server module.

SUN NFS • Sun Microsystems's *Network File System* (NFS) has been widely adopted in industry and in academic environments since its introduction in 1985. The design and development of NFS were undertaken by staff at Sun Microsystems in 1984. Although several distributed file services had already been developed and used in universities and research laboratories, NFS was the first file service that was designed as a product. The design and implementation of NFS have achieved success both technically and commercially.

Andrew File System • Andrew is a distributed computing environment developed at Carnegie Mellon University (CMU) for use as a campus computing and information system. The design of the Andrew File System (henceforth abbreviated AFS) reflects an intention to support information sharing on a large scale by minimizing client-server communication. This is achieved by transferring whole files between server and client computers and caching them at clients until the server receives a more up-to-date version.

Write about file system architecture

An architecture that offers a clear separation of the main concerns in providing access to files is obtained by structuring the file service as three components:

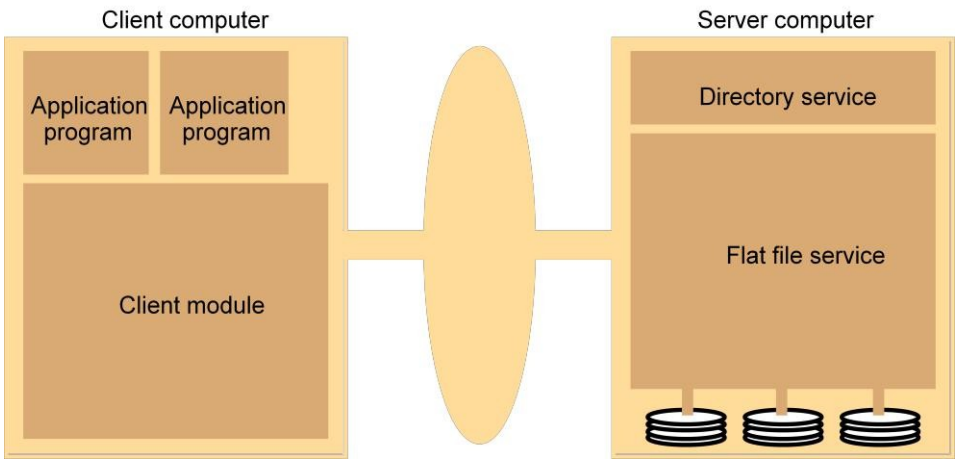
A flat file service

A directory service

A client module.

The relevant modules and their relationship is shown in Figure 5.

Figure 5. File service architecture



The Client module implements exported interfaces by flat file and directory services on server side.

Responsibilities of various modules can be defined as follows:

Flat file service:

Concerned with the implementation of operations on the contents of file. Unique File Identifiers (UFIDs) are used to refer to files in all requests for

flat file service operations. UFIDs are long sequences of bits chosen so that each file has a unique among all of the files in a distributed system.

Directory service:

Provides mapping between text names for the files and their UFIDs. Clients may obtain the UFID of a file by quoting its text name to directory service. Directory service supports functions needed generate directories, to add new files to directories.

Client module:

It runs on each computer and provides integrated service (flat file and directory) as a single API to application programs. For example, in UNIX hosts, a client module emulates the full set of Unix file operations.

It holds information about the network locations of flat-file and directory server processes; and achieve better performance through implementation of a cache of recently used file blocks at the client.

Flat file service interface:

Figure 6 contains a definition of the interface to a flat file service.

Read(FileId, i, n) -> Data -throws BadPosition	if *1≤i≤Length(File):* Reads a sequence of up to n items from a file starting at item *i* and returns it in *Data.*
Write(FileId, i, Data) -throws BadPosition	if *1≤i≤Length(File)+1*: Write a sequence of *Data* to a file, starting at item *i*, extending the file if necessary.
Create() -> FileId	Creates a new file of length0 and delivers a UFID for it.
Delete(FileId)	Removes the file from the file store.
GetAttributes(FileId) -> Attr	Returns the file attributes for the file.
SetAttributes(FileId, Attr)	Sets the file attributes (only those attributes that are not shaded in Figure 3.)

Access control

In distributed implementations, access rights checks have to be performed at the server because the server RPC interface is an otherwise unprotected point of access to files.

Directory service interface

Figure 7 contains a definition of the RPC interface to a directory service.

Figure 7. Directory service operations

Lookup(Dir, Name) -> FileId -throws NotFound	Locates the text name in the directory and returns the relevant UFID. If *Name* is not in the directory, throws an exception.
AddName(Dir, Name, File) -throws NameDuplicate	If *Name* is not in the directory, adds(*Name,File*) to the directory and updates the file's attribute record. If *Name* is already in the directory: throws an exception.
UnName(Dir, Name)	If *Name* is in the directory, the entry containing *Name* is removed from the directory. If *Name* is not in the directory: throws an exception.
GetNames(Dir, Pattern) -> NameSeq	Returns all the text names in the directory that match the regular expression *Pattern*.

Hierarchic file system

A hierarchic file system such as the one that UNIX provides consists of a number of directories arranged in a tree structure.

File Group

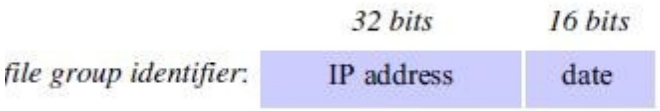
A file group is a collection of files that can be located on any server or moved between servers while maintaining the same names.

– A similar construct is used in a UNIX file system.

– It helps with distributing the load of file serving between several servers.

– File groups have identifiers which are unique throughout the system (and hence for an open system, they must be globally unique).

To construct globally unique ID we use some unique attribute of the machine on which it is created. E.g: IP number, even though the file group may move subsequently.



NFS (Network File System)

Developed by Sun Microsystems (in 1985)

Most popular, open, and widely used.

NFS protocol standardized through IETF (RFC 1813)

AFS (Andrew File System)

Developed by Carnegie Mellon University as part of Andrew distributed computing environments (in 1986)

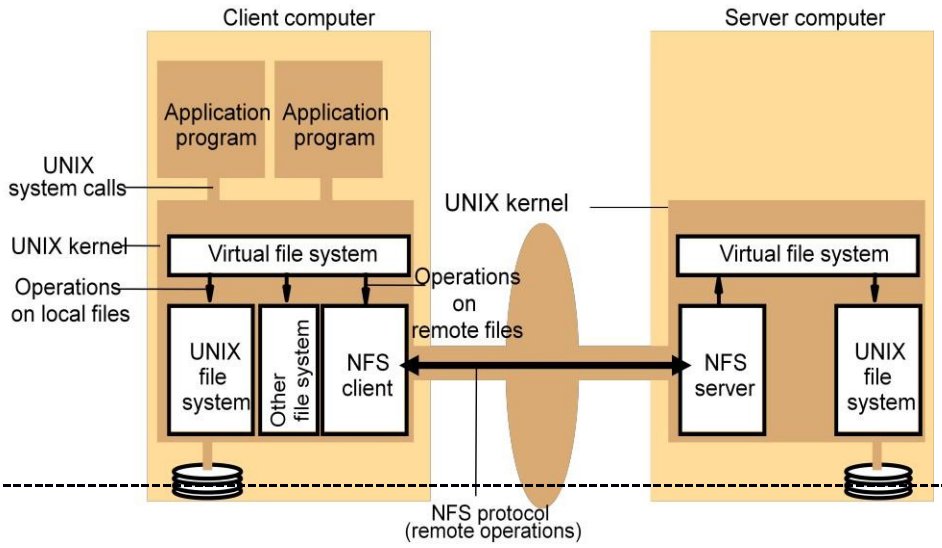
A research project to create campus wide file system.

Public domain implementation is available on Linux (LinuxAFS)

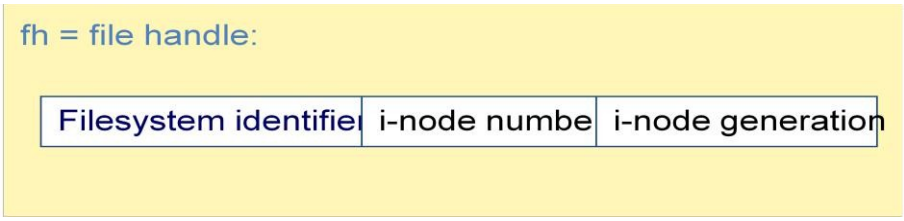
It was adopted as a basis for the DCE/DFS file system in the Open Software Foundation (OSF, www.opengroup.org) DEC (Distributed Computing Environment

Write about NFS

Figure 8 shows the architecture of Sun NFS



The file identifiers used in NFS are called file handles.



A simplified representation of the RPC interface provided by NFS version 3 servers is shown in Figure 9.

Figure 9. NFS server operations (NFS Version 3 protocol, simplified)

- *read(fh, offset, count) -> attr, data*
- *write(fh, offset, count, data) -> attr*
- *create(dirfh, name, attr) -> newfh, attr*
- *remove(dirfh, name) status*
- *getattr(fh) -> attr*
- *setattr(fh, attr) -> attr*
- *lookup(dirfh, name) -> fh, attr*
- *rename(dirfh, name, todirfh, toname)*
- *link(newdirfh, newname, dirfh, name)*
- *readdir(dirfh, cookie, count) -> entries*
- *symlink(newdirfh, newname, string) -> status*
- *readlink(fh) -> string*
- *mkdir(dirfh, name, attr) -> newfh, attr*
- *rmdir(dirfh, name) -> status*
- *statfs(fh) -> fsstats*

The NFS server is stateless server, so the user's identity and access rights must be checked by the server on each request.

In the local file system they are checked only on the file's access permission attribute.

Every client request is accompanied by the userID and groupID

It is not shown in the Figure 8.9 because they are inserted by the RPC system.

Kerberos has been integrated with NFS to provide a stronger and more comprehensive security solution.

Mount service

Mount operation:

mount(remotehost, remotedirectory, localdirectory)

server maintains a table of clients who have mounted filesystems at that server.

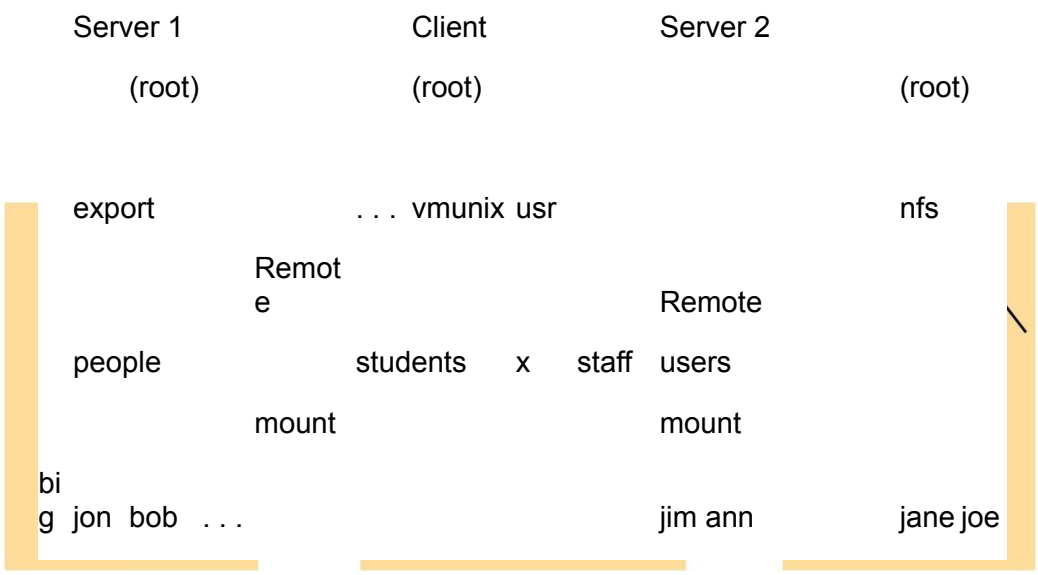
Each client maintains a table of mounted file systems holding:

IP address, port number, file handle>

Remote file systems may be hard-mounted or soft-mounted in a client computer.

Figure 10 illustrates a Client with two remotely mounted file stores.

Figure 10. Local and remote file systems accessible on an NFS client



Automounter

The automounter was added to the UNIX implementation of NFS in order to mount a remote directory dynamically whenever an 'empty' mount point is referenced by a client.

Automounter has a table of mount points with a reference to one or more NFS servers listed against each.

it sends a probe message to each candidate server and then uses the mount service to mount the file system at the first server to respond.

Automounter keeps the mount table small.

Automounter Provides a simple form of replication for read-only file systems.

E.g. if there are several servers with identical copies of /usr/lib then each server will have a chance of being mounted at some clients.

Server caching

Similar to UNIX file caching for local files:

pages (blocks) from disk are held in a main memory buffer cache until the space is required for newer pages. Read-ahead and delayed-write optimizations.

For local files, writes are deferred to next sync event (30 second intervals).

Works well in local context, where files are always accessed through the local cache, but in the remote case it doesn't offer necessary synchronization guarantees to clients.

NFS v3 servers offers two strategies for updating the disk:

Write-through - altered pages are written to disk as soon as they are received at the server. When a write() RPC returns, the NFS client knows that the page is on the disk.

Delayed commit - pages are held only in the cache until a commit() call is received for the relevant file. This is the default mode used by NFS v3 clients. A commit() is issued by the client whenever a file is closed.

Client caching

Server caching does nothing to reduce RPC traffic between client and server

further optimization is essential to reduce server load in large networks.

NFS client module caches the results of read, write, getattr, lookup and readdir operations

synchronization of file contents (one-copy semantics) is not guaranteed when two or more clients are sharing the same file.

Timestamp-based validity check

It reduces inconsistency, but doesn't eliminate it.

It is used for validity condition for cache entries at the client:

(T - Tc < t) v (Tmclient = Tmserver)

t	freshness guarantee
Tc	time when cache entry was last validated
Tm	time when block was last updated at server
T	current time

it is configurable (per file) but is typically set to 3 seconds for files and 30 secs. for directories.

it remains difficult to write distributed applications that share files with NFS.

Other NFS optimizations

Sun RPC runs over UDP by default (can use TCP if required).

Uses UNIX BSD Fast File System with 8-kbyte blocks.

reads() and writes() can be of any size (negotiated between client and server).

The guaranteed freshness interval t is set adaptively for individual files to reduce getattr() calls needed to update Tm.

File attribute information (including Tm) is piggybacked in replies to all file requests.

NFS performance

Early measurements (1987) established that:

Write() operations are responsible for only 5% of server calls in typical UNIX environments.

hence write-through at server is acceptable.

Lookup() accounts for 50% of operations -due to step-by-step pathname resolution necessitated by the naming and mounting semantics.

More recent measurements (1993) show high performance.

see www.spec.org for more recent measurements.

NFS summary

NFS is an excellent example of a simple, robust, high-performance distributed service.

Achievement of transparencies are other goals of NFS:

Access transparency:

The API is the UNIX system call interface for both local and remote files.

Location transparency:

Naming of filesystems is controlled by client mount operations, but transparency can be ensured by an appropriate system configuration.

Mobility transparency:

Hardly achieved; relocation of files is not possible, relocation of filesystems is possible, but requires updates to client configurations.

Scalability transparency:

File systems (file groups) may be subdivided and allocated to separate servers.

Replication transparency:

– Limited to read-only file systems; for writable files, the SUN Network Information Service (NIS) runs over NFS and is used to replicate essential system files.

Hardware and software operating system heterogeneity:

– NFS has been implemented for almost every known operating system and hardware platform and is supported by a variety of filling systems.

Fault tolerance:

– Limited but effective; service is suspended if a server fails. Recovery from failures is aided by the simple stateless design.

Consistency:

– It provides a close approximation to one-copy semantics and meets the needs of the vast majority of applications.

– But the use of file sharing via NFS for communication or close coordination between processes on different computers cannot be recommended.

Security:

– Recent developments include the option to use a secure RPC implementation for authentication and the privacy and security of the data transmitted with read and write operations.

– Efficiency:

NFS protocols can be implemented for use in situations that generate very heavy loads.

write about AFS

AFS differs markedly from NFS in its design and implementation. The differences are primarily attributable to the identification of scalability as the most important design goal. AFS is designed to perform well with larger numbers of active users than other distributed file systems. The key strategy for achieving scalability is the caching of whole files in client nodes. AFS has two unusual design characteristics:

Whole-file serving: The entire contents of directories and files are transmitted to client computers by AFS servers (in AFS-3, files larger than 64 kbytes are transferred in 64-kbyte chunks).

Whole file caching: Once a copy of a file or a chunk has been transferred to a client computer it is stored in a cache on the local disk. The cache contains several hundred of the files most recently used on that computer. The cache is permanent, surviving reboots of the client computer. Local copies of files are used to satisfy clients' *open* requests in preference to remote copies whenever possible.

Like NFS, AFS provides transparent access to remote shared files for UNIX programs running on workstations.

AFS is implemented as two software components that exist at UNIX processes called Vice and Venus**.**

Scenario • Here is a simple scenario illustrating the operation of AFS:

When a user process in a client computer issues an *open* system call for a file in the shared

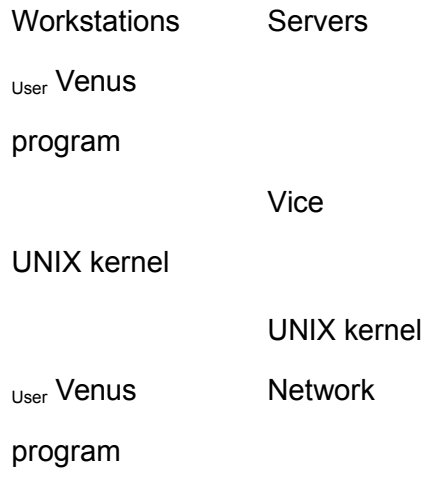
file space and there is not a current copy of the file in the local cache, the server holding the file is located and is sent a request for a copy of the file.

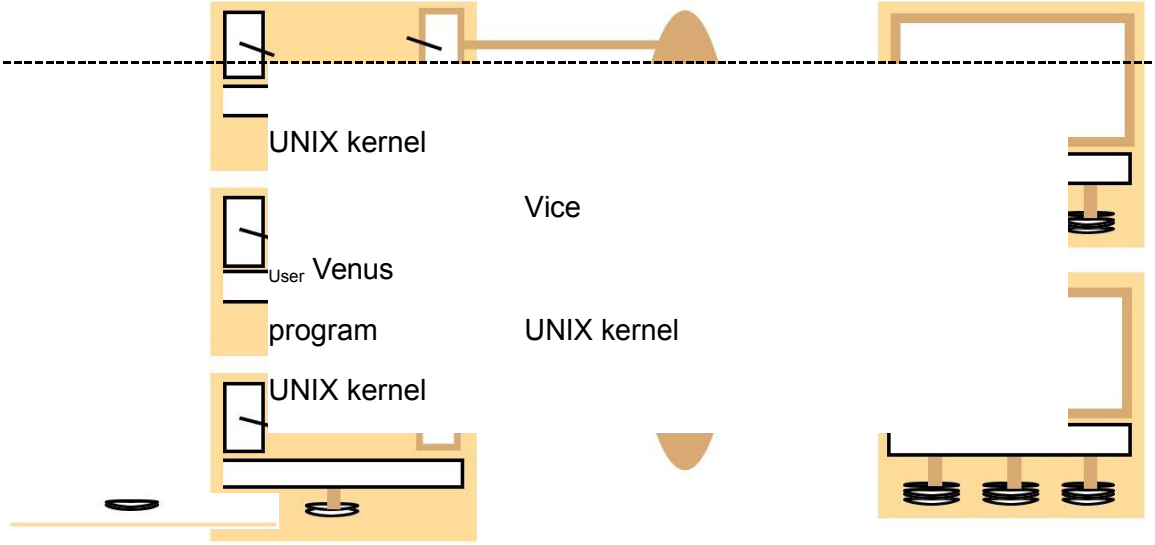
The copy is stored in the local UNIX file system in the client computer. The copy is then *open*ed and the resulting UNIX file descriptor is returned to the client.

Subsequent *read*, *write* and other operations on the file by processes in the client computer are applied to the local copy.

When the process in the client issues a *close* system call, if the local copy has been updated its contents are sent back to the server. The server updates the file contents and the timestamps on the file. The copy on the client's local disk is retained in case it is needed again by a user-level process on the same workstation.

Figure 11. Distribution of processes in the Andrew File System





The files available to user processes running on workstations are either local or shared.

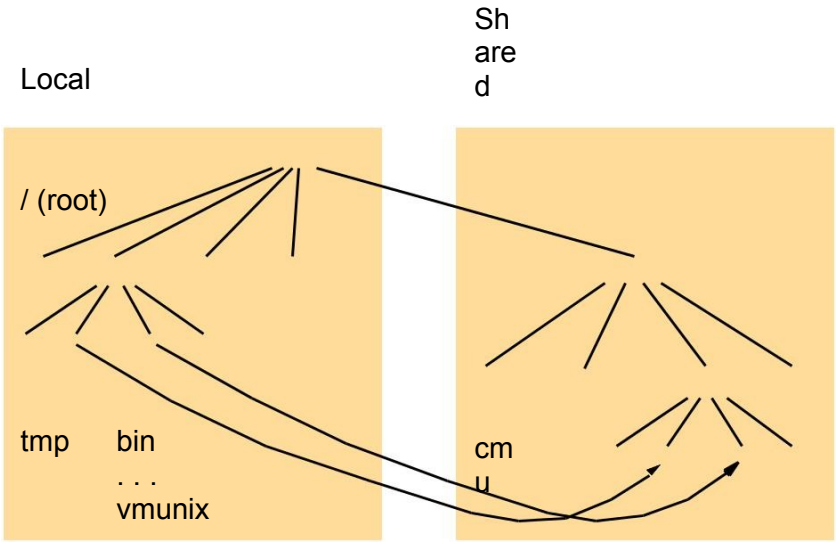
Local files are handled as normal UNIX files.

They are stored on the workstation's disk and are available only to local user processes.

Shared files are stored on servers, and copies of them are cached on the local disks of workstations.

The name space seen by user processes is illustrated in Figure 12.

Figure 12. File name space seen by clients of AFS



Symbolic

links

The UNIX kernel in each workstation and server is a modified version of BSD UNIX.

The modifications are designed to intercept open, close and some other file system calls when they refer to files in the shared name space and pass them to the Venus process in the client computer. (Figure 13)

Figure 13. System call interception in AFS

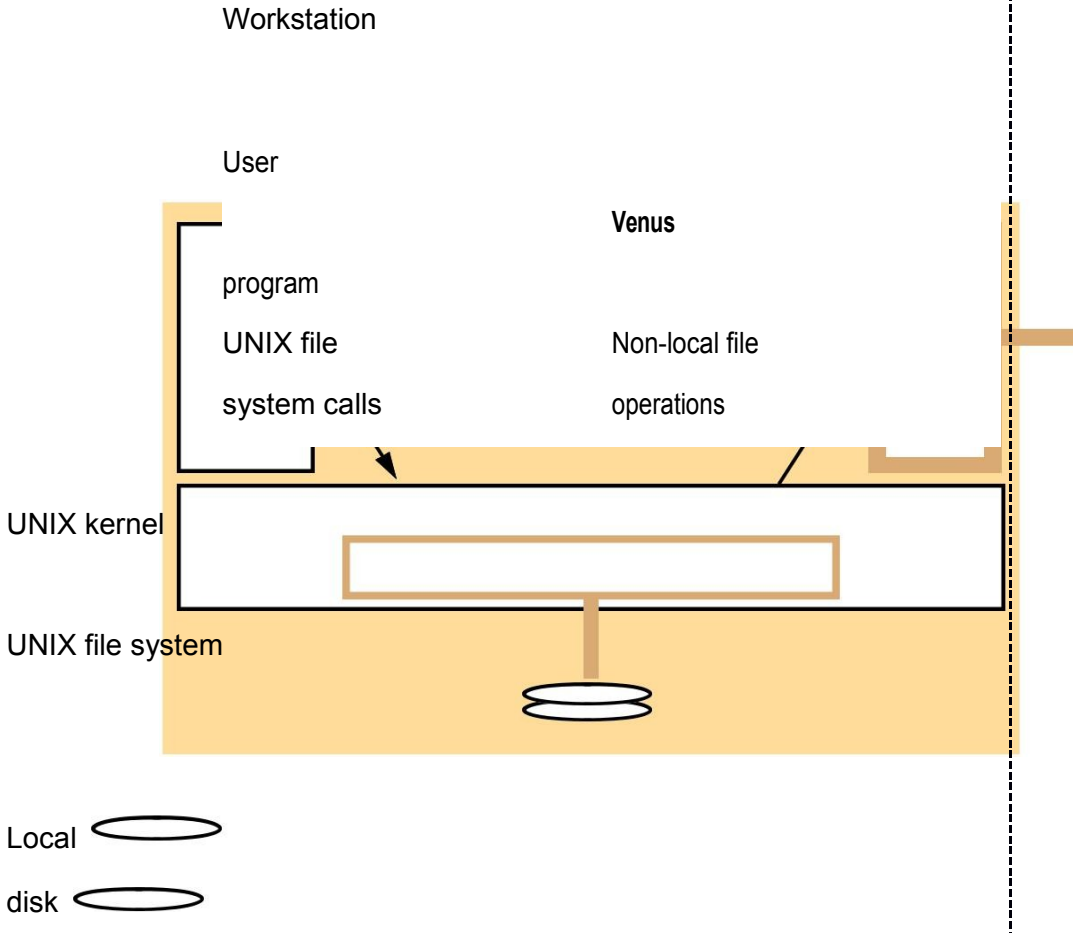


Figure 14 describes the actions taken by Vice, Venus and the UNIX kernel when a user process issues system calls.

Figure 14. implementation of file system calls in AFS

User process	*UNIX kernel*	*Venus*	*Net*	*Vice*
open(FileName, mode)	If *FileName* refers to a file in shared file space, pass the request to Venus.	Check list of files in local cache. If not present or there is no valid *callback promise*, send a request for the file to the Vice server that is custodian of the volume containing the file.		Transfer a copy of the file and a *callback promise*to the workstation . Log the callback promise.
	Open the local file and return the file descriptor to the application .	Place the copy of the file in the local file system, enter its local name in the local cache list and return the local name to UNIX.		
read(FileDescriptor, Buffer, length)	Perform a normal UNIX read operation on the local copy .			
write(FileDescriptor,	Perform a normal			

Buffer, length)	UNIX write operation on the local copy .		
close(FileDescriptor)	Close the local copy and notify Venus that the file has been closed .	If the local copy has been changed, send a copy to the Vice server that is the custodian of the file.	Replace the file contents and send a *callback* to all other clients holding*ca llba ck* *promises*on the file.

Figure 15 shows the RPC calls provided by AFS servers for operations on files.

Figure 15. The main components of the Vice service interface

Fetch(fid) -> attr, data	Returns the attributes (status) and, optionally, the contents of file identified by the *fid* and records a callback promise on it.
Store(fid, attr, data)	Updates the attributes and (optionally) the contents of a specified file.
Create() -> fid	Creates a new file and records a callback promise on it.
Remove(fid)	Deletes the specified file.
SetLock(fid, mode)	Sets a lock on the specified file or directory. The mode of the lock may be shared or exclusive. Locks that are not removed expire after 30 minutes.
ReleaseLock(fid)	Unlocks the specified file or directory.
RemoveCallback(fid)	Informs server that a Venus process has flushed a file from its cache.
BreakCallback(fid)	This call is made by a Vice server to a Venus process. It cancels the callback promise on the relevant file.

Write about Naming system?

Introduction

In a distributed system, names are used to refer to a wide variety of resources such as:

Computers, services, remote objects, and files, as well as users.

Naming is fundamental issue in DS design as it facilitates communication and resource sharing.

A name in the form of URL is needed to access a specific web page.

Processes cannot share particular resources managed by a computer system unless they can name them consistently

Users cannot communicate within one another via a DS unless they can name one another, with email address.

Names are not the only useful means of identification: descriptive attributes are another.

Naming Services

How do Naming Services facilitate communication and resource sharing?

– An URL facilitates the localization of a resource exposed on the Web.

e.g., abc.net.au means it is likely to be an Australian entity?

– A consistent and uniform naming helps processes in a distributed system to interoperate and manage resources.

e.g., commercials use .com; non-profit organizations use .org

– Users refers to each other by means of their names (i.e. email) rather than their system ids

– Naming Services are not only useful to locate resources but also to gather additional information about them such as attributes

In a Distributed System, a Naming Service is a specific service whose aim is to provide a consistent and uniform naming of resources, thus allowing other programs or services to localize them and obtain the required metadata for interacting with them.

Key benefits

– Resource localization

– Uniform naming

– Device independent address (e.g., you can move domain name/web site from one server to another server seamlessly).

The role of names and name services

Resources are accessed using *identifier* or *reference*

- An identifier can be stored in variables and retrieved from tables quickly

– Identifier includes or can be transformed to an address for an object

E.g. NFS file handle, Corba remote object reference

– A *name* is human-readable value (usually a string) that can be *resolved* to an identifier or address

Internet domain name, file pathname, process number

E.g ./etc/passwd, http://www.cdk3.net/

For many purposes, names are preferable to identifiers

– because the binding of the named resource to a physical location is deferred and can be changed

– because they are more meaningful to users

Resource names are *resolved* by name services

– to give identifiers and other useful attributes

Requirements for naming system?

Allow simple but meaningful names to be used

Potentially infinite number of names

Structured

– to allow similar subnames without clashes

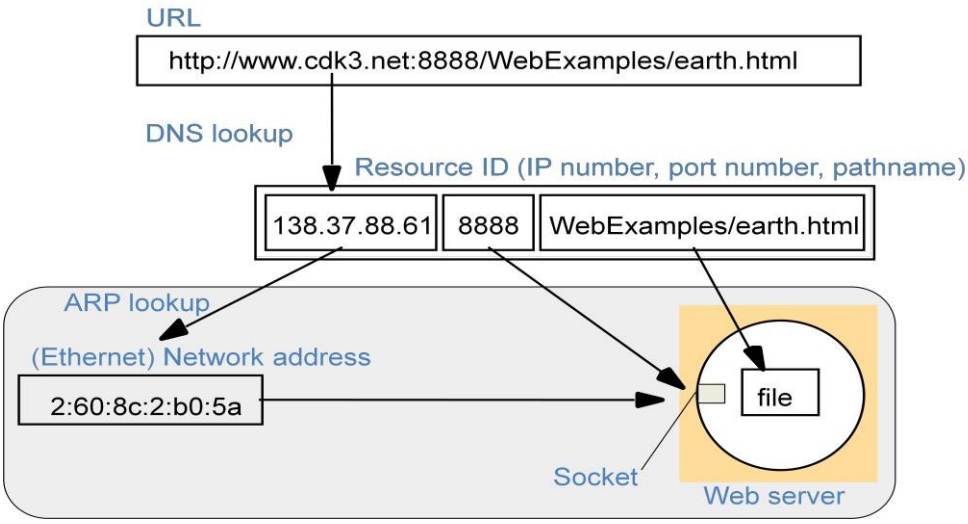
– to group related names

Allow re-structuring of name trees

– for some types of change, old programs should continue to work

Management of trust

Composed naming domains used to access a resource from a URL



A key attribute of an entity that is usually relevant in a distributed system is its address. For example:

The DNS maps domain names to the attributes of a host computer: its IP address, the type of entry (for example, a reference to a mail server or another host) and, for example, the length of time the host's entry will remain valid.

The X500 directory service can be used to map a person's name onto attributes including their email address and telephone number.

The CORBA Naming Service maps the name of a remote object onto its remote object reference, whereas the Trading Service maps the name of a remote object onto its remote object reference, together with an arbitrary number of attributes describing the object in terms understandable by human users.

Name Services and the Domain Name System

A name service stores a collection of one or more naming contexts, sets of bindings between textual names and attributes for objects such as computers, services, and users.

The major operation that a name service supports is to resolve names.

Uniform Resource Identifiers

Uniform Resource Identifiers (URIs) came about from the need to identify resources on the Web, and other Internet resources such as electronic mailboxes. An important goal was to identify resources in a coherent way, so that they could all be processed by common software such as browsers. URIs are 'uniform' in that their syntax incorporates that of indefinitely many individual types of resource identifiers (that is, URI schemes), and there are procedures for managing the global namespace of schemes. The advantage of uniformity is that it eases the process of introducing new types of identifier, as well as using existing types of identifier in new contexts, without disrupting existing usage.

Uniform Resource Locators: Some URIs contain information that can be used to locate and access a resource; others are pure resource names. The familiar term Uniform Resource Locator (URL) is often used for URIs that provide location information and specify the method for accessing the resource.

Uniform Resource Names: Uniform Resource Names (URNs) are URIs that are used as pure resource names rather than locators. For example, the URI:

mid:0E4FC272-5C02-11D9-B115-000A95B55BC8@hpl.hp.com

Write about distributed Navigation?

Navigation is the act of chaining multiple Naming Services in order to resolve a single name to the corresponding resource.

Namespaces allows for structure in names.

URLs provide a default structure that decompose the location of a resource in

– protocol used for retrieval

– internet end point of the service exposing the resource

– service specific path

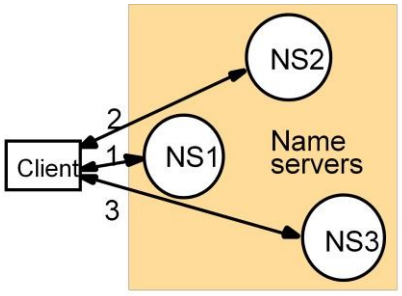
This decomposition facilitates the resolution of the name into the corresponding resource

Moreover, structured namespaces allows for iterative navigation…

Iterative navigation

Reason for NFS iterative name resolution

This is because the file service may encounter a symbolic link (i.e. an *alias*) when resolving a name. A symbolic link must be interpreted in the client's file system name space because it may point to a file in a directory stored at another server. The client computer must determine which server this is, because only the client knows its mount points



A client iteratively contacts name servers NS1–NS3 in order to resolve a name

Server controlled navigation

In an alternative model, name server coordinates naming resolution and returns the results to the client. It can be:

– Recursive:

it is performed by the naming server

the server becomes like a client for the next server

this is necessary in case of client connectivity constraints

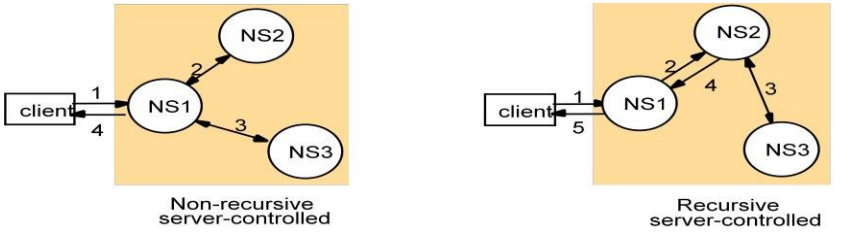
– Non recursive:

it is performed by the client or the first server

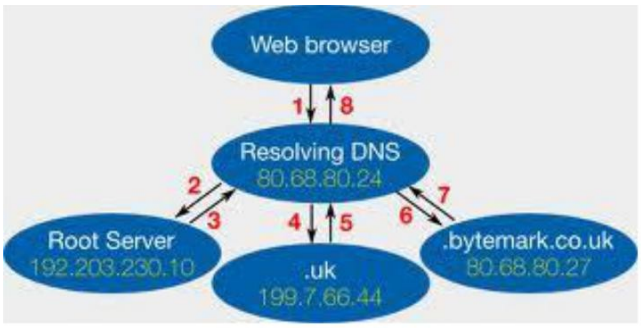
the server bounces back the next hop to its client

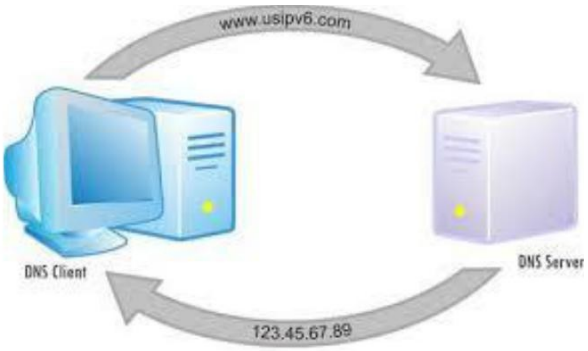
Non-recursive and recursive server-controlled navigation

DNS offers recursive navigation as an option, but iterative is the standard technique. Recursive navigation must be used in domains that limit client access to their DNS information for security reasons.



A name server NS1 communicates with other name servers on behalf of a client





The Domain Name System is a name service design whose main naming database is used across the Internet.

This original scheme was soon seen to suffer from three major shortcomings:

It did not scale to large numbers of computers.

Local organizations wished to administer their own naming systems.

A general name service was needed – not one that serves only for looking up computer addresses.

Explain about DNS server functions and configuration

Main function is to resolve domain names for computers, i.e. to get their IP addresses

– caches the results of previous searches until they pass their 'time to live'

Other functions:

– get *mail host* for a domain

– reverse resolution - get domain name from IP address

– Host information - type of hardware and OS

– Well-known services - a list of well-known services offered by a host

– Other attributes can be included (optional)

DNS resource records

The DNS architecture allows for recursive navigation as well as iterative navigation. The resolver specifies which type of navigation is required when contacting a name server. However, name servers are not bound to implement recursive navigation. As was pointed out above, recursive navigation may tie up server threads, meaning that other requests might be delayed.

Record type	*Meaning*	*Main contents*
A	A computer address	IP number
NS	An authoritative name server	Domain name for server
CNAME	The canonical name for an alias	Domain name for alias
SOA	Marks the start of data for a zone	Parameters governing the zone
WKS	A well-known service description	List of service names and protocols
PTR	Domain name pointer (reverse lookups)	Domain name
HINFO	Host information	Machine architecture and operating system
MX	Mail exchange	List of <*preference, host*> pairs
TXT	Text string	Arbitrary text

The data for a zone starts with an *SOA*-type record, which contains the zone parameters that specify, for example, the version number and how often secondaries should refresh their copies. This is followed by a list of records of type *NS* specifying the name servers for the domain and a list of records of type *MX* giving the domain names of mail hosts, each prefixed by a number expressing its preference. For example, part of the database for the domain *dcs.qmul.ac.uk* at one point is shown in the following figure where the time to live *1D* means 1 day.

DNS zone data records

domain name	*time to live*	*class*	*type*	*value*
dcs.qmul.ac.uk	*1D*	*IN*	*NS*	*dns0*
dcs.qmul.ac.uk	*1D*	*IN*	*NS*	*dns1*
dcs.qmul.ac.uk	*1D*	*IN*	*MX*	*1 mail1.qmul.ac.uk*
dcs.qmul.ac.uk	*1D*	*IN*	*MX*	*2 mail2.qmul.ac.uk*

domain name	*time to live*	*class*	*type*	*value*
www	*1D*	*IN*	*CNAME*	traffic
traffic	*1D*	*IN*	*A*	138.37.95.150

If the domain has any subdomains, there will be further records of type *NS* specifying their name servers, which will also have individual *A* entries. For example, at one point the database for *qmul.ac.uk* contained the following records for the name servers in its subdomain *dcs.qmul.ac.uk*:

domain name	*time to live*	*class*	*type*	*value*
dcs	*1D*	*IN*	*NS*	*dns0.dcs*
dns0.dcs	*1D*	*IN*	*A*	*138.37.88.249*
dcs	*1D*	*IN*	*NS*	*dns1.dcs*
dns1.dcs	*1D*	*IN*	*A*	*138.37.94.248*

DNS issues

Name tables change infrequently, but when they do, caching can result in the delivery of stale data.

– Clients are responsible for detecting this and recovering

Its design makes changes to the structure of the name space difficult. For example:

– merging previously separate domain trees under a new root

– moving subtrees to a different part of the structure (e.g. if Scotland became a separate country, its domains should all be moved to a new country-level domain.)

Explain about directory service?

Directory service: 'yellow pages' for the resources in a network

Retrieves the set of names that satisfy a given description

– e.g. X.500, LDAP, MS Active Directory Services

(*DNS holds some descriptive data, but:*

the data is very incomplete

DNS isn't organised to search it)

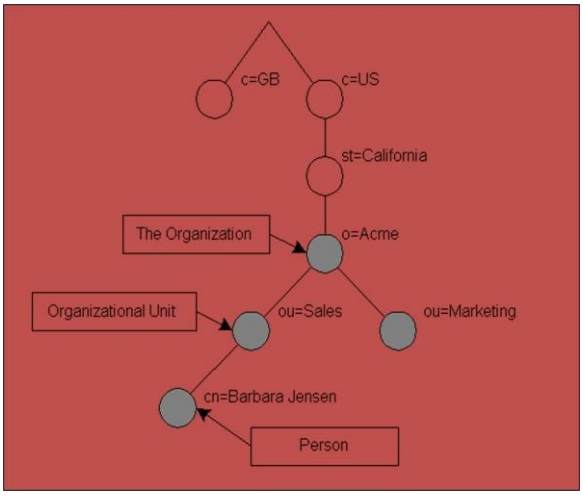
Discovery service:- a directory service that also:

is automatically updated as the network configuration changes meets the needs of clients in spontaneous networks (Section 2.2.3)

discovers services required by a client (who may be mobile) within the current *scope*, for example, to find the most suitable printing service for image files after arriving at a hotel.

Examples of- discovery services: Jini discovery service, the 'service location

– protocol', the 'simple service discovery protocol' (part of UPnP), the 'secure discovery service'.



The name services store collections of <*name, attribute*> pairs, and how the attributes are looked up from a name. It is natural to consider the dual of this arrangement, in which *attributes* are used as values to be looked up. In these services, textual names can be considered to be just another attribute. Sometimes users wish to find a particular person or resource, but they do not know its name, only some of its other attributes.

For example, a user may ask: 'What is the name of the user with telephone number 020-555 9980?' Likewise, sometimes users require a service, but they are not concerned with what system entity supplies that service, as long as the service is conveniently accessible.

For example, a user might ask, 'Which computers in this building are Macintoshes running the Mac OS X operating system?' or 'Where can I print a high-resolution colour image?'

A service that stores collections of bindings between names and attributes and that looks up entries that match attribute-based specifications is called a *directory service*.

Examples are Microsoft's Active Directory Services, X.500 and its cousin LDAP, Univers and

Profile.

Directory services are sometimes called *yellow pages services*, and conventional name services are correspondingly called *white pages services*, in an analogy with the traditional types of telephone directory. Directory services are also sometimes known as *attribute-based name services*.

A directory service returns the sets of attributes of any objects found to match some specified attributes. So, for example, the request 'TelephoneNumber = 020 5559980' might return {'Name = John Smith', 'TelephoneNumber = 020 555 9980', 'emailAddress = john@dcs.gormenghast.ac.uk', ...}.

The client may specify that only a subset of the attributes is of interest – for example, just the email addresses of matching objects. X.500 and some other directory services also allow objects to be looked up by conventional hierarchic textual names. The Universal Directory and Discovery Service (UDDI), which was presented in Section 9.4, provides both white pages and yellow pages services to provide information about organizations and the web services they offer.

UDDI aside, the term *discovery service* normally denotes the special case of a directory service for services provided by devices in a spontaneous networking environment. As Section 1.3.2 described, devices in spontaneous networks are liable to connect and disconnect unpredictably. One core difference between a discovery service and other directory services is that the address of a directory service is normally well known and preconfigured in clients, whereas a device entering a spontaneous networking environment has to resort to multicast navigation, at least the first time it accesses the local discovery service.

Attributes are clearly more powerful than names as designators of objects: programs can be written to select objects according to precise attribute specifications where names might not be known. Another advantage of attributes is that they do not expose the structure of organizations

to the outside world, as do organizationally partitioned names. However, the relative simplicity of use of textual names makes them unlikely to be replaced by attribute-based naming in many applications.

Discovery service

A database of services with lookup based on service description or type, location and other criteria, E.g.

Find a printing service in this hotel compatible with a Nikon camera

Send the video from my camera to the digital TV in my room.

Automatic registration of new services

Automatic connection of guest's clients to the discovery service

Write about Global Name Service

Designed and implemented by Lampson and colleagues at the DEC Systems Research Center (1986)

Provide facilities for resource location, email addressing and authentication

When the naming database grows from small to large scale, the structure of name space may changethe service should accommodate it

The GNS manages a naming database that is composed of a tree of directories holding names and values. Directories are named by multi-part pathnames referred to a root, or relative to a working directory, much like file names in a UNIX file system. Each directory is also assigned an integer, which serves as a unique *directory identifier* (DI). A directory contains a list of names and references. The values stored at the leaves of the directory tree are organized into *value trees*, so that the attributes associated with names can be structured values.

Names in the GNS have two parts: <*directory name*, *value name*>. The first part identifies a directory; the second refers to a value tree, or some portion of a value tree.

GNS Structure

Tree of directories holding names and values

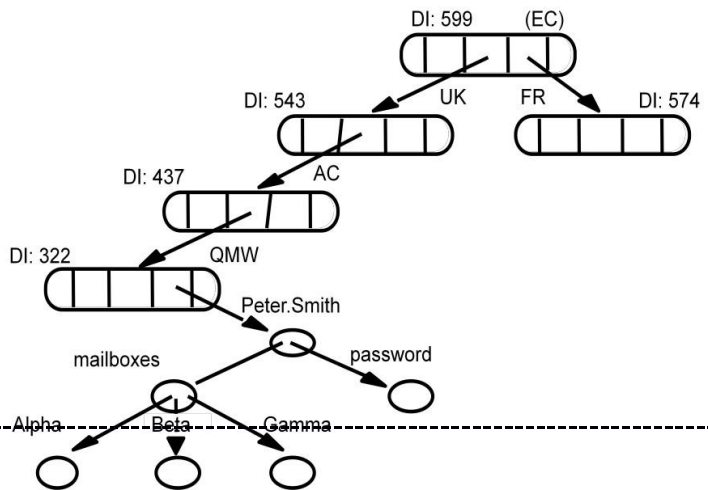
Muti-part pathnames refer to the root or relative working directory (like Unix file system)

Unique Directory Identifier (DI)

A directory contains list of names and references

Leaves of tree contain value trees (structured values)

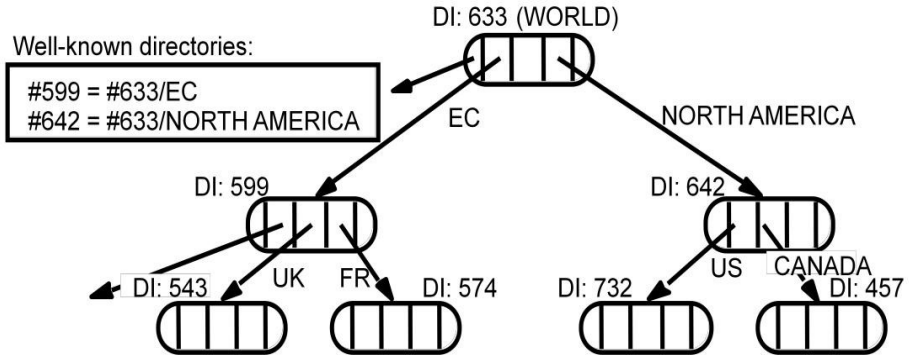
GNS directory tree and value tree



Accommodating changes

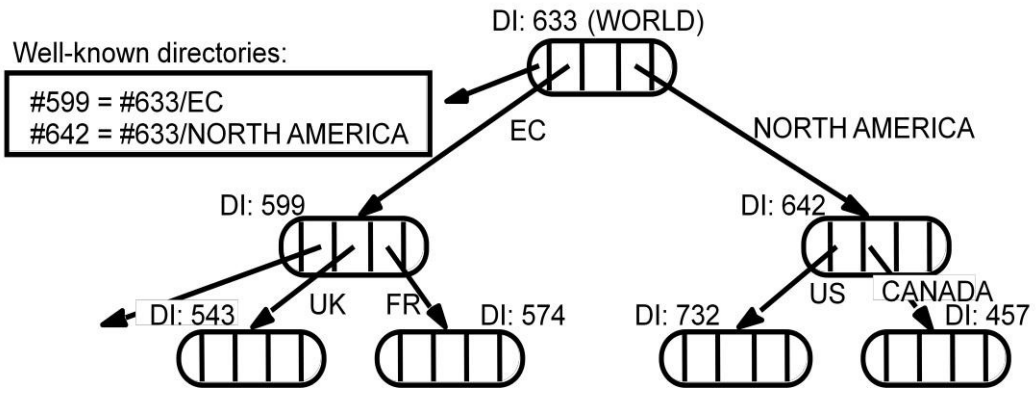
How to integrate naming trees of two previously separate GNS services

But what is for '/UK/AC/QMV, Peter.Smith' ?



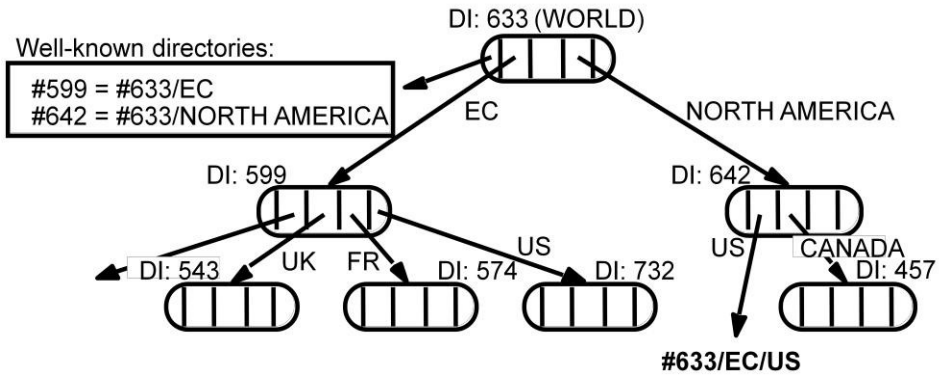
Using DI to solve changes

- Using the name '#599/UK/AC/QMV, Peter.Smith'



Restructuring of database

- Using symbolic links



Explain about X.500 Directory Service?

X.500 is a directory service used in the same way as a conventional name service, but it is primarily used to satisfy descriptive queries and is designed to discover the names and attributes of other users or system resources. Users may have a variety of requirements for searching and browsing in a directory of network users, organizations and system resources to obtain information about the entities that the directory contains. The uses for such a service are likely to be quite diverse. They range from enquiries that are directly analogous to the use of telephone directories, such as a simple 'white pages' access to obtain a user's electronic mail address or a 'yellow pages' query aimed, for example, at obtaining the names and telephone numbers of garages specializing in the repair of a particular make of car, to the use of the directory to access personal details such as job roles, dietary habits or even photographic images of the individuals.

Standard of ITU and ISO organizations

Organized in a tree structure with name nodes as in the case of other name servers

A wide range of attributes are stored in each node

Directory Information Tree (DIT)

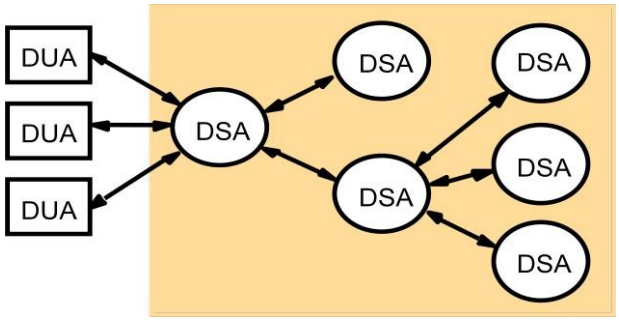
Directory Information Base (DIB)

X.500 service architecture

The data stored in X.500 servers is organized in a tree structure with named nodes, as in the case of the other name servers discussed in this chapter, but in X.500 a wide range of attributes are stored at each node in the tree, and access is possible not just by name but also by searching for entries with any required combination of attributes. The X.500 name tree is called the *Directory Information Tree* (DIT), and the entire directory structure including the data associated with the nodes, is called the *Directory Information Base* (DIB). There is intended to be a single integrated DIB containing information provided by organizations throughout the world, with portions of the DIB located in individual X.500 servers. Typically, a medium-sized or large organization would provide at least one server. Clients access the directory by establishing a connection to a server and issuing access requests. Clients can contact any server with an enquiry. If the data required are not in the segment of the DIB held by the contacted server, it will either invoke other servers to resolve the query or redirect the client to another server.

Directory Server Agent (DSA)

Directory User Agent (DUA)



In the terminology of the X.500 standard, servers are *Directory Service Agents* (DSAs), and their clients are termed *Directory User Agents* (DUAs). Each entry in the DIB consists of a name and a set of attributes. As in other name servers, the full name of an entry corresponds to a path through the DIT from the root of the tree to the entry. In addition to full or *absolute* names, a DUA can establish a context, which includes a base node, and then use shorter relative names that give the path from the base node to the named entry.

An X.500 DIB Entry

info Alice Flintstone, Departmental Staff, Department of Computer Science, University of Gormenghast, GB	
commonName Alice.L.Flintstone Alice.Flintstone Alice Flintstone A. Flintstone	*uid* alf *mail* alf@dcs.gormenghast.ac.uk Alice.Flintstone@dcs.gormenghast.ac.uk
surname Flintstone	*roomNumber* Z42
telephoneNumber +44 986 33 4604	*userClass* Research Fellow

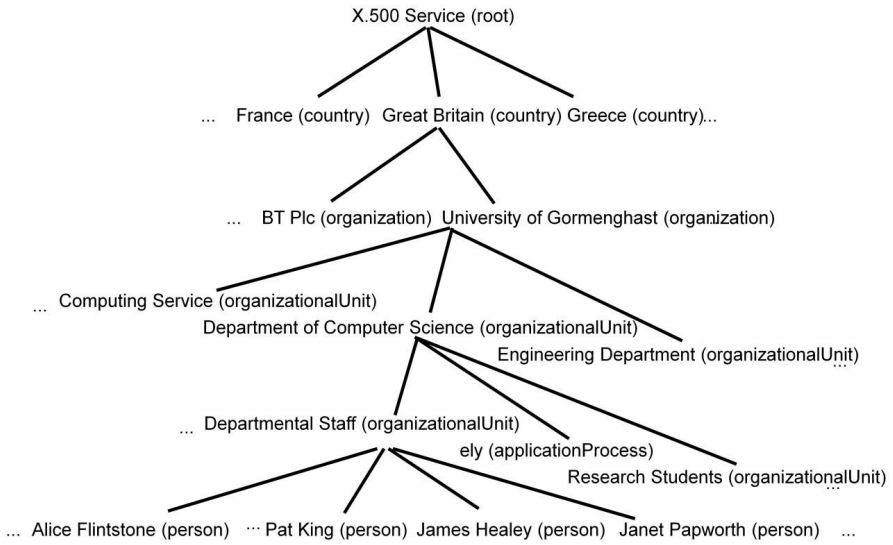
Part of the X.500 Directory Information Tree

The data structure for the entries in the DIB and the DIT is very flexible. A DIB entry consists of a set of attributes, where an attribute has a *type* and one or more *values*. The type of each attribute is denoted by a type name (for example, *countryName*, *organizationName*, *commonName*, *telephoneNumber*, *mailbox*, *objectClass*). New attribute types can be defined if they are required. For each distinct type name there is a corresponding type definition, which includes a type description and a syntax definition in the ASN.1

notation (a standard notation for syntax definitions) defining representations for all permissible values of the type.

DIB entries are classified in a manner similar to the object class structures found in object-oriented programming languages. Each entry includes an *objectClass* attribute, which determines the class (or classes) of the object to which an entry refers. *Organization*, *organizationalPerson* and *document* are all examples of *objectClass* values. Further classes can be defined as they are required.

The definition of a class determines which attributes are mandatory and which are optional for entries of the given class. The definitions of classes are organized in an inheritance hierarchy in which all classes except one (called *topClass*) must contain an *objectClass* attribute, and the value of the *objectClass* attribute must be the names of one or more classes. If there are several *objectClass* values, the object inherits the mandatory and optional attributes of each of the classes.



Administration and updating of the DIB • The DSA interface includes operations for adding, deleting and modifying entries. Access control is provided for both queries and updating operations, so access to parts of the DIT may be restricted to certain users or classes of user

Lightweight Directory Access Protocol • X.500's assumption that organizations would provide information about themselves in public directories within a common system has proved largely unfounded. group at the University of Michigan proposed a more lightweight approach called the *Lightweight Directory Access Protocol* (LDAP), in which a DUA accesses X.500 directory services directly over TCP/IP instead of the upper layers of the ISO protocol stack.

MULTIPLE CHOICE QUESIONS

1)_____ is a unique tag, usually a number, identifies the file within the file system.
a) File identifier
b) File name
c) File type
d) none of the mentioned

Ans A

2. To create a file
a) allocate the space in file system
b) make an entry for new file in directory
c) both (a) and (b)
d) none of the mentioned

Ans C

3) By using the specific system call, we can
a) open the file
b) read the file
c) write into the file
d) all of the mentioned

Ans D

4. File type can be represented by
a) file name
b) file extension
c) file identifier
d) none of the mentioned

Ans B

5. Which file is a sequence of bytes organized into blocks understandable by the system's linker?
a) object file
b) source file
c) executable file
d) text file

Ans A

6. What is the mounting of file system?
a) crating of a filesystem
b) deleting a filesystem
c) attaching portion of the file system into a directory structure
d) removing portion of the file system into a directory structure

Ans C

7. Mapping of file is managed by
a) file metadata
b) page table
c) virtual memory
d) file system

Ans A

8. Mapping of network file system protocol to local file system is done by
a) network file system
b) local file system
c) volume manager
d) remote mirror

Ans A

9. Which one of the following explains the sequential file access method?
a) random access according to the given byte number
b) read bytes one at a time, in order
c) read/write sequentially by record
d) read/write randomly by record

Ans B

10. file system fragmentation occurs when
a) unused space or single file are not contiguous
b) used space is not contiguous
c) unused space is non-contiguous
d) multiple files are non-contiguous

Ans A

11. A file control block contains the information about
a) file ownership
b) file permissions
c) location of file contents
d) all of the mentioned

Ans D

12. The data structure used for file directory is called
a) mount table
b) hash table

c) file table
d) process table

Ans B

13.Which protocol establishes the initial logical connection between a server and a client?
a) transmission control protocol
b) user datagram protocol
c) mount protocol
d) datagram congestion control protocol

Ans C

14. A server crash and recovery will _____ to a client.
a) be visible
b) affect
c) be invisible
d) harm

Ans C

15. A machine in Network file system (NFS) can be _____.
a) client
b) server
c) both a and b
d) neither a nor b

Ans C

16. A _____ directory is mounted over a directory of a _____ file system.
a) local, remote
b) remote, local
c) None of these

Ans C

17. The _____ becomes the name of the root of the newly mounted directory.
a) root of the previous directory
b) local directory
c) remote directory itself
d) None of these

Ans B

18) _____ mounts, is when a file system can be mounted over another file system, that is remotely mounted, not local.
a) recursive
b) cascading
c) trivial
d) None of these

Ans B

19) In UNIX, the file handle consists of a _____ and _____.
a) file-system identifier
b) an inode number
c) a FAT
d) a file pointer

Ans A abd B

20) The server maintains a/an _____ that specifies local file systems that it exports for mounting, along with names of machines that are permitted to mount them.
a) export list
b) import list
c) sending list
d) receiving list

Ans A

21) What are the different ways in which clients and servers are dispersed across machines ? (Choose Two)
a) Servers may run on dedicated machines
b) Servers and clients can be on same machines
c) Distribution cannot be interposed between a OS and the file system
d) OS cannot be distributed with the file system a part of that distribution

Ans A and B

22) What are the characteristics of a DFS ? (Choose Two)
a) login transparency and access transparency
b) Files need not contain information about their physical location
c) No Multiplicity of users
d) No Multiplicity if files

Ans A and B

23) What is not a characteristic of a DFS ? (Choose three)
a) Fault tolerance
b) Scaleability
c) Heterogeneity of the system
d) Upgradation

Ans A,B &C

24) What are the different ways file accesses take place ? (Choose three)
a) sequential access
b) direct access
c) random
d) indexed sequential access

Ans a,b & d

25) What are the major components of file system ? (Choose three)
a) Directory service
b) Authorization service
c) Shadow service
d) System service

Answer : a,b & d

26) What are the different ways mounting of file system ? (Choose three)
a) Boot mounting
b) root mounting
c) explicit mounting
d) auto mounting

Answer : a,c & d

27)What is the advantage of caching in remote file access ?
a) Reduced network traffic by retaining recently accessed disk blocks
b) Faster network access.
c) Copies of data creates backup automatically
d) None of these

Answer : a

28) What is networked virtual memory ?
a) Caching
b) Segmentation
c) RAM disk
d) None of these

Answer : a

29) What are examples of state information ? (Choose three)
a) opened files and their clients
b) file descriptors and file handles
c) current file position pointers
d) current list of total users

Answer : a,b & c

30)What are examples of state information ? (Choose three)
a) Mounting information
b) Description of HDD space
c) Session keys
d) Lock status

Answer : a,c & d

31) Expansion of FTP is
a) Fine Transfer Protocol
b) File Transfer Protocol
c) First Transfer Protocol
d) None of the mentioned

Answer: b

32) FTP is built on _____ architecture
a) Client-server
b) P2P
c) Both of the mentioned
d) None of the mentioned

Answer: a

33) FTP uses _____ parallel TCP connections to transfer a file
a) 1
b) 2
c) 3
d) 4

Answer: b

34) Identify the incorrect statement
a) FTP stands for File Transfer Protocol
b) FTP uses two parallel TCP connections
c) FTP sends its control information in-band
d) FTP sends exactly one file over the data connection

Answer: c

35) If 5 files are transfered from server A to client B in the same session. The number of TCP connection between A and B is
a) 5
b) 10
c) 2
d) 6

Answer: d

36) FTP server
a) Mantains state
b) Is stateless
c) Has single TCP connection for a file transfer
d) None of the mentioned

Answer: a

37) The commands, from client to server, and replies, from server to client, are sent across the control connection in ____ bit ASCII format
a) 8

b) 7
c) 3
d) 5

Answer: b

38) Find the FTP reply whose message is wrongly matched
a) 331 – Username OK, password required
b) 425 – Can’t open data connection
c) 452 – Error writing file
d) 452 – Can’t open data connection

Answer: d

39) Mode of data transfer in FTP, where all the is left to TCP
a) Stream mode
b) Block mode
c) Compressed mode
d) None of the mentioned

Answer: a

40) The password is sent to the server using _____ command
a) PASSWD
b) PASS
c) PASSWORD
d) None of the mentioned

Answer: b

UNIT-5

SMALL ANSWER QUESTIONS

1. **Define transaction** recovery?

In a distributed data base system, the actions of a transaction (an atomii: unit of consistency and recovery) may occur at more than one site. The purpose of a distributed commit algorithm is to insure the atomicity of distributed transactions, thereby insuring the consistency of the data bases involved.

2. **Define** phantom deadlocks?
   Deadlock can occur in systems that implement locking for concurrency control during transactions, so these systems need some kind of mechanism to detect this and resolve the problem when it occurs.

One way to detect deadlock within a single system is to use what's known as a Wait-for graph, deadlock has occurred when there is a cycle within the graph and can be resolved by aborting one of the transactions/processes.

Deadlock can also occur in distributed systems where transaction locks are held in different servers, this means that the loop in the entire wait-for graph will not be apparent to any one server. A solution to this problem is to introduce a coordinator to which each server forwards it's wait-for graph, the idea here is that the coordinator will be able to produce a wait-for graph for the entire system and can therefore make a decision about which process/transaction should be aborted to resolve the deadlock.

The above introduces another potential problem known as Phantom Deadlock. This is when the information gathered at the coordinator regarding each servers wait-for graph is out of date, so a transaction may have released a lock, but the global wait-for graph shows it as still holding. Thus a deadlock might be detected that never existed!

3. **Define** wait-for-graph?

This is a simple method available to track if any deadlock situation may arise. For each transaction entering into the system, a node is created. When a transaction T_i requests for a lock on an item, say X, which is held by some other transaction T_j, a directed edge is created from T_i to T_j. If T_j releases item X, the edge between them is dropped and T_i locks the data item.

The system maintains this wait-for graph for every transaction waiting for some data items held by others. The system keeps checking if there's any cycle in the graph.

4. **Explain** deadlock recovery?

There are three basic approaches to recovery from deadlock:

1. Inform the system operator, and allow him/her to take manual intervention.
2. Terminate one or more processes involved in the deadlock
3. Preempt resources.

1 Process Termination

Two basic approaches, both of which recover resources allocated to terminated processes:

Terminate all processes involved in the deadlock. This definitely solves the deadlock, but at the expense of terminating more processes than would be absolutely necessary.

Terminate processes one by one until the deadlock is broken. This is more conservative, but requires doing deadlock detection after each step

2 Resource Preemption

When preempting resources to relieve deadlock, there are three important issues to be addressed:

Selecting a victim - Deciding which resources to preempt from which processes involves many of the same decision criteria outlined above.

Rollback - Ideally one would like to roll back a preempted process to a safe state prior to the point at which that resource was originally allocated to the process. Unfortunately it can be difficult or impossible to determine what such a safe state is, and so the only safe rollback is to roll back all the way back to the beginning. (I.e. abort the process and make it start over.)

Starvation - How do you guarantee that a process won't starve because its resources are constantly being preempted? One option would be to use a priority system, and increase the priority of a process every time its resources get preempted. Eventually it should get a high enough priority that it won't get preempted any more

4. **Explain** deadlock detection?

   If deadlocks are not avoided, then another approach is to detect when they have occurred and recover somehow. In addition to the performance hit of constantly checking for deadlocks, a policy / algorithm must be in place for recovering from deadlocks, and there is potential for lost work when processes must be aborted or have their resources preempted.

   1 Single Instance of Each Resource Type

   If each resource category has a single instance, then we can use a variation of the resource-allocation graph known as a wait-for graph.A wait-for graph can be constructed from a resource-allocation graph by eliminating the resources and collapsing the associated edges.

   2 Several Instances of a Resource Type

   The detection algorithm outlined here is essentially the same as the Banker's algorithm, with two subtle differences:

   In step 1, the Banker's Algorithm sets Finish[i] to false for all i. The algorithm presented here sets Finish[i] to false only if Allocation[i] is not zero. If the currently allocated resources for this process are zero, the algorithm sets Finish[i] to true. This is essentially assuming that IF all of the other processes can finish, then this process can finish also. Furthermore, this algorithm is specifically looking for which processes are involved in a deadlock situation, and a process that does not have any resources allocated cannot be involved in a deadlock, and so can be removed from any further consideration.

   Steps 2 and 3 are unchanged

   In step 4, the basic Banker's Algorithm says that if Finish[i] == true for all i, that there is no deadlock. This algorithm is more specific, by stating that if Finish[i] == false for any process Pi, then that process is specifically involved in the deadlock which has been detected

5. **Define** nested transactions?

   A **nested transaction** is a database transaction that is started by an instruction within the scope of an already started transaction.

   Nested transactions are implemented differently in different databases. However, they have in common that the changes are not made visible to any unrelated transactions until the outermost transaction has committed. This means that a commit in an inner transaction does not necessarily persist updates to the database.

In some databases, changes made by the nested transaction are not seen by the 'host' transaction until the nested transaction is committed. According to some,[who?] this follows from the isolation property of transactions.

6. **Explain** concurrency control in distributed transactions?

   In database systems and transaction processing (transaction management) distributed concurrency control refers primarily to the concurrency control of a distributed database. The most common distributed concurrency control technique is strong strict two-phase locking (SS2PL, also named rigorousness), which is also a common centralized concurrency control technique. SS2PL provides both the serializability, strictness, and commitment ordering properties. Strictness, a special case of recoverability, is utilized for effective recovery from failure, and commitment ordering allows participating in a general solution for global serializability. For large-scale distribution and complex transactions

7. **Define** distributed deadlock?

   The same conditions for deadlock in uniprocessors apply to distributed systems. Unfortunately, as in many other aspects of distributed systems, they are harder to detect, avoid, and prevent. Four strategies can be used to handle deadlock:

   ignorance: ignore the problem; assume that a deadlock will never occur. This is a surprisingly common approach.

   detection: let a deadlock occur, detect it, and then deal with it by aborting and later restarting a process that causes deadlock.

   prevention: make a deadlock impossible by granting requests so that one of the necessary conditions for deadlock does not hold.

   avoidance: choose resource allocation carefully so that deadlock will not occur. Resource requests can be honored as long as the system remains in a safe (non-deadlock) state after resources are allocated.

8. **Define** atomic commit protocols?

   Informally, an atomic commitment problem requires processes to agree on a common outcome

   which can be either Commit or Abort. More specifically, an atomic commit protocol must

   guarantee the atomic commitment properties [1]:

   • AC1: All processes that reach an outcome reach the same one.

   • AC2: A process cannot reverse its outcome after it has reached one.

   • AC3: The Commit outcome can only be reached if all participants voted Yes.

   • AC4: If there are no failures and all participants voted Yes, then the outcome will be

   Commit.

   • AC5: Consider any execution containing only failures that the protocol is designed to

   tolerate. At any point in this execution, if all existing failures are repaired and no new

failures occur for sufficiently long, then all processes will eventually reach an outcome.

9. **Define** deadlock?

   A deadlock is a situation in which two computer programs sharing the same resource are effectively preventing each other from accessing the resource, resulting in both programs ceasing to function.

   The earliest computer operating systems ran only one program at a time. All of the resources of the system were available to this one program. Later, operating systems ran multiple programs at once, interleaving them. Programs were required to specify in advance what resources they needed so that they could avoid conflicts with other programs running at the same time. Eventually some operating systems offered dynamic allocation of resources. Programs could request further allocations of resources after they had begun running. This led to the problem of the deadlock.

10. **Explain** nested transactions?

    A **nested transaction** is a database transaction that is started by an instruction within the scope of an already started transaction.

    Nested transactions are implemented differently in different databases. However, they have in common that the changes are not made visible to any unrelated transactions until the outermost transaction has committed. This means that a commit in an inner transaction does not necessarily persist updates to the database.

    In some databases, changes made by the nested transaction are not seen by the 'host' transaction until the nested transaction is committed.

11. **List** the methods of the concurrency control?

    lock-based protocols use a mechanism by which any transaction cannot read or write data until it acquires an appropriate lock on it. Locks are of two kinds −

    Binary Locks − A lock on a data item can be in two states; it is either locked or unlocked.

    Shared/exclusive − This type of locking mechanism differentiates the locks based on their uses. If a lock is acquired on a data item to perform a write operation, it is an exclusive lock. Allowing more than one transaction to write on the same data item would lead the database into an inconsistent state. Read locks are shared because no data value is being changed.

12. **Define** time stamp ordering?

    A timestamp is the current time of an event that is recorded by a computer. Through mechanisms such as the Network Time Protocol (NTP), a computer maintains accurate current time, calibrated to minute fractions of a second. Such precision makes it possible for networked computers and applications to communicate effectively. The timestamp mechanism is used for a wide variety of synchronization purposes, such as assigning a sequence order for a multi-event transaction so that if a failure occurs the transaction can be voided. Another way that a timestamp is used is to record time in relation to a particular starting point. In IP telephony , for example, the Real-time Transport Protocol (RTP) assigns sequential timestamps to voice packet s so that they can be buffer ed by the receiver, reassembled, and delivered without error. When writing a program, the programmer is usually provided an application program interface for a timestamp that the operating system can provide during program execution.

13. **Define** shrinking and growing phase?

According to the two-phase locking protocol, a transaction handles its locks in two distinct, consecutive phases during the transaction's execution:

Expanding phase (aka Growing phase): locks are acquired and no locks are released (the number of locks can only increase).

Shrinking phase: locks are released and no locks are acquired.

The two phase locking rule can be summarized as: never acquire a lock after a lock has been released. The serializability property is guaranteed for a schedule with transactions that obey this rule.

Typically, without explicit knowledge in a transaction on end of phase-1, it is safely determined only when a transaction has completed processing and requested commit. In this case all the locks can be released at once

14. **Define** 2PL?

In databases and transaction processing, two-phase locking (2PL) is a concurrency control method that guarantees serializability It is also the name of the resulting set of database transaction schedules (histories). The protocol utilizes locks, applied by a transaction to data, which may block (interpreted as signals to stop) other transactions from accessing the same data during the transaction's life.

By the 2PL protocol locks are applied and removed in two phases:

Expanding phase: locks are acquired and no locks are released.

Shrinking phase: locks are released and no locks are acquired.

Two types of locks are utilized by the basic protocol: Shared and Exclusive locks. Refinements of the basic protocol may utilize more lock types. Using locks that block processes, 2PL may be subject to deadlocks that result from the mutual blocking of two or more transactions.

15. **Define** the types of locks?

Database locks can be used as a means of ensuring transaction synchronicity. i.e. when making transaction processing concurrent (interleaving transactions), using 2-phased locks ensures that the concurrent execution of the transaction turns out equivalent to some serial ordering of the transaction. However, deadlocks become an unfortunate side-effect of locking in databases. Deadlocks are either prevented by pre-determining the locking order between transactions or are detected using waits-for graphs. An alternate to locking for database synchronicity while avoiding deadlocks involves the use of totally ordered global timestamps.

Pessimistic locking: a user who reads a record, with the intention of updating it, places an exclusive lock on the record to prevent other users from manipulating it. This means no one else can manipulate that record until the user releases the lock. The downside is that users can be locked out for a very long time, thereby slowing the overall system response and causing frustration.

Optimistic locking: this allows multiple concurrent users access to the database whilst the system keeps a copy of the initial-read made by each user. When a user wants to update a record, the application determines whether another user has changed the record since it was last read.

ESSAY QUESTIONS

1) Nested Distributed Transactions

In general case, a transaction accesses objects managed by multiple servers. invokes operations in several different servers. Atomic property of a distributed transaction. To achieve it, one server takes the coordinator position, to ensure the same outcome at all the servers. All the servers involved in a distributed transaction are called participant.

Two-phase commit protocol: communicate with each other to reach a joint decision about commit or abort. Distributed transactions need to be serialized globally, with local concurrency control.

Distributed deadlock:

1. Centralized algorithm ^

2. Distributed algorithm

Transaction recovery is used to ensure that all the objects involved in transactions are recoverable.

Structured in an invert-root tree.^

The outermost transaction is the top-level transaction. Others are sub-transactions. ^

A sub-transaction is atomic to its parent transaction.^

Sub-transactions at the same level can run concurrently. ^

Each sub-transaction can fail independently of its parent and of the other sub-transactions.

Main advantages of nested transactions^

Additional concurrency in a transaction: Sub transactions at one level may run concurrently with other sub-transactions at the same level in the hierarchy.

Transaction T:

a.withdraw(100);

b.deposit(100);

c.withdraw(200);

d.deposit(200);

More robust: Sub-transactions can commit or abort independently.

For example, a transaction to deliver a mail message to a list of recipients.

The rules for committing of nested transactions:

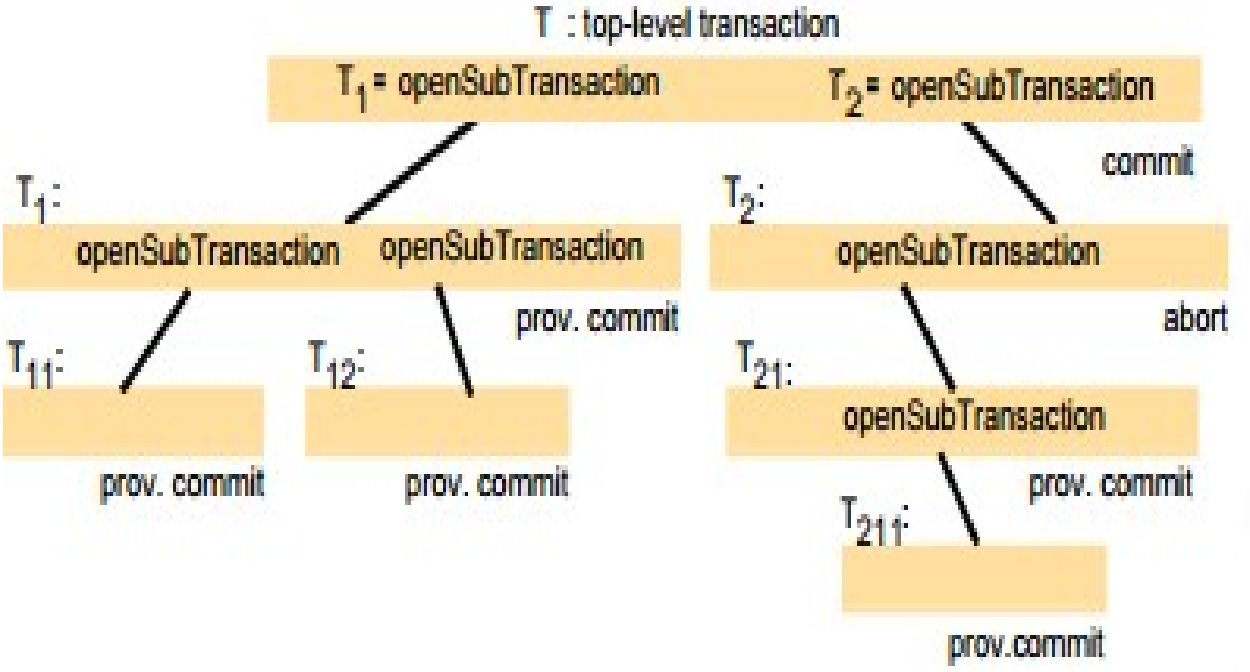
A transaction commits or aborts only after its child transactions have completed; ˆ

When a sub-transaction completes, it makes an independent decision on provisionally commit or abort. Its decision to abort is final. ˆ

When a parent aborts, all of its sub-transactions are aborted, even though some of them may have provisionally committed. ˆ

When a sub-transaction aborts, the parent can decide whether to abort or not. ˆ

When the top-level transaction commits, then all of the sub-transactions that have provisionally committed can commit



2) Atomic commit protocols

One-phase atomic commit protocol

The coordinator to communicate the commit or abort request to all the participants, and to keep on repeating the request until all the participants have acknowledged. ˆ

Problem: when the client requests a commit, it doesn't allow a server to make a decision to abort a transaction. Using concurrency control technique, it's possible for a server to abort a transaction (i.e. deadlock).

In the first phase, each participant votes for the transaction to be committed or aborted.

Once a participant has voted to commit a transaction, it is not allowed to abort it. It is in a prepared state. In the second phase of the protocol, every participant in the transaction performs the joint decision.

Two-phase commit protocol

When participants join a transaction, they will inform the coordinator. No communication during the progress of the transaction. ˆ

A client's request to commit (or abort) a transaction is directed to the coordinator. ˆ

When client requests abort Transaction, or one participant is aborted, the coordinator informs the participants immediately. ˆ

The two-phase commit protocol is used when the client asks the coordinator to commit the transaction.

In the first phase, the coordinator asks all the participants if they are prepared to commit.ˆ

In the second phase, it tells them to commit (or abort) the transaction.

Failures in two-phase commit protocol

Server failure ˆ

Each server saves information about two-phase commit protocol in its permanent storage.

Communication failure

There are several stages, where the coordinator or a participant cannot progress until it receives another request or reply message from others.

Timeouts:

To avoid process blocking, caused by waiting for reply, request messages. ˆ

For example, after a participant has voted “Yes”, it will wait for the coordinator to report the vote result.

send a get Decision request to the coordinator to determine the result.

Problem: coordinator failure wait for a long time .

Fix: obtain the vote result by contact other participants instead of only contacting the coordinator.

Two-phase commit protocol for nested transactions

Each sub-transaction starts after its parent and finishes before it. ˆ When a sub-transaction completes, it makes an independent decision about commit provisionally or abort. ˆ Difference between provisional commit and prepared to commit .ˆ

Provisional commit:

It’s not saved on permanent storage; It only means it has finished correctly and will agree to commit when it is asked to. Prepared commit: guarantees a sub-transaction will be able to commit.

After all sub-transactions are completed, the provisionally committed sub-transactions participate in a two-phase commit protocol. When a top-level transaction completes, its coordinator performs a two-phase commit protocol.

3) Time stamp ordering concurrency control

In a single-server transaction, the coordinator assigns a unique timestamp to each transaction.

And the serial equivalence is achieved by committing object versions in the order based on the timestamps of transactions. ˆ

In distributed transactions, the timestamp of a distributed transaction is issued by the first coordinator, and then passed to other coordinators. ˆ

Global timestamps: to achieve the serial equivalence requirement. All coordinators agree on the order of transaction timestamps. Server-id is less significant. ˆ

The same ordering of the timestamps at all the servers even if their local clocks are not synchronized. ˆ But, to be efficiency, roughly synchronized are needed. Conflict checks for each operation, So when transaction requests commit, it is always able to commit.

Optimistic concurrency control

A distributed transaction is validated by a collection of independent servers. ˆ

Example: ˆ T access A before U and U access B before T ˆ server X validates T first and server Y validates U first Æ commitment deadlock.ˆ

One approach is to use globally unique transaction number to define ordering of transactions, similar to the globally unique timestamps.

Transaction T		Transaction U	
Read(A)	*at X*	*Read(B)*	*at Y*
Write(A)		*Write(B)*	
Read(B)	*at Y*	*Read(A)*	*at X*
Write(B)		*Write(A)*	

4) Distributed Deadlocks

In a distributed system involving multiple servers accessed by multiple transactions, a global wait-for graph is constructed from the local ones.

Detection: find a cycle in the global wait-for graph.It is required to communicate between servers, to find cycles.

Simple approach: centralized deadlock detection

One server is selected as global deadlock detector ,Each server will send the latest copy of its local wait-for graph to this distinguished server.

Problems:

poor availability, lack of fault tolerance, no ability to scale, and high traffic

Phantom deadlock: a situation where a deadlock that is detected but is not really a deadlock.

It takes time to transmit local wait-for graphs. During that time, it’s possible some locks are released and there is no cycle any more in the new global wait-for graph.

Fix phantom deadlock: Since, in two-phase lock scheme, transactions cannot release objects before committing or aborting. So a phantom deadlock only happens when some transactions abort.

So, a phantom deadlock can be detected by informing aborted transactions.

A distributed approach ^

No global wait-for graph. ^

Servers try to find cycles by forwarding probe messages.

A probe message contains transaction wait-for relationships representing a path in the global waitfor graph.

When a server sends out a probe message- if there is a new edge inserted and this insertoperation may cause a potential distributed deadlock.

Example ^

If the server X adds the edge W ÆU and at this moment, U is waiting to access object B at server Y, in this case, X will send a probe message to server Y. Otherwise, X doesn't need to send a probe message.

How X knows that U is waiting or not? the coordinator of U knows that whether U is active or U is waiting for an object at some server.

Initiation: ^

When a server X finds that T starts waiting for U, and U is waiting to access an object at another server Y, X will initiate detection by sending a probe message containing the edge T ->U to Y.

Detection: ^

Consists of receiving probe messages and deciding whether deadlock has happened and whether to forward the probe messages. ^

i.e., first, Y finds that U is waiting for V, then it inserts the edge U ->V, check if there is a cycle, and if no cycle and transaction V is waiting for another object at other server, the new probe message is forwarded. ^

The path in probe message is increased, one edge at a time.

Example: ^

Server X initiate's detection by sending probe message to the server Y; ^

Y appends V to produce, forward it to Z; ^

Z appends W to produce. A cycle is detected.

5) Optimistic Concurrency Control

Concurrency control is the problem of synchronizing concurrent transactions (i.e., order the operations of concurrent transactions) such that the following two properties are achieved.

Principle – transaction proceeds without checking conflict with others and prior to commit, validates its change by checking to see if data items have changed by committed transactions –

Each transaction has three phases: Š

Read phase ƒ

committed version of data items for read - read set ƒ tentative version of data items for write - write set.

Validation phase ƒ

starts with End Transaction request ƒ validate its change by checking to see if data items have changed by other transactions ƒ if no conflicts, commit; otherwise, abort. Š

Write phase ƒ

make changes permanent.

Validation test rule:

Tj is serializable with respect to overlapping Ti if their operations conform to the following rules.

Ti	Tj	Rule
Read	*Write*	1. T*i* must not read data items written by T*j*
Write	*Read*	2. T*j* must not read data items written by T*i*
Write	*Write*	3. T*i* must not write data items written by T*j* and T*j* must not write data items written by T*i*

(Assumption: T*i* always preceeds T*j* if *i* < *j* and T*i* overlaps with T*j*)

Validation mechanisms

backward validation.

forward validation.

Backward validation

algorithm Š

checks transaction in validation phase with other preceding overlapping transactions that have entered validation phase.

Write operations are ok since Read operations of earlier transactions are done already (Rule1) ƒ

check if Read operations have any conflict with Write operations of earlier overlapping transactions(Rule 2) => if yes, abort transaction.

Valid := True;

for Ti := startTn + 1 to finishTn

do

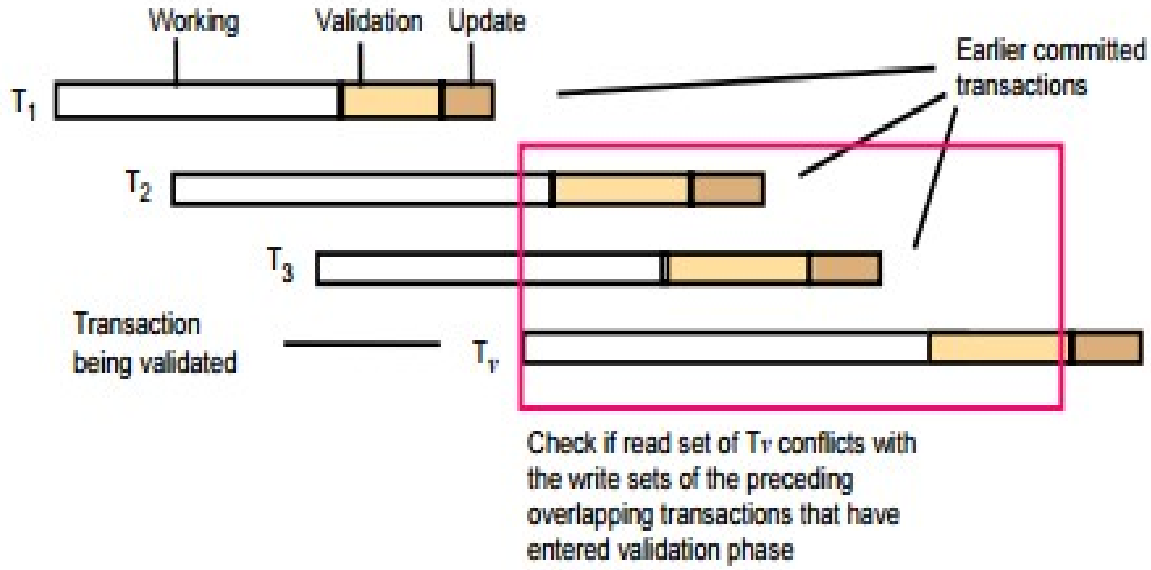
if read set of Tj intersects write set of Ti

Valid := False;

End.

no check is needed for transaction with only Write operations.

- Backward validation example



Forward validation

Algorithm

checks transaction in validation phase with other overlapping active transactions.

Read operations are ok since later transactions do not write until the Tj is done (Rule 2). ƒ

check if Write operations have any conflict with Read operations of overlapping active transactions (Rule 1) => if yes, abort transaction.

Valid := True;

for Ti := active1 to activeN

do

if write set of Tj intersects read set of Ti

Valid := False;

End.

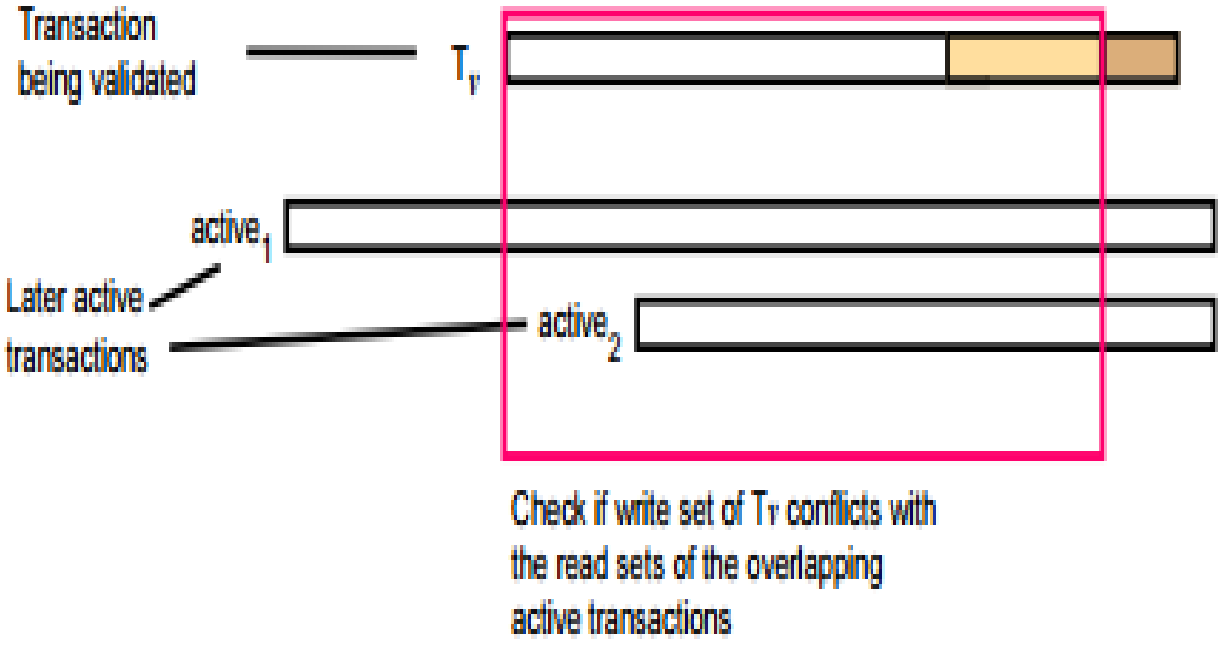
No check is needed for transaction with only Read operations.

other options than aborting the current transaction.Š

defer validation until conflicting transaction is done .Š

abort conflicting transaction instead.

Forward validation example



Issues in optimistic concurrency control

Overhead Š

Backward validation ƒ

if there exists long transaction, retention of old write sets of data item may be a problem

Š Forward validation.ƒ

A new transaction can start during the validation process -> increase chances by which the current transaction is forced to abort or delay.

Starvation Š

Prevention of a transaction ever being able to commit.

6) Timestamp Ordering

Assumption

Each transaction is given a unique timestamp when it starts.

Each transaction is given a unique timestamp when it starts.

Rule

Write operation is valid only if the data was last read and written by earlier transaction.

Rule1: Tj must not write data item read by any Ti where Ti > Tj (i.e. Tj >= max read time stamp of data item) Š

Rule 2: Tj must not write data item written by any Ti where Ti > Tj (i.e. Tj > max write time stamp of committed data item)

Read operation is valid only if the data was last written by earlier transaction.

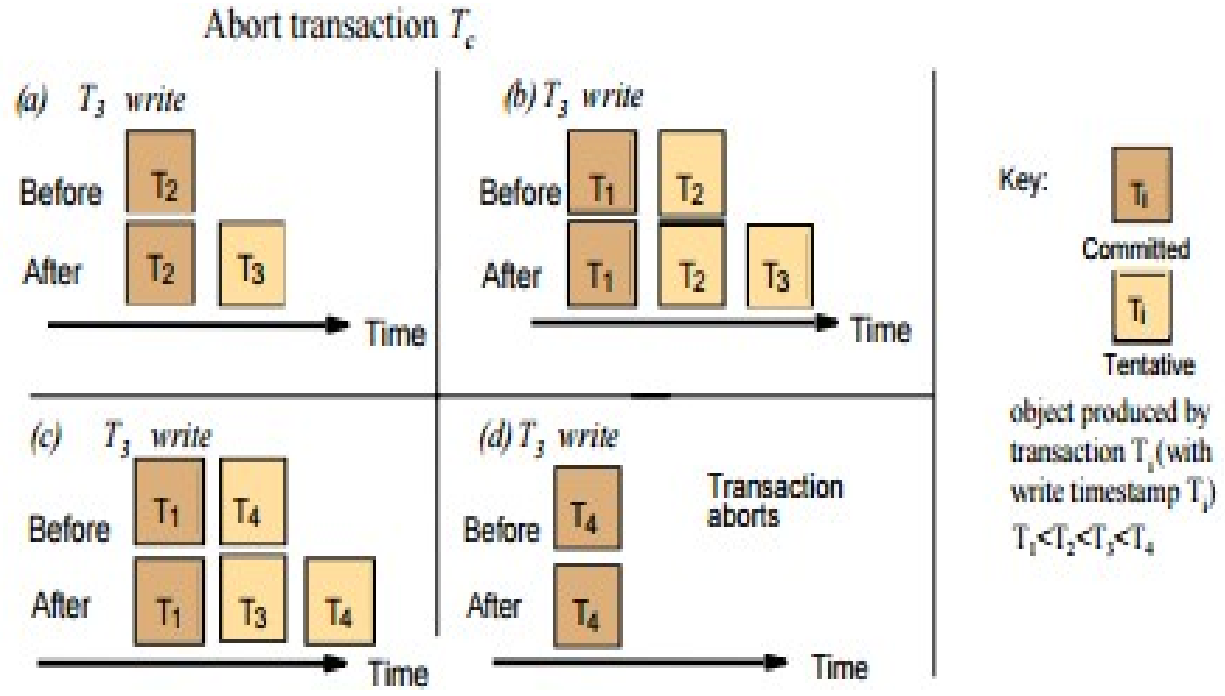
Rule 3: Tj must not read data item written by Ti where Ti > Tj (i.e. Tj > write time stamp of committed data item).

Write operations and time stamp

if (Tc ≥ maximum read timestamp on D &&

Tc > write timestamp on committed version of D)

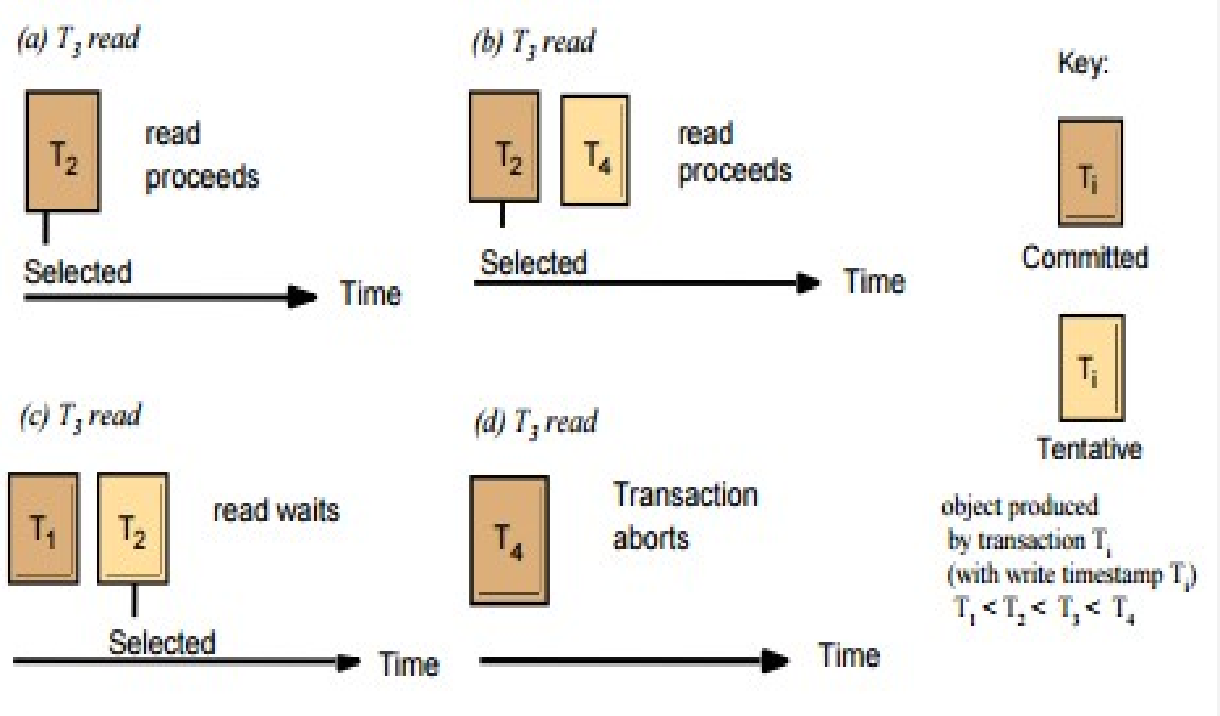
Perform write operation on tentative version of D with write timestamp Tc else /* write is too late */



Read operations and time stamp

```
if ( Tc > write timestamp on committed version of D) {
    let Dselected be the version of D with the maximum write timestamp ≤ Tc
     if (Dselected is committed)
     perform read operation on the version D selected
    else
        Wait until the transaction that made version D selected commits or aborts
then reapply the read rule
} else
    Abort transaction Tc.
```

Read operations and time stamp – example



Multi-version timestamp ordering

Keep old versions of committed data as well as tentative versions

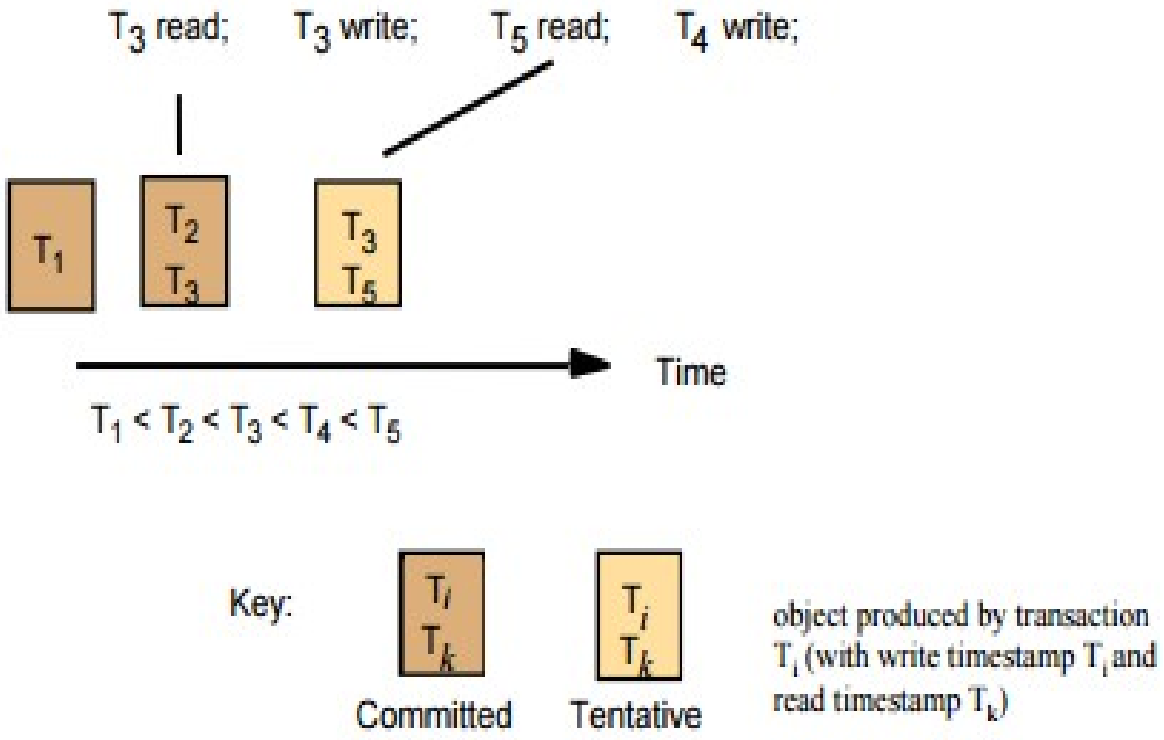
Š read operation is always allowed; may need to wait for earlier transactions to complete Š

No conflict between write operations since each transaction writes its own committed version (remove rule 2)

Write rule Š

If read time stamp (most recent version) <= Tj then perform write operation on a tentative version with write time stamp Tj

Multi-version timestamp ordering – example



Locking vs. timestamp ordering

Both are pessimistic

Dynamic vs static ordering

Write-dominated vs. read-dominated

Optimistic

Efficient when there are few conflicts

MULTIPLE CHOICE QUESTIONS

1) What is the reusable resource?
a) That can be used by one process at a time and is not depleted by that use
b) that can be used by more than one process at a time
c) that can be shared between various threads
d) none of the mentioned

Ans A

2) Which of the following condition is required for deadlock to be possible?
a) mutual exclusion
b) a process may hold allocated resources while awaiting assignment of other resources

c) no resource can be forcibly removed from a process holding it
d) all of the mentioned

Ans D

3. A system is in the safe state if
a) the system can allocate resources to each process in some order and still avoid a deadlock
b) there exist a safe sequence
c) both (a) and (b)
d) none of the mentioned

Ans C

4. The circular wait condition can be prevented by
a) defining a linear ordering of resource types
b) using thread
c) using pipes
d) all of the mentioned

Ans A

5. Which one of the following is the deadlock avoidance algorithm?
a) banker's algorithm
b) round-robin algorithm
c) elevator algorithm
d) karn's algorithm

Ans A

6. What is the drawback of banker's algorithm?
a) in advance processes rarely know that how much resource they will need
b) the number of processes changes as time progresses
c) resource once available can disappear
d) all of the mentioned

Ans D

7. For effective operating system, when to check for deadlock?
a) every time a resource request is made
b) at fixed time intervals
c) both (a) and (b)
d) none of the mentioned

Ans C

8. A problem encountered in multitasking when a process is perpetually denied necessary resources is called
a) deadlock
b) starvation

c) inversion
d) aging

Ans B

9. Which one of the following is a visual (mathematical) way to determine the deadlock occurrence?
a) resource allocation graph
b) starvation graph
c) inversion graph
d) none of the mentioned

Ans A

10. To avoid deadlock
a) there must be a fixed number of resources to allocate
b) resource allocation must be done only once
c) all deadlocked processes must be aborted
d) inversion technique can be used

Ans A

11) Those processes should be aborted on occurrence of a deadlock, the termination of which :
a) is more time consuming
b) incurs minimum cost
c) safety is not hampered
d) All of these

Ans B

12)If the resources are always preempted from the same process, _____ can occur.
a) deadlock
b) system crash
c) aging
d) starvation

Ans D

13)The solution to starvation is :
a) the number of rollbacks must be included in the cost factor
b) the number of resources must be included in resource preemption
c) resource preemption be done instead
d) All of these

Ans A

14) To _____ to a safe state, the system needs to keep more information about the states of processes.
a) abort the process
b) roll back the process

c) queue the process
d) None of these

Ans B

15) If we preempt a resource from a process, the process cannot continue with its normal execution and it must be :
a) aborted
b) rolled back
c) terminated
d) queued

Ans B

16) Cost factors of process termination include : (choose all that apply)
a) number of resources the deadlock process is holding
b) CPU utilization at the time of deadlock
c) amount of time a deadlocked process has thus far consumed during its execution
d) All of the above

Ans A and C

17) The two ways of aborting processes and eliminating deadlocks are : (choose all that apply)
a) Abort all deadlocked processes
b) Abort all processes
c) Abort one process at a time until the deadlock cycle is eliminated
d) All of these

Ans A and C

18) A deadlock can be broken by : (choose all that apply)
a) abort one or more processes to break the circular wait
b) abort all the process in the system
c) preempt all resources from all processes
d) to preempt some resources from one or more of the deadlocked processes

Ans A and D

19) Each request requires that the system consider the _____, _____, _____ to decide whether the current request can be satisfied or must wait to avoid a future possible deadlock. (choose three)
a) resources currently available
b) processes that have previously been in the system
c) resources currently allocated to each process
d) future requests and releases of each process

Ans C and D

20) A deadlock avoidance algorithm dynamically examines the _____, to ensure that a circular wait condition can never exist.

a) resource allocation state
b) system storage state
c) operating system
d) resources

Ans A

21) A state is safe, if :
a) the system does not crash due to deadlock occurrence
b) the system can allocate resources to each process in some order and still avoid a deadlock
c) the state keeps the system protected and safe
d) All of these

Ans B

22) All unsafe states are :
a) deadlocks
b) not deadlocks
c) fatal
d) None of these

Ans B

23) If deadlocks occur frequently, the detection algorithm must be invoked _____.
a) rarely
b) frequently
c) None of these

Ans B

24) The disadvantage of invoking the detection algorithm for every request is :
a) overhead of the detection algorithm due to consumption of memory
b) excessive time consumed in the request to be allocated memory
c) considerable overhead in computation time
d) All of these

Ans C

25) A deadlock eventually cripples system throughput and will cause the CPU utilization to _____.
a) increase
b) drop
c) stay still
d) None of these

Ans B

26) Which are the two complementary deadlock-prevention schemes using time stamps ?
(Choose two)
a) The wait-die scheme
b) The wait-n-watch scheme
c) The wound-wait scheme
d) The wait-wound scheme

Answer : a & c

27) If the transaction is rolled back, all the database changes made inside the transaction are .....

A. made permanent

B. made temporary

C. copied to the log

D. undone

Ans D

28) Which of the following is not a property of transactions?

A. Atomicity

B. Concurrency

C. Isolation

D. Durability

Ans B

29) A ..... ensures that transactions are performed as expected.

A. transaction processing monitor

B. transaction procedure monitor

C. isolation monitor

D. transaction log

Ans A

30) A transaction that completes its execution successfully is said to be .....

A. committed

B. rolled back

C. partially committed

D. Aborted

Ans A

31) ..... means that a transaction must execute exactly once completely or not at all.

A. durability

B. consistency

C. atomicity

D. isolation

Ans C

32) ..... means that when it ends, a transaction must leave the database in a consistent state.

A. Data isolation

B. Data duration

C. Data consistency

D. Data non-reputability

Ans C

33) The number of transactions executed in a given amount of time is called .....

A. utilization

B. execution rate

c. throughput

D. atomicity

Ans C

34) Isolation means .....

A. transaction must not interfere with each other

B. transaction must interfere with each other

C. transaction must be in consistent state

D. transaction must be executed immediately

Ans A

35) Which of the following ensures the atomicity of the transaction?

A. Transaction management component of DBMS

B. Application Programmer

C. Concurrency control component of DBMS

D. Recovery management component of DBMS

Ans A

36) ..... means that a transaction must make its changes permanent to the database ends.

A. isolation

B. locking

C. durability

D. consistency

Ans C

37) Throughput means

A. number of transactions that are committed in one hour

B. number of operations in a transaction

C. number of transaction that can be aborted in a given amount of time

D. number of transaction that can be executed in a given amount of time

Ans D

38) ..... deals with individual transactions.

A. isolate transactions

B. transaction recovery

C. system recovery

D. media recovery

Ans B

39) The part of a database management system which ensures that the data remains in a consistent state is

A. authorization and integrity manager

B. buffer manager

C. transaction manager

D. file manager

Ans C

40) ..... protocol is used to perform multiple transactions that execute on different database.

A. commit

B. two phase lock

C. two phase commit

D. locking

Ans C