B.TECH.

TCS-801

EIGHTH SEMEMSTER EXAMINATION 2010-11

DISTRIBUTED SYSTEMS

Time: 3 Hours *Total Marks: 100*

Note: ***Attempt all questions.***

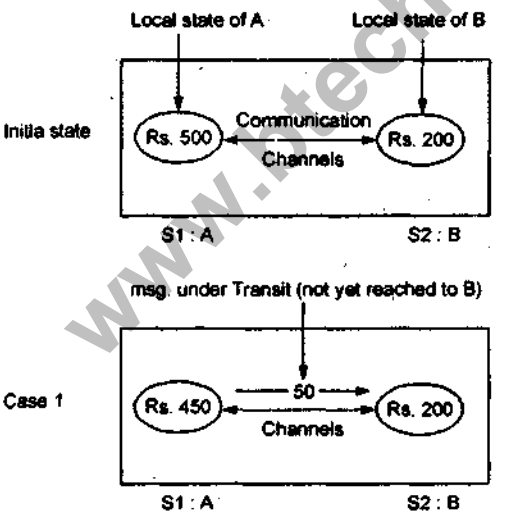
1. Attempt any two parts of the following

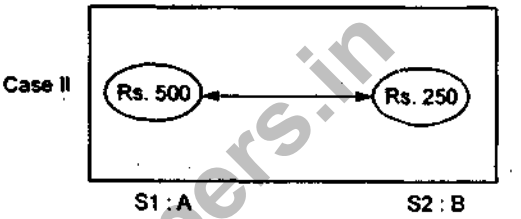
(a) what are the inherent limitations of distributed system? and shared memory?

Ans. The inherent limitations of the distributed system are:

1. **Concurrency:** The capacity of the system to handle stored resources can be increased by adding more resources to the network.
2. **Lack of Global Clock:** The only communication is by sending messages through a network.
3. **Independent failures:** The programs may not be able to detect whether the network has failed or has become usually slow. Above limitation create difficulty in obtaining coherent global state.

For example:





Case I: S1 records its local state (Rs 450) just after debit (– 50) and S2 records its location (200) before receiving. If transit message is not taken care off

Global stage = local state S1 + local State S2

= 450 + 200

= 650 = Rs. 50 missing i.e in-coherent system.

Case II: S1 records its state (Rs 500) before debit and S2 records its state after credit hence there is surplus (Rs. 50) which is also in-coherent. From above two example, it is clear that communication channel cannot record its state by itself. Hence sites must have to coordinate their state recording activities in order to record the channel state or coherent state of system.

Absence of Global clock: It will be difficult to order event. That is temporal ordering of events. This will result in difficulty in design and development of distributed system. We cannot think of existence of distributed system if we do not have mechanism by which we can order the event temperailly.

Absence of shared Memory: Normally computer do not share a common memory in distributed system. So the whole system is not available and up to date.

(b) Describe Causal ordering of messages and explain with a suitable example how it can be implemented by a system of vector-clocks.

Ans. The casual ordering of messages was first proposed by Birman and joseph and was implemented

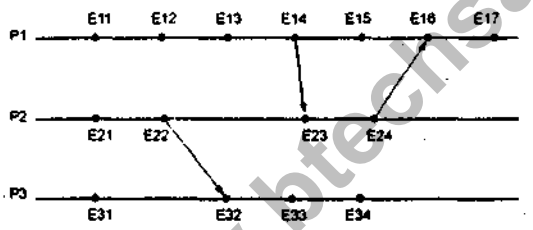
in ISIS. The casual ordering of messages deals with the notion of maintaining the same casual relationship that hold among "message receive" events. The casual ordering of messages should not be confused with casual ordering of events, which deals with notion of casual relationship among the events. In distributed system, the casual ordering of messages is not automatically guaranteed.

Implementation by a system of vector clocks

For keeping track of transitive dependencies among process for recovery purposes, a clock is maintained which is called vector clock.

- $C_i[1...n]$: is a 'Vector' clock at process P_i whose entries are the "assumed/best guess" clock values of different process.
- $Ci[Ji]$ $(Ji = i)$ is the best guess of Pi for Pj's clock.
- Vector clock rule:
- $Ci[i] = Ci[i] + d$, $(d > 0)$; for successive events in P_i
- For all k, $Cj[k] = \max(Cj[k], tm[k])$, when a message m with time stamp tm is received by Pj from Pi.

Example: The following fig stores the computers over three processes:



In process $P1$: $E11$, $E12$, $E13$, $E15$, $E17$ are the internal events. $E14$ is casually related with $E23$. $E16$ is again casually related with $E24$ event.

In process $P2$: $E21$ is internal event. $E22$ is casually related with $E32$.

In process P3: E31, E33, E34 are internal events. The vector clock according to the events occurred.

Initially:

$\{E11\,(1, 0, 0)\ E12\,(2, 0, 0)\ E13\,(3, 0, 0)\ E14\,(4, 0, 0)$
$\{E15\,(5, 0, 0)\ E16\,(6, 0, 0)\ E17\,(7, 0, 0)$

$\{E21\,(0, 1, 0)\ E22\,(0, 2, 0)\ E23\,(0, 3, 0)\ E24\,(0, 4, 0)$

$\{E31\,(0, 0, 1)\ E32\,(0, 0, 2)\ E33\,(0, 0, 3)\ E34\,(0, 0, 4)$

$E22$ is casually related with $E32$ i.e $E22 \rightarrow E32$ so $E32$ is updated with $E32$ (0, 2, 2). Now $E14 \rightarrow E23$, so $E23$ is updated with (4, 3, 0)

Accordingly $E24$ is updated with (4, 4, 0) which is casually related with $E16$ i.e $E24 \rightarrow E16$ so, $E16$ is updated with $E16$ (6, 4, 0)

(c) Define the problem of distributed mutual exclusion. What are the performance matrices for distributed mutual exclusion algorithms? Explain with a suitable example.

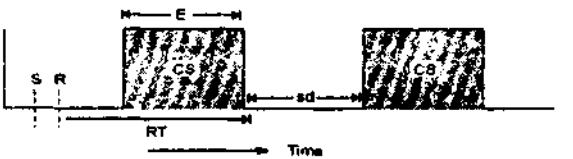
Ans. Mutual exclusion require that shared resources must be accessed by single process or a site at a time. An example is directory management, where an update to a directory must be done automatically because if updates and reads to a directory proceeds concurrently, reads may obtain inconsistent information. If an entry contains serval fields, a read operation may read some fields before the update and some after the update. But in distributed system, concurrent access to shared resources by several sites is involved. Resource shared is important issue in distributed system.

Performance metric for distributed mutual exclusion algorithm

S: CS request arrears

R: CS request message is sent out

sd: Synchronization delay i.e time between two CS.



(i) **Response Time(RT):** It is time between R and end CS, i.e time between request message sent out and completion of CS. For small 'RT' performance of ME algo will be high.

(ii) **Synchronization delay (sd):** It is time between two consecutive CS. That is time between end of CS and begin of CS. In this period messages are exchanged to arrive at ME decision.

For small 'sd', performance of ME algo will be high.

(iii) **Number of message per *CS*:** between two *CS*, there are messages exchanged as discussed A_3 number of exchanged messages reduces, the performance will improve.

(iv) **System Throughout:** It is derived from 'sd' and

'E' as system throughout = $\frac{1}{sd + E}$

This can also be called as average time for each CS invocater or rate at which CS request complete.

2. Attempt any two parts

(a) What are the deadlock handling strategies in distributed system? What are control organizations for distributed deadlocks detection? Discuss a algorithm which can remove the possibility of Phantom deadlock detection.

Ans. (a) There are three strategies to handle deadlocks: Deadlock prevention deadlock avoidance, and deadlock detection

1. Deadlock prevention: It is achieved by either having a process acquire all the needed resource simultaneously before it begins execution or by precempting a process that holds the needed resources. There are four condition which should simultaneously for deadlock to occur, mutual exclusion, hold and wait, no precemption, circular wait. So a deadlock can be prevented if one of there condition holds false.

2. Deadlock Avoidance: Resources are allocated only if resultant global system state is safe. Each site maintain its local state that require storage in the form of memory which is an overhead. Due to large number of process and resources, it will be computationally expensive to check for a safe state. This approach is impractical which further reduce concurrency and throughout as system ascertain before any resources is allocated.

3. Deadlock detection: Resources are allocated to process if available and every sit maintain a graph (WFG) to detect a cycle. Whenever a cycle is determined the site apply for deadlock detection algo and deadlock is resolved by either precemption of resource or by aborting some of the process control organization for distributed deadlock detection

(a) **centralized deadlock detection algo:** Control site has responsibility to construct RAG or WFG

(b) **Distributed DD algo:** Responsibility is shared equally. WFG is spread over many site, and these site participate to detect deadlock.

(c) **Hiearcheal deadlock detection algo:** This approach combines benefit of centralized and distributed approach. Here sites detect deadlock in only descenders sites (child sites) and resultant WFG or RAG is forwarded to site to cumulate the result.

False or non existence deadlock are known as phantom deadlock. These deadlocks are due to incosistency of global state attributed to absence of global memory. One phase algorithm does not cause detection of false deadlock in the system. In one phase algo, one status report is received from each site. A single report consist of two status table.

(a) resource status table (b) Process status table from every site periodically. The centralized site use the single report (resource status, process states) to construct global WFG. The resource status table for a site holds all the transactions going on or waiting. The process status table keeps track of all the resource locked or waited. The central site search for a cycle or wait in WFG. If cycle is found, there is a deadlock in the system otherwise not. This algo does not detect false deadlock due to two status report only common information in these two status report is used for constructing WFG

(b) What do you mean by agreement protocol? What are differences between Byzantine Agreement Problem, the consensus problem and the interactive consistency problem? Discuss impossibility results for Byzantine Agreement.

Ans. In distributed system, it is often required that sites (or processor)reach mutual agreement. There are significant chance that processor may behave maliciously preventing other processors from reaching a common agreement. Agreement protocol are used to allow all the processor to reach a common agreement.

Byzantine agreement problem The agreement problem concerns problem of single value agreement by source processor. In this single source has initial value.

Agreement: All non faulty processors must agree upon same value by all non faulty processes must be the same as initial value of source.

Termination : Each non faulty process must eventually decide on a value.

Consensus Problem: This agreement problem also concerns single value agreement by source process. in this all the processor have an initial value.

Agreement: All non faulty processes must agree on same (single) value.

Validity: If all non faulty process have the same initial value, then the agreed upon value by all non faulty processes must be that same value.

Termination: Each non faulty process must eventually decide on a value.

Interactive consistency Problem: Here agreement concerns a set of common values. Performance aspect for agreement protocol are generally determined by following matrices.

TIME, MESSAGE, TRAFFIC, STORAGE OVERHEAD.

Impossibility results for Byzantine agreement

It states that byzantine agreement cannot be reached among three processors, when one processor is faulty.

Proof: Consider a system with three processor P_0, P_1, P_2. Processors has to agree upon only two value 0 and 1 P_0 initiate initial value.

Case I: P_0 is non faulty: P_2 is faulty. P_0 broadcast initial value of 1 to both P_1 and P_2. P_2 acts maliciously and communicate value 0 to P_1. P_1 receives conflicting values from P_1 and P_2. Since 0 is non faulty, P must agree upon value '1'

Case II: P_0 is faulty: Suppose P_0 sends '1' to P_1 and '0' to P_2. P_2 will communicate value '0' to P1 to P_1 as far as P_1 is concerned the case will look identical to case I. So any agreement protocol which works for three processors cannot distinguish between the two cases and must force P_1 as agreed upon value. However in case II, this will work only if P_2 is also made to accept '1' as agreed upon value.

(c) What are the differences in resources and communication deadlock? Discuss salient feature of a path pushing algorithm and explain how wait for dependencies are propagated in the form of paths.

Ans. Resource Deadlock: A process needs multiple resources for an activity. Deadlock occurs if each process in a set request resources held by another process in the same set, and it must receive all the requested resource to move further.

Communication deadlock: Processor wait to communicate algo, information about wait for depending is propagated in the form of paths. Here algorithm has two features:

1. The non local portion of global TWF (transaction wait for graph) at a site is abstracted by a distinguished node (called external or Ex) which helps is determining potential multisided deadlock without requiring huge global TWF graph to be stored at each site.
2. Transactions are totally ordered, which reduces the number of messages and con decreases deadlock detection overhead. It also ensure exactly one transaction in each cycle detects the deadlock

Example

Server A $t_1 \rightarrow t_2$

Server B $t_2 \rightarrow t_3$

Server C $t_1 \rightarrow t_2 \rightarrow t_3$

Now server C knows $t_3 \rightarrow t_1$ locally and detects the deadlock transaction lock and wait for resources. One trasaction can intiate atmost one subtraction at a given point in time. A site waits for deadlock relation information and local graph to build an updated graph. Non local portion of graph is distinguished by nodes called *Ex* (External). The site detects all cycle and breaks local cycles: those do not contain *Ex* nodes. cycles with ex nodes are potential global cycles. This string is transmitted to all other sites where a substraction of T_2 is waiting for a message from another transaction in other site. A string *Ex*, T_1, T_2, T_3, *Ex*" is sent to other sites only of $T1$ is (exceally) higher than T_3.

3. Attempt any two parts

(a) Explain the RPC mechanism for communication among distributed objects and also discuss different design issues in RPC.

Ans. Distributed object system may adopt client-server architecture. In their case object are managed by servers and client program calls a procedure in another program running in a server process. When a process on machine A calls a production located an machine B, calling process A is suspended and execution of called procedure takes place on B. Information can be transported from caller to callee in parameter and can come back in procedure result.

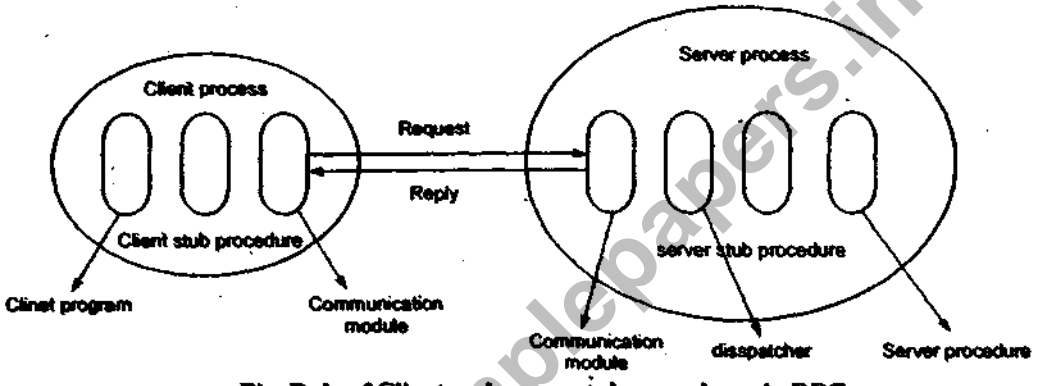


Fig. Role of Client and server stub procedures in RPC

Design issues in RPC

1. RPC protocol must provide, unique specification of a procedure to be called.
2. It must have provisions for matching response message to request messages.
3. It must have provisions for authenticating the caller to service and vice versa.

Beside these requirements, feature that detect following because of roll over errors, implementation bugs, user error and network administration are:

→ RPC protocol mismatches.

→ Remote program protocol version mismatches

→ Protocol errors

→ remote authentication failure

(b) (i) Give the architecture of Sun Network file system.

Ans. Sun micro system's network file system is usually called NFS.

NFS architecture: It allows an arbitrary collection of clients and server to share common file system. NFS allows every machine to be both a client and server at same tine. The basic architecture chracterstics is that server export directories and clients mount them remotely.

NFS Protocols NFS support a heterogenous system with clients and servers possibly running different OS on different h/w, it is essential that interface between clients and servers be well defined. NFS accomplishes this goal by defining two client servers protocol. The first NFS protocol handles mounting. A clients can send a path name to a server and request permission to mount that directory. The second protocol is for directory and file access. Clients can send messages to server to manipulate directions and to read and write file.

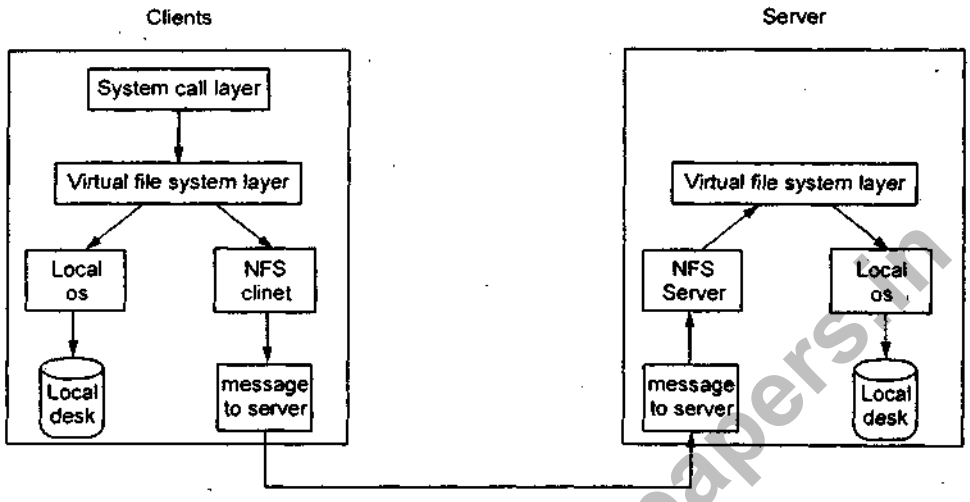


Fig. NFS Layer Structure

The top layer is system call layer. This handles calls like OPEN, READ and CLOSE. The task of VFS layer is to maintain a table with one entry for each open file. The communication on server happen via NFS server.

(ii) Discuss the mechanisms for building distributed file system.

Ans. Mechanism for building distributed file system:

1. **Mounting:** It allows binding of different file name spaces together. This combination forms a hierarchical structured tree. It involves setting up of a mount among various name feeds of name space tree. A name space can be bounded to a node in namespaces tree. The node may be internal node or leaf node of name space tree.
2. **Caching:** It is used to reduce access time of data file and so delay in accessing data is also reduced. This mechanism involves creation of copy of data file from server to client. When client makes request, data is searched locally at client, side and if data not found on client, data is brought to client cache from server.
3. **Bulk data transfer:** Multiple consecutive data blocks are transferred from server to clients. It amortizes protocol processing overhead and desk seek time over many consecture blocks of a file.
4. **Encryption:** It is used for enforcing security in distributed system.

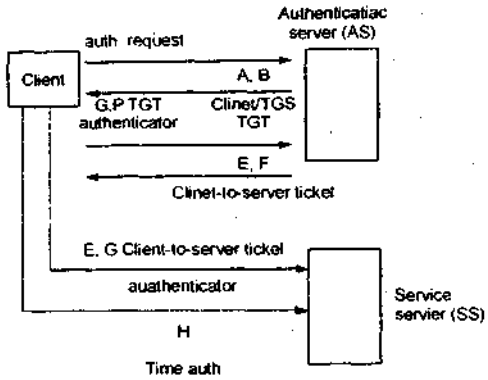
(c) Discuss the Kerberos with its steps towards achieving the authentication.

Ans. Kerberos is computer network authentication protocol and is built on symmetric key cryptography and requires trusted third party. It may use public key cryptography by utilizing asymmetric key cryptography during certain phase of authenticating. It is a free software published by Massachusetts institute of Technology (MIT)

The client authenticates itself to the authentication server and receives a ticket. It then contact TGS (Ticket grating server) and recieves a ticket. It demonstrates its identity and ask for a server. If client is eligible for the server, TG server sends another ticket to client.

AS : Authentication server, SS: Service server

TGS : Ticket granting server TGT: Ticket granting Ticket.



The client authenticate to AS once and receives TGT from AS. When client wants to contact SS, it can use these ticket to get additional tickets from TGS, for SS, without resorting to using the shared server. The phases:

Uses client based login: User entries a username and password on client machine.

The client performs a one way function on entered password and this becomes secret key.

Client authentication: Client sends a clear text message of user ID to AS requesting services server on behalf of user. As generates the secret key by hashing password of user found at db.

As checks to see if client is in db, If it is, AS sends back messages to clients:

Message A: Client/TGS session key encrypted using secret key.

Message B: TGT encrypted using secret key of TGS.

Once client recieves messages A and b, it attempt's to decrypt messages A with secret key generated from password entered by user. With valued password and secret key the client decrypts message A to obtain client/TGS sersion key. This sersion key is used for further communication with TGS.

Client Server aunthenticaion: When requesting services, the client sends following messages to TGS:

Message C: Composed of TGT from B and id of requested service.

Message D: Authenticator (Composed of client ID and timestamp), enerypted using client/TGS sersion key.

Upon receiving message C and D, TGS retrieve message B out of message C. It decrypts message B using TGS secret key. This gives client/TGS session key. Using this key, TGS decrypts message D and send following two messages to clients.

Message E: Client to server ticket encrypted using service's secret key.

Message P: Client/ server session key encrypted with client/TGS session key.

4. Attempt any two parts

(a) Give the classification of distributed concurrences control techniques.

Ans. Classification of distributed concurrence control techniques.

1. **Locking:** Locks on objects are held locally. When a process needs to read or write a file, it first locks the file. Locking can be done using a single centralized lock messages or with local lock manager for managing local file.

   Read lock: No. of process can apply read lock on a file.

   Write lock: When a file is locked with write lock, no other locks of any kind is permitted.

   There are two phase locking and three phase locking protocol for applying locks on a file.

2. **Timestamp ordering:** It allows each coordinator to issue global unique timestamp. A globally unique timestamp is issued to client by first coordinator accessed by a transaction. The transaction timestamp is passed to the coordinator at each server whose objects perform an operation in transaction. To achieve same ordering, coordinators must agree as to ordering of their timestamps. The timestamp consists of a pair local timestamp, server ID's.

3. **Optimistic concernecy control:** A distributed transaction is validated by a collection of independent server, each of which valedates transactions that access its own objects. The validation of all the servers take place during first phase of two phase commit protocol.

(b) "Three-Phase is a non-blocking protocol, "Justify the statement with its working and state transition diagrams.

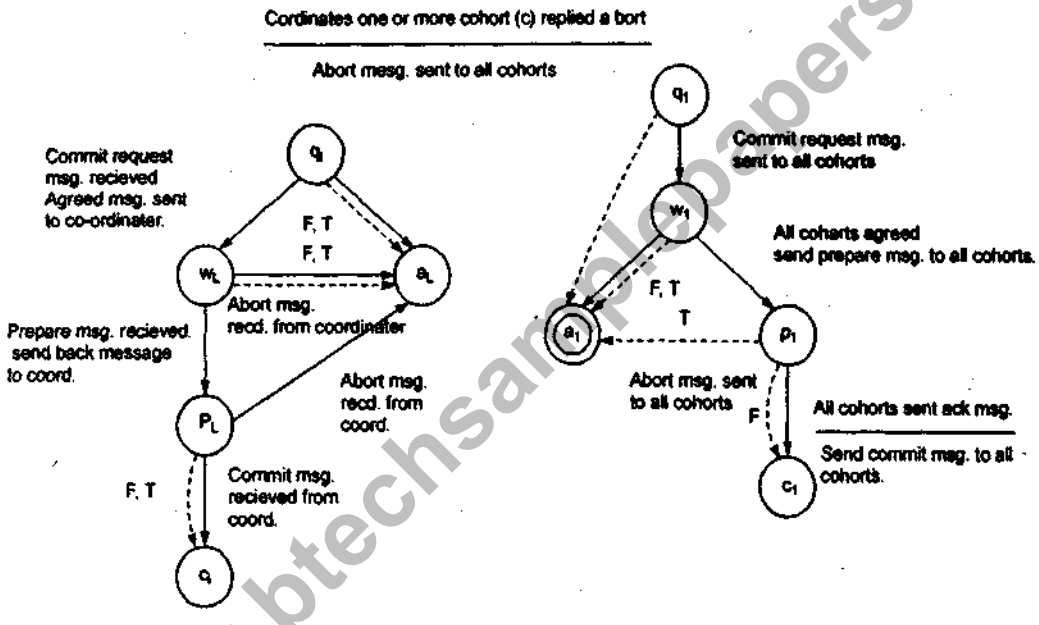
Ans. Two phase commit protocol guarantees global atomicity, it biggest drawback is blocking protocol. Whenever the coordinator fails, coherent sites will have to wait for the recovery of coordinator. But three phase commit protocol is non blocking

Assumption:

1. Each site uses write-ahead log protocol.
2. At most one site can fail during execution of transaction.

Basic idea is that before comment protocol begins, all the sites are in stage. If coordinator fails while in state g. all the cohernets perform timeout transition, thus aborting the transaction. Upon recovery coordinator performs failure transaction.

Cohort i ($i = 2, 3...n$)



$T \rightarrow$ Timeout transition

$F \rightarrow$ Failure transition

$F_iT \rightarrow$ Failure/ Timeout transition

Phase I: During first phase, coordinator is in state w_L and each coherent is either is stage a or w or q depending on whether it has received commit-request message or not. If a cohort fails, coordinator timeout waiting for a agreed message from failed cohort. In this case, coordinator aborts transaction and send abort messages to all cohorts.

Phase II: The coordinator sends a prepare messages to all cohorts if all the cohorts have sent agreed message in phase I.

Otherwise coordinator will send and abort message. On receiving a prepare messages, a cohorts sends an ack message to co-ordinator.

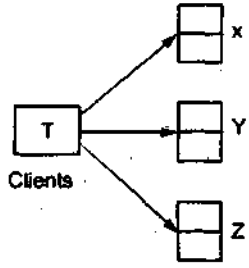
Phase III: On receiving ack to prepare messages from all cohorts coordinator sends a commit messages to all cohorts. A cohorts on receiving commit massage commit the transaction. If coordinator fails before sending commit message, it commits the transaction upon recovery.

(c) Explain following with suitable example

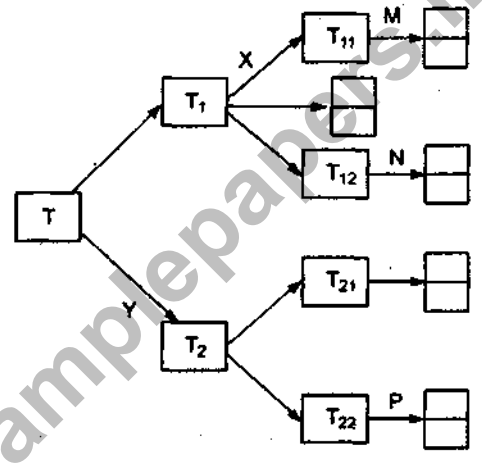
(i) Flat and Nested transaction

Ans. Flat transaction: In a flat transaction, a client make request to more than one server. A flat client transaction completes each of its request before going on the next one. Therefore each transaction access server's objects sequentially. When server use locking a transaction can only be waiting for one objects at a time.

Nested transaction: The top level transaction can open subtransaction and each subtransaction initiate further subtransactions down to any depth of nesting



Flat transaction



Nested transaction

(ii) 2PL and strict 2PL

Ans. According to two phase locking protocol a transaction handles its locks in two consecutive phases:

1. **Expanding phase (growing phase):** Locks ae acquired and no locks are released.

Shrinking phase: locks are released and no locks are acquired.

Serializability property is guaranteed in 2PL. Typically, without explicit knowledge in a transaction on end of phase, it is safety determined only when a transaction has ended its ready state in all its processes. In this case, phase 2 can end immediately.

Strict two phase locking

A transaction needs to comply with 2PL, and release it's write (exclusive) locks only after it has ended, being either committed or aborted. On the other hand, read (shared) locks are released regularly during phase 2.

5. Attempt any two parts

(a) Discuss the all pair Shortest Path (APSP) problem with its application. Discuss the complexity of this algorithm

Ans. The problem of computing a shortest path between any two nodes of a graph is known as all pairs shortest path algo.

Floyd Warshall algo: The algo considers "intermediate" vertices of a shortest path, where intermediate vertex of a simple path $P = <V_1, V_2 ... V_l>$ is any vertex of p other then V_1 that any vertex is set $\{V_2 ... V_3 .... V_{l-1}\}$

Recursive solution

$$d_{ij}(k) = \begin{cases} w_{ij} & \text{if } k = 0 \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{if } k \geq 1 \end{cases}$$

$d_{ij}(k) \Rightarrow$ weight of a shortest path from vertex i to vertex j for which all intermediate vertices are in set $\{1, 2, ...k\}$

1. $n \leftarrow$ rows $[w]$
2. $D^{(0)} \leftarrow w$
3. for $k \leftarrow 1$ to n
4. do for $i \leftarrow 1$ to n
5. do for $j \leftarrow 1$ to n.
6. do $d_{ij}^{(k)}$ min $(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
7. Return $D^{(n)}$

complexity of the algo will be $0(n^3)$

(b) What are wave algorithm? Discuss the usage and application of wave algorithm. What are the requirements of wave algorithm?

Ans. In the design of distributed system algorithm for various application, several very general problems for process network appear frequently as subtasks. These elementary task include the broadcasting of information

- Achieving a global synchronization between process.
- Triggering executione of some event in each process.
- Computing a function of each process holds part of the I/P.

For performing these tasks, wave algorithms are employed in which messages is passed according to some prescribed topology dependent scheme.

Initial knowledge:

1. Each process knows its own unique name
2. Each process knows names of its neighbours
3. Number of decisions to occur in each process.

Usage and application of wave algorithm

1. The introduction of concept facilitate later treatment of more involved algorithm because properties of their sub routines have already been studied.
2. Certain problems is distributed computing can be solved by general constructions that yield a specify algo. when parameterized with a specific wave algo.

Three requirements satisfied by wave algo.

1. **Termination:** Each computation if finite

$$\forall |c| < \infty$$

2. **Decision:** Each computation contains at least one decide event

$$\forall C : \exists e \in C : e \text{ is a decide event}$$

3. **Dependence:** In each computation, each decide event is casually preceded by an event in each process.

$$\forall C : \forall C \in C : (\text{e is a decide event}$$

$$\forall_q \in p \exists f \in C_q, f \leq e)$$

(c) Write short Notes on

(i) CORBA and its services

Ans. CORBA: It is a middlewave design that allows application programs to communicate with one another irrespective of their programming language. Their harware and software platforms and network communicates implements.

Main components of CORBA architecture

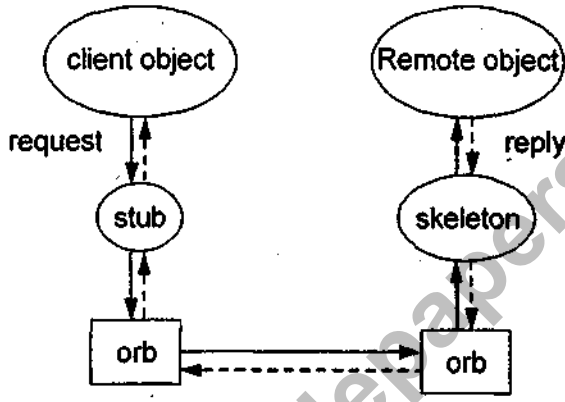
ORB Core: Its role is of communication module. It provides an interface that include operations enabling it to be started and stopped operations to convert between remote objects reference and strings.

Objects adapter: It bridge the gap between CORBA objects with IDL interface and programming languages interface skeleton: It is generated at server by IDL.

Client proxies: Stub procedures are generated at client by IDL interface. Client proxies marshals argument in invocation request and unmarshalls exception and results in replies.

Implementation repository: It is responsible for activating registered server in demand and for locating them.

Interface repository: Its role is to provide information about registered IDL interfaces to clients and server



Client host machine

Services:

1. **Naming services:**
   (a) register objects with a name
   (b) Look up objects by name
2. **Life cycle service**
   (a) Creating, copying, moving ,removing objects
3. **Concurrency control services**
   (a) Obtain and release exclusive locks
4. **Transaction service**
   (a) Two phase commit coordination
   (b) Supports nested transaction
5. **Persistence services**
   (a) Storing objects in a variety of data base
6. **Security services**
   (a) Authentication

(ii) Election based algorithm

Ans. Many distributed algo. require one process to act as a coordinator, initiator, sequence. Election algorithm is a procedure to find a coordinator to ensure that all the processes a agrees on whole will be new coordinators.

Properties of election algo:

1. Each process has same local algorithm
2. The algorithm is decentralized i.e a computation can be initiated by an arbitrary non empty subset of process.
3. The algo. reaches a turned configuration in each computation and in each reachable terminal configuration there is exactly one process in state leader and all other process are in state lost.

Bully election algo

Assume:

1. Reliable message delivery
2. System is synchronous
3. Each process knows which process have higher indentifier and can communicate with them

Other alogirthm is ping algorithm.