ANISH ADNANI
D17B/01

DC

ASSIGNMENT 1

1) A transparency is some aspect of the distributed system that is hidden from the user (programmer, system developer, user or application program). A transparency is provided by including some set of mechanism in the distributed system at a layer below the interface where the transparency is required. A number of basic transparency have been defined for a distributed system.

The transparencies are:

1. Access Transparency: There should be no apparent difference between local and remote access methods. In other words, explicit communication may be hidden. For instance, from a user point of view, access to a remote services such as a printer should be identical with access to a local printer. From a programmer point of view, the access method of a remote object may be identical to access of remote object of same class.

2. Location Transparency: The details of the topology of the system should be of no concern to the user. The location of an object in the system may not be visible to the user or programmer.

3. Concurrency Transparency: users and application should be able to access shared data or objects without interference about each other. This requires very complex mechanisms in a distributed system, since there exists true concurrency rather than simulated concurrency of a central system.

4. Replication Transparency: If the system provides replication (for availability or performance reasons) it should not concern the user.

5. Fault Transparency: If software or hardware failure occur, this should be hidden from the user

6. Migration Transparency: If objects (processes or data) migrate (to provide better performance, or, reliability, or to hide differences between hosts) this should be hidden from the users.

7. Performance Transparency: The configuration of the system should not be apparent to the user in terms of performance

8 Scaling Transparency: A system should be able to grow without affecting application programs. The application should have the ability of horizontal and vertical expandability.

2)

Components	Distributed OS		Network OS	Middleware OS
i) Definition	Tightly coupled operating system for multiprocessor and homogenous multicomputers		Loosely coupled operating system for heterogenous multicomputers	Additional layer on top of NOS implementing general purpose services
ii) Degree of transparency	Multiproc very high	Multicomp high	Low	High
iii) Same OS on all nodes	Yes	Yes	No	No
iv) Number of copies of OS	1	N	N	N
v) Basis for communication	Shared memory	Message	Files	Model specific
vi) Resource management	Global, central	Global, distributed	per Node	per Node
vii) Scalability	No	Moderate	Yes	Varies
viii) Openness	closed	closed	open	open

	Components	Distributed OS	Network OS	Middleware OS
ix)	Goals	Hides and manages hardware resources, transparent performance	Offer local services to remote clients	Provide distribution transparency, resolve heterogeneity
x)	Examples	Locus - can be accessed local and remote file simultaneously without any location hindrance. MiNOS, AIX	Microsoft Windows server 2003, UNIX, LINUX	Database middleware application server middleware, message oriented middleware, web middleware

3) Lightweight RPC

- Lightweight RPC is a communication facility that is designed and optimized for communication between protected domains in the same machine.
- LRPC simplify aspects of RPC such as control transfer, data transfer, linkage and stubs
- The communication traffic on operating system are of two types i.e cross domain that involves

communication between domains on the same machine and cross machine that involves communication between domains located on seperate machine.
- The LRPC is a facility designed and optimized for cross-domain communication where user level server processes have its own address space.
- Execution model of LRPC is borrowed from "protected procedure call" of capability systems
- LRPC is safe & transparent which represents a viable communication alternative for microkernel OS

To enhance the performance of conventional LRPC systems, the following techniques are used by LRPC.

i) Simple control Transfer
- LRPC : used a control transfer mechanism where a client's thread executes the requested procedure in the servers domain
- In this mechanism a client binds to a server interface before making its first call.

ii) simple Data Transfer
- In a cross domain RPC argument copying involves data to be copied four times
A) Stub to RPC message
B) client message to kernel
C) Kernel to server
D) Server to stack

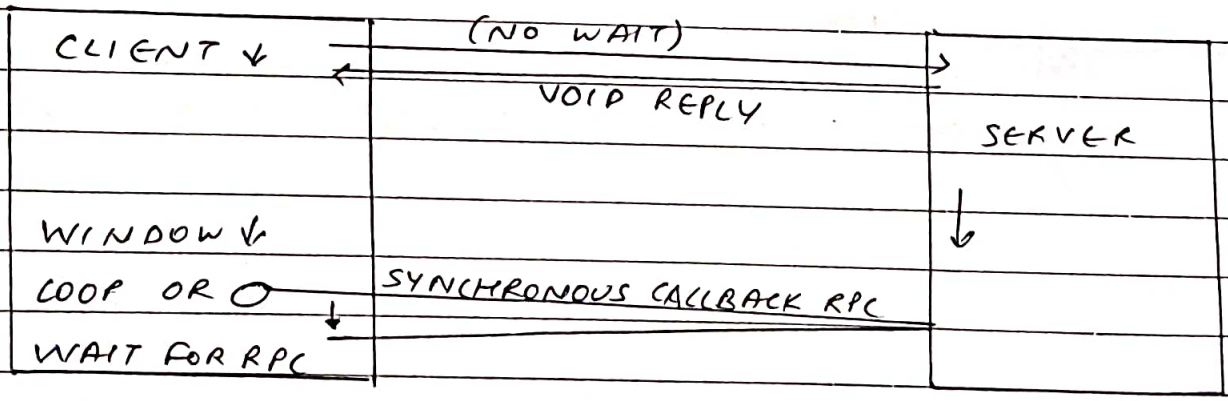
iii) Simple stub

- LRPC facilitates the generation of highly optimized stubs due to the use of control & data transfer model
- Every procedure has a call stub in the client domain and an entry stub in server domain

iv) Design for concurrency

- To achieve high call throughput and low call latency in LRPC having multiple processor which shared memory, special mechanism are used.

CALLBACK RPC



- In the usual RPC protocol, the caller and callee processes have a client-server relationship. Unlike this, the callback RPC facilitates a peer to peer diagram among the participating processes. It allows a process to be both client and server

- callback RPC facility is very useful in certain distributed application
- For example, remotely processed interactive applications that need user input from time to time under special conditions for further processing require this type of facility
- In such 48 applications, the client process makes an RPC to the concerned server process, and during procedure execution for the client, the server process makes a callback RPC to client process.
- The client process take necessary action based on the servers request and returns a replay for the callback RPC to the server process
- On receiving this reply, the server resumes the execution of the process and finally returns the result of the initial call to the client

4)

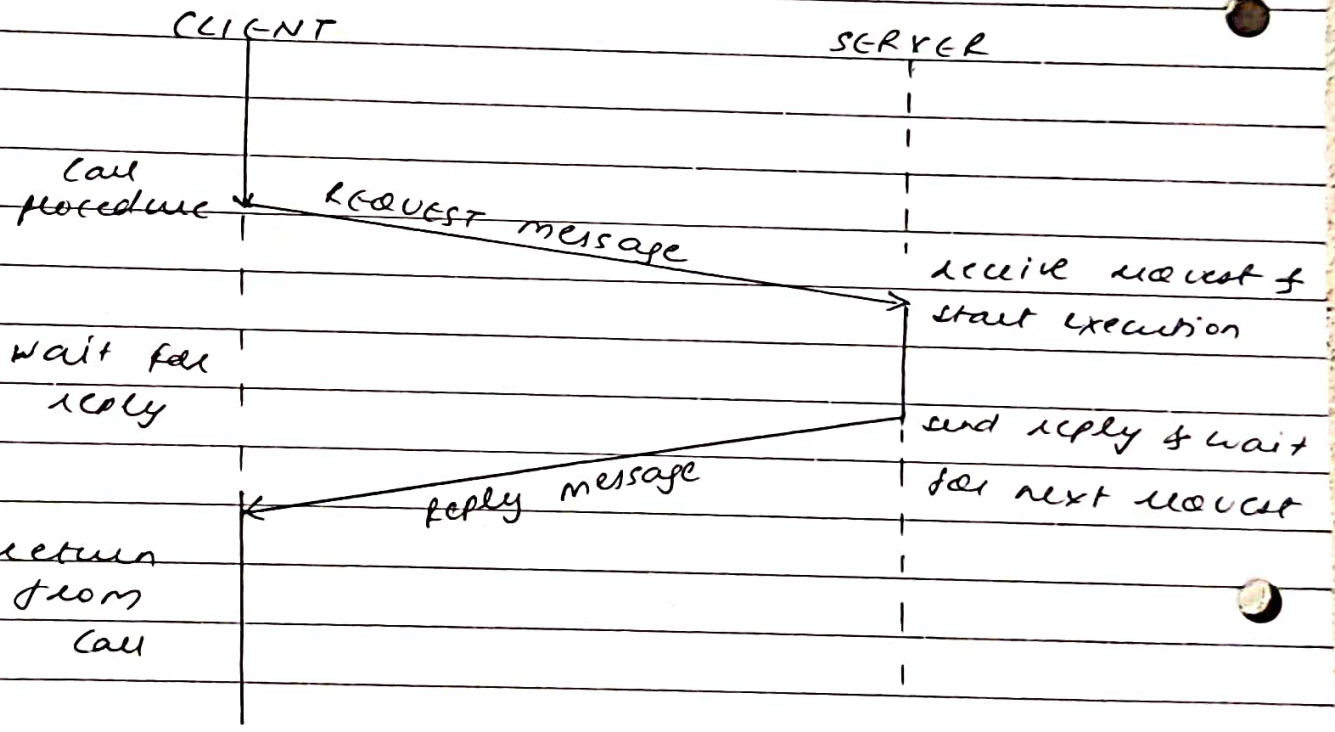
- A transparent RPC is one in which the local and remote procedure calls are indistinguishable to programmer
- The main issue in designing an RPC facility is its transparency property
- There are two types of transparency required

i) syntactic transparency: A remote procedure and a local procedure call should have the same syntax

ii) semantic transparency: The semantic of a

remote procedure call and local procedure are identical.

- Achieving syntactic transparency is not an issue but semantic transparency is difficult due to some differences between remote procedure calls and local procedure calls.



- The basic idea of RPC is to make a remote procedure call look transparent
- The calling procedure should not be aware that the called procedure is executing on a different machine
- RPC achieves transparency in an analogous way. When read is a remote procedure a different versions of read called client stub

is put in the library. It is called using the calling sequence

- The parameters are packed into a message and send to the server. When the message arrives at the server, the server OS passes it to server stub which calls receive & locks incoming messages
- The server unpacks parameters and calls server procedure and performs its work and returns the result to caller in the usual way of packing the result and sending it to the client stub
- When the client receives the message from server the OS of client sees that it is addressed to client process. When the caller gets control following the call to read, all it knows and that its data are available
- It has no idea that the work was done remotely instead of the local OS. In this way transparency is achieved

5) Messaging system assumes both side applications are running while communication is going on. message Queueing system permits processes to communicate although receiving side is not running at the time communication is initiated.
Communication servers are responsible for routing the messages between the hosts connected in network. Buffer is available at each host & communication server

user agent program at host submits message to host for transmission. The host then forwards this message to its local server. This mail server shares the received message in output buffer & look up transport layer address of destination mail server.
At destination mail server, message is put in input buffer in order to deliver to designated receiver.

* message sent by source host is buffered by communication server as long as it successfully delivers to next communication server. As so, it is not necessary for application to be executed when message was submitted. This form of communication is called as persistant-communication. E-mail is an example of persistant form of communication
* In transient communication system, the system stores the message if both sender & receiver applications are executing. otherwise, message is discarded. All the transport-level communication services offer transient communication
* In asynchronous communication server continues execution after submitting message for transmission. This sent message either remains in buffer of sending host or at first communication server.
* In synchronous communication sender is blocked until message is stored local buffer of destination host

6) RSVP is a transport layer protocol that is used to reserve resources in a computer network to get different quality of services (QoS) while accessing internet applications. It operates over internet protocol (IP) and initiates resources reservation from the receiver's end

Features

- RSVP is a receiver oriented signaling protocol. The receiver initiates and maintains resource reservation
- It is used both for unicasting (sending data from one source to one destination) and multicasting (sending data simultaneously to a group of destination computers)
- RSVP supports dynamic automatic adaptation to changes in network
- It provides a number of reservation styles. It also provides support for addition of future styles

RSVP messages

There are two types of RSVP messages

- Path messages (path): A path message is sent by the sender to all receivers by multicasting storing the path state at each node in its path. It stores the necessary information so that the receivers can make the reservation

- Reservation messages (resv) : The resv message is sent by the receiver to the sender along the reverse path of the path message. It identifies the resources that is required by the data flow.

An RSVP packet is very flexible; it can vary in size and in the number of data types and objects. Where packets need to travel through gateways that dont support RSVP, they can be "tunneled" through as ordinary packets.

7) Message ordering in group communication are as follows

1. unordered
2. FIFO
3. Total
4. Casual
5. Hybrid ordering such as Total-Casual & Total-FIFO

FIFO ordering

- Messages from a process should be delivered in the order in which they were sent

Solution: sender numbers the messages, receivers hold back those that have been received out of order.

Total ordering

messages from all processes should get a (unique) group wide ordering number, so all processes can deliver messages in single order.

implementation of total ordering consists of sequencer and ISIS algorithm

sequencer is logically external to the group

messages are sent to all members, including sequencer (initially have no "ordering" number)

sequencer maps messages identifiers to ordering numbers.

Casual ordering

Captures casual (cause and effect) relationships via happened before ordering

vector clocks ensure that replies are delivered after the message that they are replying to.

Casual ordering is not unique.

Hybrid ordering

casual ordering is not unique

- concurrent messages

FIFO is not unique

- FIFO only guarantees per process not inter-process

Total order only guarantees a unique order

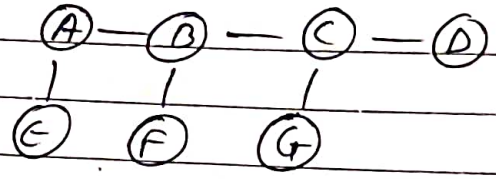
- combine with others to get stronger delivery semantics

8)

Parameters	Centralized Algorithm	Distributed Algorithm	Token ring Algorithm
1. Election	One process is elected as coordinator	Total ordering of all events in the systems	uses tokens for entering critical section
2. Messages per entry / exit	Requires 3 messages to enter and exit a critical region	Requires 2(n-1) messages	variable number of messages required
3. Delay in message times	Delay for messages is 2 messages	Delay for messages is 2(n-1)	The time varies from 0 to n-1 tokens
4. Mutual Exclusion	Guarantees mutual exclusion	mutual Exclusion guaranteed without deadlock	mutual Exclusion is guaranteed
5. Starvation	No starvation	No starvation	No starvation
6. Complexity	Easy to implement	complicated process	implementation is easy
7. Used for	used for general allocation	used for small group processes that do not change group membership	used processes in ring configuration

	parameters	Centralized Algorithm	Distributed Algorithm	Token ring Algorithm
8.	problems	Entire system can go down due to single point of failure	N points of failure	Detecting the lost token and regeneration is difficult
9.	Expense	Less expensive	More expensive	Less expensive
10.	Robustness	More robust	Less robust	More robust

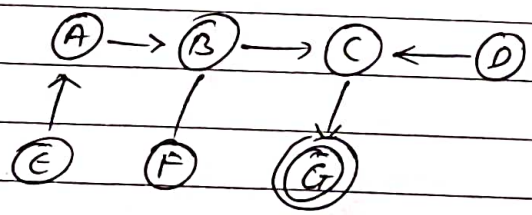
9) Raymonds Tree Based algorithm uses a spanning tree to reduce the number of messages exchanged per critical section execution.
The network is viewed as a graph, a spanning tree of a network is a tree that contains all the N nodes.
The algorithm assumes that the underlying network guarantees message delivery. All nodes of the network are completely reliable. The algorithm operates on a minimal spanning tree of the network topology or a logical structure imposed on the network.
The algorithm assumes the network nodes to be arranged in an unrooted tree structure.



A node needs to hold information about and communicate only to its immediate-neighbouring nodes. The concept of tokens used in token based algorithms is of privilege.

Only one node can be in possession of privilege, it remains in possession of the node that last used it.

The HOLDER variable each node maintains a HOLDER variable that provides information about the placement of the privilege in relation to the node itself. A node stores in its HOLDER variable to the identity of a node that it thinks has the privilege or leads to the node having the privilege.



Now suppose Node B that does not hold the privilege wants to execute the critical section. B sends a REQUEST message to HOLDER B, i.e, C which in turn forwards the REQUEST message to HOLDER, i.e, G. The privileged node G if no longer needs the privilege, sends the privilege message to its neighbour C and is continued.

ASSIGN PRIVILEGE

This is a routine sends a PRIVILEGE message. A privileged node sends a PRIVILEGE message if it holds the privilege but is not using it, it REQUESTS Q is not empty, and the element at the head of its REQUEST Q is not "self"

MAKE REQUEST

This is a routine sends a request message. An unprivileged node sends a REQUEST message if it does not hold the privilege, its REQUEST Q is not empty, i.e, it requires the privilege for itself, or on behalf of one of its immediate neighbouring nodes, and it has not sent a REQUEST message already.

10) Clock drift refers to several related phenomena where a clock does not run at exactly at the same rate as a reference clock. That is, after some time the clock "drift apart" or gradually desynchronizes from the other clock. All clocks are subject to drift, causing eventual divergence unless resynchronized.

Logical clocks refer to implementing a protocol on all machines within your distributed systems, so that the machines are able to maintain consistent ordering of events within some virtual timespan.

A logical clock is a mechanism for capturing chronological and casual relationships in a distributed system. Distributed systems may have no physically synchronous global clock, so a logical clock allows global ordering on events from different processes in such systems.

For example if we have more than 10 PCs in a distributed system & every PC is doing its own work but then how we make them work together. Hence a logical clock is required

Method 1

To order events across processes, try to sync clock in 1 approach. If one PC has time 2:00 pm then every PC should have the same time which is quite not possible. Hence, we cannot follow this method.

Method 2

This assigns Timestamps to events. If we give each PC an individual number, then it will be organized in a way that 1st PC will complete its process first, then second & so on. But timestamps work as long as they obey casuality.