1.TITLE: RMI

OBJECTIVE: To study:- 1.Multi-threading 2.How Remote Method Invocation works.

3.How RMI allows objects to invoke methods on remote objects.

4.How to write Distributed Object Application.

TOOLS: S/W: 1. Can be developed on any platform Windows /Linux.

2.Java Language, Java Programming Environment, rmiregistry, jdk

H/W: Any basic configuration loaded machine

Thread: A thread of execution is the smallest sequence of programmed instructions that can be managed independently by a scheduler, which is typically a part of the operating system.

1.RMI:-

1.RMI provides communication between java applications that are deployed on different servers and connected remotely using objects called stub and skeleton.

2.This communication architecture makes a distributed application seem like a group of objects communicating across a remote connection.

3.These objects are encapsulated by exposing an interface, which helps access the private state and behavior of an object through its methods.

2.RMI Registry: It is a remote object registry, a Bootstrap naming service, that is used by RMI SERVER on the same host to bind remote objects to names.

3.Remote object: This is an object in a specific JVM whose methods are exposed so they could be invoked by another program deployed on a different JVM.

4.Remote interface: This is a Java interface that defines the methods that exist in a remote object.

5.RMI: This is a way of invoking a remote object's methods with the help of a remote interface.

6.Stub: This is a Java object that acts as an entry point for the client object to route any outgoing requests

7.Skeleton: This is an object that behaves like a gateway on the server side. It acts as a remote object with which the client objects interact through the stub.

2.TITLE: CORBA

OBJECTIVE: Students should be able to understand:- 1. Common Object Request Broker Architecture 2.Inter-ORB communication 3.Java Support for CORBA

TOOLS : S/W: 1. Can be developed on any platform Windows /Linux.

2.Java Language, Java Programming Environment, rmiregistry, jdk

H/W: Any basic configuration loaded machine (e.g. P IV)

CORBA : 1.It is a standard defined by the Object Management Group (OMG) that enables software components written in multiple computer languages and running on multiple computers to work together.

2. The following diagram shows a single-process ORB CORBA architecture with the IDL configured as client stubs with object skeletons

3. The client and server use stubs and skeletons as proxies, respectively.

4. In CORBA, each object instance acquires an object reference for itself with the electronic token identifier

5. Client invocations are going to use these object references that have the ability to figure out which ORB instance they are supposed to interact with

6. The stub and skeleton represent the client and server, respectively, to their counterparts.

***To run this client-server application on the development machine:**

1. Change to the directory that contains the file Hello.idl.

2. Run the IDL-to-Java compiler, idlj, on the IDL file to create stubs and skeletons. This step assumes that you have included the path to the java/bin directory in your path.

```
idlj -fallHello.idl
```

3. Compile the .java files, including the stubs and skeletons . This step assumes the java/bin directory is included in your path.

4. Start orbd.

To start orbd from a UNIX command shell, enter: orbd -ORBInitialPort 1050&

5. Start the HelloServer:

To start the HelloServer from a UNIX command shell, enter:

java HelloServer -ORBInitialPort 1050 -ORBInitialHost localhost&

6. Run the client application:

java HelloClient -ORBInitialPort 1050 -ORBInitialHost localhost

3.TITLE: MPI

OBJECTIVE: Students should be able to understand:- 1. Basic concept of Message Passing Interface. 2. Communication using MPI. 3.Concept of OpenMPI.

TOOLS : S/W - Fedora/ Ubantu. Openmpi, Terminal, CC, editor·

H/W: Any basic configuration loaded machine (e.g. P IV)

MPI: It is a standardized means of exchanging messages between multiple computers running a parallel program across distributed memory. MPI addresses primarily the message-passing parallel programming model, in which data is moved from the address space of one process to that of another process through cooperative operations on each process.

An Interface Specification:

1.M P I = Message Passing Interface

2. MPI is a specification for the developers and users of message passing libraries. By itself, it is NOT a library - but rather the specification of what such a library should be. · MPI primarily addresses the message-passing parallel programming model: data is moved from the address space of one process to that of another process through cooperative operations on each process.

3.Simply stated, the goal of the Message Passing Interface is to provide a widely used standard for writing message passing programs. The interface attempts to be:

I.Portable

Ii.Efficient

Iii.lexible

iv. Practical

4. The MPI standard has gone through a number of revisions, with the most recent version being MPI 3.

5.Interface specifications have been defined for C and Fortran90 language bindings:

C++bindings from MPI-1 are removed in MPI-3

5.MPI-3 also provides support for Fortran 2003 and 2008 features

6. MPI implementers adapted their libraries to handle both types of underlying memory architectures seamlessly.

Basic MPI Operations:

`MPI_Comm`	the basic object used by MPI to determine which processes are involved in a communication
`MPI_Status`	the `MPI_Recv` operation takes the address of an `MPI_Status` structure as an argument (which can be ignored with `MPI_STATUS_IGNORE`).
`MPI_Init`	Initialize the MPI execution environment `int MPI_Init( int *argc, char ***argv )`
`MPI_Comm_size`	Determines the size of the group associated with a communicator `int MPI_Comm_size( MPI_Comm comm, int *size )`
`MPI_Send`	`MPI_Send` performs a standard-mode, blocking send.
`MPI_Close_port`	MPI_Close_port releases the network address represented by `port_name`.

4.TITLE: BERKELEY ALGORITHM

OBJECTIVE:

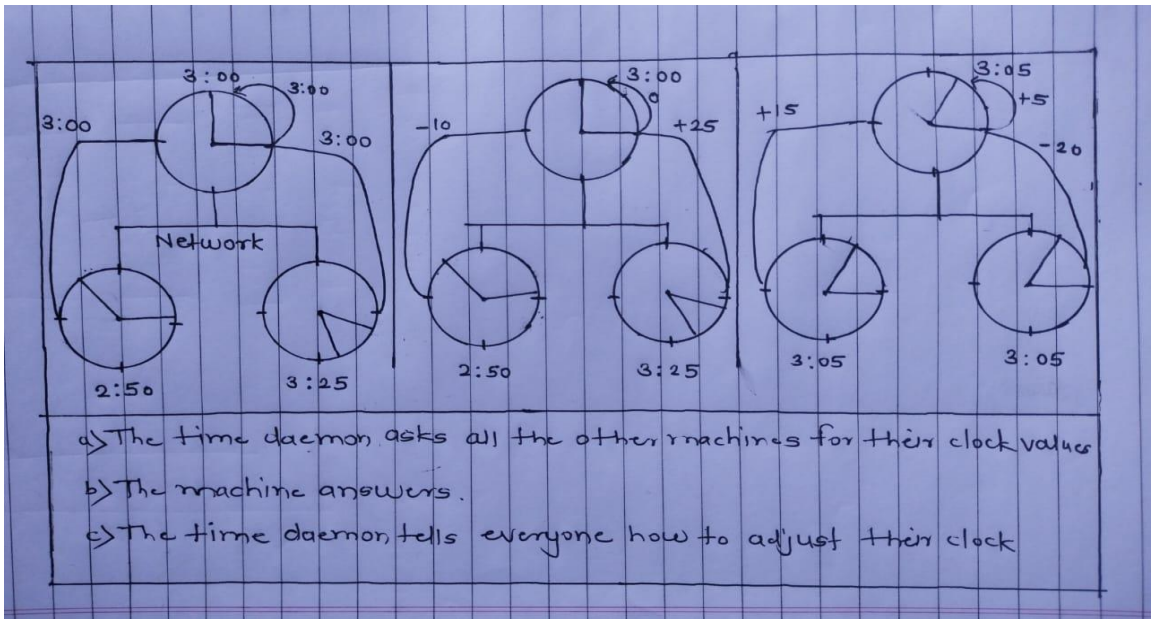
Students should be able to understand:- 1.Basic concept of Clock. 2.ClockSynchronization. 3.Berkeley algorithm for clock synchronization

The Berkeley algorithm

1.Here the time server (actually, a time daemon) is active, polling every machine from time to time to ask what time it is there. Based on the answers, it computes an average time and tells all the other machines to advance their clocks to the new time or slow their clocks down until some specified reduction has been achieved.

2. This method is suitable for a system in which no machine has a UTC receiver.

3. In Figure at 3:00, the time daemon tells the other machines its time and asks for theirs. In they respond with how far ahead or behind the time daemon they are. Armed with these numbers, the time daemon computes the average and tells each machine how to adjust its clock.



4. Note that for many purposes, it is sufficient that all machines agree on the same time. It is not essential that this time also agrees with the real time as announced on the radio every hour.

5. If in our example of Figure 6.6 the time daemon's clock would never be manually calibrated, no harm is done provided none of the other nodes communicates with external computers.

6. Everyone will just happily agree on a current time, without that value having any relation with reality

7. The Berkeley algorithm is thus typically an internal clock synchronization algorithm.

Example:-

Let's sum-up steps followed to synchronize the clock using the Berkeley algorithm, Nodes in the distributed system with their clock timings −

N1 -> 14:00 (master node)

N2 -> 13: 46

N3 -> 14: 15

Step 1 – The Leader is elected, node N1 is the master in the system.

Step 2 – leader requests for time from all nodes.

N1 -> time : 14:00

N2 -> time : 13:46

N3 -> time : 14:20

Step 3 – The leader averages the times and sends the correction time back to the nodes.

N1 -> Corrected Time 14:02 (+2)

N2 -> Corrected Time 14:02 (+16)

N3 -> Corrected Time 14:02 (-18)

5.TITLE: TOKEN RING

OBJECTIVE: Students should be able to understand:- 1.Basic concept of Mutual exclusion. 2.Algorithm for Mutual exclusion. 3. Token-Ring Algorithm.

Mutual exclusion: Sometimes processes will need to simultaneously access the same resources. To prevent that such concurrent accesses corrupt the resource, or make it inconsistent, solutions are needed to grant mutual exclusive access by processes.

Token-Ring Algorithm:-

Essence of Token-Ring Algorithm: Organize processes in a logical ring, and let a token be passed between them. The one that holds the token is allowed to enter the critical region (if it wants to)

(a) An unordered group of processes on a network.

(b) A logical ring constructed in software.

When the ring is initialized, process 0 is given a token.

1.The token circulates around the ring.

2.It is passed from process k to process k+1 (modulo the ring size) in point-to-point messages.

3.When a process acquires the token from its neighbor, it checks to see if it needs to access the shared resource.

I.If so, the process goes ahead, does all the work it needs to, and releases the resources.

ii.it has finished, it passes the token along the ring.

Iii.It is not permitted to immediately enter the resource again using the same token.

4.If a process is handed the token by its neighbor and is not interested in the resource, it just passes the token along.

5 **.** As a consequence, when no processes need the resource, the token just circulates at high speed around the ring.

Advantages:

1.Only one process has the token at any instant, so only one process can actually get to the resource.

2.Since the token circulates among the processes in a well-defined order, starvation cannot occur.

3.Once a process decides it wants to have access to the resource, at worst it will have to wait for every other process to use the resource

6.TITLE: ELECTION

OBJECTIVE: Students should be able to understand:- 1.Election algorithms 2.The Bully Algorithm 3.The Ring Algorithm

Distributed Algorithm is a algorithm that runs on a distributed system. Distributed system is a collection of independent computers that do not share their memory. Each processor has its own memory and they communicate via communication networks

We have two election algorithms for two different configurations of distributed system.

A. The Bully Algorithm – This algorithm applies to system where every process can send a message to every other process in the system.

Algorithm – Suppose process P sends a message to the coordinator.

1. If coordinator does not respond to it within a time interval T, then it is assumed that coordinator has failed.

2. Now process P sends election message to every process with high priority number. 3. It waits for responses, if no one responds for time interval T then process P elects itself as a coordinator.

4. Then it sends a message to all lower priority number processes that it is elected as their new coordinator.

5. However, if an answer is received within time T from any other process Q, a. Process P again waits for time interval T' to receive another message from Q that it has been elected as coordinator.

b. If Q doesn't responds within time interval T' then it is assumed to have failed and algorithm is restarted.

B. The Ring Algorithm

1. If process P1 detects a coordinator failure, it creates new active list which is empty initially. It sends election message to its neighbor on right and adds number 1 to its active list.

2. If process P2 receives message elect from processes on left, it responds in 3 ways: a. If message received does not contain 1 in active list then P1 adds 2 to its active list and forwards the message.

b. If this is the first election message it has received or sent, P1 creates new active list with numbers 1 and 2. It then sends election message 1 followed by 2. c. If Process P1 receives its own election message 1 then active list for P1 now contains numbers of all the active processes in the system. Now Process P1 detects highest priority number from list and elects it as the new coordinator.

7.TITLE: WEB SERVICE

OBJECTIVE: Students should be able to understand:- 1· Web service 2. Web service architectures

TOOLS : · S/W: 1. Can be developed on any platform Windows /Linux. 2.Java Programming Environment, JDK 8, Netbeans IDE with GlassFish Server

H/W: Any basic configuration loaded machine (e.g. P IV)

Web Service:

A web service can be defined as a collection of open protocols and standards for exchanging information among systems or applications.

A service can be treated as a web service if:

1. The service is discoverable through a simple lookup

2.It uses a standard XML format for messaging

3.It is available across internet/intranet networks.

4. It is a self-describing service through a simple XML syntax

5.The service is open to, and not tied to, any operating system/programming language

Types of Web Services: 1. SOAP: SOAP stands for Simple Object Access Protocol. SOAP is an XML based industry standard protocol for designing and developing web services. Since it's XML based, it's platform and language independent. So, our server can be based on JAVA and client can be on .NET, PHP etc. and vice versa.

2. REST: REST (Representational State Transfer) is an architectural style for developing web services. It's getting popularity recently because it has small learning curve when compared to SOAP. Resources are core concepts of Restful web services and they are uniquely identified by their URIs.

Web service architectures:

As part of a web service architecture, there exist three major roles.

1. Service Provider is the program that implements the service agreed for the web service and exposes the service over the internet/intranet for other applications to interact with.

2. Service Requestor is the program that interacts with the web service exposed by the Service Provider. It makes an invocation to the web service over the network to the Service Provider and exchanges information.

3. Service Registry acts as the directory to store references to the web services.