Institute of Information Technology

Noakhali Science and Technology University

Course Title: Distributed System Lab Course Code: CSE3202

Introduction to message passing technology and applications

1) Write a server (TCP) Program that opens a listening socket and waits to serve client.

2)Write a client (TCP) program that connects with the server program that connects with the server program knowing IP address and port number.

3)Get the string from console on client and send it to server, server echoes back that string to client.

4) Write a server Program using Shared memory and semaphore (server increments counter between sem_wait and sem_post). Create shared memory using mmap.

5) Write a client Program that reads counter value between sem_wait and sem_post. Access shared memory using open.

Sockets Programming

1) Write a server (TCP) Program that opens a listening socket and waits to serve client
2) Write a client (TCP) Program that connects with the server program knowing IP address and port number.
3) Get the input string from console on client and send it to server, server echoes back that string to client.
4) Write a server (UDP) Program that waits in recvfrom
5) Write a client (UDP) Program that calls sendto to send string to server program knowing IP address and port number.
6) Server replies current date and time (using time, and ctime calls) to client.

Remote Invocation

1) Program to implement Remote Method Invocation

Sequence of events during a RMI:

- ✓ The client calls the client stub. The call is a local procedure call, with parameters pushed on to the stack in the normal way.
- ✓ The client stub packs the parameters into a message and makes a system call to send the message. Packing the parameters is called marshalling.
- ✓ The kernel sends the message from the client machine to the server machine.
- ✓ The kernel on the server machine passes the incoming packets to the server stub.
- ✓ Finally, the server stub calls the server procedure. The reply traces the same steps in the reverse direction.

2) Program to implement Remote Procedure Call.

Indirect Communication

Group Communication in distributed Systems

Communication between two processes in a distributed system is required to exchange various data, such as code or a file, between the processes. When one source process tries to communicate with multiple processes at once, it is called Group Communication. A group is a collection of interconnected processes with abstraction. This abstraction is to hide the message passing so that the communication looks like a normal procedure call. Group communication also helps the processes from different hosts to work together and perform operations in a synchronized manner, therefore increases the overall performance of the system.

Types of Group Communication in a Distributed System:

Broadcast Communication:

When the host process tries to communicate with every process in a distributed system at same time. Broadcast communication comes in handy when a common stream of information is to be delivered to each and every process in most efficient manner possible. Since it does not require any processing whatsoever, communication is very fast in comparison to other modes of communication. However, it does not support a large number of processes and cannot treat a specific process individually.

Multicast Communication:

When the host process tries to communicate with a designated group of processes in a distributed system at the same time. This technique is mainly used to find a way to address problem of a high workload on host system and redundant information from process in system. Multitasking can significantly decrease time taken for message handling.

Unicast Communication:

When the host process tries to communicate with a single process in a distributed system at the same time. Although, same information may be passed to multiple processes. This works best for two processes communicating as only it has to treat a specific process only. However, it leads to overheads as it has to find exact process and then exchange information/data.

1. Implement java Unicast Group Communication
2. Implement java Multicast Group Communication
3. Implement java Broadcast Group Communication
4. Implement Java messaging service (JMS)

Distributed Objects and Components

CORBA

Aim: Create CORBA based server-client application.

Tools/ Apparatus: CORBA Runtime: inbuilt in Java Standard Edition Runtime JDK

Procedure:

1) Define the remote interface : Write IDL file

2) Compile the remote interface : Mapping IDL file to platform specific files

C:\...>idlj -fall Hello.idl

3) Implement the server

4) Implement Client

5) Start name service

start orbd -ORBInitialPort 1050 -ORBInitialHost localhost

6) Start server

start java HelloServer -ORBInitialPort 1050 -ORBInitialHost

localhost

7) Start client

java HelloClient -ORBInitialPort 1050 -ORBInitialHost localhost

Time and Global states

1. Implement Lamport's Logical Clock Algorithm
2. The design and implementation of a sequencer multicast protocol using Java

Coordination and agreement

1. Implement the Token based mutual exclusion algorithm in Distributed system.
2. Implement Bully election algorithm in distributed system
3. Write a code for ring-based algorithm for distributed system.

Distributed Transactions

Define two problems by yourself and solve that two problem.