Sessional Assignment

Course: - Distributed Computing

Deadline: - Mentioned on SIC **Marks: - 20**

Program: - MS (CS) **Dated: 15 May 2020**

Student Name: Masood Ur Rehman **Student ID#: 14354**

Class and Section: MS(CS) 4rth Semmester

Question: **Assume you have a Client Server Environment in which the client requests the server to multiply three given number i.e 67, 90, 34, and return the result. Discuss the steps of the system in each of the following scenarios.**

a) How the Request-Reply Protocols functions will be used with UDP (refer to figure 5.3 in book), how will be the message identifiers used, what will be its failure model, how time outs will be used, how will the system handle duplicate messages and how will the system react if reply is lost. (8)

Ans: In Request-Reply protocol the **"*do-operation*"** will be used by client to send the given numbers and the operation to be performed to the server in order to invoke remote operation on it, **"*get-request*"** will be used by the server to do the specified operation on the given numbers and then the server uses **"*send-reply*"** function to send the reply message with result to the client. . i.e:

Message Type (i.e 111),
OperationId (i.e 222),
Arguments (values to be multiplied i.e 67,90,34)
Remote reference (IP Address of Server)
Request id (i.e 555)
public byte[] doOperation (111,555,192.168.1.1: 3030,222,[67,90,34])
public byte[] getRequest (3030)
public void sendReply (byte[951] reply, 192.168.1.2,357).

Message Identifiers: For reliable message delivery message identifiers will be used in which **"*requestId*"** is used to make the identifier unique to the sender and its port and internet address is used to make it unique in the distributed system.

Failure Model: By the failure of message the ***"do-operation"*** will use a timeout when it waits for server reply, for timeouts the ***"do-operation"*** is returned to the client with the indication that the ***"do-operation"*** has failed.

Timeouts: The request-message will be sent repeatedly by the ***doOperation*** until it gets a reply to make it sure that the delay is due to the lack of response from the server side and not to lost the message.

Duplicate Messages: To avoid duplicate messages the protocol is designed to recognize successive messages (from the same client) with the request identifier and to filter out duplicates.

Reply lost: If the reply is lost an ***"idempotent"*** operation will be performed repeatedly with the same effect as if it had been performed exactly once.

b) How the above system can be implemented using Remote Procedure Calls (RPC)? (Hint: Read Section 5.3.2 in the book). (6)

Ans: Remote Procedure calls (RPC) is generally implemented over a Request-Reply Protocol. The above system will be implemented by converting the given numbers and multiply operation into a client stub procedure then the stub procedure will marshal the given numbers and identifier into a request message, which will be sent through a communication module to the server. When the request message arrives to the server the dispatcher on server side will select the stub procedure according to the procedure identifier in the request message. The server stub procedure then un-marshals the given numbers, calls the required multiplication procedure and then marshals the return value for the reply message: i.e

- The client having IP Address i.e (192.168.1.2) calls the local stub procedure with arguments (67, 90 and 34).
- The client stub procedure will marshal the arguments(67,90,34) and multiply operation into a message and will send the message to the server on server IP Address i.e (192.168.1.1).
- The O/S of server will put the message in server Stub Procedure. Server will un marshal the message and will get the three numbers and its multiply operation. The server will implement the required actions and will send the reply back to the client following the same steps but in reverse format.

c) How the above system can be implemented using Remote Method Invocation (RMI)?

(Hint: Read Section 5.4.2 in the book). (6)

Ans: Remote Method Invocation generally uses object oriented Programming concept. The two cooperating communication modules carry out the request-reply protocol.

- The above three numbers and it's multiply operation along with **requestId** and the remote reference of the object to be invoked will be sent from client side.
- After that marshalling and un marshalling of the message and its result is the concern of RMI software. The communication modules will provide the semantics of the expression. i.e (67*90*34=?).
- On the server side the communication module will select the dispatcher for the class of the object to be invoked, passing on its local reference, which it gets from the remote reference module in return for the remote object identifier in the request message.
- In RMI the first object will contain the information of the request message and the second object will contain the reference of the first object pretending the first local object as remote object and this process will be reversed on server side to reply the result. i.e:

Obj1 will contain (67*90*34) and Obj2 will contain (Ref:Obj1)