Important Instructions:

1) **Open this MS-Word document and start writing answers below each respective question given on page 2.**
2) **Answers the question in the same sequence in which they appear.**
3) **Provide to the point and concrete answers.**
4) **First read the questions and understand what is required of you before writing the answer.**
5) **Attempt the paper yourself and do not copy from your friends or the Internet. Students with exactly similar answers or copy paste from the Internet will not get any marks for their assignment.**
6) **You can contact me for help if you have any doubt in the above instructions or the assignment questions.**
7) **All questions must be attempted.**
8) **Do not forget to write your name, university ID, class and section information.**
9) **Rename you answer file with your university ID# before uploading to SIC.**
10) **When you are finished with writing your answers and are ready to submit your answer, convert it to PDF (no MS Word) and upload it to SIC unzipped, before the deadline mentioned on SIC.**
11) **Do not make any changes to the format provided.**
12) **Failure in following the above instructions might result in deduction of marks.**

Sessional Assignment
Course: - Distributed Computing

Deadline: - 05 June, 2020 **Marks: - 20**

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

Student Name: RAHMAT ULLAH **Student ID#:**14334

Class and Section: MS CS, 4th Semester

Question: Assume you have a Client Server Environment in which the client request 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)

Answer:

Use of Request-Replay Protocols functions with UDP:

Request-reply protocols will be used to provide lightweight and minimal support for client-server computing with UDP. With reference to the figure, it illustrates that the client sends the request of multiplying the three numbers to the server and returns the reply. The server acquires the client's request of multiplying the numbers via the server port. Finally, the server sends the reply message reply, which is the result of the multiplication of the three numbers, to the client at its Internet address and port.

How will the message identifiers be used?

Message identifiers will be used by using a request Id, which is taken from an increasing sequence of integers by the sending process and an identifier for the sender process, for example, its port and Internet address. Request Id makes the identifier unique to the sender and an Identifier makes it unique in the distributed system.

What will be its failure model?

The failure model can be of:

- Suffering from omission failures.
- Messages are not guaranteed to be delivered in sender order
- The protocol can suffer from the failure of processes
- Processes may have crash failures
- There may be occasions when a server has failed or a request or reply message is dropped

How time outs will be used?

Time outs will be used to allow for occasions when a server has failed or a request or reply message is dropped. The doOperation uses a timeout when it is waiting to get the server's reply message. The action taken when a timeout occurs depends upon the delivery guarantees being offered. There are various options as to what doOperation can do after a timeout. The simplest option is to return immediately from doOperation with an indication to the client that the doOperation has failed.

How will the system handle duplicate messages?

To avoid duplicate messages, Request-Replay Protocol is designed to recognize successive messages (from the same client) with the same request identifier and to filter out duplicates. If the server has not yet sent the reply, it need take no special action – it will transmit the reply when it has finished executing the operation.

How will the system react if reply is lost?

The system will react normally and will regenerate the same result by redoing the operation. If the server has already sent the reply when it receives a duplicate request it will need to execute the operation again to obtain the result. The system will recalculate the multiplication of the given three numbers after the timeout.

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

Answer:

Remote Procedure Call is generally implemented over a request-reply protocol. The communication module will implement the desired design choices in terms of retransmission of requests, dealing with duplicates and retransmission of results.

Using RPC for the above system, the client that accesses a service includes one stub procedure for each procedure in the service interface. The stub procedure behaves like a local procedure to the client, but instead of executing the call, it marshals the procedure identifier and the arguments into a request message, which it sends via its communication module to the server. When the reply message arrives, it unmarshals the results. The server process contains a dispatcher together with one server stub procedure and one service procedure for each procedure in the service interface. The dispatcher selects one of the server stub procedures according to the procedure identifier in the request message. The server stub procedure then unmarshals the arguments in the request message, calls the corresponding service procedure and marshals the return values for the reply message.

c) How can the above system be implemented using Remote Method Invocation (RMI)? (Hint: Read Section 5.4.2 in the book). (6)

Answer:

The above system can be implemented by using Remote Method Invocation, involving several separate objects and modules. An application-level object A invokes a method in a remote application-level object B for which it holds a remote object reference. This is achieved by the communication and remote reference modules and then with the RMI software that runs over them. The two cooperating communication modules carry out the request-reply protocol, which transmits request and reply messages between the client and server. The contents of request in this situation are the three numbers and reply message is the product of these numbers. The communication module uses only the first three items, which specify the message type, its request and the remote reference of the object to be invoked. The operationId and all the marshalling and unmarshalling are the concern of the RMI software. The communication modules are together responsible for providing a specified invocation semantics. The

communication module in the server selects 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 which includes the calculation of given numbers.