Sessional Assignment
Course: - Distributed Computing

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

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

Student Name: **AQEEL KHAN** **Student ID#:** **5953**

Class and Section **MS CS**

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 : UDP provides simple datagram delivery to remote sockets, that is, to ⟨host, port⟩ pairs. TCP provides a much richer functionality for sending data, but requires that the remote socket first be connected. In this chapter, we start with the much-simpler UDP, including the UDP-based Trivial File Transfer Protocol. UDP is unreliable, in that there is no UDP-layer attempt at timeouts, acknowledgment and retransmission; applications written for UDP must implement these. As with TCP, a UDP ⟨host, port⟩ pair is known as a socket (though UDP ports are considered a separate namespace from TCP ports). UDP is also unconnected, or stateless; if an application has opened a port on a host, any other host on the Internet may deliver packets to that ⟨host, port⟩ socket without preliminary negotiation.

Examples of UDP

include Voice over IP (VoIP), online games, and media streaming. Speed – **UDP's** speed makes it useful for query-response protocols such as DNS, in which data packets are small and transactional.

- **HOW will the system handle duplicate messages and how will the system react if reply is lost.**

It developing a protocol over UDP to be used in a local network, there will be only a switch (Cisco, 3com, etc) between source and destination, both Linux systems, same MTU. How often should I expect udp packets to be duplicated (obviously not by me but by the switch or maybe the server) at the destination in this case? I need this to know if to implement a duplication check in my code or not.

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

Answer : **Remote Procedure Call (RPC)** is a powerful technique for constructing distributed, client-server based applications. It is based on extending the conventional local procedure calling so that the called procedure need not exist in the same address space as the calling procedure. The two processes may be on the same system, or they may be on different systems with a network connecting them.

RPC Works. An RPC is analogous to a function call. Like a function call, when an RPC is made, the calling arguments are passed to the remote procedure and the caller waits for a response to be returned from the remote procedure. ... The client makes a procedure call that sends a request to the server and waits

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

Answer : RMI is used when we have to invoke methods from distance on remote objects (these objects are located on other systems). RMI is very limited (is the single thing which is able to do it) and it give us a platform-independent understanding. When we start to use RMI, the programming of streams and sockets disappear. By having the objects remotely stored somewhere, the access to them becomes very transparent for the programmer. I would like to mention from start that the article is for those (students and not only) who wish to understand the basic principles of accessing objects remotely. The examples from here are based on original.

- **Package is used for remote method invocation RMI**

RMI creates a public remote server object that enables client and server side communications packages through simple method calls on the server object. The communication packages between client and server is handled by using two intermediate objects: Stub object (on client side) and Skeleton object.