CS5352.0501/0502, Distributed Computing, Summer I, 2009
Assignment 3

Issued: 06/29/2009
Due: 07/06/2009

Modify/enhance the Java UDP program that you wrote in Assignment 2 as follows (The main intention is to upgrade it to an RMI application. But as RMI is not introduced as yet, it cannot be done in this assignment).

1 Introduction

We now have multiple clients, instead of single client in Assignment 2. For simplicity, in this assignment you can focus on the case of two clients. The clients and server still communicate with each other using UDP in a stop-and-go fashion (i.e. a client will not send another new request message to the server until a correct reply for previous request message has been received). But now they implement the RR protocol semantics. Namely, for each request message sent from a client, the server will send a reply.

2 UDP simulation

Because UDP is unreliable, some requests/replies may be lost. Therefore the clients have to use some timeout mechanism to deal with loss/corruption of messages. However, if you run a client and server on two different computers on the same Ethernet segment, most likely each UDP datagram will be delivered reliably. Therefore a method of simulating message losses is used.

Specifically, for each client, the *server* will read a file that consists of a list of non-negative integers.

> If the ith integer in the list has value j, then the ith request message from that specific client will have to be retransmitted j times.

For example, if the first integer in the list is zero, then the first request message is assumed to be reliable and will not be lost. Furthermore the reply to the first message is also assumed reliable. If the second integer in the list is 2, then the second request message is assumed to be un-reliable and will be lost twice. When the first copy of the second request message arrives at the server, the server will discard it. The server will also discard the second copy of the second request message. Therefore the client is forced to send the second request message three times!

3 Timeout and its handling

As we use Java2 in our class, you can use the *SocketOptions* interface to set timeout value on a socket. Once a timeout occurs, an exception will

be thrown. Your code in the *cache* section then retransmits the request message.

You can estimate the RTT (round-trip time) between the clients and server in many different ways. However, because the clients and server are on the same LAN, the RTT is normally very short. As Java allows timeout values in unit of milliseconds, you can set your timeout as 10 milliseconds.

4 Structure of messages

For simplicity, we assume that the server will not do any handling after receiving a request message from a client. Namely, the client can send a dummy request message to server.

As we use stop-and-go protocol (also called idel RQ in networking terminologoes) for message exchanges, your request messages have to use two sequence numbers to differentiate the current request message with a previous message. The server's replies are simple: they only contain an integer that denotes the sequence number of the request message the reply is for.