5.5 A request-reply protocol is implemented over a communication service with omission failures to provide at-least-once RMI invocation semantics. In the first case the implementor assumes an asynchronous distributed system. In the second case the implementor assumes that the maximum time for the communication and the execution of a remote method is \( T \). In what way does the latter assumption simplify the implementation?

5.5 Ans.

In the first case, the implementor assumes that if the client observes an omission failure it cannot tell whether it is due to loss of the request or reply message, to the server having crashed or having taken longer than usual. Therefore when the request is re-transmitted the client may receive late replies to the original request. The implementation must deal with this.

In the second case, an omission failure observed by the client cannot be due to the server taking too long. Therefore when the request is re-transmitted after time \( T \), it is certain that a late reply will not come from the server. There is no need to deal with late replies.

6.16 Network transmission time accounts for 20% of a null RPC and 80% of an RPC that transmits 1024 user bytes (less than the size of a network packet). By what percentage will the times for these two operations improve if the network is upgraded from 10 megabits/second to 100 megabits/second?

6.16 Ans.

\[ T_{null} = \text{null RPC time} = f + w_{null}, \text{ where } f = \text{fixed OS costs}, \ w_{null} = \text{time on wire at 10 megabits-per-second}. \]

Similarly, \( T_{1024} = \text{time for RPC transferring 1024 bytes} = f + w_{1024} \).

Let \( T'_{null} \) and \( T'_{1024} \) be the corresponding figures at 100 megabits per second. Then

\[ T'_{null} = f + 0.1w_{null}, \text{ and } T'_{1024} = f + 0.1w_{1024}. \]

Percentage change for the null RPC = \( 100(T_{null} - T'_{null})/T_{null} = 100*0.9w_{null}/T_{null} = 90*0.2 = 18\%. \)

Similarly, percentage change for 1024-byte RPC = \( 100*0.9*0.8 = 72\%. \)

6.17 A null RMI that takes no parameters, calls an empty procedure and returns no values delays the caller for 2.0 milliseconds. Explain what contributes to this time.

In the same RMI system, each 1K of user data adds an extra 1.5 milliseconds. A client wishes to fetch 32K of data from a file server. Should it use one 32K RMI or 32 1K RMIs?

6.17 Ans.

Page 236 details the costs that make up the delay of a null RMI.

one 32K RMI: total delay is \( 2 + 32*1.5 = 50 \text{ ms} \).

32 1K RMIs: total delay is \( 32(2+1.5) = 112 \text{ ms} \) -- one RMI is much cheaper.

6.21 A client makes RMIs to a server. The client takes 5 ms to compute the arguments for each request, and the server takes 10ms to process each request. The local OS processing time for each send or receive operation is 0.5 ms, and the network time to transmit each request or reply message is 3 ms. Marshalling or unmarshalling takes 0.5 ms per message.

Estimate the time taken by the client to generate and return from 2 requests (i) if it is single-threaded, and (ii) if it has two threads which can make requests concurrently on a single processor. Is there a need for asynchronous RMI if processes are multi-threaded?
6.21 Ans.
(i) Single-threaded time: \(2(5\, \text{prepare}) + 4(0.5\, \text{marsh/unmarsh}) + 0.5\, \text{(local OS)} + 2\times 3\, \text{(net)} + 10\, \text{(serv)} = 50\, \text{ms.}\\

(ii) Two-threaded time: (see figure 6.14) because of the overlap, the total is that of the time for the first operation's request message to reach the server, for the server to perform all processing of both request and reply messages without interruption, and for the second operation's reply message to reach the client.

This is: \(5 + (0.5+0.5+3) + (0.5+0.5+10+0.5+0.5) + (0.5+0.5+10+0.5+0.5) + (3 + 0.5+0.5) = 37\, \text{ms.}\\