# 6 - RPC

*Tutor version*

**Dr Leonardo Mostarda,**
**School of Science and Technology,**
**Camerino University, London**
Version 1.0, 10 Mar 2013

## Question 1

Why are transport-level communication services often inappropriate for building distributed applications?

## Solution 1

They hardly offer distribution transparency meaning that application developers are required to pay significant attention to implementing communication, often leading to proprietary solutions. The effect is that distributed applications, for example, built directly on top of sockets are difficult to port and to interoperate with other applications.

## Question 2

Consider a procedure incr with two integer parameters. The procedure adds one to each parameter. Now suppose that it is called with the same variable twice, for example, as incr(i, i). If i is initially 0, what value will it have afterward if call-by-reference is used?

## Solution 2

If call by reference is used, a pointer to i is passed to incr. It will be incremented two times, so the final result will be two.

## Question 3

C has a construction called a union, in which a field of a record (called a struct in C) can hold any one of several alternatives. At run time, there is no sure-fire way to tell which one is in there. Does this feature of C have any implications for remote procedure call? Explain your answer.

## Solution 3

If the runtime system cannot tell what type value is in the field, it cannot marshal it correctly. Thus unions cannot be tolerated in an RPC system unless there is a tag field that unambiguously tells what the variant field holds. The tag field must not be under user control.

## Question 4

One way to handle parameter conversion in RPC systems is to have each machine send parameters in its native representation, with the other one doing the translation, if need be. The native system could be indicated by a code in the first byte. However, since locating the first byte in the first word is precisely the problem, can this actually work?

## Solution 4

First of all, when one computer sends byte 0, it always arrives in byte 0. Thus the destination computer can simply access byte 0 (using a byte instruction) and the code will be in it. It does not matter whether this is the low-order byte or the high-order byte. An alternative scheme is to put the code in all the bytes of the first word. Then no matter which byte is examined, the code will be there.

## Question 5

Assume a client calls an asynchronous RPC to a server, and subsequently waits until the server returns a result using another asynchronous RPC. Is this approach the same as letting the client execute a normal RPC? What if we replace the asynchronous RPCs with asynchronous RPCs?

## Solution 5

No, this is not the same. An asynchronous RPC returns an acknowledgment to the caller, meaning that after the ?rst call by the client, an additional message is sent across the network. Likewise, the server is acknowledged that its response has been delivered to the client. Two asynchronous RPCs may be the same, provided reliable communication is guaranteed. This is generally not the case.