Concurrent Systems and Applications 2004
Paper 5 Question 4 (TLH)

This question is examining material from the lectures on 'Distributed systems' and 'RPC and RMI'.

(a) Consider a simple client-server system implemented using Java Remote Method Invocation (RMI). Describe:

(i) how the interface between the client and the server is defined, [4 marks]

Ordinary Java `interface` definitions are used in place of an external IDL. Two constraints are placed on the programmer: the interface must extend `Remote` from the standard libraries and all of the methods must throw `RemoteException` (used for signaling fatal problems in making a remote invocation).

(ii) how a particular instance of the server is named. [4 marks]

A basic two-level naming scheme is used. Names take the form `rmi://registry/service-name` in which `registry` identifies the RMI registry with which the server has been registered and `service-name` is the name under which the registration was made. This is an *impure* naming scheme because a service cannot move to a different registry without changing name.

(b) One operation provided by the server is to merge the contents of two hashtables, returning the result in a new hashtable. On a centralized system the operation's signature could be defined as follows:

```
Hashtable mergeTables(Hashtable a, Hashtable b)
```

Describe in detail the semantics with which parameters are passed and results are returned when this operation is implemented over RMI. [4 marks]

Object serialization is used in each case. This means that the server receives a *deep copy* of the parameters passed in to it and the client receives a deep copy of the result that the server produces. Objects are not shared directly between the client and the server; for instance, updates made by the server to an object linked from `b` will not be seen by the client. Objects which implement `Remote` are treated specially – they represent links that the client (or server) has to an RMI service and the name of that service is sent rather than a copy of the objects implementing it.

(c) The designer of a new RMI system proposes lazily copying the contents of objects that are passed between the client and the server. For instance, a large data item passed to `mergeTables` would only have to be sent if the server actually tries to access it. It is hoped that this scheme will make distributed systems faster because it will reduce the volume of data sent over the network.

(i) Describe a situation in which the new system is likely to be faster than traditional

RMI and a separate situation in which it is likely to be slower. [2 marks each]

The new system is likely to be faster when it reduces the volume of data sent *without introducing further network round-trips*. It is likely to be slower when it causes data to be transferred piecemeal: the same volume is sent but extra network delays are incurred.

(*ii*) How would this new scheme change the semantics with which parameters are passed and results are returned? [4 marks]

Two main observations, applicable to both the client and the server: (i) another thread may update objects between making an RPC call and receiving a 'lazy' request for further objects, (ii) if one party crashes then lazy requests may not be able to be honoured – how should this failure be exposed to the other party?