Assignment 2

SOFE4790/CSCI4640 Distributed Systems
Total mark: 60

Due: October 20

1. (a) Outline a scheme to solve the problem of dropped messages in IP multicast. Use message retransmissions. You should assume that (1) there may be multiple senders, (2) only a small fraction of messages are dropped, (3) recipients may not necessarily send a message within any particular time limit, (4) messages not dropped arrive in sender order. [5]

   (b) Does your solution in part (a) differ from the definition of reliable multicast? If so, explain how. [5]

2. Provide an application example for each of the following RPC exchange handshakes and provide a reasoning of why that particular handshake is suitable for the application you listed. [10]

	Messages sent by		
Handshake	***Client***	***Server***	***Client***
R	*Request*		
RR	*Request*	*Reply*	
RRA	*Request*	*Reply*	*Acknowledge reply*

3. Show how to use Java reflection to construct the client proxy class for the *Election* interface. Give the details of the implementation of one of the methods in this class, which should call the method *doOperation* with the following signature: [10]

   byte [] doOperation (RemoteObjectRef o, Method m, byte[] arguments);

   Hint: an instance variable of the proxy class should hold a remote object reference

   The Election interface provides two remote methods:
   vote: with two parameters through which the client supplies the name of a candidate (a string) and the 'voter's number' (an integer used to ensure each user votes once only). The voter's numbers are allocated sparsely from the range of integers to make them hard to guess.
   result: with two parameters through which the server supplies the client with the name of a candidate and the number of votes for that candidate.

4. The *Hello* RMI example has been given to you. Extend the programs to support RMI callback feature. Modify the clients so that they provide a *notify* remote method. Modify the server so that it provides two additional remote methods, *registerForCallback* and *unregisterForCallback*. Each new client registers with the server by remote invocation of *registerForCallbacks*, and upon this event, the server makes callbacks to *notify* method of all the current clients, informing them of the current number of clients. The server must use a data structure to maintain the list of clients. The client should also let user specify the length of time to remain registered, after this time expires, the clients unregisters with the server.

   You should provide screen shots of execution of your programs to show that the RMI callback feature has been implemented. You should also show the code you have added to implement the RMI callback. [20]

5. The text book presents a Use Case for tuple space communications based on the JavaSpaces software application. You can read this on pages 271-274. An example is presented of a Fire Alarm and its propagation to interested parties. For the FireAlarm example create a Class diagram and appropriate sequence diagram for the scenario of raising an alarm. [10]