

S249/307P

**DUBLIN INSTITUTE OF TECHNOLOGY
KEVIN STREET, DUBLIN 8.**

BSc Information Technology

Stage 3

SUMMER EXAMINATIONS 2006

DISTRIBUTED INFORMATION SYSTEMS

MR. C. O'LEARY
DR. B. O'SHEA
MR. P. LAWLESS

TIME ALLOWED: 3 HOURS

ATTEMPT 4 QUESTIONS

ALL QUESTIONS CARRY EQUAL MARKS

1. (a) Explain what is meant by *garbage collection* in an object oriented system, and using a clear example, show how garbage collection is made more difficult when objects are hosted in separate processes. Describe one approach to garbage collection in distributed systems.

(6 marks)

- (b) Methods invoked on objects by other objects in the same process will not suffer from the types of problems that can occur when the *caller* and *callee* are located in separate processes on separate hosts. Middleware platforms can attempt to solve or mask the problems introduced by distributing objects. When doing so the platform designers define the *invocation semantics* for their middleware platform.

Identify the *three* different types of invocation semantics that can be guaranteed for a middleware platform and distinguish between each type according to the fault tolerance measures that need to be adopted to provide the guarantees.

(9 marks)

- (c) It is widely felt that *web services* represent the future for middleware and Internet-wide distributed computing. Describe clearly the core technologies of the web service approach, and provide an analysis of its strengths and weaknesses when compared with competing approaches.

(10 marks)

2. (a) DES and RSA are two well known cryptographic algorithms with diverse characteristics. Distinguish clearly between the families of algorithms to which DES and RSA belong.

(6 marks)

- (b) The DES algorithm in its original form was famously cracked for the first time in 1997. Discuss the approach that was adopted in cracking DES on that occasion, and show how algorithm designers can protect against this type of codebreaking.

(9 marks)

- (c) Java tools such as `keytool` and `jarsigner` can be used to digitally sign documents and authenticate signatures.

Discuss the common approaches taken to perform *authentication* in distributed systems under the following headings:

- Authentication server approaches
- Authentication with public / private key pairs
- Authentication with shared secret keys

In your answer you must provide an analysis of the strengths and weaknesses of each approach as well as a clear description of the requirements for *secure hash algorithms*.

(10 marks)

3. (a) Describe clearly the approach adopted by Ethernet to avoid collisions between transmitted packets and show how this is adapted for Wireless Ethernet (WiFi).
(6 marks)
- (b) Describe clearly the function and operation of the Internet Protocol (IP), and show how some of its addressing and routing limitations are improved upon by the next generation of IP.
(9 marks)
- (c) UDP, TCP and HTTP can all be used to transport data for higher level applications. Evaluate each protocol according to its suitability for a diverse set of applications, treating in particular each protocol's mechanism for handling failure.
(10 marks)
4. (a) Define the term *transparency* in relation to distributed systems, and using as examples *four* different types of transparency, show how they *are* or *are not* provided for by some well known distributed applications (e.g. World-Wide-Web).
(6 marks)
- (b) *Heterogeneity, openness, scalability, security* and *naming* are key issues in the design and implementation of distributed systems. Provide a detailed discussion of how the World-Wide-Web addresses these important issues for distributed systems.
(9 marks)
- (c) Provide a detailed analysis of the *End-to-End* argument in system design.
(10 marks)
5. (a) Explain clearly what is meant by *distributed mutual exclusion* and define the three essential requirements for distributed mutual exclusion.
(6 marks)
- (b) Using examples, describe and evaluate three of the following algorithms for achieving mutual exclusion in a distributed system:
- Central server based algorithm
 - Ring-based algorithm
 - Multicast with Logical Clocks
 - Maekawa's Voting algorithm
- (9 marks)
- (c) Discuss the requirement for elections in distributed systems and provide a description and analysis of two well known approaches to conducting elections among multiple processes.
(10 marks)

6. (a) Consider the code shown below:

```
import java.io.*;
public class W {
    public static void main(String args[]) {
        try {
            ObjectOutputStream out =
                new ObjectOutputStream(
                    new FileOutputStream("x"));
            Book book = new Book("DaVinci Code", "D Brown");
            out.writeObject(book);
            out.close();
        } catch (IOException ioe) {}
    }
}

import java.io.*;
public class R {
    public static void main(String args[]) {
        try {
            ObjectInputStream in =
                new ObjectInputStream(
                    new FileInputStream("x"));
            Book b = (Book)in.readObject();
            System.out.println("Name    : " + b.getName());
            System.out.println("Author  : " + b.getAuthor());
        } catch (Exception e) {}
    }
}
```

Describe the intended functionality of the above programs and provide the code for the missing class.

(6 marks)

(b) Consider the code for the RMI application below.

```
import java.rmi.*;
public interface Stock extends Remote {
    public int getPrice() throws RemoteException;
    public void setPrice(int price) throws RemoteException;
}
```

```
import java.rmi.*;
import java.rmi.server.*;
public class StockImpl extends UnicastRemoteObject
implements Stock {
    private String name;
    private int price;
    public StockImpl() throws RemoteException {}
    public void setName(String name) { this.name = name; }
    public String getName() { return name; }
    public void setPrice(int price) throws RemoteException {
        this.price = price;
    }
    public int getPrice() throws RemoteException {
        return price;
    }
}
```

```
import java.rmi.*;
public interface StockManager extends Remote {
    public void createStock(String name, int price)
throws RemoteException;
    public Stock getStock(String name) throws RemoteException;
}
```

```
import java.rmi.*;
import java.rmi.server.*;
import java.util.Vector;
public class StockManagerImpl extends UnicastRemoteObject
implements StockManager {
    private Vector<StockImpl> stocks = new Vector<StockImpl>();
    public StockManagerImpl() throws RemoteException {}
    public void createStock(String name, int price)
throws RemoteException {
        StockImpl stock = new StockImpl();
        stock.setName(name);
        stock.setPrice(price);
        stocks.add(stock);
    }
    public Stock getStock(String name) throws RemoteException {
        for(int i = 0; i < stocks.size(); i++)
            if(stocks.get(i).getName().equals(name))
                return stocks.get(i);
        return null;
    }
}
```

```
import java.rmi.*;
public class S {
    public static void main(String[] args) throws Exception {
        StockManagerImpl o = new StockManagerImpl();
        String name = "rmi://localhost/StockManager";
        Naming.rebind(name, o);
    }
}
```

This application is composed of two remote classes. Instances of *StockManager* allow *Stock* objects to be created and hosted in the *S* process. Stock objects can then be queried for their *price* and can also have their price changed.

Provide the code for a client that will allow the user:

- Create a stock object by providing the *name* and *price* of the stock.
- View the *price* of a stock.
- Change the *price* of a stock for which they provide the *name*.

Note: You may use pseudo code for STDIN/STDOUT.

(9 marks)

- (c) Show how to modify all the code from part (b) so that
- Stock objects are not remote.
 - Stock object are created on the client side.
 - Stock objects are passed from the client to the server.
 - Stock objects are all stored by the server.
 - Stock objects can be requested by the client as before.

Briefly describe the impact this change in design has on the ability of the client to update the *price* of a stock on the server.

Note: You need only provide pseudocode once it is accompanied by a detailed explanation of the changes required.

(10 marks)