CMPT-401 Operating Systems II (Fall 2005)

Midterm

School of Computing Science
Simon Fraser University
October 18, 2005

4:40pm - 6:10pm (90 min)

Last name: _________________________

First name: _________________________

Student number: ____________________

Signature: _________________________

Note: 1. Marks in total: 20
2. Pages (excluding this one): FIVE
3. Questions: THREE
4. Use either pen or pencil. Make sure your writing is legible.
5. Write your answers only in the provided spaces. The space for answering each question is more than sufficient. Be concise! Ambiguous, off-point, or self-conflicting answers hurt.

<table>
<thead>
<tr>
<th>Question</th>
<th>Your score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1 (5 points)</td>
<td></td>
</tr>
<tr>
<td>Question 2 (7 points)</td>
<td></td>
</tr>
<tr>
<td>Question 3 (8 points)</td>
<td></td>
</tr>
<tr>
<td>In Total (20 points)</td>
<td></td>
</tr>
</tbody>
</table>
Question 1 (5 points)

1.1. An idempotent operation is one that can be executed repeatedly with the same effect.

(1) (1 point) Can you convert non-idempotent operations into idempotent operations? Always, never, or sometimes? Explain using examples.

Answer: sometimes. For example, “increment X” when X = 3 can be changed to “write 4 into X”. The former is not idempotent, while the latter is. But this conversion is based on our knowledge of the status of X. In case we do not have the knowledge, we cannot perform the conversion.

(2) (1 point) Suppose all methods maintained by server A are idempotent while server B maintains some non-idempotent operations too. Using the failure model (arbitrary failure and omission failure), analyze the advantage and disadvantage of at-least-once invocation semantics comparing to at-most-once, for the case of server A and server B respectively.

Answer: For server A, at-least-once is almost the same as at-most-once. They both suffer from omission failure but never arbitrary failure. For server B, at-most-once is more advantageous because it only suffers from omission failure while at-least-once will suffer from arbitrary failure in addition to omission failure.

1.2. Several separate objects and modules are involved in achieving a remote method invocation. Based on the given figure, answer the following questions:

(1) (0.5 point) In no more than 3 sentences, explain the role played by communication modules.

Answer: carry out the request-reply protocol, which transmits request and reply messages between client and server.
(2) (0.5 point) In no more than 3 sentences, explain the role played by remote reference modules.

Answer: responsible for translating between local and remote object references (and for creating remote object references).

(3) (2 points) Which components implement the methods in remote interface? And how?

Answer: the class of a servant, the class of a proxy, and the skeleton.
Each method in a servant object eventually handles the remote request for that method.
Each method of a proxy (one for each remote object as its local representative) marshals the request message and sends it to the target, awaits the reply message, unmarshals it and returns the results to the invoker.
Each method in the skeleton unmarshals the arguments in the request message and invokes the corresponding method in the servant, awaits for the invocation to complete and then marshals the results, sends it in a reply message to the sending proxy’s method.

Question 2 (7 points)

2.1. (2 point) Cryptography has some well-accepted common terminologies and notations. Explain the following terms:

Alice: First participant
Bob: Second participant
Eve: Eavesdropper
Mallory: Malicious attacker

$K_A$: Alice’s secret key
$K_B$: Bob’s secret key
$K_{AB}$: Secret key shared between Alice and Bob

$K_{Apriv}$: Alice’s private key (known only to Alice)

$K_{Apub}$: Alice’s public key (published by Alice for all to read)

$\{M\}_K$: Message $M$ encrypted with key $K$
2.2. (5 points) Consider the following scenario in which Alice wishes to access file server Bob. Sara is an authentication server holding $K_A$ and $K_B$.

Step I. Alice sends a message to Sara with her identity requesting a ticket for access to Bob.
Step II. Sara responds with $\{\{\text{Ticket}\}_K, K_{AB}\}$, where ticket = $\{K_{AB}, Alice\}$.
Step III. Alice decrypts the response, and sends the ticket to Bob together with her identity and a request $R$ to access a file: $\{\text{Ticket}\}_K, Alice, R$.
Step IV. Bob decrypts the ticket, and starts to communicate with Alice using $K_{AB}$.

(1) Can another person other than Alice pretend to be Alice to access Bob? Why and why not?

Answer: No. Alice can use $K_A$ to decrypt Sara’s response in step II, another person does not have her secret key $K_A$ therefore cannot get the ticket for her.

(2) After obtaining the ticket, can Alice use it and pretend to be another person other than herself to access Bob? Why and why not?

Answer: No. Alice’s ticket is encrypted with $K_B$, she cannot decrypt and tamper with the ticket. The ticket contains her identity and Bob will check it in step IV.

(3) Another person other than Bob may intercept Alice’s request after step II. Can he pretend to be Bob and provide the file access service to Alice? Why and why not?

Answer: No. He does not have $K_B$ to decrypt Alice’s ticket, thus he cannot obtain $K_{AB}$ and use it to communicate with Alice.

(4) This scenario makes use of a cryptographic challenge. In which step is it used and what is the advantage of this technique?

Answer: It is used in step II. The advantage is user’s password does not have to be submitted to an authentication service each time it is authenticated.

(5) This scenario is not appropriate for E-commerce or wide-area applications. Why? What could be a solution (in one sentence)?

Answer: It is inappropriate because the use of a separate channel is extremely inconvenient and the requirement for a trusted third party is unrealistic. Public key cryptography can be a solution.
Question 3 (8 points)

3.1. (6 points) The CMPT-401 instructor wants to set up a service using Java RMI for students to check their midterm marks. For some reason, he was not able to accomplish the work completely. Please help him to get it done by filling in the blanks.

First, he defined a remote interface:
//Gradebook.java
public interface Gradebook extends Remote{
    double check(String studentName) throws RemoteException;
}

Then, he wrote the server program, for which he wrote a main program and a servant class separately as follows:
//GradebookServer.java //runs on css.css.sfu.ca with port# 2099
public class GradebookServer{
    public static void main(String args[]){
        System.setSecurityManager(new RMISecurityManager());
        try{
            Gradebook aGradebook = new GradebookServant();
            Naming.rebind("GradebookServer", aGradebook);
            System.out.println("Gradebook server ready");
            }catch ... }}

//GradebookServant.java, the servant class
public class GradebookServant extends UnicastRemoteObject implements Gradebook{
    private ... ; //data structure containing students’ marks
    public GradebookServant()throws RemoteException{...}

    public double check(String studentName) { ... } }

The he wrote the client program for student “Nerd” as follows:
//GradebookClient.java
public class GradebookClient{
    public static void main(String args[]){
        System.setSecurityManager(new RMISecurityManager());
        Gradebook aGradebook = null;
        try{
            String host = "css.css.sfu.ca:2099";
            String name = "rmi://" + host + "/GradebookServer";
            aGradebook = (Gradebook)Naming.lookup(name);
            double marksForNerd = aGradebook.check(Nerd);
        }catch ... }}
3.2. (1 point) In the above programs, there is one line of code, whose execution causes the server to start listening for network requests. Copy the line down here and explain why.

Answer: the line is:

```java
Gradebook aGradebook = new GradebookServant();
```

The GradebookServant class extends UnicastRemoteObject, which automatically creates socket and listens for network requests, and makes its services available by exporting them. Thus once an instance of GradebookServant is created, the server starts to listen for network requests automatically.

3.3. (1 point) Fill in the following blanks.

In Java RMI, the parameters of a method are assumed to be __input__ parameters and the result of a method is a single __output__ parameter. Any object that is __serializable__ can be passed as an argument or result in Java RMI. Remote objects are always passed as __remote object reference__. Non-remote objects are passed by __value__. For the latter case, if the recipient does not already possess __the class for the object__, its code is downloaded automatically; similarly, for the former case, if the recipient does not already possess __the class for a proxy__, its code is downloaded automatically.