5.1 Define a class whose instances represent request and reply messages. The class should provide a pair of constructors, one for the other for reply messages, showing how the request identifier also provides a method to marshal itself into an array of bytes or of bytes into an instance.

5.2 Program each of the three operations of the request-reply protocol in UDP communication, but without adding any fault-tolerance mechanisms from the classes you defined in the previous chapter for remote object communication above for request and reply messages (Exercise 5.1).

5.3 Give an outline of the server implementation, showing how the sendRequest and sendReply are used by a server that creates a new thread request. Indicate how the server will copy the requestId from the request message and how it will obtain the client IP address and request Id from the reply message. After a timeout, it retransmits the request message, no reply, it informs the caller.

5.4 Define a new version of the doOperation method that sets a time reply message. After a timeout, it retransmits the request message, no reply, it informs the caller.

5.5 Describe a scenario in which a client could receive a reply from

5.6 Describe the ways in which the request-reply protocol masks operating systems and computer networks.

5.7 Discuss whether the following operations are idempotent:
   i) pressing a lift (elevator) request button;
   ii) writing data to a file;
   iii) appending data to a file.

Is it a necessary condition for idempotence that the operation should be idempotent with any state?

5.8 Explain the design choices that are relevant to minimizing the amount of data at a server. Compare the storage requirements when the RR and RP

5.9 Assume the RRA protocol is in use. How long should servers retain reply data? Should servers repeatedly send the reply in an at acknowledgement?

5.10 Why might the number of messages exchanged in a protocol be performance than the total amount of data sent? Design a variant in which the acknowledgement is piggy-backed on — that is, transmit message as — the next request where appropriate, and otherwise message. (Hint: use an extra timer in the client.)
5.11 An Election interface provides two remote methods:

- **vote**: This method has two parameters through which the client supplies of a candidate (a string) and the 'voter's number' (an integer used to user votes once only). The voter’s numbers are allocated sparsely from integers to make them hard to guess.

- **result**: This method has two parameters through which the server supplies client with the name of a candidate and the number of votes for that candidate. Which of the parameters of these two procedures are input and which parameters?

Discuss the invocation semantics that can be achieved when the request-reject is implemented over a TCP/IP connection, which guarantees that data is delivered order sent, without loss or duplication. Take into account all of the conditions connection to be broken.

5.13 Define the interface to the Election service in CORBA IDL and Java RMI CORBA IDL provides the type long for 32-bit integers. Compare the methods languages for specifying input and output arguments.

5.14 The Election service must ensure that a vote is recorded whenever any user have cast a vote.

Discuss the effect of maybe call semantics on the Election service.

Would at-least-once call semantics be acceptable for the Election service or recommend at-most-once call semantics?

5.15 A request-reply protocol is implemented over a communication service with failures to provide at-least-once invocation semantics. In the first case the request assumes an asynchronous distributed system. In the second case the implementation assumes that the maximum time for the communication and the execution of the method is T. In what way does the latter assumption simplify the implementation?

5.16 Outline an implementation for the Election service that ensures that its record is consistent when it is accessed concurrently by multiple clients.

5.17 Assume the Election service is implemented in RMI and must ensure that a safely stored even when the server process crashes. Explain how this can be done with reference to the implementation outline in your answer to Exercise 5.16.

5.18 Show how to use Java reflection to construct the client proxy class for the interface. Give the details of the implementation of one of the methods in which should call the method doOperation with the following signature:

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

Hint: an instance variable of the proxy class should hold a remote object refe
We have not considered issues related to quality of service in this analysis. Message queue systems do offer intrinsic support for reliability in the transactions. More generally, however, quality of service remains a key challenge. Indirect communication paradigms. Indeed, space and time uncoupling by nature make it difficult to reason about end-to-end properties of the system, such as time behaviour or security, and hence this is an important area for further research.

**EXERCISES**

6.1 Construct an argument as to why indirect communication may be appropriate in environments. To what extent can this be traced to time uncoupling, space uncoupling, or indeed a combination of both?

6.2 Section 6.1 states that message passing is both time- and space-coupled. Messages are both directed towards a particular entity and require the recipient to be present at the time of the message send. Consider the case, though, where messages are directed towards a name rather than an address and this name is resolved at some time later. Does such a system exhibit the same level of indirection?

6.3 Section 6.1 refers to systems that are space-coupled but time-uncoupled. Messages are directed towards a given receiver (or receivers), but that receiver’s lifetime is independent from the sender’s. Can you construct a communication paradigm with these properties? For example, does email fall into this category?

6.4 As a second example, consider the communication paradigm referred to as queuing systems programming in mobile environments where participants in communication may become disconnected for periods of time. The system offers the RPC paradigm. Hence calls are directed towards a given server (clearly space-coupled). Though, are routed through an intermediary, a queue at the sending side, maintained in the queue until the receiver is available. To what extent is time uncoupled? Hint: consider the almost philosophical question of whether a recipient is temporarily unavailable exists at that point in time.

6.5 If a communication paradigm is asynchronous, is it also time-uncoupled? Exp answer with examples as appropriate.

6.6 In the context of a group communication service, provide example message exchanges that illustrate the difference between causal and total ordering.

6.7 Consider the *FireAlarm* example as written using JGroups (Section 6.2.3). Suppose it was generalized to support a variety of alarm types, such as fire, flood, intrusion on. What are the requirements of this application in terms of reliability and order?
6.9 Suggest a design for a notification mailbox service that is intended on behalf of multiple subscribers, allowing subscribers to specify notifications to be delivered. Explain how subscribers that are not make use of the service you describe. How will the service deal crash while they have delivery turned on?

In publish-subscribe systems, explain how channel-based approach implemented using a group communication service? Why is this a for implementing a content-based approach?

6.10 Using the filtering-based routing algorithm in Figure 6.11 as a start alternative algorithm that illustrates how the use of advertise significant optimization in terms of message traffic generated.

6.11 Construct a step-by-step guide explaining the operation of the alt based routing algorithm shown in Figure 6.12.

6.12 Building on your answer to Exercise 6.11, discuss two possible $EN(e)$ and $SN(s)$. Why must the intersection of $EN(e)$ and $SN(s)$ be $e$ that matches $s$ (the intersection rule)? Does this apply implementations?

6.13 Explain how the loose coupling inherent in message queues can be Application Integration. As in Exercise 6.1, consider to what extent time uncoupling, space uncoupling or a combination of both.

6.14 Consider the version of the FireAlarm program written in JMS would you extend the consumer to receive alarms only from a giv-

6.15 Explain in which respects DSM is suitable or unsuitable for client

6.16 Discuss whether message passing or DSM is preferable for fault-t

6.17 Assuming a DSM system is implemented in middleware without and in a platform-neutral manner, how would you deal with the pro representations on heterogeneous computers? Does your solution

6.18 How would you implement the equivalent of a remote procedure space? What are the advantages and disadvantages of implementin call–style interaction in this way?

6.19 How would you implement a semaphore using a tuple space?

6.20 Implement a replicated tuple space using the algorithm of Xu Explain how this algorithm uses the semantics of tuple space oper: replication strategy.