munication facilities that may allow one-to-many, many-to-one, and many-to-many types of interactions between the senders and receivers. Some issues related to group communication are group management, group addressing, atomicity, and ordered message delivery.

EXERCISES

3.1. What are the main reliability issues in designing a message-passing system? Describe a suitable mechanism for handling each of these issues.

3.2. What are the main issues related to the correctness of the IPC protocols of a message-passing system? Describe a suitable mechanism for handling each of these issues.

3.3. Describe some flexibility features that a message-passing system should provide to its users. Write suitable IPC primitives that will allow the users to take advantage of these flexibility features.

3.4. Describe blocking and nonblocking types of IPC. Which is easier to implement and why? Discuss their relative advantages and disadvantages.

3.5. Write the code for implementing a producer-consumer pair of processes for the following two cases:

(a) They use a single-message buffer.

(b) They use a buffer that can accommodate up to n messages.

The producer produces messages and puts them in the message buffer, and the consumer consumes messages from the message buffer. Assume that all messages are of fixed size.

3.6. What is a datagram? Why are multidatagram messages used in IPC? What are the main issues in IPC of multidatagram messages? Describe a mechanism for handling each of these issues.

3.7. In a multidatagram communication, explain how the recipient can properly recognize and arrange the datagrams of the same message and how can it recognize the last datagram of a message.

3.8. A linked list of characters is to be sent across a network in the form of a stream of bytes. Write the code in any suitable programming language for encoding the data structure on the sender side and decoding it on the receiver side.

3.9. Write the code for implementing the bulletin-board semantics for one-to-many communication for the following cases:

(a) A message posted on the bulletin board is automatically removed after n receivers have read it.

(b) A message posted on the bulletin board is automatically removed after the lapse of time t. Time t is a parameter specified by the sender of a message.

Assume that only one message can be posted on the bulletin board at a time. If the bulletin board is not empty at the time a sender wants to post a message, a "not empty, try again" message is returned to the sender. You may make any other assumptions that you feel necessary, but clearly state your assumptions.

3.10. Give two examples of applications for which each of the following types of multicast communication is most suitable:

(a) The 0-reliable

(b) The 1-reliable

(c) The m-out-of-n reliable ($1 < m < n$)

(d) All-reliable

3.11. What is meant by "ordered message delivery"? Do all applications need the same semantics for this property? If yes, explain why. If no, give examples of two applications that need different semantics for this property.

3.12. Explain what is meant by absolute ordering, consistent ordering, and causal ordering of messages. Give a mechanism to implement each one.

3.13. Describe a mechanism for implementing consistent ordering of messages in each of the following cases:
(a) One-to-many communication
(b) Many-to-one communication
(c) Many-to-many communication

3.14. Describe three different process-addressing mechanisms. Discuss their relative advantages and disadvantages. Which of the mechanisms described by you is most suitable for each of the following cases (give reasons for your answer):
(a) For communication between a server process and several client processes. The client processes send request messages to the server process and the server process returns a reply for each client request.
(b) For allowing a sender process to send messages to a group of processes.
(c) For allowing a sender process to send messages to a receiver process that is allowed to migrate from one node to another.
(d) For allowing a sender process to send messages to a receiver process that is allowed to migrate from one node to another and to allow the receiver process to return a reply to the sender process.
(e) For allowing a client process to receive service from any one of the several server processes providing that service.

3.15. What is an idempotent operation? Which of the following operations are idempotent:
(a) *read_next_record* (filename)
(b) *read_record* (filename, record_number)
(c) *append_record* (filename, record)
(d) *write_record* (filename, after_record_n, record)
(e) *seek* (filename, position)
(f) *add* (integer_1, integer_2)
(g) *increment* (variable_name)

3.16. Suggest a suitable mechanism for implementing each of the following types of IPC semantics:
(a) Last one
(b) At least once
(c) Exactly once

3.17. The operations performed by a server are nonidempotent. Describe a mechanism for implementing exactly-once IPC semantics in this case.

3.18. Suggest whether at-least-once or exactly-once semantics should be used for each of the following applications (give reasons for your answer):
(a) For making a request to a file server to read a file
(b) For making a request to a file server to append some data to an existing file
(c) For making a request to a compilation server to compile a file
(d) For making a request to a database server to update a bank account
(e) For making a request to a database server to get the current balance of a bank account
(f) For making a request to a booking server to cancel an already booked seat

3.19. Suggest a suitable mechanism for implementing reliable IPC with exactly-once semantics in each of the following cases:

(a) The computers of the sender and receiver processes are reliable but the communication links connecting them are unreliable.

(b) The computers of the sender and receiver processes are unreliable but the communication links connecting them are reliable.

(c) The computer of the sender process and the communication links connecting the computers of the sender and receiver processes are reliable but the computer of the receiver process is unreliable.

(d) The computer of the receiver process and the communication links connecting the computers of the sender and receiver processes are reliable but the computer of the sender process is unreliable.

3.20. Is it always necessary for the sender of a message to know that the message arrived safely at its destination? If yes, explain why. If no, give two examples in support of your answer.

3.21. What is the main purpose of using an acknowledgment message in an IPC protocol? Are acknowledgment messages always needed for reliable communication? Give reasons for your answer.

3.22. What is "piggybacking" of a message? How does it help in reducing network traffic? Give some examples of where the piggybacking scheme may be used to improve performance in distributed systems.

3.23. In many client-server systems, the timeout mechanism is used to guard against the hanging of a client forever if the server crashes. That is, in these systems, if after sending a request message to a server a client does not receive a reply from the server within a fixed timeout interval, the client assumes that the server has crashed and can take necessary corrective actions. What should be the ideal length of the timeout period in these systems? If the server computer is fully reliable, is it still useful to use the timeout mechanism? Give reasons for your answer.

3.24. A file server services file read/write requests of multiple clients. Clients can directly communicate with the file server by sending messages to it and receiving messages from it.

(a) Describe the contents of the message that a client must send to the file server for reading a portion of a file.

(b) Describe the contents of the message that the server must return to the client in reply to part (a).

(c) Describe the contents of the message that a client must send to the file server for writing some data to an existing file.

(d) Describe the contents of the message that the server must return to the client in reply to part (c).

(e) Describe a mechanism by which the file server can cope with multiple client requests arriving almost simultaneously.

3.25. In a client-server IPC, a client sends a request message to a server, and the server processes the request and returns the result of the request processing to the client. Is it useful for a process to behave both as a client and a server? If no, explain why. If yes, give an example in support of your answer.

3.26. A file storage and retrieval system consists of the following types of processes: several client processes, a file manager process, a cache manager process, and several disk manager processes. These processes perform the following jobs:

Bershad et al. [1990] proposed the use of Lightweight Remote Procedure Cal (LRPC), which is a communication facility designed and optimized for cross-domai communications in microkernel operating systems. For achieving better performance tha conventional RPC systems, LRPC uses the following four techniques: simple contro transfer, simple data transfer, simple stubs, and design for concurrency.

Some optimizations that may be used to improve the performance of distributed applications that use an RPC facility are concurrent access to multiple servers, serving multiple requests simultaneously, reducing per call workload of servers, reply caching of idempotent remote procedures, proper selection of timeout values, and proper design of RPC protocol specification.

EXERCISES

4.1. What was the primary motivation behind the development of the RPC facility? How does an RPC facility make the job of distributed applications programmers simpler?

4.2. What are the main similarities and differences between the RPC model and the ordinary procedure call model?

4.3. In the conventional procedure call model, the caller and the callee procedures often use global variables to communicate with each other. Explain why such global variables are not used in the RPC model.

4.4. In RPC, the called procedure may be on the same computer as the calling procedure or it may be on a different computer. Explain why the term *remote procedure call* is used even when the called procedure is on the same computer as the calling procedure.

4.5. What are the main issues in designing a transparent RPC mechanism? Is it possible to achieve complete transparency of an RPC mechanism? If no, explain why. If yes, explain how.

4.6. Achieving complete transparency of an RPC mechanism that allows the caller and callee processes to be on different computers is nearly impossible due to the involvement of the network in message communication between the two processes. Suppose an RPC mechanism is to be designed in which the caller and callee processes are always on the same computer. Is it possible to achieve complete transparency of this RPC mechanism? Give reasons for your answer.

4.7. What is a "stub"? How are stubs generated? Explain how the use of stubs helps in making an RPC mechanism transparent.

4.8. A server is designed to perform simple integer arithmetic operations (addition, substraction, multiplication, and division). Clients interact with this server by using an RPC mechanism. Describe the contents of the call and reply messages of this RPC application, explaining the purpose of each component. In case of an error, such as division by zero or arithmetic overflow, the server must suitably inform the client about the type of error.

4.9. Write marshaling procedures for both tagged and untagged representations for marshaling the message contents of the RPC application of Exercise 4.8.

4.10. A user-defined program object is a structure consisting of the following basic data types in that order: a Boolean, an integer, a long integer, and a fixed-length character string of eight characters. Write marshaling procedures for both tagged and untagged representations for this program object. Assume that the RPC software provides marshaling of the basic data types.

4.11. The caller process of an RPC must wait for a reply from the callee process after making a call. Explain how this can actually be done.

4.12. Differentiate between stateful and stateless servers. Why do some distributed applications use stateless servers in spite of the fact that stateful servers provide an easier programming paradigm and are typically more efficient than stateless servers?

4.13. Suggest a suitable server creation semantics for each of the following types of applications:
(a) A service is needed only once in a while, and the session for which a client interacts with the server of this service involves the exchange of a single call and a single reply message between the client and server processes.
(b) A service is needed only once in a while, and the session for which a client interacts with the server of this service normally involves the exchange of several call and reply messages between the client and server processes.
(c) A server can service the requests of multiple clients.

4.14. A server is to be shared by multiple clients. Describe a scheme for designing the remote procedures offered by the server so that interleaved or concurrent requests from different clients do not interfere with each other.

4.15. Why do most RPC systems support call-by-value semantics for parameter passing?

4.16. Discuss the similarities and differences between the following parameter-passing semantics that may be used in an object-based system:
(a) Call-by-object-reference
(b) Call-by-move
(c) Call-by-visit

4.17. Explain why RPC semantics is normally different from the conventional procedure call semantics. Clarify the differences among may-be, last-one, last-of-many, at-least-once, and exactly-once call semantics. Explain how each of these may be implemented.

4.18. What is an *orphan call*? How are orphan calls handled in the implementation of the following types of call semantics:
(a) Last-one call semantics
(b) Last-of-many call semantics
(c) At-least-once call semantics

4.19. Suggest whether may-be, last-one, last-of-many, at-least-once, or exactly-once call semantics should be used for each of the following applications (give reasons for your answer):
(a) For making a request to a time server to get the current time.
(b) For making a request to a node's resource manager to get the current status of resource availability of its node.
(c) For periodically broadcasting the total number of current jobs at its node by a process manager in a system in which the process managers of all nodes mutually cooperate to share the overall system load.
(d) For making a request to a computation server to compute the value of an equation.
(e) For making a request to a booking server to get the current status of availability of seats.
(f) For making a request to a booking server to reserve a seat.
(g) For making a request to a file server to position the *read-write pointer* of a file to a specified position.
(h) For making a request to a file server to append a record to an existing file.

(i) For making a request to a name server to get the location of a named object in a system that does not support object mobility.
(j) For making a request to a name server to get the location of a named object in a system that supports object mobility.

4.20. Explain why most RPC systems do not use acknowledgment messages. Differentiate among R, RR, and RRA protocols for RPCs. Give an example of an application in which each type of protocol may be the most suitable one to use.

4.21. Suppose it takes time T (T is very large) for a server to process an RPC request. Even though a client making the RPC request knows that it will receive the reply for its request from the server only after time T, it will unnecessarily keep waiting for the reply from the server for this entire duration in situations where the request message does not reach the server due to failure of the communication link between the client and the server or the server crashes while processing the client's request. Devise a mechanism to avoid this situation. That is, it should be possible for a client to detect an exception condition and to take corrective action as early as possible.

4.22. Suppose you have to design an RPC mechanism for interaction between clients and a file server, frequently requiring transfer of large volume of data in between them. However, the underlying network has a limitation of maximum packet size of 4 kilobytes. Suppose the time to transfer a 4-kilobyte packet is 4 ms, and the time to do a null RPC (i.e., 0 data bytes) is 0.5 ms. If the average amount of data transferred for each RPC request is 16 kilobytes, which of the following two methods will you prefer to use in your design:
(a) Using several physical RPCs for one logical RPC, each physical RPC transferring a single packet of data
(b) Using a single RPC with the data transferred as multidatagram messages

4.23. What are the main advantages of an RPC system that allows the binding between a client and a server to change dynamically? What are the main issues involved in providing this flexibility? Describe a mechanism to handle each of the issues mentioned by you.

4.24. A server is normally designed to service multiple clients and is often bound simultaneously to multiple clients. Does a situation ever arise when a client should be simultaneously bound to multiple servers? If no, explain why. If yes, give two examples of such a situation.

4.25. Discuss the relative advantages and disadvantages of binding a client and a server at compile time, at link time, and at call time.

4.26. Given the interface name of a server, discuss the relative advantages and disadvantages of using the broadcast method and the method of using a name server for locating the server.

4.27. What is *callback RPC* facility? Give an example of an application where this facility may be useful. What are the main issues involved in supporting this facility in an RPC system? Describe a mechanism to handle each of these issues.

4.28. Give an example of an application where each of the following facilities may be useful:
(a) Broadcast RPC facility
(b) Multicast RPC facility
Describe a mechanism to implement each of these facilities.

4.29. Give the characteristics of applications for which the batch-mode RPC facility may be useful. What are the main problems in using this facility?

4.30. Find out the details of the client-server binding mechanism of the HRPC system [Bershad et al. 1987], and explain how the choices of transport protocol, data representation, and control protocol are delayed until bind time in this system.