Exercises for Chapter 1: Characterization of Distributed Systems

Exercise 1.1
Give five types of hardware resource and five types of data or software resource that can usefully be shared. Give examples of their sharing as it occurs in practice in distributed systems. pages 2, 14

Exercise 1.2
How might the clocks in two computers that are linked by a local network be synchronized without reference to an external time source? What factors limit the accuracy of the procedure you have described? How could the clocks in a large number of computers connected by the Internet be synchronized? Discuss the accuracy of that procedure. page 2

Exercise 1.3
Consider the implementation strategies for massively multiplayer online games as discussed in Section 1.2.2. In particular, what advantages do you see in adopting a single server approach for representing the state of the multiplayer game? What problems can you identify and how might they be resolved? page 5

Exercise 1.4
A user arrives at a railway station that she has never visited before, carrying a PDA that is capable of wireless networking. Suggest how the user could be provided with information about the local services and amenities at that station, without entering the station’s name or attributes. What technical challenges must be overcome? page 13

Exercise 1.5
Compare and contrast cloud computing with more traditional client-server computing? What is novel about cloud computing as a concept? Pages 13, 14
Exercise 1.6

Use the World Wide Web as an example to illustrate the concept of resource sharing, client and server. What are the advantages and disadvantages of HTML, URLs and HTTP as core technologies for information browsing? Are any of these technologies suitable as a basis for client-server computing in general?  

Pages 14, 16

Exercise 1.7

A server program written in one language (for example C++) provides the implementation of a BLOB object that is intended to be accessed by clients that may be written in a different language (for example Java). The client and server computers may have different hardware, but all of them are attached to an internet. Describe the problems due to each of the five aspects of heterogeneity that need to be solved to make it possible for a client object to invoke a method on the server object.

Page 16

Exercise 1.8

An open distributed system allows new resource sharing services such as the BLOB object in Exercise 1.7 to be added and accessed by a variety of client programs. Discuss in the context of this example, to what extent the needs of openness differ from those of heterogeneity.

Page 17

Exercise 1.9

Suppose that the operations of the BLOB object are separated into two categories – public operations that are available to all users and protected operations that are available only to certain named users. State all of the problems involved in ensuring that only the named users can use a protected operation. Supposing that access to a protected operation provides information that should not be revealed to all users, what further problems arise?

Page 18

Exercise 1.10

The INFO service manages a potentially very large set of resources, each of which can be accessed by users throughout the Internet by means of a key (a string name). Discuss an approach to the design of the names of the resources that achieves the minimum loss of performance as the number of resources in the service increases. Suggest how the INFO service can be implemented so as to avoid performance bottlenecks when the number of users becomes very large.

Page 19

Exercise 1.11

List the three main software components that may fail when a client process invokes a method in a server object, giving an example of a failure in each case. Suggest how the components can be made to tolerate one another’s failures.

Page 21
Exercise 1.12

A server process maintains a shared information object such as the BLOB object of Exercise 1.7. Give arguments for and against allowing the client requests to be executed concurrently by the server. In the case that they are executed concurrently, give an example of possible ‘interference’ that can occur between the operations of different clients. Suggest how such interference may be prevented.

page 22

Exercise 1.13

A service is implemented by several servers. Explain why resources might be transferred between them. Would it be satisfactory for clients to multicast all requests to the group of servers as a way of achieving mobility transparency for clients?

page 23

Exercise 1.14

Resources in the World Wide Web and other services are named by URLs. What do the initials URL denote? Give examples of three different sorts of web resources that can be named by URLs.

Page 26

Exercise 1.15

Give an example of an HTTP URL. List the main components of an HTTP URL, stating how their boundaries are denoted and illustrating each one from your example. To what extent is an HTTP URL location-transparent?

page 26

Exercises for Chapter 2:

System models

From Coulouris, Dollimore, Kindberg and Blair
Distributed Systems: Concepts and Design
Edition 5, © Addison-Wesley 2012

Exercise 2.1

Provide three specific and contrasting examples of the increasing levels of heterogeneity experienced in contemporary distributed systems as defined in Section 2.2.

page 39
Exercise 2.1
What problems do you foresee in the direct coupling between communicating entities that is implicit in remote invocation approaches? Consequently, what advantages do you anticipate from a level of decoupling as offered by space and time uncoupling? Note: you might want to revisit this answer after reading Chapters 5 and 6.

Exercise 2.3
Describe and illustrate the client-server architecture of one or more major Internet applications (for example the Web, email or netnews).

Exercise 2.4
For the applications discussed in Exercise 2.1 state how the servers cooperate in providing a service.

Exercise 2.5
A search engine is a web server that responds to client requests to search in its stored indexes and (concurrently) runs several web crawler tasks to build and update the indexes. What are the requirements for synchronization between these concurrent activities?

Exercise 2.6
The host computers used in peer-to-peer systems are often simply desktop computers in users’ offices or homes. What are the implications of this for the availability and security of any shared data objects that they hold and to what extent can any weaknesses be overcome through the use of replication?

Exercise 2.7
List the types of local resource that are vulnerable to an attack by an untrusted program that is downloaded from a remote site and run in a local computer.
Exercise 2.8

Give examples of applications where the use of mobile code is beneficial.

Exercise 2.9

Consider a hypothetical car hire company and sketch out a three-tier solution to the provision of their underlying distributed car hire service. Use this to illustrate the benefits and drawbacks of a three-tier solution considering issues such as performance, scalability, dealing with failure and also maintaining the software over time.

Exercise 2.10

Provide a concrete example of the dilemma offered by Saltzer’s end-to-end argument in the context of the provision of middleware support for distributed applications (you may want to focus on one aspect of providing dependable distributed systems, for example related to fault tolerance or security).

Exercise 2.11

Consider a simple server that carries out client requests without accessing other servers. Explain why it is generally not possible to set a limit on the time taken by such a server to respond to a client request. What would need to be done to make the server able to execute requests within a bounded time? Is this a practical option?

Exercise 2.12

For each of the factors that contribute to the time taken to transmit a message between two processes over a communication channel, state what measures would be needed to set a bound on its contribution to the total time. Why are these measures not provided in current general-purpose distributed systems?

Exercise 2.13

The Network Time Protocol service can be used to synchronize computer clocks. Explain why, even with this service, no guaranteed bound is given for the difference between two clocks.
Exercise 2.14

Consider two communication services for use in asynchronous distributed systems. In service A, messages may be lost, duplicated or delayed and checksums apply only to headers. In service B, messages may be lost, delayed or delivered too fast for the recipient to handle them, but those that are delivered arrive with the correct contents. Describe the classes of failure exhibited by each service. Classify their failures according to their effect on the properties of validity and integrity. Can service B be described as a reliable communication service?

page 67 and page 71

Exercise 2.15

Consider a pair of processes X and Y that use the communication service B from Exercise 2.14 to communicate with one another. Suppose that X is a client and Y a server and that an invocation consists of a request message from X to Y, followed by Y carrying out the request, followed by a reply message from Y to X. Describe the classes of failure that may be exhibited by an invocation.

page 67

Exercise 2.16

Suppose that a basic disk read can sometimes read values that are different from those written. State the type of failure exhibited by a basic disk read. Suggest how this failure may be masked in order to produce a different benign form of failure. Now suggest how to mask the benign failure.

page 70

Exercise 2.17

Define the integrity property of reliable communication and list all the possible threats to integrity from users and from system components. What measures can be taken to ensure the integrity property in the face of each of these sources of threats.

pages 71, 74

Exercise 2.18

Describe possible occurrences of each of the main types of security threat (threats to processes, threats to communication channels, denial of service) that might occur in the Internet.

page 74, 75

Exercises for Chapter 3:
Networking and Internetworking

From Coulouris, Dollimore, Kindberg and Blair
Distributed Systems: Concepts and Design
Edition 5, © Addison-Wesley 2012
Exercise 3.1

A client sends a 200 byte request message to a service, which produces a response containing 5000 bytes. Estimate the total time to complete the request in each of the following cases, with the performance assumptions listed below:

i) Using connectionless (datagram) communication (for example, UDP);  
   (Latency per packet (local or remote, incurred on both send and receive): 6 ms, Data transfer rate: 10 Mbps, MTU: 1000 bytes, Server request processing time: 2 ms)

ii) Using connection-oriented communication (for example, TCP);  

   The server process is in the same machine as the client.

Assume that the network is lightly loaded.

Exercise 3.2

The Internet is far too large for any router to hold routing information for all destinations. How does the Internet routing scheme deal with this issue?

pages 98, 114

Exercise 3.3

What is the task of an Ethernet switch? What tables does it maintain?

pages 105, 130

Exercise 3.4

Make a table similar to Figure 3.5 describing the work done by the software in each protocol layer when Internet applications and the TCP/IP suite are implemented over an Ethernet.

pages 94, 122, 130

Exercise 3.5

How has the end-to-end argument [Saltzer et al. 1984] been applied to the design of the Internet? Consider how the use of a virtual circuit network protocol in place of IP would impact the feasibility of the World Wide Web.

pages 61, 96, 106, [www.reed.com]

Exercise 3.6

Can we be sure that no two computers in the Internet have the same IP addresses?

page 108
Exercise 3.7

Compare connectionless (UDP) and connection-oriented (TCP) communication for the implementation of each of the following application-level or presentation-level protocols:

i) virtual terminal access (for example, Telnet);
ii) file transfer (for example, FTP);
iii) user location (for example, rwho, finger);
iv) information browsing (for example, HTTP);
v) remote procedure call.

Exercise 3.8

Explain how it is possible for a sequence of packets transmitted through a wide area network to arrive at their destination in an order that differs from that in which they were sent. Why can’t this happen in a local network? Can it happen in an ATM network?

pages 97, 131

Exercise 3.9

A specific problem that must be solved in remote terminal access protocols such as Telnet is the need to transmit exceptional events such as ‘kill signals’ from the ‘terminal’ to the host in advance of previously-transmitted data. Kill signals should reach their destination ahead of any other ongoing transmissions. Discuss the solution of this problem with connection-oriented and connectionless protocols.

page 122

Exercise 3.10

What are the disadvantages of using network-level broadcasting to locate resources:

i) in a single Ethernet?
ii) in an intranet?

To what extent is Ethernet multicast an improvement on broadcasting?

page 130

Exercise 3.11

Suggest a scheme that improves on MobileIP for providing access to a web server on a mobile device which is sometimes connected to the Internet by mobile phone and at other times has a wired connection to the Internet at one of several locations.

page 120

Exercise 3.12

Show the sequence of changes to the routing tables in Figure 3.8 that would occur (according to the RIP algorithm given in Figure 3.9) after the link labelled 3 in Figure 3.7 is broken.

pages 98–101
Exercise 3.13

Use the diagram in Figure 3.13 as a basis for an illustration showing the segmentation and encapsulation of an HTTP request to a server and the resulting reply. Assume that request is a short HTTP message, but the reply includes at least 2000 bytes of HTML.

page 93, 107

Exercise 3.14

Consider the use of TCP in a Telnet remote terminal client. How should the keyboard input be buffered at the client? Investigate Nagle’s and Clark’s algorithms [Nagle 1984, Clark 1982] for flow control and compare them with the simple algorithm described on page 103 when TCP is used by
(a) a web server,
(b) a Telnet application,
(c) a remote graphical application with continuous mouse input.

pages 102, 123

Exercise 3.15

Construct a network diagram similar to Figure 3.10 for the local network at your institution or company.

page 104

Exercise 3.16

Describe how you would configure a firewall to protect the local network at your institution or company. What incoming and outgoing requests should it intercept?

page 125

Exercise 3.17

How does a newly-installed personal computer connected to an Ethernet discover the IP addresses of local servers? How does it translate them to Ethernet addresses?

page 111

Exercise 3.18

Can firewalls prevent denial of service attacks such as the one described on page 96? What other methods are available to deal with such attacks?

page 112, 125
Exercises for Chapter 4:
Interprocess Communication

Exercise 4.1
Is it conceivably useful for a port to have several receivers? page 148

Exercise 4.2
A server creates a port which it uses to receive requests from clients. Discuss the design issues concerning the relationship between the name of this port and the names used by clients. page 148

Exercise 4.3
The programs in Figure 4.3 and Figure 4.4 are available at cdk5.net/ipc. Use them to make a test kit to determine the conditions in which datagrams are sometimes dropped. Hint: the client program should be able to vary the number of messages sent and their size; the server should detect when a message from a particular client is missed. page 156

Exercise 4.4
Use the program in Figure 4.3 to make a client program that repeatedly reads a line of input from the user, sends it to the server in a UDP datagram message, then receives a message from the server. The client sets a timeout on its socket so that it can inform the user when the server does not reply. Test this client program with the server in Figure 4.4. page 150

Exercise 4.5
The programs in Figure 4.5 and Figure 4.6 are available at cdk5.net/ipc. Modify them so that the client repeatedly takes a line of user’s input and writes it to the stream and the server reads repeatedly from the stream, printing out the result of each read. Make a comparison between sending data in UDP datagram messages and over a stream. page 153
Exercise 4.6

Use the programs developed in Exercise 4.5 to test the effect on the sender when the receiver crashes and vice-versa.  

page 153

Exercise 4.7

Sun XDR marshals data by converting it into a standard big-endian form before transmission. Discuss the advantages and disadvantages of this method when compared with CORBA’s CDR.  

page 160

Exercise 4.8

Sun XDR aligns each primitive value on a four byte boundary, whereas CORBA CDR aligns a primitive value of size n on an n-byte boundary. Discuss the trade-offs in choosing the sizes occupied by primitive values.  

page 160

Exercise 4.9

Why is there no explicit data-typing in CORBA CDR?  

page 160

Exercise 4.10

Write an algorithm in pseudocode to describe the serialization procedure described in Section 4.3.2. The algorithm should show when handles are defined or substituted for classes and instances. Describe the serialized form that your algorithm would produce when serializing an instance of the following class Couple.  

```java
class Couple implements Serializable {
    private Person one;
    private Person two;
    public Couple(Person a, Person b) {
        one = a;
        two = b;
    }
}
```

page 162

Exercise 4.11

Write an algorithm in pseudocode to describe deserialization of the serialized form produced by the algorithm defined in Exercise 4.10. Hint: use reflection to create a class from its name, to create a constructor from its parameter types and to create a new instance of an object from the constructor and the argument values.  

page 162
Exercise 4.12

Why can’t binary data be represented directly in XML, for example, by representing it as Unicode byte values? XML elements can carry strings represented as base64. Discuss the advantages or disadvantages of using this method to represent binary data. page 164

Exercise 4.13

Define a class whose instances represent remote object references. It should contain information similar to that shown in Figure 4.13 and should provide access methods needed by the request-reply protocol. Explain how each of the access methods will be used by that protocol. Give a justification for the type chosen for the instance variable containing information about the interface of the remote object. page 168

Exercise 4.14

IP multicast provides a service that suffers from omission failures. Make a test kit, possibly based on the program in Figure 4.20, to discover the conditions under which a multicast message is sometimes dropped by one of the members of the multicast group. The test kit should be designed to allow for multiple sending processes. page 170

Exercise 4.15

Outline the design of a scheme that uses message retransmissions with IP multicast to overcome the problem of dropped messages. Your scheme should take the following points into account:

1. there may be multiple senders;
2. generally only a small proportion of messages are dropped;
3. unlike the request-reply protocol, recipients may not necessarily send a message within any particular time limit.

Assume that messages that are not dropped arrive in sender ordering. page 173

Exercise 4.16

Your solution to Exercise 4.15 should have overcome the problem of dropped messages in IP multicast. In what sense does your solution differ from the definition of reliable multicast? page 173

Exercise 4.17

Devise a scenario in which multicasts sent by different clients are delivered in different orders at two group members. Assume that some form of message retransmissions are in use, but that messages that are not dropped arrive in sender ordering. Suggest how recipients might remedy this situation. page 173
Exercise 4.18

Revisit the Internet architecture as introduced in Chapter 3 (see Figures 3.12 and 3.14). What impact does the introduction of overlay networks have on this architecture, and in particular on the programmer’s conceptual view of the Internet?  

page 175

Exercise 4.19

What are the main arguments for adopting a super node approach in Skype?  

page 177

Exercise 4.20

As discussed in Section 4.6, MPI offers a number of variants of send including the MPI_Rsend operation, which assumes the receiver is ready to receive at the time of sending. What optimizations in implementation are possible if this assumption is correct and what are the repercussions of this assumption being false?  

page 180

Exercise 5.1

Define a class whose instances represent request and reply messages as illustrated in Figure 5.4. The class should provide a pair of constructors, one for request messages and the other for reply messages, showing how the request identifier is assigned. It should also provide a method to marshal itself into an array of bytes and to unmarshal an array of bytes into an instance.  

page 188

Exercise 5.2

Program each of the three operations of the request-reply protocol in Figure 5.3, using UDP communication, but without adding any fault-tolerance measures. You should use the classes you defined in the previous chapter for remote object references (Exercise 4.13) and above for request and reply messages (Exercise 5.1).  

page 187
Exercise 5.3

Give an outline of the server implementation, showing how the operations `getRequest` and `sendReply` are used by a server that creates a new thread to execute each client request. Indicate how the server will copy the requestid from the request message into the reply message and how it will obtain the client IP address and port.  

Exercise 5.4

Define a new version of the `doOperation` method that sets a timeout on waiting for the reply message. After a timeout, it retransmits the request message n times. If there is still no reply, it informs the caller.  

Exercise 5.5

Describe a scenario in which a client could receive a reply from an earlier call.  

Exercise 5.6

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

Exercise 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 not be associated with any state?  

Exercise 5.8

Explain the design choices that are relevant to minimizing the amount of reply data held at a server. Compare the storage requirements when the RR and RRA protocols are used.
Exercise 5.9

Assume the RRA protocol is in use. How long should servers retain unacknowledged reply data? Should servers repeatedly send the reply in an attempt to receive an acknowledgement?

Page 191

Exercise 5.10

Why might the number of messages exchanged in a protocol be more significant to performance than the total amount of data sent? Design a variant of the RRA protocol in which the acknowledgement is piggy-backed on – that is, transmitted in the same message as – the next request where appropriate, and otherwise sent as a separate message. (Hint: use an extra timer in the client.)

Page 191

Exercise 5.11

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.

Which of the parameters of these two procedures are input and which are output parameters?

Page 195

Exercise 5.12

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

Section 4.2.4 and page 198

Exercise 5.13

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

Figure 5.8 and page 198

Exercise 5.14

The Election service must ensure that a vote is recorded whenever any user thinks they 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 would you recommend at-most-once call semantics?

Page 199
Exercise 5.15

A request-reply protocol is implemented over a communication service with omission failures to provide at-least-once RMI invocation semantics. In the first case the implementor assumes an asynchronous distributed system. In the second case the implementor assumes that the maximum time for the communication and the execution of a remote method is $T$. In what way does the latter assumption simplify the implementation? page 198

Exercise 5.16

Outline an implementation for the Election service that ensures that its records remain consistent when it is accessed concurrently by multiple clients. page 199

Exercise 5.17

Assume the Election service is implemented in RMI and must ensure that all votes are safely stored even when the server process crashes. Explain how this can be achieved with reference to the implementation outline in your answer to Exercise 5.16. pages 213–214.

Exercise 5.18

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:

$$\text{byte[]} \text{doOperation} (\text{RemoteObjectRef o, Method m, byte[]} \text{arguments});$$

Hint: an instance variable of the proxy class should hold a remote object reference (see Exercise 4.13).

Figure 5.3, page 224

Exercise 5.19

Show how to generate a client proxy class using a language such as C++ that does not support reflection, for example from the CORBA interface definition given in your answer to Exercise 5.13. Give the details of the implementation of one of the methods in this class, which should call the method doOperation defined in Figure 5.3. Page 211

Exercise 5.10

Explain how to use Java reflection to construct a generic dispatcher. Give Java code for a dispatcher whose signature is:

$$\text{public void dispatch(Object target, Method mMethod, byte[] args)}$$

The arguments supply the target object, the method to be invoked and the arguments for that method in an array of bytes. page 224
Exercise 5.21

Exercise 5.18 required the client to convert Object arguments into an array of bytes before invoking doOperation and Exercise 5.20 required the dispatcher to convert an array of bytes into an array of Objects before invoking the method. Discuss the implementation of a new version of doOperation with the following signature:

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

which uses the ObjectOutputStream and ObjectInputStream classes to stream the request and reply messages between client and server over a TCP connection. How would these changes affect the design of the dispatcher?

Section 4.3.2 and page 224

Exercise 5.22

A client makes remote procedure calls to a server. The client takes 5 milliseconds to compute the arguments for each request, and the server takes 10 milliseconds to process each request. The local operating system processing time for each send or receive operation is 0.5 milliseconds, and the network time to transmit each request or reply message is 3 milliseconds. Marshalling or unmarshalling takes 0.5 milliseconds per message.

Calculate the time taken by the client to generate and return from two requests:

(i) if it is single-threaded, and
(ii) if it has two threads that can make requests concurrently on a single processor.

You can ignore context-switching times. Is there a need for asynchronous RPC if client and server processes are threaded?

Exercise 5.23

Design a remote object table that can support distributed garbage collection as well as translating between local and remote object references. Give an example involving several remote objects and proxies at various sites to illustrate the use of the table. Show the changes in the table when one of the proxies becomes unreachable.

Exercise 5.24

A simpler version of the distributed garbage collection algorithm described in Section 5.2.6 just invokes addRef at the site where a remote object lives whenever a proxy is created and removeRef whenever a proxy is deleted. Outline all the possible effects of communication and process failures on the algorithm. Suggest how to overcome each of these effects, but without using leases.

Exercise 5.17

Explain how a forwarding observer may be used to enhance the reliability and performance of objects of interest in an event service.

Exercise 5.18

Suggest ways in which observers can be used to improve the reliability or performance of your solution to Exercise 5.15
Exercises for Chapter 6: Operating System Support

Exercise 6.1
Discuss each of the tasks of encapsulation, concurrent processing, protection, name resolution, communication of parameters and results, and scheduling in the case of the UNIX file service (or that of another kernel that is familiar to you).

Exercise 6.2
Why are some system interfaces implemented by dedicated system calls (to the kernel), and others on top of message-based system calls?

Exercise 6.3
Smith decides that every thread in his processes ought to have its own protected stack – all other regions in a process would be fully shared. Does this make sense?

Exercise 6.4
Should signal (software interrupt) handlers belong to a process or to a thread?

Exercise 6.5
Discuss the issue of naming applied to shared memory regions.
Exercise 6.6

Suggest a scheme for balancing the load on a set of computers. You should discuss:

i) what user or system requirements are met by such a scheme;
ii) to what categories of applications it is suited;
iii) how to measure load and with what accuracy; and
iv) how to monitor load and choose the location for a new process. Assume that processes may not be migrated.

How would your design be affected if processes could be migrated between computers? Would you expect process migration to have a significant cost?

Page 217

Exercise 6.7

Explain the advantage of copy-on-write region copying for UNIX, where a call to fork is typically followed by a call to exec. What should happen if a region that has been copied using copy-on-write is itself copied?

Page 219

Exercise 6.8

A file server uses caching, and achieves a hit rate of 80%. File operations in the server cost 5 ms of CPU time when the server finds the requested block in the cache, and take an additional 15 ms of disk I/O time otherwise. Explaining any assumptions you make, estimate the server’s throughput capacity (average requests/sec) if it is:

i) single-threaded;
ii) two-threaded, running on a single processor;
iii) two-threaded, running on a two-processor computer.

Page 220

Exercise 6.9

Compare the worker pool multi-threading architecture with the thread-per-request architecture.

Page 221

Exercise 6.10

What thread operations are the most significant in cost?

Page 223

Exercise 6.11

A spin lock (see Bacon [1998]) is a boolean variable accessed via an atomic test-and-set instruction, which is used to obtain mutual exclusion. Would you use a spin lock to obtain mutual exclusion between threads on a single-processor computer?

Page 227
Exercise 6.12
Explain what the kernel must provide for a user-level implementation of threads, such as Java on UNIX.

page 228

Exercise 6.13
Do page faults present a problem for user-level threads implementations?

page 228

Exercise 6.14
Explain the factors that motivate the hybrid scheduling approach of the 'scheduler activations' design (instead of pure user-level or kernel-level scheduling).

page 229

Exercise 6.15
Explain the factors that motivate the hybrid scheduling approach of the 'scheduler activations' design (instead of pure user-level or kernel-level scheduling).

page 230

Exercise 6.16
Network transmission time accounts for 20% of a null RPC and 80% of an RPC that transmits 1024 user bytes (less than the size of a network packet). By what percentage will the times for these two operations improve if the network is upgraded from 10 megabits/second to 100 megabits/second?

page 234

Exercise 6.17
A 'null' RMI that takes no parameters, calls an empty procedure and returns no values delays the caller for 2.0 milliseconds. Explain what contributes to this time.
In the same RMI system, each 1K of user data adds an extra 1.5 milliseconds. A client wishes to fetch 32K of data from a file server. Should it use one 32K RMI or 32 1K RMIs?

page 234
Exercise 6.18
Which factors identified in the cost of a remote invocation also feature in message passing? page 235

Exercise 6.19
Explain how a shared region could be used for a process to read data written by the kernel. Include in your explanation what would be necessary for synchronization. page 236

Exercise 6.20
i) Can a server invoked by lightweight procedure calls control the degree of concurrency within it?
ii) Explain why and how a client is prevented from calling arbitrary code within a server under lightweight RPC.
iii) Does LRPC expose clients and servers to greater risks of mutual interference than conventional RPC (given the sharing of memory)? page 237

Exercise 6.21
A client makes RMIs to a server. The client takes 5 ms to compute the arguments for each request, and the server takes 10ms to process each request. The local OS processing time for each send or receive operation is 0.5 ms, and the network time to transmit each request or reply message is 3 ms. Marshalling or unmarshalling takes 0.5 ms per message. Estimate the time taken by the client to generate and return from 2 requests (i) if it is single-threaded, and (ii) if it has two threads which can make requests concurrently on a single processor. Is there a need for asynchronous RMI if processes are multi-threaded? page 240

Exercise 6.22
Explain what is security policy and what are the corresponding mechanisms in the case of a multi-user operating system such as UNIX. page 242

Exercise 6.23
Explain the program linkage requirements that must be met if a server is to be dynamically loaded into the kernel’s address space, and how these differ from the case of executing a server at user level. page 244
Exercise 6.24

How could an interrupt be communicated to a user-level server?

page 245

Instructor’s Guide for Coulouris, Dollimore

Exercise 6.25

On a certain computer we estimate that, regardless of the OS it runs, thread scheduling costs about 50 microsec, a null procedure call 1 millisec, a context switch to the kernel 20 microsec and a domain transition 40 microsec. For each of Mach and SPIN, estimate the cost to a client of calling a dynamically loaded null procedure.

page 245

Instructor’s Guide for Coulouris, Dollimore

Exercises for Chapter 7:

From Coulouris, Dollimore and Kindberg
Distributed Systems: Concepts and Design
Edition 3, © Addison-Wesley 2001

Exercise 7.1

Describe some of the physical security policies in your organization. Express them in terms that could be implemented in a computerized door locking system.

page 252

Instructor’s Guide for Coulouris, Dollimore

Exercise 7.2

Describe some of the ways in which conventional email is vulnerable to eavesdropping, masquerading, tampering, replay, denial of service. Suggest methods by which email could be protected against each of these forms of attack.

page 254

Instructor’s Guide for Coulouris, Dollimore

Exercise 7.3

Initial exchanges of public keys are vulnerable to the man-in-the-middle attack. Describe as many defences against it as you can.

page 261.298

Instructor’s Guide for Coulouris, Dollimore
Exercise 7.4
PGP is widely used to secure email communication. Describe the steps that a pair of users using PGP must take before they can exchange email messages with privacy and authenticity guarantees. What scope is there to make the preliminary key negotiation invisible to users? (The PGP negotiation is an instance of the hybrid scheme.)

Exercise 7.5
How would email be sent to a large list of recipients using PGP or a similar scheme? Suggest a scheme that is simpler and faster when the list is used frequently.

Exercise 7.6
The implementation of the TEA symmetric encryption algorithm given in Figures 7.8–7.10 is not portable between all machine architectures. Explain why. How could a message encrypted using the TEA implementation be transmitted to decrypt it correctly on all other architectures?

Exercise 7.7
Modify the TEA application program in Figure 7.10 to use cipher block chaining (CBC).

Exercise 7.8
Construct a stream cipher application based on the program in Figure 7.10.

Exercise 7.9
Estimate the time required to crack a 56-bit DES key by a brute-force attack using a 500 MIPS (million instruction per second) workstation, assuming that the inner loop for a brute-force attack program involves around 10 instructions per key value, plus the time to encrypt an 8-byte plaintext (see Figure 7.14). Perform the same calculation for a 128-bit IDEA key. Extrapolate your calculations to obtain the cracking time for a 50,000 MIPS parallel processor (or an Internet consortium with similar processing power).
Exercise 7.10

In the Needham and Shroeder authentication protocol with secret keys, explain why the following version of message 5 is not secure:

\[ A \rightarrow B: (N_B)^k \]

Exercise 8.1

Why is there no `open` or `close` operation in our interface to the flat file service or the directory service. What are the differences between our directory service `Lookup` operation and the UNIX `open`?

Exercise 8.2

Outline methods by which a client module could emulate the UNIX file system interface using our model file service.

Exercise 8.3

Write a procedure

\[ \text{PathLookup(Pathname, Dir) } \rightarrow \text{UFID} \]

that implements `Lookup` for UNIX-like pathnames based on our model directory service.

Exercise 8.4

Why should UFIDs be unique across all possible file systems? How is uniqueness for UFIDs ensured?
Exercise 8.5

To what extent does Sun NFS deviate from one-copy file update semantics? Construct a scenario in which two user-level processes sharing a file would operate correctly in a single UNIX host but would observe inconsistencies when running in different hosts.

page 330

Instructor’s Guide for Coulouris, Dollimore

Exercise 8.6

Sun NFS aims to support heterogeneous distributed systems by the provision of an operating system-independent file service. What are the key decisions that the implementer of an NFS server for an operating system other than UNIX would have to take? What constraints should an underlying filing system obey to be suitable for the implementation of NFS servers?

page 324

Instructor’s Guide for Coulouris, Dollimore

Exercise 8.7

What data must the NFS client module hold on behalf of each user-level process?

page 324-5

Instructor’s Guide for Coulouris, Dollimore

Exercise 8.8

Outline client module implementations for the UNIX open() and read() system calls, using the NFS RPC calls of Figure 8.9, (i) without, and (ii) with a client cache.

page 326,330

Instructor’s Guide for Coulouris, Dollimore

Exercise 8.9

Explain why the RPC interface to early implementations of NFS is potentially insecure. The security loophole has been closed in NFS 3 by the use of encryption. How is the encryption key kept secret? Is the security of the key adequate?

page 325,332

Instructor’s Guide for Coulouris, Dollimore

Exercise 8.10

After the timeout of an RPC call to access a file on a hard-mounted file system the NFS client module does not return control to the user-level process that originated the call. Why?

page 327

Instructor’s Guide for Coulouris, Dollimore
Exercise 8.11
How does the NFS Automounter help to improve the performance and scalability of NFS?

Exercise 8.12
How many lookup calls are needed to resolve a 5-part pathname (for example, /usr/users/jim/code/xyz.c) for a file that is stored on an NFS server? What is the reason for performing the translation step-by-step?

Exercise 8.13
What condition must be fulfilled by the configuration of the mount tables at the client computers for access transparency to be achieved in an NFS-based filing system?

Exercise 8.14
How does AFS gain control when an open or close system call referring to a file in the shared file space is issued by a client?

Exercise 8.15
Compare the update semantics of UNIX when accessing local files with those of NFS and AFS. Under what circumstances might clients become aware of the differences?

Exercise 8.16
How does AFS deal with the risk that callback messages may be lost?
Exercise 8.17
Which features of the AFS design make it more scalable than NFS? What are the limits on its scalability, assuming that servers can be added as required? Which recent developments offer greater scalability?

Exercise 9.1
Describe the names (including identifiers) and attributes used in a distributed file service such as NFS (see Chapter 8).

Exercise 9.2
Discuss the problems raised by the use of aliases in a name service, and indicate how, if at all, these may be overcome.

Exercise 9.3
Explain why iterative navigation is necessary in a name service in which different name spaces are partially integrated, such as the file naming scheme provided by NFS.

Exercise 9.4
Describe the problem of unbound names in multicast navigation. What is implied by the installation of a server for responding to lookups of unbound names?
Exercise 9.5
How does caching help a name service’s availability?

Exercise 9.6
Discuss the absence of a syntactic distinction (such as use of a final `.`) between absolute and relative names in DNS.

Exercise 9.7
Investigate your local configuration of DNS domains and servers. You may find a program such as nslookup installed on UNIX systems, which enables you to carry out individual name server queries.

Exercise 9.8
Why do DNS root servers hold entries for two-level names such as yahoo.com and purdue.edu, rather than one-level names such as edu and com?

Exercise 9.9
Which other name server addresses do DNS name servers hold by default, and why?

Exercise 9.10
Why might a DNS client choose recursive navigation rather than iterative navigation? What is the relevance of the recursive navigation option to concurrency within a name server?
Exercise 9.11

When might a DNS server provide multiple answers to a single name lookup, and why?

page 369

Exercise 9.12

The Jini lookup service matches service offers to client requests based on attributes or on Java typing. Explain with examples the difference between these two methods of matching. What is the advantage of allowing both sorts of matching?

page 372

Exercise 9.13

Explain how the Jini lookup service uses leases to ensure that the list of services registered with a lookup server remains current although services may crash or become inaccessible.

page 372

Exercise 9.14

Describe the use of IP multicast and group names in the Jini ‘discovery’ service which allows clients and servers to locate lookup servers.

page 372

Exercise 9.15

GNS does not guarantee that all copies of entries in the naming database are up-to-date. How are clients of GNS likely to become aware that they have been given an out-of-date entry? Under what circumstances might it be harmful?

page 374

Exercise 9.16

Discuss the potential advantages and drawbacks in the use of a X.500 directory service in place of DNS and the Internet mail delivery programs. Sketch the design of a mail delivery system for an internetwork in which all mail users and mail hosts are registered in an X.500 database.

page 378
Exercise 9.17

What security issues are liable to be relevant to a directory service such as X500 operating within an organization such as a university?

Exercise 10.1

Why is computer clock synchronization necessary? Describe the design requirements for a system to synchronize the clocks in a distributed system.

Exercise 10.2

A clock is reading 10:27:54.0 (hr:min:sec) when it is discovered to be 4 seconds fast. Explain why it is undesirable to set it back to the right time at that point and show (numerically) how it should be adjusted so as to be correct after 8 seconds has elapsed.

Exercise 10.3

A scheme for implementing at-most-once reliable message delivery uses synchronized clocks to reject duplicate messages. Processes place their local clock value (a 'timestamp') in the messages they send. Each receiver keeps a table giving, for each sending process, the largest message timestamp it has seen. Assume that clocks are synchronized to within 100 ms, and that messages can arrive at most 50 ms after transmission.

(i) When may a process ignore a message bearing a timestamp $T$, if it has recorded the last message received from that process as having timestamp $T'$?

(ii) When may a receiver remove a timestamp 175,000 (ms) from its table? (Hint: use the receiver's local clock value.)

(iii) Should the clocks be internally synchronized or externally synchronized?

Exercise 10.4

A client attempts to synchronize with a time server. It records the round-trip times and timestamps returned by the server in the table below. Which of these times should it use to set its clock? To what time should it set it? Estimate the accuracy of the setting with respect to the server's clock. If it is known that the time between sending and receiving a message in the system concerned is at least 8 ms, do your answers change?

<table>
<thead>
<tr>
<th>Round-trip (ms)</th>
<th>Time (hr:min:sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>10:54:23.674</td>
</tr>
<tr>
<td>25</td>
<td>10:54:25.450</td>
</tr>
<tr>
<td>20</td>
<td>10:54:28.342</td>
</tr>
</tbody>
</table>
Exercise 10.5

In the system of Exercise 10.4 it is required to synchronize a file server’s clock to within ±1 millisecond. Discuss this in relation to Cristian’s algorithm.

Exercise 10.6

What reconfigurations would you expect to occur in the NTP synchronization subnet?

Exercise 10.7


Exercise 10.8

Discuss the factors to be taken into account when deciding to which NTP server a client should synchronize its clock.

Exercise 10.9

Discuss how it is possible to compensate for clock drift between synchronization points by observing the drift rate over time. Discuss any limitations to your method.

Exercise 10.10

By considering a chain of zero or more messages connecting events e and e’ and using induction, show that e → e’ ⇒ L(e) < L(e').
Exercise 10.11

Show that $V[i] \leq V[j]$.

Exercise 10.12

In a similar fashion to Exercise 10.10, show that $e \rightarrow e' \Rightarrow V(e) < V(e')$.

Exercise 10.13

Using the result of Exercise 10.11, show that if events $e$ and $e'$ are concurrent then neither $V(e) \leq V(e')$ nor $V(e') \leq V(e)$.

Hence show that if $V(e) < V(e')$ then $e \rightarrow e'$

Exercise 10.14

Two processes $P$ and $Q$ are connected in a ring using two channels, and they constantly rotate a message $m$. At any one time, there is only one copy of $m$ in the system. Each process's state consists of the number of times it has received $m$. At a certain point, $P$ sends the message and its state is 101. Immediately after sending $m$, $P$ initiates the snapshot algorithm. Explain the operation of the algorithm in this case, giving the possible global state(s) reported by it.

Exercise 10.15

The figure above shows events occurring for each of two processes, $P_1$ and $P_2$. Arrows between processes denote message transmission. Draw and label the lattice of consistent states ($P_1$ state, $P_2$ state), beginning with the initial state (0,0).

Exercise 10.16

Jones is running a collection of processes $p_1, p_2, \ldots, p_N$. Each process $p_i$ contains a variable $v_i$. She wishes to determine whether all the variables $v_1, v_2, \ldots, v_N$ were ever equal in the course of the execution.

(i) Jones’ processes run in a synchronous system. She uses a monitor process to determine whether the variables were ever equal. When should the application processes communicate with the monitor process, and what should their messages contain?

(ii) Explain the statement possibly $(v_1 = v_2 = \ldots = v_N)$. How can Jones determine whether this statement is true of her execution?
Exercises for Chapter 11:  
COORDINATION AND AGREEMENT  

From: Coulouris, Dollimore and Kindberg  
Distributed Systems:  
Concepts and Design  
Edition 3, © Addison-Wesley 2001

Exercise 11.1  
Is it possible to implement either a reliable or an unreliable (process) failure detector using an unreliable communication channel?  

Exercise 11.2  
If all client processes are single-threaded, is mutual exclusion condition ME3, which specifies entry in happened-before order, relevant?  

Exercise 11.3  
Give a formula for the maximum throughput of a mutual exclusion system in terms of the synchronization delay.  

Exercise 11.4  
In the central server algorithm for mutual exclusion, describe a situation in which two requests are not processed in happened-before order.  

Exercise 11.5  
Adapt the central server algorithm for mutual exclusion to handle the crash failure of any client (in any state), assuming that the server is correct and given a reliable failure detector. Comment on whether the resultant system is fault tolerant. What would happen if a client that possesses the token is wrongly suspected to have failed?
Exercise 11.6

Give an example execution of the ring-based algorithm to show that processes are not necessarily granted entry to the critical section in happened-before order.  

page 427

Exercise 11.7

In a certain system, each process typically uses a critical section many times before another process requires it. Explain why Ricart and Agrawala’s multicast-based mutual exclusion algorithm is inefficient for this case, and describe how to improve its performance. Does your adaptation satisfy liveness condition ME2?  

page 429

Exercise 11.8

In the Bully algorithm, a recovering process starts an election and will become the new coordinator if it has a higher identifier than the current incumbent. Is this a necessary feature of the algorithm?  

page 434

Exercise 11.9

Suggest how to adapt the Bully algorithm to deal with temporary network partitions (slow communication) and slow processes.  

page 436

Exercise 11.10

Devise a protocol for basic multicast over IP multicast.  

page 438

Exercise 11.11

How, if at all, should the definitions of integrity, agreement and validity for reliable multicast change for the case of open groups?  

page 439
Exercise 11.12

Explain why reversing the order of the lines

`R-deliver m` and

`if (q≠p) then B-multicast(g, m); end if`

in Figure 11.10 makes the algorithm no longer satisfy uniform agreement. Does the reliable multicast algorithm based on IP multicast satisfy uniform agreement?  

*page 440*

Exercise 11.13

Explain whether the algorithm for reliable multicast over IP multicast works for open as well as closed groups. Given any algorithm for closed groups, how, simply, can we derive an algorithm for open groups?  

*page 440*

Exercise 11.14

Consider how to address the impractical assumptions we made in order to meet the validity and agreement properties for the reliable multicast protocol based on IP multicast. Hint: add a rule for deleting retained messages when they have been delivered everywhere; and consider adding a dummy ‘heartbeat’ message, which is never delivered to the application, but which the protocol sends if the application has no message to send.  

*page 440*

Exercise 11.15

Show that the FIFO-ordered multicast algorithm does not work for overlapping groups, by considering two messages sent from the same source to two overlapping groups, and considering a process in the intersection of those groups. Adapt the protocol to work for this case. Hint: processes should include with their messages the latest sequence numbers of messages sent to all groups.  

*page 445*

Exercise 11.16

Show that, if the basic multicast that we use in the algorithm of Figure 11.14 is also FIFO-ordered, then the resultant totally-ordered multicast is also causally ordered. Is it the case that any multicast that is both FIFO-ordered and totally ordered is thereby causally ordered?  

*page 446*

Exercise 11.17

Suggest how to adapt the causally ordered multicast protocol to handle overlapping groups.  

*page 449*
Exercise 11.18

In discussing Maekawa’s mutual exclusion algorithm, we gave an example of three subsets of a set of three processes that could lead to a deadlock. Use these subsets as multicast groups to show how a pairwise total ordering is not necessarily acyclic.

page 450

Exercise 11.19

Construct a solution to reliable, totally ordered multicast in a synchronous system, using a reliable multicast and a solution to the consensus problem.

page 450

Exercise 11.20

We gave a solution to consensus from a solution to reliable and totally ordered multicast, which involved selecting the first value to be delivered. Explain from first principles why, in an asynchronous system, we could not instead derive a solution by using a reliable but not totally ordered multicast service and the ‘majority’ function. (Note that, if we could, then this would contradict the impossibility result of Fischer et al.) Hint: consider slow/failed processes.

page 455

Exercise 11.21

Show that byzantine agreement can be reached for three generals, with one of them faulty, if the generals digitally sign their messages.

page 457

Exercise 11.22

Explain how to adapt the algorithm for reliable multicast over IP multicast to eliminate the hold-back queue – so that a received message that is not a duplicate can be delivered immediately, but without any ordering guarantees. Hint: use sets instead of sequence numbers to represent the messages that have been delivered so far.

page 441
Exercise 12.1

The TaskBag is a service whose functionality is to provide a repository for task descriptions. It enables clients running on several computers to carry out parts of a computation in parallel. A master process places descriptions of sub-tasks of a computation in the TaskBag, and worker processes select tasks from the TaskBag and carry them out, returning descriptions of results to the TaskBag. The master then collects the results and combines them to produce the final result.

The TaskBag service provides the following operations:
- `aTask()` allows clients to add task descriptions to the bag;
- `takeTask()` allows clients to take task descriptions out of the bag;

A client makes the request `takeTask()`, when a task is not available but may be available soon. Discuss the advantages and drawbacks of the following alternatives:
- (i) the server can reply immediately, telling the client to try again later;
- (ii) make the server operation (and therefore the client) wait until a task becomes available.
- (iii) use callbacks.

A newly created object like `Z` in Exercise 12.4 is sometimes called a phantom. From the point of view of transaction `U`, `Z` is not there at first and then appears (like a ghost). Explain, with an example, how a phantom could occur when an account is deleted.

Exercise 12.2

A server manages the objects `a_1`, `a_2`, …, `a_n`. The server provides two operations for its clients:
- `read(i)` returns the value of `a_i`;
- `write(i, Value)` assigns `Value` to `a_i`.

The transactions `T` and `U` are defined as follows:
- `T`: `x = read(i); y = read(j); write(j, 44); write(i, 33);`
- `U`: `x = read(k); write(i, 55); y = read(j); write(k, 66).

Give three serially equivalent interleavings of the transactions `T` and `U`.

Exercise 12.3

Give serially equivalent interleavings of `T` and `U` in Exercise 12.2 with the following properties: (1) that is strict; (2) that is not strict but could not produce cascading aborts; (3) that could produce cascading aborts.

A task in Exercise 12.4 is defined as follows:
```
T: aBranch.create(Z);
U: z.deposit(10); z.deposit(20);
```

Assume that `Z` does not yet exist. Assume also that the deposit operation does nothing if the account given as argument does not exist. Consider the following interleaving of transactions `T` and `U`:

```
<table>
<thead>
<tr>
<th>T</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>aBranch.create(Z);</td>
<td>z.deposit(10);</td>
</tr>
<tr>
<td></td>
<td>z.deposit(20);</td>
</tr>
</tbody>
</table>
```

State the balance of `Z` after their execution in this order. Are these consistent with serially equivalent executions of `T` and `U`?

Exercise 12.5

The 'transfer' transactions `T` and `U` are defined as:
- `T`: `a.withdraw(4); b.deposit(4);`
- `U`: `c.withdraw(3); b.deposit(3);`

Suppose that they are structured as pairs of nested transactions:
- `T1`: `a.withdraw(4); T2: b.deposit(4);`
- `U1`: `c.withdraw(3); U2: b.deposit(3);`

Compare the number of serially equivalent interleavings of `T1`, `T2`, `U1` and `U2` with the number of serially equivalent interleavings of `T` and `U`. Explain why the use of these nested transactions generally permits a larger number of serially equivalent interleavings than non-nested ones.

Exercise 12.6

The operation `create` inserts a new bank account at a branch. The transactions `T` and `U` are defined as follows:
```
T: aBranch.create(Z);
U: z.deposit(10); z.deposit(20).
```

Assume that `Z` does not yet exist. Assume also that the deposit operation does nothing if the account given as argument does not exist. Consider the following interleaving of transactions `T` and `U`:

```
<table>
<thead>
<tr>
<th>T</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>aBranch.create(Z);</td>
<td>z.deposit(10);</td>
</tr>
<tr>
<td></td>
<td>z.deposit(20);</td>
</tr>
</tbody>
</table>
```

State the balance of `Z` after their execution in this order. Are these consistent with serially equivalent executions of `T` and `U`?
Exercise 12.10

Consider a relaxation of two-phase locks in which read-only transactions can release read locks early. Would a read-only transaction have consistent retrievals? Would the objects become inconsistent? Illustrate your answer with the following transactions 

T: \( x = \text{read}(i); y = \text{read}(j); \)
U: \( \text{write}(i,55); \text{write}(j,66); \)

in which initial values of \( a_i \) and \( a_j \) are 10 and 20.

page 480

Exercise 12.11

The executions of transactions are strict if read and write operations on an object are delayed until all transactions that previously wrote that object have either committed or aborted. Explain how the locking rules in Figure 12.16 ensure strict executions.

page 486
Exercise 12.12

Describe how a non-recoverable situation could arise if write locks are released after the last operation of a transaction but before its commitment.

page 480

Exercise 12.13

Explain why executions are always strict, even if read locks are released after the last operation of a transaction but before its commitment. Give an improved statement of rule 2 in Figure 12.16.

page 480

Exercise 12.14 (first part)

Consider a deadlock detection scheme for a single server. Describe precisely when edges are added to and removed from the wait-for-graph. Illustrate your answer with respect to the following transactions T, U and V at the server of Exercise 12.8

<table>
<thead>
<tr>
<th>T</th>
<th>U</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>write(1, 66);</td>
<td></td>
</tr>
<tr>
<td></td>
<td>write(1, 55);</td>
<td></td>
</tr>
<tr>
<td></td>
<td>write(1, 77);</td>
<td></td>
</tr>
<tr>
<td></td>
<td>commit</td>
<td></td>
</tr>
</tbody>
</table>

Instructor’s Guide for Coulouris, Dollimore

Exercise 12.14 continued

When U releases its write lock on a, both T and V are waiting to obtain write locks on it. Does your scheme work correctly if T (first come) is granted the lock before V? If your answer is ‘No’, then modify your description.

page 492

Exercise 12.15

Consider hierarchic locks as illustrated in Figure 12.26. What locks must be set when an appointment is assigned to a time slot in week w, day d, at time t? In what order should these locks be set? Does the order in which they are released matter? What locks must be set when the time slots for every day in week w are viewed? Can this be done when the locks for assigning an appointment to a time slot are already set?

page 495

Exercise 12.16

Consider optimistic concurrency control as applied to the transactions T and U defined in Exercise 12.9. Suppose that transactions T and U are active at the same time as one another. Describe the outcome in each of the following cases:

(i) T’s request to commit comes first and backward validation is used;
(ii) U’s request to commit comes first and backward validation is used;
(iii) T’s request to commit comes first and forward validation is used;
(iv) U’s request to commit comes first and forward validation is used.

In each case describe the sequence in which the operations of T and U are performed, remembering that writes are not carried out until after validation.

page 497
Exercise 12.17
Consider the following interleaving of transactions T and U:

\[
\begin{array}{c|c}
T & U \\
\hline
\text{openTransaction} & \text{openTransaction} \\
\hline
y = \text{read}(i); & \text{write}(j, 55); \\
\hline
\text{write}(i, 55); & \text{write}(j, 66); \\
\hline
\text{commit} & \text{commit} \\
\hline
y = \text{read}(i); & \text{write}(j, 44); \\
\hline
\text{write}(j, 44); & \text{commit} \\
\end{array}
\]

The outcome of optimistic concurrency control with backward validation is that T will be aborted because its read operation conflicts with U's write operation on i, although the interleavings are serially equivalent. Suggest a modification to the algorithm that deals with such cases.

Exercise 12.18
Make a comparison of the sequences of operations of the transactions T and U of Exercise 12.8 that are possible under two-phase locking (Exercise 12.9) and under optimistic concurrency control (Exercise 12.16).

T: \( x = \text{read}(i); \) write(j, 44);
U: \( \text{write}(i, 55); \text{write}(j, 66); \)

Exercise 12.19
Consider the use of timestamp ordering with each of the example interleavings of transactions T and U in Exercise 12.9. Initial values of i and j are 10 and 20, respectively, and initial read and write timestamps are t0. Assume that each transaction opens and obtains a timestamp just before its first operation; for example, in (a) T and U get timestamps \( t_1 \) and \( t_2 \) respectively, where \( t_0 < t_1 < t_2 \). Describe in order of increasing time the effects of each operation of T and U. For each operation, state the following:
(i) whether the operation may proceed according to the write or read rule;
(ii) timestamps assigned to transactions or objects;
(iii) creation of tentative objects and their values.

What are the final values of the objects and their timestamps?

Exercise 12.20
Repeat Exercise 12.19 for the following interleavings of transactions T and U:

\[
\begin{array}{c|c}
T & U \\
\hline
\text{openTransaction} & \text{openTransaction} \\
\hline
\text{write}(i, 55); & \text{write}(j, 55); \\
\hline
\text{write}(j, 66); & \text{write}(j, 66); \\
\hline
x = \text{read}(i); & x = \text{read}(i); \\
\hline
\text{write}(j, 44); & \text{commit} \\
\hline
\text{commit} & \text{write}(j, 44); \\
\end{array}
\]

Exercise 12.21
Repeat Exercise 12.20 using multiversion timestamp ordering.

Exercise 12.22
In multiversion timestamp ordering, read operations can access tentative versions of objects. Give an example to show how cascading aborts can happen if all read operations are allowed to proceed immediately.
Exercise 12.23

What are the advantages and drawbacks of multiversion timestamp ordering in comparison with ordinary timestamp ordering.

Exercise 12.24

Make a comparison of the sequences of operations of the transactions $T$ and $U$ of Exercise 12.8 that are possible under two-phase locking (Exercise 12.9) and under optimistic concurrency control (Exercise 12.16)

$T$: $x = \text{read}(i); \text{write}(j, 44)$;
$U$: $\text{write}(i, 55); \text{write}(j, 66)$;

Exercise 13.1

In a decentralized variant of the two-phase commit protocol the participants communicate directly with one another instead of indirectly via the coordinator. In phase 1, the coordinator sends its vote to all the participants. In phase 2, if the coordinator's vote is No, the participants just abort the transaction; if it is Yes, each participant sends its vote to the coordinator and the other participants, each of which decides on the outcome according to the vote and carries it out. Calculate the number of messages and the number of rounds it takes. What are its advantages or disadvantages in comparison with the centralized variant?

Exercise 13.2

A three-phase commit protocol has the following parts:

1. Phase 1: is the same as for two-phase commit.
2. Phase 2: the coordinator collects the votes and makes a decision; if it is No, it aborts and informs participants that voted Yes; if the decision is Yes, it sends a preCommit request to all the participants. Participants that voted Yes wait for a preCommit or doAbort request. They acknowledge preCommit requests and carry out doAbort requests.
3. Phase 3: the coordinator collects the acknowledgments. When all are received, it Commits and sends doCommit to the participants. Participants wait for a doCommit request. When it arrives they Commit.

Exercise 13.3

Explain how the two-phase commit protocol for nested transactions ensures that if the top-level transaction commits, all the right descendants are committed or aborted.
Exercise 13.4

Give an example of the interleavings of two transactions that is serially equivalent at each server but is not serially equivalent globally.  

page 528

Instructor’s Guide for Coulouris, Dollimore

Exercise 13.5

The getDecision procedure defined in Figure 13.4 is provided only by coordinators. Define a new version of getDecision to be provided by participants for use by other participants that need to obtain a decision when the coordinator is unavailable. Assume that any active participant can make a getDecision request to any other active participant. Does this solve the problem of delay during the ‘uncertain’ period? Explain your answer. At what point in the two-phase commit protocol would the coordinator inform the participants of the other participants’ identities (to enable this communication)?

page 520

Instructor’s Guide for Coulouris, Dollimore

Exercise 13.6

Extend the definition of two-phase locking to apply to distributed transactions. Explain how this is ensured by distributed transactions using strict two-phase locking locally.

page 528 and Chapter 12

Instructor’s Guide for Coulouris, Dollimore

Exercise 13.7

Assuming that strict two-phase locking is in use, describe how the actions of the two-phase commit protocol relate to the concurrency control actions of each individual server. How does distributed deadlock detection fit in?

page 520 and 528

Instructor’s Guide for Coulouris, Dollimore

Exercise 13.8

A server uses timestamp ordering for local concurrency control. What changes must be made to adapt it for use with distributed transactions? Under what conditions could it be argued that the two-phase commit protocol is redundant with timestamp ordering?

page 520 and 529

Instructor’s Guide for Coulouris, Dollimore

Exercise 13.9

Consider distributed optimistic concurrency control in which each server performs local backward validation sequentially (that is, with only one transaction in the validate and update phase at one time), in relation to your answer to Exercise 13.4. Describe the possible outcomes when the two transactions attempt to commit. What difference does it make if the servers use parallel validation?

Chapter 12 and page 530

Instructor’s Guide for Coulouris, Dollimore
Exercise 13.10

A centralized global deadlock detector holds the union of local wait-for graphs. Give an example to explain how a phantom deadlock could be detected if a waiting transaction in a deadlock cycle aborts during the deadlock detection procedure.

page 533

Exercise 13.11

Consider the edge-chasing algorithm (without priorities). Give examples to show that it could detect phantom deadlocks.

page 534

Exercise 13.12

A server manages the objects $a_1, a_2, ..., a_n$. It provides two operations for its clients:

- Read($i$) returns the value of $a_i$
- Write($i$, Value) assigns Value to $a_i$

The transactions $T$, $U$ and $V$ are defined as follows:

$T$: $x = \text{Read}(i)$; Write($j$, 44);

$U$: Write($i$, 55); Write($j$, 66);

$V$: Write($k$, 77); Write($k$, 88);

Describe the information written to the log file on behalf of these three transactions if strict two-phase locking is in use and $U$ acquires $a_i$ and $a_j$ before $T$. Describe how the recovery manager would use this information to recover the effects of $T$, $U$ and $V$ when the server is replaced after a crash. What is the significance of the order of the commit entries in the log file?

pages 540-542

Exercise 13.13

The appending of an entry to the log file is atomic, but append operations from different transactions may be interleaved. How does this affect the answer to Exercise 13.12?

pages 540-542

Exercise 13.14

The transactions $T$, $U$ and $V$ of Exercise 13.12 use strict two-phase locking and their requests are interleaved as follows:

<table>
<thead>
<tr>
<th>$T$</th>
<th>$U$</th>
<th>$V$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read($i$)</td>
<td>Write($i$, 55)</td>
<td>Write($k$, 77)</td>
</tr>
<tr>
<td>Write($j$, 44)</td>
<td>Write($j$, 66)</td>
<td>Write($k$, 88)</td>
</tr>
</tbody>
</table>

Assuming that the recovery manager appends the data entry corresponding to each Write operation to the log file immediately instead of waiting until the end of the transaction, describe the information written to the log file on behalf of the transactions $T$, $U$ and $V$. Does early writing affect the correctness of the recovery procedure? What are the advantages and disadvantages of early writing?

pages 540-542

Exercise 13.15

The transactions $T$ and $U$ are run with timestamp ordering concurrency control. Describe the information written to the log file on behalf of $T$ and $U$, allowing for the fact that $U$ has a later timestamp than $T$. Why is it essential that the commit entries in the log file be ordered by timestamps? Describe the effect of recovery if the server crashes (i) between the two Commits and (ii) after both of them.

<table>
<thead>
<tr>
<th>$T$</th>
<th>$U$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read($i$)</td>
<td>Write($i$, 55)</td>
</tr>
<tr>
<td>Write($j$, 44)</td>
<td>Write($j$, 66)</td>
</tr>
<tr>
<td>Write($k$, 88)</td>
<td>Commit</td>
</tr>
</tbody>
</table>

What are the advantages and disadvantages of early writing with timestamp ordering?

pages 540-542

Instructor’s Guide for Coulouris, Dollimore
Exercise 13.16

The transactions $T$ and $U$ in Exercise 13.15 are run with optimistic concurrency control using backward validation and restarting any transactions that fail. Describe the information written to the log file on their behalf. Why is it essential that the commit entries in the log file be ordered by transaction numbers? How are the write sets of committed transactions represented in the log file?

pages 540-542

Exercise 13.17

Suppose that the coordinator of a transaction crashes after it has recorded the intentions list entry but before it has recorded the participant list or sent out the canCommit? requests. Describe how the participants resolve the situation. What will the coordinator do when it recovers? Would it be any better to record the participant list before the intentions list entry?

page 546

Exercise 14.1

Three computers together provide a replicated service. The manufacturers claim that each computer has a mean time between failure of five days; a failure typically takes four hours to fix. What is the availability of the replicated service?

page 554

Exercise 14.2

Explain why a multi-threaded server might not qualify as a state machine.

page 557

Exercise 14.3

In a multi-user game, the players move figures around a common scene. The state of the game is replicated at the players' workstations and at a server, which contains services controlling the game overall, such as collision detection. Updates are multicast to all replicas.

(i) The figures may throw projectiles at one another and a hit debilitates the unfortunate recipient for a limited time. What type of update ordering is required here? Hint: consider the 'throw', 'collide' and 'revive' events.

(ii) The game incorporates magic devices which may be picked up by a player to assist them. What type of ordering should be applied to the pick-up-device operation?

page 558
Exercise 14.4

A router separating process \( p \) from two others, \( q \) and \( r \), fails immediately after \( p \) initiates the multicasting of message \( m \). If the group communication system is view-synchronous, explain what happens to \( p \) next.

Exercise 14.5

You are given a group communication system with a totally ordered multicast operation, and a failure detector. Is it possible to construct view-synchronous group communication from these components alone?

Exercise 14.6

A sync-ordered multicast operation is one whose delivery ordering semantics are the same as those for delivering views in a view-synchronous group communication system. In a thingumajig service, operations upon thingumajigs are causally ordered. The service supports lists of users able to perform operations on each particular thingumajig. Explain why removing a user from a list should be a sync-ordered operation.

Exercise 14.7

What is the consistency issue raised by state transfer?

Exercise 14.8

An operation \( X \) upon an object \( o \) causes \( o \) to invoke an operation upon another object \( o' \). It is now proposed to replicate \( o \) but not \( o' \). Explain the difficulty that this raises concerning invocations upon \( o' \), and suggest a solution.

Exercise 14.9

Explain the difference between linearizability and sequential consistency, and why the latter is more practical to implement, in general.
Exercise 14.10

Explain why allowing backups to process read operations leads to sequentially consistent rather than linearizable executions in a passive replication system.

page 569

Exercise 14.11

Could the gossip architecture be used for a distributed computer game as described in Exercise 14.3?

page 572

Exercise 14.12

In the gossip architecture, why does a replica manager need to keep both a ‘replica’ timestamp and a ‘value’ timestamp?

page 576

Exercise 14.13

In a gossip system, a front end has vector timestamp (3, 5, 7) representing the data it has received from members of a group of three replica managers. The three replica managers have vector timestamps (5, 2, 8), (4, 5, 6) and (4, 5, 8), respectively. Which replica manager(s) could immediately satisfy a query from the front end and what is the resultant time stamp of the front end? Which could incorporate an update from the front end immediately?

page 577

Exercise 14.14

Explain why making some replica managers read-only may improve the performance of a gossip system.

page 581

Exercise 14.15

Write pseudocode for dependency checks and merge procedures (as used in Bayou) suitable for a simple room-booking application.

page 583
Exercise 14.16

In the Coda file system, why is it sometimes necessary for users to intervene manually in the process of updating the copies of a file at multiple servers? page 590

Exercise 14.17

Devise a scheme for integrating two replicas of a file system directory that underwent separate updates during disconnected operation. Use either Bayou's operational transformation approach, or supply a solution for Coda. page 591

Exercise 14.18

Available copies replication is applied to data items A and B with replicas A₁, A₂, B₁, B₂. The transactions T and U are defined as:

T: Read(A), Write(B, 44). U: Read(B), Write(A, 55).

Show an interleaving of T and U, assuming that two-phase locks are applied to the replicas. Explain why locks alone cannot ensure one copy serializability if one of the replicas fails during the progress of T and U. Explain with reference to this example, how local validation ensures one copy serializability. page 594

Exercise 14.19

Gifford's quorum consensus replication is in use at servers X, Y and Z which all hold replicas of data items A and B. The initial values of all replicas of A and B are 100 and the votes for A and B are 1 at each of X, Y and Z. Also R = W = 2 for both A and B. A client reads the value of A and then writes it to B.

(i) At the time the client performs these operations, a partition separates servers X and Y from server Z. Describe the quora obtained and the operations that take place if the client can access servers X and Y.

(ii) Describe the quora obtained and the operations that take place if the client can access only server Z.

(iii) The partition is repaired and then another partition occurs so that X and Z are separated from Y. Describe the quora obtained and the operations that take place if the client can access servers X and Z. page 599