Distributed Systems: Concepts and Design

Edition 5

By George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair

Chapter 2 Exercise Solutions

2.1 What is the main disadvantage of distributed systems which exploit the infrastructure offered by the Internet? How can this be overcome?

2.1 Ans.

The level of heterogeneity in distributed systems which exploited the infrastructure offered by the Internet was significant in terms of networks, computer architecture, operating systems, languages employed. This led to an increasing emphasis on open standards and associated middleware technologies such as CORBA.

2.2 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.

2.2 Ans.

The client is intrinsically bound to the server and vice versa and this is inflexible in terms of dealing with failure, for example if the server fails and a backup server takes over managing requests. More generally, this level of coupling makes it hard to deal with change.

Clients and servers must exist at the same time and hence it is not possible to operate in more volatile environments when either party may be unavailable, for example disconnected in the case of a mobile node.

The benefits of space uncoupling is in providing more degrees of freedom in dealing with change, for example if a new server starts dealing with requests.

The benefit of time uncoupling is in allowing entities to communicate when entities may come and go.

2.3 What is the range of techniques covered by remote invocation? Briefly explain each technique.

2.3 Ans.

Request-reply protocols: Request-reply protocols are effectively a pattern imposed on an underlying message passing service to support client-server computing.

Remote procedure calls: In RPC, procedures in processes on remote computers can be called as if they are procedures in the local address space.

Remote method invocation: Remote method invocation (RMI) strongly resembles remote procedure calls but in a world of distributed objects.

2.4 How are entities, such as objects or services, mapped on to the underlying physical distributed infrastructure?

2.4 Ans.

Objects or services map on to the underlying physical distributed infrastructure using placement strategies. Placement is crucial in terms of determining the properties of the distributed system, most obviously related to performance but also to other aspects such as reliability and security.

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?

2.5 Ans.

The crawler tasks could build partial indexes to new pages incrementally, then merge them with the active index (including deleting invalid references). This merging operation could be done on an off-line copy. Finally, the environment for processing client requests is changed to access the new index. The latter might need some concurrency control, but in principle it is just a change to one reference to the index which should be atomic.

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?

2.6 Ans.

Problems:

- people often turn their desktop computers off when not using them. Even if on most of the time, they will be off when user is away for an extended time or the computer is being moved.
- the owners of participating computers are unlikely to be known to other participants, so their trustworthiness is unknown. With current hardware and operating systems the owner of a computer has total control over the data on it and may change it or delete it at will.
- network connections to the peer computers are exposed to attack (including denial of service).

The importance of these problems depends on the application. For the music downloading that was the original driving force for peer-to-peer it isn't very important. Users can wait until the relevant host is running to access a particular piece of music. There is little motivation for users to tamper with the music. But for more conventional applications such as file storage availability and integrity are all-important.

Solutions:

Replication:

- if data replicas are sufficiently widespread and numerous, the probability that all are unavailable simultaneously can be reduced the a negligible level.
- one method for ensuring the integrity of data objects stored at multiple hosts (against tampering or accidental error) is to perform an algorithm to establish a consensus about the value of the data (e.g. by exchanging hashes of the object's value and comparing them). This is discussed in Chapter 15. But there is a simpler solution for objects whose value doesn't change (e.g. media files such as music, photographs, radio broadcasts or films).

Secure hash identifiers:

- The object's identifier is derived from its hash code. The identifier is used to address the object. When the object is received by a client, the hash code can be checked for correspondence with the identifier. The hash algorithms used must obey the properties required of a secure hash algorithm as described in Chapter 7.

2.7 How is caching useful in placement strategies? What are its disadvantages?

2.7 Ans.

A *cache* is a store of recently used data objects that is closer to one client or a particular set of clients than the objects themselves. A significant disadvantage is that caches are expensive.

2.8 What is a mobile agent? How can it be a potential security threat?

2.8 Ans.

A mobile agent is a running program (including both code and data) that travels from one computer to another in a network carrying out a task. Mobile agents can themselves be vulnerable – they may not be able to complete their task if they are refused access to the information they need.

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.

2.9 Ans.

A three-tier solution might consist of:

- a web-based front-end offering a user interface for the car hire service (the presentation logic);
- a middle tier supporting the core operations associated with the car hire business including locating a particular make and model, checking availability and pricing, getting a quote and purchasing a particular car (the application logic);
- a database which stores all the persistent data associated with the stock (the data logic).

In terms of performance, this approach introduces extra latency in that requests must go from the web-based interface to the middle tier and then to the database (and back). However, processing load is also spread over three machines (especially over the middle tier and the database) and this may help with performance. For this latter reason, the three-tier solution may scale better. This may be enhanced though by other, complementary placement strategies including replication.

In terms of failure, there is an extra element involved and this increases the probability of a failure occurring in the system. Equally, failures are more difficult to deal with, for example if the middle tier is available and the database fails.

The three-tier approach is much better for evolution because of the intrinsic separation of concerns. For example, the middle tier only contains application logic and this should therefore be easier to update and maintain.

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).

Saltzer's end-to-end argument states that communication-related functions can only be completely and reliably implemented with the knowledge and help of the application and therefore providing that function as a feature of the communication system itself (or middleware) is not always sensible. One concrete example is in secure communication. Assume that in a given system, the communication subsystem provides encrypted communication. This is helpful but insufficient. For example, the pathway from the network to the application software may be compromised and this is unprotected. This solution also does not deal with malicious participants in the exchange of data.

Consider also the reliable exchange of data implemented by introducing checksum protection on individual hops in the network. Again, this is insufficient as data may be corrupted by intermediary nodes, for example, gateways, or indeed in the end systems.

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?

2.11 Ans.

The rate of arrival of client requests is unpredictable.

If the server uses threads to execute the requests concurrently, it may not be able to allocate sufficient time to a particular request within any given time limit.

If the server queues the request and carries them out one at a time, they may wait in the queue for an unlimited amount of time.

To execute requests within bounded time, limit the number of clients to suit its capacity. To deal with more clients, use a server with more processors. After that, (or instead) replicate the service....

The solution may be costly and in some cases keeping the replicas consistent may take up useful processing cycles, reducing those available for executing requests.

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?

2.12 Ans.

Time taken by OS communication services in the sending and receiving processes - these tasks would need to be guaranteed sufficient processor cycles.

Time taken to access network. The pair of communicating processes would need to be given guaranteed network capacity.

The time to transmit the data is a constant once the network has been accessed.

To provide the above guarantees we would need more resources and associated costs. The guarantees associated with accessing the network can for example be provided with ATM networks, but they are expensive for use as LANs.

To give guarantees for the processes is more complex. For example, for a server to guarantee to receive and send messages within a time limit would mean limiting the number of clients.

2.13 What are the two variants of the interaction model in distributed systems? On what points do they differ?

2.13 Ans.

Synchronous distributed systems: Makes assumption of time.

Asynchronous distributed systems: Makes no assumption of time.

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 order and 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?

2.14 Ans.

Service A can have:

arbitrary failures:

– as checksums do not apply to message bodies, message bodies can corrupted.

– duplicated messages,

omission failures (lost messages).

Because the distributed system in which it is used is asynchronous, it cannot suffer from timing failures.
Validity - is denied by lost messages
Integrity - is denied by corrupted messages and duplicated messages.

Service B can have:

omission failures (lost messages, dropped messages).

Because the distributed system in which it is used is asynchronous, it cannot suffer from timing failures.

It passes the integrity test, but not the validity test, therefore it cannot be called reliable.

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 (that carries out the request) followed by a reply message from Y to X. Describe the classes of failure that may be exhibited by an invocation.

2.15 Ans.

An invocation may suffer from the following failures:

- *crash failures*: X or Y may crash. Therefore an invocation may suffer from crash failures.
- *omission failures*: as SB suffers from omission failures the request or reply message may be lost.

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.

2.16 Ans.

The basic disk read exhibit arbitrary failures.

This can be masked by using a checksum on each disk block (making it unlikely that wrong values will go undetected) - when an incorrect value is detected, the read returns no value instead of a wrong value - an omission failure.

The omission failures can be masked by replicating each disk block on two independent disks. (Making omission failures unlikely).

2.17 How can the security of a distributed system be achieved? How can processes and their interactions be secured?

2.17 Ans.

The security of a distributed system can be achieved by securing the processes and the channels used for their interactions and by protecting the objects that they encapsulate against unauthorized access. Objects are intended to be used in different ways by different users-access rights specify who is allowed to perform the operations of an object. With the help of cryptography process interaction can be secured.

2.18 Cryptography is the science of keeping messages secure. Explain, with an example, how it can be used in authentication to maintain confidentiality.

2.18 Ans.

Cryptography is the science of keeping messages secure, and *encryption* is the process of scrambling a message in such a way as to hide its contents.

The basic authentication technique is to include in a message an encrypted portion that contains enough of the contents of the message to guarantee its authenticity.

