Distributed Systems: Concepts and Design

Chapter 9 Exercise Solutions

9.1 Compare the request reply protocol as described in Section 4.4 with the implementation of client-server communication in SOAP. State two reasons why the use of asynchronous messages by SOAP is more appropriate for use over the Internet. To what extent does the use of HTTP by SOAP reduce the difference between the two approaches?

9.1 Ans.

The request reply communication protocol proposes an infrastructure consisting of a synchronous request-reply exchange, whereas SOAP specifies the use of a pair of asynchronous messages. In addition, the request reply protocol passes the remote object reference of an object to be invoked, whereas SOAP does not support remote objects. In the request reply protocol the method to be invoked is part of the communication protocol, whereas in SOAP it is specified in the message body.

Use over the internet:

i) SOAP uses asynchronous messages in order to reduce the closer coupling between client and server which occurs with a synchronous protocol.

ii) In addition, SOAP allows the transport of single documents.

The use of HTTP enables the request and reply messages to be related to one another in that the HTTP response carries the reply to a SOAP request. In addition, the method name may be specified as an action header in HTTP.

9.2 Compare the structure of URLs as used for web services with that of remote object references as specified in Section 4.3.4. State in each case how they are used to get a client request executed.

9.2 Ans.

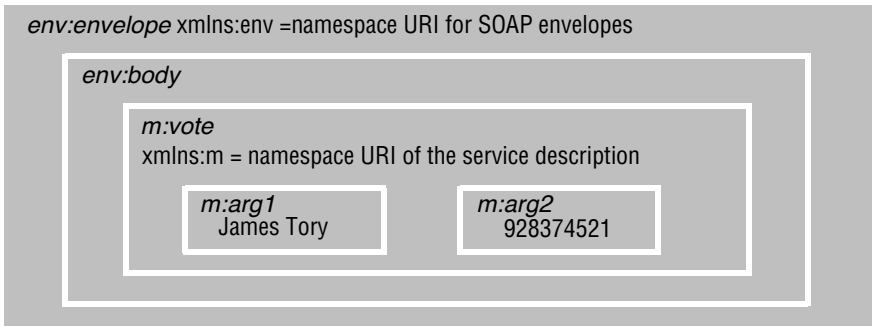
A URL specifies the name of a server as a DNS name (and an optional port) and the path name of a resource managed by that server. For a web service, the resource in the URL refers to an action to be performed by a web service.

A remote object reference contains the internet address and port of a server, time, object number and type information about the interface of a remote object.

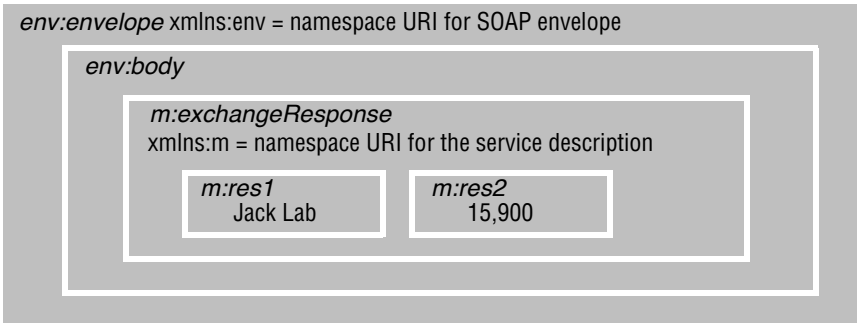
In the web services case, DNS is used to convert the DNS name into the internet address of a server. The client request is sent to that location where the server dispatcher selects an operation to be performed.

9.3 Illustrate the contents of a SOAP Request message and corresponding Reply message in the Election service example of Exercise 5.1, using the pictorial version of XML as shown in Figure 9.4 and Figure 9.5.

The question as written in the book does not make sense. Please replace as above.



9.3 Ans.



9.4 Outline the five main elements of a WSDL service description. In the case of the *Election* service defined in Question 5.1, state the type of information to be used by the request and reply messages – does any of this need to be included in the target namespace? For the *vote* operation, draw diagrams similar to Figure 19.11 and Figure 19.13.

9.4 Ans.

The five main components of WSDL are types, messages, interface, bindings and services. (outline..)

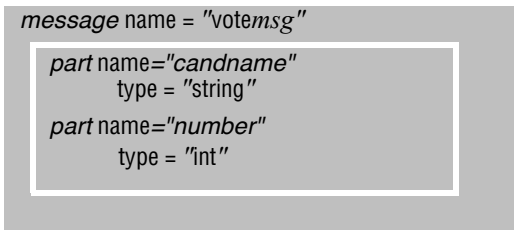
Types for the Election service:

vote: input parameters *string* and *integer*

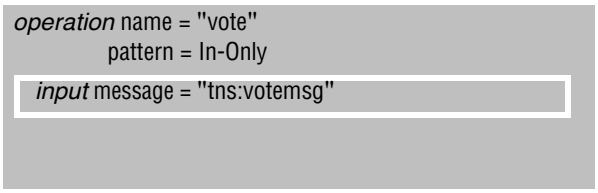
result: output parameters *string* and *integer*.

String and integer are XML defined types, so no need to define them for this application.

First define a message with two parts, in a diagram similar to 19.11:



Then for the vote operation, draw a diagram similar to Fig 19.13 name = vote, pattern = Robust In-only or In-only



9.5 Continuing with the example of the *Election* service, explain why the part of the WSDL defined in Question 9.4 is referred to as 'abstract'. What would need to be added to the service description to make it completely concrete?

9.5 Ans.

The part defined so far is abstract in the sense that no details have been given i) as to what particular communication or transport protocols are to be used.

ii) as to the location of the server

To make it concrete, WSDL has to specify the protocols in use, for example SOAP with HTTP. This can be done in a WSDL binding element. It also has to specify the address of the server, for example, in a WSDL service element.

9.6 Define a Java interface for the *Election* service suitable for use as a web service. State why you think the interface you defined is suitable. Explain how a WSDL document for the service is generated and how it is made available to clients.

9.6 Ans.

The interface to the service is defined as a Java class.

We need to define a class for the result e.g.

```
class Result {
        String name;
        int votes;
}
```

The interface is:

```
import java.rmi.*;
public interface Election extends Remote{
    void vote(String name, int number) throws RemoteException;
    Result result () throws RemoteException;
};
```

This is suitable because it does not pass values of *Remote* as arguments or results.

The tool *wscompile* is used to generate a WSDL document describing the service from the above interface. When the service is deployed, this document is installed in the servlet container from whence clients may get it.

9.7 Describe the contents of a Java client proxy for the *Election* service. Explain how the appropriate marshalling and unmarshalling methods can be obtained for a static proxy.

9.7 Ans.

The proxy must contain the URL of the server for the *Election* service. It contains one method for *vote* and another for *result*. Each of these methods marshalls its own method name and its arguments into a SOAP request envelope. When the reply arrives it is unmarshalled by extracting the results from the XML envelope.

The client obtains the WSDL document containing the service description and uses *wscompile* to generate the methods in the proxy.

9.8 Explain the role of a servlet container in the deploying of a web service and the execution of a client request.

9.8 Ans.

When the service is created, a tool called *wsdeploy* creates the skeleton (or tie) classes from the service implementation. When the service is deployed in a servlet container, it is supplied with the servlet name and WSDL description, its class and skeleton classes. The WSDL contains the URL of the end point.

Client SOAP request messages are addressed to the URL of the service where a dispatcher uses the name of the method in the *Action* header to locate the appropriate skeleton.

The skeleton is given a SOAP request from which the arguments are extracted, these are used to call the corresponding method in the servlet. When the result is returned, it translates these back into a SOAP reply message.

9.9 In the Java example illustrated in Figure 9.8 and Figure 9.9, both client and server are dealing with objects, although web services do not support distributed objects. How can this be the case? What are the limitations imposed on the interfaces of Java web services?

9.9 Ans.

In this example, the objects exchanged by client and server are of class *GraphicalObject*. But the SOAP requests and replies deal with an XML representation of a structure with the same fields, but no methods.

Limitations - no passing of remote references and no factory methods.

9.10 Outline the replication scheme used in UDDI. Supposing that vector timestamps are used to support this scheme, define a pair of operations for use by registries needing to exchange data.

9.10 Ans.

The members of a registry propagate copies of data structures to one another as follows: a server that has made changes notifies the other servers in the registry, which then request the changes.

1. all changes to a particular data structure are made at the same server;
2. updates from a particular server are received in sequential order by the other members, but no particular ordering is imposed between update operations made by different servers.

call the vector timestamp V, then the operations are

notify(Id of registry, V)

a UDDI server uses this operation to inform another that it has new updates to data structures. The recipient compares the vector V with its own vector timestamp

get(V)-> sequences of updates

a UDDI server uses this operation to requests another server to send it updates it needs. The argument is that server's vector timestamp. The recipient uses V to determine which updates the requester requires before returning them.

9.11 Explain why UDDI can be described as being both a name service and a directory service, mentioning the types of enquiries that can be made. The second 'D' in the name UDDI refers to 'discovery' – is UDDI really a discovery service?

9.11 Ans.

a white pages (name) service - it holds lists of organisations enabling clients to look up a service against an organisation;

a yellow pages (directory) service - it allows clients to look up services by category or by interface;

information can also be looked up by a key which is a sort of URI.

A discovery service is a directory service that is designed to register the services provided in a spontaneous networking environment. It allows the set of clients and services to change dynamically without user intervention. UDDI does not do those things.

9.12 Outline the main difference between TLS and XML security. Explain why XML is particularly suitable for the role it plays, in terms of these differences.

9.12 Ans.

The main difference between TLS and XML security is that the former provides a means for the secure transmission of information, whereas XML security applies the security measures to documents to be exchanged and stored.

XML is a self-defining format enabling security to be applied within a document. It is extensible, allowing security measures to be added as requirements evolve.

9.13 Documents protected by XML security may be signed or encrypted long before anyone can predict who will be the ultimate recipients. What measures are taken to ensure that the latter have access to the algorithms used by the former?

9.13 Ans.

The algorithms used are specified within the XML document itself, so that recipients know which one was used for a particular task, e.g encryption or digital signing. In addition, the standard states the algorithms that must be available to any implementation of XML encryption and XML digital signature. The creator of such documents should be sure to use one of the standard algorithms.

9.14 Explain the relevance of canonical XML to digital signatures. What contextual information can be included in the canonical form? Give an example of a breach of security where the context is omitted from canonical form

9.14 Ans.

Digital signatures are used to ensure that the information content has not changed since the document was digitally signed. The signature is initially generated using a digest calculation based on the canonical form of the document, which eliminates insignificant differences. After its transmission, the receiving application applies the same algorithm to the document, producing another digest of the canonical form, and the two digests are compared. If they differ, it means the canonical forms differ, so the document must have been altered since it was signed.

The contextual information consists of all of the namespaces declared and the values of attributes that surround a particular XML element. The inclusion of context in canonical XML ensures that the particular element cannot be used out of context - or in a different context.

If the context is omitted, the canonicalised element can be placed in a different context. For example, the value of an attribute in the context may be crucial to the document being signed. It might for example, be an exchange rate.

9.15 A coordination protocol could be carried out in order to coordinate the actions of web services. Outline an architecture for (i) a centralized and (ii) a distributed coordination protocol. In each case, describe the interactions needed to establish coordination between a pair of web services.

9.15 Ans.

Centralised coordination. A particular server would play the role of coordinator and the web services that require coordination would send their coordination messages to the coordinator, rather like the protocol used in the 2PC and illustrated in Figure 13.3. Call the web services A and B, and the coordinator C.

A asks C to create a new context for coordination, which is returned;

A sends B a message, passing on an identifier for the coordination and B communicates with C about it.

In distributed coordination, each participating web service would have a local coordinator that it communicates with and the set of local coordinators would communicate amongst them selves. Call the web services A and B, and their local coordinators Ca and Cb.

A asks Ca to create a new context for coordination, which is returned;

A sends B a message, passing on an identifier for the coordination and B asks Cb to create a context for it.

9.16 Compare RPC call semantics with the semantics of *WS-ReliableMessaging*:

i) State the entities to which each refers.

ii) Compare the differing meanings of the available semantics (for example, *at-least-once*, *at-most-once*, *exactly-once*).

9.16 Ans.

In *WS-ReliableMessaging*, there is a message exchange between and application source and an application destination and the goal is to provide a given semantics over this message delivery. The semantics are as follows:

At-least-once: The message is delivered at least once (with possible duplicate messages), but an error is reported if it cannot be delivered.

At-most-once: The message is delivered at most once (with no duplicate messages, but without any error report if it cannot be delivered.

Exactly-once: The message is delivered exactly once, but an error is reported if it cannot be delivered.

In RPC call semantics, there is a two way interaction between the client and the server with the client sending a request, the server executing the (remote) procedure and the server sending a reply back to the client. The semantics are defined over this complete path and not just the delivery of the initial message (the request):

At-least-once: The client receives a result, in which case the requested procedure is executed at least once (with possible duplicate executions), or receives an error, with no result obtained.

At-most-once: The client receives a result, in which case the requested procedure is executed at most once (no duplicates, but possible failure in executing the procedure), or receives an error, with no result obtained.

Exactly-once: The client receives a result, in which case the requested procedure is executed exactly once (no duplicates, and no failure part way through), or receives an error, with no result obtained.