


Today's Topics



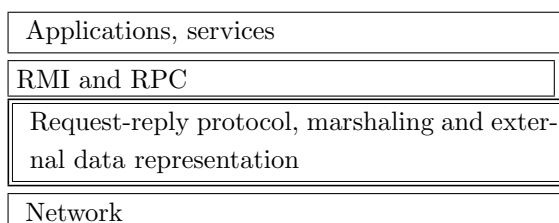
Slide 1

- External Data Representation and Marshalling (Sec. 4.3)
- Request reply protocol (failure modes) (Sec. 4.4)
- Distributed Objects and Remote Invocations (5.1)
 - Communication between Distributed Objects (5.2)
 - Remote Procedure calls (5.2)
 - Events and Notifications. (5.3)

Middleware Layers



Slide 2



We will look at the request-reply layer and data representation layer.

External Data Representation and Marshaling

Slide 3

Remember the problem in lecture one,

- **Heterogeneity**. (Everybody is different).

Different operation systems, different programming languages,
different hardware architectures.

Example, Big Endian/Little Endian

Most numbers that you store are larger than a byte and there is some choice about how you lay out multibyte numbers in memory.

Slide 4

- **Big Endian** in big endian you store the most significant byte in the smallest memory address. (For example DECs)
- **Little Endian** in little endian you store the least significant byte in the smallest memory address. (For example Suns, Motorolas).
- For example to store the number **0x90AB**

Address	Big Endin Value	Little Endin Value
1000	90	AB
1001	AB	90

What to do?

One of the two following methods can be used:

Slide 5

- The values can be converted to an agreed external format before transmission and converted to the local form on receipt.
- The values are transmitted in the sender's format together with an indication of the format used, and the recipient converts the values if necessary.

An agreed standard for the representation of data structures and primitive values is called an *external data representation*

Marshaling is the process of taking a collection of data items and assembling them into a form suitable for transmission.

Two Examples

We will briefly discuss two alternative approaches to external data representation and Marshalling.

Slide 6

- CORBA's common data representation, which is concerned with an external representation of data that can be passed as arguments and results of remote invocations.
- Java's object serialisation which flattens objects or tree of objects that need to be transmitted.

In both cases, the Marshalling and unmarshalling activities are intended to be carried out by a middleware layer without any involvement on the part of the application programmer.

CORBA

Slide 7

- There is a design program to be solved, if middleware is supposed support many different systems and you have to convert the data then how do you specify what is going on in a modular way?
- Corba has a common external form CDR (Common Data Representation).
- The interface of objects are described in CORBA IDL which is then compiled by the CORBA interface compiler and the marshaling/unmarshaling operations are generated automatically.
- For each new platform a CORBA IDL compiler has to be provided....

Java Object Serialization

Slide 8

- In Java RMI, both objects and primitive data values may be passed as arguments and results of method invocations.
- If an object implements the interface `Serializable` then the system provides a default serialisation method which the programmer can override if it not sufficient. The contents of all the data members of the class is serialised.

Request Reply (Acknowledge) Protocol

Slide 9

- Nothing deep, common format for protocols. Send the request to the remote server/object what ever ... wait for the reply of acknowledgement.
- What you have to think about are the failure modes.
- Depending on the type of connection you use messages could be:
 - Lost;
 - might not be received in the same order that they were sent.

Request reply, failure modes

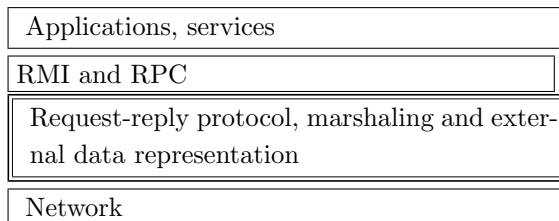
Slide 10

- If messages can be lost the reply message could be lost. You don't know if the operation has been carried out.
- If possible the client could resend the request message until a reply is received.
- But if you don't want to send the same message twice then there is nothing much you can do.
- The problem is that you can't distinguish between a lost and a late message.

Distributed Objects and Remote Invocation

Middleware layers:

Slide 11



Distributed Objects and Remote Invocation

We will look at three things:

Slide 12

- The earliest example, remote procedure calls which allows procedures to be executed on a remote host.
- Remote Method invocation (RMI) like procedures just objects and methods instead of procedures.
- Event based programming model, allows objects to receive notification of the events at other objects in which they have registered interest.

Middleware

- Slide 13**
- Remember, one of the goals of middleware is to make things as invisible to the application.
 - Location transparency. One of the design goals of a distributed system is that the application programmer should not need to be aware of where resources are located.

Remote Procedure Calls

- Remote procedure calls can be made invisible to the programmer by procedure stubs.

Slide 14

```
do_something(int a, char* b);

void do_something(int x, char* str) {
    construct_message;
    send_message_over_net;
    wait_and_unpack_reply;
    return_answer;
}
```

Remote Procedure Calls

Then a server would run code something like:

Slide 15

```
main() {
    wait_for_do_something_message_and_unpack;
    do_something(. . . .);
    return_any_results_from_do_something;
}
```

The server calls the real `do_something` while the client is just running a stub.

Of course the server and client stubs could be generated automatically. For example the CORBA IDL compiler.

Remote Object Call

- Slide 16
- Only real difference between remote procedure calls is that the server has to maintain the object state.
 - Again we can use interface stubs to make this invisible to the application

```
class silly {
    int do_something(. . . . ) {
        send_message_and_wait_for_reply;
        unpack_message_and_return_results;
    }
}
```

Again an IDL compiler could generate the stubs for you.

Problems with local references

- Slide 17**
- If you pass a parameter which is a reference (pointer) to a local piece of data, then what do you do? Possible solutions include:
 - Don't pass pointers!
 - Copy the local data and send it over the network.
 - Modify the server so that it requests the data on demand.

Events and Notifications

- Slide 18**
- **Publish, subscribe** paradigm. Objects publish a list of types of events that it might generate.
 - Other object subscribe to certain events. When the objects generate events they notify the objects.
 - Often used in GUI toolkits, for example a button on the screen generates an event that could be monitored by different objects.