

CS 5523 Lecture 20:
Remote Invocation and RPCs

- *Design issues for remote calls and invocations*
- *Strategies for providing delivery guarantees*
- *Invocation semantics*
- *Overview of RPC*
- *XDR format*
- *Multicast and broadcast*

Design issues for remote calls and invocation:

- *What are invocation semantics? (Local calls are invoked exactly once. Under what circumstances can this fail to happen for remote calls?)*
- *Transparency (Local calls are made to in environment of the calling process. How is the choice of environment handled for remote calls?)*

Strategies for providing delivery guarantees:

- *Retry the request message*
- *Filter duplicate messages at the server*
- *Keep a history of results so that if duplicate requests are received the results can be retransmitted*

Types of invocation semantics:

- *Exactly once semantics* – every method is executed exactly once
- *Maybe semantics* – caller can not determine whether or not the remote method has been executed
- *At-least-once semantics* – caller either receives a result (in which case the user knows the method was executed at least once) or an exception
- *At-most-once semantics* - caller either receives a result (in which case the user knows the method was executed at exactly once) or an exception

Figure 5.5
Invocation semantics

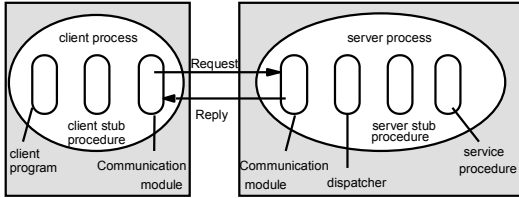
Fault tolerance measures			Invocation semantics
Retransmit request message	Duplicate filtering	Re-execute procedure or retransmit reply	
No	Not applicable	Not applicable	Maybe
Yes	No	Re-execute procedure	At-least-once
Yes	Yes	Retransmit reply	At-most-once

Introspect's Guide for Condemi, Dellmann and Krollberg: Distributed Systems: Concepts and Design, Edn 3
© Addison-Wesley Publishers 2000

Service interface (RPC):

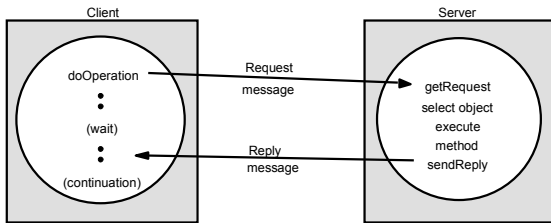
- A server provides a set of procedures available to client
- These procedures are specified by a service interface
- Input and output parameters are specified
- Use:
 - ! When the remote procedure is invoked, the values of arguments corresponding to the input parameters are converted to a standard external representation and copied into a packet (marshaling).
 - ! The client sends the marshaled packet to the server.
 - ! The server demarshals the packet, performs the procedure, marshals the return packet, and sends the marshaled return packet to the client.
 - ! Client demarshals the return.
 - ! The entire procedure is concealed in the call.

Figure 5.7
Role of client and server stub procedures in RPC



Instructor's Guide for Conrad, Dillman and Krollberg Distributed Systems: Concepts and Design Edn 3
© Addison-Wesley Publishers 2000

Figure 4.11
Request-reply communication



Instructor's Guide for Conrad, Dillman and Krollberg Distributed Systems: Concepts and Design Edn 3
© Addison-Wesley Publishers 2000

Figure 4.12
Operations of the request-reply protocol

```
public byte[] doOperation (RemoteObjectRef o, int methodId, byte[] arguments)
    sends a request message to the remote object and returns the reply.
    The arguments specify the remote object, the method to be invoked and the arguments of
    that method.
public byte[] getRequest ();
    acquires a client request via the server port.
public void sendReply (byte[] reply, InetAddress clientHost, int clientPort);
    sends the reply message reply to the client at its Internet address and port.
```

Instructor's Guide for Conrad, Dillman and Krollberg Distributed Systems: Concepts and Design Edn 3
© Addison-Wesley Publishers 2000

Figure 4.13
Request-reply message structure

messageType	<i>int (0=Request, 1= Reply)</i>
requestId	<i>int</i>
objectReference	<i>RemoteObjectRef</i>
methodId	<i>int or Method</i>
arguments	<i>array of bytes</i>

Instructor's Guide for Conradin, Dollmann and Krollberg Distributed Systems: Concepts and Design Edn 3
© Addison-Wesley Publishers 2000

Figure 4.14
RPC exchange protocols

Name	Messages sent by		
	Client	Server	Client
R	Request		
RR	Request	Reply	
RRA	Request	Reply	Acknowledge reply

Instructor's Guide for Conradin, Dollmann and Krollberg Distributed Systems: Concepts and Design Edn 3
© Addison-Wesley Publishers 2000

Figure 4.15
HTTP request message

method	URL or pathname	HTTP version	headers	message body
GET	//www.dcs.qmw.ac.uk/index.html	HTTP/ 1.1		

Instructor's Guide for Conradin, Dollmann and Krollberg Distributed Systems: Concepts and Design Edn 3
© Addison-Wesley Publishers 2000

Figure 4.16
HTTP reply message

<i>HTTP version</i>	<i>status code</i>	<i>reason</i>	<i>headers</i>	<i>message body</i>
HTTP/1.1	200	OK		resource data

Instructor's Guide for Conrad, Dollimore and Krollberg Distributed Systems: Concepts and Design, Edn 3
© Addison-Wesley Publishers 2000

Idempotent operations:

- *An idempotent information can be performed repeatedly with same effect.*
- *The HTTP 1.0 GET is idempotent*
- *Idempotent operations don't maintain state on the server*

How would you make a file server based on idempotent operations?

RPC based on TCP or UDP:

RPC can be based on TCP or UDP – what are the design issues with respect to invocation semantics?

Example: Sun RPC:

- RFC 1831
- Used in the Sun NFS network file system
- Sometimes called Open Network Computing RPC (ONC RPC)
- Can use either UDP or TCP or broadcast UDP.
- Uses XDR as an interface definition language
- Only single input and output parameters are allowed
- Sun RPC runs a local binding services called a port mapper on each host

Figure 5.8
Files interface in Sun XDR

```
const MAX = 1000;
typedef int FileIdentifier;
typedef int FilePointer;
typedef int Length;
struct Data {
    int length;
    char buffer[MAX];
};
struct writeargs {
    FileIdentifier f;
    FilePointer position;
    Data data;
};
struct readargs {
    FileIdentifier f;
    FilePointer position;
    Length length;
};
program FILEREADWRITE {
    version VERSION {
        void WRITE(writeargs)=1; 1
        Data READ(readargs)=2; 2
    }=2;
} = 9999;
```

Interne's Guide for Cookson, Dillman and Kniberg Distributed Systems: Concepts and Design, Edn 3
© Addison-Wesley Publishers 2000

Multicast communication:

- Multicast – sends a single message from one process to members of a group of processes (hosts)
- Broadcast – sends a single message from one process to all processes (hosts)

Why is the word hosts in parentheses?

What issues have to be addressed when using multicast?

Uses of multicast:

- *Fault tolerance based on replicated services*
- *Discovery in spontaneous networking*
- *Managing replicated data*
- *Propagation of event notifications in a distributed environment*

Figure 4.17
Multicast peer joins a group and sends and receives datagrams

```
import java.net.*;
import java.io.*;
public class MulticastPeer{
    public static void main(String args[]){
        // args give message contents & destination multicast group (e.g. "228.5.6.7")
        try {
            InetAddress group = InetAddress.getByName(args[1]);
            MulticastSocket s = new MulticastSocket(6789);
            s.joinGroup(group);
            byte [] m = args[0].getBytes();
            DatagramPacket messageOut =
                new DatagramPacket(m, m.length, group, 6789);
            s.send(messageOut);
```

// this figure continued on the next slide

Instructor's Guide for Cookson, Dillman and Krollberg Distributed Systems: Concepts and Design, Edn. 3
© Addison-Wesley Publishers 2000

Figure 4.17
continued

```
        // get messages from others in group
        byte[] buffer = new byte[1000];
        for(int i=0; i<3; i++){
            DatagramPacket messageIn =
                new DatagramPacket(buffer, buffer.length);
            s.receive(messageIn);
            System.out.println("Received:" + new String(messageIn.getData()));
        }
        s.leaveGroup(group);
    }catch (SocketException e){System.out.println("Socket: " + e.getMessage());}
    }catch (IOException e){System.out.println("IO: " + e.getMessage());}
}
}
```

Instructor's Guide for Cookson, Dillman and Krollberg Distributed Systems: Concepts and Design, Edn. 3
© Addison-Wesley Publishers 2000

For next time:



Read CDK Section 4.3.1

Read CDK Chapter 17
