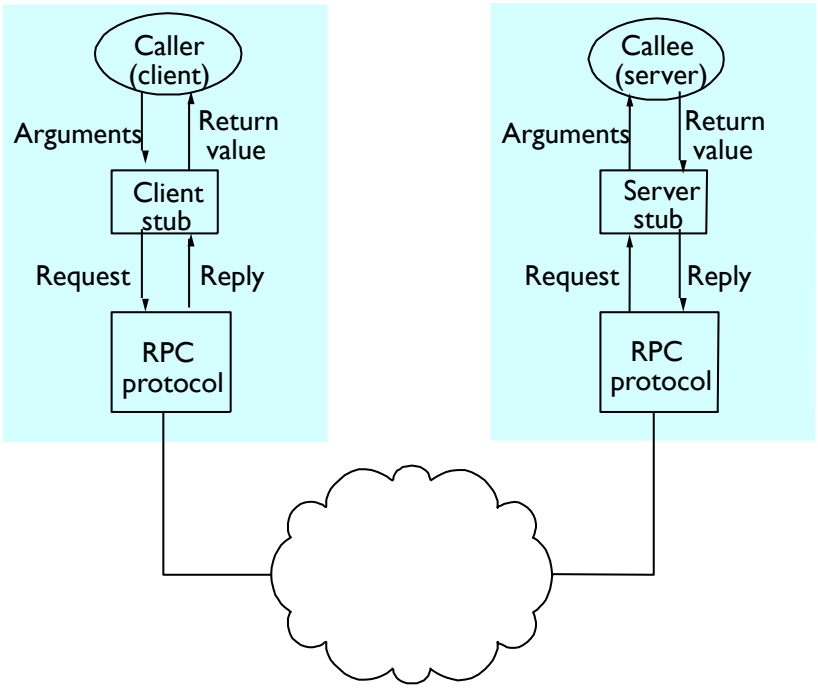
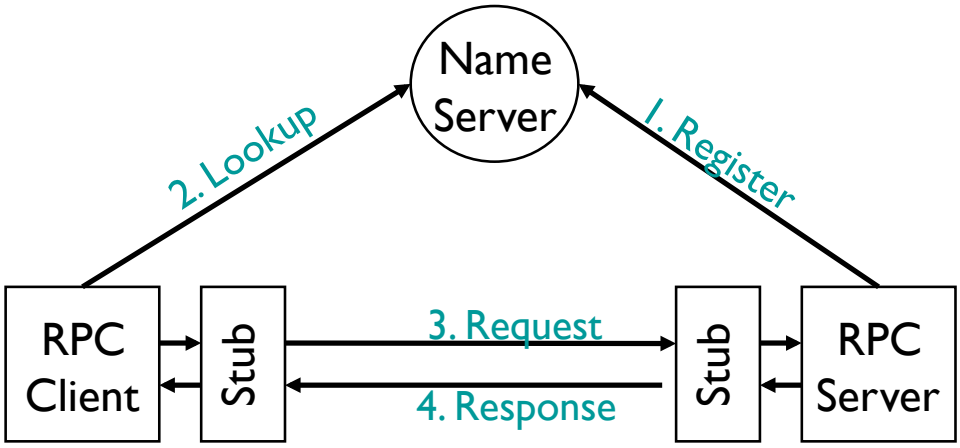
Remote Procedure Calls (RPCs)

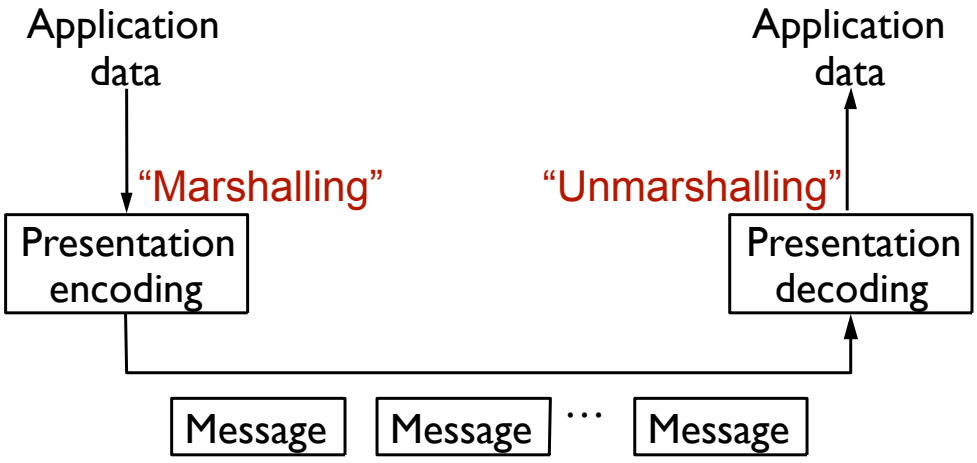
RPC Architecture Overview



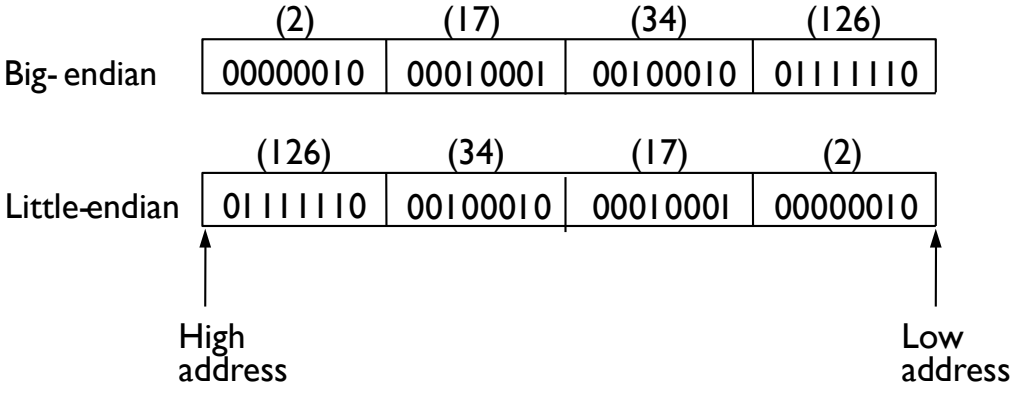
RPC Name Servers



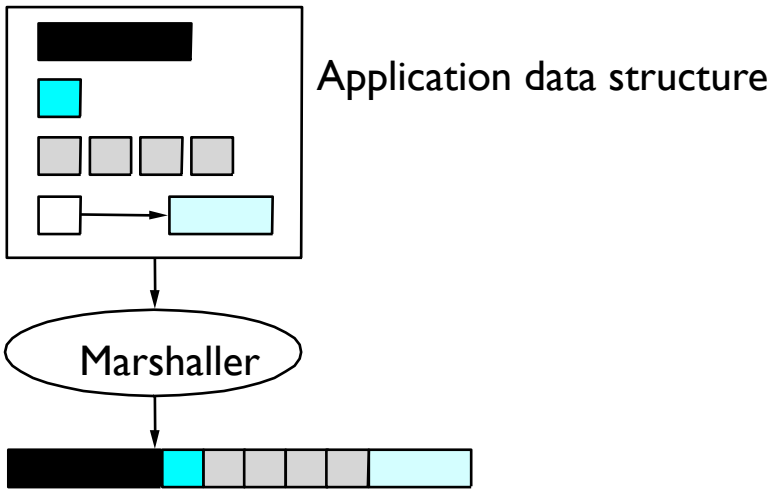
Stub Encoding/Decoding



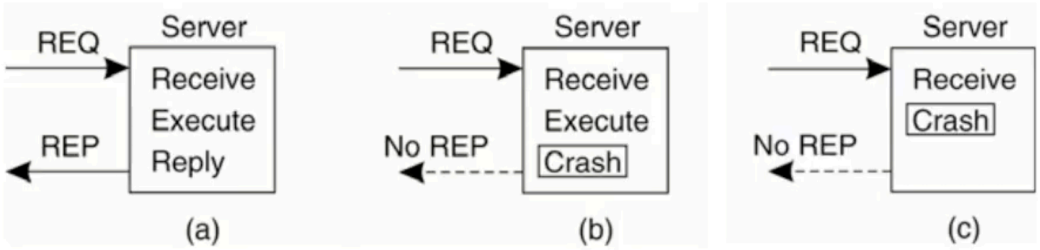
System Encodings



Converting Structures



Failures and RPC Semantics



Normal

Crash after execution

Crash before execution

XML-RPC

```
public Integer[] SumAndDifference(int x, int y) {
    Integer[] array = new Integer[2];
    array[0] = new Integer(x+y);
    array[1] = new Integer(y-x);
    return array;
}
```

XML-RPC Request

```
POST /RPC2 HTTP/1.1
Content-Type: text/xml
User-Agent: XML-RPC.NET
Content-Length: 278
Expect: 100-continue
Connection: Keep-Alive
Host: localhost:8080
<?xml version="1.0"?>
<methodCall>
<methodName>SumAndDifference</methodName>
<params>
<param><value><i4>40</i4></value></param>
<param><value><i4>10</i4></value></param>
</params>
</methodCall>
```


XML-RPC Response

```
HTTP/1.1 200 OK
Date: Wed, 5 Mar 2014 21:52:34 GMT
Server: Microsoft-IIS/6.0
Content-Type: text/xml
Content-Length: 467
<?xml version="1.0"?>
<methodResponse>
<params><param>
<value><struct>
<member><name>sum</name><value><i4>50</i4></value></member>
<member><name>diff</name><value><i4>30</i4></value></member>
</struct></value>
</param></params>
</methodResponse>
```

XML-RPC Data Type Examples

```
<array>
   <data>
      <value><i4>12</i4></value>
      <value><string>Egypt</string></value>
      <value><boolean>0</boolean></value>
      <value><i4>-31</i4></value>
   </data>
</array>
```


XML-RPC Server Example (Java)

```
import org.apache.xmlrpc.webserver.WebServer;
import org.apache.xmlrpc.server.XmlRpcServer;
import org.apache.xmlrpc.server.PropertyHandlerMapping;
import org.apache.xmlrpc.XmlRpcException;

public class Server {
   public Integer[] SumAndDifference(int x, int y) {
      Integer[] array = new Integer[2];
      array[0] = new Integer(x+y);
      array[1] = new Integer(y-x);
      return array;
   }
   public static void main (String [] args) {
      try {
         PropertyHandlerMapping phm = new PropertyHandlerMapping();
         XmlRpcServer xmlRpcServer;
        WebServer server = new WebServer(8888);
         xmlRpcServer = server.getXmlRpcServer();
         phm.addHandler("sample", Server.class);   // registering with name server
         xmlRpcServer.setHandlerMapping(phm);
         server.start();
      } catch (Exception exception) {  System.err.println("Server: " + exception);   }
   }
}
```

XML-RPC Client Example (Java)

```
import java.util.*;
import java.net.URL;
import org.apache.xmlrpc.*;
import org.apache.xmlrpc.client.XmlRpcClient;
import org.apache.xmlrpc.client.XmlRpcClientConfigImpl;

public class Client {
   public static void main (String [] args) {
      XmlRpcClientConfigImpl config = new XmlRpcClientConfigImpl();
      XmlRpcClient client=null;
      try {
         config.setServerURL(new URL("http://" + args[0] + ":" + 8888));
         client = new XmlRpcClient();
         client.setConfig(config);
      } catch (Exception e) { System.err.println("Problem! "'+ e); }

      Vector<Integer> params = new Vector<Integer>();
      params.addElement(new Integer(10));
      params.addElement(new Integer(40));

      try {
         Object[] result = (Object[])client.execute("sample.SumAndDifference", params.toArray());
         int sum = ((Integer) result[0]).intValue();    System.out.println("The sum is: "+ sum);
         int diff = ((Integer) result[1]).intValue();      System.out.println("The difference is: "+ diff);
      } catch (Exception exception) { System.err.println("Client: " + exception);  }
   }
}
```


XML-RPC Client Example (Python)

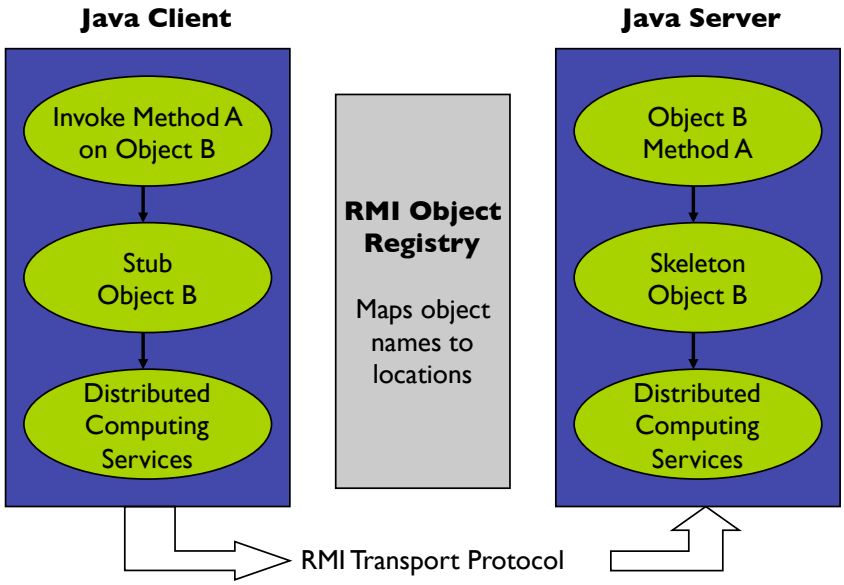
```
import xmlrpc.client

server = xmlrpc.client.Server("http://myserver:8888")

answer = server.sample.SumAndDifference(10, 40)

print("Sum:", answer[0])
print("Difference:", answer[1])
```

Java Remote Method Invocation (RMI)



RMI Example: Interface

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface Hello extends Remote {
    String sayHello() throws RemoteException;
}
```

RMI Example: Server

```
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class Server implements Hello {
    public Server() {}
    public String sayHello() {  return "Hello, world!";  }
    public static void main(String args[]) {
        try {
            Server obj = new Server();
            Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);
            Registry registry = LocateRegistry.createRegistry(8888);
            registry.bind("Hello", stub);
        } catch (Exception e) {  System.err.println("Server exception: " + e.toString());  }
    }
}
```


RMI Example: Client

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class Client {
    private Client() {}
    public static void main(String[] args) {
        String host = (args.length < 1) ? "localhost" : args[0];
        try {
            Registry registry = LocateRegistry.getRegistry(host, 8888);
            Hello stub = (Hello) registry.lookup("Hello");
            String response = stub.sayHello();  // RPC!
            System.out.println("response: " + response);
        } catch (Exception e) {  System.err.println("Client exception: " + e.toString());  }
    }
}
```