Remote Procedure Calls

RPC Model

Name Server

1. Register
2. Lookup
3. Request
4. Response

RPC Client

RPC Server

RPC protocol

Client stub

Server stub

Caller (client)

Callee (server)

Arguments

Return value

Request

Reply

RPC protocol

RPC Call

Arguments

Return value

Request

Reply

RPC protocol

Client

Server

Transparency

Semantics

Examples?

Idempotency

Can you re-execute operations without harmful effects?

Idempotent operations

Make "remote-ness" invisible to programmer

Choice of timeout values

Choice of TCP vs. UDP

Can also be automatically generated with language support

Initial, network programming involved hand-coded stubs

Popular variant: Remote Method Invocation (RMI)

Remote Procedure Call (RPC)

Remote Procedure Call (RPC) Issues

Fate sharing

Advantages/Disadvantages?

UDP vs. TCP

25/10/14

Sean Barker

11/12/14

RPC Semantics

At-most once

At-least once

Exactly once

Idempotent operations

Can you re-execute operations without harmful effects?

Idempotency

RPC Layer deals with network issues (e.g., TCP vs. UDP)

Stubs can be automatically generated with compiler

With language support, can make remote procedure call

Server stub unmarshals arguments

Client stub indicates which procedure should run at requests, blocks waiting for response

Server stub marshals arguments, transmits to client

Client issues request by calling stub procedure

Marshals results, transmits to client

Obtain handle to remote object, invoke method on object

E.g., file system abstraction to hide block layout, process abstraction for scheduling/fault isolation

Need common primitives/abstractions to hide complexity

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?

Ask DHT to route update until nodes who are responsible for keeping track of Load in desired range are found

Timeout value

Which nodes have Load = [0,1.0]?
Encoding calls in XML and uses HTTP for transport.

- Compiler layout of structures
- Representation of base types

The server stub marshals the return parameters and returns them to the client program.

Floating point: IEEE 754 versus non-standard

A well-formed XML document corresponds to an element tree encoded as name-value pairs.

RPC marshalling:

```java
public static void main (String[] args) {
    array[1] = new Integer(y-x);
    array[0] = new Integer(x+y);
    public class Server {
        import org.apache.xmlrpc.server.PropertyHandlerMapping;
        import org.apache.xmlrpc.server.XmlRpcServer;
        WebServer server = new WebServer(8888);
        xmlRpcServer = server.getXmlRpcServer();
        try {
            xmlRpcServer.setHandlerMapping(phm);
            public Integer[] SumAndDifference(int x, int y) {
                return parameters from the message.
            }
        } catch (Exception exception) {  System.err.println("Server: "+ exception);   }
    }
}
```

Conversion strategy:
- Flat types (e.g., structures, arrays); must pack data
- Base types (e.g., integers, floats); must convert

System Encodings:

Big-endian:
- `00000010 00010001 00100010 01111110`

Little-endian:
- `01111110 00100010 00010001 00000010`

High address: `01111110`
Low address: `00000010`
Converting Structures

Failures and RPC Semantics
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 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><i4>50</i4></value></param>
<param><value><i4>30</i4></value></param>
</params>
<value><struct><value><i4>139</i4></value>
<value><i4>31</i4></value></struct>sum
<value><i4>12</i4></value>diff</value>
</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>
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);
            xmlRpcServer.setHandlerMapping(phm);
            server.start();
        } catch (Exception exception) { System.err.println("Server: "+ exception);  }
    }
}

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);
        xmlRpcServer.setHandlerMapping(phm);
        server.start();
    } catch (Exception exception) { System.err.println("Server: "+ exception);  }
}

import java.util.List;
import java.util.Arrays;
import org.apache.xmlrpc.client.XmlRpcClient;
import org.apache.xmlrpc.client.XmlRpcClientConfigImpl;
import org.apache.xmlrpc.

public class Client {
    public static void main (String[] args) {
        try {
            XmlRpcClientConfigImpl config = new XmlRpcClientConfigImpl();
            XmlRpcClient client = new XmlRpcClient();

            config.setServerURL(new URL("http://" + args[0] + ":8888"));
            client = new XmlRpcClient(config);

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

            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);  }
    }
}
import xmlrpcLib
server = xmlrpcLib.Server('http://lucana:8888')

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

print "Sum:", answer[0]
print "Difference:", answer[1]
RMI Example: Interface

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

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

RMI Example: Server

```java
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());
        }
    }
}
```
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();
            System.out.println("response: " + response);
        } catch (Exception e) {
            System.err.println("Client exception: " + e.toString());
        }
    }
}