CS 497MDM, Project Report
Design and Construction of Distributed Object Component Systems
Fall 2000

Integrating Jini/RMI with Soap

Mathew George, Ritesh Jain and Ashish Agarwal

Introduction

JINI technology provides simple mechanisms which enable devices to plug together to form an impromptu community - a community put together without any planning, installation, or human intervention. However since the Jini technology is based on Java and RMI, its deployment requires the presence of a JVM and associated RMI machinery on both the client and server side. But with the proliferation of Internet enabled handheld devices, it is becoming increasingly important to enable access to services provided by Jini communities on the network. However resource limitations make it impractical to build a JVM on these devices.

The Simple Object Access Protocol (SOAP) is a lightweight protocol based on HTTP/XML that facilitates interoperability among a wide range of programs and platforms, making existing distributed applications accessible to a broader range of users. The presence of an HTTP transport on these devices makes SOAP an attractive option for mobile devices to interact with JINI communities. The firewall friendly nature of HTTP traffic makes SOAP especially attractive in a corporate environment. But the textual structure of the SOAP encoding makes it somewhat inefficient in dealing with large amounts of data.

Our goal in this project is to design a framework that can provide interoperability between JINI/Java-RMI services and SOAP-RMI based services.

Why SOAP?

Since there is lack of a common standard, that is adhered to, by all communicating entities in Distributed Systems, we need to provide a solution that uses the lowest common denominator of the capabilities that these competing standards share. There are two problems to solve here:

1. What is the communicating medium?

HTTP is a readily available transport protocol, which by virtue of its extensive usage on the Internet, has been comprehensively tested for reliability and accepted as the standard for communication on the World Wide Web. It seems to be the best choice of the communicating protocol between the diverse technologies. The only problem being, lack of 'imposed' structure in the data sent across HTTP. HTML frames are the most common way of sending data across the Internet. The problem with HTML is that, since the data is not isolated from the rendering information, it is unstructured. This brings us to the second question.

2. How will the data be represented?

XML isolates data from its rendition information, and instead keeps meta-data along with the data, making the document easy to query. Since this emerging standard has been widely accepted by everyone as the next standard for information transfer on the Internet, there has already been a lot of work extending support for XML documents.

SOAP uses HTTP as the communicating medium and represents documents using XML. It is hence the best choice as a communicating standard among diverse distributed component technologies.

A client wanting to call a function formats a request in the SOAP-XML encoding, and sends it to the server over any mutually agreeable communication protocol. The server runs some sort of listener that accepts the incoming SOAP calls, reads the information from the XML SOAP packet, and maps them to software functionality on the server.

This business logic processes the request and returns output to the listener, which formats the output into a response packet in the SOAP-XML encoding and returns it to the client. A separate XML file contains a description of the services provided by the server in and encoding called SDL (Service Descriptor Language).

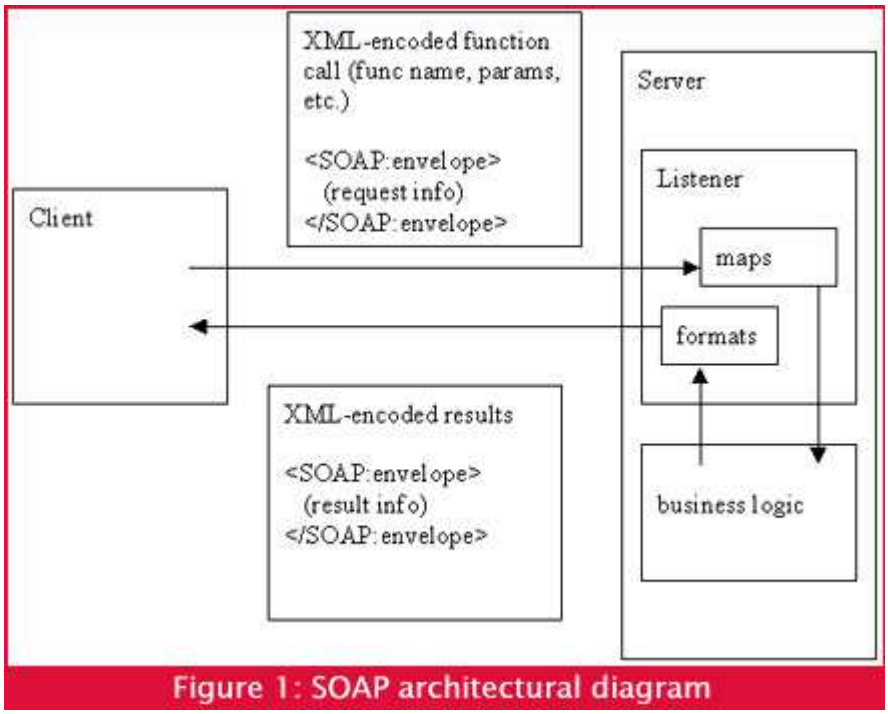


Figure 1: SOAP architectural diagram

Description of System Architecture:

The bridge-architecture is extensible to other distributed component technologies. Our vision is to make a SOAP-JINI bridge whose implementation is generic, so that COM, CORBA bridges for soap can be plugged into the bridge, with relative ease.

The bridge is essentially composed of the following components:

[1] **Soap Client:**

This component is language independent. It can be coded in C++/Java/Python/Perl or any language/scripting tool, which is capable of using SOAP. Since SOAP uses HTTP as the transport mechanism and XML for the document structure, this can be used to extend the Jini community to a large number of applications and users.

[2] **Bridge Core:**

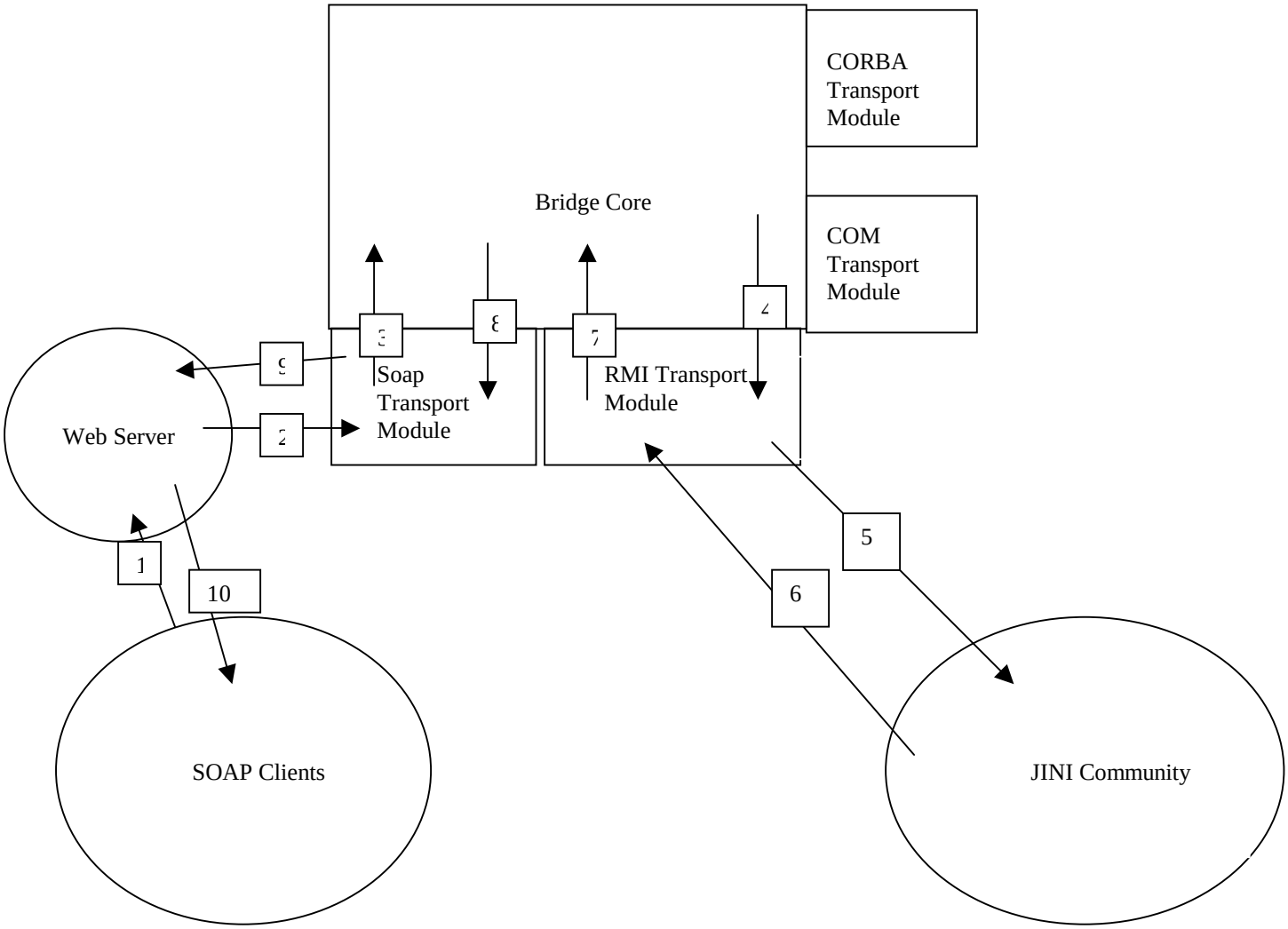
This is the centralized core component of our system. It provides 'hooks' to which we can plug in transport modules of any distributed component technology that we wish to interface with SOAP.

[3] **SoapTransportModuleServlet:**

This component plugs into the BridgeCore. It allows heterogeneous soap clients to communicate with the cluster of communities (only Java/RMI at present) of distributed component technologies, which may have plugged in their transport modules into the BridgeCore.

[4] **RMITransportModule**

This is the plug-in of the Java/RMI world into the bridge. It provides access to the services offered in the Jini domain, by using the reflection API to dynamically get the object reference of a particular service and invoke a method on that object.



Future Work :

We have made a Bridge for SOAP and Java/RMI. Our architecture is such that, this can be easily extended for other technologies. CORBA, DCOM transport modules can be made which plug into the Bridge Core and provide support for SOAP clients.

The Bridge Core can be augmented with a leasing mechanism to make the entire system 'self-healing'. This will also allow for commercial viability of the BridgeDeployment, since the users can be charged for a lease.

A Security mechanism can be added to the system to authenticate users and services. This should also authenticate new plug-ins for competing technologies.

Conclusion:

The Bridge Architecture that we have proposed can be used for successful communication across different distributed component technologies. In particular, it will allow non-Java-aware clients to interact with the Java/RMI community, which will work towards the Jini vision of providing a community, where ubiquitous and light-weight clients, can use Jini service.