Universal Messaging Framework

(Interim Project Report)

Amit Dev, Amit Remesan
Arun M & Rosch F K

1 Requirement Analysis

Our study of requirements in this project we identified following components.

- Embedded Device

  A major part of our project is creating a component to control devices using XML. But unfortunately there no device available which can be controlled through XML based control string.

- XML DTDs

  All the communications in our system should be through XML. So for validation and other purposes we need to create extensible DTDs. The separation or tags through the namespace is also needed.

- XML Routing

  Fast and low latency routing of data is needed. The data will be routed from one station to another through the network.

- Remote Method Invocation

  A server is to be created to translate XML based procedure calls to corresponding system calls and result should be send back to client. The server should do this translation.

- Telephony Servers.

  Telephony servers are needed to translate telephonic commands. Its needed in fax and email handling.

2 Project Plan

To achieve at the aim of creation of a Universal Messaging Framework we are planing our work in the following way. The implementation details of the system is available in next section.

2.1 Embedded Device

Since there is no Embedded device controlled by XML we are going to create one. We are planing it on x86 since it can be easily tested in normal Personal Computers. It will be a device which can play MPEG-Audio or Ogg data. It should be a multi threaded application. It should have facilities to authenticate and network support.

We are planning embedded software in the following way.

Embedded Linux will act as the kernel of the OS. We dont need Real Time OS for the device we are planning. Since time is not very critical. Embedded debian is a good choice.

The player will be a multi threaded program. We will be using POSIX thread library in Linux with portability in consideration. The player will be a demon process listening to a port for XML control instructions. BSD sockets are used for network communication. The decoding of data and process listening to socket will run in two different threads.

Routing of the data will be done with the help of embedded Jabber servers. We are not decided on where to place the Jabber server. In the device or outside.

2.2 XML DTDs

For the most basic jobs XML DTDs used by the Jabber is more than sufficient. Our extensions will be put into a different name space so that both wont conflict each other.

2.3 XML Routing

XML Routing component is an important part of this project. We are using a customised version of Jabber IM for this.

Jabber is an instant messaging platform with the ability to packetize messages in XML documents–in effect as data transport. Any PC or Net-native device can instantly communicate with all of the infrastructure on the Internet that can parse and act on XML. This puts new XML databases–which are designed to parse XML and store structured data or documents–in a powerful context. Anywhere XML is the preferred way to receive and send information, we have infrastructure already conformed to instant messaging.

The Jabber system is distinguished from existing instant messaging (IM) services by several key features:

- XML foundation
- distributed network
- open protocol and codebase
- modular, extensible architecture

The Jabber.org Project (http://jabber.org) was started by Jeremie Miller in early 1998. Jabber's open XML protocol has remained relatively unchanged since t he 1.0 release, and was submitted as an RFC to the IETF's Instant Messaging and Presence Protocol working group on 2000-06-15. And so we will be using this as the standard protocol of our system. Since there is a good chance that it will soon become the Internet Standard.

In this sense, Jabber is a ubiquitous XML router, streamlined for (and by) embedded Linux, or any other device-driving OS that supports XML-based messaging. Real (enough) time is the key. While e-mail is technically capable of routing XML documents, it lacks awareness of presence, which is key to Instant Messaging in general. This makes Jabber something of a new message transport breed.

2.4 Remote Method Invocation

XML-RPC and SOAP standards will be used as the standards for the XML. Python is the only language which is having a strong XML-RPC library. It can be used to help in translating XML instructions to native system calls.

2.5 Telephony Server

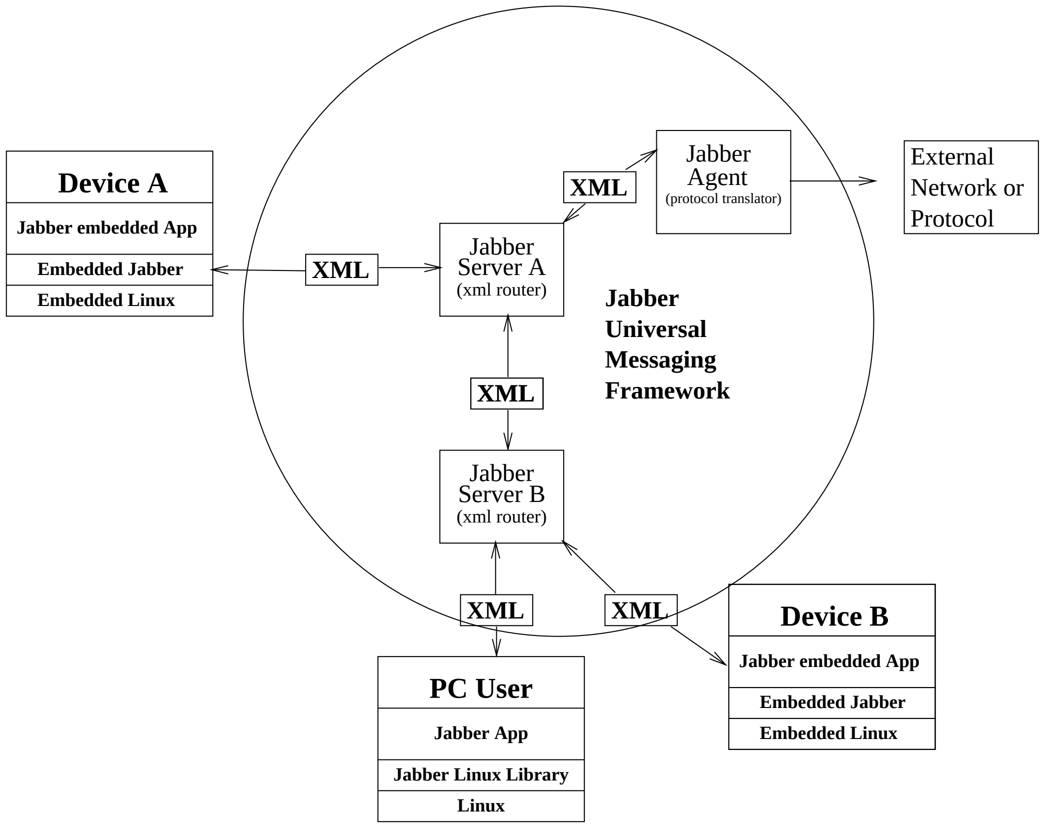
We need several different telephony servers to handle telephony related functions. Primarily we will be using VOICE/DATA/FAX Modem as our telephony hardware. Later it can be extended for other complex telephony hardware.

3 Detailed Design

Detailed design and diagrams are given below.

3.1 Overview of Universal Messaging Framework

From a higher level the system can be viewed as collection of Jabber server and Clients communicating each other through XML. The following figure gives a good view of the system.

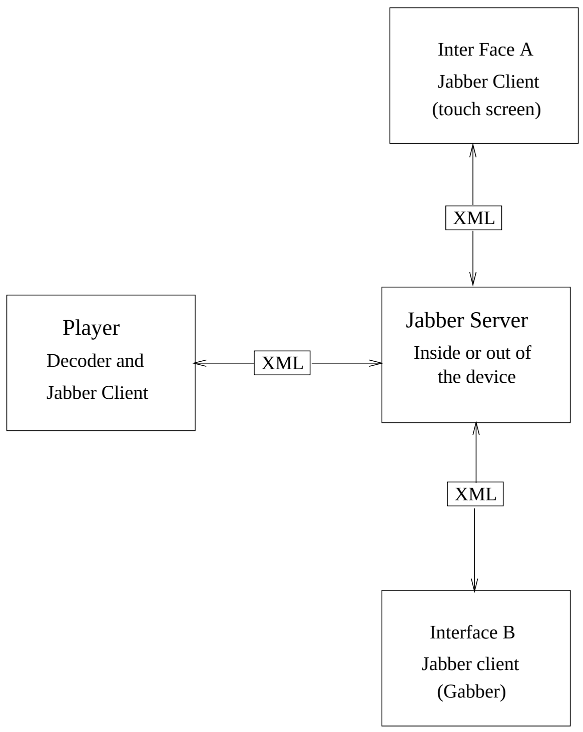


Emerging tools like XML optimized databases can be used to leverage XML communications streams and to provide whole new services. The fact that all communication with Jabber is via XML documents means that emerging XML tools and services can be easily “plugged” into the message stream carried by Jabber. Extrapolate that, and you’ve got a device driving framework for the known world. Because it is now quite simple for you, or a contractor you hire, to program your home–say through X10, which was designed for that purpose–you can operate, and receive notification about, all the home electronic conveniences you already have: your security system, your lighting, your sprinklers. You can also program those devices to tell you if they’re on, functioning and ready to receive instructions. So the concept of presence applies in more than one direction. Two way conversation that includes presence is Jabber infrastructure on a common XML framework.

3.2 Embbeded MP3 Player

The embedded device we are planing is a x86 hardware with Embedded GNU/Linux as the OS. The core application running in the system will be a multi-threaded Jabber Client. It will run the decoding of MPEG stream in one thread.

Jabber server can be place inside the device or external device can be used as the server. The figure shows the architecture of the player.



3.3 XML DTDs

As said earlier the XML Protocols devised in Jabber architecture will be sufficient for most of the work. Also the standards like SOAP and XML-RPC will be followed when ever possible.

Following example will give an idea of XML tags that will be used in the control messages to devices.

So let's say you've got a computer running the service as x10.homelan.net, and a couple x10 devices such as a light on your desk as desklight@x10.homelan.net and a motion sensor as driveway@x10.homelan.net.

The Jabber client on your wireless web pad is connected to your homelan.net server, and you're subscribed to the presence of both of the x10 devices. Your client receives:

```
<presence from="driveway@x10.homelan.net">
<status>Motion Detected</status>
</presence>
```

alerting that some one arrived, and system sends

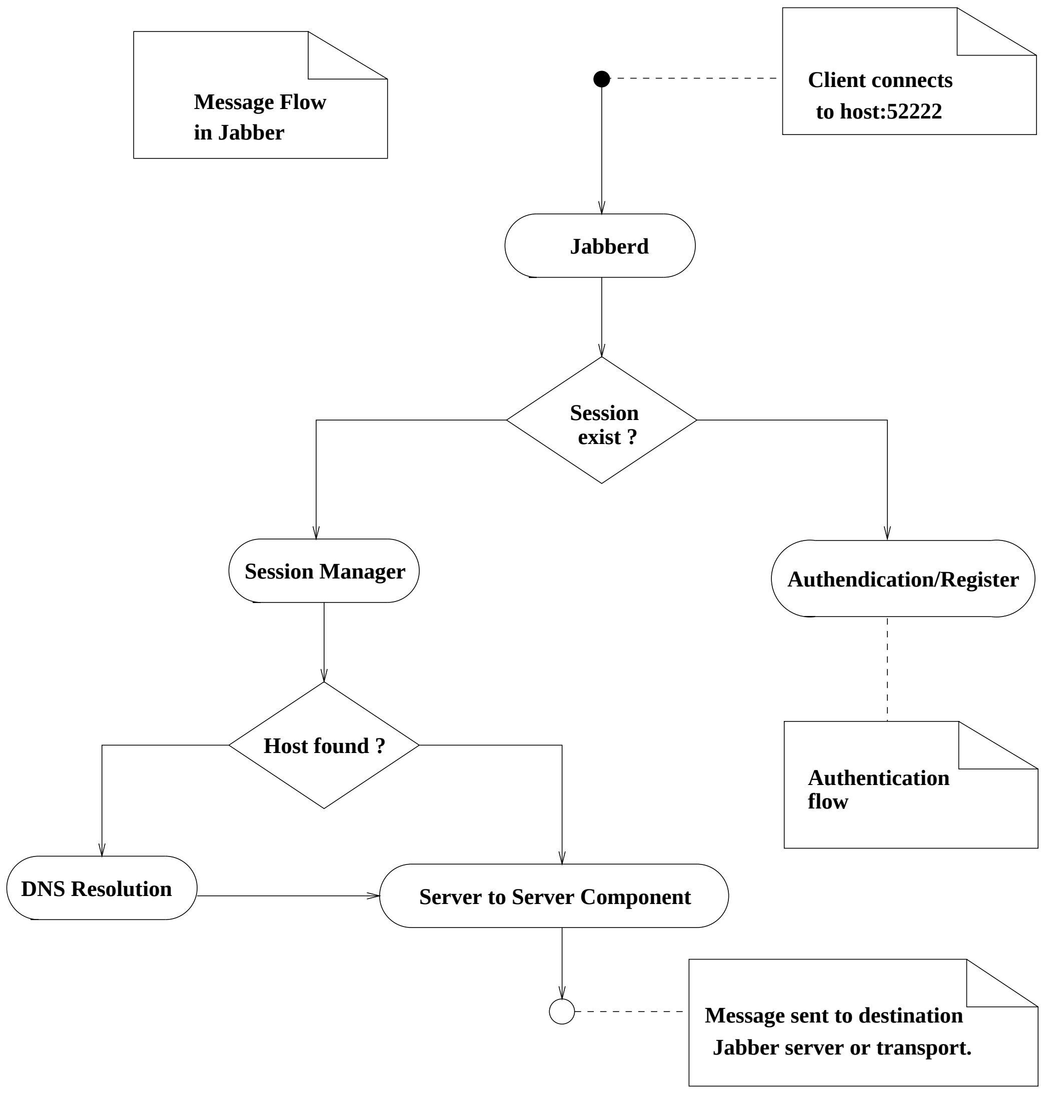
```
<message to="desklight@x10.homelan.net">
<body>on</body>
</message>
```

Similarly this system can be extended for different proposes.

3.4 XML Routing

Jabber does the most of the XML Routing part. The URI used in the Jabber protocol is same as that of electronic mail. It may create some difficulty first.

The following diagram of Jabber Message flow gives the idea of routing in Jabber.



3.5 Remote Method Invocation

For the Remote Method Invocation XML-RPC standards are used. The XML request will be given to a Jabber client which acts as the XML-RPC server. Server does the translation from and to XML. Details of XML-RPC is as follows.

3.5.1 Overview

XML-RPC is a Remote Procedure Calling protocol that works over the Internet.

An XML-RPC message is an HTTP-POST request. The body of the request is in XML. A procedure executes on the server and the value it returns is also formatted in XML.

Procedure parameters can be scalars, numbers, strings, dates, etc.; and can also be complex record and list structures.

3.5.2 Request example

Here's an example of an XML-RPC request:

```
POST /RPC2 HTTP/1.0
User-Agent: Gabber/1.0 (GNU/Linux)
Host: foo.foodomain
Content-Type: text/xml
Content-length: 181


<?xml version="1.0"?>
<methodCall>
<methodName>examples.getStateName</methodName>
  <params>
    <param>
       <value><i4>41</i4></value>
    </param>
  </params>
</methodCall>
```

3.5.3 Header requirements

The format of the URI in the first line of the header is not specified. For example, it could be empty, a single slash, if the server is only handling XML-RPC calls. However, if the server is handling a mix of incoming HTTP requests, we allow the URI to help route the request to the code that handles XML-RPC requests. (In the example, the URI is /RPC2, telling the server to route the request to the "RPC2" responder.)

We are using the Jabber id schema as the URI. So that the routing can be done.

A User-Agent and Host must be specified.

The Content-Type is text/xml.

The Content-Length must be specified and must be correct.

3.5.4 Payload format

The payload is in XML, a single <methodCall> structure.

The <methodCall> must contain a <methodName> sub-item, a string, containing the name of the method to be called. The string may only contain identifier characters, upper and lower-case A-Z, the numeric characters, 0-9, underscore, dot, colon and slash. It's entirely up to the server to decide how to interpret the characters in a methodName.

For example, the methodName could be the name of a file containing a script that executes on an incoming request. It could be the name of a cell in a database table. Or it could be a path to a file contained within a hierarchy of folders and files.

If the procedure call has parameters, the <methodCall> must contain a <params> sub-item. The <params> sub-item can contain any number of <param>s, each of which has a <value>.

3.5.5 Response example

Here's an example of a response to an XML-RPC request:

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 158
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:08 GMT
Server: UserLand Frontier/5.1.2-WinNT


<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value><string>South Dakota</string></value>
    </param>
  </params>
</methodResponse>
```

3.5.6 Response format

Unless there's a lower-level error, always return 200 OK.

The Content-Type is text/xml. Content-Length must be present and correct.

The body of the response is a single XML structure, a <methodResponse>, which can contain a single <params> which contains a single <param> which contains a single <value>.

The <methodResponse> could also contain a <fault> which contains a <value> which is a <struct> containing two elements, one named <faultCode>, an <int> and one named <faultString>, a <string>.

A <methodResponse> can not contain both a <fault> and a <params>.

3.6 Telephony Server

The telephony system will be a simple extension of the XML-RPC mechanism given earlier.