www.euro6ix.net

Title:	Document Version:
Technical Report TR4.1A.8 **Advanced Applications**	0.9

Project Number:	Project Acronym:	Project Title:
IST-2001-32161	Euro6IX	European IPv6 Internet Exchanges Backbone

Contractual Delivery Date:	Actual Delivery Date:	Deliverable Type* - Security**:
31/12/2002	04/03/2003	R – PU

* Type: P - Prototype, R - Report, D - Demonstrator, O - Other
** Security Class: PU- Public, PP – Restricted to other programme participants (including the Commission), RE – Restricted to a group defined by the consortium (including the Commission), CO – Confidential, only for members of the consortium (including the Commission)

Responsible and Editor/Author:	Organization:	Contributing WP:
David Fernández	UPM	WP4

Authors (organizations):

Cesar Martin (novaGnet), Jesus Muñoz (novaGnet), Sathya Rao (Telscom), Tim Chown (UoS), Xiang Fei (UoS), Antonio F. Gómez-Skarmeta (UMU), Diego Acosta (UPM), Tomas P. de Miguel (UPM), Maria J. Perea (UPM), Juan Quemada (UPM), Tomás Robles (UPM), Javier Sedano (UPM).

Abstract:

This Technical Report associated to deliverable D4.1A summarizes the objectives and results of the A4.2 Advanced Applications sub-activity during the first year of Euro6IX project. Work in this sub-activity has concentrated on four different areas: IPv6 capable Audioconference, IPv6 capable ISABEL, IPv6 Unified Instant Messaging and IPv6 Groupware Applications.

Keywords:

Agent-based computing, AGWS, AudioConferencing, Euro6IX, Instant Messaging, IPsec, IPv6, IX, Peer-to-Peer, SIP, Unified IM, VNC, SDP, SIP Proxy, SIP UA.

© Euro6IX Consortium

Revision History

Revision	Date	Description	Author (Organization)
v0.1	15/12/2002	Document creation	David Fernández (UPM)
v0.2	17/12/2002	Integration of Unified Instant Messaging Contributions	Xiang Fei (UoS)
v0.3	18/12/2002	Edition of Unified Instant Messaging contribution for typos, consistency, etc	Tim Chown (UoS) Cesar Martin (novaGnet) Jesus Muñoz (novaGnet)
v0.4	20/12/2002	Integration of SIP for IPv6 contribution	Tomás Robles (UPM)
v0.5	27/01/2003	Integration of VOCAL porting contribution	Tim Chown (UoS)
v0.6	29/01/2003	6VOICE porting	Sathya Rao (Telscom)
v0.7	15/02/2003	Integration of IPv6 capable ISABEL and IPv6 Groupware contributions	David Fernández (UPM)
v0.8	30/02/2003	Conclusions and executive summary added. Final revision.	David Fernández (UPM)
v0.9	04/03/2003	Logos added an PDF generated	Jordi Palet (Consulintel)

Executive Summary

This Technical Report associated to deliverable D4.1A summarizes the objectives and results of the A4.2 Advanced Applications subactivity during the first year of Euro6IX project. Work in this subactivity has concentrated on four different areas:

- **IPv6 capable Audioconference**, which deals with the porting of audioconferencing services to IPv6. Audioconferencing services require the involvement of several key technologies, especially SIP, which is also investigated. Three different works are described: 6Voice and Vocal, which are audioconferencing applications with SIP support; and a SIP library (jSIP) which is ported to IPv6.

- **IPv6 capable ISABEL**, which focuses on the evolution of ISABEL application to support and make use of the IPv6 features, such as security, QoS, mobility, etc, which will be offered by the Integrated Service Framework created over the Euro6IX network.

- **IPv6 Unified Instant Messaging,** which deals with IPv6 instant and unified messaging applications. Instant Messaging (IM) has become one of the most popular tools for people to communicate with peers over the Internet. With the advent of IPv6, many more computers and (mobile) devices will be connected to the Internet, and the Peer-to-Peer (P2P) applications will be more and more pervasive. The IPv6 Unified Instant Messaging system aims at the vision of providing advanced messaging services for peers to exchange information, share resources and cooperate or collaborate. The document summarizes the research and development work on the IPv6 Unified IM system, showing the advantages of IPv6 in peer-to-peer systems.

- **IPv6 Groupware Applications,** which deals with the porting of a couple of groupware representative tools, including a shared project space called Agora Groupware Web Server (AGWS), which includes doc repository, calendar, forums, groups, etc, and the Virtual Network Computing (VNC) remote application sharing tool. The general goal in this area is to assess the state of porting of a representative set of open software developments and getting feedback about groupware application porting.

Table of Contents

1. Introduction ***7***

2. IPv6 Capable Audioconferencing ***8***

2.1 Introduction **8**

2.2 Objectives **8**

2.3 6VOICE: Voice over IP (VOIP) on IPv6 **9**

2.3.1 Session initiated protocols for real-time applications 9

2.3.2 Architecture 10

2.3.3 Implementation 13

2.3.3.1 Telscom VoIP Implementation 13

2.3.3.2 User Interface (UI) 13

2.3.3.3 Networking 13

2.3.3.4 SIP 14

2.3.3.5 RTP 14

2.3.3.6 Messages 14

2.3.3.7 System Requirements 14

2.3.3.8 Application download 14

2.3.4 Evaluation 14

2.4 VOCAL **17**

2.4.1 Vocal Features 17

2.4.2 Porting VOCAL 17

2.4.3 IST2002 19

2.4.4 ENUM 19

2.5 SIP Over IPv6 **19**

2.5.1 Objectives 20

2.5.2 Technical Approach 20

2.5.2.1 SIP Overview 21

2.5.2.2 The jSIP Library 22

2.5.3 System Architecture 22

2.5.3.1 Design Criteria: 23

2.5.3.2 Enhancements and modification over the initial library 23

2.5.3.3 Porting o IPv6 23

2.5.3.4 Implementing SIP elements 24

2.5.3.5 Evaluation and Testing 24

2.6 Conclusions and Future Work **25**

3. IPv6 capable ISABEL ***26***

3.1 Objectives **26**

3.2 Security and Mobility for ISABEL **26**

3.3 First Year Results **27**

3.4 Conclusions and Future Work **28**

4. IPv6 Unified Instant Messaging Systems ***30***

4.1 Introduction **30**

4.1.1 IPv4 (NAT) Limitations for P2P and IM....31
4.1.2 IPv6 Advantages....31

4.2 IETF and other IM related standards....32
4.2.1 Instant Messaging and Presence Protocol (IMPP) Working Group....32
4.2.2 Jabber Software Foundation....33
4.2.3 SIMPLE Working Group....34
4.2.4 Wireless Village....35

4.3 IPv6 Unified Instant Messaging Systems....36
4.3.1 Two approaches to the same goal....36
4.3.1.1 The Unified Instant Messaging system....36
4.3.1.2 A complete Peer-to-Peer system....37
4.3.2 The network platform and agent-based approach....37
4.3.2.1 Network Platform for Messaging over IPv6....37
4.3.2.2 Agent-Based Unified Instant Messaging System....38

4.4 First Year Results....39
4.4.1 Unified Instant Messaging....39
4.4.2 Towards a peer-to-peer system....40
4.4.2.1 Java development details....40
4.4.3 System convergence and interoperability....42

4.5 Conclusions....43

4.6 Future Work....43

5. IPv6 Groupware Applications....44

5.1 Objectives....44

5.2 AGWS and VNC....44

5.3 Conclusions and Future Work....46

6. References....47

Table of Figures

Figure 2-1: Protocol Architecture in SIP ***11***
Figure 2-2: SIP Signaling ***13***
Figure 2-3: 6VOICE Caller interface window ***14***
Figure 2-4: 6VOICE Callee interface window ***15***
Figure 2-5: 6VOICE interface after establishment of a connection ***15***
Figure 2-6: 6VOICE GUI used to register user locations ***15***
Figure 2-7: 6VOICE test scenario ***16***
Figure 2-8: 6VOICE Mobile IPv6 test network ***16***
Figure 2-9: JUNIT structure ***24***
Figure 2-10: Validation Scenario ***25***
Figure 4-1: A Model for Presence System ***32***
Figure 4-2: A Model for Messaging System ***33***
Figure 4-3: A Generalized Architecture of XMPP ***34***

1. INTRODUCTION

This Technical Report associated to deliverable D4.1A summarizes the objectives and results of the A4.2 Advanced Applications subactivity during the first year of Euro6IX project. Work in this subactivity has concentrated on four different areas:

- **IPv6 capable Audioconference**, which deals with the porting of audioconferencing services to IPv6. Audioconferencing services require the involvement of several key technologies, especially SIP, which is also investigated. Three different works are described: 6Voice and Vocal, which are audioconferencing applications with SIP support; and a SIP library (jSIP) which is ported to IPv6.

- **IPv6 capable ISABEL**, which focuses on the evolution of ISABEL application to support and make use of the IPv6 features, such as security, QoS, mobility, etc, which will be offered by the Integrated Service Framework created over the Euro6IX network.

- **IPv6 Unified Instant Messaging,** which deals with IPv6 instant and unified messaging applications. Instant Messaging (IM) has become one of the most popular tools for people to communicate with peers over the Internet. With the advent of IPv6, many more computers and (mobile) devices will be connected to the Internet, and the Peer-to-Peer (P2P) applications will be more and more pervasive. The IPv6 Unified Instant Messaging system aims at the vision of providing advanced messaging services for peers to exchange information, share resources and cooperate or collaborate. The document summarizes the research and development work on the IPv6 Unified IM system, showing the advantages of IPv6 in peer-to-peer systems.

- **IPv6 Groupware Applications,** which deals with the porting of a couple of groupware representative tools, including a shared project space called Agora Groupware Web Server (AGWS), which includes doc repository, calendar, forums, groups, etc, and the Virtual Network Computing (VNC) remote application sharing tool. The general goal in this area is to assess the state of porting of a representative set of open software developments and getting feedback about groupware application porting.

Next sections in detail the objectives and results of the work carried out in each of the areas presented above during the first year of the project.

2. IPv6 CAPABLE AUDIOCONFERENCING

2.1 Introduction

Audioconference services are one of the key services any IPv6 network is going to offer to end user. In the framework of Euro6IX, we have explored different videoconference applications in order to deploy videoconference services over IPv6 networks.

Telscom has developed 6Voice, the IPv6 voice over IP application and has been tested across wide are networks end-to-end. The tests were also conducted using DiffServ and proprietary QoS flowlabel implementation need the QoS inbuilt into the IP network for the good performance. For the session negotiating SIP has been used, as in the other experiments carried out in the activity.

The initial IPv6 porting for Vovida Open Communication Application Library (Vocal) has been added by University of Southampton. VOCAL is an open source project targeted at facilitating wider adoption of Voice over IP technology (see http://www.vovida.org). VOCAL was originally developed by Vovida Networks, which has since been acquired by Cisco systems. However the project remains open source.

Finally we investigated the provision of general SIP services over IPv6 networks. This work, performed by University of Madrid, was based on the Open source code library jSIP (http://jsip.sourceforge.net). This work intends to provide generate a SIP library IPv6 enabled, which allows the creation of mayor SIP elements (User Agents and Proxies); and to develop a mechanism for allowing interworking between IPv4 and IPv6 SIP.

We selected an OpenSource library (jSIP) which was originally developed for SIP over IPv4 networks and is implemented using Java. This provides some additional benefits such gaining experience in porting general purpose libraries, and not only final applications, and the development of general patterns for porting software which is much more general applicable when porting general software than when porting ad-hoc applications.

Finally the final library has been evaluated by the creation of key SIP elements such as UAs and Proxies, enabling the creation of a Sip service provision environment, which will be further investigated for interworking between IPv4 and IPv6 audioconferencing applications, using the features of the new library.

2.2 Objectives

The general objective of this subactivity is the study and evaluation of VoIP services over IPv6, providing a complete set of component for the deployment of audioconferencing services of the Euro6IX network.

To reach that general objective, some particular goals are defined:

- To migrate to IPv6 of a VoIP audioconferencing system already developed for IPv4
- To provide a SIP support for the deployment of Audioconferencing services over the Euro6IX network

- To enable IPv4 based services to interact with IPv6 services running at Euro6IX network

- Evaluate and demonstrate the correct operation of the application and its installation in several platforms.

2.3 6VOICE: Voice over IP (VOIP) on IPv6

Telscom has developed 6VOICE, the IPv6 voice over IP application and has been tested across wide area networks end-to-end. The tests were also conducted using DiffServ and proprietary QOS flowlabel implementation system to test the impact of QoS models and it is important that voice like applications need the QoS inbuilt into the IP network for the good performance.

VoIP means transmission of voice over IP networks. Advantages of VoIP include:

- Lower cost long distance and reduces access charges.
- More efficient backbone for new services.
- Reducing operating costs.

On the other hand, there are also some challenges in VoIP. Since IP packets carrying voice are treated just like IP packets carrying any other type of data, they are subjected to delays, loss and retransmissions. This happens when the network is congested. So the quality of service becomes a very important issue.

2.3.1 Session initiated protocols for real-time applications

The Session Initiation Protocol (SIP) is an application-layer protocol that can establish and control multimedia sessions or calls. These multimedia sessions include multimedia conferences, distance learning, Internet telephony and similar applications. SIP can invite a person to both unicast and multicast sessions; the initiator does not necessarily have to be a member of the session it is inviting to. Media and participants can be added to an existing session. SIP can be used to "call" both persons and "robots", for example, to invite a media storage device to record an ongoing conference or to invite a video-on-demand server to play a video into a conference.

SIP can be used to initiate sessions as well as invite members to sessions that have been advertised and established by other means. (Sessions may be advertised using multicast protocols such as Session Announcement Protocol (SAP), electronic mail, news groups, web pages or directories (LDAP), among others.)

SIP supports some or all of five facets of establishing and terminating multimedia communications:

1. **User location:** determination of the end system to be used for communication;

2. **User capabilities:** determination of the media and media parameters to be used;

3. **User availability:** determination of the willingness of the called party to engage in communications;

4. **Call setup:** "ringing", establishment of call parameters at both called and calling party;

5. **Call handling:** including transfer and termination of calls.

SIP may also be used in conjunction with other call setup and signalling protocols. In that mode, an end system uses SIP protocol exchanges to determine the appropriate end system address and protocol from a given address that is protocol-independent. As an example, it may be used to determine that the caller is reachable via the public switched telephone network (PSTN) and indicate the phone number to be called, possibly suggesting an Internet-to-PSTN gateway to be used.

SIP can also initiate multi-party calls using a multipoint control unit (MCU) or fully-meshed interconnection instead of multicast. Internet telephony gateways that connect PSTN parties may also use SIP to set up calls between them.

SIP can invite users to sessions with and without resource reservation. SIP does not reserve resources, but may convey to the invited system the information necessary to do this. Quality-of-service guarantees, if required, may depend on knowing the full membership of the session; this information may or may not be known to the agent performing session invitation.

SIP is designed as part of the overall IETF multimedia data and control architecture currently incorporating protocols such as RSVP for reserving network resources, the real-time transport protocol (RTP) for transporting real-time data and providing QOS feedback, the real-time streaming protocol (RTSP) for controlling delivery of streaming media, the session announcement protocol (SAP) for advertising multimedia sessions via multicast and the session description protocol (SDP) for describing multimedia sessions, but the functionality and operation of SIP does not depend on any of these protocols

2.3.2 Architecture

Figure 2-1 above shows the SIP protocol architecture. A connection between the participants can be obtained either by TCP or UDP. TCP provides a guaranteed connection between the participants. For real time data transmission RTP is used on top of UDP. SIP protocol supports both IPv4/IPv6. SIP protocol supports different types of network like Circuit switched network (SONET /SDH) or Semi Packet switched network (ATM) or Packet switched network (Ethernet) and Dial up network (V.34).

The Session Initiation Protocol has been designed by the Internet Engineering Task Force (IETF). The Multiparty Multimedia Session Control (MMUSIC) working group of IETF developed SIP. SIP is a lightweight protocol based on HTTP. It was originally designed for multimedia conferencing on the Internet.

SIP is rather independent of the environment and can be used with several protocols. In fact, any datagram or stream protocol that delivers a whole SIP request or response in full can be used. Such protocols are UDP and TCP in the Internet.

SIP does not require any reliable transport protocol and simple clients can be implemented using only UDP transport. However, it is recommended that servers should support both UDP and TCP. A TCP connection is opened only if a UDP connection cannot be established. Reliable transport is achieved by retransmitting requests. The system works much like a three-way handshake. The use of application layer reliability has the advantage that the timers can be adjusted according to the requirements.

The functionality of SIP is concentrated on signalling which is contrary to H.323 where the protocol includes all the required functions of conferencing. The SIP protocol includes all basic signalling, user location, registration and as an extension, advanced signalling. The other services such as directory access reside in separate protocols integrated into the SIP environment.

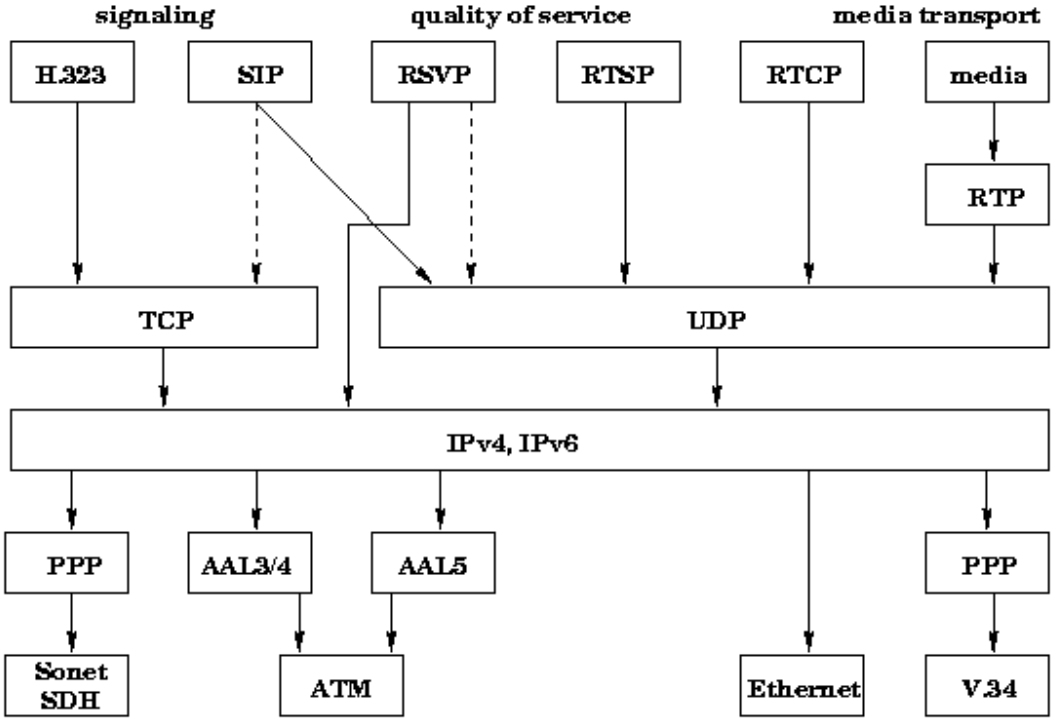

Figure 2-1: Protocol Architecture in SIP

SIP uses Session Description Protocol (SDP) to describe the capabilities and media types supported by the terminals. SDP is developed by IETF and messages are text based as in SIP. SDP messages list the features that must be implemented in the endpoints. SDP messages are mainly sent within SIP messages but also other ways can be used. SIP depends on the Session Description protocol (SDP) for carrying out the negotiation for codec identification. SIP supports session descriptions that allow participants to agree on a set of compatible media types

Sessions can be informed to a larger group of users using IETF's Session Announcement Protocol (SAP). SAP is primarily used for informing about large public conferences and broadcast streams like Internet television and radio. However, SIP could be used for this because of multicast signalling feature. For real-time transmission SIP architecture uses RTP.

Since Internet is an open network where everyone can receive and transmit packets relatively easily, so it requires protection to prevent spoofing of calls, denial of service, spamming (disturbing), etc.

Because SIP is based on HTTP it naturally has security mechanisms similar to HTTP. Authentication of the caller and the caller can be realised with HTTP mechanisms, including basic (clear-text password) and digest (challenge-response) authentication. Keys for media encryption are conveyed using SDP. SIP could use any transport layer or HTTP's security mechanisms, such as Secure Shell (SSH) or Secure-HTTP.

There are two transport layer protocols in the Internet protocol suite, TCP and UDP. TCP provides a reliable flow between two hosts. It is connection-oriented and thus can't be used for multicast. UDP provides a connectionless unreliable datagram service. To use UDP as a transport protocol for real-time traffic, some functionality has to be added. Functionality that is needed for many real-time applications is combined into RTP, the real-time transport protocol. RTP is standardized in RFC 1889. Applications typically run RTP on top of UDP as part of the transport layer protocol.

In practice, RTP is usually implemented within the application. RTP is designed to be independent from the underlying transport protocol and can be used over unicast as well as multicast.

The services provided by RTP include time stamping, sequence numbering, payload identification and source identification. The most important information in the RTP header is the timing information. The sender timestamps each RTP packet with the point in time the first sample in the packet was encoded. The receiver then uses these timestamps to reconstruct the original timing before playing back the data.

Sequence numbers are necessary to handle out-of-order packets and to detect lost packets.

A payload type identifier tells the receiving application how to interpret the payload. It identifies the format of the payload and specifies which encoding and compression schemes are used. Examples for payload formats are different versions of PCM, JPEG or H.261.

In audio conferences, source identification allows the receiving application to indicate to the user who is talking at the moment. In video applications with several senders, all sources send to the same multicast address, so source identification is needed to assign incoming packets to the proper video image.

The transport protocol RTP is augmented by RTCP, the real-time control protocol. Its main objective is to provide feedback on the quality of the data distribution. Applications may use this feedback to adapt to different network conditions.

Another function of RTCP is to keep track of participants when their internal identifiers change, and to distribute information about all participants in a session, for example their names and email addresses. Control packets are periodically transmitted to all participants in the session. Every control packet contains reception statistics in the form of receiver reports. The Reports are issued by every receiver for every source. They report the number and fraction of lost packets and the interarrival jitter as well as to which packets these statistics refer. We use the same method to measure and calculate delay variations and jitter as specified for the RTCP reports.

2.3.3 Implementation

2.3.3.1 Telscom VoIP Implementation

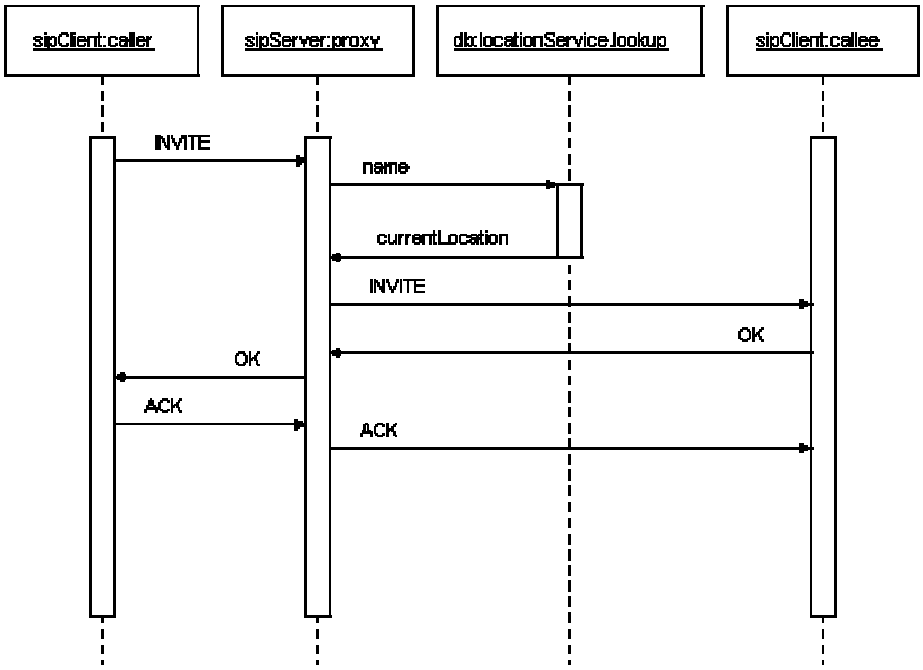

Figure 2-2: SIP Signaling

The design is based on object oriented programming style. So we have chosen Java for that purpose. The main reason to select Java was they have provided all the functionality needed for transferring real time data.

The project is divided into sub modules. It is broken down as follows.

- User Interface
- Networking
- SIP
- RTP
- Messages.

2.3.3.2 User Interface (UI)

This module basically consists of the front-end interface to user. It is a stand-alone application that uses the functionality of other modules specified. In our implementation, only an IP interface and hence the PC is used as a user terminal, instead of telephone. This module is used for gathering data and presenting data through the UI.

2.3.3.3 Networking

This module deals with networking part i.e. to open a connection with remote client, send messages to client, and receive messages from client. This is heart of the project. This is the communications medium between two users.

2.3.3.4 SIP

This module deals with how to initiate a session using SIP protocol. It contains details of SDP, SDP header, SIP header, SIP Server, and SIP Client.

2.3.3.5 RTP

This module deals with how to initiate an RTP session for real-time data transfer. This module also consists of how to control the RTP data with RTCP.

2.3.3.6 Messages

This module is used to display error and success messages during the entire process.

2.3.3.7 System Requirements

- Windows 2000 platform with IPv6 support.
- Sound card
- Microphone
- Speakers
- Java 2 Runtime environment
- JMF 2.1.1

2.3.3.8 Application download

6VOICE application package can be downloaded from Telscom website and tested at user sites: http://www.telscom.ch/6voice

2.3.4 Evaluation

Next Figures show the output of our current 6Voice release with the interoperability tests conducted.

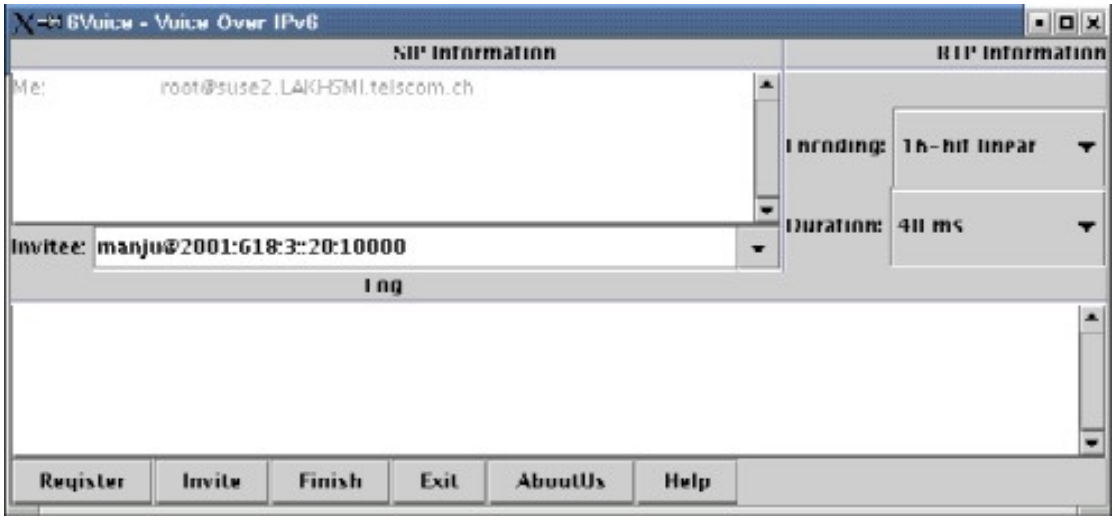

Figure 2-3: 6VOICE Caller interface window

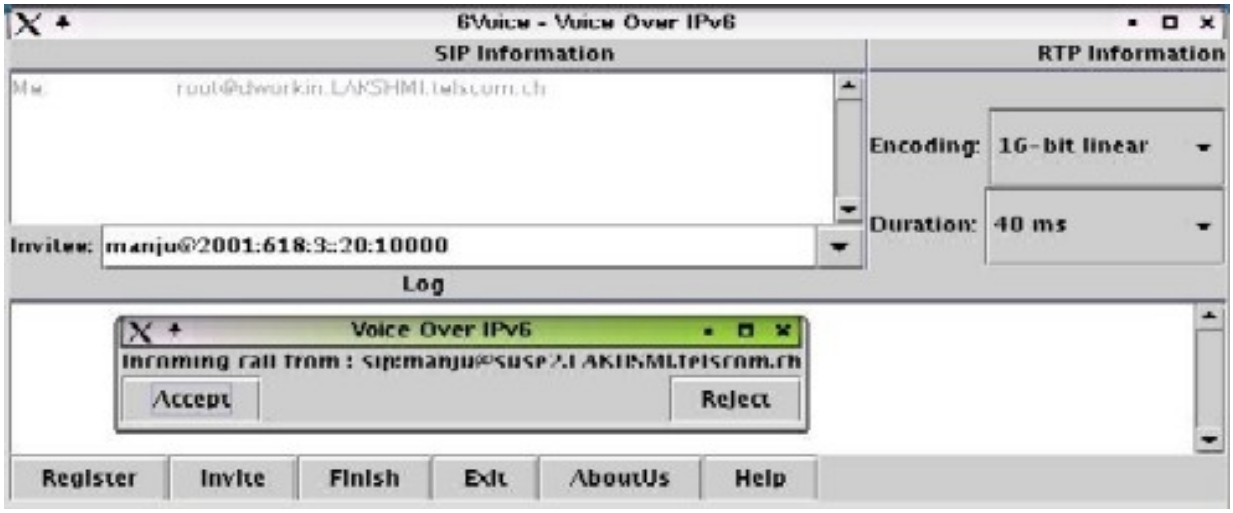

Figure 2-4: 6VOICE Callee interface window

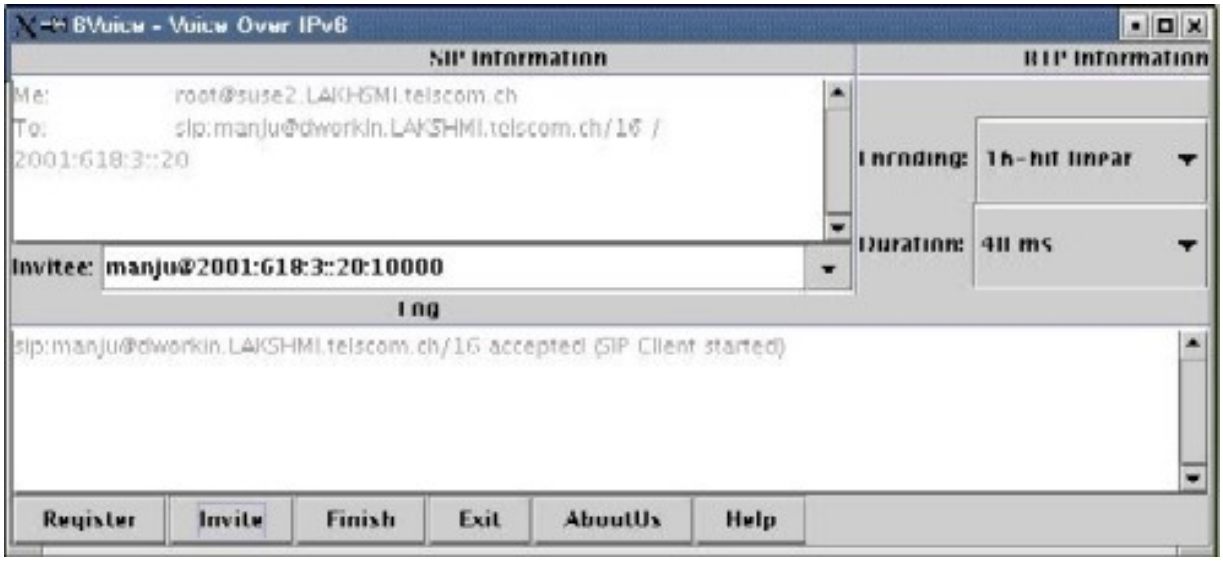

Figure 2-5: 6VOICE interface after establishment of a connection

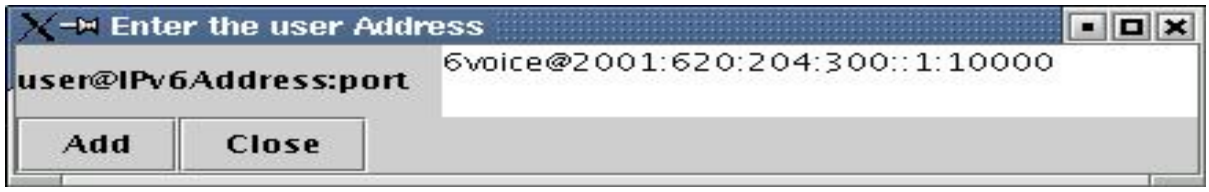

Figure 2-6: 6VOICE GUI used to register user locations

Currently we are testing 6Voice with SIP gateways to check interoperability. Here is the basic test scenario we are planning

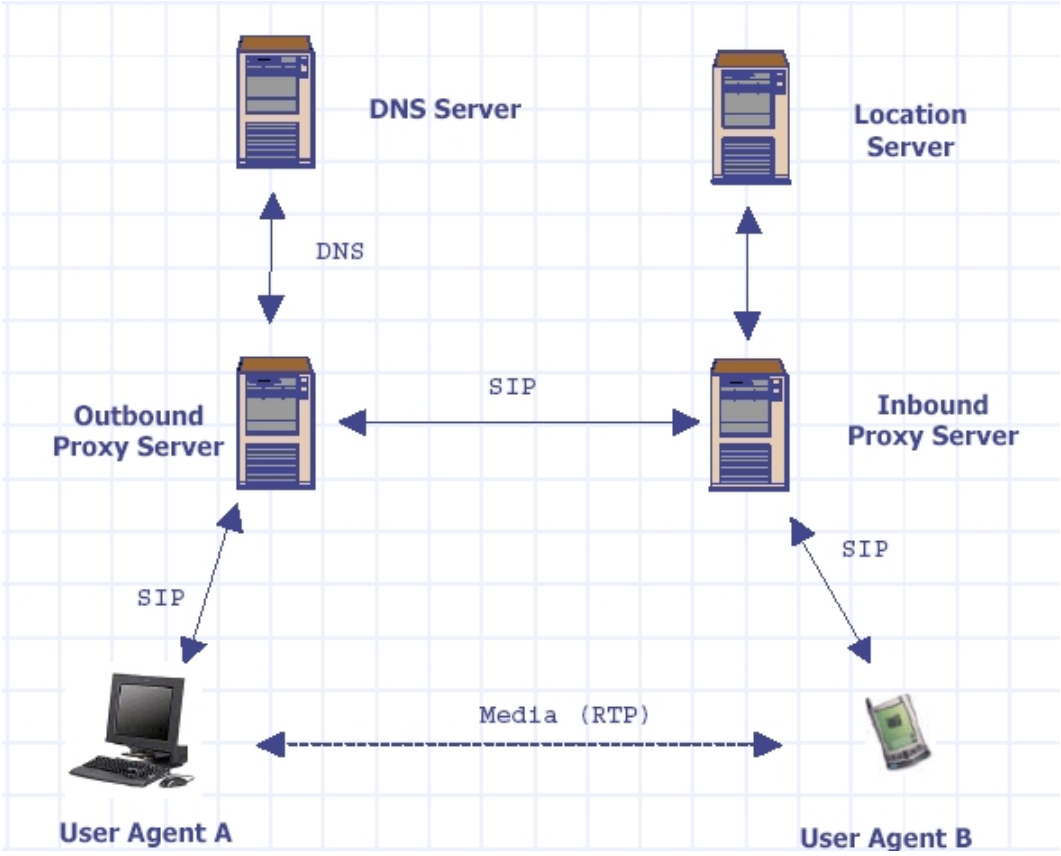

Figure 2-7: 6VOICE test scenario

We have tested 6Voice across Mobile IPv6 network (HUT implementation). Results were satisfactory no much data losses were observed. The test network used for testing across Mobile IPv6 network is as shown in the figure below.

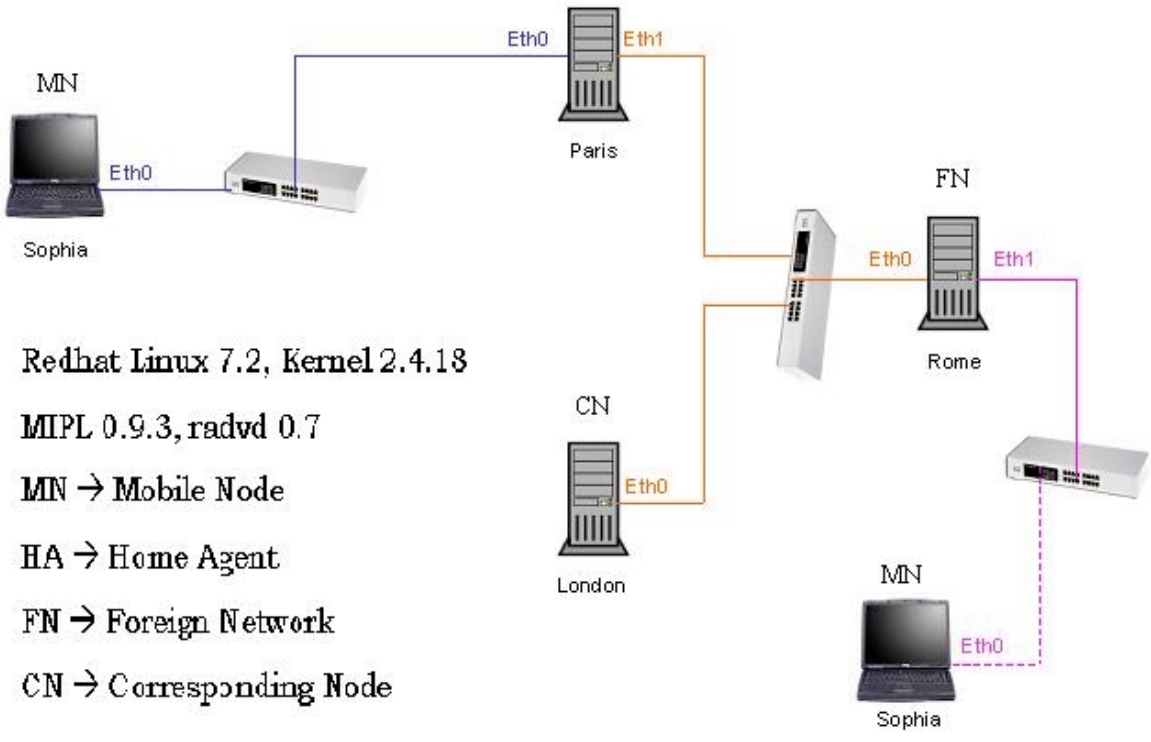

Figure 2-8: 6VOICE Mobile IPv6 test network

2.4 VOCAL

The Vovida Open Communication Application Library (VOCAL) is an open source project targeted at facilitating wider adoption of Voice over IP technology (see http://www.vovida.org). VOCAL was originally developed by Vovida Networks, which has since been acquired by Cisco systems. However the project remains open source.

2.4.1 Vocal Features

As described on the VOCAL web site, VOCAL provides the development community with software and tools needed to build new VoIP features, applications and services. The software in VOCAL includes a SIP based User Agent, Redirect Server, Feature Server, Provisioning Server and Marshal Proxy.

VOCAL will benefit greatly from the additional address space offered by IPv6 especially in where embedded into hardware devices such as VoIP phones. The ability to “plug-and-play” install IPv6 phones, the potential for end-to-end transport mode IPsec (unhindered by NAT devices), and support for enhanced mobility with Mobile IPv6, all combine to make VoIP a potentially very good “peer to peer” demonstrator for IPv6.

2.4.2 Porting VOCAL

The initial IPv6 support for VOCAL has been added by Southampton (UoS). Vovida has been supporting the effort by merging these changes back into the main development tree (under version 1.5.0, which at the time of writing is at Alpha version 2). The initial porting work began under version 1.4.0, but the patch has been officially adopted within the VOCAL project as of version 1.5.0.

The porting effort aims to provide an IP independent version of VOCAL that will work effectively in both IPv4 and IPv6 networks, preferring the latter when available. VOCAL has hundreds of thousands of lines of code, and includes a wide range of supported devices and features. As such, it is an already advanced VoIP application, for which making an IPv6-enabled version available is a useful exercise.

The bulk of the porting work has been done within the Euro6IX project, although testing has been done within 6NET and 6WINIT (the notes on VOCAL porting were first reported in 6WINIT, as a convenient deliverable was being written at the time of the testing). In 6WINIT the SIP gateway/proxy service from TZI (another open source package) was used, which can relay between IPv4 and IPv6 and between SIP and H.323. This gateway has been successfully tested to allow calls between Vocal and BonePhone SIP applications via the gateway. The goal in porting VOCAL has been to enable VoIP communication between members of various IPv6-related projects including 6NET and Euro6IX.

The primary platform for VOCAL is Linux. While other operating systems are supported in part (including Solaris, FreeBSD and Windows 2000), the IPv6 porting work has been concentrated on Linux.

Thus far work has been done by UoS to add IPv6 support to the Sipstack when using UDP. The bulk of the changes have been made to the NetworkAdress class, this has been re-written to use the IP independent functions and structures, namely getnameinfo() and getaddrinfo(). Previously the class stored IP addresses as integers and used IPv4-specific functions to resolve hostnames, etc. Fortunately these two classes encompassed almost all of the low-level network code meaning that it wasn't necessary to change the networking code throughout the rest of VOCAL.

However, a problem which arose lay in the fact that much of the code makes assumptions about the format of IP addresses, in particular the use of a ":" as a port/address separator, which obviously fails with IPv6 addresses. An additional problem was that unlike the network code the parsing is done in many different classes and often in slightly different ways. Locating and updating these sections of code proved to be a time-consuming task.

VOCAL uses SIP and SDP, both of which embed IP addresses within the packets; this is what led to many of the parsing issues mentioned previously. Below is an IPv6 SIP header using the [] notation:

REGISTER sip:[2001:798:80:103:a00:46ff:fe04:4c45]:5060 SIP/2.0
Via: SIP/2.0/UDP [2001:630:d0:111:250:4ff:fef6:2e35]:5060
To: Mike<sip:user@[2001:798:80:103:a00:46ff:fe04:4c45]:5060>
From: Mike<sip:user@[2001:630:d0:111:250:4ff:fef6:2e35]:5060>
Call-ID: 0915fa0682685be52a9dfd9ee24ea40a@2001:630:d0:111:250:4ff:fef6:2e35
CSeq: 1 REGISTER
Max-Forwards: 70
Expires: 60000
Contact:
Content-Length: 0

The SDP classes also had to be updated, in particular the "o line" (session origin and owner's name) had to be updated to accept IPv6 as a net type.

There are interesting issues which arise when the two IP protocols meet, in particular detection of available protocols on both the local and remote machine, which one should be favoured, falling back to the alternative and what behaviour should be taken when an IPv6 only host attempts to call an IPv4-only one. These issues are beginning to be tackled, however unlike the initial porting work there are no hard and fast rules as to the best methods to use; these are to a large extent policy issues.

As a result of testing the VOCAL code within 6WINIT, and at IST2002 (as described later in this section), a number of enhancements and bug fixes have been made to the VOCAL IPv6 code. UoS plans to contribute notes on lessons learnt in porting to the porting cookbooks being produced within 6NET and Euro6IX. Porting effort will continue in the additional feature modules of the VOCAL project. In addition, UCL plans to add rat audio functionality to VOCAL as part of the 6NET project work (by default, VOCAL uses a codec which operates using around 100-150Kbit/s for audio, where rat would be more flexible and potentially more robust). UoS is investigating the potential to make VOCAL compact enough to operate on Linux-based PDA devices (which may include the Compaq iPAQ or Sharp Zaurus).

The VOCAL code has been verified and tested by a number of groups, including TZI, UCL and BT Exact within the 6WINIT project. TZI has verified that VOCAL works successfully with their SIP gateway and with BonePhone. Some testing has also been performed by users on the VOCAL developer community mailing list.

The goal for VOCAL development is to get a user base within 6NET and Euro6IX using the IPv6-enabled code for project communication. This would represent a new collaboration (joint activity) between the projects, expanding upon the ideas for Deliverable 7.9.

If the hardware can be acquired, UoS plans to run tests with ISDN gatewaying of VOCAL calls to regular handsets, e.g. using a Linux PC with ISDN card or, more likely, a suitable router with software and ISDN line card (e.g. a Cisco router).

2.4.3 IST2002

VOCAL was demonstrated at the IST2002 event using Linux laptop devices. These were Sony laptops of the order of 800MHz CPU with 256MB RAM, using built-in sound devices and microphones. RedHat Linux 7.3 was used with the 2.4.18 kernel, modified to use the ALSA sound libraries. VOCAL was run point to point at the IST event over WLAN, and also between the conference site and UoS. Audio quality was good, although the demonstration to UoS was hampered by inefficiencies in the codec at the remote site; lag in the call can be heavily dependent on the hardware being used, often a more important factor than network delay. For some Linux laptops, the speaker and microphone sound volumes can be quite sensitive; the aumix utility allows customisation on the fly. Further tests are ongoing to determine optimal configurations and setup.

2.4.4 ENUM

UoS has joined the UK ENUM pilot, which began in December 2002. The aim of the pilot is to deploy and validate ENUM (RFC2916) via which E.164 telephone numbers can be mapped to DNS space under the e164.arpa domain, e.g.:

+44 23 8059 3257

becomes

7.5.2.3.9.5.0.8.3.2.4.4.e164.arpa

In the DNS record for an E.164 entry, a NAPTR record can indicate services, protocols and addresses to use, e.g. SIP with an endpoint address. VOCAL has ENUM support, so UoS plans to ensure the UK ENUM pilot includes IPv6 DNS and also use IPv6 VOCAL in the pilot tests.

2.5 SIP Over IPv6

SIP and H323 are the standards used for controlling Multimedia sessions over IP. Nevertheless in spite of the first force of H323 SIP is becoming the more used protocol for development of this kind of services on Internet. Nevertheless there is a lot of research around IPv6; just a few of them pay attention of the use of IPv6. The last release of the standard [1] included the possibility of using IPv4 and IPv6 address. This is of course a necessary step, but it is not sufficient for providing good support for the deployment of Sip services at Euro6IX networks.

We identified to main lack at the current state of the art of the SIP services with IPV6 support:

- There is a lack of Sip libraries of commercial product with support for IPv6
- There is not any strategy or methodology for porting SIP

We assume that almost at the beginning of any commercial service deployment, many services will have users and/or services in both types of networks, that is IPv4 and IPv6, it is not enough to provide IPv6 support for Sip, but we also need that IPV4 and IPV6 version may interwork with in the most seamless way, so the user don't have to care about the type of network he is or the type of network the server or many intermediate /auxiliary component is.

2.5.1 Objectives

The objectives of this work may be classifies into two main groups:

1. Provision of a SIP library IPv6 enabled, which allows the creation of mayor SIP elements (User Agent and Proxies).
2. Development of a mechanism for interworking between IPv4 and IPV6 SIP components.

The first objective includes the choice of creating a new SIP library from scratch or reusing one existing one. In our case we decide to start from one existing library jSIP (http://jsip.sourceforge.net), wich will offer several advantages: we do not have to start the design from the very beginning

We may study de advantages and inconveniences of porting existing libraries, which have not been designed for porting to IPv6.

The second requirements derived in the creation of a library that enabled for working both over IPv4 and IPv6 will provide a transparent session negotiation service over SIP independently of the network where the Client and the server are located. Event this transparency may be also applied to auxiliary services requires which may be accessed by the use of SIP proxies implemented using the same library.

2.5.2 Technical Approach

In this work, the first step was the selection of the base library for implementing SIP elements. We decide to review existing applications that may be used as basic for porting SIP service to IPv6. This approach will offers several advantages:

- The results may be applies to other middleware and service infrastructure
- We do not plan to create a new SIP library, but to study and analyze the problem of porting to IPv6

After analyzing several alternatives, we finally choose jSIP.

2.5.2.1 SIP Overview

SIP is a lightweight, transport independent, text-based protocol that is used for multi-media call control and enhanced telephony services. SIP is lightweight in that it has only six different method types. These methods, when combined together, allow for complete control over a multi-media call session while limiting complexity. SIP is transport-layer independent because it can be used with any datagram or stream protocol (UDP, TCP, ATM, …). SIP is text-based in that a method is formed via a textual header that has fields that contain call properties. This text-based approach is easy to parse, thin in terms of packet overhead and extremely flexible.

SIP clients, typically called user agents, communicate with SIP servers in a client-server fashion. User agents also act as servers when the SIP request reaches its final destination. These user agents contain the full SIP state machine and can be used without intermediate servers.

There are two components within SIP. The SIP User Agent and the SIP Network Server. The User Agent is effectively the end system component for the call and the SIP Server is the network device that handles the signaling associated with multiple calls.

The User agent itself has a client element, the User Agent Client (UAC) and a server element, the User Agent Server (UAS.) The client element initiates the calls and the server element answers the calls. This allows peer-to-peer calls to be made using a client-server protocol.

There are effectively three forms of SIP Network Server that can exist in the network - the SIP stateful proxy server, the SIP stateless proxy server and the SIP re-direct server. The main function of the SIP servers is to provide name resolution and user location, since the caller is unlikely to know the IP address or host name of the called party. What will be available is perhaps an email-like address or a telephone number associated with the called party. Using this information, the caller"s user agent can identify with a specific server to "resolve" the address information - it is likely that this will involve many servers in the network.

A SIP proxy server receives requests, determines where to send these, and passes them onto the next server (using next hop routing principals). There can be many server hops in the network.

The difference between a stateful and stateless proxy server is that a stateful proxy server remembers the incoming requests it receives, along with the responses it sends back and the outgoing requests it sends on. A stateless proxy server forgets all information once it has sent on a request. This allows a stateful proxy server to fork requests to try multiple possible user locations in parallel and only send the best responses back. Stateless Proxy servers are most likely to be the fast, backbone of the SIP infrastructure. Stateful proxy servers are then most likely to be the local devices close to the User Agents, controlling domains of users and becoming the prime platform for the application services.

A re-direct server receives requests, but rather than passing these onto the next server it sends a response to the caller indicating the address for the called user. This provides the address for the caller to contact the called party at the next server directly.

2.5.2.2 The jSIP Library

jSIP [2] looks to implement the Session Initiation Protocol, a text-based protocol for initiating collaborative sessions between users. SIP has been designated by the IETF as an Internet protocol, with its specifications in RFC 2543 and RFC 3261. In addition to building the SIP protocol as specified, jSIP will look to provide prototype implementations of some of the SIP extensions.

JSIP provides a library for building SIP elements. Another interesting feature of jSIP is that it is implemented using Java, which offers several key advantages for IPv6 porting:

- Using Java, most of the porting issues related with sockets are solved since JDK1.4 release.
- It is possible to use IPv4 or IPv6 network support without changing the basic Java code
- Java is portable and does not require any change from Linux to Windows.

Nevertheless this still offers some drawback, specially related to the support for Windows were there is not still IPv6 support.

Key point in favor of jSIP may be summarized as follows:

- **Availability:** the current jSIP library is OpenSource code, which may be used and modified according to our purposes, without being limited by any company.
- **Independency:** jSIP is not part of any applications or system, but a library intended for creating SIP elements, which fit very well with our purposes.
- **Portability:** The use of Java provided high-level portability between machines architectures.
- **Quality of the code:** this has been indeed the weakest point of our choice. As many other OpenSource programs, jSIP does not implements all the elements of the standards, and testing of the code has not been so deep as desired. So our first step was the completion of the key features, and the provision of a Test suite for testing the libra4y.

2.5.3 System Architecture

SIP's basic architecture is client/server in nature. The main entities in SIP are the User Agent, the SIP Proxy Server, the SIP Redirect Server and the Registrar.

The User Agents, or SIP endpoints, function as clients (UACs) when initiating requests and as servers (UASs) when responding to requests. User Agents communicate with other User Agents directly or via an intermediate server. The User Agent also stores and manages call states.

SIP intermediate servers have the capability to behave as proxy or redirect servers. SIP Proxy Servers forward requests from the User Agent to the next SIP server, User Agent within the network and also retain information for billing/accounting purposes. SIP Redirect Servers respond to client requests and inform them of the requested server's address. Numerous hops can take place until reaching the final destination. SIP's tremendous flexibility allows the servers to contact external location servers to determine user or routing policies, and therefore, does not bind the user into only one scheme to locate users. In addition, to maintain scalability, the SIP servers can either maintain state information or forward requests in a stateless fashion.

The third entity that comprises SIP is the SIP Registrar. The User Agent sends a registration message to the SIP Registrar and the Registrar stores the registration information in a location service via a non-SIP protocol. Once the information is stored, the Registrar sends the appropriate response back to the user agent.

2.5.3.1 Design Criteria:

The criteria have been applied to the design of the IPV6 enables Sip library:

- To keep separate the logic of SIP from any other issues, specifically the presentation and network issues
- To ensure the independency of the SIP library with respect to the applications that will use them.
- To reduce the modification over jSIP library to the minimum indispensable.
- To take into account future enhancements, so modifications on the classes structure will be done when required.
- To provide support for both IPv4 and IPV6 using the same library.

2.5.3.2 Enhancements and modification over the initial library

Starting to work over the 0.6 version of JSIP, we change to 0.8 because it was a more mature implementation. It was created a Test Suite for testing both the original version of jSIP and our enhanced versions. Minor modification and bug corrections were required for SIP function, but the support for SDP have to been heavily enhanced, because of the pour initial support for this important part of the SIP negotiation support.

About 10 mayor modification, including enhancement of DP support have to be performed, which involves around 10% of the total amount of the initial code.

2.5.3.3 Porting o IPv6

Once we have a good and complete library code, we pass to the main phase: porting the code to IPv6. The main modifications required for adapting jSIP to IPv6 were:

- The initial data structure used for storing address was just one String. We created the class IpAddress, which is used for deriving IPAddress4 and IPAddress6 for storing IPV$ and IPV6 addresses.
- Other classes that deal with IP address such as SipVia have to be also modified creating classes structures similar to the one created for IpAddress.

- The class SdpMessage, which stored IP addresses, has to be modified in order to allow it to manage IPv4 and IPV6 IP addresses.
- Special care required the modification of the classes related with the parsing of Sip and SDP messages in order to identify IPv6 address in addition ot the original IPv4 addresses.

We may summarize the main modifications performed over the original jSIP library:

- Enhancement of the jSIP code
- Enhancement of “Route” and “Record-Route” headers.
- Provision of support for IPv6
- Enhancing the SDP capabilities, including support for IPv6

2.5.3.4 Implementing SIP elements

In order to evaluate the use of the library we build the next SIP elements:

- **UA:** we created a complete SIP User Agent, which may be used by any application for negotiationg a session over SIP. May characteristics of this UA I that it provides transparent SIP support independently of the underlying network (IPv4 or IPv6). Of course the Applications has to be able to mange IPv4 or IPv6 address depending of the network where it is running, because at the end the application who decided which ports and applications to use for interchanging multimedia information.

- **SIP Proxy:** SIP proxies are key elements in the deployment of Sip services, so we created a stateless Sip proxy using the library.

2.5.3.5 Evaluation and Testing

For testing the library we provided a test Suite implemented using JUNIT framework (http://www.junit.org), with Test Cases for several realistis scenarios. JUNIT is also implemented in Java, so our test Suite is as portable as the SIP library.

Next Figure describes the internal structure of Junit which uses key facilitate sof Object Oriented Programming.

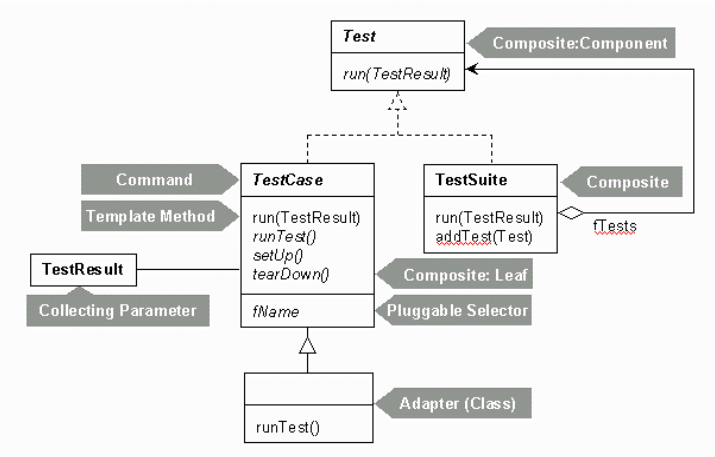

Figure 2-9: JUNIT structure

Finally we created one scenario with a simple application in order to show the behavior of a complete system.

Test Cases creation has been base on the scenarios proposed by Columbia University (http://www.cs.columbia.edu/~hgs/sip/sipit/) and other specific documents describing SIP (RFCs and drafts) which describes examples and typical usage cases.

The final objective of this work is to complete the scenarios of the next figure, where two applications, typically videoconferencing over RTP may use Sip for negotiating a session. A key added value of our approach is that it will allow the interworking of IPv4 and IPv6 applications without any modification of the java based SIP implementations. Further work is still necessary in order to solve interworking for the application itself.

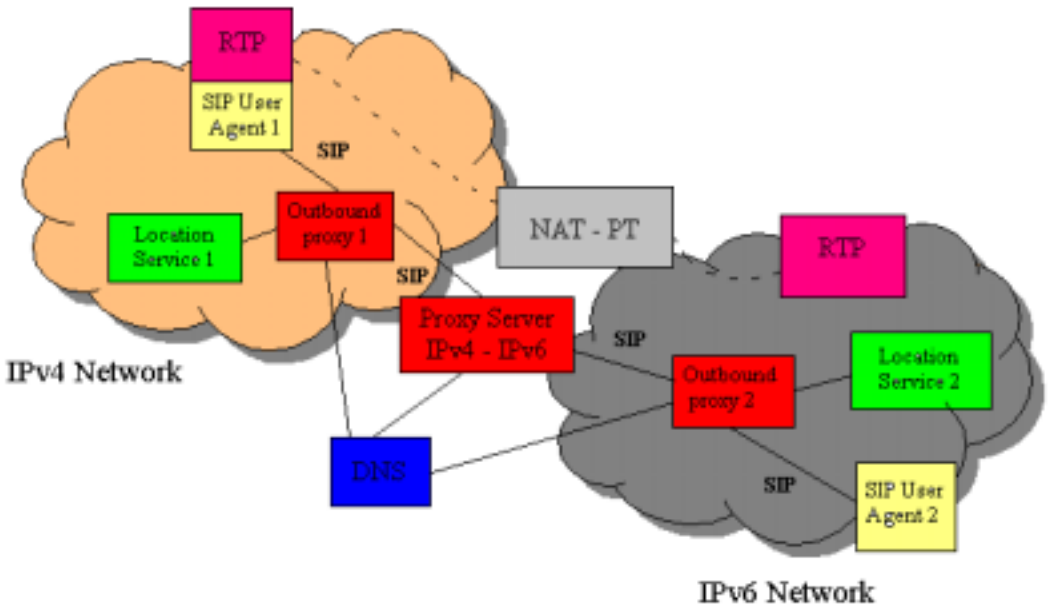

Figure 2-10: Validation Scenario

2.6 Conclusions and Future Work

This activity explored the migration of Audioconferencing applications to IPv6. Due to the importance of session negotiation in this framework, SIP porting has been also deeply investigated.

Two applications and one general purpose SIP library have been ported to IPv6. The combination of these experiments provides the basic for building more complex scenarios during the second year of the projects, and basic pieces developed during this first year are ready for being used in the global experiment of Euro6IX.

Porting Audioconference applications involved not only working with basic communications and multimedia code, but also some key standards such as SDP, SIP and ENUM are involved. Those services may be used, and in deed they will be mandatory, for many other services that will be deployed in future IPv6 networks.

Nevertheless the porting activities have not being completed several lessons have been learned, which will be fed into the project “porting” guide document.

3. IPv6 CAPABLE ISABEL

3.1 Objectives

This subactivity focuses on the evolution of ISABEL application to support and make use of the IPv6 features, such as security, QoS, mobility, etc, that will be offered by the Integrated Service Framework created over the Euro6IX network.

During the first year, work has concentrated on adding security capabilities to ISABEL, although some preliminary studies about ISABEL over mobile scenarios have also been made.

Related to security, this year objectives were:

- To initiate a general study about how to adapt ISABEL collaboration tool to allow secure communications over the IPv6 security framework being developed on the context of A4.1 activity.

- To test and demonstrate ISABEL over IPsec scenarios based on IKE network services using pre-shared keys, as well as to evaluate its behavior and performance.

- To integrate the set-up of IPsec tunnels inside the ISABEL general set-up procedure.

Related to mobility, this year objectives were:

- To study and test ISABEL application behavior in a simple mobile scenario where a mobile node participates in an ISABEL session. In particular, the study will concentrate on application behavior and performance during handovers.

- To define the requirements for Correspondent Nodes and central nodes like Flow Servers and MASTER node.

3.2 Security and Mobility for ISABEL

Security is a key service to be offered by the Integrated Services Framework of Euro6IX network. As described before, several experiences related to security have been initiated in the context of A4.1. For example, IPsec based VPNs have been experimented, ensuring confidentiality, integrity, and authenticity of data communications across public networks, such as the Internet. IPsec, which is mandatory for IPv6, is the framework of open standards for ensuring secure private communications over IP networks.

However, there is a lack of security aware applications, especially when speaking about multimedia applications, that makes difficult to experiment and evaluate security services in real scenarios.

The ISABEL CSCW application is a group communication tool for the Internet, based on advanced videoconferencing features. ISABEL allows efficient organization of working procedures over the Internet in large enterprises or groups. ISABEL has been ported to IPv6 on the context of IST LONG project.

However, ISABEL, as many other multimedia applications, lacks security support in its data or control communications. That is to say, communications are not protected in any way and therefore no confidentiality, integrity, or authenticity is offered as an application service.

Being ISABEL a complex application, involving multiple multimedia and data control flows in a great variety of possible topologies, it is a very good candidate to be used to test security services in Euro6IX project, apart from the added value of developing a secure ISABEL application.

For all the above reasons, the main objective of this subactivity is the adaptation of ISABEL to take advantage of network security services, in order to offer secure communications to the user. For the first year, simple configurations and scenarios have been used, mainly based on IPsec and IKE using pre-shared keys using a star interconnection topology.

For that purpose, new security parameters have been added to ISABEL sessions, being the main one the pre-shared key that is known by all participants. The MCU and master terminal, located at the central point in the star configuration, create each one an IPsec tunnel for each participant, therefore, each terminal maintains two secure tunnels. Other security parameters can be configured with default values, for example 3DES for data encryption, HMAC-SHA1 for data integrity, and other values for IKE parameters.

Besides, work has been invested in studying the behavior of an IPv6 Mobile Node connected to an ISABEL Session, focusing on how to maintain the communication during handovers. As in the case of security, only simple star based scenarios have been studied. Due to the network architecture of ISABEL, an ISABEL Node does not require connectivity with all the other participant nodes, only with a central control node called the “Master” and a “FlowServer”, which acts as data distribution center. So, a mobile node participating in an ISABEL Session would keep TCP and UDP communications with the Master and Flowserver (MCU) systems. Consequently, at least these two systems, apart from the mobile node itselft, will need basic Mobile IPv6 support when simple star interconnection topology is used. For more complex scenarios more systems will need to support mobility.

ISABEL does not require special modifications for supporting IPv6 Mobility, since it is an IPv6 enabled application. In principle, the mobility is transparent for ISABEL.

However, the main problem is the loss of connectivity the mobile node experiments during handovers. The loss has typically a variable duration around few seconds, but it can take minutes sometimes. Apart from the effect on multimedia data (audio and video), if the handover takes a long time, the effect is very problematic on the TCP control connections between the Mobile Node and the ISABEL master, as the mobile node can be even disconnected from the session. So it is critical to have a bounded handover time and adjust application timers according to it.

A more detailed description of the experiments made to evaluate ISABEL behavior in mobile scenarios can be found in the Technical Report [3] that describes the detailed results of Mobility subactivity of A4.1.

3.3 First Year Results

Secure ISABEL sessions using IPsec have been successfully demonstrated in a trial (documented in deliverable D4.3 [4]) using FreeS/WAN implementation. A specific tool, named *isabelvpncfg*, has been developed to take care of the IPsec configuration, determining the IKE parameters and inserting the pre-shared key in the system.

This tool has to be used manually from the session terminals to configure the tunnels before the session is up. Once the tunnels are established, all ISABEL sessions started use IPsec secure data channels to communicate with the master and flowserver nodes. This tool is being integrated at present inside ISABEL application to automate the process.

As mentioned before, in the scheme designed the responsibility of initiating the secure communications resides on the terminals, which are in charge of establishing the secure communications to the flowserver and to the session master. The designed scheme has been tested locally in local network testbeds and also between two Euro6IX partners (UMU and UPM). The application behavior using secure communications was very similar as it is over traditional insecure scenarios. Therefore, no performance problems have been observed when powerful workstation is used.

The main problems experimented were related to the simultaneous configuration of IPsec services and drivers for audio and video boards. FreeS/Wan IPsec services require a specific Linux kernel version together with specific modules loaded that conflict with driver's requirements. That made the workstation installation very difficult or almost impossible when using some Linux distributions. However, this problem it is expected to disappear when IPsec is included as standard module in Linux distributions (late tests made with distributions available at the end of 2002 showed some simplifications in this aspect).

Related to mobility, the preliminary experiments made have allowed to asses the correct behavior of ISABEL application over simple mobile scenarios based on start topology configurations. As experimented, after each handover, the mobile node stops receiving traffic for few seconds, but it keeps connected to the session, and goes on receiving traffic once the handover has succeeded. Handover times measured over WLAN networks were not long enough to break the TCP control between the mobile node and the ISABEL master.

In summary, the behavior of the application was basically as expected: the UDP data traffic was lost during handover, therefore video and audio signals experimented interruptions, but TCP control connections kept alive.

3.4 Conclusions and Future Work

ISABEL has been successfully secured when working over star interconnection topologies. However, ISABEL is able to work over much more complex scenarios based on trees and multicast topologies. Tree topologies allow the interconnection of any number of ISABEL nodes using Flow Servers as intermediate nodes. Besides, ISABEL can interconnect all or part of participant nodes using multicast network services. Therefore, there are two pending tasks.

The first task is the adaptation to a general tree interconnection topology, defining how to distribute shared keys or designing a new authentication procedure. There are two possible solutions. The first solution is to try to integrate the tunnel creation and set up within the ISABEL EDL language -used to define the sessions, participants and other general characteristics-. The extension of this description language will be necessary in order to include the specification of the security parameters. The second solution is to progress in the use of digital certificates X.509 within the VPN security services as an alternative to the use of pre-shared keys. The second task is the study definition of secure communication environment over multicast networks.

Related to mobility, future tests will concentrate on more complex ISABEL scenarios, where, for example, the main nodes, a FlowServer or even a Master, are also mobile nodes. In this case, all participants will need IP mobile support, as all of them communicate directly with the central nodes. Besides, deeper investigations could be made in order to reduce handover effect, for example, giving priority to audio flows over video.

4. IPV6 UNIFIED INSTANT MESSAGING SYSTEMS

This section summaries the research and implementation work on the IPv6 Unified Instant Messaging activity for the first year of the Euro6IX project in the context of activities A4.2-4 and A4.2-8 (which have now been merged to one activity of two collaborating developments). The document is organized as follows:

Section 4.1 introduces the basic idea of Instant Messaging and the peer-to-peer (P2P) systems that have become more and more popular for the basis of applications running over the Internet. Then, the IPv4 limitations and the IPv6 advantages for P2P and IM are given.

As interoperability is currently the key issue in IM systems, section 4.2 presents the main IETF and other IM related standards groups, which include Instant Messaging and Presence Protocol (IMPP) Working Group, Jabber Software Foundation, SIMPLE Working Group, and Wireless Village.

In order to implement an interoperable IPv6 based IM system and extend it to support more general peer-to-peer applications, the IPv6 Unified Instant Messaging system is proposed in section 4.3. UoS and nGn will cooperate to develop such a system.

Section 4.4 provides the main results of the research and development up to now and section 5 gives the conclusion.

In section 4.6, the future work is outlined which constitutes the main work which should be done next towards fulfilling the project target.

4.1 Introduction

With the evolution of the Internet, more and more applications are available for users. Instant Messaging (IM or IMing) is, among others, one of the most popular tools for people to communicate with peers over the Internet. The basic idea behind Instant Messaging is the ability to easily find, retrieve, and subscribe to changes in the presence information (e.g. "online" or "offline") of other users, and to send small, simple messages that are delivered immediately to online users. Instant messaging differs from ordinary messaging such as e-mail in that it enables the immediacy of the message exchange and also enables a continued exchange, which is much simpler than sending e-mail back and forth. Compared to the classic UNIX *talk* application and IRC (Internet Relay Chat) system, the innovation in today's IM system is the packaging of these systems into a managed messaging platform. Some most popular Instant Messaging systems include AOL Instant Messenger, ICQ, Windows Messenger and Yahoo Messenger, etc. Originally, the messages were only text based, and instant messages were delivered among users over the wired Internet. Nowadays, multimedia messages have come into being which are the combination of the new forms of rich content, such as audio and video clips, photographs, and images, with text messaging, and the messaging also happens over wireless networks, such as SMS, or wireless Internet.

An Instant Messaging system provides the typical P2P (Peer-to-Peer) services for users. In addition, with the pervasive deployment of Internet and computers, P2P is nowadays increasingly receiving attention in research, product development, and investment circles.

For example, Napster and Gnutella have been developed to provide file sharing over Internet; Distributed Computing or the Grid is another popular P2P applications; The JXTA project, initiated by SUN, is to provide a collaboration platform that supports a wide range of distributed computing applications; and further more, Web Services has become a new P2P service which aims at creating a single global electronic marketplace where enterprises of any size and in any geographical location can meet and conduct business with each other through the exchange of XML based messages. In all these P2P systems, messaging is the main mechanism for the information delivery between peers. So, messaging between individuals is not the only benefit IM can provide. IM can be extended to support more general P2P applications, such as business-to-business (B2B) exchanges. Products like IBM MQSeries, Microsoft MSMQ, TIBCO Rendezvous Open Horizon Ambrosia, etc. have been in enterprise use for years. These kinds of messages can be defined as: asynchronous requests, reports or events that are consumed by enterprise applications, not humans. They contain vital information needed to coordinate these systems. In addition, they contain precisely formatted data that describe specific business actions. Similarly we can also expect more device-to-device communication to be required.

Conventional Instant Messengers use a central server to exchange messages between peers. However, for some other P2P applications, such as Gnutella and JXTA, the decentralized model is preferred. Decentralized P2P model features ad-hoc behaviour, improved cost of ownership, anonymity, and scalability while client/server model features simplicity, security and the ability to control the data. So, there exist different kinds of messaging model: client/server model, pure decentralized model, and the hybrid model.

4.1.1 IPv4 (NAT) Limitations for P2P and IM

It could be concluded that in the near future, more and more users and devices, especially the 3G mobile handsets and the pervasive devices with embedded CPUs, will be using IP addresses to be connected to the Internet and to send/receive messages to/from peers. In this case, a large amount of globally routable IP addresses is needed. However, the existing messaging systems are built over IPv4. The limited IPv4 address space will be the bottleneck of the future P2P messaging system. The Network Address Translation (NAT) scheme can enable more IPv4 hosts to exist on the Internet by masquerading multiple hosts behind a pool of IP addresses. However, it is not a good solution for P2P systems because of its poor scalability, and the crucial fact that it allows connectivity out of a NAT network, but not (so readily) into it. NAT doesn't work for those always-on devices. As a result, most P2P applications that run behind an IPv4 NAT have to communicate via a common server, or use some complex tricks that make application and service development overly complex and potentially less robust.

Also, if we wish to use end-to-end transport mode IPsec between two P2P hosts, IPv4 NAT will prevent a direct security association being established.

4.1.2 IPv6 Advantages

IPv6, the next generation Internet, provides a huge address space that allows each individual device to have its own globally routable IP address. This is the key benefit, and one critical for new generations of P2P and IM applications. In addition, it provides several advanced services. Mobile IPv6 improves upon Mobile IP(v4) by offering standard features such as route optimisation, while removing the requirement for foreign agents (routers) to be deployed.

Using MIPv6 enables roaming access and allows communicating devices to find an individual node anywhere in the world and route traffic to it. In a full implementation of the IPv6 specification (as per RFC2460), support for encryption and authentication of each packet comes as standard, which helps ensure privacy and proof of identity in data exchanges. Thus the research and development of Instant Messaging over IPv6 is of great importance (which we call "Unified Instant Messaging", referring to a convergence of IM services over IPv6). "Plug and play" of IPv6 devices on a network, as offered by IPv6 stateless autoconfiguration, is also an important feature for ad-hoc connectivity for IM devices.

4.2 IETF and other IM related standards

Up to now, there have been several working groups and bodies which focus on Instant Messaging. Here we discuss three of the main working groups and bodies: the IETF IMPP working group, the Jabber software foundation, and the IETF SIMPLE working group.

4.2.1 Instant Messaging and Presence Protocol (IMPP) Working Group

As applications of presence and instant messaging currently use independent, non-standard and non-interoperable protocols developed by various vendors, the goal of the Instant Messaging and Presence Protocol (IMPP) Working Group is to define a standard protocol so that independently developed applications of instant messaging and/or presence can interoperate across the Internet.

In February 2000, IMPP working group published two RFCs: *Model for Presence and Instant Messaging (RFC 2778)* and *Instant Messaging / Presence Protocol Requirements (RFC2779).*

A Model for Presence and Instant Messaging (RFC 2778) defines an abstract model for a presence and instant messaging system. It defines the various entities involved, defines terminology, and outlines the services provided by the system. The goal is to provide a common vocabulary for further work on requirements for protocols and markup for presence and instant messaging.

Instant Messaging / Presence Protocol Requirements (RFC2779) defines a minimal set of requirements that IMPP must meet.

Figure 4-1 and Figure 4-2 respectively illustrate the models for presence system and messaging system proposed by IMPP working group.

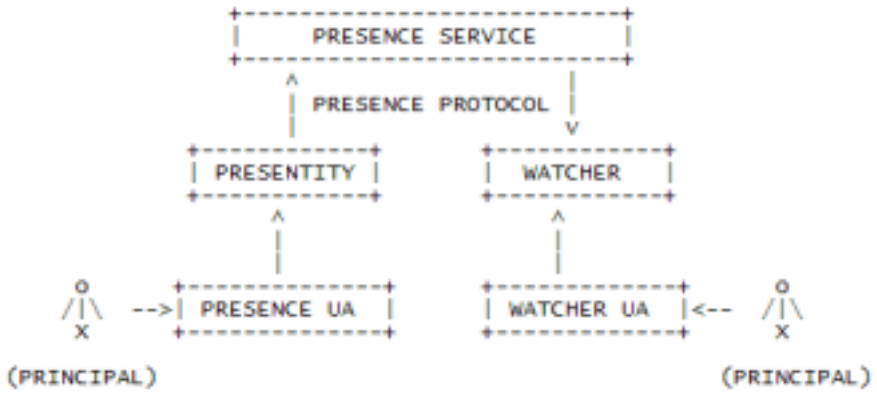

Figure 4-1: A Model for Presence System

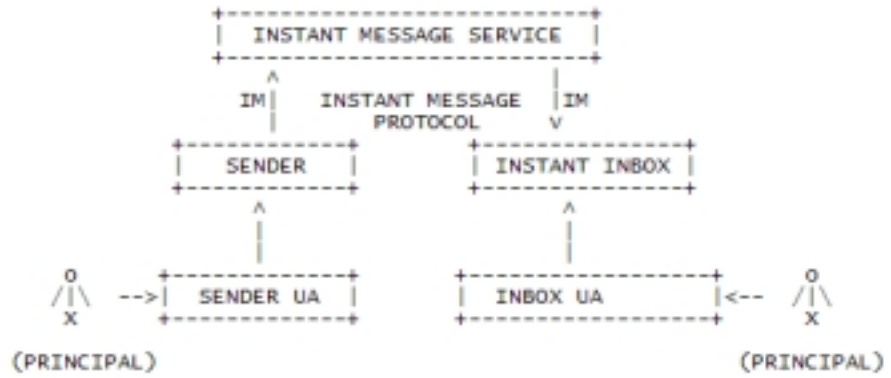

Figure 4-2: A Model for Messaging System

In August 2002, the IMPP working group submitted the latest version of its Internet-Draft: *The Common Profile for Instant Messaging (CPIM <'draft-ietf-impp-cpim-03'>).* This specification defines a number of operations to be supported and criteria to be satisfied for interworking diverse instant messaging protocols. It expires in February 2003.

In addition, *CPIM: Message Format (<draft-ietf-impp-cpim-msgfmt-07.txt>),* submitted in October 2002, defines the mime type 'Message/CPIM', a message format for protocols that conform to the CPIM specification. It expires in April 2003; *CPIM Presence Information Data Format (<draft-ietf-impp-cpim-pidf-06.txt>),* submitted in October 2002, specifies the CPIM Presence Information Data Format (PIDF) as a common presence data format for CPIM-compliant Instant Messaging and Presence protocols, and also defines a new media type "application/cpim-pidf+xml" to represent the XML MIME entity for PIDF. It also expires in April 2003.

A method for locating the instant inbox and presentity resources associated with URIs that employ these schemes via the Domain Name Service has recently been presented in *<draft-ietf-impp-srv-01>*, updated in December 2002. Common profiles for Presence *(<draft-ietf-impp-pres-01>*) and for Instant Messaging *(<draft-ietf-impp-im-01>)* have also been defined and revised in December 2002. These are both designed to help address the interoperability problems of current IM systems.

4.2.2 Jabber Software Foundation

Jabber is an open, XML-based protocol for instant messaging and presence. The Jabber Software Foundation (JSF) is a not-for-profit membership organization that helps the Jabber community foster freedom of conversation among people, applications, and systems, regardless of medium, through the continued development of an open protocol for XML-based communications.

JSF has made several protocol submissions to the Internet Engineering Task Force (IETF). The first of Jabber (*draft-miller-jabber-00*) was submitted in February 2002. In June 2002, the JSF submitted three interrelated Internet-Drafts that superseded the earlier submission: *XMPP Core (draft-miller-xmpp-core-02), XMPP Instant Messaging (draft-miller-xmpp-im-02), and XMPP CPIM Mapping (draft-miller-xmpp-cpim-00).*

XMPP Core (draft-miller-xmpp-core-02) describes the core features of the eXtensible Messaging and Presence Protocol (XMPP), which is used by numerous applications that are compatible with the open-source Jabber instant messaging system. Figure 4-3 shows the generalized architecture of XMPP, where "-" represents communications that use XMPP and "=" represents communications that use any other protocol.

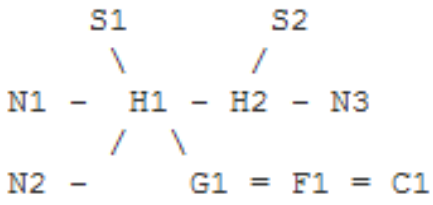

Figure 4-3: A Generalized Architecture of XMPP

The symbols are as follows:

- N1, N2, N3 - Nodes on the Jabber network
- H1, H2 - Hosts on the Jabber network
- S1, S2 - Services that add functionality to a primary host
- G1 - A gateway that translates between XMPP and the protocol(s) used on a foreign messaging network
- F1 - A foreign messaging network
- C1 - A client on a foreign messaging network

XMPP Instant Messaging (draft-miller-xmpp-im-02) describes the specific extensions to and applications of the XMPP that are necessary to create a basic instant messaging and presence application (specifically, an application that is compatible with the open-source Jabber instant messaging system).

XMPP CPIM Mapping (draft-miller-xmpp-cpim-00) describes a mapping of the XMPP to the Common Presence and Instant Messaging specification (CPIM).

4.2.3 SIMPLE Working Group

The SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE) working group focuses on the application of the *Session Initiation Protocol (SIP, RFC 3261)* to the instant messaging and presence (IMP).

The *Session Initiation Protocol (SIP)* is an application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants. These sessions include Internet telephone calls, multimedia distribution, and multimedia conferences. SIP invitations used to create sessions carry session descriptions that allow participants to agree on a set of compatible media types. SIP makes use of elements called proxy servers to help route requests to the user's current location, authenticate and authorize users for services, implement provider call-routing policies, and provide features to users. SIP also provides a registration function that allows users to upload their current locations for use by proxy servers. SIP runs on top of several different transport protocols. The first RFC on SIP is RFC 2543 which is published in March, 1999. In June 2002, RFC 2543 is replaced by RFC 3261.

As the most common services for which SIP is used share quite a bit in common with IMP outlined in RFC 2779 (including the security and privacy requirements there) and in the Common Presence and Instant Messaging (CPIM) specification, developed within the IMPP working group, the adaptation of SIP to IMP seems a natural choice given the widespread support for (and relative maturity of) the SIP standard.

Session Initiation Protocol (SIP) Extensions for Presence (draft-ietf-simple-presence-09.txt), submitted in December 2002 and expiring in June 2003, proposes the usage of the SIP for presence. This is accomplished through a concrete instantiation of the general event notification framework defined for SIP, and as such, makes use of the SUBSCRIBE and NOTIFY methods defined there. Specifically, this document defines an event package, as described in SIP-specific event notification.

An Extensible Markup Language (XML) Based Format for Watcher Information (draft-ietf-simple-winfo-format-03.txt), submitted in December 2002 and expiring in June 2003, describes an XML document format for the fairly complex state of watchers on a resource.

A Session Initiation Protocol (SIP) Event Template-Package for Watcher Information (draft-ietf-simple-winfo-package-04.txt), submitted in December 2002 and expiring in June 2003, defines the watcher information template-package for the SIP event framework.

CPIM Mapping of SIMPLE Presence and Instant Messaging (draft-ietf-simple-cpim-mapping-01), submitted in June 2002 and expires in December 2002, describes how those mechanisms proposed before (SIP events package for distribution of presence information, MESSAGE extension for the transport of instant messages, etc) map to the abstract CPIM service, in order to interoperate with other CPIM compliant presence and instant messaging services.

A SIP Event Package for List Presence (draft-ietf-simple-presencelist-package-00.txt), submitted in June 2002 and expires in December 2002, presents a SIP event package for subscribing to a list of presentities. Instead of the subscriber sending a SUBSCRIBE to each presentity individually, the subscriber can subscribe to their presence list as a whole, and then receive notifications when the state of any of the presentities on the list changes.

Furthermore, the 3rd Generation Partnership Project (3GPP) is using SIP RFC 3261 as the session establishment protocol for the 3GPP IP Multimedia Core Network Subsystem (IM CN Subsystem, IMS). 3GPP has recently started working on messaging over the IM CN Subsystem. It is intended that this will utilize the IM CN Subsystem for delivery and may be based on IETF protocols including the work being performed in the SIMPLE working group.

As an aside, UoS has recently been active in leading the porting of the Vocal SIP-based VoIP package (see www.vovida.org) to support IPv6 communications. We plan to use this tool within the Euro6IX project.

4.2.4 Wireless Village

Today's wireless landscape is rapidly changing as mobile phones and networks are being enhanced to provide services beyond just voice services. The wireless industry is now seeing the rapid expansion of mobile data services. This expansion is being fueled by a variety of factors:

- Internet and wireless domains are converging
- Tremendous adoption rates of SMS and its lucrative business model

- Mobile consumers and professionals are asking for new wireless applications
- Operators need to leverage their investment in 3G spectrums
- Operators are extending their brand to consumers via portals and new services

Chief among the technologies consumers are asking for is mobile instant messaging and presence services (IMPS). One of the challenges in bringing IM to the wireless market is to enable a standards-based approach that supports the goals of interoperability and roaming, ensuring the success of an application that will be as popular as email.

It is the goal of the *Wireless Village* initiative, launched by Ericsson, Motorola and Nokia, to ensure interoperability of mobile instant messaging and presence services while building community both around the initiative and through the deployment of innovative new IMPS services.

It is the strategy of the Wireless Village initiative to help the wireless operator succeed in attracting and retaining customers, leveraging their investment in current 2G and 2.5G as well as emerging 3G networks and increasing profits by providing a comprehensive solution that addresses both the network operator's requirements and the end-user's needs. The Wireless Village solution enables the operator to leverage their existing customer base, SMS usage patterns and business models – while attracting new customers, enabling partnerships with existing IM providers, providing new value-add services, all while building their own IMPS communities.

The Wireless Village Instant Messaging and Presence Service (IMPS) includes four primary features:

- Presence
- Instant Messaging
- Groups
- Shared Content

These features, taken in part or as a whole, provide the basis for innovative new services that build upon a common interoperable framework. The Wireless Village initiative will use its community of supporters as a forum in which to test that framework.

4.3 IPv6 Unified Instant Messaging Systems

Here we define "Unified Instant Messaging" as a general messaging service among peers over the Internet. The peer represents the person using or the application running on various kinds of devices.

4.3.1 Two approaches to the same goal

Here we describe the goals of the Euro6IX work being done by UoS and nGn in P2P and IM systems.

4.3.1.1 The Unified Instant Messaging system

For UoS, the Unified Instant Messaging system proposed would contain the following features:

- Messages are precisely formatted using XML;
- Messages between peers are asynchronous, concurrent, reliable and real-time;
- Peers can exchange messages with peers on the Internet at any time, anywhere or even while they are on the move;
- Properties such as presence and message communication should follow interoperable IETF standards;
- Security and privacy should be guaranteed;
- For different P2P application requirements, different messaging models (C/S. pure decentralized, hybrid) should be provided.

The objective of the Unified Instant Messaging System is to develop a network platform for messaging over IPv6, and then use that platform to develop an Agent based Unified Instant Messaging service for P2P applications. We will start with Instant Messaging (C/S messaging model) for users, then extend it to other messaging models and more general messaging services.

4.3.1.2 A complete Peer-to-Peer system

For nGn, the objectives of this sub-activity are:

- To study how to apply mobility and messaging in IPv6 to P2P tools, and demonstrate the advantages and benefits of the use of IPv6 in this type of applications.
- To coordinate with the obtained results by IPv6 Mobility sub-activity in A4.1 and with Unified Messaging System (6UMS) sub-activity of UoS (see above) in A4.2.
- To develop a complete P2P Instant Messaging application using the libraries developed.
- To study the possibilities to use P2P Instant Messaging between devices of different types, mobile telephones, PDA, Digital TV, etc. We will try to use XML to develop a complete structure of libraries in this sense.

The nGn Habitual Messengers application uses a central server to exchange short messages. This Instant Messaging application is a peer-to-peer tool to exchange different data types and share files without a server, although it may use other additional central services. This is a new messaging application structure because traditional applications do not use a direct peer-to-peer concept and the message exchanges are supported through central servers.

Having two different versions running over different platforms we can see the convenience of working, as a main task to deploy, on protocols and standards to guarantee compatibility between platforms, implementations, security policies, data types, devices even different applications and architectures.

4.3.2 The network platform and agent-based approach

The system being designed by UoS can be viewed at two layers:

4.3.2.1 Network Platform for Messaging over IPv6

The network platform for messaging over IPv6 being designed by UoS aims at the provision of communications API for peers by making the best use of the IPv6 services. These communications API features:

- Reliable connectivity: the fundamental challenge of communications in a P2P community is overcoming the problems associated with the dynamic nature of peers. The network platform should be able to maintain the connectivity between the peers while they are on the move. The MIPv6 (Mobile IPv6) and other related IPv6 service, such as IPv6 address autoconfiguration, Seamless roaming, Binding update, Anycast, etc, will be used to provide the reliable connectivity between peers.

- High performance: performance is a significant concern in messaging system. Instant messaging means that the messages are delivered in an asynchronous and real-time manner. QoS mechanism in IPv6 including the IntServ and DiffServ service model make the bandwidth management and real time messaging possible.

- Group communications: IPv6 multicast service will make a group of peers communicate with one another in a more convenient way.

- Security and privacy: as IPsec is mandated in IPv6, authentication and encryption are provided as standards which will meet the security and privacy requirements of the peers.

The IPsec tools should be provided by work in other subactivities, which can be deployed here.

4.3.2.2 Agent-Based Unified Instant Messaging System

The Agent based Unified Messaging system is built directly upon the network platform.

It is well known that Unified Instant Messaging systems are becoming more and more complex with the dramatic increase of the number of users and features. Agent-based processing is an appropriate way to improve the messaging system as a whole.

Basically, the agents share the following main features:

- Autonomy: the ability of an agent to effect elements of its behavioral repertoire without intervention or direct control from actors external to the agent system.

- Social ability: the capacity to communicate with other agents in the system-it is an existence criterion for our framework; an agent that does not communicate, by definition, is not a particular agent.

- Pro-activity: as part of their autonomy, agents must as least possess *opportunism* as a key goal-directed behavior; that is, they must actively search for and use the abilities of our agents to complete their tasks.

All coordinated activity of a multi-agent system is composed of agents with simple (usually singular) abilities who possess the above three criteria. The notion of *opportunism* enables us to build systems where agents can potentially discover new functionality through cooperation-emergent behavior, in short.

The notion of an *agent* has been adopt in the model of messaging system in RFC 2778. In this project, we put emphasis on the multi-agent system where the functionality of the system is provided by the individual simple agent *opportunistically* exploiting services offered by other agents.

The core functions provided by the agent based Unified Messaging system consist of:

- Instant messaging: Short text-only; File or URL; or Rich message (Multimedia Message);

- Real-time online chatting and game: Text, voice or video-voice; One2one or one2many, Fixed or mobile

- Others: Search & Meet People; Security & Privacy; or Archives Management

In order to fulfill these functions, the following agents are needed:

- User agents: A peer interacts with the system via one of several user agents. In RFC 2778, several user agents are defined: INBOX USER AGENT, SENDER USER AGENT, PRESENCE USER AGENT, and WATCHER USER AGENT. A user agent is purely coupling between a peer and some core entity of the system.

- Registry agents: Every agent knows one registry agent which it can use to advertise its services, registration state for a user, and other information used to maintain the whole messaging system, or to find out about the registration state and the services offered by other agents. The registry acts as a broker, matching agents' service needs to other agents which can provide those services.

- Unified Instant Messaging Service agents: The Unified Instant Messaging Service agents participate in and manage all Unified Instant Messaging related communications. Their primary responsibility is to provide Unified Instant Messaging services to clients. The most important services they offer to clients are packet routing and user account management. Unified Instant Messaging Service agents can be sub-divided into:

    - Presence service agents: Presence service agents are responsible for accepting, storing and distributing PRESENCE information.

    - Instant message service agents: Instant message service agents are responsible for accepting and delivering INSTANT MESSAGES.

- Other agents, such as Database agents, to provide the interface to a database of user data for messaging.

4.4 First Year Results

Both UoS and nGn are now developing Instant Messaging systems over IPv6.

4.4.1 Unified Instant Messaging

For UoS, the Unified Instant Messaging is based on the Jabber IM system at this stage. The main reason of choosing Jabber IM is simply because Jabber provides open source code and it conforms to CPIM. However, Jabber IM is IPv4-based. It has no mobility support and security guarantee, and there is a lot to do to improve its performance. Up to now UoS has:

- Built up a IPv6 based networking environment, including IPv6 connectivity, mobile connectivity, etc over which IM system will be running;
- Deployed the basic Instant Messaging protocol using Java and XML;
- Deployed the basic Instant Messaging Server based on Jabber IM;

- Migrated some of the Jabber based IM services to IPv6 platform, such as message delivery, presence state delivery, etc;
- Implement the basic Group Messaging over IPv6;

UoS has two Java development servers (Solaris and Linux) running JDK1.4.1 with IPv6 functionality; UoS is on the Sun CAP program to gain early access to new Java development code.

4.4.2 Towards a peer-to-peer system

For the application of Instant Messaging nGn has developed an only-session prototype including IPv6 functionality, to demonstrate the appropriate working of IPv6-based connections, as the first step towards a complete peer-to-peer application.

We have selected the Java language to develop it on Linux systems due to the ability to manage IPv6 functionality of J2SE 1.4.1, JSDK by Sun Microsystems. This JSDK was not available for Windows operating systems at the beginning of the subactivity, so nGn started to build a new version for Windows with Visual C++ and following Microsoft instructions to develop IPv6 applications. The prototype has been designed thinking on flexibility and compatibility and now, working with two different versions, these features are more remarked. Compatibility is enhanced by a basic communication protocol described using XML documents. By sharing the protocol these different versions of the prototype can interconnect one another.

We have taken this first step to start working to identify possible useful modules to group in a set of libraries to allow further implementations. Actually we have completed this work by adding other functionalities like peers presence information, users information publishing system and improving GUI to adequate it to new requirements.

The current state of the subactivity is as follows:

- Adapting GUI.
- Developing multi-session capability.
- Including presence service in the system.
- Including ‘white papers’ information.
- Preparing documentation to allow us to coordinate our work with 6UMS subactivity.

The first two points may require a central server to manage information. It does not imply that we forget the peer-to-peer structure, because data exchange is directly between peers. We are comparing alternatives thinking on scalability issues, and studying a further protocol to include centralized services into the whole system.

4.4.2.1 Java development details

Two different versions of a prototype of Instant Messaging application have been developed. The original idea was implemented by using Java language, knowing J2SE 1.4.1 includes IPv6 capability into socket API. But a JVM for Windows operating systems is not available yet so we started to work in a Visual C++ version prototype too.

- **Java.** This version has not presented any difficulty to run over Linux distributions. The same java code is completely useful over IPv4 or IPv6 network, selecting automatically the address type to work with.

- **Visual C++.** Microsoft publishes information to migrate applications to IPv6, even showing a sample of a console mode client – server application. To develop over IPv6 three libraries are needed, winsock2.h, ws2tcpip.h and tpipv6.h, all them include in Platform SDK from January 2000 by Microsoft.

  The January 2000 Platform SDK, Visual C++ 6.0 or later, Microsoft IPv6 Technology Preview are needed to get these libraries and manage them. This is for Windows 2000, but if you use Windows XP to develop, Microsoft IPv6 Technology Preview is not required because it includes these libraries too but tpipv6.h is included in the Platform SDK.

  If you use MFC to design GUI, be careful not to select the winsock functionality at creating the project because winsock.h will be included and you will find compatibility problems and data structure redefinitions.

  To let an agent work properly over IPv4 or IPv6 we use the data structure 'sockaddr_storage', defined in tpipv6.h, and in its field 'ss_family' we can load 'AF_INET', 'AF_INET6' or 'PF_UNSPEC' values. Selecting 'PF_UNSPEC' two addresses types would be allowed like in Java occurs. It is not so with Microsoft support as 'winsock2.h' shows:

```
/*
 * Protocol families, same as address families for now.
 */
#define PF_UNSPEC       AF_UNSPEC
.

/*
 * Address families.
 */
#define AF_UNSPEC       0               /* unspecified */
/*
 * Although  AF_UNSPEC  is  defined for backwards compatibility, using
 * AF_UNSPEC for the "af" parameter when creating a socket is STRONGLY
 * DISCOURAGED.  The interpretation  of  the  "protocol"  parameter
 * depends  on the actual address family chosen.  As environments grow
 * you  include the more  address families that use overlapping
 * protocol  values  the more  chance of choosing an
 * undesired address family when AF_UNSPEC is used.
 */
```

  Using 'PF_UNSPEC' value for 'ss_family' field in 'sockaddr_storage' we have found it works with IPv4 in an 'only IPv4 equipment' and only with IPv6 addresses in a 'dual-stack equipment'. A solution for working in the same application with IPv4 and IPv6 addresses, like Microsoft sample shows, is receiving IPv4 or IPv6 packets with a server socket defined as 'PF_UNSPEC', and sending IPv4 packets over a 'AF_INET' socket and IPV6 packets over another 'AF_INET6' socket, forcing to develop different codes.

Message structure

MessengerV6 does not manage plain text but structured data. It works using a basic information structured system defined. The way to do it is through an XML canonical form as defined in RFC 3076 'Canonical XML Version 1.0'.

For example, this is a packet format sent to a peer:

<stream:stream><user>User_1</user>

<code>03</code><mess>Hello!</mess>

</stream:stream>

As you can see, there is a general container to show the packet´s context and three different elements.

<user> tell us the name of the user operating as a peer,

<code> used to add protocol information,

<mess> to enclose the text message.

This is only a basic system defined to show the utility of xml structured data transmission based on

- draft-miller-xmpp-core-00 , 'XMPP Core' ,
- draft-miller-xmpp-im-00 , 'XMPP Instant Messaging' ,
- draft-miller-xmpp-cpim-00 , 'XMPP CPIM Mapping' .

'XMPP Core' from Jabber Software Foundation, and dated in June 21, 2002, could be a good base to develop a fully operating format design to transmit structured data between applications.

The user tag lets us identify different users connected from the same IPv6 address. Currently MessengerV6 only implements an established session but when multisession functionality is working, this feature will enable MessengerV6 use locators in the form user@equipment.

MessengerV6 is supported by an application protocol to identify remote peer state and to define request and response commands.

The code tag enables us to send protocol information to a remote peer.

The mess tag is only used to contain plain text to be transmitted in a session context. Adding new namespaces to this basic definition lets us include another MIME type's information.

4.4.3 System convergence and interoperability

We do not expect the messaging systems being developed by nGn and UoS to merge to one package. However we plan to allow the systems to interoperate by following open standards wherever possible. We expect to report more fully on interoperability in the next stage report.

In order to get a full integration of MessengerV6 into UMSv6 development, a set of XML schemas should be defined indicating the way for exchanging different content types, methods to manage them and a protocol to use.

4.5 Conclusions

IPv6 applications play an important role of getting IPv6 accepted by end users. The IPv6 Unified Instant Messaging system aims at the vision of providing advanced messaging services for peers to exchange information, share resources and cooperate. We leverage the advantages of IPv6 in the context of IM, namely its suitability for Peer-to-Peer communications through expanded address space, improved Mobile IP, autoconfiguration, and support for IPsec and Multicast.

We propose to build an agent-based unified IM system on top of this IPv6 networking platform, using IETF standards for presence and per - communication session initialisation, for maximum interoperability. We expect to re-use some existing components where possible, and to build prototype components towards trials of a working system in the scope of the Euro6IX project. Up to now some of the IM services have been developed over IPv6 networks, based on which the completed IPv6 Unified IM system will be implemented in the near future.

The P2P system being developed also under this activity will also follow IETF and related standards where possible. By using such open standards, and agreeing common message exchange formats, we expect the two messaging applications being developed to be interoperable.

4.6 Future Work

In the near future, UoS plans to continue the research and development of IPv6 Unified IM system in the following areas:

- Deploy and implement a prototype of IM system over IPv6 based on the Jabber IM system (in the future, the SIP based IM will be deployed), which includes Presence service, Info/Query service, Client Registration and Authentication, etc.

- Improve the Groupchat service by making use of multicast service provided by IPv6;

- Research and develop Mobile IM services including mobile messaging, mobile presence management, mobile IM over ad hoc networks, etc. Mobile IPv6 itself is being studied under A4.1, and UoS has its own MIPv6 test-bed.

- Research and deploy the end-to-end security mechanism for IM based on IPsec; this should be provided by other project partners from work done elsewhere in the project.

- Research and compare different messaging models (C/S, pure decentralized and hybrid) in terms of performance and their effects on different P2P applications.

For nGn, they will continue working to finish current open issues and complete their P2P application:

- We plan to coordinate this activity with the activity Unified Messaging System (6UMS) studied in A4.2 Application Development Sub-activities with the objective of getting an Instant Messaging application completely integrated in 6UMS, exchanging different data types, sharing files and publishing connection status. Mainly focused on defining common protocols, Unified Messaging System and Instant Messaging would be much more related tasks. In a further stage we will study to include security considerations in data exchanges

5. IPv6 GROUPWARE APPLICATIONS

5.1 Objectives

This subactivity is performing the porting of a couple of groupware representative tools, including a shared project space called Agora Groupware Web Server (AGWS), which includes doc repository, calendar, forums, groups, etc, and the Virtual Network Computing (VNC) remote application sharing tool. The general goal of this subactivity is to assess the state of porting of a representative set of open software developments and getting feedback about groupware application porting.

The particular objectives of the subactivity during the first year were:

- To migrate AGWS to IPv6 and to integrate it into the Euro6IX security framework, including confidentiality, data integrity, data origin authentication and non-repudiation services.
- To analyze the issues and problems arising from the porting to IPv6 of a collaborative web based tool.
- To analyze the issues and problems arising from the integration of AGWS into the Euro6IX security framework.
- To migrate VNC application to IPv6, supporting Windows, Linux and Java versions.

5.2 AGWS and VNC

AGWS (Agora Groupware Web Server) is a web based collaboration suite, which provides a rich set of collaboration functions accessible from a standard Web browser and is therefore a good example of a Web collaboration tool. It has been chosen to validate the integration of collaborative tools into the Euro6IX service integration framework.

AGWS supports the creation of “collaboration spaces” which provide a virtual meeting place for teams in projects, distributed classrooms, groups or disperse enterprises. The AGWS server is accessed by standard Web access or email. Each collaboration space includes:

- A shared file repository with version control, notification, etc
- A calendar of events and shared activities
- A virtual room for sharing Windows™ applications in synchronous collaborations
- Group management functions, including mailing list, group repository, calendar, etc
- A membership management function
- Other support functions such as group and mailing list management, forums or voting.

The architecture of AGWS is based on an Apache Web Server, which provides access to the AGWS collaboration application. AGWS makes use of the following servers: a database (Postgres SQL or Oracle) accessed via SQL, a mail SMTP server and a ROOMS server, which uses VNC and chat servers. The database and the SMTP server are accessed via TCP sockets and can be hosted in the same machine as AGWS or in a different one. The ROOMS server is very CPU intensive and is usually installed in another machine, although it can run in the same machine as AGWS.

In order to port AGWS to IPv6 as well as to adapt it to use Euro6IX services framework, the following plan was designed:

- **Step 1: Enabling external IPv6 Access.** Porting of the external interface to IPv6 and support for email addresses mapped to IPv6 only domains. After Step 1 AGWS will support external access from IPv6 Web clients and will be able to send and receive emails from email addresses mapped to IPv6-only domains. Step 1 implies the migration of AGWS to an IPv6 capable version of Apache web server, and the connection to a properly installed and configured IPv6 capable SMTP mail server.

- **Step 2: Fully IPv6 compliant AGWS.** Make AGWS fully IPv6 compliant, avoiding any need of IPv4 addresses or connections in any part of it. This includes: the migration of IP address check in AGWS license; the migration of the database interface, and the migration of the interface between the AGWS software and the ROOMS server.

- **Step 3: Integrating AGWS in the Euro6IX security framework.** Integrate AGWS with the IPv6 based Euro6IX security framework, such that it makes use of the standard facilities for confidentiality, data integrity, data origin authentication and non-repudiation services.

The integration of security into AGWS is based on the PKI developed by UMU, which allows certificate based user authentication, avoiding the need of user identifiers and passwords. The certificate of the CA can be included in the Apache web server as a trusted CA, without any specific developments in the AGWS. Therefore the adaptation of AGWS can be performed just by reconfiguring the installation procedure. This step can be performed in parallel with steps 1 and 2.

VNC is a client/server remote display software package allowing remote network access to graphical desktops. Using VNC a user can gain access to his computer 'desktop' environment from anywhere on the Internet using a wide variety of machine architectures. VNC has lots of applications, either used isolated or combined with other applications. In our case, VNC is used as an integral part of the AGWS collaboration suite.

VNC application has client and server support for Windows and UNIX operating systems, as well as a platform independent java client named vncviewer.

VNC was originally distributed by AT&T Laboratories Cambridge. However, this subactivity will concentrate on migrating TightVNC, which is an enhanced version of VNC -grown from the VNC Tight Encoder project- optimized to work over slow network connections such as low-speed modem links. Besides bandwidth optimizations, TightVNC also includes many other improvements, optimizations and bugfixes over VNC. TightVNC is free, cross-platform and compatible with the standard VNC.

Although some work were already done to migrate VNC to IPv6 (inside KAME project), it was neither available for Linux nor based on TightVNC version.

5.3 Conclusions and Future Work

At the end of the first year of the project, Steps 1 and 3 of the AGWS migration plan have been successfully performed, enabling external IPv6 access, as well as the integration of AGWS into the Euro6IX security framework. The HTTPS server provided with the application is now IPv6 compliant, so it can accept IPv6 browsers. The SMTP server has been integrated with the AGWS installation and can accept email addresses mapped into IPv6 only domains.

Besides, TightVNC has been completely migrated, adding IPv6 support to VNC clients and servers running over Linux and Windows operating systems, as well as to the VNC java client (vncviewer).

Both applications were demonstrated in trial experiments described in [4].

This subactivity will continue along the second year. In the case of AGWS, work will focus on completing the migration, dealing with step 2 of the migration plan. This step requires additional work and will begin exploring in detail the changes needed to fully support IPv6. Key points on the migration follow:

- An IPv6 enable DB server is needed, so available solutions must be explored. And the application needs to be changed to use such IPv6 DB server.
- The core application license mechanism needs to be reviewed, in order to allow the use of IPv6 addresses.
- The ROOMS working team must be contacted, to coordinate with them a joint migration to IPv6.

The main impediment for the realization of step 2 is the lack of support to IPv6 in Java applets for Windows. Therefore we will monitor the status of porting of Java 1.4 and the support for IPv6 on Windows browsers before performing the complete migration. The realization of step 2 will only start when this condition is fulfilled.

6. REFERENCES

[1] *"SIP: Session Initiation Protocol"*. RFC 3261, June 2002.

[2] *"jSIP Library"*. http://jsip.sourceforge.net.

[3] *"Mobility over IPv6 Networks"*. Euro6IX Technical Report TR4.1A.1. February, 2003.

[4] *"First Internal Trial Report"*. Euro6IX D4.3 Deliverable. February 2003.

[5] Iain Shigeoka, "Instant Messaging in Java, the Jabber Protocols." Manning Publications Co. 2002.

[6] AOL Instant messenger: http://www.aol.co.uk/aim/

[7] ICQ: http://web.icq.com/

[8] Avaki: http://www.avaki.com/company/backgrounder.html

[9] Web Services: http://www.w3.org/2002/ws/

[10] .NET XML Web Services: http://www.microsoft.com/net/basics/xmlservices.asp

[11] IBM WebSphere SDK for Web Services:

http://www-106.ibm.com/developerworks/webservices/wsdk/

[12] IONA's XMLBus Edition: http://www.iona.com/products/webserv-xmlbus.htm

[13] SOAP Messaging Framework: http://www.w3.org/TR/soap12-part1/

[14] Instant Messaging and Presence Protocol (impp):

http://www.ietf.org/html.charters/impp-charter.html

[15] Jabber: http://www.jabber.org/

[16] SIP for Instant Messaging and Presence Leveraging Extensions (simple): http://www.ietf.org/html.charters/simple-charter.html

[17] Day, M., Rosenberg, J. and H. Sugano, "A Model for Presence and Instant Messaging", RFC 2778, February 2000.

[18] Luc Moreau, Nick Gibbins, David DeRoure, et al, "SoFAR with DIM Agents: an Agent Framework for Distributed Information Management", In The Fifth International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents, pages 369-388, Manchester, UK, April 2000.

[19] Asynchronous Reliable Messaging:

http://wwws.sun.com/software/sunone/docs/arch/chapter5.pdf

[2] Multimedia Messaging Service (MMS):

http://www.forum.nokia.com/main/1,35452,1_2,00.html