Project Number:	**IST-2001-32603**
Project Title:	**6NET**
CEC Deliverable Number:	**32603/FhG/DS/D5.7/A1**
Contractual Date of Delivery to the CEC:	August 31st, 2003
Actual Date of Delivery to the CEC:	September 30th, 2003
Title of Deliverable:	**Report on Integration of SIP and IPv6**
Work package contributing to Deliverable:	WP5
Type of Deliverable*:	R
Deliverable Security Class**:	PU
Editor:	FhG FOKUS
Contributors:	Workpackage 5

* 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)

Abstract:
This deliverable reports on the integration of SIP and IPv6. It aims at trying to provide an assessment of the role SIP will presumably play in the IPv6 world and how far the integration of SIP in IPv6 related applications and protocols has taken place.
The proliferation of communication services like IP telephony and advanced supplementary services, video conferencing, application data exchange, presence and instant messaging using SIP increases the need for public IP addresses that can only become available with the deployment of IPv6. The lack of IP addresses will become even more evident with the introduction of IP based communication in next generation 3GPP networks.
In this document, we are taking a look at the benefits of using IPv6 for SIP based applications and the extensions needed for SIP to work properly in an IPv6 environment. Besides the need for applications and SIP components that can deal with IPv6 there is a clear need for components that provide for a smooth transition from IPv4 to IPv6 networks. In this context we will describe the IPv6 SIP applications and components developed in 6net.
Finally the document presents a general overview on the testing activities of IPv6 SIP components in the context of 6net.

Keywords:
SIP, IPv6, VoIP, IP telephony

Executive Summary

The primary goal of WP5 is to develop solutions that seamlessly evolve the deployment of IPv6-enabled business applications and middleware as well as understand the impact of novel, demanding applications on an IPv6 infrastructure, and to assess and report on the benefits and impact of IPv6 when used to deliver those applications. Of primary importance is the integration and testing of existing applications and middleware with minor modification. The incorporation of applications and middleware that have not been specifically modified for IPv6 is also an important issue.

SIP and IPv6 must co-habit in many applications. The fact that this combination will be required in several of the applications, will allow different comments from the user communities.

This deliverable expands on these issues and provides a framework document for organisations wishing to implement and deploy SIP based applications in an IPv6 environment.

Table of Contents

1. Introduction

Over the past decade, the telecommunications industry has witnessed rapid changes in the way people and organizations communicate. Many of these changes spring from the explosive growth of the Internet and from applications based on the Internet Protocol (IP). The Internet has become a ubiquitous means of communication, and the total amount of packet based network traffic has quickly surpassed traditional voice (circuit switched) network traffic (DataQuest, 1998).

In the wake of these technology advancements, it has become clear to telecommunications carriers, companies, and vendors that voice traffic and services will be one of the next major applications to take full advantage of IP. This expectation is based on the impact of a new set of technologies generally referred to as voice over IP (VoIP) or IP telephony.

VoIP supplies many unique capabilities to the carriers and customers who depend on IP or other packet based networks. The most important benefits include the following:

- *Cost savings*—By moving voice traffic to IP networks, companies can reduce or eliminate the toll charges associated with transporting calls over the Public Switched Telephone Network (PSTN). Service providers and end users can also conserve bandwidth by investing in additional capacity only when it is needed. This is made possible by the distributed nature of VoIP and by reduced operations costs as companies combine voice and data traffic onto one network.
- *Open standards and multivendor interoperability*—By adopting open standards, both businesses and service providers can purchase equipment from multiple vendors and eliminate their dependency on proprietary solutions.
- *Integrated voice and data networks*—By making voice "just another IP application," companies can build truly integrated networks for voice and data. These integrated networks not only provide the quality and reliability of today's PSTN, they also enable companies to quickly and flexibly take advantage of new opportunities within the changing world of communications.

In 1995, the first commercial VoIP products began to hit the market. These products were targeted at companies looking to reduce telecommunications expenses by moving voice traffic to packet networks. Early adopters of VoIP networks built toll-bypass solutions to take advantage of favourable regulatory treatment of IP traffic. Without any established standards, most early implementations were based on proprietary technology.
As these packet telephony networks grew and interconnection dependencies emerged, it became clear that the industry needed standard VoIP protocols. Several groups took up the challenge, resulting in independent standards, each with its own unique characteristics. In particular, network equipment suppliers and their customers were left to sort out the similarities and differences between different signalling and call-control protocols for VoIP:

- Session Initiation Protocol (SIP)
- H.323
- Media Gateway Control Protocol (MGCP)
- H.248/Megaco

The present paper will outline these different protocols and discuss our experience when deploying SIP in an IPv6 network.

2. Overview of Preferred Signalling Protocols for IP Telephony

2.1. H.323

H.323 was originally created to provide a mechanism for transporting multimedia applications over local area networks. Although H.323 is still used by numerous vendors for videoconferencing applications, it has rapidly evolved to address the growing needs of VoIP networks. Because of its early availability and these advancements, H.323 is currently the most widely used VoIP signalling and call-control protocol, with international and domestic carriers relying on it to handle billions of minutes of use each year.

H.323 is considered an "umbrella protocol" because it defines all aspects of call transmission, from call establishment to capabilities exchange to network resource availability. H.323 defines Registration, Admission, and Status Protocol (RAS) protocols for call routing, H.225 protocols for call setup, and H.245 protocols for capabilities exchange.

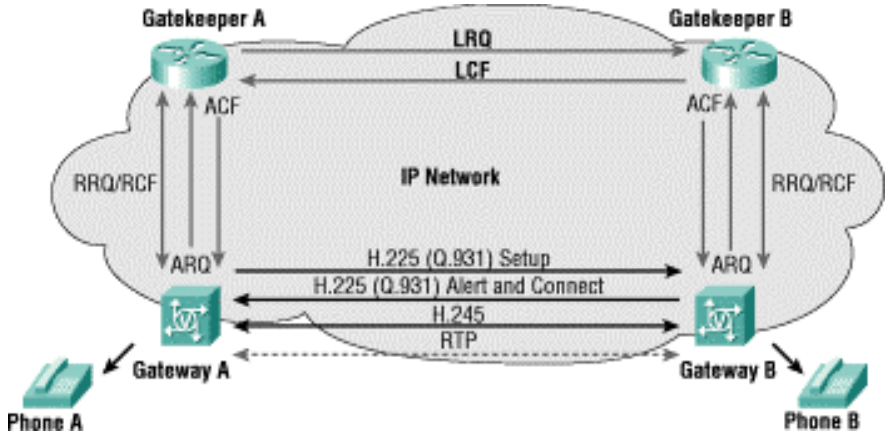

Figure 1 H.323 protocol components

H.323 is based on the Integrated Services Digital Network (ISDN) Q.931 protocol, which allows it to easily interoperate with legacy voice networks such as the PSTN or Signalling System 7 (SS7).

As a protocol used in a distributed architecture, H.323 allows companies to build large scale networks that are scalable, resilient, and redundant. It provides mechanisms for interconnecting with other VoIP networks, and supports network intelligence on either the endpoints or the gatekeepers.

2.2. MGCP/H.248/Megaco

MGCP and H.248/Megaco were designed to provide an architecture where call control and services could be centrally added to a VoIP network. In that sense, an architecture using these protocols closely resembles the existing PSTN architecture and services.

MGCP and H.248/Megaco define most aspects of signalling using a model called packages. These packages define commonly used functionality, such as PSTN signalling, line-side device connectivity, and features such as transfer and hold. In addition, Session Definition Protocol (SDP) is used to convey capabilities exchange.

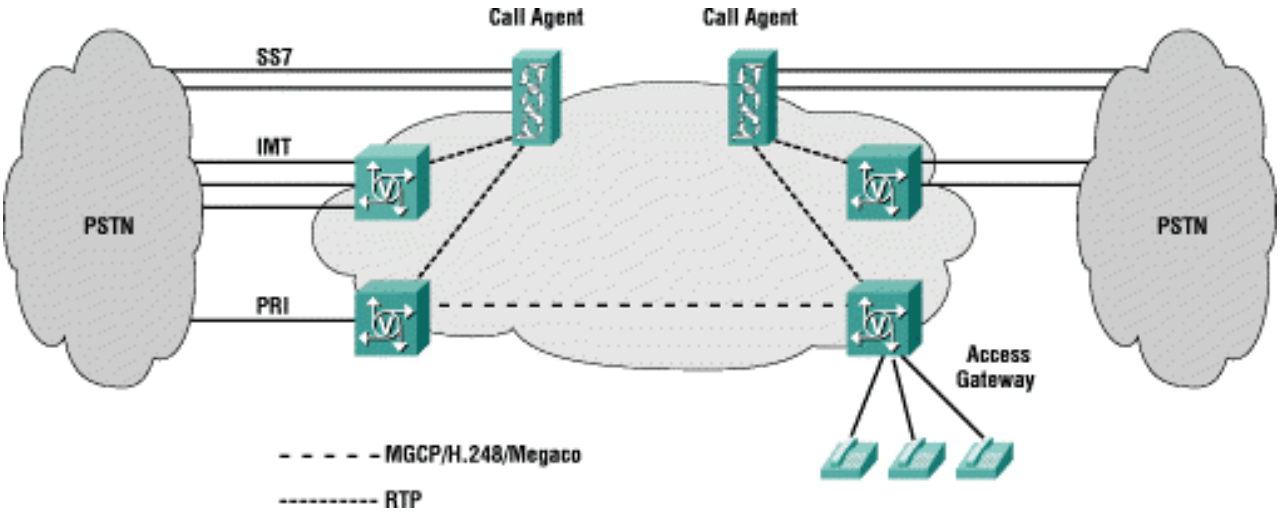

Figure 2 MGCP/H.248/MEGACO architecture

In a centralized architecture, MGCP and H.248/Megaco allow companies to build large scale networks that are scalable, resilient, and redundant. It provides mechanisms for interconnecting with other VoIP networks and for adding intelligence and features to the call agent.

2.3. SIP

SIP was designed as a multimedia protocol that could take advantage of the architecture and messages found in popular Internet applications. By using a distributed architecture—with URLs for naming and text based messaging—SIP attempts to take advantage of the Internet model for building VoIP networks and applications. In addition to VoIP, SIP is used for videoconferencing and instant messaging.

As a protocol, SIP only defines how sessions are to be set up and torn down. It utilizes other IETF protocols to define other aspects of VoIP and multimedia sessions, such as SDP for capabilities exchange, URLs for addressing, Domain Name System (DNS) for service location, and Telephony Routing over IP (TRIP) for call routing.

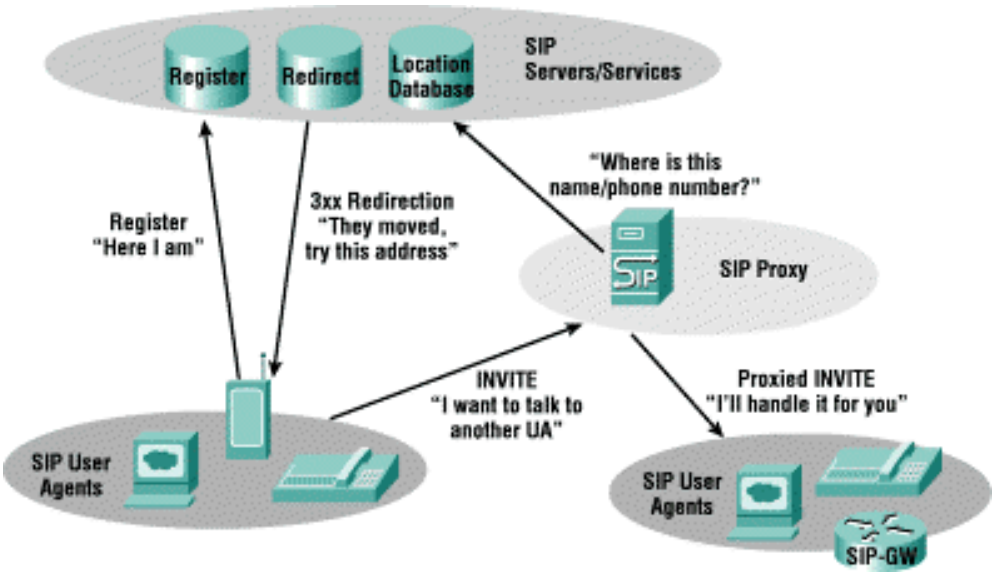

Figure 3 SIP distributed architecture

Although the IETF has made great progress defining extensions that allow SIP to work with legacy voice networks, the primary motivation behind the protocol is to create an environment that supports next generation communication models that utilize the Internet and Internet applications.

As a protocol used in a distributed architecture, SIP allows companies to build large scale networks that are scalable, resilient, and redundant. It provides mechanisms for interconnecting with other VoIP networks and for adding intelligence and new features on either the endpoints or the SIP proxy or redirect servers.

2.4. SIMPLE

SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE) uses the Session Initiation Protocol to implement services collectively known as instant messaging and presence (IMP). As the most common services for which SIP is used share quite a bit in common with IMP, the adaptation of SIP to IMP seems a natural choice given the widespread support for the SIP standard.

3. SIP Benefits from IPv6

The most obvious reason for using IPv6 with SIP is naturally the huge amount of available addresses. This is especially important when considering 3G architectures with millions of SIP based mobile phones all requiring their own IP addresses. But phones are not the only IP capable devices from the SIP point of view. Internet capable gaming stations or even appliances are also thought to be triggered by SIP. However, besides this obvious reason, IPv6 provides SIP with a row of other advantages especially in the area of dynamically configuring end systems and load balancing.

3.1. Dynamic configuration

When starting a SIP user agent in some network, the UA needs to establish a new IP address besides some SIP specific parameters such as the address of an outbound proxy, a registrar and home domain name. As such information might change depending on the location of the UA, i.e. is which network it is currently in, the setting of this information needs to be realized dynamically. The dynamic configuration capabilities of IPv6 can be very helpful in such a situation resulting in simpler configuration of user agents in a standardized manner.

3.2. Anycast

For a user agent to start a communication session it might need to send all its SIP messages to a registrar or an outbound SIP proxy which might be responsible for the authentication of the user or controlling a firewall. Finding out the location of this registrar or outbound proxy might be statically configured in the user agents. A more flexible solution is to have all proxies with similar functionalities under the same anycast address. In this scenario the messages will get directed to the closest entity. For example if for load balancing reasons more than one registrar exists, an UA might send its registration to the general address of the registrars and the registration gets to the closest one.

3.3. IPv6 solves the SIP NAT Problem?

NAT traversal is one of the biggest problems for SIP telephony in IPv4 networks. NAT servers and firewalls were not originally designed to communicate with SIP servers or to accommodate SIP's dynamically assigned ports and real-time traffic. However, under the IPv6 protocol in principle NAT shouldn't longer be an issue because the reasons for NAT—address shortage and security requirements—are not longer existent with IPv6's extended address space and its integrated IPsec functions. On the other hand there is not only a small chance that NAT during the transition period to IPv6 indeed will play a role as not all NAT devices will be completely removed in the IPv6 networks because of the tendency to retain the old NAT structures instead of configuring the new IPv6 networks so they could do without NAT components.

4. Transition Strategies

Due to the tremendous technological and administrative effort required on behalf of the user, system administrators and service provides, in order to reconfigure end devices, routers and servers from IPv4 to IPv6, the transition to IPv6 networks will be done gradually [7]. Three major transition mechanisms can be distinguished:

- **Dual stack operation**
  This presumes support for both IPv6 and IPv4 at the same time. That is, the networks run both IPv4 and IPv6 routing protocols and the end systems are capable of sending and receiving IPv4 and IPv6 data packets and possess both an IPv4 and an IPv6 address. In case a data packet with an IPv4 address is received, the end system replies with packets carrying its IPv4 address. Sending data to another host is done in a similar manner. If a DNS query resulted in an AAAA record then the IPv6 address is used. While this is a simple transition mechanism it requires providing IPv4 addresses to all end systems, which negates the major advantage of IPv6. Further, it complicates

the network architecture, as it requires managing both IPv4 and IPv6 routing protocols.[1]

- **Tunnelling**
  With this approach IPv6 islands are connected through tunnels established over IPv4 networks. Both ends of the tunnel act as dual stack routers connected to both IPv6 and IPv4 networks. IPv6 packets arriving at one end of the tunnel are encapsulated in an IPv4 packet and sent over the IPv4 network. At the other end of the tunnel, the packets are decapsulated and sent as IPv6 packets to their final destination. While simple to deploy in restricted areas, managing a large number of tunnels becomes complicated.
- **Protocol Translation (NATPT)**
  With this approach a gateway is placed between IPv4 and IPv6 networks. These gateways act similar to network address translators [23]. That is, such a gateway manages a pool of IPv4- and IPv6 addresses. After receiving a packet from one interface—e.g., IPv4—the source destination is replaced with one of the gateways addresses —e.g., IPv6—and vice versa. This approach has the advantage that end devices and networks need only to run either IPv4 or IPv6. However, this approach breaks the end-to-end transparency of the Internet and is accompanied by similar problems as introduced by NATs [12]. That is, besides translating the IP headers special attention needs to be paid to protocols such as SIP and FTP, which carry addressing information in the protocol messages themselves.

5. Available Applications

5.1. Vocal

The Vovida Open Communication Application Library (VOCAL) is an open source project targeted at facilitating the adoption of VoIP in the marketplace. VOCAL provides the development community with software and tools needed to build new and exciting VoIP features, applications and services.

Vovida Networks has been acquired by Cisco Systems in 2000 and Cisco continues the support of the Vocal open source project.

The VOCAL system is a distributed network of servers that provides Voice over Internet Protocol (VoIP) telephony services. VOCAL supports devices that communicate using the Session Initiation Protocol (SIP, RFC 3261). VOCAL also supports analogue telephones via residential gateways.

VOCAL supports on-network and off-network calling. Off-network calling enables subscribers to connect to parties through either the Internet or the Public Switched Telephone Network (PSTN).

5.1.1. High-Level System View

The next figure shows a high-level, simplified view of the system.

[1] There are different approaches for overcoming this requirement, but they further increase the complexity of the network.

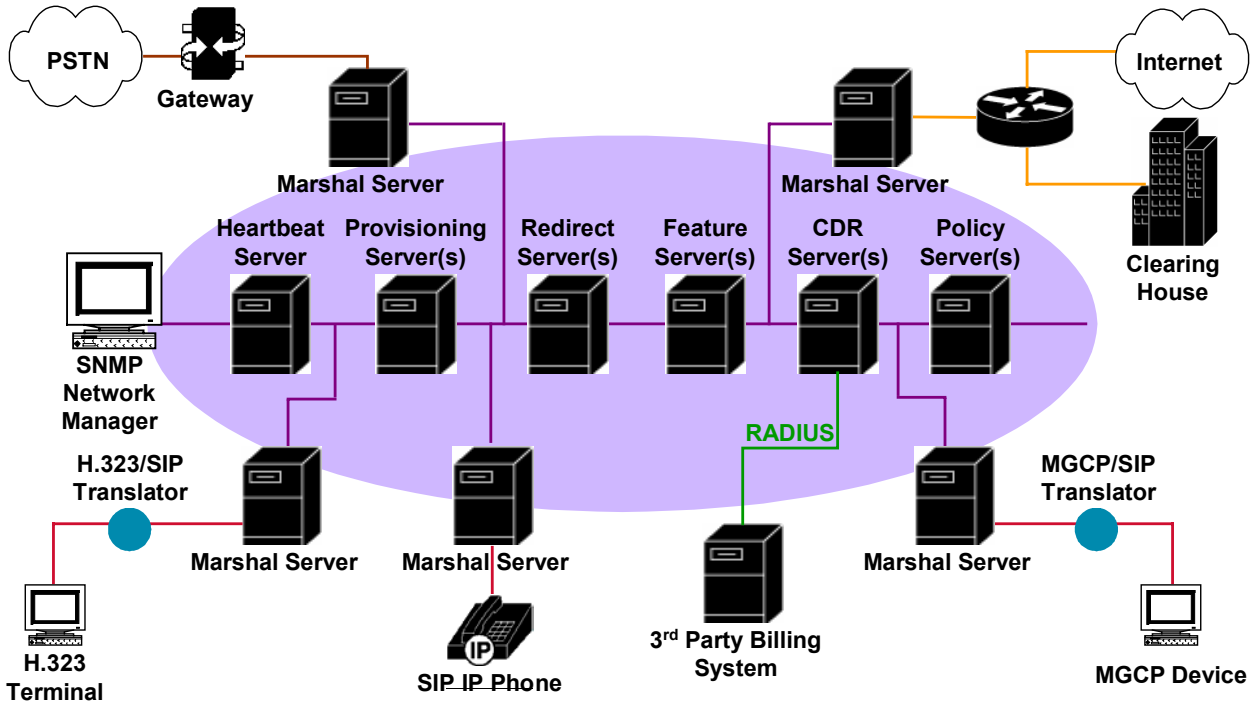

Figure 4 VOCAL Architecture

5.1.2. System Components

From a high-level point-of-view, the VOCAL system appears as an assembly of basic components. These components are described below in Table 1.

Component	Description
VOCAL System	This is the telephony application. Figure 2 shows an abstract representation of the VOCAL system server modules. A description of each server appears in the next section, see "Servers"
Protocols	The VOCAL system uses several protocols to communicate between its components. The call signalling processes use SIP messaging to communicate internally within the VOCAL system and externally with gateways and IP phones.
GUI	The graphical user interface (GUI) enables technicians to provision the system, and administrators to set up users and monitor the system's performance.
IP Phone	VOCAL supports a variety of phone appliances including SIP phones and SIP User Agent (UA) software applications. SIP phones may be connected to the VOCAL system over any IP network.
Gateways	Gateways not only provide entry points between networks, they also provide translation between SIP-based networks and other network types. The VOCAL system works with two types of gateways, the Residential Gateway and the Trunking Gateway. **Residential Gateway**

	Residential gateways translate analogue signals into IP packets, to permit subscribers with analogue phone sets/devices to make and receive SIP-based calls. **Trunking Gateway** Trunking gateways permit SIP-based networks to exchange calls with end-points on the PSTN, by providing translation between SIP messages and one of these signal types: • Analogue • Channel Associated Signalling (CAS) • Primary Rate Interface

Table 1 VOCAL basic components

5.1.3. Servers

Table 2 describes the server modules included in the VOCAL system.

Component	Description
Marshal Server	The Marshal Server (MS) is an implementation of the SIP proxy server and acts as the initial point of contact for all SIP signals that enter the VOCAL system. The MS provides authentication, forwarding and billing functions.
Redirect Server	The Redirect Server (RS) is a combined implementation of the SIP redirect, registration and location servers. The RS stores contact and feature data for all registered subscribers and a dialling plan to enable routing for off-network calls.
Call Detail Record Server	The Call Detail Record (CDR) server receives call data from the Marshal Servers and formats it into data that can be transmitted to third party billing systems for invoicing.
Voice Mail Server	The Voice Mail server provides unified messaging whereby voice mail messages can be distributed as .wav files attached to e-mail messages.
Feature Server	The Feature Servers are another implementation of the SIP proxy server. These servers are scripted in Call Processing Language (CPL) and provide basic system features such as Call Forward and Call Blocking.
Provisioning Server	The Provisioning Server (PS) stores data records about each system user and server module, and distributes this information throughout the system via a subscribe-notify model. The PS provides a web-enabled graphical user interface to permit technicians and system administrators to manage the system.
Heartbeat Server	The Heartbeat Server monitors the flow of signals emitted by the other servers, and provides information about to the flow of heartbeats to the Simple Network Management Protocol (SNMP, RFC 1157) GUI. This information helps the System Administrator know if the server modules are up or down.

Table 2 VOCAL server modules

5.1.4. Support for IPv6 Mbone tools in Vocal SIPSet User Agent

SIPSet is a SIP User Agent (UA) used for making and receiving phone calls from a Linux PC. The application consists of a GUI front end that interacts with the Vocal SIP stack.

SIPSet includes support for audio and video communication using a simple device plugin architecture. There are three device classes as standard; Audio, Video and a Null Device for testing call control:

The *Audio* device is a simple interface between the soundcard and the RTP stack. RTP data received is sent directly to the soundcard and audio captured from the soundcard is passed to the RTP functions.

The *Video* device uses an external MPEG library to render video data. Call configuration details (addresses, port numbers) are extracted from the SIP packet and the data is written into the users home directory, as a configuration file for the MPEG library. A window is created within the current user interface to accommodate the video display and calls are made to the external MPEG library. This dynamically loads the MPEG library which reads in the configuration file.

The *Null* device is an empty class which just outputs the status of the call - i.e. "connect", "disconnect" etc.

A configuration file is read on startup for selecting which audio hardware to use and whether to use the MPEG video library. Video support is only available if SIPSet has been built to include the MPEG header files.

UCL has started a three phased project to make the SipSet application capable of starting and configuring the IPv6 versions of the UCL MBone tools, VIC and RAT. This will complete the work on making the Vocal system IPv6 and enable users to make IPv6 audio and video calls.

Phase one adds a new *External-Media* device that starts the Mbone tools when a call is initiated. The device interacts with the SIP stack to extract call configuration details from the SIP packet. This information is then used to start the appropriate media tool with the remote address of the caller and the remote/local ports for sending/receiving the RTP data. When a call ends the tool is terminated. Configuration of which media tool to execute can be done in either the SIPSet configuration file or through the user interface. This phase has successfully been completed and patches are being compiled for submission to the Vocal project.

Phase two improves on the configuration of the media tools, so as to support extra parameters within the SDP (such as redundant transmission, interleave audio etc). This will be done by implementing SDP parsing into the UCL Mbone tools. The whole SDP packet will then be passed from SIPSet to the tool, which will then extract the relevant parameters for configuration.

Phase three will use an inter-process communications channel, such as the Mbus [14], for configuring the media tool. This has the advantage that if any format changes occur during the call, the media tool can be easily re-configured without the need to restart it with updated parameters.

In addition to the above work we have also been conducting tests using the Cisco 7960 IP phones and the UCL audio tool RAT. Using the phones to make a call, RAT is started by the SIPSet application with the correct configuration. RTP Audio data is then sent from the Cisco phone to RAT for playout. Unfortunately the Cisco phones do not send RTCP control data alongside the RTP data and therefore RAT did not play the received packets. We have fixed this by adding an *RTP promiscuous* mode to RAT, that plays received RTP packets without first waiting for source identification from the corresponding RTCP packet.

5.1.5. VOCAL porting

The VOCAL package was ported by the University of Southampton to support IPv6.

VOCAL as a whole is a complex package. The porting work done focused on the core network stack, the SIP stack and the SIPSet user agent, i.e. enough of the system to get user-to-user SIP-based VoIP calls working.

The Vovida SIPSet is actually a standalone package SIP User Agent with a GUI front end that works with the Vovida SIP stack. You can use the SIPSet as a soft phone, to make and receives phone calls from your Linux PC.

The ported components were:

Network Stack:

- IPv6 support added to the UDP stack (used much more widely than TCP). Amounted to a near total re-write
- Use of IP-independent data structures to store addresses, not integers
- Tries to store names rather than addresses for IP-independence
- Making it less likely to pass an IPv6 address to an IPv4 client

SIP Stack:

- Extensively changed to support IPv6
- Parsing issues
  - Mainly the use of ‘:’s as a port separator—the solution was to use RFC2732-style delimiters, i.e. ‘[’, ‘]’ around the literal IPv6 address.
- Rewrite new IP-independent API calls to resolve DNS, and retrieve AAAA and A records.
- Support extensions of the SIP protocol to allow support IPv6 addresses, and embed them in the packets.
- API changes kept minimal to ensure changes were as transparent as possible.

SIPSet:

- Modified to properly parse, display and accept IPv6 addresses.

The porting process highlighted a few implementation-specific issues. For example, there are some subtleties between platforms, in terms of how they handle binding to IPv4 and IPv6

simultaneously. The new IPv6 API's, in particular the IP-independent calls such as `getaddrinfo()`, may not be present.

Some classic porting issues were encountered. Properly written network code would have been easier to port. The passing of IP addresses as integers forced API changes. The format of IP addresses (dotted quad) is assumed in many places; thus the code for parsing IP addresses would have better been centralised. Porting involved repeatedly changing `inet_aton` statements.

SIPSet is working, both in dual-stack and in IPv6 or IPv4 only environments. The code is integrated into the v1.5 release of VOCAL.

However VOCAL as a whole is not so well supported. In particular the "all-in-one" configuration script relies on Perl, which is currently lacking IPv6 support. Other components will need updating, however this should be restricted to UI input checks and parsing.

Among items remaining to be done is the TCP Stack, which was undergoing substantial changes while the porting work was in progress. Some improvements with regards to protocol selection would be desirable; this is currently DNS based, i.e. it assumes IPv6 enabled if AAAA record is present (which is not a great assumption!). Clean fallback mechanisms are required, e.g. if IPv6 fails attempt IPv4.

We will test further application layer IPv4-IPv6 SIP gateways for interoperability between protocols. We also plan to integrate ENUM in IPv6 VOCAL. As for the rest of VOCAL, other components may be partially IPv6 enabled "for free" as a result of using the SIPSet and SIP and Network Stacks.

The user agent has been tested between partners in 6NET informally at the time of writing. It was demonstrated at IST2002 in late 2002. In the 6WINIT[2] IST project review demonstration in early 2003 the VOCAL IPv6 code was shown to interoperate with the Bonephone IPv6 application for SIP-based VoIP. In addition, the SIP gateway developed by the TZI Institute at the University of Bremen[3] also allowed calls to be placed to a UMTS endpoint from a VOCAL client. This was a collaboration between TZI, Ericsson and UoS. The full demonstration included VOCAL, Bonephone, a Cisco 7960 and Microsoft Messenger.

The TZI gateway is written in Java (v1.4). It also allows bridging of SIP-H.323 and IPv4 to IPv6. It is available from the 6NET applications database[4].

5.2. SIP Express Platform

The SIP Express Platform includes support for registrar, proxy and redirect mode. Further it acts as an application server with support for CPL, instant messaging and presence (IM&P) including a SMS gateway, a call control policy language, call number translation, private dial plans and accounting, authorization and authentication (AAA) services. The SIP Express platform runs on Sun/Solaris, PC/Linux, IPAQ/Linux platforms and supports both IPv4 and IPv6.

[2] http://www.6winit.org/
[3] http://www.tzi.de/
[4] http://6net.laares.info/apps.phtml?appID=43

5.2.1. SIP Express Router (SER)

Iptel.org's SIP Express Router (SER) is a high-performance, configurable, free SIP (RFC 3261) server which was implemented in C and ported to Linux (PC, IPAQ), BSD (PC) and Solaris (Sun). Originally developed for IPv4 only, today support for IPv6 is included.

The IPv6-ability of the SER was achieved as part of FhG's framework of 6net activities. The following IPv4 code constituents were mainly affected and had to be modified resp. extended:

- IP addresses and socket handling:
  - Change of internal SER ip address and sockaddr representation to handle both IPv6 and IPv4 addresses
  - As portability was a major goal, SER+IPv6 works on linux, solaris, freebsd, netbsd and open BSD both on 32 bit and 64 bit architectures.
- name resolution:
  - Internal name resolution functions were changed to handle AAAA records (records containing IPv6 addresses).
- parsing:
  - Header parsing and URI parsing functions were changed to handle IPv6 address references (literal address syntax[5]) or normal IPv6 addresses (canonical form) as necessary. E.g. IPv6 addresses must be generally specified as IPv6 address references (to avoid ambiguity when determining the port number) with the exception of the IPv6 address in the "received" VIA parameter which must not be a reference.
- message modifications / construction:
  - IPv6 addresses are added to VIA, VIA received parameter, Record-Route etc.
- IP address realm transitions:
  SER is able to forward messages between different address realms, i.e. from IPv4 to IPv6 and from IPv6 to IPv4.
  - In the VIA header always the outgoing interface address is used (so "our" VIA will always contain the proper address realm).
  - Double record-routing is used: We add 2 record route headers when we detect an address realm transition. One will contain the address in the origin realm (e.g. IPv4), and the other the address in the other realm (e.g. IPv6). This will ensure that all the other messages in the dialog will pass through our proxy, which will take care of the realm transition. The double record routing solution doesn't involve keeping any state (of course record routing must be enabled in the `.config` file).

The SER can act as registrar, proxy or redirect server. SER features an application-server interface, presence support, SMS gateway, SIMPLE2Jabber gateway, RADIUS/syslog

[5] A literal IPv6 address in a URL is the address enclosed in square brackets. For example, to reach www.ipv6.iptel.org at 2001:610:148:dead:210:18ff:fe02:e38, the URL is http://[2001:610:148:dead:210:18ff:fe02:e38]/. The use of the IPv6 literal address syntax is described in RFC 2732 [11].

accounting and authorization, server status monitoring, FCP[6] security, etc. A web based user provisioning, serweb[7] is also available.

Its high performance allows it to cope with exceptional operational burdens, such as broken network components, attacks, power-up reboots and rapidly growing user populations. SER's configuration ability meets needs of a whole range of scenarios including small-office use, enterprise PBX replacements and carrier services.

SERv4 has been successfully tested with SIP products from other vendors such as Microsoft, Cisco, Mitel, snom, Pingtel, and Siemens. Specific IPv6 based interoperation tests had been performed as described in section 6 “Interoperability of SIP implementations”.

With the help of a mailing list it is possible to obtain additional help from the SER community. To participate in the mailing list, subscription at the following web address is required: http://mail.iptel.org/mailman/listinfo/serusers.

Discussion of development, new features and SER status as on CVS takes place at the following mailing list: http://mail.iptel.org/mailman/listinfo/serdev.

5.2.2. Mini SIP Proxy (MSP)

The Mini SIP Proxy[8] (MSP) receives SIP messages, modifies them, installs UDP mappings for RTP communication and forwards the SIP messages to another proxy. The MSP depends on two outbound proxies: one for IPv4 and one for IPv6 targets, as it does not route requests itself. If a SIP request message is received by an IPv4 interface, it is sent out to the IPv6 proxy and vice versa. SIP response messages are routed by their second via header. Only the following parts of a SIP message are modified:

a) **Contact header**:
   This is modified by replacing the original contact header with the URI of the SIP-PGW with an additional parameter reflecting the original contact address (“real_uri” parameter). A peer that sends subsequent requests (e. g., BYE) must send them to this modified contact header, i.e., the SIP-PGW with the real_uriparameter.
b) **Request URI**:
   This is modified only if the Request URI has a real_uri-parameter. It is then replaced by the original URI represented by the real_uri.
c) **SDP-headers**:
   Located in the body:
   - originator (o=)
   - contact (c=)
   - media description (m=)

   They hold IP-Addresses or ports and must be modified to match the target protocol family. The addresses included in the SDP part are allocated by sending a mapping request to the UFWDD.

[6] www.iptel.org/fcp/
[7] developer.berlios.de/projects/serweb/
[8] The MSP is still under development. Software and instructions can be obtained from iptel.org in case of eager demand.

d) **Content-length**:
When the body (SDP) is modified, the content length needs to be recalculated.
e) **VIA**:
A via header is inserted in request messages and removed from response messages.

As an integration mechanism for enabling the communication between an IPv4 only capable device and an IPv6 only capable device we chose the protocol translator approach. This allows for simple end devices and networks that only need to support one of the IP versions.

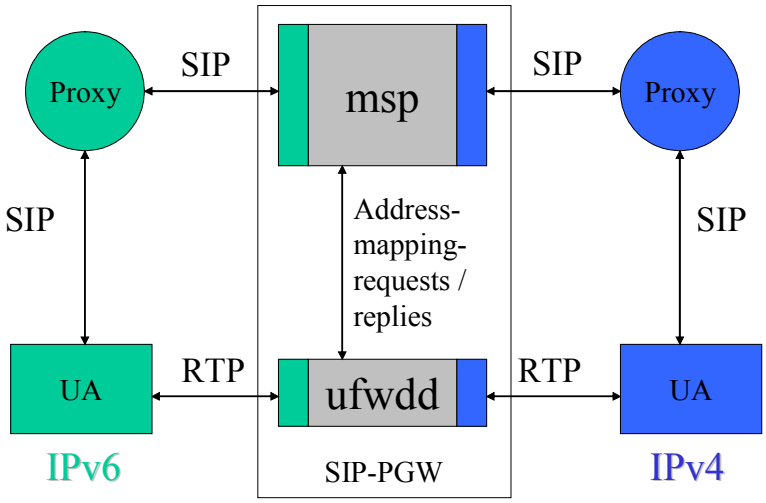

Figure 5 SIP gateway for heterogeneous environments (PGW)

This protocol translator called the SIP Protocol Gateway (PGW) is intended to be located on the borderline between pure IPv6 and pure IPv4 clients. It runs on a dual-stack machine to be able to speak and listen to both protocol-families. It can be considered as a proxy, which modifies the SIP messages sent by an IPv4/IPv6 host to be understood by an IPv6/IPv4 host.

The SIP-PGW consists of three components:

1. **MSP**
2. **UDP-forwarding daemon (ufwdd)**:
   This entity manages the IPv4 and IPv6 address spaces of the gateway and acts as a network address translator. Packets received on the IPv4/IPv6 side are sent to an IPv6/IPv4 host using a sending address and port number allocated from the IPv6/IPv4 address and port space of the gateway.
3. **Control protocol**:
   For requesting the allocation of addresses and mapping between the protocol families, both the MSP and FWDD communicate via UDP messages. This also allows both components to reside on different machines. This is done in a fashion similar to the middlebox architecture of the IETF.

Besides the gateway, Figure 5 shows two SIP proxies. Each proxy represents the SIP provider in one side of the heterogeneous network. I.e.—for the example of a SIP provider such as iptel.org requests originating from the IPv4/IPv6 network and destined for subscribers of iptel.org should be directed to the iptel.org proxy located in the IPv4/IPv6 network. In case the proxy in the originating network, e.g., IPv4, is incapable of locating the user, it forwards

the request to the gateway, which forwards the request further to the proxy of iptel.org in the IPv6 network after translating its content. While the functionalities of both proxies could also have been integrated into the SIP-PGW this would have increased the complexity of the gateway and would increase the load on the gateway as it would need to process a higher number of SIP messages and possibly execute some services such as CPL or similar besides translating and routing the data. This distribution further allows the gateway provider to be an independent provider concentrating on the translation of signalling and media packets and relieves the SIP service provider from having to deal with media packets as well.

5.3. Bonephone

The bonephone application was developed in order to integrate the telephony service into the PC. The application utilizes voice over IP (VoIP) [3] to carry digital voice data over the internet to another VoIP capable application or to a traditional phone through a translating gateway.

To be ready for the future, this application is based upon the latest international standards. These are the internet protocol (IP) [1], the internet protocol version 6 (IPv6) [5], the user datagram protocol (UDP) [16], the real time transport protocol (RTP) [20], the real time transport control protocol (RTCP) [20] and the session initiation protocol (SIP) [20].

Besides of supporting the newest standards the architecture of the application has been designed in an open and extendable manner. The design, for example, allows the easy extension of the tool with other media like video or other types of communication like conferencing. On the other hand, to allow for easy customization and personalization of the tool to the specific needs of a user or a vendor the tool supports easy replacement of the entire graphical interface with a new one. Finally, the tool was designed and implemented in a portable and scalable manner allowing its usage over different hard- and software platforms.

Bonephone was implemented allowing for easy extension of the functionality of the application and replacement of the GUI.

To provide for adequate replacement of high-end phones the tool is able to deal with parallel calls at the same time, maintain call logs and provides callback features in addition to phonebook functionalities.

The software was implemented as a distributed program with components communicating over the message bus (MBus). In this way it is possible to add new media engines, and even use media engines that do not reside on the same machine.

Bonephone provides different configuration options (cf. bonephone manuals for installation and usage[9]) . There is one for the phonebook, which holds names of people and their associated SIP address. There is one for the general information like the executable image of the media image, whether to use IP or IPv6, etc. In a third configuration file new profiles for data communication can be defined.

[9] http://www.iptel.org/products/bonephone/bonephone-user-guide and
http://www.iptel.org/products/bonephone/bonephone-installation-guide

5.3.1. General Architecture

The bonephone VoIP application consists of the following components shown in Figure 6:

- an instance that manages RTP/RTCP communication for audio data transport
- a SIP capable user agent which utilizes a SIP protocol stack
- a graphical user interface to interact with the user
- an instance that coordinates the media engine, which does not interact with the signalling itself, with the SIP user agent
- a phonebook

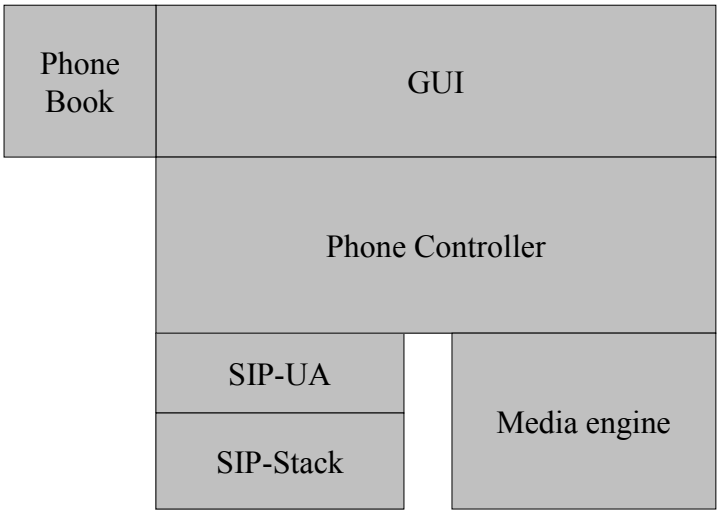

Figure 6 General architecture of bonephone

Three already existing open source software products could be used for the IP phone but had to be redesigned in parts. These are:

5.3.1.1 Media engine

The media engine is responsible for the audio communication. It reads samples from the audio device of the PC, encodes them and sends them through the internet using the RTP/RTCP protocols. It is also responsible for the reverse process, i.e. it must be able to receive encoded audio samples via RTP/RTCP, decode them and write them to the audio device for play back.

It provides an interface allowing the control of the following parameters:

- The remote IPv6 address and port number to which the RTP/RTCP data will be sent as well as the local port number where it receives audio data, and
- the audio encoding algorithm to be used for sending audio data.
- The interface allows the modification of runtime parameters like volume or active output channel number.
- Finally, the control interface allows for reporting status information either periodically or on demand.

The media engine supports a large number of audio encoding algorithms to be interoperable with as many other end devices as possible. Only minimum configuration has to be done to enable detection and interaction with the audio device.

Figure 7 depicts the data flow of the media engine.

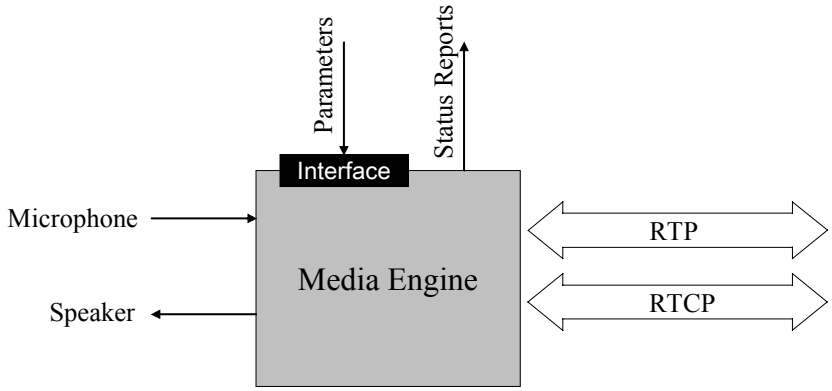

Figure 7 Media data flow

5.3.1.1.1 Robust audio tool

The robust audio tool (RAT)[10] is basically a software tool that acquires data from an audio device, encodes it by means of a pre-defined codec and sends it through the internet to a given destination. Codecs are realized as small modules located in the RAT media engine that can encode raw audio data into a standardized, mostly compressed format and decode it back to raw audio data. RAT also receives encoded audio data from the remote machine through the internet, decodes it and re-plays it on the audio device.

RAT is mainly composed of three components as depicted in Figure 8: The graphical user interface, the media engine and the controller. They are realized as separate processes which communicate with each other using the MBus technology as the inter process communication mechanism.

[10] RAT can be obtained from the internet http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/

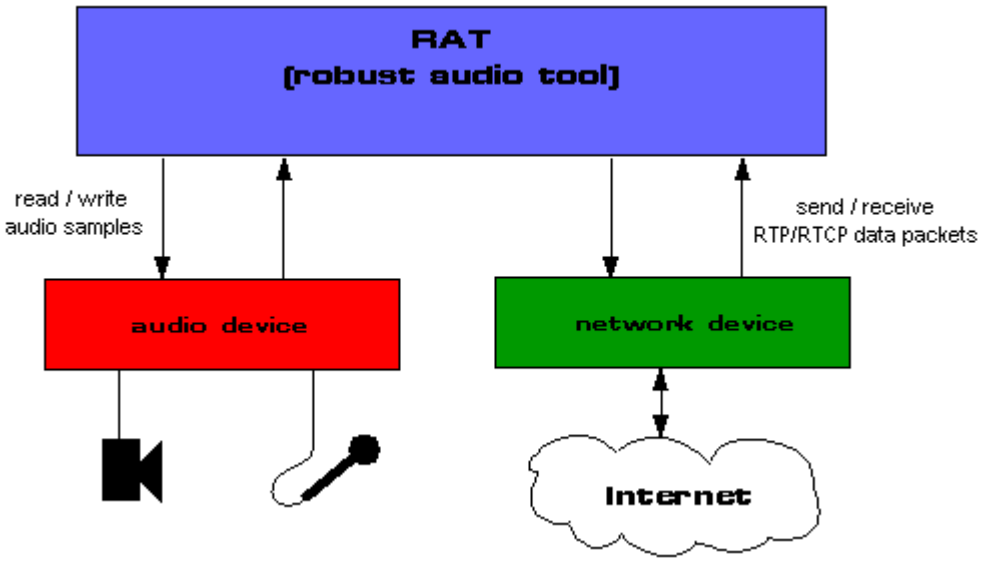

Figure 8 General RAT functionality

The graphical user interface interacts with the user. It provides graphical control elements like buttons and icons to give the user the ability to change the volume or mute a specific audio source. It receives status information from the media engine which is displayed to the user like the packet loss or other RTP related information.

The RAT media engine does the task of audio data transport.

The controller's purpose is to start the GUI and the media engine. It controls the MBus communication during the session and orderly shuts down every component when the session terminates.

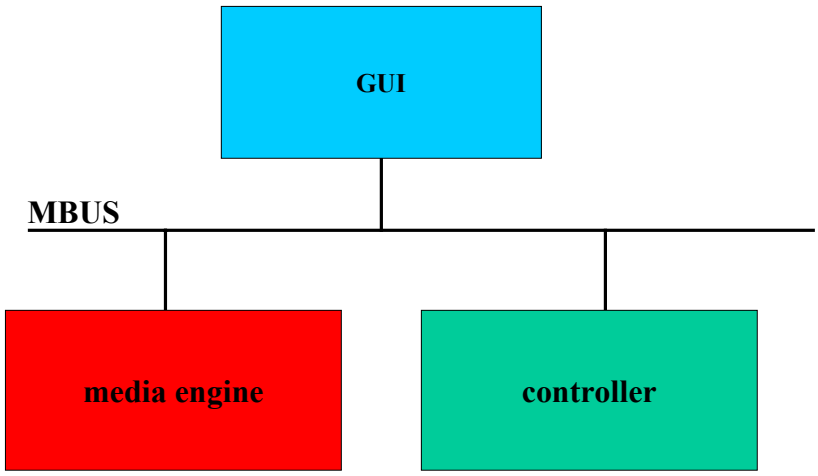

Figure 9 RAT processes

The graphical user interface, the media engine and the controller exchange messages with the semantics of function calls. So, e.g., the GUI can query the media engine to start sampling audio data and send them using RTP and RTCP to a known peer.

RAT does not support any kind of signalling. This means that one has to know the IP address and port number where the partner is listening. And the partner needs to know this also. With this, only the media engine will be of further interest for our project. It provides the required interface, which is implemented as a MBus interface, and meets the other demands.

5.3.1.1.2 Message Bus

The MBus is used in the internet multimedia environment for communication between software components. With MBus distributed programming can be realized even over different nodes.

MBus utilizes IP multicast [4] for its message transport. There is no central instance that manages communication. Therefore each component has its own interface to send and receive messages to or from the specific multicast address.

MBus defines its own address scheme to distinguish participating components

MBus is implemented in different ways. For C/C++ there is a library from the University College of London (UCL) providing the functionality to attach to an MBus and to send and receive messages over it. For Java there exists a package providing the required functionalities[11].

5.3.2. SIP Protocol Stack

The purpose of a SIP protocol stack is to provide an interface that is independent of the SIP protocol primitives and transactions.

There was no SIP stack available that was satisfying these requirements. What is available is a SIP protocol suite with basic SIP related functionalities and a SIP user agent which is able to use this protocol suite.

5.3.2.1 NIST SIP protocol suite

The SIP protocol suite was developed by the National Institute of Standards and Technology (NIST)[12] as a Java package. It has an interface conforming to the Java APIs for integrated networks (JAIN) which is far too complex to be simply used by a telephony application as it requires the application to deal directly with SIP transactions. So the application cannot abstract from SIP as underlying signalling protocol. The NIST SIP protocol suite has limited capabilities to react to SIP messages itself if an incoming message appears to be damaged, broken, or incomplete. In such a case a negative response, describing the problem, is automatically generated and sent to the originator of the message that triggered the error. Another feature is that it can prepare the response message to a received request message from the request header fields. The higher layer software can then modify and complete these data before the response is sent. Figure 10 depicts the data flow of the SIP protocol suite.

[11] The Java package and documentation is available at http://sourceforge.net/projects/jmbus/
[12] More information can be obtained from http://www.nist.gov

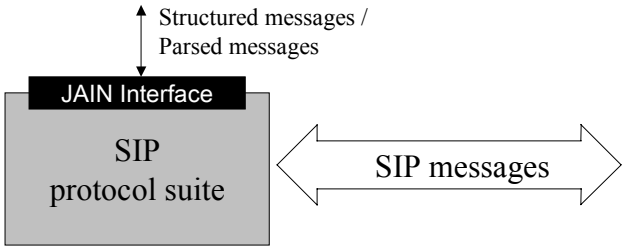

Figure 10 SIP protocol suite data flow

5.3.2.2 SIP user agent

As mentioned above, the interface to the SIP stack is not usable by a telephony application. This requires a piece of software, the SIP user agent, to be placed on top of the JAIN interface of the NIST SIP protocol suite. The purpose of the SIP user agent is to provide a functional interface to the telephony application for call signalling, which abstracts from SIP as signalling protocol and hides SIP call state handling as well as SIP session and transaction management. It implements the correct protocol behaviour, i.e. it knows which replies have to be sent in response to previous request messages, etc. A SIP user agent that fits the specification already exists as a Java based package[13].

5.3.3. Phone Controller

The phone controller has the task of coordinating the media engine and the SIP user agent. It provides an interface giving the functionality of a telephone. It is capable to manage multiple calls at the same time and provides functions for call initiation, call muting and termination as well as functions to change sound parameters like volume and the input- and output channels in use.

5.3.4. Graphical User Interface

The GUI is a graphical way to represent the phone controller system to enable easy ergonomic interaction.

5.3.5. Phonebook

The phonebook is a database of names of people and their associated SIP-URI which is needed to start calls with them.

5.4. KPhone

KPhone is a Linux based Voice-over-IP telephony application which supports IPv4, and IPv6 as well. It is based on the *K desktop environment* (KDE) version 3.1[14] and the corresponding graphics and widget library QT version 3.1. KPhone implements SIP for call signalling and RTP for audio streaming. KPhone is written in the C++ programming language.

[13] The SIP user agent / documentation can be obtained from FhG FOKUS. (Stefan.Foeckel@fokus.fraunhofer.de)
[14] http://developer.kde.org/build/compile_kde3_1.html

5.4.1. KPhone Components

KPhone is basically composed of 4 components: The *libdissipate* library, the KPhone application itself and the GSM and iLBC libraries.

The *libdissipate* library holds the basic networking code, i.e. the classes for creating sockets and resolving names to IPv4/IPv6 addresses. It also holds the SIP protocol stack for call signalling.

The following Figure 11 gives an impression of the basic components of a VoIP phone and their relation to the involved KPhone components.

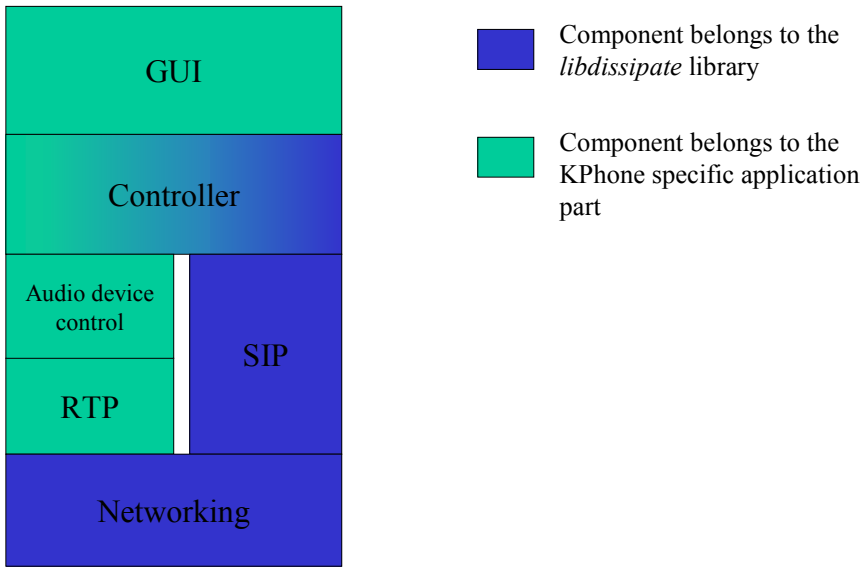

Figure 11 Components of a VoIP phone

The GUI holds all classes that deal with graphical elements to communicate with the user in order to inform him about incoming calls or accept his outgoing calls. The Controller's job is to handle the SIP stack and to program the audio engine with the appropriate information about the used media encoding and the peer address to send it to. The audio device control instance will collect samples from the audio device (microphone) and writes samples to it to be played over the speakers or headphones. Those samples are obtained from and written to the RTP protocol suite which does the audio data compression and encoding with standardized encoding/decoding schemes (codecs). The SIP stacks purpose is to generate and interpret RFC 3261 conform messages to handle call signalling, i.e. to negotiate addresses for audio data streams, call setup, modification (muting) and termination (hang-up). The networking code consists of classes to implement different types of message sockets and to do the host name lookup procedure.

In section 5.4.1.1 the changes in the libdissipate library are discussed. The changes to the application part are part of section 5.4.1.2 while section 4 gives a summary / outlook.

5.4.1.1 libdissipate

The libdissipate library holds the SIP client code and the networking code. It holds also a very small part of the GUI. When KPhone starts up, it detects all assigned IP addresses. If it detects more than one IP address, it will ask the user by a popup box which one to use. This happens in the `SipUtil` class of the libdissipate library.

5.4.1.1.1 Class SipUtil

This class holds the fully qualified domain name (FQDN) which is to use for signalling. KPhone uses the textual representation of an IP address for this. The class holds functions to detect the actual IP addresses and to ask the user if there is more than one address detected. The function `findFqdn()` detects the IP addresses by first opening an IPv4 socket and then using the `ioctl()` function with the command SIOCGIFCONF to obtain a list of all network interfaces and their currently assigned parameters. Unfortunately this does not return the IPv6 addresses which are currently assigned to the interfaces.

Class `siputil` was extended in the way that it now holds also a FQDN for IPv6 which is the textual representation of an IPv6 address. This IPv6 FQDN is obtained by a new function, `findFqdn6()`, which obtains a list of all existing IPv6 addresses in the system by parsing the kernel supplied file /proc/net/if_inet6. An example of this file is shown in the Table 3 below:

20010638080620010250 04fffe56196b	03	40	00	00	eth1
00000000000000000000000000000001	01	80	10	80	lo
fe8000000000000025004fffe56196b	03	0a	20	80	eth1
fe8000000000000020476fffe0bb85b	02	0a	20	80	eth0

Table 3 Example entries of the `/proc/net/if_inet6` file

Thus the first column is the IPv6 address and the last column is the interface name it is assigned to. The loop back device (lo) is also listed but filtered out by the detection loop as it is for IPv4, too. This method is very specific to Linux kernel version 2.4, but there seems to be no easy way to do this detection.

The function `getLocalFqdn()` still returns the IPv4 FQDN as before while the function `getLocalFqdn6()` returns the IPv6 FQDN.

5.4.1.1.2 Class MessageSocket

The `MessageSocket` class is a base class which inherits host name lookup functionality, i.e. to resolve a symbolic name like "`xhosa.mobis.ip6`" to an IPv4/IPv6 address like "`2001:0638:0806:2001:0250:04ff:fe56:196b`" which then can be used for sending a message over a socket. It also defines the interface a derived class must implement in order to transport messages over a defined transport protocol. In this case this will be UDP.

In its original form this class used the function `gethostbyname()` to resolve a symbolic host name and stored its IPv4 address in an instance variable. `Gethostbyname()` does not support IPv6 address lookups. That's why it is not used anymore. The successor of it is the

function `gethostbyname2()` which allows to pass the desired address family as a parameter.

In the new form of the class a variable is introduced which marks this socket as an IPv4 or IPv6 socket. The host name lookup process is depicted in the following flow diagram (Figure 12).

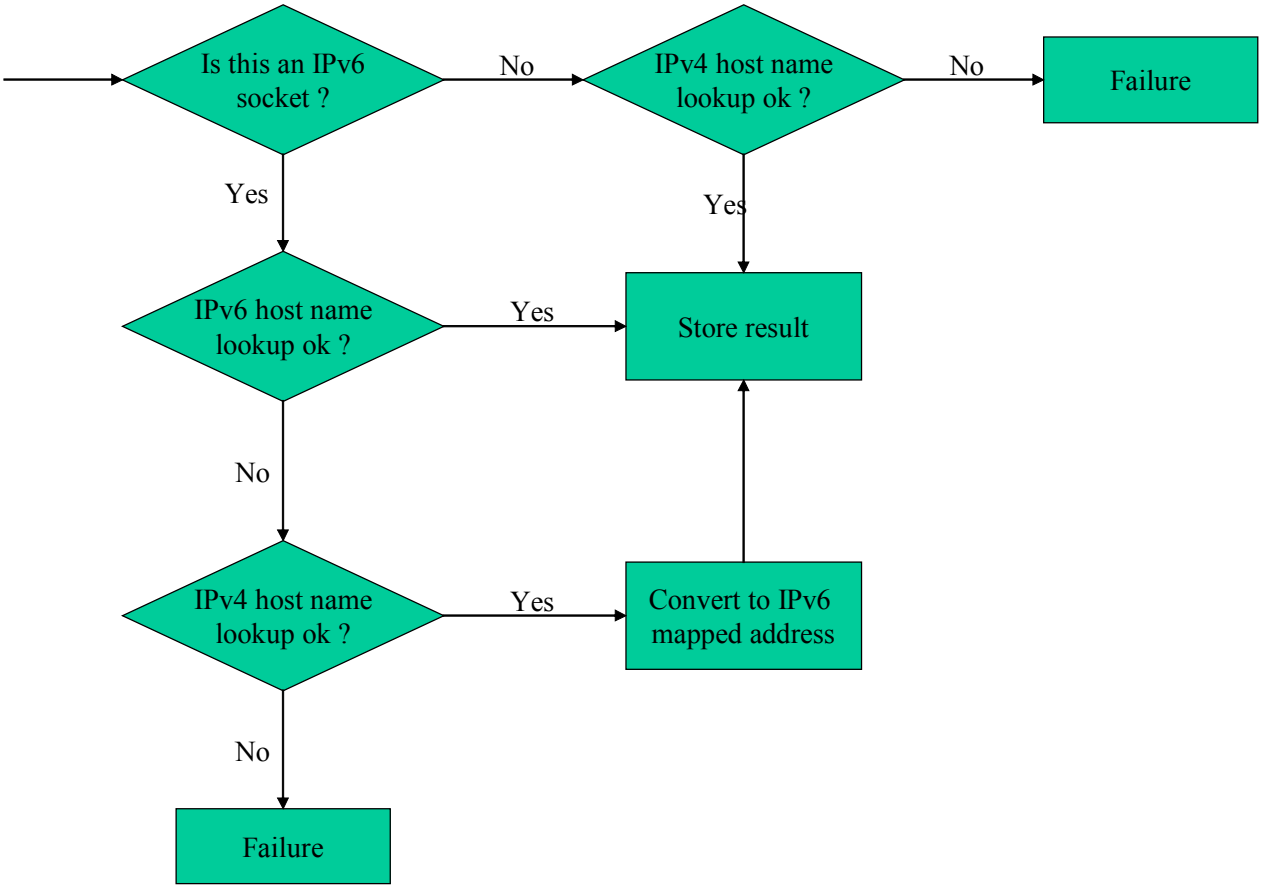

Figure 12 Name lookup process

Another extension to the MessageSocket class is the capability to simply return the address family in which a host name lookup was successful, without storing the resulting address. This is needed by some higher classes in order to decide their behaviour when constructing SIP messages.

5.4.1.1.3 Class UDPMessageSocket

This class is directly derived from the MessageSocket class. Its purpose is to send and receive messages over the UDP transport protocol.

The constructor function `UDPMessageSocket::UDPMessageSocket(void)` creates an unbound UDP socket. It tries to do this in the PF_INET6 protocol family first. On failure it falls back to the PF_INET (which is IPv4). In both cases the socket address family indicator from the inherited MessageSocket class is set appropriately. The port number space of UDP

over IPv4 is basically shared with the port number space of UDP over IPv6. This means that one can receive IPv4 originated messages at an IPv6 socket. In order to do so, one must bind to the empty IPv6 address, which consists of 128 zero bits. This is the receiver's counterpart to the feature of sending to IPv4 addresses using IPv4-in-IPv6 mapped addresses. Thus we can use an IPv6 socket for all communication if it is available. If it is not, the (old) IPv4-only version of KPhone applies.

The changes in the remaining member functions are quite similar to each other, because these functions are basically wrapping functions for the known networking system calls like `send(), recv(), bind(), listen(),` etc. So the general scheme looks like this:

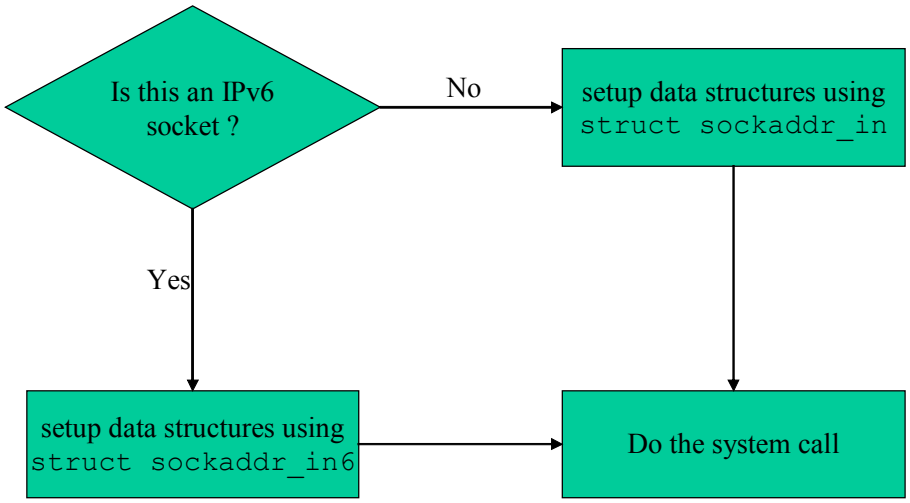

Figure 13 System call wrapping scheme

5.4.1.1.4 Class SdpMessage

The SdpMessage class holds the information about the local media engine. It has the IPv4/IPv6 address and port number where the RTP data is to be sent to by the other call participants. The SdpMessage class is responsible for encoding and decoding SDP conform message bodies from SIP messages. This is why it needed to be updated for IPv6. Table 4 shows the important SDP fields which hold the address and port number to use and how they must look like for IPv4, and IPv6, repectively.

IPv4:	IPv6:
`c=IN IP4 192.168.24.177` `m=audio 4712 RTP/AVP 0`	`c=IN IP6 2001:0638:0806:2001:0250:04ff:fe56:196b` `m=audio 4712 RTP/AVP 0`

Table 4 Example of SDP field entries

5.4.1.1.5 Class SipVia

Each instance of the SipVia class holds one "via" entity from a SIP message. Each via header field in SIP represents a SIP capable host, which can be an end system or a proxy, that was passed by this request message. Thus the list of vias increase with each visited host. When the request message reaches its final destination a response is generated. The response message travels the same way back to the originator of the request as instructed by the via list. Typical via header fields for IPv4 and IPv6 are shown below.

```
Via: SIP/2.0/UDP 192.168.56.177:5060
Via: SIP/2.0/UDP [2001:0638:0806:2001:0250:4ff:fe56:196b]:5060
```

Table 5 Via header entries

It is quite normal that a via stack holds entities with different address family types, as it is possible that SIP messages travel over intermediate networks of a different address type.

Another concept for writing down IPv6 addresses is introduced with this, the IPv6 reference. As in other level 5 protocols, it is necessary not only to notate an address but also an associated port number. The usual notation for this is *host:port* which works fine for hosts of the form IP-address or a host name. With an IPv6 address the problem arises that the colon is used as a separating token for the single fields of the IPv6 address too. This would not mean any trouble if the number of hex-numbers in an IPv6 address would be constant (8), but it is not. It is possible to write an IPv6 address with the zero-compressing abbreviation token :: (double-colon), which stands for as many zero fields as necessary to reach the 128 bit address size. So an IPv6 address has a variable number of fields and so a variable number of colons in it. So it is necessary to find a delimiting mechanism to encapsulate the IPv6 address in every case where it could be followed by a port number. This is nearly everywhere in SIP except for the "received" tag and SDP bodies. The following list shows different valid notations for one and the same address, the local loop back address.

```
0000:0000:0000:0000:0000:0000:0000:0001
0:0:0:0:0:0:0:1
::1
::0:0:1
0:0::0:1
0::1
```

Table 6 Different notations for one and the same IPv6 address

Modifications were made to be able to parse and encode IPv6 references as well.

5.4.1.1.6 Class SipViaList

The SipViaList class holds the via stack list which was introduced in section 5.4.1.1.5. It was extended in order to be able to fetch the bottom most via entity as well as the top most via entity. Usually, end-devices like KPhone only need to interpret the top most via entity in order to check them and to know where responses need to be sent to. In our case it can be necessary to read the bottom most via entity to know the address type of the originator of a request. This is the case if a SIP "INVITE" request is received which does not hold any SDP information, which conforms to RFC 3261 (cf. 5.4.1.2).

5.4.1.1.7 Class SipUri

This class represents a unified resource identifier (URI) for SIP. A SIP-URI is used in many header fields and identifies not only an interface and port number, but also a user, optional password, tags and parameters. Basically it is used whenever a user is to be addressed. The main header fields where URIs are found are the request header line and the "To", "From" and "Contact" header fields.

The changes that were done to the SipUri class are similar to the SipVia class. This means parsing and generation of the stored information. Table 7 records some typical SIP-URIs.

```
sip:joe:xylomorph@Freddie.Krueger.org:5060
sip:jack@[2001:34e4::19b6]:5062;origin=super;level=5
sip:horst@iptel.org;account=456?hval=56
```

Table 7 Example SIP-URIs

5.4.1.1.8 Class Sip

The Sip class provides constants for request method enumeration and functions to retrieve the local address which is the FQDN introduced in section 5.4.1.1.1. This function retrieves the FQDN from the SipUtil class. The appropriate function for the IPv6 FQDN was added in the same fashion in order to provide a clean interface to the application.

5.4.1.1.9 Class SipClient

The SipClient class deals with sending/receiving SIP messages and dispatching them to the correct Calls and their transaction managers. On startup the SipClient class detects the local FQDNs by using the mechanism as explained in 5.4.1.1.1. It constructs the SIP contact address for IPv4 and another one for IPv6. A SIP contact is basically a SIP-URI describing the point of contact for subsequent requests for this phone. The contact always refers to the local SIP user agent, while the URI in the "To" or "From" field can refer to a symbolic domain address like "jef@iptel.org".

The SipClient class is also responsible for building the SIP messages and header fields for outgoing requests and responses. When generating a request, the function `sendRequest()` is used, which works as follows: First the header fields are constructed with the appropriate values. After this, the target is determined and the message is sent to that target over a MessageSocket. Setting the header fields includes setting the "via" and "contact" header fields. Both can be either an IPv4 or an IPv6 address. In order to choose the right one, one must determine the target prior to setting the header fields. The via header field must correspond to the address family of the next hop, which can be an outbound proxy or the final target of the request, while the contact header field corresponds to the target end system only. The modifications done are depicted in Figure 14.

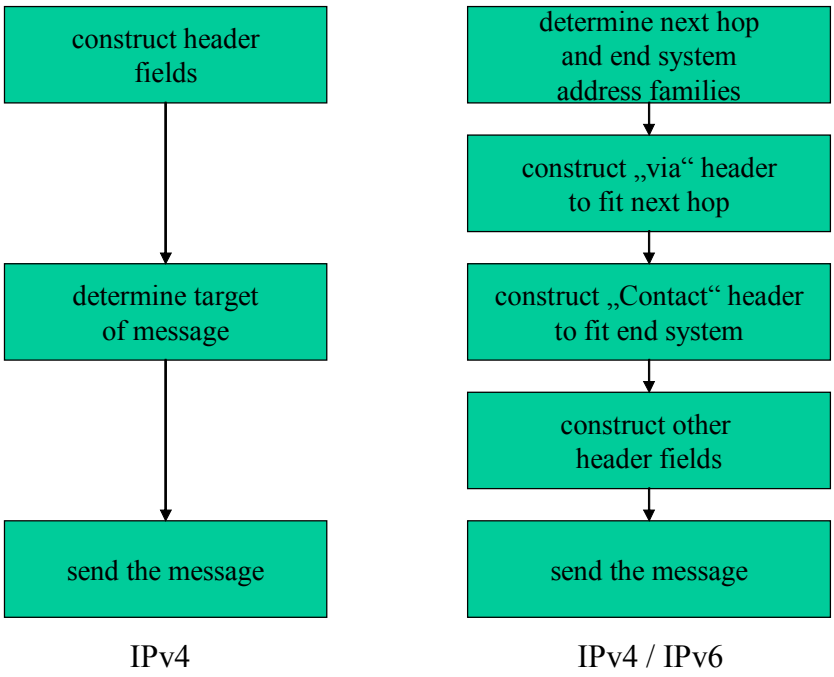

Figure 14 Request generation

When sending responses, only a contact header field may be inserted. To determine the correct FQDN to present (IPv4 or IPv6) it is necessary to look at the bottom most via header field of the request that triggered this response. The lowermost via header field identifies the originator of the request. Because the contact header field is to be valid for the end system, its address family must match the one of the end system. So the address family for the contact header field is the same as the address family of the lowermost via header field.

5.4.1.2 KPhone application classes

In the KPhone application only two classes were modified. Nevertheless, the modifications involve significant changes in protocol behaviour, as they use the previously introduced mechanisms.

5.4.1.2.1 Class CallAudio

The CallAudio class represents the media engine that sends and receives RTP encoded audio streams. Its underlying classes use the previously described UDPMessageSocket class to do the networking. Thus there are no changes required in these low-level classes. The member function `listenForRtp()` returns an SdpMessage object as described in section 5.4.1.1.4. To be able to encode the correct IPv4/IPv6 address in the SdpMessage object, it needs to know the address family of the communication partner. This information must be supplied by a higher instance in the application. To support this a new member function was introduced, `setIpType()`, which sets the target address family to the CallAudio object.

5.4.1.2.2 Class KCallWidget

This is the highest class that this application extension deals with. It represents the graphical element that enables the user to initiate and accept calls. It was modified for two aspects: initiating calls and accepting incoming calls. In both cases it is necessary to call the `listenForRtp()` function from the CallAudio class to obtain an SdpMessage object that

can be shipped to the remote side of the call. The methods of taking this decision whether to use the IPv4 or IPv6 address for signalling are different for incoming calls and for outgoing (initiated) calls. The following Figure 15 illustrates the decision process.

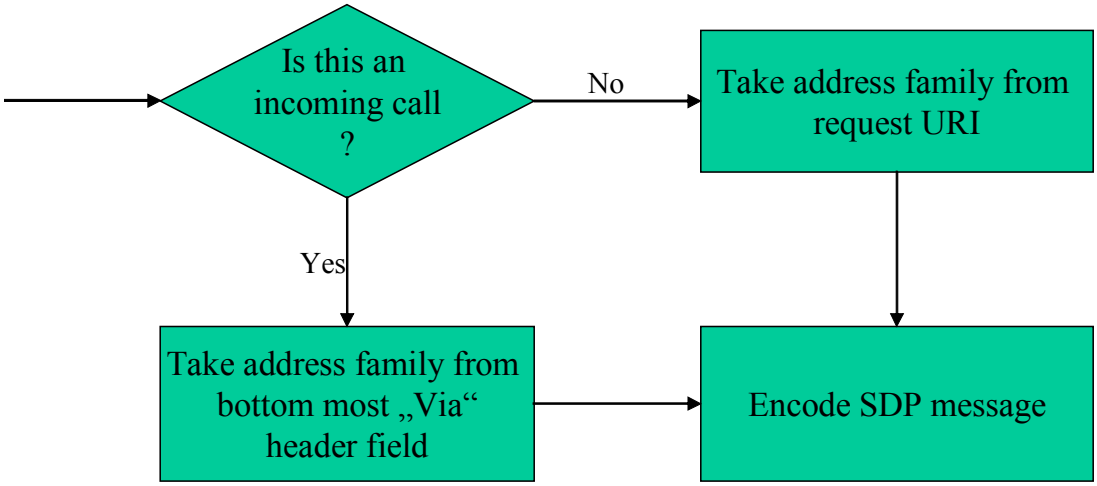

Figure 15 SDP message selection

5.4.1.3 Basic testing and outlook

Basically two test scenarios have been investigated:

1. Direct addressing: Two instances of KPhone running on different IPv6 hosts residing in a common network.
2. Proxy mode: The first scenario extended by an additional SIP proxy and registrar server.

5.4.1.3.1 Direct addressing

In this scenario two machines are involved. The call is initiated from one machine which addresses the other machine directly in the SIP request URI. The scenario and the message flow are depicted in Figure 16 below.

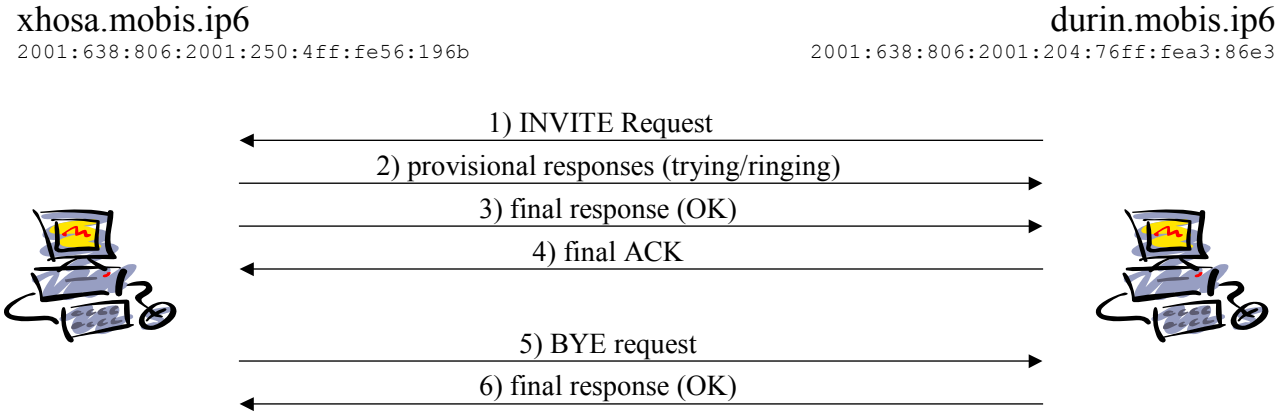

Figure 16 Message flow in case of direct addressing

Messages 1-4 are initializing the call while messages 5-6 are terminating the session.

5.4.1.3.2 Proxy mode

In the proxy mode there is a SIP proxy/registrar server located between the caller and the callee. The software in use for this is the SIP express router (SER[15]). The scenario is depicted in the following Figure 17.

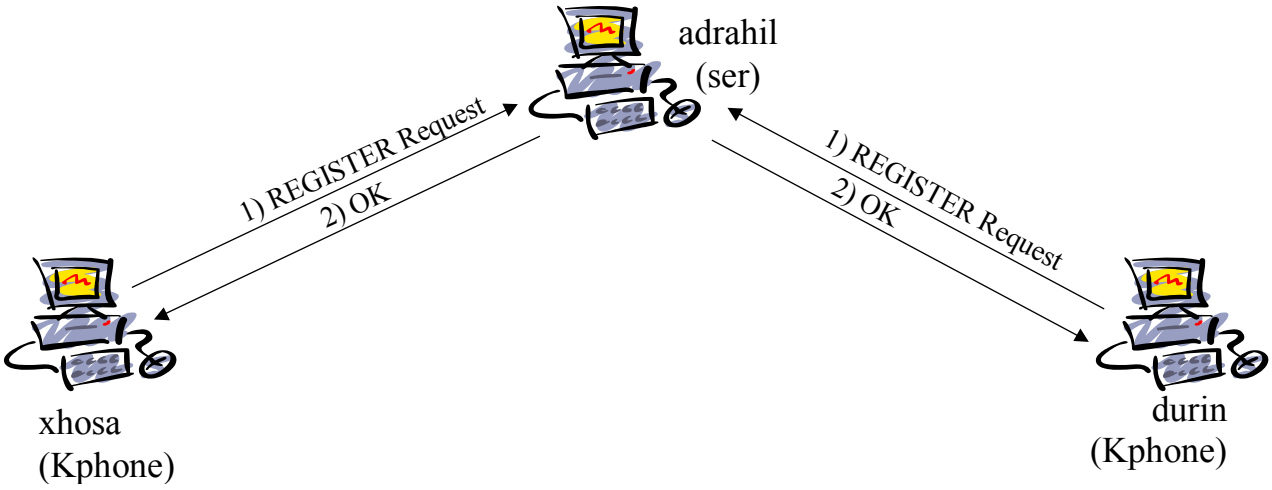

Figure 17 Message flow in proxy mode

Before a call can be established, both parties need to register with the SIP registrar server. The message flow for a call setup is shown in Figure 18 below.

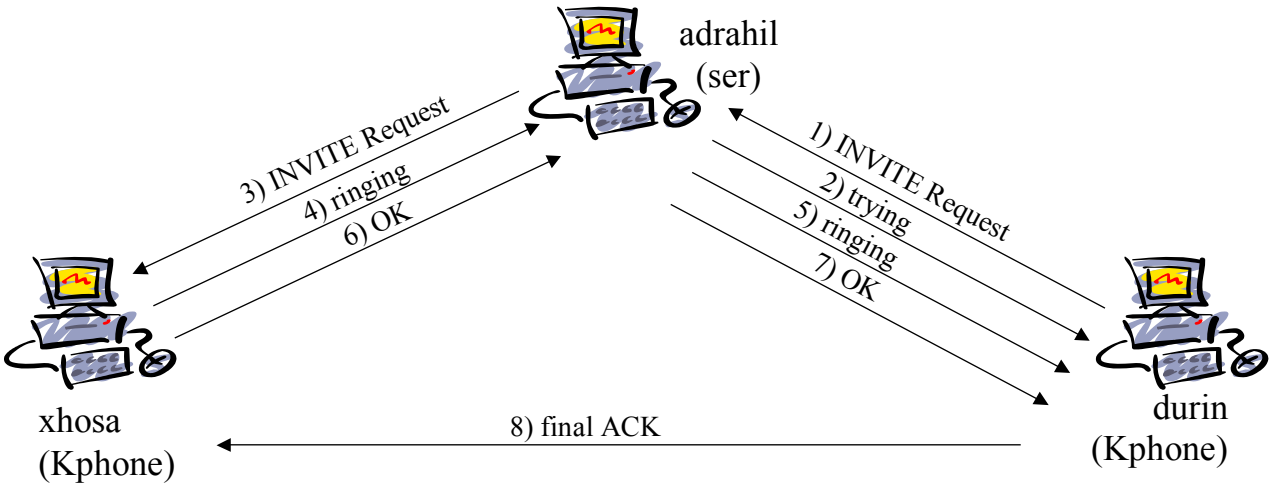

Figure 18 Message flow for call setup

5.4.1.3.3 Outlook

It is intended to incorporate the KPhone extensions described in the official code tree at the Wirlab[16] CVS.

[15] Version 0.8.10 and version 0.8.11. Download from iptel.org.

5.5. 6VOICE

The Swiss company Telscom has developed another Linux based VoIPv6 application called 6VOICE. The system uses the Java Runtime Environment and the Java Media Framework. It was tested with redhat 7.2 and SuSE 8.0 with kernel 2.4.18.

Little documentation is available. Testing has been done across fixed and mobile networks. In particular the following tests have been mentioned:

- Cross Platform testing across redhat 7.2 and SuSE 8.0.
- Cross network testing across fixed and Mobile IPv6 (HUT[17] MIPv6 implementation).
- Testing over public internet between Bern and Brussels (NGNLAB[18] testbed).

6. Interoperability of SIP Implementations

During SIPit 13[19] various interoperability tests had been executed with FhG Fokus / iptel's SIP Express Router ([20]) involved. Because SIPit events are covered by NDA, it is not possible to reveal specific information about experiences with other implementations in test.

There had been individual and multi party tests as well.

6.1. Individual Tests

Three implementations supporting IPv6 had been tested. The testing scenario included proxy servers and agents, SIP registration and deregistration and establishing calls over IPv6.

A very first test of TLS [6] over IPv6 was also done, and it worked fine. The testing scenario included SIP registration and deregistration over IPv6 and sending a SIP MESSAGE through the SER proxy server. REFER [23] over IPv6 was also tested. After minor software modifications REFER worked well.

All problems detected could be resolved on site and testing could be resumed.

6.2. Multi-party Tests

Multi-party tests included spiral tests. In such scenario several proxies are placed in chain forwarding requests to each other. Different IPv6 spirals had been investigated., involving 4 parties in total, iptel one of them. Pure IPv6 spirals mostly worked. Heterogeneous settings including IPv6 and IPv4 conversion had been tested, too. With this configuration iptel's SER was acting as a proxy at the boundary responsible for IPv4/IPv6 protocol translation.

[16] Wirlab, located in Seinäjoki, Finland, is a research center that does research in future networking real-life application environments (www.wirlab.net).
[17] Helsinki University of Technology, Finland
[18] Next Generation Networks Laboratory is a project started in January 2001 in the scope of the European Commission "Information Society Technologies" (IST) programme.
[19] SIPit 13, 2003-08-18 to 22, Ottawa, Canada
[20] http://www.iptel.org/ser/

One observation from these tests is that SIP Messages containing IPv6 addresses are pretty big. Each message can contain several Via, Route and Record-Route header fields and each of the header fields contains an IPv6 address bringing about very big messages.

Overall, there was a couple of working IPv6 implementations and most of the things under test worked well. There was obviously significant progress compared to test results from previous SIPit conferences when it was hardly possible to schedule a multi party testing due to lack of implementations.

7. ENUM

ENUM is a protocol, defined in RFC2916 [7], for mapping between telephone numbers (e.164 format) and the DNS. RFC2916 is currently under modification to an RFC2916bis (currently an I-D exists for *draft-ietf-enum-rfc2916bis-06* [8]).

In it's simplest form, a number of the form

```
+44 1234 567890
```

is mapped into the DNS under the e164.arpa hierarchy as:

```
0.9.8.7.6.5.4.3.2.1.4.4.e164.arpa
```

This mapping allows a DNS hierarchy of (national) registrars to generate DNS records corresponding to well-known telephone numbers. The DNS record used to store ENUM services is the NAPTR record. The NAPTR records may list one or more ENUM services, with preferences, for any such number. These might be SIP, H.323 or e-mail services. In the context of an ENUM-aware application, the application would need to look up the NAPTR record for a target number, and determine whether a SIP entry existed, and if it did initiate the call to the listed name.

The example listed in RFC2916 shows records for sip:, h323: and mailto: services:

```
$ORIGIN 4.3.2.1.6.7.9.8.6.4.e164.arpa.

IN NAPTR 10 100 "u" "E2U+sip"

"!^.*$!sip:info@example.com!" .

IN NAPTR 10 101 "u" "E2U+h323"

"!^.*$!h323:info@example.com!" .

IN NAPTR 10 102 "u" "E2U+msg:mailto"

"!^.*$!mailto:info@example.com!" .
```

The resolution method then depends on the protocol. An ENUM application may be able to use multiple protocols, and thus (for example) fall back to offer the user the option to send an email or instant message if the recipient is not available for a SIP voice call.

VOCAL had ENUM support as of version 1.4. However while the code remains in the CVS it is not in the v1.5 release code (Cisco removed it through lack of use, which with hindsight may not be the best way to get the code used!). We hope to reintegrate the ENUM code within the 6NET project.

8. Impact of 3G networks on integration of SIP and IPv6

Major driving forces behind next generation All-IP multimedia architecture are technologies such as SIP, MEGACO and IPv6. Future networking with embedded and mobile devices and particularly the advent of SIP-based peer-to-peer services demand for IPv6 for addressing the multitude of devices because peer-to-peer services work well only with public IP addresses.

A key development for future mobile communication is the IP Multimedia System technology (IMS) standardized by 3GPP. IMS enables the establishment of popular person-to-person IP connections between terminals based on IETF protocols such as SIP, IPv6, DIAMETER, COPS, etc. where IMS adopts the ability to find a person via SIP to initiate a session.

The IMS infrastructure is based on the Session Initiation Protocol and allows for easy integration with other IP protocols and applications, uncomplicated integration of voice, images, video and other interactive communications services, as well as seamless service provision comprising various access networks. Actually SIP builds the core of future IMS and Mobile Internet service architectures.

IMS is a technology standardized by 3GPP standardization organization in 3GPP Release 5 and 6 and 3GPP2 specifications. SIP as a protocol is standardized by the IETF (Internet Engineering Task Force), while 3GPP is standardizing the way SIP is used in mobile networks, such as listed in Table 11 of Appendix A.

9. Conclusion

While IPv6 is considered as the core protocol for next generation networks, the session initiation protocol designed within the IETF is constantly gaining in popularity and acceptance as the signalling protocol for next generation multimedia communication.

Today all Operating Systems support IPv6, including Windows XP and CE .Net, the Pocket PC OS for PDA. All major vendors of routers currently support IPv6 providing a clear indication that the deployment of IPv6 is changing from trial to operational systems in some markets. Asia, and here especially Japan, ahead both Europe and the US, is the major driver for the commercial deployment of IPv6 networks due to an imminent shortage of IPv4 addresses.

At the same time the interest for the deployment of SIP with IPv6 is gaining attraction. Although there are currently more than 100 vendors on the market, including Dynamicsoft, Cisco, Nokia, Ericsson, Lucent, Nortel, Samsung and Microsoft, offering SIP related products ranging form core network SIP and presence servers, software clients, hardware phones to application level gateways, and residential gateways supporting IPv4, the understanding and deployment for SIP and IPv6 together is still less common.

In particular there is a need to address the challenges that arise when initiating person-to-person communication in both IPv4 and IPv6 networks:

- naming and addressing using DNS and ENUM
- security
- billing
- 6to4 access
- enterprise case
- residential case

Although SIP does inherently support IPv6 there is an actual demand for gaining experience of SIP in IPv6 networks in order to verify proper operation and to investigate IPv4 related interoperability issues such as

- SIP deployment in native IPv6 networks
- SIP 6to4 deployment
- PSTN gatewaying
- dual stack client functionality
- autoconfiguration
- SIP/H.323 interoperability
- NAT removal

An area where SIP and IPv6 have an important role is VoIP. As an alternative to legacy telephony VoIP promises a significant business potential. VoIP not only offers voice services but also a rich suite of multiple IP-based value-added services. These services include advanced supplementary services, real-time video, presence, instant messaging, and application data exchange. Development and deployment of SIP and H.323 in the IPv4 world show a steady upturn in both markets. Market surveys show that VoIP vendors will continue to support SIP product lines as well as H.323.

In the context of the application trials to be conducted in workpackage 5 different SIP based VoIP applications will be evaluated:

- The Vocal system has already been run between at least two 6Net partners informally. It includes a SIP based Redirect Server, Feature Server, Provisioning Server and Marshal Proxy. The system also includes two SIP user agents. The SIP user agents have been ported for use with IPv6. The University of Southampton is planning to demonstrate ENUM working with VOCAL.
- Bonephone, a SIPv6 capable user agent, is part of a SIP infrastructure based on a high performance, configurable, free SIP server (SER).
- SER is an open-source project that aims to make available a fully functional and scalable SIP server. It can act as registrar, proxy or redirect server.
- KPhone is a Linux based Voice-over-IP telephony application which supports IPv4, and IPv6 as well. It is based on the *K desktop environment* (KDE).

The user community for these application comprise the WP5 partners, mainly for testing the application. Additionally, the application will be advertised to the population of 6NET-connected users. The SER SIP server will be available for use with other SIP based applications to support interoperation testing between different 6NET trial applications.

Another domain where SIP and IPv6 have important roles to play is in the 3G networks. Both, SIP and IPv6, are defined mandatory in release 5 of 3GPP. So SIP will be used as the signalling protocol in the IP Multimedia Subsystem (IMS) of the 3rd Generation Partnership Project (3GPP) UMTS networks. 3GPP IP Multimedia Subsystem (IMS) will be a major driving force for IPv6 because IPv6 offers simple, easy access to the new services. Nokia and other infrastructure vendors support it in their IP Multimedia Subsystem. Infrastructure product suites are expected in 2003, followed by IMS capable terminals in 2004.

10. References

[1] Berners-Lee T., Fielding R., Masinter L., "**Uniform Resource Identifiers (URI): Generic Syntax**," RFC 2396, IETF, August 1998

[2] Campbell, B. (Ed.), Rosenberg J. Schulzrinne H., Huitema C., Gurle D., "**Session Initiation Protocol (SIP) Extension for Instant Messaging**," RFC 3428, IETF, December 2002

[3] Davidson J., Peters J., Gracely B., "**Voice over IP Fundamentals**," Cisco Press, March 2000, ISBN 1578701686

[4] Deering S., "**Host extensions for IP multicasting**," RFC 1112, IETF, August 1989

[5] Deering S., Hinden R., "**Internet Protocol Version 6**," RFC 1883, IETF, December 1995

[6] Dierks T., Allen C., "**The TLS Protocol ,**" RFC 2246, IETF, January 1999

[7] Faltstrom P., "**E.164 number and DNS**," RFC 2916, IETF, September 2000

[8] Faltstrom P., Mealling M., "**The E.164 to URI DDDS Application (ENUM)**," draft-ietf-enum-rfc2916bis-06, IETF, May 2003

[9] Gilligan R., Nordmark E., "**Transition Mechanisms for IPv6 Hosts and Routers,**" RFC 2893, IETF, August 2000

[10] Handley M., Jacobson V., "**SDP: Session Description Protocol**," RFC 2327, IETF, April 1998

[11] Hinden R., Carpenter B., Masinter L., "**Format for Literal IPv6 Addresses in URL's**," RFC 2732, December 1999

[12] Holdrege M., Srisuresh P., "Protocol complications with the IP network address translator (NAT)," RFC 3027, January 2001

[13] Mills D., "Network Time Protocol (Version 3) Specification, Implementation," RFC 1305, IETF, March 1992

[14] Ott J., Perkins C., Kutscher D., "**A Message Bus for Local Coordination**," RFC 3259, IETF, April 2002

[15] Postel J., "**Internet Protocol**," RFC 791, IETF, September 1981

[16] Postel J., "**Transmission Control Protocol**," RFC 793, IETF, September 1981

[17] Postel J., "**User Datagram Protocol**," RFC 768, IETF, August 1980

[18] Rosenberg J., "**A Framework for Conferencing with the Session Initiation Protocol**," draft-rosenberg-sipping-conferencing-framework-01, IETF, February 12, 2003

[19] Rosenberg J., Mahy R., Sen S., "**NAT and Firewall Scenarios and Solutions for SIP**", draft-ietf-sipping-nat-scenarios-00, IETF, June 24, 2002

[20] Rosenberg J, Schulzrinne H., Camarillo G., Johnston A., Peterson J., Sparks R., Handley M., Schooler E., "**SIP: Session Initiation Protocol**," RFC 3261, IETF, June 2002

[21] Schulzrinne H., "RTP Profile for Audio and Video Conferences with Minimal Control," RFC 1890, IETF, January 1996

[22] Schulzrinne H., Casner S., Frederick R., Jacobson V., "**RTP: A Transport Protocol for Real-Time Applications**," RFC 1889, IETF, January 1996

[23] Sparks R., "The Session Initiation Protocol (SIP) Refer Method," RFC 3515, April 2003

[24] Srisuresh P., Holdrege M., "IP Network Address Translator (NAT) Terminology and Considerations," RFC 2663, August 1999

11. Abbreviations

3GPP	3rd Generation Partnership Project
CPL	Call Processing Language
DNS	Domain Name System
IMP	Instant Messaging and Presence
MEGACO	Media Gateway Controller
MGCP	Media Gateway Control Protocol
NAT	Network Address Translation
NDA	Non-Disclosure Agreement
PGW	SIP Protocol Gateway
PSTN	Public Switched Telephone Network
PTT	Push-to-talk
RTP	Real-Time Protocol
SDP	Session Description Protocol
SER	SIP Express Router
SIMPLE	SIP for Instant Messaging and Presence Leveraging Extensions
SIP	Session Initiation Protocol
SIPDev	SIPtech Application Development Framework
SS7	Signalling System 7
TLS	Transport Layer Security
TRIP	Telephony Routing over IP
UA	User Agent
UDP	User Datagram Protocol
UoS	University of Southampton
URI	Uniform Resource Identifiers
VOCAL	Vovida Open Communication Application Library

Table 8 Abbreviations

12. Glossary

callee	SIP entity that is requested to participate to a session by receiving an

	INVITE message.
caller	SIP entity that initiates a session by sending an INVITE message.
decapsulation	Process of unpacking an encapsulated datagram (▸ encapsulation) at the end of a ▸ tunnel.
encapsulation	Process of enveloping a datagram in another (opposite operation: ▸ decapsulation). This operation is done at the entry of a ▸ tunnel
proxy server	A proxy server receives a request and then forwards it towards the current location of the callee -either directly to the callee or to another server that might be better informed about the actual location of the callee.
push-to-talk (PTT)	Wireless service allowing cell phones to act like long range walkie-talkies. Instead of *calling* someone, one simply has to push a button and can talk to a person instantaneously.
redirect	A redirect server receives a request and informs the caller about the next hop server. The caller then contacts the next hop server directly.
redirect server	A server that accepts SIP requests, maps the SIP address of the called party into zero (if there is no known address) or more new addresses and returns them to the client. Unlike proxy servers, redirect servers do not pass the request on to other servers.
registrar server	To assist SIP entities in locating the requested communication partners SIP supports so called register servers. The register server is mainly thought to be a database containing locations as well as user preferences as indicated by the user agents. It accepts register requests and places the information it receives in those requests into the location service for the domain it handles. It is typically colocated with a proxy or redirect server.
TCP	*Transmission Control Protocol* – application protocol for connection oriented protocols, e.g. HTTP, FTP, Telnet.
tunnel	The path followed by a datagram while it is ▸ encapsulated.
UA client	This client resides on the host where the caller is located. It is capable of initiating outgoing calls according to the user's actions.
UA server	This server resides on the host where the callee is located. It is capable of querying the user about what to do with the incoming call, i.e., accept, reject or forward and sending the response back to the caller.
UDP	*User Datagram Protocol* – application protocol of the ▸ IP protocol family for datagram based applications, e.g. RPC, Domain etc.
user agent (UA)	A logical entity in the terminal equipment that can act as both a user agent client and user agent server.

Table 9 Glossary

13. SIP Activities and Sites

The following overview gives a list of (deep) links related to SIP, VoIP, and IPv6.

(Status of August 03)

Description	URL	v6
Working Groups / Standards lists:		
Official charter of the 'Session Initiation Protocol (SIP)' working group	**www.ietf.org/html.charters/sip-charter**	

Session Initiation Protocol (SIP) Working Group Supplemental Home Page	www.softarmor.com/sipwg/	
Official charter of the 'Session Initiation Protocol Project INvestiGation (SIPPING)' working group	www.ietf.org/html.charters/sipping-charter	
Official charter of the 'SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE)' working group	www.ietf.org/html.charters/simple-charter	
Official charter of the 'Instant Messaging and Presence Protocol (impp)' working group	www.ietf.org/html.charters/impp-charter	
IETF Draft Tracker	datatracker.ietf.org/public/pidtracker	
An overview of current and expired SIP / SIPPING related drafts	www.softarmor.com/wgdb/	
Forums:		
SIP Forum, promoting SIP products and -service	www.sipforum.org	
HotSip, products for SIP, -presence services and -infrastructure	www.hotsip.com	
iptel	www.iptel.org	*
Henning Schulzrinne's SIP site at Columbia university	www.cs.columbia.edu/~hgs/sip/	
VOIP FORUM	www.voip-forum.com	
GecKo's IP xStream - IP telephony news and market research	www.iptelephony.org	
Conferences:		
SIPit 12	www.hotsip.com/sipit12/	*
SIPit 13, 2003-08-18 to 22, Ottawa, Canada	www.mitel.com/sipit/	*
SIPit 14, 2004-02-08 to 13, Cannes, France		*
SIPit 15, 2004-09, Taiwan		*
Fall 2003 VON, Boston, MA, September 22-25	www.pulver.com/von/	*
SIPop! 2003, colocated with Fall 2003 VON	www.pulver.com/sipop/	
Spring 2004 VON, Santa Clara, CA, March 29 – April 1		*
NYSSA Telecom / VoIP Industry Conference, 2003-12-03, NYC	www.nyssa.org/events/120303-telecom-voip-industry-conference.html	
Internet Telephony Conference and Expo, Long Beach, CA, October 14-16, 2003	www.tmcnet.com/itexpo/	

The pulver conference calendar	www.pulver.com/conference/	
Companies:		
Accord Telecommunications -> Polycom	www.polycom.com/home/	
ACT Networks		
Alcatel		
AudioCodes	www.audiocodes.com/	
BreezeCOM		
FWD – Free World Dialup	www.freeworldialup.com/	
InfoInterActive	www.infointeractive.com/	
Natural MicroSystems	www.nmscommunications.com/	
NexTone Communications	www.nextone.com/	
Polycom®	www.polycom.com/home/	
Quicknet Technologies	www.quicknet.net/	
RADVISION toolkits	www.radvision.com/TBU/	*
Sonus Networks	www.sonusnet.com/	
Telogy Networks	www.telogy.com/	
VCON	www.vcon.com/	
VOVIDA (Cisco)	www.vovida.org	*
Wind River	www.windriver.com/	
Providers:		
Free SIP registrar service	www.iptel.org	*
SIP service providers	www.sipcenter.com/showcase/showserviceproviders	
Vonage digital voice	www.vonage.com/	
Germany's 1st IPv6 ADSL Internet provider	www.space.net/	*
Projects:		
Vocal	www.vovida.org/applications/downloads/vocal	*
Product sites:		
Bonephone IPv6 SIP phone	www.iptel.org/products/bonephone/	*
Ingate	www.ingate.com	
Intertex	www.intertex.se	
KPhone IPv6 SIP phone	www.iptel.org/products/kphone/	*
linphone	www.linphone.org/ or simon.morlat.free.fr/	*
more IP telephony products	www.iptel.org/info/products/	*
Simple Traversal of UDP through NATs (STUN server)	www.vovida.org/applications/downloads/stun/	

SIP Express Router (SER)	www.iptel.org/ser	*
SIP hard phones	www.iptel.org/info/products/index.php?category=hardphone&name=Hardphones&siponly=1	*
SIP Residential Gateway (SIPRG)	www.vovida.org/applications/downloads/siprg/	
SIP soft phones	www.iptel.org/info/products/index.php?category=softphone&name=Softphones&siponly=1	*
SIP software products	www.iptel.org/products/	*
SIP User Agent (SIPSet)	www.vovida.org/applications/downloads/sipset/	
SIPTiger is a web-based provisioning utility for Cisco's SIP-phones	www.vovida.org/applications/downloads/siptiger/	
SJPhone	www.sjlabs.com/	
Snom	www.snom.com	
SOFTFRONT-VoIP technology	www.softfront.co.jp/en/tech/ipv6.html	
The Vovida Open Communication Application Library (VOCAL)	www.vovida.org/applications/downloads/vocal/	*
6VOICE IPV6 SIP phone	www.telscom.ch/6voice/6voice_version.htm	*
VegaStream	www.vegastream.com	
VON's SIP product list	www.pulver.com/products/sip/	
FAQs:		
Columbia SIP FAQ	www.cs.columbia.edu/sip/faq/	
Others:		
ENUM refers to the IETF protocol that takes a complete, international telephone number and resolves it to a series of URLs using a Domain Name System (DNS)-based architecture.	www.enum.org	
Selected IETF Internet Drafts and RFCs Relevant to the Internet Telephony	www.ietf.org/html.charters/simple-charter.html	

*** provides IPv6 specific information**

Table 10 SIP activities and sites

14. SIP Related Standards Status

The master IETF status for SIP related items is available in the IETF Draft Tracker[21].

The tracker is especially useful for things that have been approved by the IESG and are in the RFC editor queue, or that have completed WGLC and are under IESG review.

An overview of current and expired SIP / SIPPING related drafts can be found under http://www.softarmor.com/wgdb/.

[21] https://datatracker.ietf.org/public/pidtracker

15. Appendix A

15.1. 3GPP standardization for IMS and SIP

Spec	ETSI No./ Reference	Version/ Achieved Date	Title
24.228	TS 124 228 RTS/TSGN-0124228v550	5.5.0 2003-07-10	Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS);Signalling flows for the IP multimedia call control based on SIP and SDP; Stage 3(3GPP TS 24.228 version 5.5.0 Release 5)
24.229	TS 124 229 RTS/TSGN-0124229v550	5.5.0 2003-07-02	Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS);IP Multimedia Call Control Protocol based on SIP and SDP; Stage 3(3GPP TS 24.229 version 5.5.0 Release 5)
29.998-4	TR 129 998-4-4 RTR/TSGN-0529998-04-4v500	5.0.0 2002-06-27	Universal Mobile Telecommunications System (UMTS);Open Service Access (OSA) Application Programming Interface (API) Mapping for Open Service Access; Part 4: Call Control Service Mapping; Subpart 4: Multiparty Call Control SIP(3GPP TR 29.998-04-4 version 5.0.0 Release 5)
23.228	TS 123 228 RTS/TSGS-0223228v590	5.9.0 2003-07-02	Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS);IP Multimedia Subsystem (IMS); Stage 2(3GPP TS 23.228 version 5.9.0 Release 5)
23.278	TS 123 278 RTS/TSGN-0223278v530	5.3.0 2003-07-01	Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS);Customised Applications for Mobile network Enhanced Logic (CAMEL) - IP Multimedia System (IMS) interworking; Stage 2(3GPP TS 23.278 version 5.3.0 Release 5)
29.278	TS 129 278 RTS/TSGN-0229278v520	5.2.0 2003-04-04	Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS);Customized Applications for Mobile network Enhanced Logic (CAMEL); CAMEL Application Part (CAP) specification for IP Multimedia Subsystems (IMS)(3GPP TS 29.278 version 5.2.0 Release 5)

29.328	TS 129 328 RTS/TSGN-0429328v540	5.4.0 2003-07-08	Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS);IP Multimedia Subsystem (IMS) Sh interface signalling flows and message contents(3GPP TS 29.328 version 5.4.0 Release 5)
32.225	TS 132 225 RTS/TSGS-0532225v530	5.3.0 2003-07-08	Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS);Telecommunication management; Charging management; Charging data description for the IP Multimedia Subsystem (IMS)(3GPP TS 32.225 version 5.3.0 Release 5)

Table 11 IMS and SIP related 3GPP standardization