SIPtel

Phase 1: Review and Preliminary Evaluation of SIP-based components for Voice over IP (VoIP)

NR Norwegian Computing Center
APPLIED RESEARCH AND DEVELOPMENT

RAPPORT/REPORT

Norwegian Computing Center / Applied Research and Development

Report nr. 947

Peter Holmes
Eirik Maus

Oslo
August 1999

NR Norwegian Computing Center
APPLIED RESEARCH AND DEVELOPMENT

Rapport/Report

Tittel/Title:	**Dato**/Date: August
SIPtel	**År**/Year: 1999
Phase 1: Review and Preliminary Evaluation of	**ISBN**: 82-539-0450-9
SIP-based components for Voice over IP (VoIP)	**Publikasjonsnr/**
	Publication no: 947

Forfatter/Authors:
Peter Holmes, Eirik Maus

Sammendrag/Abstract:

The motivating vision behind SIPtel is to determine to what extent it is possible to facilitate full-scale deployment of a SIP-based voice over IP (VoIP) service upon UNINETT's production network.

To initiate such a determination, the primary goal of Phase 1 within SIPtel project has been to test and evaluate a SIP-based video-conferencing framework for voice over IP (VoIP) purposes. In addition, other SIP-based components have been reviewed for consideration. Lessons learned from this Phase 1 exercise have formed the basis for conclusions and recommendations concerning SIPtel Phase 2.

Emneord/Keywords: SIP, VoIP, H.323, Voice on the net (VON), audio-video conferencing

Tilgjengelighet/Availability:	Open
Prosjektnr./Project no.:	SIPtel 28012
Satsningsfelt/Research field:	Internet telephony
Antall sider/No of pages:	48

Norsk Regnesentral / Norwegian Computing Center
Gaustadalléen 23, Postboks 114 Blindern, 0314 Oslo, Norway
Telefon 22 85 25 00, telefax 22 69 76 60, telex 76518 ncc n

SIPtel

Phase 1: Review and Preliminary Evaluation of SIP-based components for Voice over IP (VoIP)

Peter Holmes
Eirik Maus

Norsk Regnesentral
August 1999

Table of Contents

Executive Summary

SIPtel is a project funded by UNINETT [2]; at its conception, SIPtel was divided into two phases. This document reports the results from the first phase of the project.

The motivating vision behind SIPtel is to determine to what extent it is possible to facilitate full-scale deployment of a SIP-based voice over IP (VoIP) service upon UNINETT’s production network.

To initiate such a determination, the primary goal of Phase 1 within SIPtel project has been to test and evaluate a SIP-based video-conferencing framework for voice over IP (VoIP) purposes. In addition, other SIP-based components have been reviewed for consideration. Lessons learned from this Phase 1 exercise have formed the basis for conclusions and recommendations concerning SIPtel Phase 2; that is, pilot deployment of a SIP-based VoIP service upon UNINETT’s production network.

SIP (Session Initiation Protocol) [28] is an international standard from the IETF (Internet Engineering Task Force) [26]. SIP specifies an application-layer control (signaling) protocol for creating, modifying and terminating sessions with one or more participants. These sessions include Internet multimedia conferences, Internet telephone calls and multimedia distribution. Members in a session can communicate via multicast or via a mesh of unicast relations, or a combination of these.

Within SIPtel Phase 1, the primary testing and evaluation work planned to target assessment of Multimedia Internet Terminal (MInT) [5], from GMD FOKUS [4], upon the Linux platform. MInT is a flexible multimedia tool set that allows the establishment and control of multimedia sessions across the Internet. MInT offers a variety of familiar desktop conferencing applications, and its system architecture is fully distributed, with no central components.

Due to certain unexpected events encountered during the course of the testing work, project effort turned to include ”light” evaluation of two other SIP-related components: Bell Labs’ siphone client [12] and sipd [21], from the University of Columbia.

By the end of SIPtel Phase 1, MInT, siphone and sipd were installed upon both Linux and Solaris. The evaluation work performed in this phase has ultimately focussed upon cross-platform and cross-component interoperation, rather than upon thorough critique of each component's functional performance.

MInT can be used on Solaris in the near-term to gain greater familiarity with SIP-based, multimedia conferencing. It can presumably be used to perform interesting and informative tests and experiments with RSVP, as well. A long-term ”commitment” to *the currently available version* of MInT is not recommended, however, since the system is based upon a pre-standardized version of SIP. Should MInT be updated to conform with the SIP standard, it could provide a long-term basis for a functionally-rich conferencing system, as well as a research vehicle for experimental development.

Bell Labs' siphone and Columbia's sipd both conform to the SIP standard. siphone appears to be a good candidate for further work with SIP-based VoIP. In this regard, it is best suited as a computer-based substitute for (or complement to) a standard telephone, rather than a vehicle for research and experimentation.

Though we had certain problems in getting sipd to operate correctly at our site, sipd is judged to be a good candidate for further work within SIPtel. It can serve to support efforts targeting the development of a simple SIP-based telephone for common consumers, as well as efforts targeting the development of SIP-based research environments.

1. Introduction

1.1 Background

The latter part of this decade has born witness to a increasing number of large-scale efforts to tackle the inevitable convergence of data- and tele-communication technologies. A significant percentage of those efforts has been devoted to the creation and implementation of standards which bridge these technologies, especially in regard to IP-based telephony. The justification for this particular technical focus lies in the *enormous* commercial potential for end-user solutions and services which can be built upon platforms and products which support IP telephony. Quite simply, IP telephony can be used as an explicit or embedded component within advanced multimedia solutions, services and products targeted for the common consumer.

Over the last few years, two standards for IP-based telephony have emerged: these are H.323 [25] and SIP [28]. From the ITU [24], H.323 represents the telecommunication sector's effort to bring together protocols for advanced audio-video-data communication within a unifying standard framework, in order that multimedia telephony can be realized upon a data network. In contrast, SIP is a product of the IETF's [26] working group on Multiparty Multimedia Session Control (MMUSIC) [27]. SIP represents the IETF's effort to define an Internet standard through which multimedia sessions (”calls”) can be initiated and handled in a standard manner. Simply stated, the SIP standard specifies an application-layer control (signaling) protocol for creating, modifying and terminating sessions with one or more participants.

1.2 Project Actors

UNINETT has funded and helped NR carry out the SIPtel project. UNINETT is the Norwegian academic network for research and education [2], and an active proponent in the deployment of Internet-based technology within Norway.

Prior to the SIPtel project, NR has been involved in other work related to IP telephony. The first project of this kind was IMiS-Ericsson [1], a project which focused upon Ericsson's H.323-based Multimedia Telephony System (MMTS). In IMiS-Ericsson, the project goals were to share knowledge of mutual value, in order to (1) stimulate creative development of new, H.323-based service concepts and architectures and (2) acquire further insight into the security requirements for MMTS.

Other IP telephony–related work performed by NR includes a comparison of SIP and H.323 technologies, especially in regard to overall functionality and certain basic service-level issues [19]. This work was based upon study of the H.323 standard (v2) and an Internet Draft for SIP, prior to SIPs standardization.

Against this background, SIPtel can be understood to be a natural extension of NR's previous work with IP telephony, coupled with UNINETT's interest is investigating the possibilities of providing SIP-based voice over IP (VoIP) upon the Norwegian academic network.

1.3 Purpose and Goals

The motivating vision behind SIPtel is to determine to what extent it is possible to facilitate full-scale deployment of a SIP-based voice over IP (VoIP) service upon UNINETT's production network.

To initiate such a determination, the primary goal of Phase 1 within the SIPtel project has been to test and evaluate a SIP-based video-conferencing framework for voice over IP (VoIP) purposes. In this regard, the *primary* testing and evaluation work planned to target assessment of Multimedia Internet Terminal (MInT) [5], from GMD FOKUS [4], upon the Linux platform. In addition to MInT, certain other SIP-based components were to be reviewed for consideration.

The tasks originally defined in Phase 1 included:

- to thoroughly study and assess the recently-released SIP standard,
- to obtain an overview of the quality and characteristics of certain SIP components and frameworks [19], especially MInT,
- for MInT, to assess the ease of integration of RSVP and DiffSrv, and
- to assess the degree to which MInT and/or other components could serve as reliable, interoperable and extensible elements for a SIP pilot.

If time permitted, Phase 1 activities were also to include obtaining a first look into emerging technologies which target the interoperation of H.323 and SIP telephony (e.g., Lucent Technologies Softswitch [23]).

1.4 Remarks about Project Activities

When the SIPtel project was defined, the original plan was to carry out a ”…thorough study and assessment (e.g., possibilities and implications) of the SIP standard.” As the project got underway, however, great progress upon SIP had been concretely demonstrated at a “SIP Bake-off” — an event designed to test interoperation of SIP implementations evolving within both the research and commercial communities [36] (see section 2.2). For this reason and others, it was decided that the time required for “thorough assessment of the SIP standard” could be better invested by a more careful review and assessment of available SIP-based components. Rather than scrutinizing the SIP standard, NR has therefore studied the parts of the SIP standard relevant to component review, as well as basic IP telephony features, services and functionality.

Within SIPtel Phase 1, the primary testing and evaluation work planned to target assessment of Multimedia Internet Terminal (MInT) upon the Linux platform. Due to certain unexpected events encountered during the course of the testing work (see section 4.3), project effort turned to include ”light” evaluation of two other SIP-related components: Bell Labs’ siphone client [12] and sipd [21], from the University of Columbia. By the end of SIPtel Phase 1, MInT, siphone and sipd were installed upon both Linux and Solaris. The evaluation work performed in Phase 1 has ultimately focussed upon cross-platform and cross-component interoperation, rather than upon thorough critique of each element’s functional performance.

Lastly, it should also be mentioned here that lack of project resources in this phase has yet prevented SIPtel from looking into the interoperation of H.323 and SIP.

1.5 Structure of the Document

This document includes a very brief introduction to the SIP standard, followed by a categorization and summary review of SIP-related components and frameworks. The results of the SIPtel evaluation work are thereafter presented. The document closes with conclusions and recommendations concerning SIPtel Phase 2; that is, possibilities for pilot deployment of a SIP-based VoIP service upon UNINETT’s production network.

2. The SIP Standard

2.1 Introduction to SIP

This section provides an extremely high-level overview of SIP. The topics in this section include: purpose and functionality of SIP; basic elements and definitions; SIP signaling and communication; illustrations of call initiation. The section closes with a brief summary concerning the kinds of functionality which SIP doesn't provide.

2.1.1 Purpose and basic functionality

The most concise description of SIP's purpose and basic functionality is found in the standard itself:

> *The Session Initiation Protocol (SIP) is an application-layer control protocol that can establish, modify and terminate multimedia sessions or calls. These multimedia sessions include multimedia conferences, distance learning, Internet telephony and similar applications. SIP can invite both persons and "robots", such as a media storage service. SIP can invite parties to both unicast and multicast sessions; the initiator does not necessarily have to be a member of the session to which it is inviting. Media and participants can be added to an existing session.*
>
> *SIP can be used to initiate sessions as well as invite members to sessions that have been advertised and established by other means. Sessions can be advertised using multicast protocols such as SAP, electronic mail, news groups, web pages or directories (LDAP), among others.*
>
> *SIP transparently supports name mapping and redirection services, allowing the implementation of ISDN and Intelligent Network telephony subscriber services. These facilities also enable personal mobility. In the parlance of telecommunications intelligent network services, this is defined as: "Personal mobility is the ability of end users to originate and receive calls and access subscribed telecommunication services on any terminal in any location, and the ability of the network to identify end users as they move. Personal mobility is based on the use of a unique personal identity (i.e., personal number)." [3] Personal mobility complements terminal mobility, i.e., the ability to maintain communications when moving a single end system from one subnet to another.*
>
> *SIP supports five facets of establishing and terminating multimedia communications:*
>
> - user location*: determination of the end system to be used for communication;*
> - user capabilities*: determination of the media and media parameters to be used;*
> - user availability*: determination of the willingness of the called party to engage in communications;*
> - call setup*: "ringing", establishment of call parameters at both called and calling party;*
> - call handling*: including transfer and termination of calls.*
>
> *SIP can also initiate multi-party calls using a multipoint control unit (MCU) or fully-meshed interconnection instead of multicast. Internet telephony gateways that connect Public Switched Telephone Network (PSTN) parties can also use SIP to set up calls between them.*
>
> *SIP is designed as part of the overall IETF multimedia data and control architecture currently incorporating protocols such as RSVP [31] for reserving network resources, the real-time transport protocol (RTP) [32] for transporting real-time data and*

providing QoS feedback, the real-time streaming protocol (RTSP) [33] for controlling delivery of streaming media, the session announcement protocol (SAP) [34] for advertising multimedia sessions via multicast and the session description protocol (SDP) [30] for describing multimedia sessions. However, the functionality and operation of SIP does not depend on any of these protocols.

(from Handley, et.al.: "SIP: Session Initiation Protocol", IETF RFC 2543 [28])

2.1.2 Basic elements and definitions

In this section, the basic elements within SIP are presented, along with specific definitions of key terms within the standard itself. The presentation begins with the basic elements[1]:

Basic elements

User agent client (UAC), calling user agent: A user agent client is a client application that initiates the SIP request.

Client: An application program that sends SIP requests. Clients may or may not interact directly with a human user. User agents and proxies contain clients (and servers).

User agent server (UAS), called user agent: A user agent server is a server application that contacts the user when a SIP request is received and that returns a response on behalf of the user. The response accepts, rejects or redirects the request.

Server: A server is an application program that accepts requests in order to service requests and sends back responses to those requests. Servers are either proxy, redirect or user agent servers or registrars.

Proxy, proxy server: An intermediary program that acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them on, possibly after translation, to other servers. A proxy interprets, and, if necessary, rewrites a request message before forwarding it.

Redirect server: A redirect server is a server that accepts a SIP request, maps the address into zero or more new addresses and returns these addresses to the client. Unlike a proxy server, it does not initiate its own SIP request. Unlike a user agent server, it does not accept calls.

Registrar: A registrar is a server that accepts REGISTER requests. A registrar is typically co-located with a proxy or redirect server and MAY offer location services

Key definitions

Call: A call consists of all participants in a conference invited by a common source. A SIP call is identified by a globally unique call-id. Thus, if a user is, for example, invited to the same multicast session by several people, each of these invitations will be a unique call. A point-to-point Internet telephony conversation maps into a single SIP call. In a multiparty conference unit (MCU) based call-in conference, each participant uses a separate call to invite himself to the MCU.

Call leg: A call leg is identified by the combination of Call-ID, To and From.

[1] The definitions here have been extracted directly from the standard itself [28], it order to preserve correctness; however, references found within the original text have been adjusted to correspond to reference numbering used within *this* document.

Conference: A multimedia session (see below), identified by a common session description. A conference can have zero or more members and includes the cases of a multicast conference, a full-mesh conference and a two-party "telephone call", as well as combinations of these. Any number of calls can be used to create a conference.

Downstream: Requests sent in the direction from the caller to the callee (i.e., user agent client to user agent server).

Final response: A response that terminates a SIP transaction, as opposed to a provisional response that does not. All 2xx, 3xx, 4xx, 5xx and 6xx responses are final.

Initiator, calling party, caller: The party initiating a conference invitation. Note that the calling party does not have to be the same as the one creating the conference.

Invitation: A request sent to a user (or service) requesting participation in a session. A successful SIP invitation consists of two transactions: an INVITE request followed by an ACK request.

Invitee, invited user, called party, callee: The person or service that the calling party is trying to invite to a conference.

Isomorphic request or response: Two requests or responses are defined to be isomorphic for the purposes of this document if they have the same values for the Call-ID, To, From and CSeq header fields. In addition, isomorphic requests have to have the same Request-URI.

Location server: See location service.

Location service: A location service is used by a SIP redirect or proxy server to obtain information about a callee's possible location(s). Location services are offered by location servers. Location servers MAY be co-located with a SIP server, but the manner in which a SIP server requests location services is beyond the scope of the current version of the SIP standard.

Parallel search: In a parallel search, a proxy issues several requests to possible user locations upon receiving an incoming request. Rather than issuing one request and then waiting for the final response before issuing the next request as in a sequential search, a parallel search issues requests without waiting for the result of previous requests.

Provisional response: A response used by the server to indicate progress, but that does not terminate a SIP transaction. 1xx responses are provisional, other responses are considered final.

Ringback: Ringback is the signaling tone produced by the calling client's application indicating that a called party is being alerted (ringing).

Session: From the SDP specification: "A multimedia session is a set of multimedia senders and receivers and the data streams flowing from senders to receivers. A multimedia conference is an example of a multimedia session." [30] (A session as defined for SDP can comprise one or more RTP sessions.) As defined, a callee can be invited several times, by different calls, to the same session. If SDP is used, a session is defined by the concatenation of the user name, session id, network type, address type and address elements in the origin field.

(SIP) transaction: A SIP transaction occurs between a client and a server and comprises all messages from the first request sent from the client to the server up to a final (non-1xx) response sent from the server to the client. A transaction is identified by the CSeq sequence number within a single call leg. The ACK request has the same CSeq number as the corresponding INVITE request, but comprises a transaction of its own.

Upstream: Responses sent in the direction from the user agent server to the user agent client.

URL-encoded: A character string encoded according to RFC 1738, Section 2.2 [35].

User agent (UA): An application which contains both a user agent client and user agent server.

2.1.3 SIP signaling and communication

SIP is based upon exchanges of text-based messages using a Request — Response approach, much like in HTTP; thus, a SIP message is either a request from a client to a server, or a response from a server to a client.

A Request message begins with a *request-line* — a single line of text which clarifies:

- which kind of *method* it is, followed by
- a *request-URI* (e.g., the callee's address, in an INVITE Request) and
- an indication the *SIP-version* in use.

Following the *request-line*, a Request message contains a number of lines representing the *message-header* field. Finally, a *message-body* may appear[2]; this could be, for example, message contents which indicate the client's suggestion as to which kinds of media formats to employ during the call.

Response messages have identical structure to that of Request messages, though they begin with a *status-line* instead of a *request-line*. The *status-line* indicates:

- the *SIP-version* in use,
- a *status-code* (i.e., a 3-digit integer result code which indicates the outcome of the attempt to understand and satisfy the request) and
- a *reason-phrase* which is intended to give a short textual description of the *status-code*.

Request methods

There are currently six kinds of request methods defined in SIP:

- *INVITE*
  The INVITE method is used to indicate that the user or service is being invited to participate in a session. The message body contains a description of the session to which the callee is being invited. Session Description Protocol (SDP) [30] is a protocol which may be used for such descriptions.
- *ACK*
  The ACK request is used to confirm that the client has received a final response to an INVITE request; ACK is used only with INVITE requests.

[2] If a *messasge-body* appears, it is separated from the *message-header* by a blank line.

- *OPTIONS*
  This method is used to query the server as to its capabilities. It must be supported by SIP proxy, redirect and user agent servers as well as clients. As with INVITE, SDP may be employed for capability descriptions.
- *BYE*
  The user agent client uses BYE to indicate to the server that it wishes to release the call.
- *CANCEL*
  The CANCEL request is used to cancel a pending request with the same Call-ID, To, From and CSeq (sequence number only) header field values, but does not affect a completed request. (A request is considered completed if the server has returned a final status response.)
- *REGISTER*
  A client uses the REGISTER method to register the address listed in the To header field with a SIP server.

Response types, codes and examples

There are currently six major types of responses defined in SIP:

- *Informational 1xx*
  examples: 100 Trying, 180 Ringing, 181 Call Is Being Forwarded, 182 Queued
- *Successful 2xx*
  examples: 200 OK
- *Redirection 3xx*
  examples: 300 Multiple Choices, 301 Moved Permanently,
  302 Moved Temporarily, 305 Use Proxy
- *Request Failure 4xx*
  examples: 400 Bad Request, 401 Unauthorized, 402 Payment Required,
  403 Forbidden, 404 Not Found, 405 Method Not Allowed
- *Server Failure 5xx*
  examples: 500 Server Internal Error, 501 Not Implemented, 502 Bad Gateway
- *Global Failures 6xx*
  examples: 600 Busy Everywhere, 603 Decline, 604 Does Not Exist Anywhere

2.1.4 *Initiating a call: three models*

The following three figures illustrate exchanges of request - response messages, in simplified form, for three different models of call initiation. The models depicted here are:

- basic call initiation, minimal model (Figure 1)
- call initiation via a Proxy Server (Figure 2), and
- call initiation supported by a Redirect Server (Figure 3).

The latter two models assume that the callee has their current address registered (either manually or automatically) with the Registrar. In addition, the figures depicting these latter models do not depict a distinction between the SIP roles (e.g., UAC vs. UAS, etc.); instead, the roles are coalesced and implicitly represented as "executing within" specific hosts.

With regard to the completion of call set-up, none of the figures illustrate the message exchanges required for negotiation and establishment of call parameters between the caller and the callee.

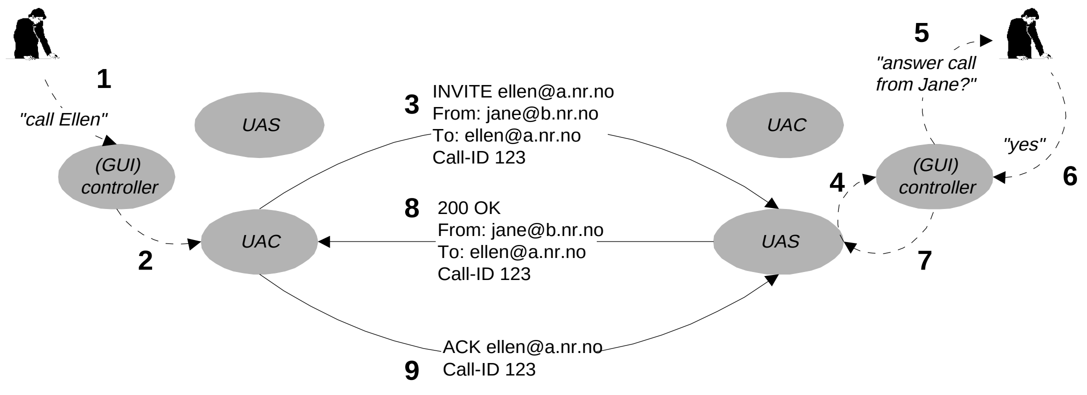
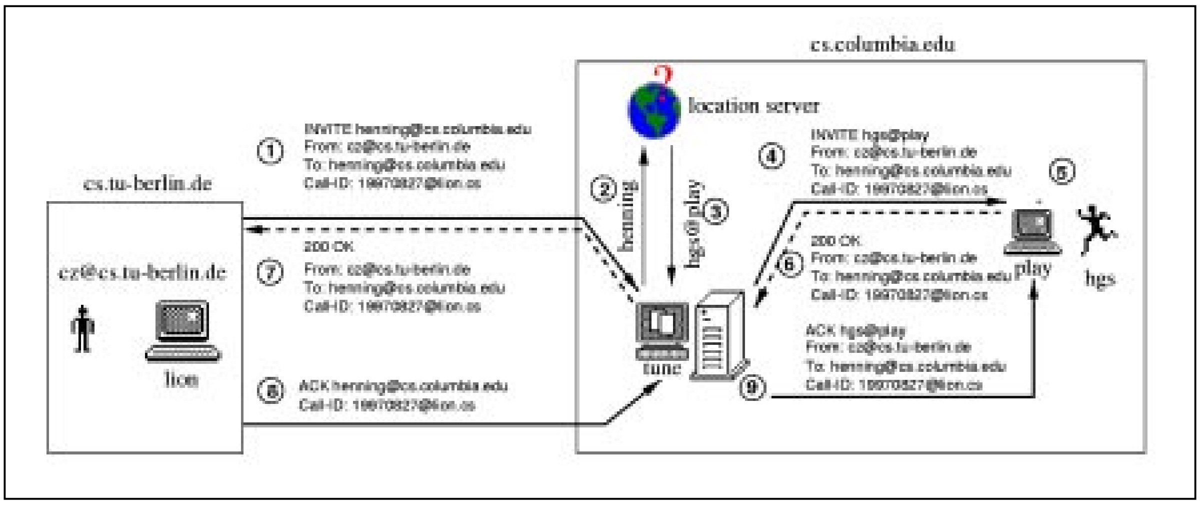


Figure 1: Basic call initiation, minimal model



INVITE henning@cs.columbia.edu
From: cz@cs.tu-berlin.de
To: henning@cs.columbia.edu
Call-ID: 19970827@lion.cs

INVITE hgs@play
From: cz@cs.tu-berlin.de
To: henning@cs.columbia.edu
Call-ID: 19970827@lion.cs

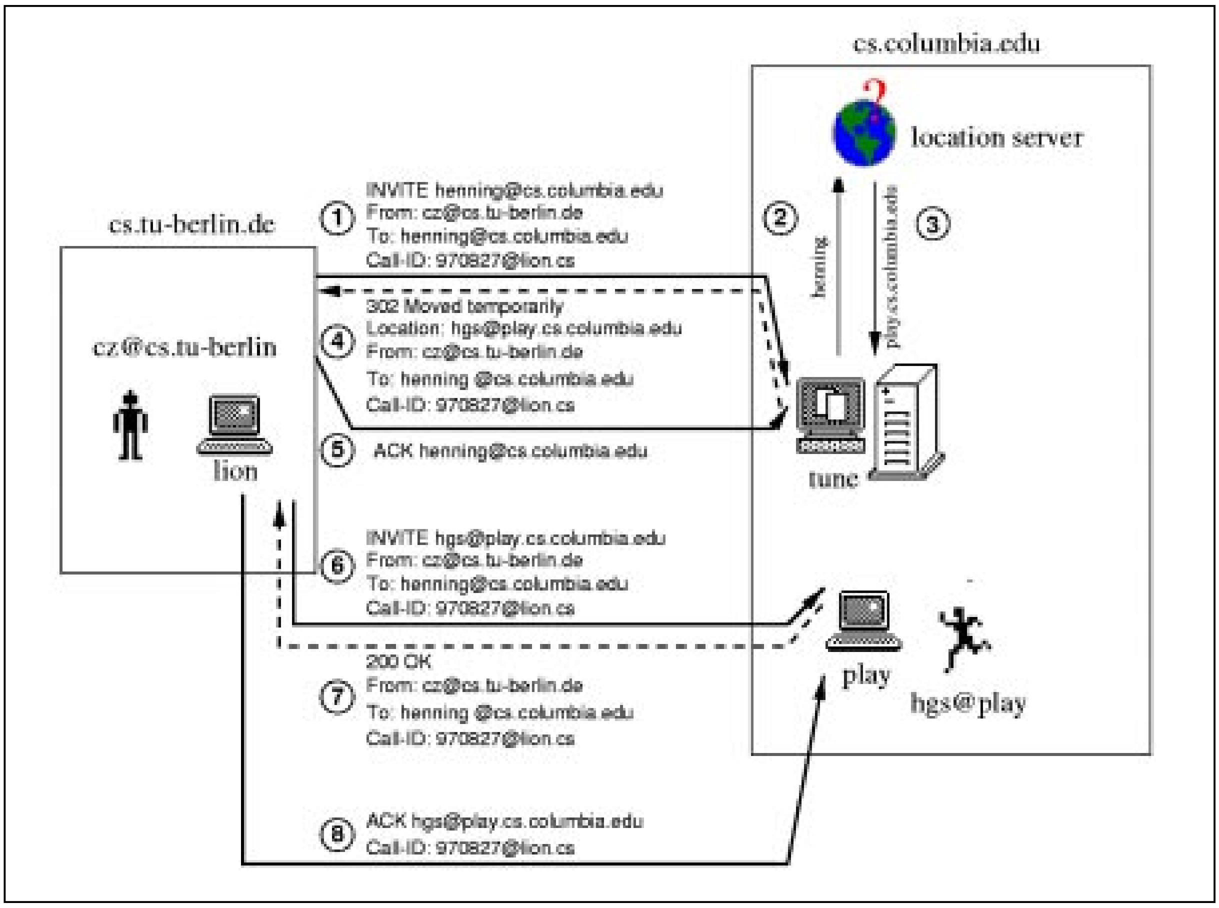
200 OK
From: cz@cs.tu-berlin.de
To: henning@cs.columbia.edu
Call-ID: 19970827@lion.cs

200 OK
From: cz@cs.tu-berlin.de
To: henning@cs.columbia.edu
Call-ID: 19970827@lion.cs

ACK hgs@play
From: cz@cs.tu-berlin.de
To: henning@cs.columbia.edu
Call-ID: 19970827@lion.cs

ACK henning@cs.columbia.edu
Call-ID: 19970827@lion.cs

Figure 2: Call initiation via a Proxy Server (figure from [29])



INVITE henning@cs.columbia.edu
From: cz@cs.tu-berlin.de
To: henning@cs.columbia.edu
Call-ID: 970827@lion.cs

302 Moved temporarily
Location: hgs@play.cs.columbia.edu
From: cz@cs.tu-berlin.de
To: henning @cs.columbia.edu
Call-ID: 970827@lion.cs

ACK henning@cs.columbia.edu

INVITE hgs@play.cs.columbia.edu
From: cz@cs.tu-berlin.de
To: henning@cs.columbia.edu
Call-ID: 970827@lion.cs

200 OK
From: cz@cs.tu-berlin.de
To: henning @cs.columbia.edu
Call-ID: 970827@lion.cs

ACK hgs@play.cs.columbia.edu
Call-ID: 970827@lion.cs

Figure 3: Call initiation via a Redirect Server (figure from [29])

2.1.5 What SIP doesn't do

To supplement this description of basic introduction to SIP and its functionality, it is important to make clear which kinds of functionality lay **outside** the scope of SIP. Examples of functionality, specifications and protocols which are not part of SIP include:

- SIP provides no specifications as to which *applications* should be used exchange audio/video data: only media *type-specifications* are agreed upon during capability negotiation (e.g., using SDP-based specifications)
- SIP does not provide any conference control services, such as floor control or voting
- SIP does not reserve resources (though it can convey to the invited system the information necessary to do this)
- SIP does not prescribe how a conference is to be managed, nor does it provide any *conferencing framework*; in other words, SIP does *not*:
  - allocate multicast addresses
  - automatically select nor launch applications for audio / video exchange, once call setup is completed
  - provide any inter-application protocols through which applications can remain synchronized or maintain consistent state (e.g., ensuring that when a participant is added to one application in a conference, that he is added to other applications within the same conference).

Answers to “Frequently Asked Questions” about SIP can be found at [20].

2.2 The SIP “Bake-Off”

The First SIP “Bake-off” was held at Columbia University on April 8-9, 1999, just three weeks after the SIP protocol was approved as an official Internet standard. Recognized IP telephony authorities such as Jeff Pulver [41] made strong comments about the success of the event, as recounted in (an adapted version of) the press release from the Bake-off:

> *The technology-oriented event was organized to test the interoperability of software and hardware devices using the Session Initiation Protocol. … Organizations participating in the interoperability testing came from the US, Canada, Sweden, Finland and the UK and included, among others: 3Com, Alcatel, Cisco, British Telecom, Columbia University, Dialogic, dynamicsoft, Ellemtel, Ericsson, Helsinki University of Technology, Hewlett-Packard, Lucent, Mediatrix, Nortel, and Pingtel.*
>
> *At the end of the event, nearly all implementations had achieved interoperability for call setup and media capability negotiation for multimedia calls. Several were interoperable on the first try, and most others after minor changes or bug fixes were made. Systems being tested included user agents and servers running on a variety of operating systems. Some groups had brought dedicated hardware, including Internet-to-phone gateways and ”Ethernet phones” that plug directly into local area networks. The event was able to show, for example, that subscribers can move from location to location, anywhere on the Internet, with phone calls following them automatically, regardless of the provider of the hardware or software. Users could also forward calls to any Internet destination or a telephone number. Some groups also tested advanced features such as call screening*

and user authentication. …

According to leading Internet telephony industry analyst and expert Jeff Pulver, the event instantly advanced the state of the art in Internet-based telephone systems and services. ”Internet telephony is already reshaping the global telephone system in dramatic new ways,” said Pulver. ”The number and names of the companies that tested products at this event means that SIP is quickly going to have a big effect on this telecom industry overhaul.”

(text taken from: “First SIP Bake-Off: press release” [37])

2.3 Status and Related Drafts

SIP became a Proposed Standard on February 2, 1999, and was published as RFC 2543 on March 17, 1999. A second SIP bake-off [38] took place at pulver.com (Melville, NY) on August 5th and 6th, 1999, and a third bake-off [39] is tentatively scheduled for December 6 through 8, 1999 (location to be announced).

Currently, there is a great deal of ongoing activity related to SIP. Some of this activity concerns proposals for extensions to the standard itself, while other proposals concern guidelines, approaches and mechanisms for interoperation between SIP and other protocols. Some of the areas under study include:

- aspects of security and admission control
- support for distributed call state
- resource management and QoS
- interworking with telecom elements and protocols
- call processing
- user control of internet telephony services
- call waiting and other user services (e.g., click-to-dial-back and third-party control)
- user location / presence
- multipart message bodies for carrying non-SIP signaling messages
- charging and
- SDP extensions for fax over IP.

A list of drafts reflecting ongoing, SIP-related work within the IETF can be found at [40].

3. Review of Existing SIP Components

A number of SIP-related implementations are summarized by Henning Schulzrinne in a WWW-based table [19]. SIPtel has used this table as a “point-of-departure” in order to review which kind of components and products have been created, and what their status is concerning availability and stability.

The review is based upon information found on the WWW, as well as information gathered through direct phone contact and email queries. In section 3.1, the results of the review are presented first in tabular form, indicating: the name of the software, the organization behind the software, which SIP elements the software implements, etc. In section 3.2, brief commentary notes are provided for each of the components in the table.

3.1 Components reviewed

Table 1

		User Agent		**Servers / Services**				
SW	**Org**	**UAC**	**UAS**	**Re-direct**	**Proxy**	**Registrar**	**Location Svc**	**Other**
TIOP	AT&T Labs	?	?	?	?	?	?	Not for distribution outside AT&T.
sipd	Columbia Univ.			X	X	X		Sip server.
aliases, tracker, lswhod, namemapper							X	See commentary notes.
sipc		X	X					See commentary notes.
jSIP	Dynamic-soft	x[3]	X	X	X	X		Supports SDP and SAP. Programming APIs.
jVoIP								Programming Framework including CallAccounting MGCP, Directory service and jSIP.
SIP-server	Ellemtel Utvecklings AB			X		X		The software is not available, but Ellemtel offers free registration on their redirect server.
sdr	ISI	x[4]						SDP and SAP program with some SIP support.

[3] Sample client code only. This is a software development package for integrating SIP support in other applications.

[4] SDR is primarily a SAP (Session Announcement Protocol) tool for listing multimedia sessions more or less like a TV program guide.

		User Agent		Servers / Services				
SW	**Org**	**UAC**	**UAS**	**Re-direct**	**Proxy**	**Registrar**	**Location Svc**	**Other**
siphone	Lucent	X	X					Java client with media clients for Solaris[5].
Softswitch								Provides interoperability across heterogeneous networks that support a wide range of signaling protocols (including SS7, MGCP, H.323, and SIP).
Audiotrix Phone Adapter II	Mediatrix	(x)	(x)[6]					Desktop device for switching ordinary phone calls on to the Internet if possible.
	Nokia Research							Reported to have a SIP implementation, but we have not been able to find it.
ErlIPhone	Ericsson	X	X	X		X		Not publicly available
MInT	GMD Fokus	X	X	X			X	Media agents, shared document display, voting and floor control.
	HP Labs	X	X	X	coming	X		Java 2 based. Not available outside HP, but might become that later.
	Pingtel	(x)	(x)					Business phone system. Not released yet.
JIP: Java Internet Phone	University College London	?	?					Sold. No longer available.
SIPhone	Hughes Software Systems	X	X		X	X		SIP prototype constructed; also includes Voice Mail Server..
	Object Software							Prototype. Development is frozen.

[5] The media client Nevot (which also may be used with MinT) is being ported to Linux by the MinT team.

[6] APA II is a device, not a program. These are the closest matching categories.

3.2 Commentary notes

TIOP

Partial SIP implementation for AT&T spoken language translator prototype. Not available outside AT&T [42].

sipd

Pre-production quality SIP server developed and licensed by Columbia University. Implements Redirect and Proxy functionality. Accepts registration from user agents. Both binary and source code licenses available. Runs on several kinds of Unix [21].

aliases, tracker, lswhod, namemapper

Utility programs used by *sipd* to locate the called user, that is, to find out which terminal(s) he is currently logged in on. The source code is freely available from Columbia University. Should compile and run on most Unixes [21].

sipc

SIP client and User Agent developed and sold by Columbia University. Currently in “alpha” stage only, but licenses are available [43].

jSIP

Commercial implementation of SIP conforming to SIP draft version 12 (not to the final standard). Based on Dynamicsoft’s jVOIP Framework. Implemented in Java. Runs on all platforms with a Java 1.1 virtual machine. The current jSIP implementation includes the following: User/Agent Server, Proxy Server, Re-direct Server, Application Programming Interface(s) in Java/C/C++, Sample Client Code, Class Libraries, Documentation, SIP DRAFT RFC documentation [46].

jVOIP Framework

Dynamicsoft’s framework for Voice Over IP technologies, all implemented in Java. Includes the jSIP implementation of the SIP standard, the jMGCP implementation of the Media Gateway Control Protocol draft standard, jCallAccounting for call accounting, jDirectory for enhancing VoIP Gateway functionality [46].

Ellemtel Utvecklings AB

Ellemtel is a non-profit research institution owned by Telia and Ericsson. The owners have decided to close down Ellemtel and phase out all activities by the end of 1999. Ellemtel has developed a redirect and registration server. The software is not available outside Ellemtel, but a copy of the program is running at sip.pcs.ellemtel.net and SIP users are welcome to register their User Agents. One of the owners will probably take over the SIP implementation. It is not known what will happen to the running server [50] [51].

sdr

sdr is a session directory listing available multimedia sessions more or less like a TV program guide. Some versions of sdr have included an implementation of an outdated version of SIP.

ErlIPhone

SIP server developed by Ericsson, partly to test out their new programming language Erlang. Not available outside Ericsson.

MInT

See “Introduction to MInT” , section 5.1.

HP Labs

HP Labs has developed a user agent, registrar and redirect server. They are planning a redirect server. Everything is made in Java 2. The software is developed for internal research and knowledge acquisition only. No commercial release is planned, but that might change if the product turns out to be good. Currently not available outside HP [52].

siphone

A user agent and user client made in Java by Lucent’s Bell Labs. Research work by Jonathan Rosenberg, who was a member of the SIP standard development committee. Comes with the Nevot and Nevit media tools, but they only work on Solaris.

Lucent Technologies Softswitch

From [23]: “The Lucent Softswitch is a Bell Labs-developed ‘software switch’ for IP networks that couples the reliability and features that customers expect from public telephony networks with the cost effectiveness and flexibility of IP technology. With the Lucent Softswitch, you will be able to provide a full range of IP-based communications services indistinguishable in quality and variety from services on traditional circuit networks.”

“Our software provides interoperability across heterogeneous networks that support a wide range of signaling protocols (including SS7, MGCP, H.323, and SIP). The Lucent Softswitch translates industry-signaling protocols into a generic call-signaling format, simplifying the addition of new protocols. This capability allows PSTN and Internet Telephony Service Providers to provide rich, seamless, interoperability between PSTN and IP network domains. In addition, this translation enables signaling interoperability between multiple vendor gateways.”

Audiotrix Phone Adapter II

APA II from Mediatrix [49] is an electronic device for switching ordinary phone calls onto the Internet, when possible [47]. The Pulver Report (July 1999 issue) [48] reports SIP interoperability with *sipd* SIP server from Columbia University. It may thus be regarded as a gateway between SIP/VoIP and an ordinary phone. The device contains a 486-chip, special D/A-hardware and RealTime OS. Applications may be developed using Visual C++. Many IETF standards are supported or planned to be supported: ftp, telnet, http, snmp etc. Supports H.323.

Pingtel

Pingtel is developing client side SIP products, both “software phones” and SIP-enabled physical telephony devices. The details are not officially available until the products are announced/released. However, they will not include any SIP servers or PSTN/IP gateways. The products will be conformant to RFC 2543 and with the upcoming call extensions being worked on by Lennox et al. Pingtel is planning beta testing in Q4 1999 and volume shipping in Q1 2000 [54].

JIP, Java Internet Phone, University College London

The JIP system has been sold to a major actor in the telecom business for product development. Further details are not public.

SIPhone, Hughes Software Systems (HSS)

HSS has reported that they are actively working on SIP. A SIP prototype has been constructed which includes a UAC, UAS, Proxy, Registrar and Voice Mail Server. HSS plans to offer a range of SIP packages to the market ranging from SIP stacks and Developers Kits to Out-of-the-box SIP clients and servers. [55]

Object Software

Object Software is reported by [19] to have a SIP implementation. The implementation was a prototype redirect server. The software is not available, and development has been put on hold [53].

4. Introduction to SIPtel Evaluation Work

4.1 SIP-based VoIP and Functional Differentiation

The primary goal of Phase 1 within the SIPtel project has been to review, test and evaluate a SIP-based framework for multimedia conferencing, as well as certain other components for voice over IP (VoIP) purposes. In preparing to carry out and report such work, it is important to keep in mind the specific roles and functions of the various elements.

First, there are SIP-based components which provide the functionality of the SIP control protocol itself (e.g., components such as UACs, UASs, SIP Proxy Servers, etc.). It is these components which are functionally responsible for delivery and support for the establishment and termination of multimedia communications, i.e.,:

- *user location*: determination of the end system to be used for communication;
- *user capabilities*: determination of the media and media parameters to be used;
- *user availability*: determination of the willingness of the called party to engage in communications;
- *call setup*: "ringing", establishment of call parameters at both called and calling party; and,
- *call handling*: including transfer and termination of calls.

Secondly, there are the components which are necessary for the acquisition and interchange of media data (e.g., audio and video applications such as vat, Nevot, vic, etc.).

Thirdly — within framework(s) for multimedia conferencing — there is the additional need to provide mechanisms and services for conference control and management, such that the conference artifacts appear (sufficiently) consistent to both the conference applications and participants. Strictly speaking, these lattermost kinds of functionality are outside the scope of the SIP standard.

4.2 Evaluation Topics and Perspectives

4.2.1 Influence of perspective

Another issue which must be kept in mind is the *perspective* adopted in the evaluation and reporting work. In SIPtel's case, two primary user-perspectives were employed. These were the consideration of the evaluation topics from the point of view of (a) the "common, computer-literate consumer" vs. (b) a highly skilled user, developer or researcher.

It is also important to keep in mind what *purpose* a specific component/framework is to serve for a particular class of users. With respect to VoIP, the common consumer may be looking for a computer-based substitute for (or complement to) her standard telephone. Highly skilled developers and researchers may be looking for a research vehicle for articulated investigation of VoIP applications and network services; yet, like the common consumer, these same developers and researchers could also be looking for a simple-to-use, computer-based telephone.

4.2.2 Topics for evaluation

In parallel with SIPtel's initial work regarding component review, consideration was given as to which topics should receive attention during the evaluation work. These are briefly summarized below:

- *documentation*: Does the software come with adequate documentation for installation and use ?
- *installation*: How easy or complex is the software to install ?
  What are the system requirements ? Which kind of configuration was used for testing ?
- *functional performance*: Which functional roles are supported by the software ? How well do the software (and/or its modules) work[7] ?
- *usability*: Does the software supply an integrated user-interface ?
  Is the interface clear and simple to use ?
- *architecture, extensibility, openness*: What is known (or made known) about the software architecture ? Is the architecture loosely coupled or 'monolithic' ? Are APIs provided ? Are open standards used ?
- *potential for interoperation with other components / applications / frameworks:* Does the software (and/or its modules) interoperate with other SIP-based components[8] ?

4.3 Remarks about the Evaluation Activities

Plans

Within Phase 1 of SIPtel, the primary testing and evaluation work planned to target assessment of Multimedia Internet Terminal (MInT) upon the Linux platform. Additionally, the project had intended to look *closely* into MInT with respect to the functional areas and criteria described above.

"Costly problems" and change of emphasis

Due to certain unexpected events encountered during the course of the testing work (see below), project effort turned to include ”light” evaluation of two other SIP-related components: Bell Labs' siphone client [12] and sipd [21], from the University of Columbia.

Cause of the problems

The first problem encountered arose in connection with the selection of a soundcard that would work with Vat on Linux. We first bought Creative Soundblaster Live (SBLive) soundcards. These were suggested by the dealer, as he was told they had Linux support, and that they were phasing out older Soundblaster cards. It turned out that Creative's SBLive driver for Linux still is in beta, and that Creative actively tries to prevent the production of open source drivers for this card. Vat did not work with this driver, though certain other sound applications did. One of the PCs was later refitted with an older Soundblaster 16 card and appropriate driver. This worked fine, but a significant amount of time had been lost in getting this "simple" bug out.

[7] SIPtel made no assessment of the audio quality delivered by the audio applications vat and nevot. Resources became focused instead upon cross-component interoperation (see section 4.3).

[8] This issue is addressed in section 8.

The second major problem encountered concerned MInT's user agent server (*sipd)* on Linux. There seemed to be no contact between the caller's UAC and the callee's UAS. It turned out that there was contact, but that *sipd* on Linux crashes during the user location procedure. The person responsible for this piece of software at GMD could later confirm this (see also test note 1, section 5.2.3).

Results

By the end of Phase 1, MInT, siphone and Columbia's sipd[9] were installed upon both Linux and Solaris. *The evaluation work which has been performed has ultimately focussed upon cross-platform and cross-component interoperation, rather than upon thorough critique of each element's functional performance.*

Other areas which could not be evaluated within this phase included:

- SIP-based video conferencing
- changes in OPTIONS (e.g., dynamic changes in audio format)
- multi-party conferencing.

4.4 Testing Diagrams and Legend

In sections 5 through 7, testing and evaluation is reported for each of *MInT*, *siphone* and the University of Columbia's *sipd* software. In those sections, as well as in the Appendix[10], diagrams are used to help illustrate exactly which tests were performed. Each of the diagrams employs a common legend, which is depicted in Figure 4.

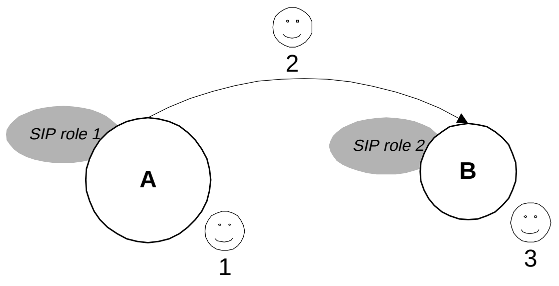


Figure 4: Legend for testing diagrams

In the legend, circles contain names of tested components (e.g., A and B). The shaded ellipse behind each component depicts which role the component manifests with respect to SIP. Interoperation between components is marked by an edge or arrow connecting the two components. The direction of an arrow indicates which component uses (or initiates interaction with) another component. Normal (i.e., non-dashed) lines indicate "normal" interoperation. Dashed lines indicate that interoperation didn't work because e.g., one of the components didn't work.

Faces near the name of each component and arrow depict a (very) rough rating for that component / interoperability edge; numbered commentary notes for each of these ratings (e.g., 1, 2, 3) are provided in the text which accompanies each test diagram.

[9] Make careful note that MInT includes a software component called sipd. The sipd component within MInT is **not** the same as the sipd component developed and licensed by the University of Columbia (though they have a common origin, see test note 1 in section 5.2.3, as well as footnote 11).

[10] The Appendix to this document includes a complete testing diagram for all components.

5. Evaluation of MInT

5.1 Introduction to MInT

MInT was developed as part of the USMInT project [6] at GMD FOKUS. The central control and inter-application communication mechanisms within MInT are based upon original work by Henning Schulzrinne [7] concerning Integrated Session Control (ISC [8]) and Pattern Matching Multicast (PMM [9]). Henning Schulzrinne is a central force within the area of Internet development, contributing to an extraordinary number of Internet drafts and standards across a wide variety of topics. His work on ISC and PMM was carried out during his stay at GMD FOKUS; it is summarized in [10].

5.1.1 Objectives of USMInT

USMInT stands for *Universal Scalable Multimedia in the Internet*. The project's objectives concentrated primarily upon multimedia communication and QoS control in the Internet, including examination, design, implementation and application testing. The enhancement and improvement of pre-existing MBONE tools was of particular interest. The specific goals within USMInT were to achieve:

- high user acceptance and easy configurability
- multimedia communication
- conference functionalities (e.g., floor control, invitation of conference participants, voting, joint viewing of documents, etc.)
- QoS control for audio/video for current conditions in heterogeneous networks, consideration of UDP applications and adaptation
- reservation of network resources.

The manifestation of these goals has been quite explicit during the review and testing of MInT.

5.1.2 MInT system overview

A very high-level description of MInT is available from its WWW-site:

> *MInT is a flexible multimedia tool set that allows the establishment and control of multimedia sessions across the Internet. The system architecture is fully distributed, with no central components. MInT offers the following features:*
>
> - *Audio transmission and reception for qualities ranging from GSM up to CD based on the NEVOT [15] and VAT [14] tools*
> - *Video transmission and reception based on VIC [13]*
> - *An integrated conference control instance that hides the complexities of the single tools*
> - *floor control*
> - *voting based on MPOLL*
> - *interface to SDR*
> - *RSVP reservation capabilities for all used applications*
> - *Adaptive video transmission*
> - *Joint viewing and remote control of postscript documents*
> - *Invitation of remote users based on the Session initiation protocol (SIP-version 3.0)*

with name resolution capabilities

(text taken from: MInT WWW-site [5])

The software modules/components which come packaged with MInT are summarized in Table 2. Make careful note that MInT includes a software component called sipd. The sipd component within MInT is not the same sipd component delivered by the University of Columbia (see footnote 9).

Table 2: Contents of the MInT software package (L = Linux, S=Solaris)

User Agent Client	**User Agent Server**	**Media / communication tools**	**Site/domain SIP server**	**User location service**
Isc	Sipd	Vic (S, L), Nevit (S), Vat (L) Nevot (S), Netghost (S, L), Mpoll (S, L), Ifloor (S, L).	Sipd (redirect only)	aliases, namemapper, lswhod.

5.2 MInT and Evaluation Topics

As a multimedia conferencing framework, MInT offers a range of functionality far exceeding the focus of SIPtel. It should therefore be made clear that, in MInT's case, the evaluation work was limited to MInT's SIP-related VoIP components (i.e., isc, sipd, Nevot and vat).

5.2.1 Documentation

The documentation concerning installation of MInT was included as part of the software package. It provided straightforward and clear information concerning the manner in which MInT should be installed.

There is also a separate user's guide for MInT [11]. The user's guide provides information concerning aspects of MInT's overall architecture, as well as other more technically-oriented information. After reading that guide, one may easily be left with the impression that MInT was created as a research vehicle for development and experimentation with protocols and mechanisms required for internet-based desktop conferencing. That is, aside from the technical information provided in the guide, the material therein also includes quite detailed explanation as to how to access and (re)set an enormous number of system parameters — parameters which influence both the high- and low-level behavior of the system.

For developers and researchers, what is wonderful is that (it appears that) nearly all of the parameters one might wish to manipulate can be accessed and (re)set through a window-based interface, *during* system execution. For the common though computer-literate consumer, these possibilities could well be a bit bewildering.

From the MInT user guide, it is not made explicitly clear whether there exist any special menu "shortcuts" (e.g., menus which pop up through right-button clicks), or any equivalent keyboard shortcuts for certain, common operations. One suggestion to possibly improve MInT's user's guide would be to break it into two documents: one which described the most bare essentials, for the common consumer; and, a second document which held the more technically-oriented information (e.g., a reference manual).

5.2.2 Installation

System requirements

Mint is available for Sun Sparc with soundcard and Solaris 2.5. (or newer), as well as for Linux with a Vat-compatible soundcard (see section 4.3). Unfortunately, the current version of the Sipd software packaged with MInT doesn't work on Linux. Some of the media communication tools can be found in standalone versions for other platforms as well; for instance, Vic and Vat, can be run on Silicon Graphics, Windows NT and Windows 95/98 in addition to Linux and Solaris.

The location service used in MInT needs a domain-wide shared directory for the user database.

Configuration employed

We tested Mint on two PCs running Red Hat Linux 6.0 (Kernel version 2.2.5). Both were configured with a Pentium II 350 MHz processor, 128 MB RAM and a 10/100 Mbps Ethernet network card. The PCs were equipped with Soundblaster Live! Soundcards. It was discovered that Vat could not get any microphone input with this card, and one of the PCs was refitted with an older Soundblaster 16 card later.

Mint was also installed on NRs Unix network and tested on two Solaris workstations: one was a Sun Ultra 5 workstation with Solaris 2.7 (AKA Solaris 7), while the second was a Sparc Classic also running Solaris 2.7. Both workstations were equipped with soundcards by the producer.

Installation complexity

Mint is fairly easy to install. The installation was well described in user manuals. The program package is unzipped and untarred to the destination directory, and the installer runs the installation script. The script will prompt for the name of a domain wide directory where the running copies of the *lswhod* location service (started automatically along with sipd) will keep a database for user location(s).

Each user has to run a personal installation script if she wants to use Mint, in order to have the *$USER/.mailcap* file modified properly. This characteristic is a drawback.

Mint comes with special startup scripts for Isc and Sipd. These should have a link from a directory in the user's path, but this is not suggested in the installation manual.

Sipd must run (as daemon / in the background) on all hosts where users want to receive calls. Only one copy of Sipd can run on the same host at one time. Sipd has to be run with the user ID of user who receives calls on that host, otherwise it won't be able to start Isc for the recipient upon incoming calls. If more than one user wants to receive calls on that host, Sipd has to be run with root permission (Isc will be started with user permissions). In order to receive calls to e-mail like addresses (*user@domain.name*, instead of *user@workstation.domain.name*), Sipd must run on the mail host as well. No root privileges are needed here, as sipd only will redirect the incoming call to a workstation where the user is logged in. The installation script does not address automatic starting and stopping of Sipd daemons. Setting this up properly must be done by the installer.

5.2.3 Functional performance

Table 2 above summarizes MInT's SIP-related functional components. The performance of these elements is summarized in Figure 5 and the associated commentary notes. Note that the legend employed in Figure 5 is explained in section 4.4.

Testing diagram

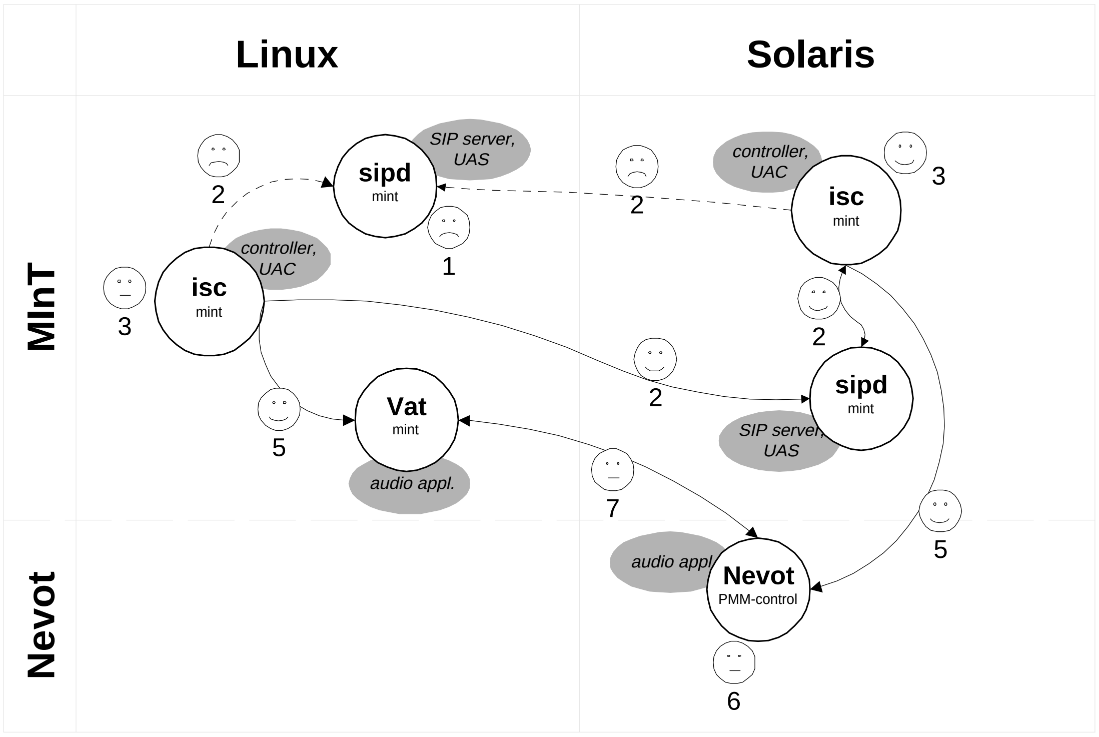


Figure 5: Testing of MInT

Notes about testing

1. MInT's *sipd* doesn't work on Linux. GMD has identified the bug, but it is unclear what changes they may or may not make to the rectify the problem[11,12]. Presently, one should not expected that GMD puts too much effort into correcting the bugs in MInT's *sipd* software for Linux.
2. MInT's *sipd* software works on Solaris. However, as long as MInT's *sipd* doesn't work on Linux, there is no way of making calls to a MInT system running on Linux or through a site- / domain-central MInT SIP server running on Linux. Only Solaris MInT users can receive calls. Linux users may, however, call MInT users on Solaris.

[11] The reason for GMD's hesitation in this matter first concerns *sipd*'s copyright and redistribution agreements. That is, MInT's *sipd* was orignally written prior to SIP standardization; additionally, the code was not written by GMD alone. Further development of the *sipd* software has been carried out at the University of Columbia. Currently, it is the University of Columbia who releases and distributes the *sipd* software, not GMD.

[12] Depending upon funding, GMD is also considering the development of a JAVA-based MInT framework which conforms to the SIP standard. Whether and how they re-implement the *sipd* functionality currently packaged with MInT, within such an effort, is currently and open issue.

3. MInT's *isc* works on both Linux and Solaris. Some Tcl errors occur on Linux. This does not seem to have impact on the overall functionality (unless the user chooses the "abort"-button on the error message box).

5. MInT uses *Nevot* and *vat* as media agents for audio. Both *vat* and *Nevot* are originally developed as stand-alone audio applications. MInT supplies special versions which are controllable over the PMM message bus. *Nevot* is the preferred audio tool, but *Nevot* exits only for Solaris[13]. Therefore, the audio media agent packaged with the Linux version of MInT is a PMM-enabled version of *vat*.

6. *Nevot* has hardly any user controls, since it is supposed to be controlled via (something like) *isc*. Lack of volume controls is annoying, for instance, when one tries to use *Nevot* without the *isc* window on the screen. This is the case when *Nevot* is used together with *siphone*. On Solaris, speaker and microphone volume can be adjusted using the Sun *audiotool*. Running all of these programs together, however, makes the screen quite crowded with windows.

7. *vat* and *Nevot* are interoperable, such that *vat* users and *Nevot* users may talk to each other. Lack of command-line options and user controls in *Nevot* makes it hard to set up the connection, however.

5.2.4 Usability

Only VoIP aspects have been addressed in this project, and problems with getting MInT and vat to work on Linux consumed an inordinate amount of project resources. Thus MInT's capacity to support video has not been tested. Additionally, the tests on Solaris were successful, but one of the two Solaris platforms in use was so outdated that we never actually got audio through the older system's audio device. For these reasons, the remarks here concerning Mint's usability are based upon the experiences gained through careful study of the user's guide, along with limited operation of the system itself.

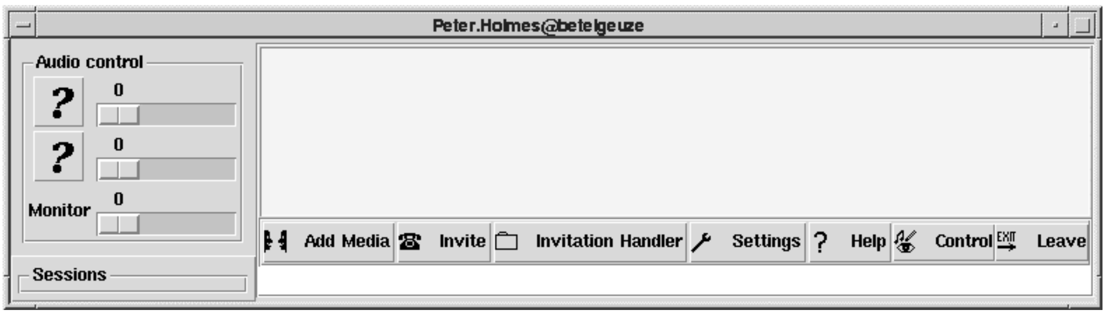


Figure 6: MInT's isc window at startup

MInT's primary window for user interaction is the isc window, see Figure 6. *Assuming a user knows what MInT can be used for* (i.e., multimedia desktop conferencing), the isc window is judged to be rather intuitive for users familiar with even a limited range of multimedia applications. It is judged that less experienced users (especially those less experienced with multimedia) may not immediately understand a few of the features in

[13] The MInT team has said that a Linux version is underway. Presumably, that version will be PMM-controllable.

the interface.

When it comes to MInT’s other windows — specifically those for manipulating system parameters such as audio and video settings, etc. — the opportunities provided for developers, researchers and highly-skilled users are excellent. That is, these skilled individuals can employ a simple, window-based interface in order to test and/or tailor system performance in run-time. For unskilled users, however, such windows could lead to avoidance rather than use; alternatively, through inappropriate settings, such users could degrade the performance of (their own instance of) the system.

5.2.5 Architecture, extensibility, openness

MInT’s architecture is distributed and loosely-coupled, thereby allowing for the easy integration and substitution of components. Being based upon and/or interoperable with a variety of internet standards and other open protocols (e.g., PMM, SDR, SAP, RTP, IP, UDP, etc.), MInT is quite an open system.

5.3 QoS Integration within MInT

MInT includes a Reservation Agent which integrates RSVP with the overall framework. This Agent serves to reserve bandwidth on *joint* behalf of applications requesting such services during the conference. Rather than integrating each of the applications with the RSVP Demon, the applications are integrated with the Agent (compare Figure 7 and Figure 8). In this way, the Demon has but a single interface to the framework. This approach moves the problem of cross-application resource competition to a point which is controllable from a central module within the framework.

For further information readers are referred to chapter 5 in MInT's User's Guide [11].

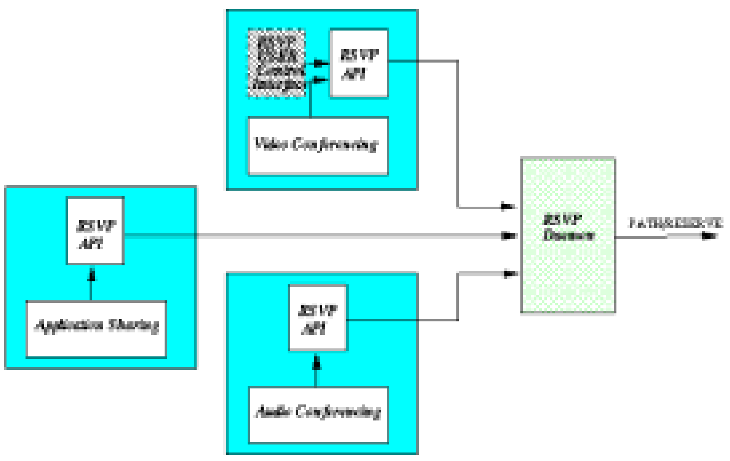


Figure 7: A common approach for integrating RSVP (figure from [11])

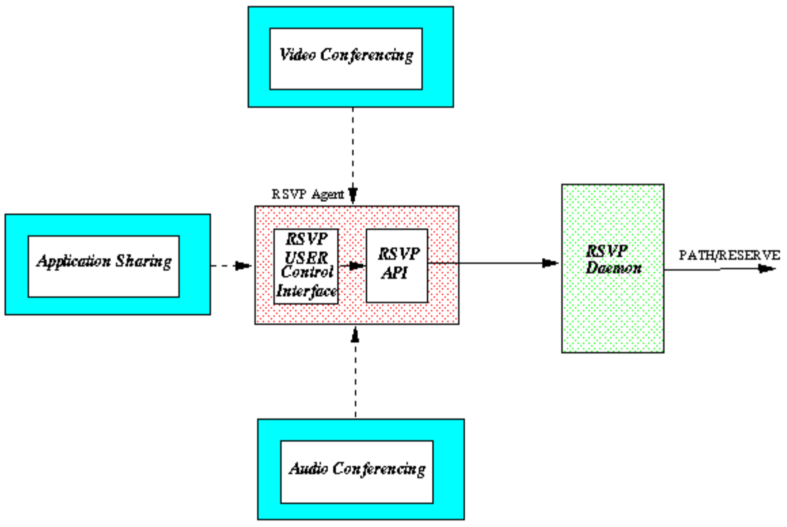


Figure 8: MInT's approach for integrating RSVP with the Reservation Agent (figure from [11])

6. Evaluation of siphone

6.1 Introduction to siphone

From Bell Labs' SIP project home page: "Siphone is a Java based SIP user agent. It provides a GUI for making, managing, and terminating IP telephone calls using the Session Initiation Protocol (SIP). Siphone supports registrations, call forwarding, transfer, call cancellations, multi-party conferencing, media negotiation, and do-not-disturb, among other features. It uses a freely available media engine, Nevot, to handle the RTP and audio." [12]

The software modules/components which come packaged with siphone are summarized in Table 3.

Table 3: Contents of the siphone software package

User Agent Client	**User Agent Server**	**Media / communication tools**	**Site/domain SIP server**	**User location service**
siphone	siphone	Nevot (Solaris only)		

6.2 siphone and Evaluation Topics

6.2.1 Documentation

The primary documentation for Siphone — a brief (8 page) installation guide and user's manual — is short and relatively straightforward. The end-user must fill in personal information (e.g., sip address) the first time she uses *siphone* on the current host. The fields to be filled in are described in siphone's user's manual. The kind of information, format, and order in which the fields are to completed is not so well described, however; thus, completion of the fields requires a certain bit of SIP knowledge. Once these parameters were set correctly, the program was easy to use.

6.2.2 Installation

System requirements

Siphone is available for Linux, Solaris and Windows. On Unix it uses the same PMM-enabled version of Nevot as Mint does. Unfortunately, a Linux version of Nevot is not yet available. It is not known what kind of media agent siphone uses on Windows. Of interest for Windows is that a Netscape plug-in for siphone is available.

The user interface is written in Java 1.1, while the SIP message parser is in a native library. A Java runtime for Java 1.1 is therefore needed in order to run the program. Note that the Java 1.2.x (x = 0, x = 1) libraries from Javasoft have a bug that blocks all PMM messages. This bug has been corrected in Javasoft's Java 1.2.2 libraries.

Configuration employed

We have tested siphone on the Linux platform and on Solaris; the hardware configuration employed was the same as that used for MInT, see section 5.2.2.

We used Siphone version 0.54, which was the latest available version. On Solaris we used this with Java versions 1.1.5 and 1.2.1. This worked fine, except for the known multicast bug in Java 1.2.

Javasoft does not provide Java runtime for Linux. Red Hat 6.0 comes with the *kaffe* Java runtime. This was found to have an insufficient level of quality. Especially, the window libraries have a lot of bugs. We later installed the Blackdown JDK 1.1.7v3, which was recommended by the Siphone producers. This JDK looked and performed like Javasoft's JDKs for Solaris.

Installation complexity

The installation of siphone is simple: Unzip the provided archive to the destination directory. The program comes with two startup scripts; one for debug operation where all the PMM and SIP messages are printed on the console, and one for normal operation where this output is redirected to /dev/null. The scripts must be edited by hand to fill in paths to the directories containing the siphone software and Java runtime. This is easy.

There is no special per-user installation, but the user has to fill in personal information the first time she uses Siphone on a given host. The names of the preferences files created is controlled by the startup script. The startup script can be edited to make siphone use the same file on all hosts; in this configuration, a user will have the same personal setup on all hosts she uses siphone on. Still, the individual user can not choose whether she wants one single setup or a per-host setup.

6.2.3 Functional performance

Table 3 above summarizes *siphone*'s SIP-related functional components. The performance of these elements is summarized in Figure 9 and the associated commentary notes. Note that the legend employed in Figure 9 is explained in section 4.4.

Testing diagram

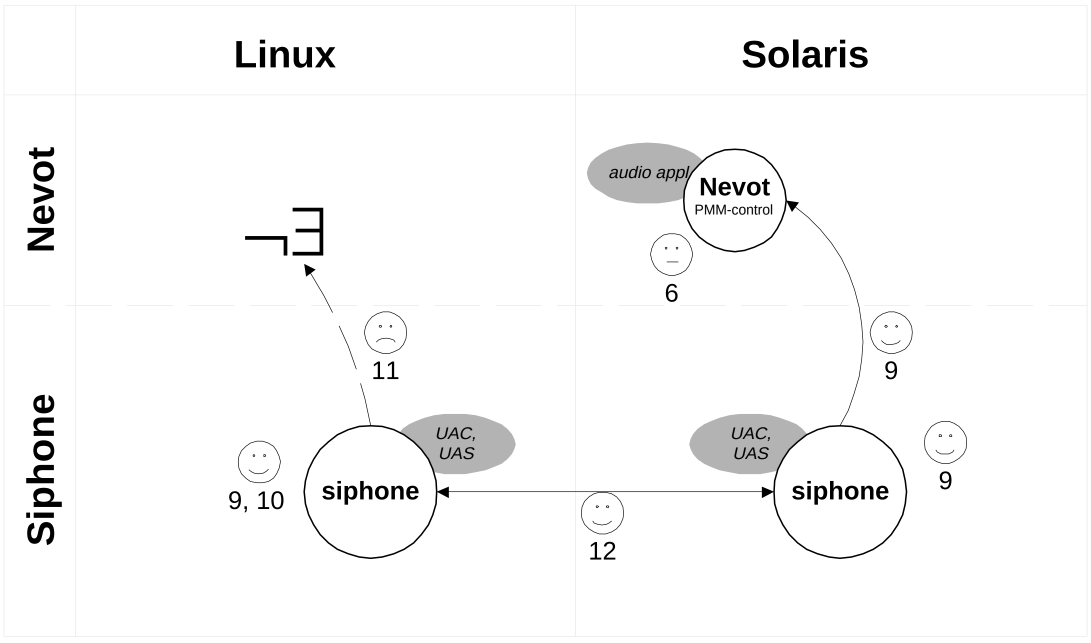


Figure 9: siphone testing

Notes about testing

6. *Nevot* has hardly any user controls, since it is supposed to be controlled via (something like) *isc*. Lack of volume controls is annoying, for instance, when one tries to use *Nevot* without the *isc* window on the screen. This is the case when *Nevot* is used together with *siphone*. On Solaris, speaker and microphone volume can be adjusted using the Sun *audiotool*. Running all of these programs together, however, makes the screen quite crowded with windows.

9. *siphone* is Java-based and works as expected on Solaris. It uses the same PMM-based version of *Nevot* as its audio media agent, just as MInT does. Lack of volume controls in *Nevot* is annoying, however (see note 6). In *siphone*, the end-user has to fill in personal information (e.g., sip address) the first time she uses *siphone* on the current host. The fields to be filled in are described in siphone's user's manual. The kind of information, format, and order in which the fields are to completed is not so well described, however; thus, completion of the fields requires a certain bit of SIP knowledge. Once these parameters were set correctly, the program was easy to use. *siphone* is also very fast compared to MInT. It should be noted that *siphone* only is in version 0.54.

10. *siphone* is Java based. Javasoft does not provide Java environments for Linux. Several projects exist to bring Java to the Linux platform. Red Hat is delivered with the *kaffe* Java runtime environment and the *pizza* and *guavac* Java-to-bytecode compilers. The compilers seems to work very fine. *Kaffe,* however, does not have the required level of quality. Especially the windowing libraries seem to fail quite often.

    Lucent's Bell Labs — the producers of *siphone* — recommends the Blackdown Java for Linux, which cooperates with Javasoft in bringing Java to the Linux platform. *siphone* worked very well with Blackdown JDK 1.1.7v3, except for problems mentioned in note 11 (below). It should be noted that bugs in the official Java libraries from Javasoft prevents the PMM message bus from working with Java version 1.2.x (x = 0, 1). This bug has been corrected in Javasoft's Java 1.2.2 libraries.

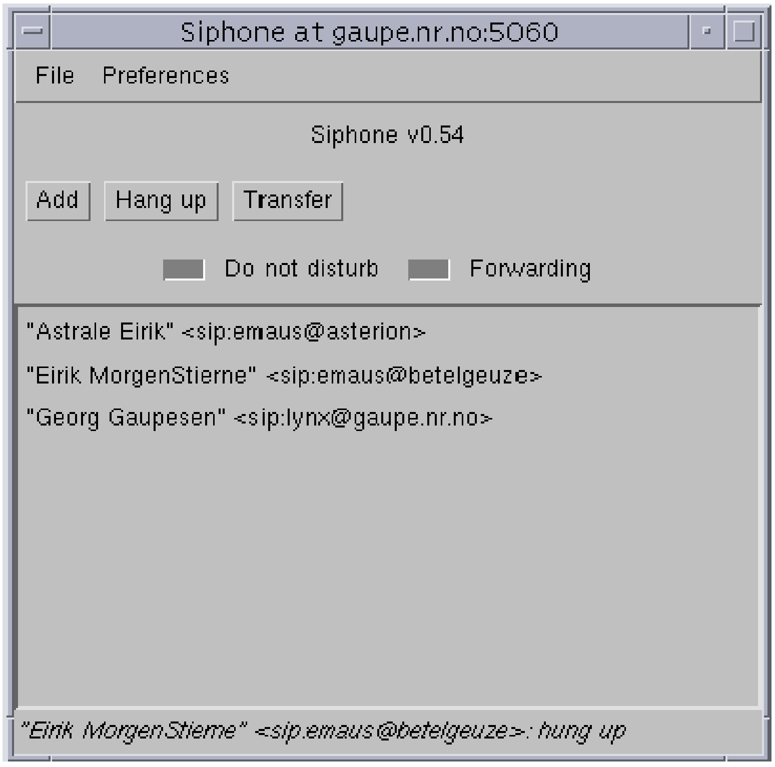
11. There is currently no media agent for *siphone* on Linux. The program tries to start *Nevot*, but this fails since there is no known Linux version of *Nevot*.

12. Calling another *siphone* user from *siphone* works very well, provided that you know which host the callee is logged in on (i.e. in this kind of call, the caller needs to knows the hostname of the host upon which the callee's instance of *siphone* is running).

    Delays sometimes exceeded 5 seconds from when the caller pushed the ”call”-button until the ”incoming call” window popped up on the callee's side. Delays in the Java windowing system appear to be the main reason for these delays. The first notifications on the callee's side are normally obtained in about one second, when used over a LAN. There seems to be no difference between *siphone*s running on Solaris and Linux at this point.

6.2.4 Usability

The user interfaces for siphone and Nevot are presented in Figure 10 and Figure 11, respectively.



"Astrale Eirik" <sip:emaus@asterion>

"Eirik MorgenStierne" <sip:emaus@betelgeuze>

"Georg Gaupesen" <sip:lynx@gaupe.nr.no>

"Eirik MorgenStierne" <sip:emaus@betelgeuze>: hung up

Figure 10: siphone user interface

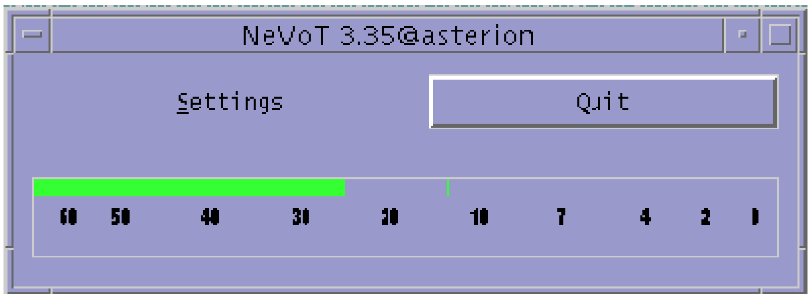


Figure 11: Nevot's user interface

As evidenced by Figure 10, the range of features and functions *immediately visible* within siphone are quite few; one gets the immediate feel that siphone should be easy to use. Behind the 'Preferences' button, however, are a number of fields which must be correctly set, prior to the first use of the application. As mentioned in section 6.2.1, siphone's user manual provides some information about these fields; yet, to fill them in correctly requires some (limited) knowledge of SIP. Once these values are correctly entered, siphone is in fact quite simple to use.

Nevot's user interface (Figure 11) does not offer controls for volume adjustment. Behind Nevot's ”Settings” button is a menu through which the user may control echo suppression, silence detection, timeouts, debugging, etc. — parameters which are essentially uninteresting for the common consumer, as long as it sounds good. It appears that the reason for this interface condition is that Nevot arises from a research institution, presumably with the primary purpose of functioning as a research tool, rather than a consumer product.

On Solaris, controls for volume adjustment are available via audiotool (Figure 12). For the purpose of making simple VoIP calls, the need to employ an ”extra” interface may easily seem rather absurd.

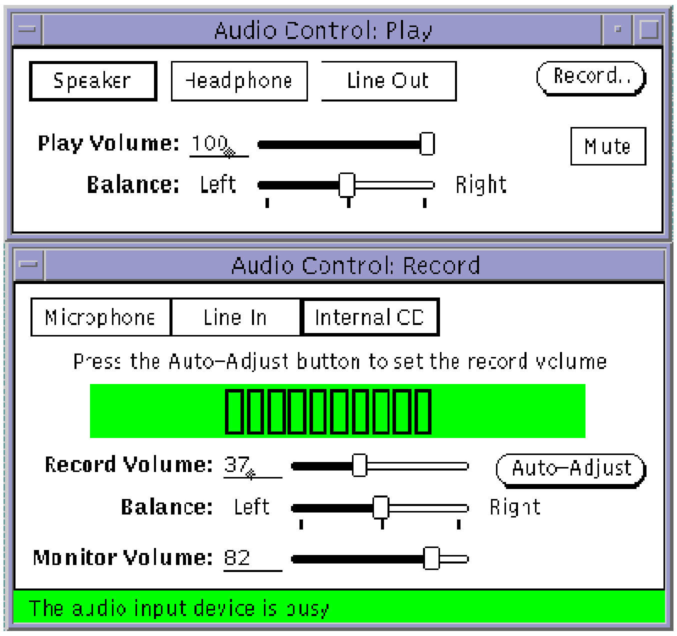


Figure 12: audiotool's user interface

6.2.5 Architecture, extensibility, openness

When downloaded, the siphone software arrives as a limited set of files; these include: textual documents (e.g., a README file, client installation documentation and a user's manual), along with binary files and libraries. There is little to no information available concerning the architecture of siphone.

Concerning extensibility and openness, siphone employs PMM signaling [9], just as MInT does. It implements SIP's UAC and UAS elements, of course — elements which can be used as components within an open and more comprehensive multimedia conferencing framework.

7. Evaluation of sipd from Columbia

7.1 Introduction to sipd

A good description of sipd is found on its WWW page [21]:

> *sipd is a SIP redirect, proxy and registration server that provides name mapping, user location and scripting services. It can use external routines to do the actual work of resolving aliases (including group names), mapping names and locating users. It also allows users to register their current location with the server. Users can be registered at multiple locations. Each user can register a script in any scripting language that will be executed when receiving a call. The scripting interface conforms to the SIP cgi-bin interface. (In version 1.0, scripting is only available for redirect services.)*
>
> *The server currently understands the ACK, BYE, CANCEL, INVITE, OPTIONS and REGISTER requests. Invitations and registrations can be authenticated using basic and digest authentication.*
>
> *If the user is not registered or cannot be found using the dynamic user location program, the server returns 480 (Temporarily unavailable).*
>
> *The server supports LDAP search based on mail and last/full name filters (e.g. Joe.M.Doe or Joe.Doe or Doe). The search starts from the object specified by LdapBaseObject configuration parameter and goes down the subtree. It retrieves mail, telephone, and fax attributes if any.*

Schulzrinne, et.al. [21]

The software modules/components which come packaged with sipd are summarized in Table 4.

Table 4: Contents of sipd software package

User Agent Client	**User Agent Server**	**Media / communication tools**	**Site/domain SIP server**	**User location service**
			Sipd	aliases, namemapper, lswhod, ishere, tracker

7.2 sipd and Evaluation Topics

7.2.1 Documentation

The sipd documentation is a short (6 page) document formatted in the style of Unix *man* pages. It describes the features which can be set by the user, as well as the options for the program. Knowledge of SIP is required to understand the manual. This is acceptable, because sipd is supposed to be installed and set up by professional system operators only. The documentation available must nevertheless be regarded as ’sparse’.

7.2.2 Installation

System requirements

Binary and source versions are available for quite a low price[14]. The code is advertised to run on Solaris, FreeBSD, Linux and MS-Windows, with other Unix platforms being available upon request; it is still regarded as being in a “beta” version by its implementers. Memory and processing requirements seem to be low.

Configuration employed

We have tested Sipd on the Linux platform and on Solaris; the hardware configuration employed was the same as that used for MInT and siphone, see section 5.2.2. At least four “new” releases of the sipd code were made available within the six weeks we were testing the system; most of these releases concerned bug fixes, though certain contained new features; this condition indicates that work on sipd is clearly in progress.

Installation complexity,

Installation is easily performed: First unzip archive to destination directory, then edit the configuration file. The documentation contains a sample configuration file for a simple setup. This can be used with minor modifications (e.g., setting the server name and domain name). Advanced features (e.g., user authentication) can be added later by editing the configuration file and restarting Sipd.

7.2.3 Functional performance

Table 4 above summarizes *siphone*’s SIP-related functional components. On the server side, the functional capabilities advertised for sipd are quite thorough (e.g., SIP redirect, proxy and registration servers, including name mapping, user location and scripting services).

To our dismay, we were unable to make Columbia University’s *sipd* fully work; **it is very important to point out, however, that SIPtel’s project calendar was only able to yield approximately two days for the investigation and eventual rectification of the problem.**

With sipd, *Register* messages seemed to work; however, when sending an *Invite* message, the user location program *lswhod* (which is run when *sipd* receives an *Invite* message) crashed with a Tcl error message. Since the crash led to no user being found, *sipd* eventually returned the message ”604 Does not exist anywhere”. The problem was experienced on both platforms.

The problem we encountered was quite a surprise and disappointment, since we expected that the SIP server delivered by Columbia University would work immediately, “without any trouble”. We believe however, that the program works fine in many other environments, and that the problem we experienced is probably either possible to solve locally or has been fixed in newer versions. The reason behind this belief is that (a) Columbia is one of the major SIP proponents and therefore (b) it is unlikely that Columbia University would ship a SIP server program that doesn’t support *Invite* messages. In addition, this problem has not been registered within the *sipd* user’s (email) group. Furthermore, messages within the sipd user’s group logically indicate

[14] A binary license for sipd and related utilities costs $495. This charge includes updates for one year. A source license for research and evaluation purposes costs $1,000. Source and binary licenses are available without charge for non-profit research institutions such as universities or government laboratories.

that other sipd users have progressed to call phases which are sequentially “downstream” with respect to the *Invite* message.

It is truly unfortunate that we haven’t had the time to determine the cause of the problem we encountered with sipd, within the first phase of SIPtel.

7.2.4 Usability

sipd operates as a server. For this reason, usability for client-end users is not immediately relevant for discussion (i.e., there is no user-interface for sipd). Of course, it is always *possible* that some client-end users may find sipd’s feature set and/or processing speed to be unsatisfactory; however, we shall not speculate upon matters of this kind.

For system administrators which choose to install and ”use” sipd within their domain, we expect that sipd performs as well as any average, beta-class software product.

7.2.5 Architecture, extensibility, openness

As mentioned above, access to sipd’s source code is available though purchase and licensing. The basic source license does *not* entitle the licensee to sell products derived from the software. However, source licenses for product development are also available, with either a one-time fee or per-copy royalties [22].

8. Interoperability Testing

In addition to the evaluation of each of MInT, siphone and sipd, certain cross-component experiments were performed. These tests are summarized below in the cross-component testing diagram Figure 13 and its associated notes. Note that the legend employed in Figure 13 is explained in section 4.4.

Cross-component testing diagram:

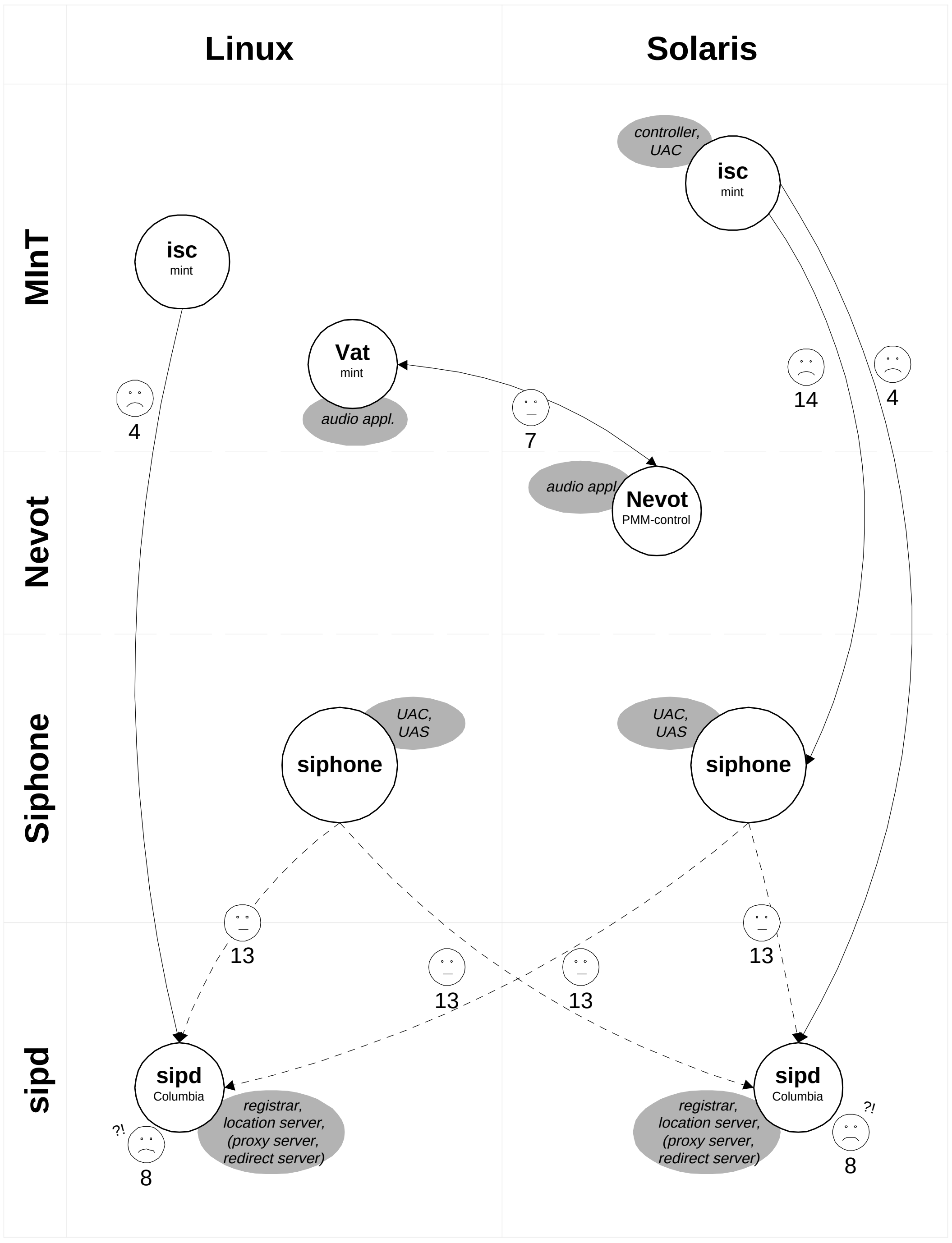


Figure 13: Cross-component testing

Notes about cross-component testing

4. GMD tested and confirmed that the MInT system is not interoperable with Columbia’s *sipd* software. This is due to the fact that the version of *sipd* used in MInT was “frozen” prior to the finalization of the SIP standard. In contrast, Columbia’s *sipd* software has kept abreast of the SIP standardization effort and is conformant with the currently released version of SIP.

7. *vat* and *Nevot* are interoperable, such that *vat* users and *Nevot* users may talk to each other. Lack of command-line options and user controls in *Nevot* makes it hard to set up the connection, however.

8. We were unable to make Columbia University’s *sipd* fully work. *Register* messages seemed to work; however, when sending an *Invite* message, the user location program *lswhod* (which is run when *sipd* receives an *Invite* message) crashed with a Tcl error message. Since the crash led to no user being found, *sipd* eventually returned the message ”604 Does not exist anywhere”. The problem was experienced on both platforms.

13. We were not able to test interoperation between Columbia *sipd* and *siphone*, since we couldn’t get Columbia’s *sipd* to work fully (see note 8). Registrations seemed to work fine, but the problems regarding user location stopped further testing of call initiation and set-up.

14. *isc* and *siphone* could not interoperate. When trying to call *siphone* from *isc*, *siphone* responded with “Bad Request”. This is due to the fact that MInT is based upon a draft version of the SIP standard, prior to the standard’s finalization. In contrast, Lucent’s *siphone* software has kept abreast of the SIP standardization effort and is conformant with the currently released version of SIP.

9. Conclusions and Recommendations

9.1 General Comments

9.1.1 VoIP and the market

During the last three years, the overwhelming majority of the large industrial players within traditional telephony have begun to develop, secure and actively deploy H.323 technology. This market movement began prior to the completion of the first version of H.323, since many of the organizational forces behind H.323 have been telecom actors. This market movement has been characterized by an intense strategic-positioning activity involving alliances, splits, fusions and take-overs. The value of the market-at-stake is enormous.

The ”early” selection of H.323 technology by many of these actors can be attributed simply to the fact that the first official version of the H.323 standard appeared in the late Fall of 1996 [45]. In contrast, SIP received its status as an IETF standard in March, 1999. Additionally, many of the large industrial players could begin the move towards H.323 since they had:

a) their names and reputations already well-established in the marketplace,

b) their customer base in place and

c) their billing systems in place.

Still, these actors have not ignored SIP. A number of them have carried out (or are carrying out) internal SIP investigation and experimentation activities (e.g., Ellemtel [51]), and many are actively following the development of SIP in order to see where it leads and whether they will ultimately need to make their H.323-based systems deal with SIP traffic. Some players, such as Lucent, appear to be designing for H.323 — SIP interoperation right from the start [23].

It may be that certain telecom actors are somewhat wary about SIP. Using a SIP-based approach, it is possible to create a client-side interface offering an ”open-ended” API for service creation by end-users. Ultimately coping with user-created services could lead to new complexities within server-side design and processing.

The market also bears witness to a number of ”new-comers” — actors such as dynamicsoft [46] which are already designing and developing concrete products for SIP. Certainly, these actors must believe that SIP is not simply a niche market. SIP has the potential to capture ”significant enough ”margins within the VoIP market that actors are beginning to wager upon its value. In fact, this has been evidenced by the increase in SIP-related implementation work which has arisen during the course of the project thus far.

9.1.2 VoIP technology and integrated IP services

One highly challenging problem for VoIP — perhaps a problem which will be an extremely significant market determinant — involves the integration of VoIP with other IP-based services such as SNMP, http, ftp, etc. In this area, one should consider the value end-users place upon having a single application — with an intuitive interface — which handles all of their basic communication needs.

A great deal of current work within the H.323 telecom domain concerns the design of service architectures which can interoperate with e.g., H.323’s Gatekeeper. These architectures need to be designed such that they can accommodate existing, as well as new kinds of supplementary services for telephony. In addition, these architectures must be open enough to allow for the seamless integration of other IP-based services. From an architectural perspective, this is one of the major challenges for H.323-based systems.

SIP is perhaps at an advantage here, since the protocol itself lends a great deal from other IETF standards (e.g., http, sdp). From an architectural perspective, it is easy to conceive the underpinnings for a browser which integrates WWW browsing, email communication and VoIP. One of the major challenges for SIP-based telephony is the integration of computer-based telephony and standard telephony. It is perhaps most likely that the solution will involve a gateway between SIP and H.323.

9.1.3 Leading SIP proponents and their roles

Henning Schulzrinne, from the Department of Computer Science at Columbia University, and Jonathan Rosenberg, from Lucent's Bell Labs, are two of the most significant developers and proponents of SIP. In this regard, it is interesting to consider the roles of the organizations in which these individuals work. Aside from educating students, Columbia's Department of Computer Science carries out research. In Schulzrinne's case, some great deal of energy goes towards the further development of the Internet and the IETF standards upon which it is based. As a result of the nature of this work, "products" are created which are on the cutting-edge of Internet technology and — perhaps best of all — the price of these products is next-to-nothing.

Lucent's Bell Labs, also has the role of creating research results; still, it is speculated that the kinds the projects funded there must have results relevant to Lucent's product organization. With Rosenberg's work on SIP, Lucent will be in the front-running when it comes to the development and release of SIP-based products; siphone is an excellent example of this condition.

9.2 Conclusions from the Evaluation

When drawing conclusions from SIPtel's review and evaluation work, it is important to remember the influence of perspective. As mentioned in section 4.2.1, it is important to keep in mind that applications can be used by different classes of users and, even within a single user class, an application can be used to serve different purposes.

9.2.1 MInT from GMD FOKUS

To begin, it must be remembered that MInT was developed in parallel with the SIP standard itself. In order to complete MInT within a fixed resource frame, it was necessary to base the system upon a pre-standardized version of SIP. In and of itself, this single condition has placed (the currently available version of) MInT in a position which likely renders it unable to interoperate with components which have been (and will be) developed in conformance with the standard. This fact has been confirmed within SIPtel's interoperability testing work (see section 8).

MInT on Linux appears to have reached an impasse (see footnote 11). From the technical details in MInT's User Guide, as well as the limited testing carried out here, the Solaris version of MInT appears to work as advertised. As mentioned earlier, we have not had time to perform any testing of video conferencing using MInT.

MInT relies on existing media applications for acquisition, exchange and playback of multimedia data (e.g., vic, vat, Nevot). On the one hand, these applications have been available for many years are quite stable. When operating over the Internet, however, their performance is subject to variations in network load. MInT incorporates a cross-application Reservation Agent for QoS control using RSVP (see section 5.3). To effectively utilize this feature, however, requires end-to-end support for RSVP within all routers along the data path.

MInT is judged to have a strong, flexible and open architecture for multimedia conferencing. Its user interface is judged to be intelligible for most users familiar with integrated, multimedia applications.

In its current state, MInT can be used in the near-term to gain greater familiarity with SIP-based, multimedia conferencing. It can presumably be used to perform interesting and informative tests and experiments with RSVP, as well. A long-term ”commitment” to *the currently available version* of MInT is not recommended, however, since the system is based upon a pre-standardized version of SIP. Extensions to the existing source code may also be difficult, due to limited documentation of the code.

9.2.2 siphone from Lucent

siphone is currently free. It is presently available as version 0.54 and it conforms to the SIP standard. It is expected that siphone (or some derivative of siphone) will continue to be a product; the grounds for this expectation are based in our perception of Lucent’s interest in SIP, especially as made evident in their WWW-based information concerning products which (will ultimately) enable SIP — H.323 interoperation.

siphone runs on the Linux, Solaris and Windows NT platforms. It uses a freely available media engine, Nevot, to handle the RTP and audio. Presently, Nevot is only available for the Solaris platform.

siphone currently supports audio only. Like MInT, it uses an existing multimedia application for the acquisition, exchange and playback of audio (i.e., Nevot); this condition can yield variations in performance when operating over the Internet.

siphone’s internal architecture is not known. Its user-interface is simple to use, once the user has correctly configured system parameters (see section 6.2.1).

siphone appears to be a good candidate for further work with SIP-based VoIP. In this regard, it is best suited as a computer-based substitute for (or complement to) a standard telephone, rather than a vehicle for research and experimentation with Internet protocols.

9.2.3 sipd from the University of Columbia

sipd conforms to the SIP standard. Source and binary licenses for sipd can currently be obtained at a low cost from the University of Columbia. These licenses are available without charge for non-profit research institutions such as universities or government laboratories. sipd is presently released as beta-class software, and the rate of new releases indicates active work upon the software. sipd is advertised to run on Linux, Solaris and Windows NT.

The organization behind sipd is a research department which, among other things, is devoted to the further development of the Internet and IETF standards, including SIP. On the one hand, sipd is a pre-product on the cutting-edge of SIP technology. Still, it is unclear to what extent sipd shall appear as a full-blooded product within the competitive marketplace.

Though we had certain problems in getting the software to operate correctly at our site (see section 7.2.3), sipd is judged to be a good candidate for further work within SIPtel. It can serve to support efforts targeting the development of a simple SIP-based telephone for common consumers, as well as efforts targeting the development of SIP-based research environments.

9.3 Trends and Near-term Expectations

9.3.1 Client side

In the near-term, free or *very* low-priced User Agents (e.g., siphone) will be manifest. The major advantages for the end user are price and availability (i.e., via WWW). The UAs will certainly work — at least in basic operational modes — and they will be loosely coupled so as to be able to operate together with the user's ”favorite” audio application; of course this will require that the audio application is enabled to work with together with the UA). The disadvantages of these UAs will be that they may not be packaged together with a free audio player (e.g., Nevot). Lack of an integrated user-interface between the UA and the audio player will likely also manifest itself, resulting in a certain less-then-optimal appearance on the screen (e.g., waste of screen real estate).

On the client side, it is also expected that low-priced — perhaps even free — products will appear which bundle together a UA and an audio player as an integrated component. Here, it is expected that the product competition will primarily concern features-for-price though, as with mobile phones, some of the competition will simply concern the product's ”look”. Since the user may pay for these products, the major advantage for user will be the product guarantee and service agreement implied by the purchase. Aside from product crashes, it is not easy to project which concrete disadvantages users may encounter when using products of this kind.

9.3.2 Server side

Server-side trends are somewhat difficult to project since, in the near-term, SIP will always be ”threatened” by H.323. However, it is speculated that relatively low-to-medium priced SIP servers will be available — servers which can quite certainly correctly handle the most routine kinds of SIP traffic (*sipd* is probably a good example). The advantage here is that these kinds of products will be sufficient in order to provide SIP clients with the kinds of support required for basic, low-cost VoIP using SIP. The kinds of disadvantages which might arise include the stability of more advanced features and services, as well as the discontinuance of a product line due to competitive losses, industrial take-overs, etc.

Assuming SIP makes a firm stand in the face of H.323 — which may well be a valid assumption — more costly SIP servers and SIP-related technologies (e.g., SIP-H.323 gateways) will certainly become available. Here, it is expected that product competition will concern features-for-price, seamless integration with H.323, integrated services, etc.

9.4 Recommendations for SIPtel

9.4.1 Importance of SIP

With regard to group communication, Baird and Weinberg conclude that verbal communication is the most important element in a communication process [56]. From a perspective of potential *profits*, this conclusion is clearly affirmed in the marketplace by the intense activity directed towards achieving global, interoperable ”voice on the net” functionality and services.

SIP is currently second-runner to H.323, but the balance may well begin to shift as new SIP-based products begin to emerge on the market. In this regard, it will be interesting to see which of these technologies is able to achieve the most rapid and seamless integration of existing IP services. The emergence of SIP — within an area which has formerly been a telecommunication strong-hold — it therefore judged to be of great significance.

9.4.2 Phase 1 status and afterthoughts

At the close of Phase 1, there are two areas which we feel fell slightly short of our original ambitions in the project. First of all, we didn’t achieve any true form of “small pilot” activity. This goal was not specified explicitly in the workplan, but we had hoped to test and evaluate MInT’s communication framework between UNINETT in Trondheim and NR in Oslo.

Secondly, we had the ambition of performing “thorough testing and evaluation” of MInT. As explained in section 4.3, this effort was hindered by certain faults. We therefore feel that the Phase 1 evaluation work was not as complete as judged necessary for a conclusive selection of hardware / software for a future pilot activity. In other words, more evaluation work should be performed before launching a pilot.

9.4.3 Opportunities for Phase 2

UNINETT’s intentions

The recommendations for SIPtel Phase 2 activity properly depend upon UNINETT’s intentions. Once again, it is necessary to consider which classes of users one wishes to support: common consumers or highly-skilled developers and users. Furthermore, it is necessary to consider which purpose a VoIP application should serve; that is, whether UNINETT should target the (ultimate) deployment of a simple-to-use computer telephone, a conferencing framework or a research vehicle for articulated investigation of VoIP applications and network services. Proposals for further work are clearly dependent upon such issues.

Recommendations for further work within SIPtel

Assuming that UNINETT wishes to support their computer-literate users — as well as their highly skilled ones — with a (relatively) simple-to-use computer telephone, it is suggested that SIPtel Phase 2 include the two primary activities:

1. deployment and evaluation of a SIP-based VoIP pilot and
2. a SIP-oriented technology watch.

The pilot activity should preferably contain user groups which represent two classes of users (e.g., high vs. low ”multimedia literacy”), in order to obtain a wider base for end-user evaluation. The tasks within the pilot activity should include:

- to test and determine the set of components to use within the pilot (it is recommended that siphone and sipd be started with);
- to identify and establish the user group(s);
- to specify the evaluation goals for each group; and,
- to carry out user testing and VoIP / component evaluation.

The recommended areas to follow within the technology watch activity should include:

- integration of VoIP and other standards concerning conferencing functionality (e.g., IMPP);
- to continually follow and review emerging SIP-related components;
- to study approaches for standards-based integration of IP services; and,
- to study activity concerning SIP ↔ H.323 interoperation approaches and technology.

References

[1] The IMiS-Ericsson project; see: http://www.nr.no/imis/imis-e/

[2] UNINETT WWW-site: http://www.uninett.no/

[3] Pandya, R., "Emerging mobile and personal communication systems," IEEE Communications Magazine , vol. 33, pp. 44--52, June 1995.

[4] GMD FOKUS WWW-site; see: http://www.fokus.gmd.de/

[5] MInT WWW-site, see: http://www.fokus.gmd.de/research/cc/glone/products/mint/content.html

[6] USMInT WWW-site, see: http://www.fokus.gmd.de/research/cc/glone/projects/usmint/content.html

[7] Henning Schulzrinne's home page: http://www.cs.columbia.edu/~hgs

[8] ISC (Integrated Session Controller); see: http://www.cs.columbia.edu/~hgs/isc

[9] PMM (Pattern Matching Multicast); see: http://www.cs.columbia.edu/~hgs/pmm

[10] Schulzrinne, Henning; "Dynamic Configuration of Conferencing Applications using Pattern-Matching Multicast", Proc.of NOSSDAV'95 (5th International Workshop on Network and Operating System Support for Digital Audio and Video); Durham, New Hampshire, April 18-21, 1995. Available as: ftp://gaia.cs.umass.edu/pub/Schu9504:Dynamic.ps.gz

[11] Sisalem, D., User Handbook of the Multimedia Internet Terminal (MInT), available on http://www.fokus.gmd.de/research/cc/glone/products/MInT/

[12] Bell Labs' SIP project home page; see: http://www.bell-labs.com/project/sip/

[13] VIC, http://www-nrg.ee.lbl.gov/vic

[14] VAT, http://www-nrg.ee.lbl.gov/vat/

[15] NEVOT, http://www.cs.columbia.edu/~hgs/nevot

[16] MPOLL, a real-time opinion polling and rating collection tool; see: ftp://debra.dgbt.doc.ca/pub/mbone/mpoll/

[17] Schubert, Ilona; Sisalem, Dorgham; Schulzrinne, Henning; "A Floor Control Application for Light-Weight Multicast Conferences", available as: ftp://ftp.fokus.gmd.de/pub/glone/papers/Schu9806:Floor.ps.gz
See also: Schubert, Ilona; Sisalem, Dorgham; Schulzrinne, Henning; "A session floor control scheme," in Proc. of International Conference on Telecommunications, (Chalkidiki, Greece), June 1998.

[18] Maus, Eirik; Holmes, Peter; "Sammenligning av SIP og H.323". Notat Nr. IMEDIA/10/98, Oktober 1998. See: http://www.nr.no/publications/imedia_10_98.ps

[19] SIP Implementations, see: http://www.cs.columbia.edu/~hgs/sip/implementations.html

[20] SIP Frequently Asked Questions (FAQ) page; see: http://www.cs.columbia.edu/~hgs/sip/faq.html

[21] sipd (a SIP redirect, proxy and registration server), see: http://www.cs.columbia.edu/~hgs/sipd/

[22] sipd licensing; see: http://www.cs.columbia.edu/~hgs/sipd/license.html

[23] Lucent Technologies Softswitch; see: http://www.lucent.com/IN/softswitch.html

[24] ITU WWW-site; see http://www.itu.int/

[25] ITU-T Recommendation H.323: Packet-based multimedia communication systems (Feb. 98); see: http://www.itu.int/itudoc/itu-t/rec/h/s_h323.htm

[26] IETF Home page; see: http://www.ietf.org/

[27] IETF working group MMUSIC (Multiparty Multimedia Session Control); see: http://www.ietf.org/html.charters/mmusic-charter.html

[28] Handley, M., Schulzrinne, H., Schooler, E., Rosenberg, J.; “SIP: Session Initiation Protocol”, IETF RFC 2543; see: http://www.ietf.org/rfc/rfc2543.txt

[29] Handley, M., Schulzrinne, H., Schooler, E., Rosenberg, J.; SIP: Session Initiation Protocol, (final) Internet Draft : available as: ftp://ftp.informatik.uni-bremen.de /pub/doc/internet-drafts/draft-ietf-mmusic-sip-12.ps

[30] Handley, M. and Jacobson, V.; "SDP: Session Description Protocol”, IETF RFC 2327, April 1998; see: http://www.ietf.org/rfc/rfc2327.txt

[31] Braden, B., Zhang, L., Berson, S., Herzog, S. and Jamin, S.; "Resource ReSerVation protocol (RSVP) -- version 1 functional specification", IETF RFC 2205, October 1997; see: http://www.ietf.org/rfc/rfc2205.txt

[32] Schulzrinne, H., Casner, S., Frederick, R. and Jacobson, V.; "RTP: a transport protocol for real-time applications", IETF RFC 1889, Internet Engineering Task Force, Jan. 1996; see: http://www.ietf.org/rfc/rfc1889.txt

[33] Schulzrinne, H., Lanphier, R. and Rao, A.; "Real time streaming protocol (RTSP)", RFC 2326, April 1998; see: http://www.ietf.org/rfc/rfc2326.txt

[34] Handley, M., Perkins, C., Whelan, E., "SAP: Session announcement protocol," IETF Internet Draft, June, 1999; see: http://search.ietf.org/internet-drafts/draft-ietf-mmusic-sap-v2-01.txt

[35] Uniform Resource Locators (URL), IETF RFC 1738; see: http://www.ietf.org/rfc/rfc1738.txt

[36] First SIP Bake-Off, see: http://www.cs.columbia.edu/~hgs/sip/bakeoff.html

[37] First SIP Bake-Off: press release April 12, 1999: http://www.cs.columbia.edu/~hgs/sip/bakeoff/bakeoff-press.txt

[38] Second SIP Bake-Off; see: http://www.pulver.com/sip/ and http://www.cs.columbia.edu/~hgs/sip/bakeoff2.html

[39] Third SIP Bake-Off; see: http://www.cs.columbia.edu/~hgs/sip/bakeoff3/

[40] SIP-related drafts; see: http://www.cs.columbia.edu/~hgs/sip/drafts.html

[41] Pulver WWW-site (the “Voice of Telephony on the Net”), see: http://pulver.com/

[42] E-mail from Hiyan Alshawi hiyan@research.att.com

[43] sipc (a SIP client and user agent), see: http://www.cs.columbia.edu/~hgs/sipc/

[44] SIP standard draft version 12: draft-ietf-mmusic-sip-12.txt. No longer available, as it is replaced by [28].

[45] ITU-T Recommendation H.323 (11/96) – “Visual telephone systems and equipment for local area networks which provide a non guaranteed quality of service”; see: http://www.itu.int/itudoc/itu-t/rec/obsolete/h/h.323.html

[46] Dynamicsoft web pages, see: http://www.dynamicsoft.com

[47] Audiotrix Phone Adapter, by Mediatrix; see: http://www.mediatrix.com/telecpro/index.html

[48] The Pulver Report monthly e-mail magazine on IP/Telephony integration, July 1999 issue. Subscriptions freely available from http://www.pulver.com

[49] Mediatrix website: http://www.mediatrix.com

[50] Ellemtel web pages, see http://www.ellemtel.se

[51] E-mail from Rabbe Folgeholm at Ellemtel.

[52] E-mail from Anders Kristensen at HP Labs (Bristol, UK)

[53] E-mail from Bobby Sardana, Object Software Inc.

[54] E-mail from Jay Batson at Pingtel.

[55] E-mail from Lakshman Bijanki, Hughes Software Systems Limited.

[56] Baird, J. Weinberg, S., “Elements of Group Communication”, in *Small Group Communication*, Fifth Edition, R. Cathcart and L. Samovar (eds.), Wm. C. Brown Publishers, pp. 260-274 (1988).

Appendix: Complete Test Diagram and Notes

Figure 14 below summarizes the SIP testing performed in SIPtel. It is a graphical union of Figure 5, Figure 9 and Figure 13.

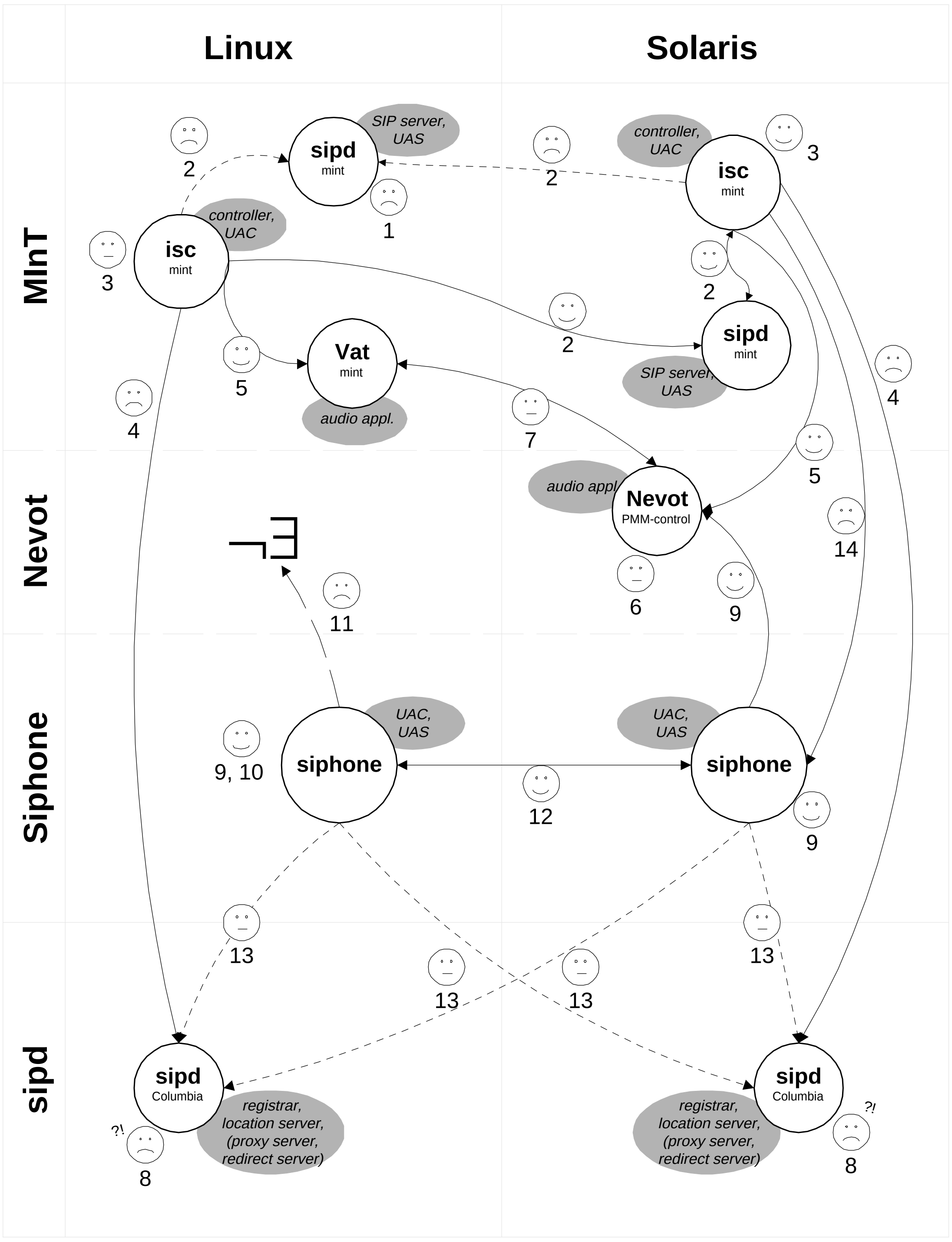


Figure 14: Complete test diagram

Notes about testing

Note that *siphone* and MInT for Solaris both come “packaged with” the same version of *Nevot*. Make special note that two versions of *sipd* have been tested: one is an early version released together with MInT, while the other is a version later released by the University of Columbia. These are marked explicitly in the figure.

1. MInT’s *sipd* doesn’t work on Linux. GMD has identified the bug, but it is unclear what changes they may or may not make to the rectify the problem[15,16]. Presently, one should not expected that GMD puts too much effort into correcting the bugs in MInT’s *sipd* software for Linux.
2. MInT’s *sipd* software works on Solaris. However, as long as MInT’s *sipd* doesn’t work on Linux, there is no way of making calls to a MInT system running on Linux or through a site- / domain-central MInT SIP server running on Linux. Only Solaris MInT users can receive calls. Linux users may, however, call MInT users on Solaris.
3. MInT’s *isc* works on both Linux and Solaris. Some Tcl errors occur on Linux. This does not seem to have impact on the overall functionality (unless the user chooses the ”abort”-button on the error message box.)
4. GMD tested and confirmed that the MInT system is not interoperable with Columbia’s *sipd* software. This is due to the fact that the version of *sipd* used in MInT was “frozen” prior to the finalization of the SIP standard. In contrast, Columbia’s *sipd* software has kept abreast of the SIP standardization effort and is conformant with the currently released version of SIP.
5. MInT uses *Nevot* and *vat* as media agents for audio. Both *vat* and *Nevot* are originally developed as stand-alone audio applications. MInT supplies special versions which are controllable over the PMM message bus. *Nevot* is the preferred audio tool, but *Nevot* exits only for Solaris[17]. Therefore, the audio media agent packaged with the Linux version of MInT is a PMM-enabled version of *vat*.
6. *Nevot* has hardly any user controls, since it is supposed to be controlled via (something like) *isc*. Lack of volume controls is annoying, for instance, when one tries to use *Nevot* without the *isc* window on the screen. This is the case when *Nevot* is used together with *siphone*. On Solaris, speaker and microphone volume can be adjusted using the Sun *audiotool*. Running all of these programs together, however, makes the screen quite crowded with windows.
7. *vat* and *Nevot* are interoperable, such that *vat* users and *Nevot* users may talk to each other. Lack of command-line options and user controls in *Nevot* makes it hard to set up the connection, however.
8. We were unable to make Columbia University’s *sipd* fully work. *Register* messages seemed to work; however, when sending an *Invite* message, the user location

[15] The reason for GMD’s hesitation in this matter first concerns *sipd*’s copyright and redistribution agreements. That is, MInT’s *sipd* was orignally written prior to SIP standardization; additionally, the code was not written by GMD alone. Further development of the *sipd* software has been carried out at the University of Columbia. Currently, it is the University of Columbia who releases and distributes the *sipd* software, not GMD.

[16] Depending upon funding, GMD is also considering the development of a JAVA-based MInT framework which conforms to the SIP standard. Whether and how they re-implement the *sipd* functionality currently packaged with MInT, within such an effort, is currently and open issue.

[17] The MInT team has said that a Linux version is underway. Presumably, that version will be PMM-controllable.

program *lswhod* (which is run when *sipd* receives an *Invite* message) crashed with a Tcl error message. Since the crash led to no user being found, *sipd* eventually returned the message ”604 Does not exist anywhere”. The problem was experienced on both platforms.

9. *siphone* is Java-based and works as expected on Solaris. It uses the same PMM-based version of *Nevot* as its audio media agent, just as MInT does. Lack of volume controls in *Nevot* is annoying, however (see note 6). In *siphone*, the end-user has to fill in personal information (e.g., sip address) the first time she uses *siphone* on the current host. The fields to be filled in are described in siphone’s user’s manual. The kind of information, format, and order in which the fields are to completed is not so well described, however; thus, completion of the fields requires a certain bit of SIP knowledge. Once these parameters were set correctly, the program was easy to use. *siphone* is also very fast compared to MInT. It should be noted that *siphone* only is in version 0.54.

10. *siphone* is Java based. Javasoft does not provide Java environments for Linux. Several projects exist to bring Java to the Linux platform. Red Hat is delivered with the *kaffe* Java runtime environment and the *pizza* and *guavac* Java-to-bytecode compilers. The compilers seems to work very fine. *Kaffe,* however, does not have the required level of quality. Especially the windowing libraries seem to fail quite often.

    Lucent’s Bell Labs — the producers of *siphone* — recommends the Blackdown Java for Linux, which cooperates with Javasoft in bringing Java to the Linux platform. *siphone* worked very well with Blackdown JDK 1.1.7v3, except for problems mentioned in note 11 (below). It should be noted that bugs in the official Java libraries from Javasoft prevents the PMM message bus from working with Java version 1.2.x (x = 0, 1). This bug has been corrected in Javasoft’s Java 1.2.2 libraries.

11. There is currently no media agent for *siphone* on Linux. The program tries to start *Nevot,* but this fails since there is no known Linux version of *Nevot.*

12. Calling another *siphone* user from *siphone* works very well, provided that you know which host the callee is logged in on (i.e. in this kind of call, the caller needs to knows the hostname of the host upon which the callee’s instance of *siphone* is running).

    Delays sometimes exceeded 5 seconds from when the caller pushed the ”call”-button until the ”incoming call” window popped up on the callee’s side. Delays in the Java windowing system appear to be the main reason for these delays. The first notifications on the callee’s side are normally obtained in about one second, when used over a LAN. There seems to be no difference between *siphone*s running on Solaris and Linux at this point.

13. We were not able to test interoperation between Columbia *sipd* and *siphone,* since we couldn’t get Columbia’s *sipd* to work fully (see note 8). Registrations seemed to work fine, but the problems regarding user location stopped further testing of call initiation and set-up.

14. *isc* and *siphone* could not interoperate. When trying to call *siphone* from *isc*, *siphone* responded with “Bad Request”. This is due to the fact that MInT is based upon a draft version of the SIP standard, prior to the standard’s finalization. In contrast, Lucent’s *siphone* software has kept abreast of the SIP standardization effort and is conformant with the currently released version of SIP.