Session Initiation Protocol (SIP) and other Voice over IP (VoIP) protocols and applications

Henrik Ingo[1]

The Session Initiation Protocol (SIP) is an Internet Engineering Task Force (IETF) standard used in Voice over IP (VoIP). The Real-Time Protocol (RTP) is used for the real-time transportation of data such as voice and video. The SIP protocol has its problems, in particular with traversing firewalls and NAT (Native Address Translation). Also encrypted connections is still a developing area. Some historical and current alternatives to SIP are also discussed.

As with other Internet technologies, Free Software has been at the forefront of innovation. Asterisk is a popular IP-PBX server for use in companies and institutions but there are popular proprietary competitors. OpenSER is a strong SIP Proxy in telecom companies.

There are several good Free Software client applications, Ekiga (former GnomeMeeting), KPhone, OpenWengo and minisip are mentioned here. Many have however not become mainstream because of initially only supporting the Linux operating system. Sometimes applications have omitted codecs due to fear of software patents. The ease of its use, especially its peer-to-peer inspired circumventing of firewall and NAT obstacles, has made proprietary Skype the most popular VoIP application so far, but it is argued that alternatives based on open standards will prevail in the end.

1 Sesca Technologies, Finland; henrik.ingo@openlife.cc

TABLE OF CONTENTS

1 SESSION INITIATION PROTOCOL AND REAL TIME PROTOCOL

1.1 Voice over IP

Voice over IP (VoIP) is any technology that facilitates the transmission of voice data over an IP network, such as the Internet. The two (or more) endpoints - often referred to as terminals or clients - could be any device with a microphone and speaker, such as a normal PC computer or something resembling a traditional phone.

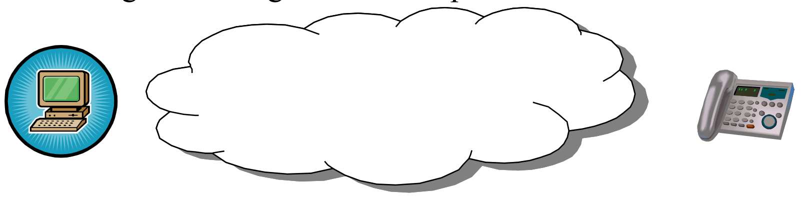


Illustration 1.1: Two devices connected by the Internet

1.2 Session Initiation Protocol

The Session Initiation Protocol (SIP) is a standard used in Voice over IP. It is standardised by the Internet Engineering Task Force (IETF). The current version of the standard is SIP/2.0 and codified by RFC 3261 (Rosenberg & al, 2002).

SIP is a signaling protocol, which means that it is not actually responsible for transmitting the voice data, rather its purpose is to initiate (hence the name), coordinate and tear down a communication session between two endpoints - peers. Compared to a traditional telephone, the ringing of a phone, the busy tone and the ending of a call are all functions the SIP protocol is responsible for.

When considering the importance of SIP it is worth to mention that SIP is used extensively in the IP Multimedia Subsystem (IMS) that provides multimedia services in third generation (3G) mobile phones (IP Multimedia Subsystem, 2007). This automatically gives it some wide industry support and ensures that it will be a relevant protocol for at least the next 10 years.

1.3 Real-Time Protocol

The Real-Time Protocol (RTP) is used for the real-time transportation of data. In the case of Voice over IP it is used to transmit voice, but it is useful for the transmit of any data, such as video or some real-time measurement data, to name other examples. Also the RTP protocol is standardised by the IETF, in RFC 3550. (Schulzrinne & al, 2003)

It is worth to note that even if there are many alternatives to SIP when it comes to signaling protocols, RTP is the protocol used for data transmission together with most of them, though not all. RTP is also used in many multimedia streaming solutions, such as the RTSP protocol.

Due to its many usage scenarios and rather low-level nature, the RTP protocol is quite complex and therefore it is outside of the scope of this article to go into further details of its inner workings. For the reader it suffices to think of it as a data transmission protocol.

Combining the SIP and RTP protocols we can then establish a VoIP call. Those readers familiar with the well known File Transfer Protocol (FTP) (Postel, 1985) can see the analogue in having separate command and data transfer channels.

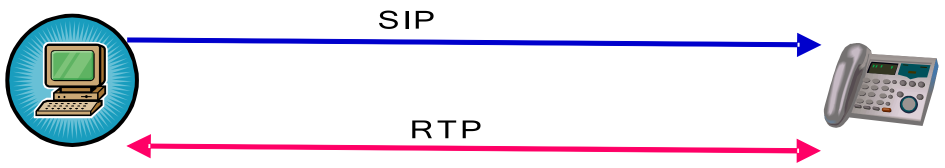


Illustration 1.2: A VoIP call between two peers using the SIP protocol for signaling and RTP for transmission of voice data.

1.4 SIP Proxy

The SIP protocol is designed as a peer-to-peer protocol (as opposed to a client-server architecture). That means that all parties are equal peers and depending of who intitiates some specific action, each party will in turn be both the client and server.

In practice however implementations separate between *User Agents* and *Proxies*. A *SIP Proxy* is comparable to a mail server or web server, whereas the User Agents are the client applications or terminals used by end users. The reasons for this architecture are the same as with Internet mail: An end user device may frequently change its IP address, or the user may use different devices and of course the device may often be turned off, thus leaving nobody to respond to incoming calls. A SIP Proxy therefore provides a constant location which is always on, waiting for your call and ready to route it to the recipient.

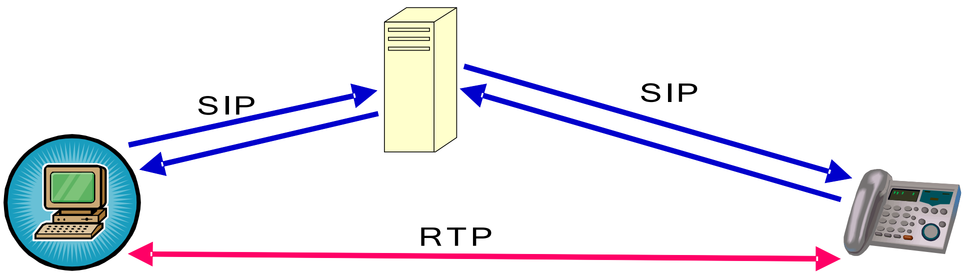


Illustration 1.3: Placing a SIP call through a proxy

Note however, that once the connection has been made using the SIP protocol, the RTP connection is made directly between the end-points, with no intermediate server (unfortunately, there are situations when this is not possible). The reasons for this are efficiency. Routing thousands of voice streams through some centralised server would be a tremendous load for a server to bear, and even in the best case would introduce extra lag between the endpoints. The SIP protocol on the other hand is a leightweight textual protocol and a SIP Proxy can easily service millions of calls per second (IPTel.org, 2007).

While only one SIP Proxy is depicted in Illustration 1.3, there can be two or more proxies mediating the connection.

1.5 The SIP protocol, a basic introduction

The SIP protocol can best be described as a hybrid between two other popular IETF protocols: SMTP (mail) and HTTP (web). A SIP call is obviously made *From* somebody, *To* somebody, just like sending email. On the other hand, where applicable, the authors of SIP has chosen to use similar status codes as in HTTP.

Some messages contain a payload of content type *application/sdp*. SDP is a protocol to exchange parameters related to what content types and codecs can be used for the RTP protocol data. For the purpose of this article, those portions can simply be ignored.

Below is a sequence of a SIP dialogue where a call is successfully established.

A --> B: Dialogue is initiated with INVITE command. Connection is made from A to a SIP Proxy, which then transmits the invitation to B.

```
INVITE sip:atilaaja@sesca.com SIP/2.0
Via: SIP/2.0/UDP 10.0.0.7:34856;branch=z9hG4bK-[...]--d87543-;rport
Max-Forwards: 70
Contact: <sip:btilaaja@10.0.0.7:34856>
To: "btilaaja"<sip:atilaaja@sesca.com>
From: "btilaaja"<sip:btilaaja@sesca.com>;tag=7c32af6e
Call-ID: OTFiNjc0YjMzZDg1Y2Q3ZGViZDIxOTY1OGRmMTg5MzU.
CSeq: 1 INVITE
```

```
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE,
SUBSCRIBE, INFO
Content-Type: application/sdp
User-Agent: X-Lite release 1006e stamp 34025
Content-Length: 269

v=0
o=- 7 2 IN IP4 127.0.0.1
s=CounterPath X-Lite 3.0
c=IN IP4 127.0.0.1
t=0 0
m=audio 26486 RTP/AVP 107 119 0 98 8 3 101
a=fmtp:101 0-15
a=rtpmap:107 BV32/16000
a=rtpmap:119 BV32-FEC/16000
a=rtpmap:98 iLBC/8000
a=rtpmap:101 telephone-event/8000
a=sendrecv
```

A <-- Proxy: Proxy responds with 407, challenging A to authenticate itself. Authentication is done with Digest method, same as in HTTP.

```
SIP/2.0 407 Proxy Authentication Required
Via: SIP/2.0/UDP 10.0.0.7:34856;branch=z9hG4bK-[...]--d87543-
;rport=34856;received=10.1.2.69
To: "btilaaja"<sip:atilaaja@sesca.com>;tag=b63f3[...]2.103d
From: "btilaaja"<sip:btilaaja@sesca.com>;tag=7c32af6e
Call-ID: OTFiNjc0YjMzZDg1Y2Q3ZGViZDIxOTY1OGRmMTg5MzU.
CSeq: 1 INVITE
Proxy-Authenticate: Digest realm="sesca.com",
nonce="45af2e8848e1339238cd8ff791d3091e73e93fb2", qop="auth"
Server: OpenXg OpenSER (1.1.0-pre2-tls (i386/linux))
Content-Length: 0
```

A --> Proxy: A sends acknowledgement to Proxy, that the previous message was received.

```
ACK sip:atilaaja@sesca.com SIP/2.0
Via: SIP/2.0/UDP 10.0.0.7:34856;branch=z9hG4bK-[...]--d87543-;rport
To: "btilaaja"<sip:atilaaja@sesca.com>;tag=b63f[...]6f2.103d
From: "btilaaja"<sip:btilaaja@sesca.com>;tag=7c32af6e
Call-ID: OTFiNjc0YjMzZDg1Y2Q3ZGViZDIxOTY1OGRmMTg5MzU.
CSeq: 1 ACK
Content-Length: 0
```

A --> B: A sends INVITE again, now with Digest authentication data included.

```
INVITE sip:atilaaja@sesca.com SIP/2.0
Via: SIP/2.0/UDP 10.0.0.7:34856;branch=z9hG4bK-[...]--d87543-;rport
Max-Forwards: 70
Contact: <sip:btilaaja@10.0.0.7:34856>
To: "btilaaja"<sip:atilaaja@sesca.com>
From: "btilaaja"<sip:btilaaja@sesca.com>;tag=7c32af6e
Call-ID: OTFiNjc0YjMzZDg1Y2Q3ZGViZDIxOTY1OGRmMTg5MzU.
CSeq: 2 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE,
SUBSCRIBE, INFO
```

```
Content-Type: application/sdp
Proxy-Authorization: Digest
username="btilaaja",realm="sesca.com",nonce="45af2e8848e1339238cd8ff791d309
1e73e93fb2",uri="sip:atilaaja@sesca.com",response="deb2703555e4fe40839a9cd0
8e89dedd",cnonce="761bce0a8ee89a6a",nc=00000001,qop=auth,algorithm=MD5
User-Agent: X-Lite release 1006e stamp 34025
Content-Length: 269

v=0
o=- 7 2 IN IP4 127.0.0.1
s=CounterPath X-Lite 3.0
c=IN IP4 127.0.0.1
t=0 0
m=audio 26486 RTP/AVP 107 119 0 98 8 3 101
a=fmtp:101 0-15
a=rtpmap:107 BV32/16000
a=rtpmap:119 BV32-FEC/16000
a=rtpmap:98 iLBC/8000
a=rtpmap:101 telephone-event/8000
a=sendrecv
```

A <-- Proxy: Proxy sends a provisional response 100, which means that it has received the invite from A and is now trying to forward it to B. (The "your call is important to us" string is typical of OpenSER, the SIP Proxy being used.)

```
SIP/2.0 100 trying -- your call is important to us
Via: SIP/2.0/UDP 10.0.0.7:34856;branch=z9hG4bK-[...]--d87543-
;rport=34856;received=10.1.2.69
To: "btilaaja"<sip:atilaaja@sesca.com>
From: "btilaaja"<sip:btilaaja@sesca.com>;tag=7c32af6e
Call-ID: OTFiNjc0YjMzZDg1Y2Q3ZGViZDIxOTY1OGRmMTg5MzU.
CSeq: 2 INVITE
Server: OpenXg OpenSER (1.1.0-pre2-tls (i386/linux))
Content-Length: 0
```

A <-- B: B has now received the invitation from the Proxy and answers with 180 which means that it is now ringing (as in "the phone is ringing") and waiting for the user to answer the call.

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 10.0.0.7:34856;received=10.1.2.69;branch=z9[...]-43-
;rport=34856
Record-Route: <sip:10.1.2.48;lr;pm;n1>
Contact: <sip:atilaaja@10.1.2.69:1026;rinstance=0ca247682fe6140e>
To: "btilaaja"<sip:atilaaja@sesca.com>;tag=673dc94f
From: "btilaaja"<sip:btilaaja@sesca.com>;tag=7c32af6e
Call-ID: OTFiNjc0YjMzZDg1Y2Q3ZGViZDIxOTY1OGRmMTg5MzU.
CSeq: 2 INVITE
User-Agent: X-Lite release 1006e stamp 34025
Content-Length: 0
```

A <-- B: User B answers (by picking up the phone or pressing a button, etc...) and B sends 200 to A, the OK command.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.0.0.7:34856;received=10.1.2.69;branch=z9[...]43-
;rport=34856
```

```
Record-Route: <sip:10.1.2.48;lr;pm;n1>
Contact: <sip:atilaaja@10.1.2.69:1026;rinstance=0ca247682fe6140e>
To: "btilaaja"<sip:atilaaja@sesca.com>;tag=673dc94f
From: "btilaaja"<sip:btilaaja@sesca.com>;tag=7c32af6e
Call-ID: OTFiNjc0YjMzZDg1Y2Q3ZGViZDIxOTY1OGRmMTg5MzU.
CSeq: 2 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE,
SUBSCRIBE, INFO
Content-Type: application/sdp
User-Agent: X-Lite release 1006e stamp 34025
Content-Length: 274

v=0
o=- 7 2 IN IP4 127.0.0.1
s=CounterPath X-Lite 3.0
c=IN IP4 10.1.2.48
t=0 0
m=audio 60394 RTP/AVP 107 119 0 98 8 3 101
a=fmtp:101 0-15
a=rtpmap:107 BV32/16000
a=rtpmap:119 BV32-FEC/16000
a=rtpmap:98 iLBC/8000
a=rtpmap:101 telephone-event/8000
a=sendrecv
```

A --> B: A sends ACK command to B as a sign that the OK command was received. After this message A and B start RTP traffic based on the parameters that have been exchanged, the call is in progress.

```
ACK sip:atilaaja@10.1.2.69:1026;rinstance=0ca247682fe6140e SIP/2.0
Via: SIP/2.0/UDP 10.0.0.7:34856;branch=z9hG4bK-[...]-d87543-;rport
Max-Forwards: 70
Route: <sip:10.1.2.48;lr;pm;n1>
Contact: <sip:btilaaja@10.0.0.7:34856>
To: "btilaaja"<sip:atilaaja@sesca.com>;tag=673dc94f
From: "btilaaja"<sip:btilaaja@sesca.com>;tag=7c32af6e
Call-ID: OTFiNjc0YjMzZDg1Y2Q3ZGViZDIxOTY1OGRmMTg5MzU.
CSeq: 2 ACK
Proxy-Authorization: Digest
username="btilaaja",realm="sesca.com",nonce="45af2e8848e1339238cd8ff791d309
1e73e93fb2",uri="sip:atilaaja@sesca.com",response="deb2703555e4fe40839a9cd0
8e89dedd",cnonce="761bce0a8ee89a6a",nc=00000001,qop=auth,algorithm=MD5
User-Agent: X-Lite release 1006e stamp 34025
Content-Length: 0
```

The "Allow:" header above actually reveals the most common SIP commands. One not used in the above dialogue is MESSAGE, which can be used to send short textual messages. In fact SIP can be used as an instant messaging protocol too. (In that case, no RTP is needed.)

1.6 Problems with SIP

A reader familiar with Internet technologies should by now already have spotted some of the obvious weaknesses of SIP:

1.Firewalls. Due to firewalls, it is often not possible for two endpoints to receive incoming RTP traffic. The behaviour of firewalls varies, so the remedies to overcome this problem are also multitude.

Often firewalls will actually allow incoming UDP packets (RTP is over UDP in the Internet protocol stack) from an IP address if the endpoint behind the firewall has sent UDP packets towards that address first. In those cases there is no problem, some of the first RTP packets might get lost but once both parties are sending RTP traffic they will also be able to receive traffic.

In other cases a firewall may really block traffic, in which case some kind of middle server (which could be called RTP Proxy) is needed to route the traffic between the two endpoints.

2. NAT (Native Address Translation). As can be seen in the above SIP dialogue, the SIP commands contain literal IP addresses in several places. Since those addresses are used for the RTP communication, the connection will fail if either endpoint is behind a NAT (Native Address Translation). This too is a serious problem in today's Internet, and a severe limitation to SIP.

Various techniques have been proposed to overcome the problems introduced by NAT. These are known as STUN, ICE and TURN (STUN, 2007; ICE, 2007; TURN, 2007). The general idea is for the SIP User Agent to try to deduce its public IP address and use that in the SIP messages. On the other hand, the SIP Proxy can also contain logic which compares the IP address given in the SIP message to the one the message is actually seen to originate from and mangle the SIP message with another IP address when needed.

3. Cryptography. Both SIP and RTP provide for secure connections, but most SIP User Agents still today do not support cryptography of any kind or at least not well.

In the case of SIP the problem is actually twofold: Securing the communication between a User Agent and the nearest SIP Proxy and between proxies on the one hand and securing the connection between the two endpoints on the other hand. The first case is solved by using TLS (Transient Layer Security), just as in the https protocol.

The latter is more complicated: Part of the information in the SIP messages must be available to the intermediate proxies anyway, after which there is not much left to encrypt. Cryptographical algorithms could be used to authenticate the caller and callee to each other. In this case we would need some Public Key Infrastructure (PKI) to exchange keys between all potential users and thus run into all the same reasons while email too is still most often sent unencrypted. So even if the SIP standard defines a way to send messages encrypted (S/MIME), it is seldom used in real life.

The RTP protocol also defines SRTP, secure RTP. Even if adding cryptography to an already complicated standard is not easy to digest, SRTP would actually be straightforward to implement but for one problem: There are multiple approaches to exchanging the encryption keys needed (as part of the SIP invitation dialogue of course) and different client applications are not compatible with each other.

While there are SIP solutions available that support encrypted SIP and RTP communication, typically they are not yet fully compatible between vendors.

4. Complexity. As a final drawback of SIP we'll mention here its complexity. While the basic mechanism of INVITEs and responses is straightforward, actual SIP dialogues tend to be rather difficult to get right. In addition to the RFC 3261 which defines the current SIP standard there are over 30 other RFC's that extend the base RFC. In practice different vendors still support different SIP "dialects", ie even though two implementations might be said to support SIP they are not necessary compatible with each other beyond the basic features of establishing a call.

One, though not by far the only one, reason to this is the desire to make SIP support all features of the traditional telephone network. (Indeed, using a *media gateway* one can call a phone on the traditional telephone network from a SIP telephone.) Because of this, there are commands in SIP that would not normally be needed in a pure IP world. For instance, when calling a traditional phone, the caller might be received with a female voice saying that *"The number you have called is currently unavailable"* - in other words, some error announcement. Since this is not a successfull call, it will also not be billed as a call. It must therefore be distinct from successfully connected calls. In SIP this means that there is no 200 OK command. But that presents a problem, since the SIP User Agent cannot start listening to RTP packets before the 200 OK, which contains needed parameters for RTP, and thus has no possibility to replay the announcement to the user. For this reason, the RFC 3960 (Camarillo, 2004) defines how to handle these announcements and other situations known as *early media*.

1.7 Further reading

To further learn about the SIP protocol the author of this article recommends books by Alan Johnston - one of the authors of the SIP protocol - in particular the title co-authored with Henry Sinnreich: "Internet Communications Using SIP: Delivering VoIP and Multimedia Services with Session Initiation Protocol" (Sinnreich & Johnston, 2006).

A popular introduction to the RTP protocol is the book "RTP: Audio and Video for the Internet" by Colin Perkins (Perkins, 2003).

Also Wikipedia (Wikipedia, 2007) has excellent articles on all Internet protocols presented here.

The NAT and Firewall problems presented are common to all peer-to-peer protocols. The IETF has a special "MIDCOM" working group which is chartered to find solutions to these problems. It has produced two documents that well highlight the problems and terminology around this issue: RFC's 3303 (Srisuresh, 2002) and 3304 (Swale, 3304).

2 ALTERNATIVES TO SIP

We will now briefly enumerate some alternative VoIP protocols and how they compare with SIP.

2.1 H.323

H.323 is another VoIP signaling protocol standardised by the ITU-T (International Telecommunications Union). This is the same body that standardises telephone network behavior and indeed, the H.323 is similar to its counterparts on that network. It could be briefly described as a kind of "ISDN over packet networks". (H.323, 2007)

H.323 was widely used in the nineties and Internet users will perhaps be familiar with such applications as Microsoft NetMeeting or GnomeMeeting, which both used this protocol. It too uses RTP for data transmission.

While H.323 provides roughly the same kind of functionality as SIP, it is older and can today be considered nearly obsolete, as the Internet community has come to prefer "its own" IETF standard (SIP) instead of this one.

2.2 Jabber, also known as XMPP

Jabber is an XML (extensible markup language) based protocol. It was originally developed in 1998 by Jeremie Miller, who started the Jabber Free Software project (Jabber, 2007). Originally its scope was only instant messaging, but as an extensible protocol it has also come to support Voice over IP. Currently Jabber too is standardised within an IETF working group and is known as the Extensible Messaging and Presence Protocol, XMPP (XMPP, 2007). The relevant RFC's for XMPP are 3920 (Saint-Andre, 2004a) and 3921 (Saint-Andre 2004b).

The VoIP extension to XMPP is known as Jingle and was developed by Google (Jingle, 2007). It too uses RTP for data transmission and uses the ICE method to overcome NAT problems.

While XMPP has long had a reputation of being a good and well designed protocol, it was the adoption of XMPP by Google for use in its Google Talk service that has lended the protocol serious credibility. XMPP is currently the main challenger to SIP. Even if SIP has all but been accepted as "the" standard to be used for VoIP and is somewhat set in stone for the use in current GSM standards, it is not out of the question that somebody could develop a better VoIP and instant messaging protocol. At least it is easy to argue that there have been quite many architects to the dozens of SIP RFC documents and perhaps one reason to its complexity is its "design by committee" nature. Perhaps XMPP, developed first by a small Free Software project and then adopted by a company with resources to put it into good use, will indeed have a good chance of being the VoIP protocol of the future.

2.3 Inter-Asterisk Exchange Protocol by Asterisk (IAX2)

Asterisk is a VoIP server that is covered separately in the next section. While Asterisk supports several VoIP protcols, Digium, the company behind the Asterisk project, has developed its own Inter-Asterisk Exchange Protocol, IAX. It was first used simply between Asterisk servers, but has later been also implemented in some VoIP terminals and software independent of the Asterisk project. (IAX, 2007)

An interesting property of the IAX protocol is that it combines both signaling and voice data into the same data stream, transmitted over UDP to port 4569. Several communications can be *multiplexed* into this data stream. This overcomes most of the problems NAT and firewalls pose to the other protocols.

2.4 Skinny Client Control Protocol by Cisco (SCCP)

The Skinny Client Control Protocol (SCCP) is a protocol used by Cisco in its CallManager server and associated phone terminals. It is a proprietary protocol so public information is scarce. But interestingly Asterisk supports this protocol too through a reverse engineered Free Software module *chan-sccp*. [http://sourceforge.net/projects/chan-sccp/]

3 SERVER APPLICATIONS

As with other Internet technologies and IETF standards, Free Software has been at the forefront of innovation with the SIP protocol.

3.1 Asterisk

The best known VoIP related Free Software server is Asterisk, started by Mark Spencer. [http://asterisk.org/] Even though it is nowadays mostly used as a VoIP solution, it is still also capable of providing traditional analog or digital telephony services and should make an interesting project for anyone interested in traditional telecommunications. The easiest way to try out Asterisk's traditional telecom features is with a computer equipped with a modem (supported by Linux) or an ISDN adapter, but Digium also sells hardware to enable Asterisk to become a true PBX system (Private Branch Exchange) serving tens or hundreds of telephones.

Asterisk supports SIP as one protocol to connect with, and out of the other protocols mentioned in this article both IAX and SCCP are also supported. Asterisk uses a concept of channels, where different terminals can connect with different VoIP protocols or traditional telephony interfaces, but administering the different channels is - as far as possible - independent of the channel type used. Therefore Asterisk is not really a great way to learn about SIP, even though it supports SIP well, since its inner architecture does not reflect the philosophy of SIP.

Due to the architecture of Asterisk it is also straightforward to use it in a configuration where all RTP traffic too is routed through the server, contrary to how SIP is intended to be used. This is one way of overcoming firewall and NAT problems but Asterisk is also sometimes used as an intermediary between two otherwise incompatible SIP terminals. It can serve as a media gateway translating from one codec to another, or simply help in connecting two devices using incompatible SIP dialects.

Asterisk can also easily be used as a media gateway, connecting a SIP client to a phone on the traditional telephone network. This is in fact a very valuable and often used function of Asterisk. In this way Asterisk can also be used in tandem with OpenSER (see below), letting OpenSER be the SIP Proxy and Asterisk function as a media gateway or also an application server (such as voice mail).

Asterisk is usually available in most Linux distributions today. A custom Linux distribution called AsteriskNow has also been produced by the Asterisk project. If one wants to build a dedicated Asterisk server it is probably the easiest way to go. Asterisk from version 1.4. comes with a comparably easy graphical interface, putting Asterisk competing with the plethora of commercial IP-PBX offerings available.

3.2 OpenSER

OpenSER is a fork of the original SIP Express Router known as SER. SER was developed initially by IPTel.org, whose members all seem to have some connection with the *Fraunhofer Gesellschaft* (most famous as the place where the mp3 format was created).

Both projects are still active but this author recommends focusing on OpenSER. While the SER homepage is still current and actually recently got a facelift, the current version of SER (0.9.6) is now over one year old.

The OpenSER project was started by active SER developers who did not belong to the inner circle of Fraunhofer developers and felt that the project was not moving rapidly enough. (The final catalyst for the fork was the inability of the SER project to produce a stable release with support for secure connection over TLS, a feature that was dangling in the development version for 6 months.) While Asterisk is more geared towards use as an office PBX, OpenSER is heavily geared towards use by telecom companies or VoIP service providers. It provides broad functionality in logging and accounting, can serve millions of calls per second and supports high availability and load balancing setups. [http://www.openser.org/]

OpenSER is a SIP Proxy and as such is not supposed to deal with RTP, codecs and such issues at all. But OpenSER does have a loadable modules system, and in fact there are also modules to route RTP or mangle IP numbers inside SIP to overcome NAT problems. Also many crucial OpenSER functions are implemented as modules. In this way the architecture is similar to the Apache web server.

Sip Express Media Server, SEMS is often used together with OpenSER to provide voicemail, conferencing and media gateway functionality. [http://developer.berlios.de/projects/sems/]

3.3 SIP foundry SipX

There are many Free Software SIP frameworks and libraries available and it is not possible to mention them all here. SIP stacks in particular seem to be a popular idea for a Free Software project, RTP libraries there are only a few available.

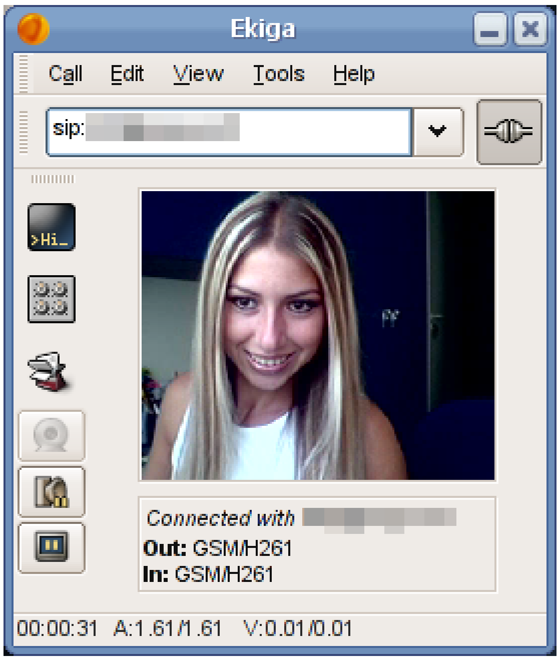
We will however mention one more Free Software project, the SIP foundry, which produces a SIP PBX known as SipX. The SIP foundry also hosts several libraries related to SIP, including reSIProcate, originally sponsored by Cisco and SipXTapi. Both of these alternative libraries are used in some proprietary SIP softphones, for instance SipXTapi by Yahoo and reSIProcate by Counterpath eyeBeam. In addition to its PBX offering, SIP foundry also publishes its own softphone, SipXezPhone. [http://www.sipfoundry.org/]

4 CLIENT APPLICATIONS

We will now finally briefly mention some of the most popular Free Software VoIP client applications. All of these are included in modern Linux distributions and are therefore easy to install and try out - you will of course need a SIP account, the homepages of Ekiga and Wengo will help you register one.

4.1 Ekiga (former GnomeMeeting)

The first widely used Free Software VoIP application available for Linux was GnomeMeeting, the lead developer is Damien Sandras. GnomeMeeting supported the H.323 protocol and was thus compatible with Microsofts NetMeeting. As both H.323 and NetMeeting started to fade into obsolesence, GnomeMeeting



Connected with
Out: GSM/H261
In: GSM/H261

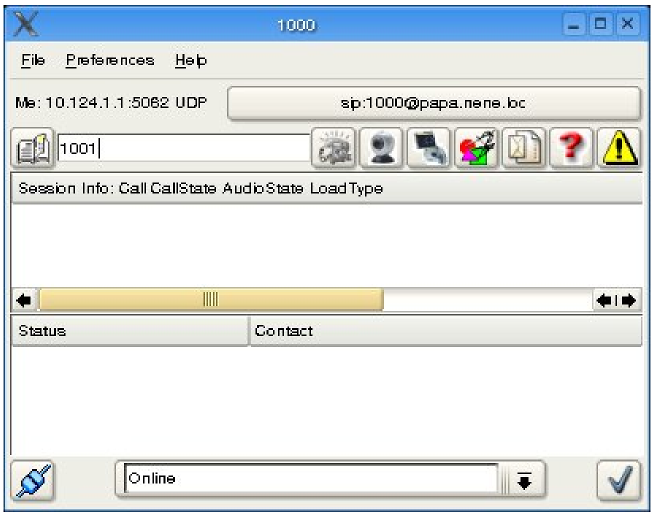
00:00:31 A:1.61/1.61 V:0.01/0.01

Illustration 4.1: Ekiga in a SIP video call. Lead developer Damien Sandras has called his beatiful wife. (Screenshot courtesy of ekiga.org) faced the risk of becoming obsolete, but today it has reinvented itself and is now known as Ekiga and supports also the SIP protocol. [http://ekiga.org/]

Ekiga is based on the GTK framework and only an experimental port for Windows exists. In the original GnomeMeeting days this was not such a problem, since its scope was more to be a Linux counterpart to NetMeeting. Compatibility between these two was fairly good, the only problem was that most codecs used by Microsoft could not be used in GnomeMeeting due to the software patent situation. The solution was to install a free codec on the NetMeeting users PC. Ekiga is still the dominant SIP client on Gnome.

4.2 KPhone

As the name suggests, KPhone is a SIP client application for the KDE desktop. It has gone through the hands of several developers, first started by Billy Biggs, then developed by some researchers in Western Finland and now by a team of volunteers. One current lead developer Jan Janak is also a developer of SER.


1000

Me: 10.124.1.1:5062 UDP

sip:1000@papa.nene.bc

1001

Session Info: Call CallState AudioState LoadType

Status Contact

Online

Illustration 4.2: Kphone 1.0.2.1 (Screenshot courtesy of KPhone project at SourceForge)

All current and past KPhone developers have been some of the worlds best SIP experts and KPhone has been this authors favorite application to learn and try out SIP features, since technical details are well exposed in the user interface. KPhone has some weaknesses, for a long time it did not support video conferencing and also its user interface has traditionally perhaps been more technical than many others.

4.3 OpenWengo

OpenWengo is a relatively new contender to the Free Software VoIP market. It is sponsored by the French VoIP provider Wengo, but the 2.0 release has already seen contributions from external developers too. The 2.0 release was still closely tied to the Wengo VoIP service, but making OpenWengo provider agnostic is a high priority goal for the developers. And of course it is already possible to support other providers by altering the source code. [http://openwengo.org/]

OpenWengo is developed with the Qt toolkit and Microsoft Windows has always been the primary target environment although Linux and Mac OS are equally supported too. The user interface is very professional and has broken away from the traditional "software phone" paradigm to emulate common instant messaging applications (and Skype) instead. OpenWengo was also the first Free Software project to support HTTP tunneling to bypass firewalls that block SIP or RTP traffic. To this authors knowledge OpenWengo was also the source for a favorite VoIP innovation: audio smileys.

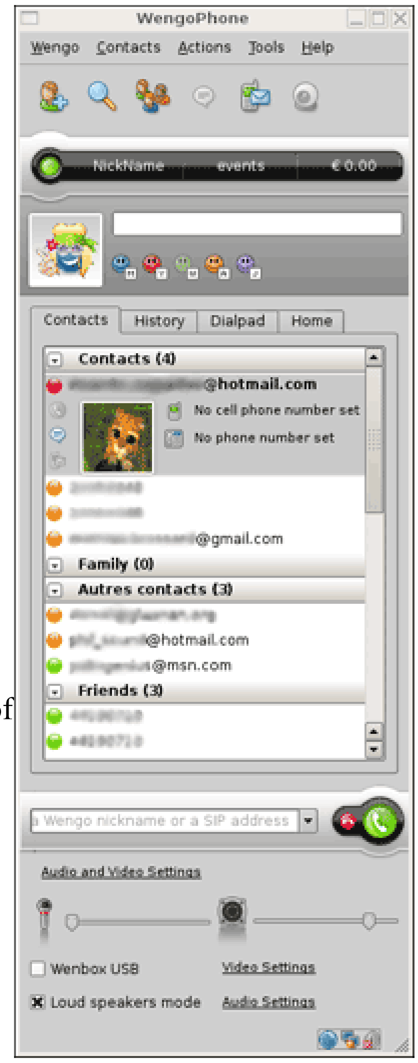


Illustration 4.3: OpenWengo 2.0 main window. (Image courtesy of OpenWengo.org)

4.4 Minisip

Minisip was developed primarily by Erik Eliason of Kungliga Tekniska Högskolan, Sweden. It is a SIP phone based on GTK that works on both Linux and Windows and both PCs and embedded platforms. [http://minisip.org/]

Although it is a capable SIP softphone, minisip is perhaps not as well known as the others mentioned here. However, it is worth mentioning because of Erik's focus on using the project to showcase the cryptographically secured versions of SIP and RTP. Readers interested in security and cryptography in VoIP are recommended to take a look at this project.

5 SKYPE

Although it has nothing to do with Free Software nor SIP, it is prudent to also mention Skype in this article. After all, Skype is currently the most popular VoIP application.We can comfort ourselves in that while being closed source, at least it is available for other operating systems than Windows.

Illustration 5.1: Skype is currently the most popular VoIP application.

It is no accident that Skype has become popular so fast. Skype has actually solved all of the problems that are seen as weaknesses of SIP clients: It is usually able to function from behind any firewall or NAT. (As the founders Niklas Zennström and Janus Friis are also creators of the peer-to-peer filesharing application KaZaa, they were able to utilise many lessons learned from building peer-to-peer applications.) Apart from Microsoft Messenger, Skype was one of the first applications to abandon the paradigm of emulating a traditional phone and instead opted a user interface similar to instant messanging applications. And finally, Skype provides for completely encrypted communication. By any comparison, Skype is technically superior to its competition today.

It is however hard to see that eventually all of the Internet would end up using a VoIP solution that is proprietary and closed and only available from one vendor. It would in that case be the first time something like that happened with an Internet communications protocol. While Skype is technically superior to SIP or other competitors now, it must therefore only be a question of time before it is superseded by an open protocol, be it SIP, XMPP or something else.

BIBLIOGRAPHY

Camarillo, G (2004): Early Media and Ringing Tone Generation in the Session Initiation Protocol (SIP). Internet Engineering Task Force, 2004. [http://www.ietf.org/rfc/rfc3960.txt]

H.323 (2007): H.323. Wikipedia: the free encyclopedia that anyone can edit. English edition. Wikimedia Foundation, 2007. [http://en.wikipedia.org/wiki/H.323]

IAX (2007): IAX. Wikipedia: the free encyclopedia that anyone can edit. English edition. Wikimedia Foundation, 2007. [http://en.wikipedia.org/wiki/IAX]

ICE (2007): Interactive Connectivity Establishment. Wikipedia: the free encyclopedia that anyone can edit. English edition. Wikimedia Foundation, 2007. [http://en.wikipedia.org/wiki/Interactive_Connectivity_Establishment]

IP Multimedia Subsystem (2007): IP Multimedia Subsystem. Wikipedia: the free encyclopedia that anyone can edit. English edition. Wikimedia Foundation, 2007. [http://en.wikipedia.org/wiki/IP_Multimedia_Subsystem]

IPTel.org (2007): SIP Express Router (www homepage). IPTel.org, 2007. [http://www.iptel.org/ser/]

Jabber (2007): Jabber. Wikipedia: the free encyclopedia that anyone can edit. English edition. Wikimedia Foundation, 2007. [http://en.wikipedia.org/wiki/Jabber]

Jingle (2007): Jingle (protocol). Wikipedia: the free encyclopedia that anyone can edit. English edition. Wikimedia Foundation, 2007. [http://en.wikipedia.org/wiki/Jingle_%28protocol%29]

Perkins, Colin (2003): RTP: Audio and Video for the Internet. Addision Wesley Professional, 2003.

Postel, J. (1985): FILE TRANSFER PROTOCOL (FTP), (RFC 959). Internet Engineering Task Force, 1985. [http://www.wu-ftpd.org/rfc/rfc959.html]

Rosenberg, J. & al (2002): SIP: Session Initation Protocol (RFC 3261). Internet Engineering Task Force, 2002. [http://www.ietf.org/rfc/rfc3261.txt]

Saint-Andre, P (ed) (2004a): Extensible Messaging and Presence Protocol (XMPP): Core (RFC 3920). Internet Engineering Task Force, 2004. [http://www.ietf.org/rfc/rfc3920.txt]

Saint-Andre, P (ed) (2004b): Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence (RFC 3921). Internet Engineering Task Force, 2004. [http://www.ietf.org/rfc/rfc3921.txt]

Schulzrinne, H. & al (2003): RTP: A Transport Protocol for Real-Time Applications (RFC 3550). Internet Engineering Task Force, 2003. [http://www.ietf.org/rfc/rfc3550.txt]

Sinnreich, Henry & Johnston, Alan (2006): Internet Communications Using SIP: Delivering VoIP and Multimedia Services with Session Initiation Protocol. 2nd edition. Wiley, 2006.

Srisuresh, P & al (2002): Middlebox communication architecture and framework (RFC 3303). Internet Engineering Task Force, 2002. [http://www.ietf.org/rfc/rfc3303.txt]

STUN (2007): STUN. Wikipedia: the free encyclopedia that anyone can edit. English edition. Wikimedia Foundation, 2007. [http://en.wikipedia.org/wiki/STUN]

Swale, R.P. & al (2002): Middlebox Communications (midcom) Protocol Requirements (RFC 3304). Internet Engineering Task Force, 2002. [http://www.ietf.org/rfc/rfc3304.txt]

TURN (2007): Traversal Using Relay NAT. Wikipedia: the free encyclopedia that anyone can edit. English edition. Wikimedia Foundation, 2007. [http://en.wikipedia.org/wiki/TURN]

Wikipedia (2007): Wikipedia: the free encyclopedia that anyone can edit. English edition. Wikimedia Foundation, 2007. [http://en.wikipedia.org/]

XMPP (2007): Extensible Messaging and Presence Protocol. Wikipedia: the free encyclopedia that anyone can edit. English edition. Wikimedia Foundation, 2007. [http://en.wikipedia.org/wiki/Extensible_Messaging_and_Presence_Protocol]

--

Henrik Ingo is the author of the book "Open Life: The Philosophy of Open Source". While working for the Finnish training provider Tieturi he taught a multitude of Internet and Open Source related programming languages as well as speaking in conferences and seminars. After working as an e-learning instructor at Tieturi he has also researched e-learning at Helsinki University of Technology. Currently Henrik is working for the software company Sesca Technologies as Senior Manager of Capital Area where he is involved in Voice over IP, Linux and Symbian projects, among others. (henrik.ingo@openlife.cc)