# IETF Protocols for IP Telephony


| Henning Schulzrinne | Jonathan Rosenberg |
|---|---|
| Dept. of Computer Science | Bell Laboratories |
| Columbia University | Lucent Technologies |
| hgs@cs.columbia.edu | jdrosen@bell-labs.com |


December 9, 1998


**Abstract**

The IETF is standardizing a range of protocols necessary to support Internet telephony. We review the Real Time Transport Protocol (RTP), the Session Initiation Protocol (SIP), the Session Description Protocol (SDP), the Session Announcment Protocol (SAP), the Real Time Streaming Protocol (RTSP), the Call Processing Language, and the Gateway Location Protocol.


Imagine this as a possible vision for the not too distant future. You are sitting at your computer, and all of a sudden your machine emits a "boing", followed by speaking "Audio and video call from Joe". It's the call you've been waiting for. You accept the call, and talk for a while. You then decide that Alice needs to be in on the call, too. From your PC, you speak "Add Alice". Your client software consults a local Internet "white pages" directory and adds ''alice@ieee.org" to the call. Your invitation reaches Alice's personal agent software. The agent tries to "ring" her cell phone, home PC and work PC, all in parallel. Alice picks up in her car, joining the call voice-only. During the conversation, Joe remembers that a segment from a recent IEEE tutorial presentation would be helpful. He finds the web page and adds the media stream to the conversation. Later, you decide to leave the call, and transfer them to a web page containing the information you had been discussing. Joe's web browser jumps to the page, and Alice's phone displays a text-only version, which they continue to discuss. Your next call is to Bob, returning an earlier call. Unfortunately, he is out of the office. But he has programmed his call answering software to return a web page with his appointments and a hyperlink to a voice mail service to you. (He only does that for people he has talked to before.) You click on the link to record a brief message; the voice mail server forwards an email message to Bob. He can retrieves the message using any Internet-connected PC using a standard audio/video player. Your final call for the day goes to Switzerland, to a "legacy" phone number. You want to avoid the high international charges, so you have your software locate a "hop-off" point from the Internet into the Swiss telephone system that minimizes cost while assuring that the voice quality is acceptable, as rated by a Consumer Reports.

This scenario illustrates many of the facets of *Internet telephony.* Originally, Internet telephony was defined to mean just the transport of voice over IP networks between two PCs. Now, its scope has expanded to encompass interactive multimedia communications, incorporating both traditional circuit switched and wireless networks and Internet applications, including web and email. The initial motivation for Internet telephony was "bypass", using the Internet to avoid the per-minute charges of monopoly telecom carriers. While this motivation remains, the emphasis has shifted to providing novel services in an environment that allows end users to program the behavior of telephone-like systems.

Accomplishing such a goal will require a range of technologies, including algorithms, architectures, and protocols. In this article, we focus on the latter, and give an overview of activities within the Internet Engineering Task Force (IETF) aimed at making Internet telephony a reality.

# 1 IETF Working Groups

The IETF is currently divided broadly into eight areas: Internet, routing, transport, security, applications, operations and management, user services, and general (policy). About 115 working groups are spread among these areas, with each working group chartered to develop specific protocols. Most of the working groups related to IP telephony are in the transport area. The related working groups are:

**Audio/Video Transport (avt):** The AVT working group has defined the Real Time Transport Protocol (RTP) [1], used for voice and video transport over the Internet. It has also developed a number of payload formats (means for carrying specific media formats over RTP). We discuss RTP further in section 2.

**Multiparty Multimedia Session Control (mmusic):** The MMUSIC working group has defined a number of protocols related to IP telephony. The first is the Session Initiation Protocol (SIP) [2], used for initiation of multimedia sessions. We discuss SIP in section 3. The second is the Real Time Streaming Protocol (RTSP) [3], used for controlling media servers, which we discuss in section 6. The third is the Session Announcement Protocol (SAP) [4], which provides a "TV Guide" like functionality for the Internet, discussed in section 5. The last is the Session Description Protocol (SDP) [5], which describes multimedia sessions, which we discuss in section 4. The relationship of these protocols is also described by Handley et al. [6].

**IP Telephony (iptel):** The IPTEL working group is developing two protocols directly related to IP telephony. The first is the Gateway Location Protocol (GLP) [7], which describes a protocol for locating IP telephony gateways (section 8). The second is a Call Processing Language (CPL) [8], for allowing end users to define call processing services (section 7).

**PSTN[1]/Internet Internetworking (pint)** The PINT working group is defining a protocol for allowing IP hosts, usually web servers, to control some basic services on the telephone network. The protocol is an extension of SIP.

**Differentiated Services (diffserv):** The Diffserv group is defining a mechanism for providing quality of service (QoS) in the Internet that scales to large numbers of users [9]. Although diffserv is not limited to a particular application, the anticipated large number of Internet telephony calls and the expectations of "telephone quality" makes Internet telephony a likely user of this technology.

In the sections which follow, we briefly discuss the function of these protocols and how they fit into the broader picture of Internet telephony.

# 2 Real Time Transport Protocol (RTP)

Data transport is at the heart of Internet telephony. Multimedia Transport refers to the ability to carry voice and video across IP networks, and at the far end, reconstruct the original media stream. Work on transport of voice began in the 1970's, with the development of the Network Voice Protocol (NVP) [10] and later NVP-II [11]. More recently, the Audio Video Transport working group (AVT) has developed the Real Time Transport Protocol (RTP) [1]. RTP can be used for Internet telephony, media distribution and media-on-demand services. It has also been adopted for the ITU conferencing architecture [12].

RTP actually consists of two components, RTP itself, and the Real Time Control Protocol (RTCP). Both RTP and RTCP run on top of UDP. RTP provides services required by most streaming multimedia applications:

**Sequencing:** An IP network may drop or reorder packets. In order to reconstruct the media stream at the remote end, these events must be detected. RTP provides sequence numbers for this purpose.

**Intra-media synchronization:** Packets within the same stream may suffer different delays (jitter). Applications use playout buffers [13, 14, 15] to compensate for delay jitter. They need the timestamps provided by RTP to measure it.

**Payload identification:** In the Internet, it is often necessary to change the encoding for the media (the "payload" of RTP) on the fly to adjust to changing bandwidth availability or the capabilities of new users joining a group. RTP provides a payload type identifier to describe the encoding of the media.

**Frame indication:** Video and audio are sent in logical units called frames. It is necessary to indicate to a receiver where the beginning or end of a frame is, in order to aid in synchronized delivery to higher layers. A frame marker bit is provided for this purpose.

Exchange of media among a group of RTP users is referred to as a *session*. RTP is engineered for unicast and multicast; it scales from small, two person sessions to symmetric conferences to large, broadcast sessions with thousands of receivers and a single sender.

A user wishing to send media in a session takes that media, breaks it up into media-appropriate pieces (such as a video frame or a block of audio), appends a codec-specific header (defined by a payload format) to convey codec-specific framing and features, prepends the RTP header, and then places the result in a UDP packet. RTP defines the syntax and some basic semantics for this header. Detailed semantics for various simple codecs are defined in a companion profile, called the *RTP Profile for Audio and Video Conferences with Minimal Control* [16]. For more complex codecs, specific payload formats are defined. These include formats for H.261 [17], H.263 [18], and MPEG1/2 [19].

RTP also has a companion control protocol, called the Real Time Control Protocol (RTCP). RTCP packets are sent by both session senders and listeners. They are sent to the same address (multicast or unicast) as the RTP data, but on a different port. RTCP packets are used for several purposes:

**QoS feedback:** Receivers in a session use RTCP to report back the quality of their reception. This information includes the number of lost packets, jitter, and round trip delays. This information can be used by senders for adaptive applications [20, 21, 22] which adjust encoding rates and other parameters based on feedback.

**Inter-media synchronization:** For flexibility, audio and video are often carried in separate packet streams, but they need to be synchronized at the receiver to provide "lip-sync". The necessary information for the synchronization of sources, even if originating from different servers, is provided by RTCP.

**Identification:** RTCP packets contain information such as the email address, phone number, and full name of the participant. This allows session members to know who else is in the session.

**Session Control:** RTCP allows participants to indicate that they are leaving a session (through a BYE RTCP packet). Participants can also send small notes to each other, such as "muting".

Since the RTCP packets contain timely information such as QoS feedback, it is desirable to send them periodically. However, the period of such transmissions cannot be fixed, since this would cause network congestion in large groups. Therefore, RTCP has a number of distributed algorithms built in which allow the period to be adjusted based on the number of group participants [23].

## 3 Session Initiation Protocol (SIP)

RTP provides multimedia transport between participants in a session. However, a mechanism is needed to invite participants to the session in the first place. This initiation requires contacting the desired participant at their current location (i.e., where they are logged on right now), and delivering to them a description of the session. This process can be complex. In SIP, users are identified by an email-like [24] SIP URL (such as `sip:alice@ieee.org`. Calls to this address must traverse servers on the network, much like email traverses message transfer agents (MTA's) [25], eventually arriving at the current location for this user. This function is provided by the Session Initiation Protocol (SIP).

SIP is a client-server protocol, similar in both syntax and semantics to HTTP [26]. Requests are generated by one entity (the client), and sent to a receiving entity (the server) which processes them, and then sends responses. A request and the responses which follow it are called a *transaction*. As a SIP call participant may either generate or receive requests, SIP-enabled end systems include a protocol client and server (called a user agent client and user agent server, respectively). SIP requests may traverse many *proxy servers*, each of which receives a request and forwards it towards one or more next hop servers, which may be another proxy server or the final user agent server. As a result, proxy servers are primarily responsible for *call routing*. A server can make use of any means at its disposal to determine where to route a call. This can be done through database queries, using the LDAP for example, or with finger [27], whois++ [28], or a local database.

A server may also act as a *redirect server*, informing the client of the address of the next-hop server, so that the client can contact it directly. There is no protocol distinction between a proxy server, redirect server, and user agent server; a client or proxy server has no way of knowing which it is communicating with. The distinction lies only in function. A proxy or redirect server cannot accept or reject a request, whereas a user agent server can. This is similar to the Hypertext Transfer Protocol (HTTP) model of clients, origin and proxy servers. A single host may well act as client and server for the same request.

As in HTTP, the client requests invoke *methods* on the server. Requests and responses are textual, and contain header fields which convey call properties and service information. SIP reuses many of the header fields used in HTTP, such as the entity headers (e.g., Content-type) and authentication headers.

SIP defines several methods. INVITE invites a user to a call. BYE terminates a connection between two users in a call. OPTIONS solicits information about capabilities, but does not set up a call. ACK is used for reliable message exchanges for invitations. CANCEL terminates a search for a user. Finally, REGISTER conveys information about a user's location to a SIP server.

A client sets up a call by issuing an INVITE request. This request contains header fields used to convey information about the call. Such fields include To and From, which contain the callee's and

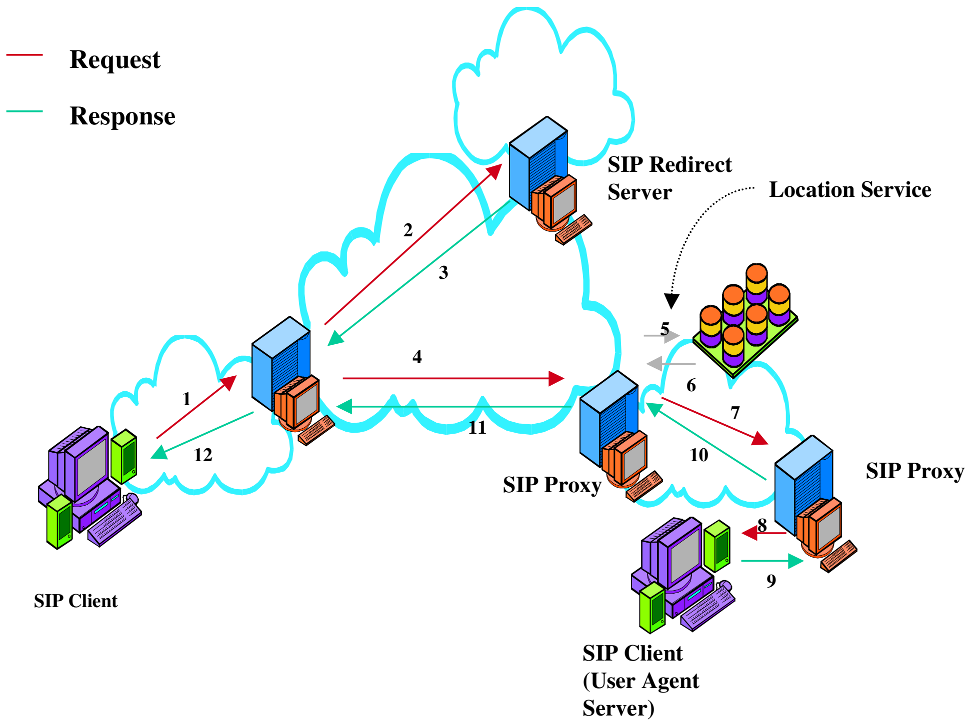


Figure 1: SIP Operation

caller's address, Subject, which identifies the subject of the call, Call-ID which contains a unique call identifier, Contact which lists addresses where a user can be contacted, and Require, which allows for feature negotiation. The call control extensions [29] define an additional header, called Also, which allows for simple third-party call control. This can enable services like transfer and multi-party conferencing [30, 31].

Once the call is set up, either party may send a new INVITE request to the other to change parameters of the call. This includes changing media formats, adding parties to the call, or changing addresses (for mobility support). Once the call is over, either side may hang up by sending a BYE request to the other.

Like HTTP, SIP requests and responses carry a body, which can be any defined MIME [32] type. The body conveys information about the session between the parties. Normally, the Session Description Protocol (SDP) [5] is used. SDP describes multimedia sessions, including codec types, port numbers, addresses, and session start and stop times, all needed to establish an RTP session.

SIP runs on top of either UDP or TCP, providing its own reliability mechanisms when used with UDP. For addressing, SIP makes use of uniform resource identifiers (URI's) [33]. SIP defines its own URI, but its header fields can carry other URI's, such as http, mailto [24], or phone [34] URL's.

## 3.1 Basic Operation

A typical SIP transaction is depicted in Figure 1.

A SIP user agent creates an INVITE request (1) for `sip:joe@company.com`. This request is forwarded to a local proxy (1). This proxy looks up `company.com` in DNS, and obtains the IP address of a server handling SIP requests for this domain. It then proxies the request to this server (2). The server for `company.com` knows about the user `joe`, but this user is currently logged in as `j.user@university.edu`. So, the server redirects the proxy (3) to try this address. The local proxy looks up `university.edu` in DNS, and obtains the IP address of its SIP server. The request is then proxied there (4). The university server consults a local database (5), which indicates (6) that `j.user@university.edu` is known locally as `j.smith@cs.university.edu`. So, the main university server proxies the request to the computer science server (7). This server knows the IP address where the user is currently logged in, so it proxies the request there (8). The user accepts the call, and the response is returned through the proxy chain (9),(10),(11),(12) to the caller.

# 4 Session Description Protocol (SDP)

Section 3 alluded to the Session Description Protocol (SDP). A SIP invitation conveys to the recipient information needed to join an RTP session. There are many different types of RTP sessions which are possible. These include unicast sessions, initiated when one user calls another (typical two-person telephony, for example), small, interactive sessions, usually transported on multicast, such as conference calls, prescheduled multiparty sessions (such as those normally supported by a dial-up bridge), and large, broadcast sessions, where there is usually a single sender, and potentially thousands of receivers. They differ mainly in whether the start and stop times are immediate or in the future, whether the sender of the invitation also contributes data, and whether the distribution is unicast or multicast.

The multimedia component of these different session types can be carried by RTP, invitations to the sessions issued by SIP and the sessions can be described by SDP. The acronym for SDP is a misnomer, since SDP is not a protocol at all, but rather a textual description of multimedia sessions carried in a range of protocols. It conveys the following information:

**Media streams:** A multimedia session can contain many media streams, for example two audio streams, a video stream and a whiteboard session. SDP conveys the number and type of each media stream. It currently defines audio, video, data, control, and application as stream types, similar to MIME types used for Internet mail.

**Addresses:** For each stream, the destination address (unicast or multicast) is indicated. Note that the address for different media streams may differ.

**Ports:** For each stream, the UDP port numbers for sending and/or receiving are indicated.

**RTP payload types:** The types of media formats (identified by an RTP payload type for RTP) which can be used during the session are also conveyed. For unicast sessions ("traditional" IP telephony), this list is often called a *capability set*.

**Start and stop times:** For broadcast-style sessions, the start, stop, and repeat times of the session are conveyed. Thus, one can announce or invite to a weekly TV show or a Tuesday/Thursday lecture.

**Originator:** For broadcast-style sessions, the session description names the originator of the session and how that person can be contacted, e.g., in case of technical difficulties.

An example of an SDP description is shown in Figure 2.

The `v` line is a version identifier for the session. The `o` line is a set of values which form a unique identifier for the session. The `u` and `e` lines given the URL and email addresses for further information about the session. The `c` line indicates the address for the session, the `b` line indicates the bandwidth (64 kb/s), and the `t` line the start and stop times. The `k` line conveys the encryption

```
v=0
o=g.bell 87728 8772 IN IP4 132.151.1.19
s=Come here, Watson!
u=http://www.ietf.org
e=g.bell@bell-telephone.com
c=IN IP4 132.151.1.19
b=CT:64
t=3086272736 0
k=clear:manhole cover
m=audio 3456 RTP/AVP 96
a=rtpmap:96 VDVI/8000/1
m=video 3458 RTP/AVP 31
m=application 32416 udp wb
a=orient:portrait
```

Figure 2: Example SDP Description

key for the session. There are three `m` lines, each of which identifies a media stream type (audio, video, and whiteboard application), the port number for that stream, the protocol, and a list of payload types. The `a` line specifies an attribute. For example, the line below the audio stream definition defines the codec parameters for RTP payload type 96.

# 5 Session Announcement Protocol (SAP)

SIP can be used to invite a user to any type of RTP session, including broadcast style sessions, like an IP TV program. However, how does the caller find out about the details of the session in the first place? This is the function of the Session Announcement Protocol (SAP). SAP is effectively a "TV Guide" for the Internet. It is a simple protocol for encapsulating session descriptions (i.e., SDP) in a multicast packet. The session descriptions are multicast to a well-known multicast address (224.2.127.254). Anyone who wishes to advertise a session sends to this address. The session advertisements are re-broadcast periodically, so that new listeners to the address can pick up all of the current sessions.

As with RTCP, SAP announcements cannot be multicast periodically with a fixed period. Otherwise, the bandwidth used by SAP would grow with the number of sessions. Therefore, SAP has provisions for increasing the period with the number of session advertisers.

In addition to a pure encapsulation, SAP provides authentication and encryption of the SDP. SAP is implemented in the popular tool, sdr [35] as well as by commercial applications.

# 6 Real Time Streaming Protocol (RTSP)

We have not said much about how the media for an RTP session is generated. Normally, in an interactive conference, conference participants make use of software on PC's or other devices to capture audio from a microphone, or video from a framegrabber, encode it, and send it into the network. This kind of media is generally called *interactive*. However, pre-recorded media can be sent from a disk and saved on a disk. In this case, its known as *stored* media. Stored media has a number of applications in Internet telephony:

- A media server can record the content of a conference for future reference.
- Media servers can play media into an existing conference. For example, if participants in a multiparty conference are discussing a movie, it would be useful to be able to bring a media server into the conference, and have it play portions of the movie into the conference. (See the introductory scenario.)
- Voicemail is only a slight variation on media recording in that it serves as an active, media-generating endpoint as opposed to just another (silent) participant. (More precisely, a multimedia mail, as there is no reason to restrict this capability to audio only).
- The session itself may be completely stored, as in media-on-demand services.

Stored media requires a control mechanism that allows users to start, stop, record, fast-forward, and rewind the media stream. In addition, it nees to to convey to the server what stream it should be playing, and to what address to play it to. Users need to be authenticated, particularly for voicemail and media-on-demand applications. These are the functions provided by the Real Time Streaming Protocol (RTSP).

Like SIP, RTSP is a client-server protocol, based heavily on HTTP. However, RTSP does differ fundamentally in several respects. Most importantly, RTSP is stateful. This is direct consequence of the application. When a client first issues a request to play a media stream, and then later on a pause request, the pause must refer to the same stream which was initiated with the previous play. This requires state at the server and client. The mechanism for maintaining state in RTSP is the *session identifier*. During initialization, the media server hands the client this identifier. Future requests must all contain this identifier.

Conveying the identifier in the headers of the message, instead of requiring a persistent TCP connection for state, has numerous advantages. Principle among them is the ability to "pass the remote control" by allowing multiple clients to use the same session identifier. For example, a family may be watching a movie that is being multicast. Any member of the family can issue pause or play requests with the same identifier, allowing any one of them to control delivery. A more interesting application is remote teaching. A set of teachers can control the playback of media to students, pausing the media when there are questions, and fast forwarding to certain points in the media stream where there is information of interest.

Media streams that can be controlled by RTSP are identified by an RTSP URL. An example of such a URL is rtsp://server.foo.com/movies/twister:5080. This identifies the movie *Twister* on the machine server.foo.com, port 5080. A URL may refer to a single component of the presentation (rtsp://server.foo.com/movies/twister/audio) or the entire presentation. Furthermore, requests to the server may refer to either a component, or to the entire presentation. Having them refer to each component individually allows the components to be on different servers. In that case, synchronization of the media is the responsibility of the client, and is typically done using mechanisms in RTCP.

The process for streaming the presentation follows a few basic steps:

1. Obtain the presentation description. The first step is to obtain the presentation description. This description enumerates the various components of the session, and defines the media parameters for them (codec type, frame rate, etc.). There may be many components of varying types (audio, video) in the session. The only requirement is that each be part of the same time axis. The presentation description can be obtained in several ways. The most direct is via the DESCRIBE method, which allows a client to request the description from a server.

It is also possible to obtain a description through other means, such as a web page. There can be many formats for the description; there is no mandatory one. The DESCRIBE method allows the user to list (in the Allows header field) those descriptions which it understands. The Session Description Protocol (SDP) [5], discussed above, is one possibility.

2. Setup the server. Once it has obtained the description, the client can issue a SETUP request to the server. This request establishes the destination to which the media should be delivered. This includes information such as the address (unicast or multicast), the port numbers, protocols, TTL, number of multicast layers, etc. All are specified as part of the Transport header, included in the request and response. Having all of this information appear in one place, in a single header, simplifies firewall operation. The firewall need only search for the single header field, and parse it, in order to open a hole in the firewall for the media. As part of the response to the SETUP message, the server will return a session id. This id is used in further requests.

3. Issue media requests. After the stream has been set up, the client can issue media requests to the server. These include operations such as PLAY, RECORD, and PAUSE. The PLAY method encompasses seek, fast forward and reverse, in addition to straight play. This is accomplished by including a Scale header which indicates the time speedup (or speeddown) at which the media should be played. The Range header specifies where in the stream playback should start from. The Range header supports specification of time in units of SMPTE, NPT, and absolute time. All play requests are idempotent, and are queued and serviced in order at the server. A pause request allows for the specification of a time at which the pause should activate. This cancels any pending play requests, queued at the server, scheduled to start after this time.

4. Teardown. Once interaction with the server is complete, the client issues a TEARDOWN request. This request destroys any state associated with the session. Further requests for the given session id will no longer be valid.

RTSP provides a number of other methods for more advanced features. ANNOUNCE has two semantics, depending on whether the client or server sends the request. When sent by a client, it tells the server to update the presentation description with the content enclosed in the request. When sent by a server, it indicates a change in the description of a presentation being used by the client. The SET_PARAMETER and GET_PARAMETER methods set and get parameters of the media server. This is useful for adjusting camera controls of a real-time video server, for example. The REDIRECT method allows the server to tell the client to access the media stream at a different location. Finally, the OPTIONS method queries the server about its capabilities.

RTSP can be integrated with SIP as well. A client can first use SIP to invite a media server to a conference. This invitation would include the multicast address of the conference, and other relevant transport parameters. The server then takes note of the Call-ID field in the SIP request. When a client wishes to control the media server (by having it play into the conference), it can issue a SETUP request, and include the Conference header in the request. This header contains the same Call-ID as used in the SIP invitation. This allows the media server to determine the media and transport parameters for subsequent RTSP requests. The response to the SETUP request contains the session id as usual. Clients can then issue PLAY, RECORD, and PAUSE requests to the server.

```
<call>
  <condition weekday="m-f" time="9-17">
    <match>
      <proxy />
    </match>
    <nomatch>
      <proxy dest="sip:jones@voicemail.example.com" />
    </nomatch>
  </condition>
</call>
```

Figure 3: Time of Day Routing Script

# 7 Call Processing Language (CPL)

SIP requests can traverse through a number of different proxy and redirect servers in the process of searching for the callee. A callee, however, may have preferences about how this search takes place. For example, a user might like to first be tried at work, and then, if no one answers, tried at home, but only before 10 pm and if the caller is known to the callee. Another user may prefer to be contacted at work only if the boss is calling; all other calls should be directed to voicemail.

To support these diverse preferences, users need to be able to express preferences about call services. These preferences are read and processed by a SIP proxy or redirect server. The server uses them to direct its behavior for SIP call handling. These preferences are called a *Call Processing Language*, or CPL. They are currently in early development in the IP Telephony (iptel) working group.

The syntax for the CPL is likely to be based on the Extended Markup Language (XML). An example of what a script written in this language might look like is shown in Figure 3. This script tells the server that if a call is received during the weekday, it should be forwarded as normal to the current location of the user. Otherwise, the call is routed to a SIP URL which corresponds to a voicemail server.

# 8 Gateway Location Protocol (GLP)

SIP allows a user on the Internet to call another user, also on the Internet. What if an Internet user wishes to call someone not on the Internet, but rather, on the telephone network? In this case, an Internet telephony gateway is needed. This device is capable of converting signaling and media between a packet network and the telephone network (PSTN, "public switched telephone network"). To complete a call to a PSTN endpoint, a user must send a SIP invitation to the gateway. However, given a phone number to call, how does the caller find and select a gateway to complete the call? This function is supported by means of the Gateway Location Protocol (GLP).

An overall architecture for GLP is shown in 4.

There are a number of domains in the Internet, each with IP telephony gateways (GW), end users that wish to make calls to the telephone network (User Agents, or UA's), and a number of Location Servers (LS's). LS's are the primary participants in the protocol. An LS learns about the gateways within its domain. This can be done through manual configuration, or via some other protocol, generically called the *intra-domain protocol*. Location servers from different domains are configured to connect to each other, and then exchange gateway information using GLP. LS's can

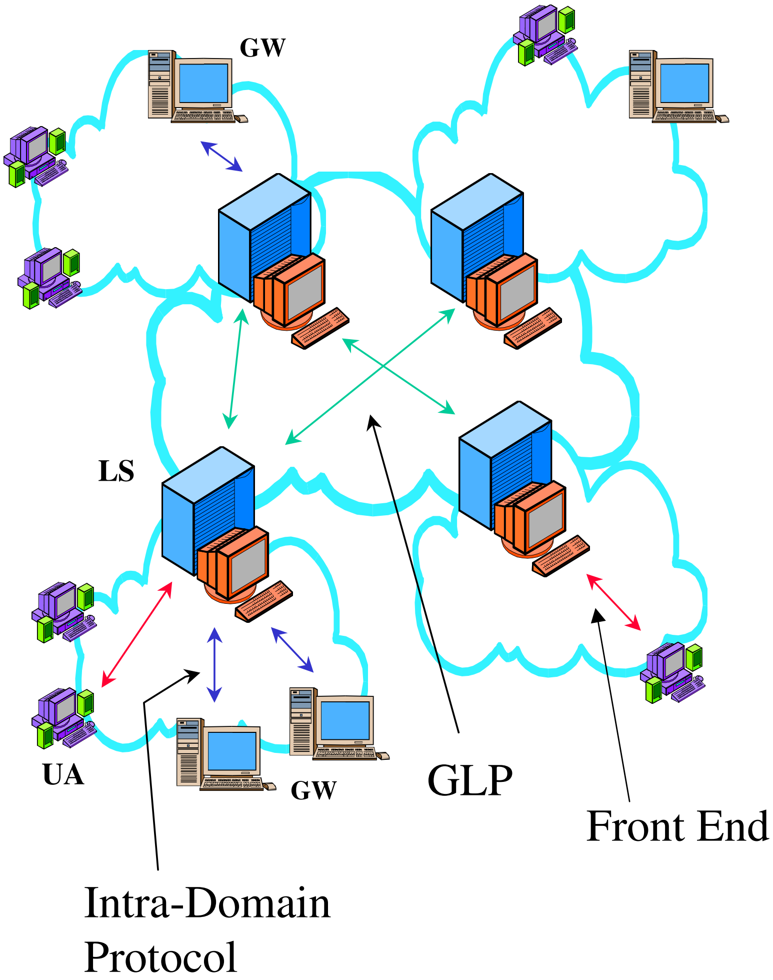


Figure 4: GLP Operation

take the information they learn, aggregate and filter it based on policy, and then forward this information to other LS's. In this way, the LS in each domain learns about gateways from remote domains.

The gateway information collected contains several parameters. These include the phone numbers which the gateway is willing to complete calls to, the number of ports (overall capacity) of the gateway, and possibly the protocols, codecs, and other administrative parameters that describe the gateway. These parameters can be used by an administrative domain to select which gateways to make available to users in the domain. It can also be examined by end users to express preferences about which gateway to use.

A user agent wishing to make a telephone call needs to access this information from its local LS in some way. The means by which this is done is called the *front end*. There are many mechanisms for the front end. Some of the possibilities include:

**Lightweight Directory Access Protocol (LDAP):** The LS can store its information in a database. When a client wishes to make a call, it sends an LDAP [36] query to the LS, containing the desired parameters for the gateway. The response contains the IP address of the gateway. The client can then send a SIP invitation to that address.

**SIP:** Rather then sending a SIP invitation directly to the gateway, the caller can send it to the LS. The invitation contains the desired phone number, so the LS can consult its database, find the right gateway, and proxy the call to it. In this case, the LS also acts as a SIP proxy. This has the benefit that it hides the gateway selection process from the caller. The caller application doesn't need to know whether the address being called is a phone number or a SIP URL. In either case, the invitation is sent to the local proxy.

**Web pages:** The LS can make its database available through web pages. A user that wishes to make a call browses the web page, finds the gateway it likes (perhaps this can be done through a web form), and the LS returns the address of the gateway on the web page. The user copies this address to their SIP software, and completes a call to the gateway.

The GLP is just beginning specification. As it is similar to existing inter-domain IP routing protocols, such as BGP4 [37], it is likely to borrow heavily from them.

## 9 Conclusion

In this paper, we have overviewed the various working groups in IETF which are focusing on IP telephony related protocols. A number of protocols being developed there were surveyed. These include the Real Time Transport Protocol (RTP), the Session Initiation Protocol (SIP), the Session Description Protocol (SDP), the Session Announcement Protocol (SAP), the Call Processing Language (CPL), and the Gateway Location Protocol (GLP).

There are a number of related efforts in the IETF which are just taking shape. For example, there will likely be a working group (sigtran) to examine the transport of PSTN signaling across the Internet, independent of whether the actual voice data uses the Internet or a traditional circuit-switched network. Also, a different group (megaco) is investigating protocols for controlling media gateways, such as Internet telephony gateways. In that approach, Internet end systems do not directly signal calls to each other, but are rather dumb end systems controlled by a media controller that tells them to, say, send audio data to a certain IP address.

The Internet multimedia architecture currently lacks a number of protocols useful for conferencing and telephony, such as floor control protocols coordinating who gets to send data in a group, or protocols for application sharing and white boards.

Important protocol issues left unresolved include how to collect and transmit billing information as well as the problems of admission, authentication and accounting. Both of these areas affect many services, so that Internet telephony will likely re-use and extend more generic work rather than define service-specific protocols.

## References

[1] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications," Request for Comments (Proposed Standard) 1889, Internet Engineering Task Force, Jan. 1996.

[2] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: session initiation protocol," Internet Draft, Internet Engineering Task Force, Nov. 1998. Work in progress.

[3] H. Schulzrinne, R. Lanphier, and A. Rao, "Real time streaming protocol (RTSP)," Request for Comments (Proposed Standard) 2326, Internet Engineering Task Force, Apr. 1998.

[4] M. Handley, "SAP: Session announcement protocol," Internet Draft, Internet Engineering Task Force, Nov. 1996. Work in progress.

[5] M. Handley and V. Jacobson, "SDP: session description protocol," Request for Comments (Proposed Standard) 2327, Internet Engineering Task Force, Apr. 1998.

[6] M. Handley, J. Crowcroft, C. Bormann, and J. Ott, "The internet multimedia conferencing architecture," Internet Draft, Internet Engineering Task Force, July 1997. Work in progress.

[7] J. Rosenberg and H. Schulzrinne, "A framework for a gateway location protocol," Internet Draft, Internet Engineering Task Force, Oct. 1998. Work in progress.

[8] H. Schulzrinne and J. Lennox, "Call processing language requirements," Internet Draft, Internet Engineering Task Force, Aug. 1998. Work in progress.

[9] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," Internet Draft, Internet Engineering Task Force, Oct. 1998. Work in progress.

[10] D. Cohen, "Specifications for the network voice protocol NVP," RFC 741, Internet Engineering Task Force, Nov. 1977.

[11] D. Cohen, "A network voice protocol: NVP-II," technical report, University of Southern California/ISI, Marina del Ray, California, Apr. 1981.

[12] International Telecommunication Union, "Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service," Recommendation H.323, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, May 1996.

[13] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Toronto, Canada), pp. 680–688, IEEE Computer Society Press, Los Alamitos, California, June 1994.

[14] W. A. Montgomery, "Techniques for packet voice synchronization," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, pp. 1022–1028, Dec. 1983.

[15] S. B. Moon, J. Kurose, and D. Towsley, "Packet audio playout delay adjustment: performance bounds and algorithms," *ACM/Springer Multimedia Systems*, vol. 5, pp. 17–28, Jan. 1998.

[16] H. Schulzrinne, "RTP profile for audio and video conferences with minimal control," Request for Comments (Proposed Standard) 1890, Internet Engineering Task Force, Jan. 1996.

[17] T. Turletti and C. Huitema, "RTP payload format for H.261 video streams," Request for Comments (Proposed Standard) 2032, Internet Engineering Task Force, Oct. 1996.

[18] C. Zhu, "RTP payload format for H.263 video streams," Request for Comments (Proposed Standard) 2190, Internet Engineering Task Force, Sept. 1997.

[19] R. Civanlar, G. Fernando, V. Goyal, and D. Hoffman, "RTP payload format for MPEG1/MPEG2 video," Request for Comments (Proposed Standard) 2250, Internet Engineering Task Force, Jan. 1998. (Obsoletes RFC2038).

[20] J.-C. Bolot and A. V. Garcia, “Control mechanisms for packet audio in the internet,” in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (San Fransisco, California), Mar. 1996.

[21] I. Busse, B. Deffner, and H. Schulzrinne, “Dynamic QoS control of multimedia applications based on RTP,” *Computer Communications*, vol. 19, pp. 49–58, Jan. 1996.

[22] C. Perkins and C. Perkins, “Options for repair of streaming media,” Request for Comments (Informational) 2354, Internet Engineering Task Force, June 1998.

[23] J. Rosenberg and H. Schulzrinne, “Timer reconsideration for enhanced RTP scalability,” in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (San Francisco, California), March/April 1998.

[24] P. Hoffman, L. Masinter, and J. Zawinski, “The mailto URL scheme,” Request for Comments (Proposed Standard) 2368, Internet Engineering Task Force, July 1998.

[25] J. Postel, “Simple mail transfer protocol,” Request for Comments (Standard) STD 10, 821, Internet Engineering Task Force, Aug. 1982. (Obsoletes RFC788).

[26] R. Fielding, J. Gettys, J. Mogul, H. Nielsen, and T. Berners-Lee, “Hypertext transfer protocol – HTTP/1.1,” Request for Comments (Proposed Standard) 2068, Internet Engineering Task Force, Jan. 1997.

[27] D. Zimmerman, “The finger user information protocol,” Request for Comments (Draft Standard) 1288, Internet Engineering Task Force, Dec. 1991. (Obsoletes RFC1196).

[28] P. Deutsch, R. Schoultz, P. Faltstrom, and C. Weider, “Architecture of the WHOIS++ service,” Request for Comments (Proposed Standard) 1835, Internet Engineering Task Force, Aug. 1995.

[29] H. Schulzrinne and J. Rosenberg, “SIP call control services,” Internet Draft, Internet Engineering Task Force, Feb. 1998. Work in progress.

[30] H. Schulzrinne and J. Rosenberg, “Signaling for internet telephony,” Technical Report CUCS-005-98, Columbia University, New York, New York, Feb. 1998.

[31] H. Schulzrinne and J. Rosenberg, “Signaling for internet telephony,” in *Proc. of 6th IEEE International Conference on Network Protocols (ICNP)*, (Austin, Texas), Oct. 1998.

[32] N. Freed and N. Borenstein, “Multipurpose internet mail extensions (MIME) part one: Format of internet message bodies,” Request for Comments (Draft Standard) 2045, Internet Engineering Task Force, Dec. 1996. (Obsoletes RFC1521); (Updated by RFC2184); (Updated by RFC2231).

[33] T. Berners-Lee, R. Fielding, and L. Masinter, “Uniform resource identifiers (URI): generic syntax,” Request for Comments (Draft Standard) 2396, Internet Engineering Task Force, Aug. 1998.

[34] A. Vaha-Sipila, “URLs for telephone calls,” Internet Draft, Internet Engineering Task Force, Aug. 1998. Work in progress.

[35] M. Handley *et al.*, “sdr (session directory).” http://www-mice.cs.ucl.ac.uk/multimedia/software/sdr/.

[36] W. Yeong, T. Howes, and S. Kille, "Lightweight directory access protocol," Request for Comments (Draft Standard) 1777, Internet Engineering Task Force, Mar. 1995. (Obsoletes RFC1487).

[37] Y. Rekhter and T. Li, "A border gateway protocol 4 (BGP-4)," Request for Comments (Draft Standard) 1771, Internet Engineering Task Force, Mar. 1995. (Obsoletes RFC1654).