A Comparison of SIP and H.323 for Internet Telephony

Henning Schulzrinne
Dept. of Computer Science, Columbia University
New York, NY 10027
hgs@cs.columbia.edu

Jonathan Rosenberg
Bell Laboratories
Holmdel, NJ 07733
jdrosen@bell-labs.com

Abstract—**Two standards have recently emerged for signaling and control for Internet Telephony. One is ITU Recommendation H.323, and the other is the IETF Session Initiation Protocol (SIP). These two protocols represent very different approaches to the same problem: H.323 embraces the more traditional circuit-switched approach to signaling based on the ISDN Q.931 protocol and earlier H-series recommendations, and SIP favors the more lightweight Internet approach based on HTTP. In this paper, we compare SIP and H.323 on complexity, extensibility, scalability, and features.**

I. Introduction

In order to provide useful services, Internet telephony requires a set of control protocols for connection establishment, capabilities exchange, and conference control. Currently, two protocols exist to meet this need. One is ITU-T H.323, and the other is the IETF Session Initiation Protocol (SIP). In this paper, we compare the two protocols on complexity, extensibility, scalability, and services.

The ITU H.323 series of recommendations ("Packet Based Multimedia Communications Systems") defines protocols and procedures for multimedia communications on, among other things, the Internet. It includes H.245 for control, H.225.0 for connection establishment, H.332 for large conferences, H.450.1 H.450.2 and H.450.3 for supplementary services, H.235 for security, and H.246 for interoperability with circuit-switched services. H.323 started out as a protocol for multimedia communication on a LAN segment without QoS guarantees, but has evolved to try and fit the more complex needs of Internet telephony.

H.323 is based heavily on the ITU multimedia protocols which preceded it, including H.320 for ISDN, H.321 for B-ISDN, and H.324 for GSTN terminals. The encoding mechanisms, protocol fields, and basic operation are somewhat simplified versions of the Q.931 ISDN signaling protocol.

The Session Initiation Protocol (SIP) [1], developed in the MMUSIC working group of the IETF, takes a different approach to Internet telephony signaling by reusing many of the header fields, encoding rules, error codes, and authentication mechanisms of HTTP.

In both cases, multimedia data will likely be exchanged via RTP, so that the choice of protocol suite does not influence Internet telephony QOS.

II. Complexity

H.323 is a rather complex protocol. The sum total of the base specifications alone (not including ASN.1 and PER) is 736 pages. SIP, on the other hand, along with its call control extensions and session description protocols totals merely 128 pages. H.323 defines hundreds of elements, while SIP has only 37 headers (32 in the base specification, 5 in the call control extensions), each with a small number of values and parameters, but that contain more information. A basic, but interoperable SIP Internet telephony implementation can get by with four headers (To, From, Call-ID, and CSeq) and three request types (INVITE, ACK, and BYE) and is small enough to be assigned as a homework programming problem. A fully functional SIP client agent, with a graphical user interface, has been implemented in just two man-months.

H.323 uses a binary representation for its messages, based on ASN.1 and the packed encoding rules (PER). ASN.1 generally requires special code-generators to parse. SIP, on the other hand, encodes its messages as text, similar to HTTP [2] and the Real Time Streaming Protocol (RTSP) [3]. This leads to simple parsing and generation, particularly when done with powerful text processing languages such as Perl. The textual encoding also simplifies debugging, allowing manual entry and perusing of messages. Its similarity to HTTP also allows for code-reuse; existing HTTP parsers can be quickly modified for SIP usage.

H.323's complexity also stems from its use of several protocol components. There is no clean separation of these components; many services require interactions between several of them. (Call forward, for example, requires components of H.450, H.225.0, and H.245.) The use of several different protocols also complicates firewall traversal. Firewalls must act as application level proxies [4], parsing the entire message to arrive at the required fields. The operation is stateful since several messages are involved in call setup. SIP, on the other hand, uses a single request that contains all necessary information.

H.323 also provides for an array of options and methods for accomplishing a single task. For example, there are three distinct ways in which H.245 and H.225.0 may be used together: the original H.323v1 approach of separate connections, H.245 tunneling through H.225.0, and FastStart in H.323v2. In the original approach, the call signaling channel is set up first, the H.245 control channel is established, and finally the media channels are opened. This can require many round trips for call setup. FastStart includes the media channel information in the original call invitation, avoiding the need to open the H.245 channel. In H.245 tunnelling, the H.245 channel is still used, but its messages are carried over the call signaling channel. Even though FastStart is much more efficient, H.323 allows any of the three and thus, firewalls, end systems, gatekeepers, and gateways must support all of them. As with any protocol, large option spaces lead to feature interaction and the need for profiles. (How does encryption of the H.245 channel work when its tunneled through H.225.0, for example?).

An additional aspect of H.323's complexity is its duplication of some of the functionality present in other parts of the protocol. In particular, H.323 makes use of RTP and RTCP. RTCP has been engineered to provide various feedback and conference control functions in a manner which scales from two-party conferences to thousand-party broadcast sessions. H.245, however, provides its own mechanisms for both feedback and simple conference control (such as obtaining the list of conference participants). These H.245 mechanisms are redundant, and have been engineered for small to medium-sized conferences only.

III. Extensibility

Extensibility is a key metric for measuring an IP telephony signaling protocol. Telephony is a tremendously popular, critical service, and Internet telephony is poised to supplant the existing circuit switched infrastructure developed to support it. As with any heavily used service, the features provided evolve over time as new applications are developed. This makes compatibility among versions a complex issue. As the Internet is an open, distributed, and evolving entity, one can expect extensions to IP telephony protocols to be widespread and uncoordinated. This makes it critical to build in powerful extension mechanisms from the outset.

SIP has learned the lessons of HTTP and SMTP (both of which are widely used protocols that have evolved over time), and built in a rich set of extensibility and compatibility functions. By default, unknown headers and values are ignored. Using the Require header, clients can indicate named feature sets that the server must understand. When a request arrives at a server, it checks the list of named features in the Requires header. If any of them are not supported, the server returns an error code and lists the set of features it does understand. The client can then determine the problematic feature and fall back to simpler operation. The feature names are based on a hierarchical namespace, and new feature names can be registered with IANA. This means that any developer can create new features in SIP, and then simply register a name for them. Compatibility is still maintained across different versions.

To further enhance extensibility, numerical error codes are hierarchically organized, as in HTTP. There are six basic classes, each of which is identified by the hundreds digit in the response code. Basic protocol operation is dictated solely by the class, and terminals need only understand the class of the response. The other digits provide additional information, usually useful but not critical. This allows for additional features to be added by defining semantics for the error codes in a class, while achieving compatibility.

The textual encoding means that header fields are self-describing. It is self-evident what the meaning of the To, From, and Subject fields are. As new header fields are added in various different implementations, developers in other corporations can determine usage just from the name, and add support for the field. This kind of distributed, documentation-less standardization has been common in the Simple Mail Transfer Protocol (SMTP), which has evolved tremendously over the years.

As SIP is similar to HTTP, mechanisms being developed for HTTP extensibility can also be used in SIP. Among these are the Protocol Extensions Protocol (PEP), which contains pointers to the documentation for various features within the HTTP messages themselves.

H.323 provides extensibility mechanisms as well. These are generally nonstandardParam fields placed in various locations in the ASN.1. These params contain a vendor code, followed by an opaque value which has meaning only for that vendor. This does allow for different vendors to develop their own extensions. However, it has some limitations. First, extensions are limited only to those places where a non-standard parameter has been added. If a vendor wishes to add a new value to some existing parameter, and there is no placeholder for a nonstandard element, one cannot be added. Secondly, H.323 has no mechanisms for allowing terminals to exchange information about which extensions each supports. As the values in non-standard parameters are not self-describing, this limits interoperability among terminals from different manufacturers.

In addition, H.323 requires full backwards compatibility from each version to the next. As various features come and go, the size of the encodings will only increase. However, SIP allows for older headers and features to gradually disappear as they are no longer needed, keeping the protocol and its encoding clean and concise.

A critical issue for extensibility are audio and video codecs. There are hundreds of codecs that have been developed, many of which are proprietary. SIP uses the Session Description Protocol (SDP) to convey the codecs supported by an endpoint in a session. Codecs are identified by string names, which can be registered by any person or group with IANA, and then used. This means that SIP can work with any codec, and other implementations can determine the name of the codec, and contact information for it, from IANA.

In H.323, each codec must be centrally registered and standardized. Currently, only ITU developed codecs have codepoints. As many of these carry significant intellectual property, there is no free, sub-28.8 kb/s codec which can be used in an H.323 system. This presents a significant barrier to entry for small players and universities.

Furthermore, SIP allows for new services to be defined through a few powerful third-party call control mechanisms. These mechanisms allow a third party to instruct another entity to create and destroy calls to other entities. As the controlled party executes the instructions, status messages are passed back to the controller. This allows the controller to take further actions based on some local program execution. This is much like the IN model in traditional telephony. As there are hundreds of telephony services currently defined, it is unreasonable to attempt to write specifications for each. SIP allows these services to be deployed by basing them on simple, standardized mechanisms. These mechanisms can be used to construct a variety of services, including blind transfer, operator assisted transfer, three-party calling, bridged calling, dial-in bridging, multi-unicast to multicast transitions, ad-hoc bridge invitation and transition, and various forwarding variations [5].

As an example of these extension and service creation mechanisms, the PSTN and Internet Internetworking (pint) working group in IETF is defining a simple SIP extension for click-to-call type of services. In this scenario, a user at a web page clicks on a button, and a PSTN entity connects the user's telephone to a customer service rep. This requires a control protocol between the web server and a PSTN-enabled device. SIP is being used as this protocol.

H.323 does provide some basic mechanisms along this line. The FACILITY message allows a callee to direct a caller to contact a different party (basically, a blind transfer). Another is the H.245 CommunicationModeCommand, which allows the MC to change the media encodings for a conference for the various participants. The former is fairly limited in scope, and the latter can only be executed by the MC for the call. Neither provide generic third party control mechanisms needed for building complex services.

Another aspect of extensibility is modularity. Internet telephony requires a large number of different functions; these include basic signaling, conference control, quality of service, directory access, service discovery, etc. One can be certain that mechanisms for accomplishing these functions will evolve over time (especially with regards to QoS). This makes it critical to apportion these functions to seperate, modular, orthogonal components, which can be swapped in and out over time. It is also critical to use seperate, general protocols for each of these functions. This allows for the function to be duplicated in other applications with ease. For example, it is more efficient to have a single QoS mechanism which is application independent, rather than invent a new QoS protocol or mechanism for each application.

SIP is reasonably modular. It encompasses basic call signaling, user location, and registration. Advanced signaling is part of SIP, but within a single extension. Quality of service, directory accesses, service discovery, session content description, and conference control are all orthogonal, and reside in separate protocols. For example, it is possible to use the H.245 capability description elements in SIP, with no changes to SIP at all.

H.323 is less modular. It defines a vertically integrated protocol suite for a single application. The mix of services provided by the H.323 components encompass capability exchange, conference control, maintenance operations, basic signaling, quality of service, registration, and service discovery. Furthermore, these are intertwined within the various sub-protocols within H.323.

SIP's modularity allows it to be used in conjunction with H.323. A user can use SIP to locate another user, taking advantage of its rich multi-hop search facilities. When the user is finally located, they can use a redirect response to an H.323 URL, indicating that the actual communication should take place with H.323.

IV. Scalability

We also find that H.323 and SIP differ in terms of scalability. We can observe scalability on a number of different levels:

Large Numbers of Domains: H.323 was originally conceived for use on a single LAN. Issues such as wide area addressing and user location were not a concern. The newest version defines the concept of a zone, and defines procedures for user location across zones for email names. However, for large numbers of domains, and complex location operations, H.323 has scalability problems. It provides no easy way to perform loop detection in complex multi-domain searches (it can be done statefully by storing messages, which is not scalable). SIP, however, uses a loop detection algorithm similar to the one used in BGP, which can be performed in a stateless manner.

Server Processing: In an H.323 system, both telephony gateways and gatekeepers will be required to handle calls from a multitude of users. Similarly, SIP servers and gateways will need to handle many calls. For large, backbone IP telephony providers, the number of calls being handled by a large server can be significant.

In SIP, a transaction through several servers and gateways can be either stateful or stateless. In the stateless model, a server receives a call request, performs some operation, forwards the request, and completely forgets about it. SIP messages contain sufficient state to allow for the response to be forwarded correctly. Furthermore, SIP can be carried on either TCP or UDP. In the case of UDP, no connection state is required. This means that large, backbone servers can be based on UDP and operate in a stateless fashion, reducing signficantly the memory requirements and improving scalability.

H.323, on the other hand, requires gatekeepers (when they are in the call loop), to be stateful. They must keep call state for the entire duration of a call. Furthermore, the connections are TCP based, which means a gatekeeper must hold its TCP connections for the entire duration of a call. This can pose serious scalability problems for large gatekeepers.

Furthermore, a gateway or gatekeeper will need to process the signaling messages for each call. The simpler the signaling, the faster it can be processed, and the more calls a gateway or gatekeeper can support. As SIP is simpler to process than H.323, SIP should allow more calls per second to be handled on particular box than H.323. [1]

Conference Sizes: H.323 supports multiparty conferences with multicast data distribution. However, it requires a central control point (called an MC) for processing all signaling, for even the smallest conferences. This presents several difficulties. Firstly, should the user providing the MC functionality leave the conference, and exit their application, the entire conference terminates. In addition, since MC and gatekeeper functionality is optional, H.323 cannot support even three party conferences in some cases. We note that the MC is a bottleneck for larger conferences. To alleviate this, the latest version of H.323 has defined the concept of cascaded MC's, allowing for a very limited application layer multicast distribution tree of control messaging. This improves scaling somewhat, but for even larger conferences, the H.332 protocol defines additional procedures. This means that three distinct mechanisms exist to support conferences of different sizes. SIP, however, scales to all different conference sizes. There is no requirement for a central MC; conference coordination is fully distributed. This improves scalability and complexity. Furthermore, as it can use UDP as well as TCP, SIP supports native multicast signaling, allowing a single

[1] The authors are not aware of any study measuring the processing overhead of SIP and H.323, however.

Feature	SIP	H.323
Blind Transfer	Yes	Yes
Operator Assisted Transfer	Yes	No
Hold	Yes; through SDP	Not yet
Multicast Conferences	Yes	Yes
Multi-unicast Conferences	Yes	Yes
Bridged Conferences	Yes	Yes
Forward	Yes	Yes
Call Park	Yes	No
Directed Call Pickup	Yes	No

TABLE I
SIP AND H.323 CALL CONTROL FEATURE COMPARISON

protocol to scale from sessions with two to millions of members. *Feedback:* H.245 defines procedures that allow receivers to control media encodings, transmission rates, and error recovery. This kind of feedback makes sense in point-to-point scenarios, but ceases to be functional in multipoint conferencing. SIP, instead, relies on RTCP for providing feedback on reception quality (and also for obtaining group membership lists). RTCP, like SIP, operates in a fully distributed fashion. The feedback it provides automatically scales from a two person point to point conference to huge broadcast style conferences with millions of participants.

V. SERVICES

H.323 and SIP offer roughly equivalent services. Some of the call control services are listed in Table 1.

As can be seen from the chart, SIP and H.323 support similar services. A comparison in these dimensions is somewhat difficult, as new services are always being added to both SIP and H.323. We expect that the above table will be different upon printing of this paper.

In addition to call control services, both SIP (when used with SDP) and H.323, provide capabilities exchange services. In this regard, H.323 provides a much richer set of functionality. Terminals can express their ability to perform various encodings and decodings based on parameters of the codec, and based on which other codecs are in use. However, most implementations don't require (or implement) these, and the basic receiver-capability indication supported by SIP ("choose any subset of these encodings for this list of media streams") seems sufficient and equivalent to current H.323 capabilities actually implemented.

SIP provides rich support for personal mobility services, however. When a caller contacts the callee, the callee can redirect the caller to a number of different locations. Each of these locations can be an arbitrary URL, and contains additional information about the terminal at that location. Information on language spoken, business or home, mobile phone or fixed, and a list of callee priorities, can be conveyed for each location. This allows the caller flexibility in choosing which location to talk to. For non-interactive terminals, the original call setup can convey caller preferences about the nature of the terminal to be contacted. This allows network proxies to forward the call based on these preferences.

SIP also supports multi-hop "searches" for a user. When a call request is made to some particular address, a SIP server is contacted at that address. As this SIP server may not be the machine that the callee is currently residing at, the server can proxy the request to one or more additional servers. These servers, in turn, may further proxy the request until the party is contacted. A server can actually proxy the request to multiple servers in parallel. This allows the search for the user to operate more rapidly. SIP also allows multiple branches of the search to accept the call, passing the responses back to the caller. The caller can then decide which party to speak to. This would allow a call for `j.doe@company.com` to be picked up by both Mr. Doe, his wife, and an answering machine. The caller can then hang up with the answering machine and continue with a three party call, if they so desire.

H.323's support for this kind of mobility is more limited. The facility message can redirect a caller to try several other addresses (much like 300 class response codes in SIP). However, it cannot be used to express preferences, nor can the caller express preferences in the original call invitation. H.323 wasn't engineered for wide area operation; it does support forwarding of call requests among servers, but has no mechanisms for loop detection. H.323 doesn't allow a gatekeeper to proxy a request to multiple servers either.

H.323 supports various conference control services, including chair selection, "mike passing", and conference participant determination. SIP does not provide conference control, relying instead on other protocols for this service. Some simple forms of conference control (such as sending notes around, and obtaining a conference participant listing), are available through RTCP, however.

VI. CONCLUSION

In this paper, we have compared SIP and H.323 in terms of complexity, extensibility, scalability, and services. We have found that SIP provides a similar set of services to H.323, but provides far lower complexity, rich extensibility, and better scalability. Future work is to more fully evaluate the protocols, and examine quantitative performance metrics to characterize these differences.

REFERENCES

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

[2] 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.

[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] Anonymous, "H.323 and firewalls: The problems and pitfalls of getting H.323 safely through firewalls," Developer note, Intel Corporation, Apr. 1997.

[5] Henning Schulzrinne and Jonathan Rosenberg, "Signaling for internet telephony," Technical Report CUCS-005-98, Columbia University, New York, New York, Feb. 1998.