The Session Initiation Protocol (SIP): An Evolutionary Study

Salman Abdul Baset, Vijay K. Gurbani, Alan B. Johnston,
Hadriel Kaplan, Brian Rosen and Jonathan D. Rosenberg

Abstract— **The Session Initiation Protocol (SIP) was developed to control multi-media sessions on the Internet. Shortly after its debut as a standard in 1999, SIP was adopted by the 3rd Generation Partnership Project (3GPP) as the preferred signaling protocol for the Internet Multimedia Subsystem (IMS). This adoption provided a boost to the nascent protocol as traditional telecommunication services were interpreted in the context of the new protocol and as SIP introduced richer services in the form of instant messaging and rich presence to traditional telephony. In this paper, we study the evolution of the protocol from its roots to its use in operational networks today and the issues it faces in such networks. We also provide a glimpse to the continued progression of SIP in Peer-to-Peer (P2P) networks and take a critical look at where SIP has succeeded, and more importantly, where it has failed to meet expectations.**

Index Terms— **Multimedia, Protocol, Signaling, SIP, RTP, Services**

I. Introduction

SIP [1], [2] is well-established today as a signaling protocol to set up, maintain, and tear down multimedia sessions over the Internet. As hard as it may now seem, SIP was not specifically invented to do telephony over the Internet. Instead, it was designed to work effectively with and integrate into the emerging Internet multimedia architecture being developed by the Internet Engineering Task Force (IETF) in 1996. This architecture was focused around multicast conference management and conference setup and discovery. The core protocols in the multimedia architecture consisted of the Session Announcement Protocol (SAP [3]), Session Description Protocol (SDP [4]), Real-Time Streaming Protocol (RTSP [5]) and Real-Time Transport Protocol (RTP [6]). SAP is a very simple announcement protocol for conveying information about sessions to potential listeners. SDP is a standard way to describe multimedia sessions. RTSP is used to set up and control (rewind, forward) multimedia streams from a media server. RTP is a protocol for transporting real-time data such as audio, video, or instant messages. In 1996, these were the protocols that powered the Internet multicast backbone (MBONE) for loosely coupled multicast conferences. SIP was introduced into this mix.

In early 1996, there were two protocols competing for session establishment in the IETF: the Session Invitation Protocol and the Simple Conference Invitation Protocol (SCIP). Session Invitation Protocol was responsible only for setting up the session and providing rudimentary capability negotiation; once the session had started, it was not used at all [7]. The protocol was designed to run over User Datagram Protocol (UDP) only, and it used SDP for describing the session. SCIP, on the other hand, was based on Transmission Control Protocol (TCP), and it persisted even after the session was established. It also aided in tearing the session down [7]. SCIP used existing Internet protocols such as Hypertext Transport Protocol (HTTP [8]) and Simple Mail Transport Protocol (SMTP), but did not use SDP for describing its sessions. Unlike Session Invitation Protocol, SCIP was designed with an eye toward telephony functionality. In late 1996, both of these protocols were merged to form Session Initiation Protocol, which borrowed ideas from each of its parents [9]. From Session Invitation Protocol, SIP inherited its UDP base and its usage of SDP; from SCIP, SIP inherited its support for TCP and its affinity to other important IETF protocols (such as SMTP and HTTP). The new protocol was called SIP/2.0 to distinguish it from SIP/1.0, the Session *Invitation* Protocol.

The combined protocol had many advantages. SIP was designed to be a rendezvous protocol — i.e., given a user to locate, the protocol would use any means at its disposal to find the user. It could relay the session invitation between SIP servers, redirect the invitation to other servers, or it could fork (replicate the session invitation to multiple search branches) in parallel or sequentially to locate the user. SIP carried the session description as a body and did a minimum amount of session negotiation. SIP supported primitives to set up and tear down sessions and query end users for capabilities. Instead of using numeric identifiers to represent users, SIP used the ubiquitous and well-known e-mail-like identifiers. All of these advantages and

Salman Abdul Baset (sabaset@us.ibm.com) is with IBM.
Vijay K. Gurbani (vkg@bell-labs.com) is with Bell Laboratories, Alcatel-Lucent.
Alan B. Johnston (alan@ese.wustl.edu) is with Avaya, Inc.
Hadriel Kaplan (hkaplan@acmepacket.com) is with Acme Packet
Brian Rosen (br@brianrosen.net) is with NeuStar, Inc.
Jonathan D. Rosenberg (jdrosen@skype.net) is with Skype

Manuscript received August 22, 2011; revised October 12, 2011; accepted November 8, 2011.

doi:10.4304/jcm.7.2.89-105

the initial simplicity of the protocol (text-based, request-response paradigm) attracted many adherents from both academia and industry.

In March 1999, SIP was published as an IETF standard, RFC 2543 [1]; it was further revised in June 2002 as RFC 3261 [2]. When RFC 2543 was issued, the Internet, in general, and voice over IP (VoIP), in particular, had made inroads into industry (VoIP had been a staple research subject in academic and industrial research laboratories for a long time. The advent of broadband infrastructure to leverage the Internet and the seed money pouring into Internet-based telephony through Wall Street simply created momentum such that it moved beyond the laboratory settings to the general marketplace). In this environment, SIP increasingly became viewed as the signaling protocol of choice for establishing VoIP sessions over the Internet. It had many advantages over its nearest rival, H.323 [10], the most important of which was its Internet roots. These roots were leveraged wisely; SIP is the only signaling protocol that can be easily extended to provide a wide variety of services. For instance, not only can SIP be used for establishing telephony sessions, it can also be used for other services such as transporting instant messages and presence. The addressing scheme in SIP and its extensibility have made it a success beyond telephony.

Today, SIP is well established in the PSTN telephony world, and it continues to adapt to provide services. Its use has been expanded from establishing multimedia multicast conferences to establishing telephony sessions, and it is now viewed as a protocol to tie together disparate services: telephony, instant messaging, and presence. In retrospect, there are certain constructs in the protocol that appear rather unorthodox to those not versed in its history. For instance, PSTN telephony adherents wonder why forking is specified in the SIP specification at the protocol handling level. After all, forking is nothing but the well-known "Find-Me" service on traditional telephony. Furthermore, parallel forking has undesirable consequences when used with multiple telephony gateways. For example, who's early media will the caller hear? The reason forking is specified, of course, is that SIPs roots are in the Internet, where forking does not typically have the undesirable side effects it exhibits in the PSTN. The protocol was designed as a rendezvous mechanism for the Internet, where it is more efficient to replicate and transmit the session setup request to multiple destinations, collate the responses, and present the best one to the caller.

The rest of this paper is structured as follows: Section II provides a background to SIP and RTP. As SIP has evolved from its research roots to a protocol used widely in service provider networks, it has had to address the operational aspects such as providing emergency services, logging, load balancing and overload control. Section III first provides a glimpse into how SIP is being used in service provider networks and Section IV then looks at the operational aspects of the protocol that are actively being addressed today. Section V explores the evolution of SIP in peer-to-peer networks. Finally, Section VI provides a conclusion by taking a critical and introspective look at where SIP succeeded and where it fell short of the expectations that were invested in the protocol.

II. Background: SIP and RTP

We now provide a general overview of SIP and RTP, the two important protocols used to establish a multimedia session. SIP is used at the signaling layer and RTP is the media-related protocol.

A. SIP

The Session Initiation Protocol is a signaling protocol used for establishing, modifying, and terminating multimedia sessions over IP networks. It is also sometimes called a *rendezvous* protocol, as it provides capabilities for discovery and location. It can be used over a number of different transports and in different architectures. SIP involves both a syntax to create well-formed and valid protocol data units (PDUs) and a state machine to govern the behaviour of the hosts involved in establishing, modifying and terminating a multimedia session.

SIP is designed based on the Hyper Text Transport Protocol (HTTP [8]), the underlying protocol of the World Wide Web. It borrows its text-based encoding and syntax from HTTP. It also uses Uniform Resource Identifiers (URI [11]) and Uniform Resource Locators (URL [11]) for addressing. SIP messages consist of a series of header fields and an optional message body. An example SIP message is shown in Figure 1 below.

SIP uses its own URI scheme which begins with sip:. It also supports a secure SIP URI scheme which begins with sips:. SIP and SIPS URIs have a similar form to an

```
INVITE sip:bob@example.com SIP/2.0
Via: SIP/2.0/TCP lab.example.com;branch=z9hG4bKn3as8hds787
Max-Forwards: 69
To: Bob <sip:bob@example.com>
From: Alice <sip:alice@example.net>;tag=9s2u830dw179740
Call-ID: 83x4bW4c6ek67p10
CSeq: 13431599 INVITE
Contact: <sip:alice@lab.example.net;transport=tcp>
Content-Type: application/sdp
Content-Length: 132

(Alice's SDP not shown)
```

Figure 1. Basic SIP Message

TABLE I.
SIP METHODS

Request purpose	Method	Reference
Establish a session with offer-answer	INVITE	RFC 3261 [2]
Acknowledge a response to an INVITE	ACK	RFC 3261 [2]
Query capabilities of a server or User Agent (UA)	OPTIONS	RFC 3261 [2]
Cancel a pending request	CANCEL	RFC 3261 [2]
Terminate an existing session	BYE	RFC 3261 [2]
Temporarily bind a device URI to an Address of Record (AoR)	REGISTER	RFC 3261 [2]
Establish a session to receive future information updates	SUBSCRIBE	RFC 3265 [12]
Deliver information after a SUBSCRIBE	NOTIFY	RFC 3265 [12]
Upload status information to a server	PUBLISH	RFC 3903 [13]
Request another UA to act upon an URI	REFER	RFC 3515 [14]
Transport an instant message	MESSAGE	RFC 3428 [15]
Update session state information	UPDATE	RFC 3262 [16]
Acknowledge a provisional response	PRACK	RFC 3262 [16]
Transport mid-call signaling information	INFO	RFC 6086 [17]

TABLE II.
SIP RESPONSES

Class	Response code	Examples
Informational or Provisional	1xx	100 Trying 180 Ringing 183 Session in Progress
Success	2xx	200 OK 202 Accepted
Redirection	3xx	300 Moved 302 Multiple Choices 305 Use Proxy
Client Error	4xx	401 Unauthorized 403 Forbidden 404 Not Found 415 Unsupported Media Type 486 Busy Here 428 Use Identity Header
Server Error	5xx	501 Not Implemented 503 Service Unavailable
Global Error	6xx	600 Busy Everywhere 603 Decline

email address, and typically have a user part and a host part. The host part can contain a domain name or an IPv4 or IPv6 host address.

SIP has been standardized to operate over a number of different transport layers including UDP (User Datagram Protocol), TCP (Transmission Control Protocol), and SCTP (Stream Control Transport Protocol). When used over an unreliable transport such as UDP, SIP has its own built-in reliability mechanisms and message retransmission timers. When used over a reliable transport such as TCP, these reliability mechanisms and timers are not used.

The set of SIP request types, the verbs of the protocol, are listed in Table I. Six methods are defined in the core SIP specification RFC 3261 [2] and eight others are defined in extension specifications. The most basic SIP method is INVITE which is used to setup or initiate a session. SIP responses use a three digit code, and are divided into six categories: Informational, Success, Redirection, Client Error, Server Error, and Global Error. Examples are shown in Table II. In addition to the response code which defines protocol behavior, there is also a reason phrase which is used for logging and display purposes.

An example SIP call flow, shown in Figure 2, establishes a session. An endpoint in SIP is known as a User Agent or UA. A UA wishing to establish a session creates an INVITE and sends it to another UA. One or more optional provisional responses can be sent, for example an *180 Ringing* to indicate that alerting is taking place. If the session request succeeds, a *200 OK* response is sent. To acknowledge receipt of the *200 OK* response, an ACK method request is sent. This completes the SIP three-way handshake that establishes a SIP session known as a dialog. Within this SIP exchange, there is also an offer/answer exchange of media capabilities, using Session Description Protocol (SDP [4]). The SIP usage for SDP is defined in RFC 3264 [12].

In addition to UAs, a SIP network can also contain a SIP proxy server. A proxy server does not originate requests, but acts in response to SIP requests from UAs. It provides routing and other services for requests, and in addition can enforce policy and facilitate NAT and firewall traversal. The call flow of Figure 3 shows a call between two UAs with two proxy servers involved. Each

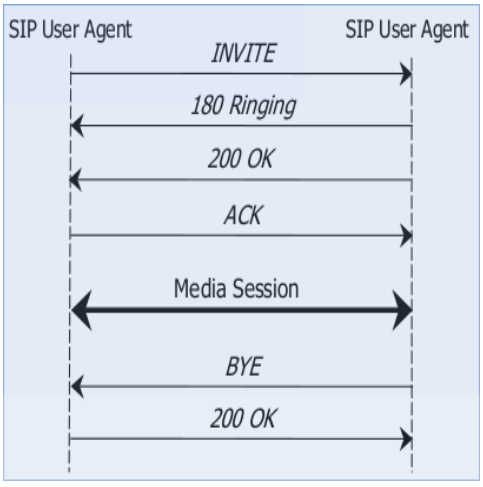

Figure 2. Basic SIP Call Flow

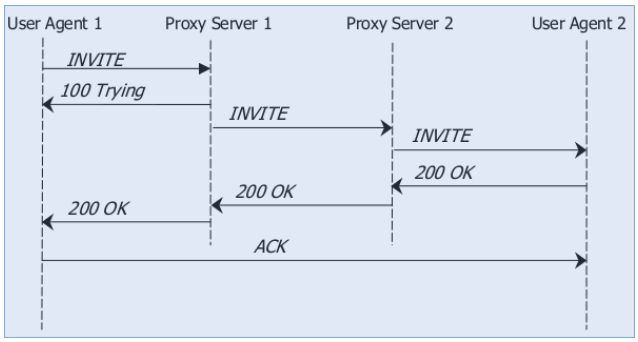

Figure 3. SIP Call Flow with Two Proxy Servers

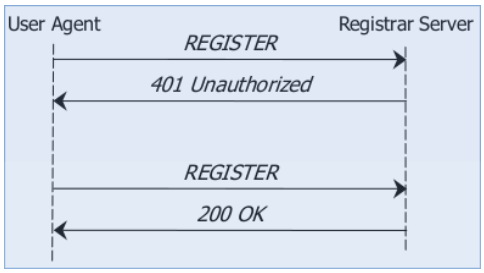

Figure 4. SIP Registration

proxy server sends an *100 Trying* response as soon as it receives an INVITE request. However, the *100 Trying* response is special in that it is a single-hop response, and is not forwarded by a proxy server. This prevents its retransmissions if an unreliable transport is used. Note that the responses to a request are routed back through the same set of proxies that forwarded the initial request. This includes provisional responses and final responses. The ACK for the *200 OK* does not need to route through the same set of proxies, but can go directly between the UAs as shown in Figure 3. SIP also provides a mechanism known as Record-Routing in which a proxy can request to stay in the path for the rest of the dialog, which includes the ACK all the way up to the BYE and its related *200 OK* response.

Routing databases used by proxy servers to route incoming requests are typically built by SIP registration. SIP registration consists of a UA sends a REGISTER request to a type of SIP server known as a registrar. There are two types of SIP URIs: an address of record (AOR) and a contact URI. An AOR represents a user or a service and is not tied to any particular device or address. A contact URI represents a particular device or host. For example, a user Alice may use the AOR sip:alice@example.com. She may have multiple devices, each represented by a Contact URI. Each of these devices can send a REGISTER request which creates a temporary binding between the AOR and the Contact URIs. That is, an incoming request (such as an INVITE) would be forwarded to the Contact URI. Registration responses contain an expiration time. The registration must be refreshed within this interval or the AOR to Contact URI binding is deleted from the database. This is shown in Figure 4, which also shows the use of a SIP authentication challenge.

When multiple Contact URIs are registered against an AOR, a proxy receiving a request has multiple routing choices. A proxy can apply policy about which Contact URI the request is forwarded to, or a proxy may choose to fork the request, i.e. send it to multiple devices. Sending all requests simultaneously to multiple devices is known as parallel forking (or, simultaneous ringing). Sending the requests in sequence, where the next request is not tried until the previous one succeeded, is called sequential forking.

SIP defines a usage of DNS in RFC 3263 [18]. This includes the usage of A (IPv4 Address), AAAA (IPv6 Address), SRV (Service Records), and NAPTR (Naming Authority Pointer Records). A and AAAA are simply used to resolve a host name to an IP address. SRV records are used to discover the hostnames of proxy servers for a given domain for a given transport. SRV records allow port number discovery, weighting, and prioritization of a number of proxies. NAPTR records are used to lookup supported transports for a given domain. Note that transport layer routing and discovery is done on a hop-by-hop basis in SIP, not an end-to-end basis. That is, different transport layer protocols can be used for different hops of the same request.

B. RTP

The Real-time Transport Protocol provides media transport. For VoIP and multimedia systems, RTP sessions are typically established using a signaling protocol such as SIP. RTP provides a container for media packets in real-time sessions for transport over IP networks. RTP typically uses UDP for transport, although a stream-oriented transport such as TCP can also be used. RTP is defined by RFC 3550 [6].

Standard audio and video profiles of RTP are defined in RFC 3551 [19]. The normal profile is know as RTP/AVP which stands for RTP Profile for Audio and Video Conferences with minimal control. The RTP/SAVP profile [20] is used for secure audio and video transported through Secure RTP (SRTP [20]). The RTP/AVPF audio-visual profile with feedback [21] enables RTP receivers to provide immediate feedback to receivers to enable short-term adaptation and efficient feedback-based repair mechanisms, while the RTP/SAVPF profile [22] performs an analogous role for secure RTP streams.

RTP uses a bit oriented header for efficiency in transport and provides a container for media including voice,

video, and real-time text. The use of RTP to transport tones and audio events such as DTMF is defined in RFC 4733 [23].

RTP streams are unidirectional and contain a single media type. Each RTP stream is usually received on a separate port. For example a bi-directional audio and video session is actually four distinct RTP sessions. RTP also has a control protocol defined, RTP Control Protocol (RTCP [6]). RTCP usually operates on a different port from RTP, often by default the next highest port from the RTP session that it controls.

RTCP is designed to be bandwidth efficient, and scale appropriately from two party unicast sessions to thousands of party multicast sessions. By discovering the number of participants in a session, RTCP ensures that control traffic never exceeds 5% of the total RTP media bandwidth, and that this bandwidth is equally divided among participants. RTCP uses basic Sender Reports (SR) and Receiver Reports (RR) to communicate basic information such as packets sent, packet loss, and jitter detected. RTCP is useful in detecting asymmetrical impairments that can arise on IP networks due to asymmetrical routing on IP networks. That is, one party may receive media without any impairments, but their sent packets may be suffering severe degradation. There are also RTCP Extended Reports (RTCP-XR [24]) that provide even more useful information about the quality of a real-time session.

The RTP header is shown in Figure 5. The main purpose of the RTP header is to provide all the information needed by the receiver of the RTP stream to play out the media packet correctly. In addition, the RTP header allows a receiver of RTP to become aware of any impairments caused by transport over an IP network. For example, it allows detection of:

- Packet loss (gaps in Sequence Number)
- Jitter or delay variation (Timestamps)
- Out of sequence delivery (Sequence Number)

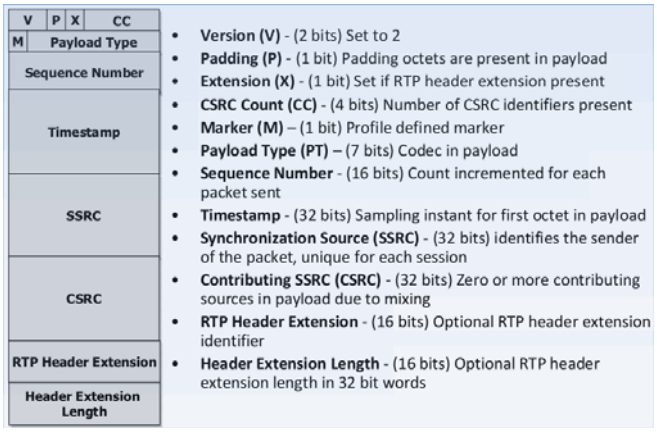

Figure 5. RTP Header

In addition, the RTP header allows for the determination of properties such as the codec used (Payload Type), the start of a new frame or talkspurt (Marker), and the identification of the source of media, including mixed media (SSRC and CSRC).

The RTP architecture can contain a number of elements besides just the sender and receiver of media. For example, it defines a mixer (combines media from a number of sources in a media specific manner), a translator (changes the payload type but does not change the SSRC), and a multipoint control unit (MCU).

RTP can carry real-time text, also known as conversational text or Text over IP (ToIP). This service is characterized by the character-by-character transport of text messages commonly implemented in Telecommunications Devices for the Deaf (TDD). Real-time text is transported as ITU-T T.140 over RTP, as defined in RFC 4103 [25].

III. SIP IN PRACTICE

Today there are multiple, distinct deployment markets for SIP: traditional telecommunications providers, over-the-top alternative providers, and enterprises. Even within these three broad markets there are sub-categories with unique use-cases; but in general the specific types of SIP capabilities and extensions used can be grouped and differentiated by those three market types. Below, we discuss the usage profiles of SIP grouped by these broad market categories. It should be pointed out that our taxonomy, though useful, still remains a generalization. There are hundreds of SIP product vendors, thousands of deployed SIP networks, and millions of SIP hosts in deployment today — enumerating the prevalent practices of all current deployments is impossible. Furthermore, the landscape is still changing, with more SIP extensions being deployed, sometimes years after standardization is complete.

A. Traditional telecommunications providers

Telecommunications providers do not typically deploy SIP all the way to the *end* (i.e., soft-clients on users computers), but rather convert to and from legacy PSTN-based technologies at the edge: in telephony gateways and multimedia terminal adoptors in the home (e.g., in the DSL modem), in mobile switching centers (MSC) for GSM/UMTS/CDMA to mobile handsets, and in PSTN and primary rate interface (PRI) gateways. Between providers (transit and wholesale) SIP is frequently used as well. It is increasingly being deployed between enterprises and providers as a replacement for private branch exchange (PBX) PRI trunks. Many provider deployments follow 3GPPs IMS specifications for deploying SIP in both wireline and wireless environments, though the majority do not.

From a SIP signaling perspective, most telecommunications providers still only support a relatively small subset of SIP extensions beyond the basic core standard RFCs. PRACK (RFC 3262 [16]), UPDATE (RFC 3311 [26]), and REFER (RFC 3515 [14]), for example, are not widely popular yet. GRUU (RFC 5627 [27]) support is rare, as is the support for the Join (RFC 3911 [28]) and Replaces mechanisms (RFC 3891 [29]) and SIP-Outbound (RFC 5626 [30]). Call forwarding information is still usually encoded using the Diversion header (RFC 5806 [31]) rather than History-Info (RFC 4244 [32]). P-Asserted-Identity (RFC 3325 [33]) is extremely popular for caller identification, while support for SIP-Identity (RFC 4474 [34]) is non-existent. The Path (RFC 3327 [35]) and Service-Route mechanisms (RFC 3608 [36]) are fairly popular, as is P-Associated-URI (RFC 3455 [37]). SIP over UDP without encryption is dominant, with a very small population using TLS, TCP or SCTP. There is virtually no SIP over IPv6 in actual deployment yet, and DNS resolution (RFC 3263 [18]) is still uncommon compared to static provisioning or simply using raw IP addresses.

From a media-plane perspective, traditional telecommunications providers almost universally support G.711 in both companding forms (a-law and μ-law), and some also support either G.729 or AMR/GSM. Fax is usually handled with T.38 over UDPTL [38], or native G.711-based T.30; DTMF is usually handled using RFC 4733 [23], or in signaling using the SIP INFO extension [39], and rarely using KPML (RFC 4730 [40]). SRTP support is still uncommon, and when performed it is only done across specific hops or links rather than end-to-end. The dominant keying scheme is security-descriptions (RFC 4568 [41]). Basic RTCP support is still not widely popular, with extensions such as RTCP-Feedback (RFC 4585 [21]) and RTCP-XR (RFC 3611 [24]) being very rare. There is virtually no support for STUN, TURN or ICE in traditional telecommunication equipment, and we posit that this situation is likely to persist for some time.

B. Over-the-top alternative providers

Unlike traditional telecommunications providers, over-the-top providers generally support SIP all the way to the *end*, namely to SIP soft-clients on computers or smartphones. A predominance of soft-clients with significant computing power and graphical human interfaces coupled with the providers typically being an alternate phone service from subscriber's perspective allows such providers to focus less on voice and more on other forms of communications (presence, instant messaging and video). The architectures of such providers differ from telecommunication providers, with less support for P-Asserted-Identity or P-Associated-URI [1] and Path or Service-Route [2]

Over-the-top providers do not own the IP network infrastructure and don't usually charge per call. This shapes the difference in their philosophy from traditional providers with respect to media handling. For example, about half of this market uses STUN [42] and ICE [43]. There is little support for RTCP, but some do support SRTP. From a codec perspective, G.711 is still universally available, but also iLBC [44] and Speex [45] are popular and preferred.

C. Enterprises

Enterprises generally remain oblivious about specific SIP extensions or standards; their requirements are driven more by user features and how these can be provided in their PBX-type system and phones. Most enterprises support similar characteristics to traditional telecommunications providers with some notable differences. Enterprises expect more support for REFER and Replaces, and less support for P-Asserted-Identity, P-Associated-URI, Path and Service-Route. Of the top five most popular enterprise SIP vendors in the world: only one supports STUN, TURN and ICE; a few support SIP over TLS as well as SRTP; one supports GRUU.

IV. Operational issues in SIP

As SIP gets ubiquitously deployed, operational issues arise to the forefront. In this section we survey three operational aspects of SIP: providing emergency services, a common log format for SIP, and traffic overload handling in SIP.

A. Providing emergency services in SIP

All calls are important, but some calls are more important that others. One of the most important kinds of calls are emergency calls. While traditionally, voice calls were the only form of emergency call, other media, and data is becoming increasingly common. Emergency call systems today are primarily based on very old telephone technology, but the beginnings of IP based emergency calling is emerging, and its based on SIP. We consider both ends of the call: the calling party and the emergency call center, commonly referred to as a Public Safety Answering Point (PSAP).

Emergency calling using SIP is defined by several IETF documents that are on the standards track. The overall

[1]This is partially due to the lack of billing in such systems and partially because these systems have their own authentication mechanisms through which the subscriber further authenticates himself or herself after launching the application.

[2]Most over-the-top providers have a flat architecture, thus obviating the need for the request to visit a series of application servers for service application.

framework for emergency calling is defined in Rosen et al. [46] and the umbrella standard for how emergency calls are placed with SIP is defined in Rosen and Polk [47]. In North America, the standard for the PSAP answering SIP based calls is defined in NENA 08-003 [48], from NENA, the North American Emergency Number Association. A similar effort is underway in EENA for the EU.

The dial string used to place an emergency call varies by location, and sometimes technology (112 from mobile phones, but 911 in North America, 999 in UK, etc.). To mark an emergency call independent of nation and technology, the Request-URI for an emergency call is always `urn:service:sos`. The route towards a PSAP would then be found in a Route header. Universal marking of emergency calls allows all intermediaries to recognize that the call is an emergency call and take appropriate action.

Figure 6 depicts the overall steps required to place an emergency call in SIP. Location is the key to getting an emergency call to the right PSAP, and for dispatching the proper responders. SIP has been enhanced to be able to carry location information in a header (Geolocation). The form of location used is encapsulated using Presence Information Data Format (PIDF), which includes a location object, PIDF-LO (RFC4119 [49]). PIDF-LO can carry either a civic (street) address or a geographical (latitude/longitude/altitude). Location can be passed by value (the PIDF is in the SIP body), or by reference (a URI is placed in the Geolocation header and the PIDF can be retrieved from the URI). The latter is often used for mobile devices where the location can change over time. The URI can be repeatedly dereferenced or a subscription (presence subscription) can be instantiated to track a moving caller.

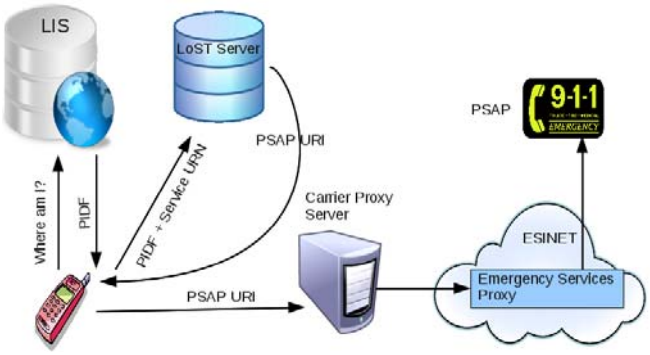

Figure 6. SIP Emergency Call Flow

With the location of the caller, a new location based routing protocol, LoST (Location to Service Translation Protocol, RFC5222 [50]), is used to choose a route. The client sends the location information and a service urn (normally `urn:service:sos`) and receives a URI in the response that is placed in the Route header. The LoST response also includes the local dial string. This allows a device or service to query a LoST server before an emergency call to find the local dial string, which can then be translated, when it is encountered, into the universal emergency call marking. Finally, the LoST response also includes a hint of when a new query to update the route may be appropriate. This allows mobile devices to move within some boundary knowing that the route is valid. If the device moves beyond the boundary, a new LoST query should be performed to (possibly) update the route, and receive an updated boundary.

Thus, an emergency call, as defined by the IETF (and NENA) standards is a normal SIP call, with a Geolocation header, a RequestURI of `urn:service:sos`, a Route header containing the URI obtained by a LoST query with the location of the caller, and normal call back information (a Contact header which leads to the device that placed the call, and a From header which contains the AoR of the caller).

IETF documents, following the normal IETF philosophy, describe emergency calling with the calling device doing most of the work, and the network simply routing where the Route header says (the Route header contains the LoST response). In many origination networks, devices are primitive and the network is intelligent; thus the IETF standards allow proxies to perform many of the functions the device would normally be expected to do if the device fully conformed to the IETF standards. This is more in line with the IMS philosophy, which has a dedicated server (an E-CSCF) that handles emergency calls. The E-CSCF would get an emergency call (IMS uses the `urn:service:sos` marking), obtain the location of the UE, query the LoST server, and send the call to the PSAP based on the LoST response, adding the Geolocation header.

Next generation emergency calling networks, such as those described by the NENA 08-003 standard [48], allow instant messages, video calls and other forms of emergency call to be accepted. This means that many kinds of service providers can send calls to what NENA calls an Emergency Services IP Network (ESInet), and not just traditional carriers, or even VoIP carriers. An instant messaging service, such as the AIM service can be used to send emergency messaging.

To accommodate these new providers, and also to provide accurate location for VoIP origination networks, video origination networks and similar services, a fundamental change is assumed in the architecture that currently supports emergency calling. In the current PSTN, there is an assumption that the origination network is also the access network. In a wireline telephone system, the company that provides the telephone service also provides the wire pair. In a traditional wireless network, the provider of the mobile telephony service is the provider of the radio network. However, even today, a VoIP service

provider may not be the broadband provider.

This implied connection is severed, making the access network and the origination network different entities. While the origination network provides the SIP signaling, it is the access network that provides location; for IP connected devices, this means the broadband network. The Internet service provider has to provide location. To do this, several location configuration protocols have been defined. For example, DHCP has been enhanced (RFC3825 [51], RFC4776 [52]) to pass location to the client. Also, a new protocol, HELD (RFC5985 [53]), has been defined as a layer 7 location configuration protocol that usually uses the IP address as the key to obtain location. Network specific protocols such as the cellular network's Secure User Plane standard (SUPL [54]), may also be used to get location.

We observe that the end device is the customer of both the access network and the origination network, but this does not require relationships to be created between the broadband network and the VoIP carrier or Internet service provider. Making Internet service providers responsible for location is a new model and likely will require regulation to become ubiquitous, however. In mobile networks, there is currently consternation over what is sometimes called over the top origination networks. If the mobile network has to provide location and access to a LoST server, that imposes new requirements on the network, and may alter the most efficient way emergency calls on the mobile operators services are handled.

B. Common log format for SIP

Servers executing on Internet hosts produce log records as part of their normal operations. A log record is, in essence, a summary of an application layer protocol data unit (PDU), that captures in precise terms an event that was processed by the server. These log records serve many purposes, including analysis and troubleshooting. Web servers such as Apache and Squid support event logging using a Common Log Format (CLF), the common structure for logging requests and responses serviced by the web server. One can argue that a good part of the success of Apache has been its CLF because it allowed third parties to produce tools that analyzed the log data and generated traffic reports and trends. The Apache CLF has been so successful that it become the de-facto standard in producing logging data for web servers. Today, one can configure many commercial web servers to produce logs in this format.

Until recent work [55] started in the IETF, SIP did not have a CLF. A typical deployment of SIP includes SIP entities from multiple vendors. Currently, if these entities are capable of producing a log file of the transactions being handled by them, the log files are in a proprietary format. The result of the multiplicity of log file formats is the inability of support staff to easily trace a call from one entity to another. Likewise, it makes it difficult to craft common tools that will perform trend analysis, debugging and troubleshooting across the SIP entities of multiple vendors. More importantly, a global view of the call, following a call trace through multiple devices from multiple vendors is very difficult to construct.

SIP CLF is being developed to address the above fundamental issues by enabling uniform output from all SIP servers, irrespective of the specific vendor of a SIP server. Although this new format builds on the success of the widely used HTTP CLF, it addresses the greater complexity of the SIP protocol (discussed below). The SIP CLF is now in the process of being standardized by the IETF [55], [56], and when implemented by the industry, it will:

- Establish a common reference for interpreting the state of SIP transactions across vendor implementations;
- Provide a semantic format of a SIP message to make it easier to train anomaly detection systems to trigger alarms;
- Enable development of innovative tools for trend analysis and traffic reports; and
- Provide a common diagnostic trail to aid in testing.

Establishing a CLF for SIP is a challenging task. The behavior of a SIP entity is more complex when compared to the equivalent HTTP entity. Unlike HTTP, where the origin server has all the information required to generate the HTTP CLF when it services a request, in SIP the information becomes available over time at a SIP server. Base protocol services such as parallel or serial forking elicit multiple final responses. Ensuing delays between sending a request and receiving a final response all add complexity when considering what fields should comprise a CLF and in what manner. Furthermore, unlike HTTP, SIP groups multiple discrete transactions into a dialog, and these transactions may arrive at a varying inter-arrival rate at a proxy. For example, the BYE transaction usually arrives much after the corresponding INVITE transaction was received, serviced and expunged from the transaction list. Nonetheless, it is advantageous to relate these transactions such that automata or a human monitoring the log file can construct a set consisting of related transactions.

ACK requests in SIP need careful consideration as well. In SIP, an ACK is a special method that is associated with an INVITE only. It does not require a response, and furthermore, if it is acknowledging a non-2xx response, then the ACK is considered part of the original INVITE transaction. If it is acknowledging a 2xx-class response, then the ACK is a separate transaction consisting of a request only (i.e., there is not a response for an ACK request.) CANCEL is another method that is tied to

an INVITE transaction, but unlike ACK, the CANCEL request elicits a final response. While most requests elicit a response immediately, the INVITE request in SIP can pend at a proxy as it forks branches downstream or at a user agent server while it alerts the user. RFC 3261 [2] require the server transaction to send a 1xx-class provisional response if a final response is delayed for more than 200 ms. A SIP CLF log file needs to include such provisional responses because they help train automata associated with anomaly detection systems and provide some positive feedback for a human observer monitoring the log file.

An example of a SIP CLF record is shown in Figure 7. A SIP CLF record consists of two lines, the first line is meta information that is stored to access the remaining CLF fields on the second line in an optimized manner. In a sense, the first line forms a table of contents, allowing a SIP CLF reader to quickly index into a specific field of interest, or to skip the entire SIP CLF record if it does not match a search criteria. This optimized search format is desirable because a SIP server could produce millions of log records over a period of weeks or months. Therefore the process of searching for a particular needle in such a haystack needs to be optimized for runtime complexity. Furthermore, the format in Figure 7 allows (human) system administrators to visually inspect the records being logged and use Linux text-based tools (`awk(1)`, `grep(1)`, `perl(1)` etc.) to operate directly on the file, if desired.

C. Overload handling in SIP

Overload occurs when SIP servers have insufficient resources to process the messages they receive. The resources required to process messages will differ, for instance, a SIP proxy generally does not deal with media. Thus overload for the proxy implies lack of resources — socket descriptors, memory to store ongoing transactions, CPU occupancy, etc. — required to handle the signaling messages. A SIP user agent, including a dense-port PSTN gateway, views overload differently. Such user agents are generally limited by the resources required to process media more than those required to process signaling. For instance, a PSTN gateway may have adequate capacity to handle signaling, but may not be able to process any more media streams; a voice mail server may be able to handle signaling messages but may not have the required disk space to store any more messages.

The traditional manner to combat overload in SIP has been to send a *503 Service Unavailable* response code to the sender of the request. The response code may be qualified with a Retry-After header, which contains the number of seconds that the overloaded server is expected to remain in this state and incapable of receiving new requests. This simple arrangement works only in the case that the overloaded server is handling signaling-only traffic (i.e., no media) and the overloaded server is being accessed by a large number of clients, each of which sends a small amount of traffic. However, the 503 is of no aid when the overloaded server has exhausted media resources or when the overloaded server is receiving signaling from a few upstream clients. Furthermore, the specific instances under which a server should send 503 remain ambiguous and the 503 may not even reach in its current form to the endpoint that is generating excessive load because intermediary proxies may change the response code to a generic *500 Server Internal Error* instead [2]. Current literature [57]–[59] has shown that using a 503 to report overload condition only serves to exacerbate the problem due to load oscillation and is unable to prevent congestion collapse of a SIP server.

Current research indicates that a feedback based system, wherein the overloaded server only receives as much traffic as it can handle, may be more appropriate for SIP [59]–[62]. Accordingly, there is work proceeding in the IETF to standardize a SIP overload control scheme based on explicit feedback [63], [64]. Explicit overload control mechanisms can be differentiated based on the type of information conveyed in the overload control feedback. *Rate-based* overload control schemes impose a strict limit in terms of requests per second on the upstream clients (i.e., the user agents sending requests to the overloaded server) [61]. Under such a scheme, a server assigns a share of its overall capacity to each upstream neighbor, and the neighbors promise not to exceed this capacity. *Loss-based* overload control schemes monitor overall system utilization and if it exceeds a certain threshold, they dynamically calculate a probability that represents the proportion of requests they are willing to accept from upstream clients [59]. *Window-based* overload control schemes allow a sender to transmit a certain number of messages before it needs to receive a confirmation for the messages sent [60], [65].

Simulation studies performed by the respective authors on their schemes indicate that the relative performance of all schemes is equal. Rate-based schemes work well in managed networks where the set of upstream clients sending requests to the overloaded server are bounded and rather static. Loss-based schemes perform well when the set of upstream clients sending traffic is unbounded and frequently changing.

V. Peer-to-Peer SIP

So far, we have looked at canonical VoIP systems, which in some sense, emulate the functionality of PSTN systems over IP. SIP is a classic example of a client-server VoIP system, where user agents (phones) communicate with SIP servers (proxies and registrars) to establish a media session. The goal of peer-to-peer SIP is to distribute

```
A0000FC,Rou,0051005A005C006B007B008D009C009E00B800C500E900F30000
0000000000.010  1 INVITE - sip:192.0.2.10 192.0.2.10:5060 192.0.2.200:56485 sip:192.0.2.10 – sip:1001@example.com DL88360fa5fc DL70dff590c1 - 1079051554@example.com u76yhj-00
```

Figure 7. A SIP CLF Record

the functionality of media session establishment to the user agents. In Section V-A, we provide the background and history of P2PSIP evolution. In Section V-B, we describe the RELOAD protocol and its key features. The RELOAD protocol can be used as a substrate to build a fully distributed p2p communication system. In Section V-C, we discuss the limitations of the RELOAD protocol.

A. Background

Skype [66], a VoIP application released in 2003, successfully pioneered a hybrid architecture for building VoIP systems. Like traditional client-server VoIP systems, Skype uses managed servers for handling a unique user name space, signed user identities, voicemail, and calling to PSTN or mobile phones. Unlike clilent-server VoIP systems, Skype distributes the functionality of locating and calling Skype user agents (SIP registrar and proxy in client-server SIP), and the use of RTP relays and SIP proxies for traversing NATs and firewalls from managed servers to Skype user agents with unrestricted connectivity.

While a SIP server running on a modern hardware can easily handle a SIP signaling workload of hundreds of thousands of SIP users [67], [68], the use of SIP proxies and media relays for addressing connectivity issues due to the presence of restrictive NATs and firewalls is costly in terms of number of servers, bandwidth, and the administrative costs. Consequently, Skype distributes the functionality of bypassing through NATs and firewalls to Skype user agents with unrestricted connectivity. A side benefit of user agents providing the relay and proxy functionality is that Skype is difficult to block using only IP or port filtering.

Singh [69] and Bryan [70] showed how to distribute the functionality of SIP registrar to the SIP user agents. Baset [71] built the first open source P2P communication system that distributed the functionality of SIP registrar and RTP relays to the user agents and demonstrated practical issues in building P2P VoIP systems.

Motivated by the success of Skype, and the development of initial prototypes to distribute the SIP functionalities, discussions were started in IETF to standardize a P2P version of the SIP protocol, aptly named as P2PSIP. The P2PSIP working group, established in 2007, discussed several proposals for a P2PSIP protocol. The three promising and competing proposals, namely, Address Settlement by Peer-to-Peer (ASP) [72], Peer-to-Peer Protocol (P2PP) [73] and Resource Location and Discovery (RELOAD) [74] were merged into one proposal. The working group continued the use of RELOAD as the name of the merged protocol. The present version of the merged RELOAD protocol [75] has significantly evolved from the original proposals, although it retains their essence.

B. Resource Location and Discovery (RELOAD)

RELOAD [75] is a flexible application layer binary protocol in which user agents collaborate to provide a distributed messaging and storage service by forming an overlay network, even if the nodes are behind restrictive NATs and firewalls. This distributed functionality provided by RELOAD can be used to distribute the functionality of SIP registrar and RTP relay to the user agents. RELOAD is flexible and it potentially allows user agents to form an overlay network using any 'overlay algorithms' such as Chord [76], and Kademlia [77].

Below, we describe the key features of RELOAD protocol.

1) Data Model: RELOAD defines a data model which allows user agents to provide a flexible distributed storage service for storing data objects. Each data object stored by nodes is defined by a variable length unique 'resource-ID' and a value. The value of the object is defined by a 'kind'. A kind is anolagous to the data type in strongly typed languages. The value of the data being stored can be a single value, array or dictionary. Each kind is associated with an access control policy. Figure 8 shows the data model of RELOAD.

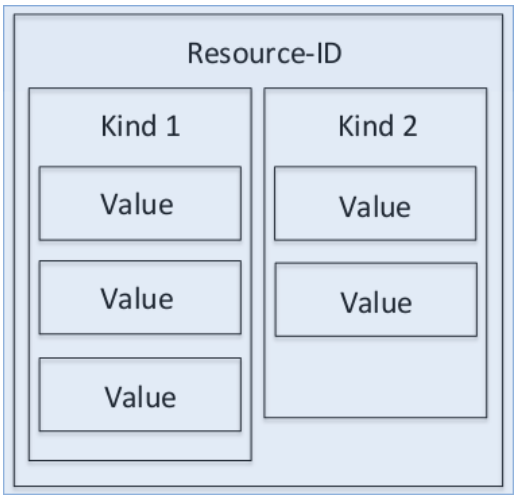

Figure 8. RELOAD data model

2) Usages: Applications can take advantage of RELOAD's flexible data model to define the kinds stored in the overlay. The definition of application specific kinds along with their access policies constitutes a usage of RELOAD. For example, the SIP usage of RELOAD [78] defines the kinds for storing a SIP registration in the overlay.

3) NAT and Firewall Traversal: Perhaps the main reason for designing a peer-to-peer SIP protocol such as RELOAD is addressed to the connectivity issues due to the prevalence of NATs and firewalls. Unlike many filesharing P2P protocols such as Bittorrent [79], NAT and firewall traversal is a first class concept in RELOAD. The reason for this emphasis is that RELOAD is designed to work in environments where many if not most nodes may be behind NATs or firewalls.

4) Pluggable Overlay Algorithms: During the standardization of a P2PSIP protocol, it was realized early on that one overlay algorithm may not be suitable for all environments. For instance, in a small office environment, a broadcast-like overlay algorithm may be more suitable for quickly discovering other devices and establishing the connectivity, whereas in the Internet, a more scalable protocol is needed. RELOAD achieves the pluggability of overlay algorithms by defining a subset of messages (namely, `Join`, `Leave`, and `Update`) that each overlay algorithm must implement. To maintain interoperability, RELOAD uses Chord [76] as the default overlay algorithm.

5) Security: RELOAD provides three levels of security. First, it defines the concept of enrollment server which allows users and their devices to obtain signed certificates from a trusted authority. In the absence of such a trusted authority, the user agents can use self signed certificates. The use of an enrollment server and signed certificates mitigates the risk of Sybil attacks [80]. Second, each user agent storing data in the overlay must cryptographically sign it. This mechanism provides protection against data tampering by malicious nodes. Third, RELOAD provides message confidentiality by using TLS [81] and DTLS [82] as the reliable and unreliable secure message transports, respectively.

6) Message Routing: RELOAD allows nodes to send messages in a recursive manner by default or an iterative manner. In the recursive routing, a message traverses multiple nodes until it reaches the destination. The response from destination then follows the path back to the message originator. In iterative routing, the message originator finds the ultimate destination of message iteratively, and then directly sends the message to that node. There are pros and cons of using each routing mechanism. Recursive routing is useful when many of the intermediate nodes in the message path may be behind NATs and firewalls; however, this mechanism requires state to be maintained within message (using Via lists) or nodes. In iterative routing, the request originator has control over the message path, but it may require the establishment of a secure connection through NATs and firewalls with intermediate hops.

C. Limitations and Complexity

RELOAD does not specify a service discovery mechanism and only defines a rudimentary mechanism for discovering a node providing a relay service. This mechanism needs to be evaluated in the Internet.

RELOAD is designed to work in many different environments. The complexity of a RELOAD implementation is exacerbated by the number of available options. For example, RELOAD defines the use of public key and self signed certificates, recursive and iterative routing, replication policies, storage permissions, and extensive NAT and firewall traversal mechanisms. An implementation may only require a subset of features provided by RELOAD. It is quite conceivable that different deployment environments will qualify the use of RELOAD features and will not make use of all of them. Consequently, it may be useful and ultimately necessary to define a deployment specific version of RELOAD. However, such a path may impact the interoperability of RELOAD.

VI. SIP: An Introspective

With its origins dating back to 1995, SIP has a long history. That history includes both successes and failures, and with those come lessons for the future.

Certainly, the successes have been many. If one measures success by the number of products which implement a technology, then SIP has been a resounding success. SIP has been implemented in hundreds (if not thousands) of different products. It has been implemented in phones, ranging from IP hardphones to soft clients to telephony adapters. It has been implemented in PSTN gateways, from single port analog gateways to massive, carrier-grade SS7 gateways. It is part of many enterprise PBX products, from small scale small-medium-enterprise solutions to large multinational IP PBXs. It is a feature on nearly every carrier softswitch. Most firewalls have a SIP module. Indeed, SIP has given rise to entirely new product categories. Session Border Controllers (SBC) were born of industry needs around inter-domain SIP deployments.

Even more important than products are deployments, and here too SIP has seen many. SIP has been deployed by dozens, if not hundreds, of service providers, and runs within countless enterprises. Billions of minutes of traffic are carried on SIP. It has been deployed for a variety of purposes. Application providers have deployed it to power consumer telephony services; existing PSTN providers have deployed it internally for backhaul, carrying traffic between their own sites, or for peering with each other. Recently, service providers have deployed SIP trunking

services, aimed at replacing traditional T1 circuits for enterprise telephony access. Skype uses SIP for its Skypeout and SkypeIn services. SIP, without a doubt, has become the lingua-franca for interconnect between providers and between equipment within a network.

SIP has also seen its share of successes in terms of industry buzz. For a while, it had its own magazine. Countless trade articles, analyst reports and press interviews were published about SIP, with carriers and vendors alike lauding it and declaring it as the technology of the future. Other standards bodies have embraced SIP. Most notable amongst them is 3GPP (the Third Generation Partnership Project), which defines the standards for wireless operators. In a pivotal moment in 2001, 3GPP selected SIP as the technology that would power all-IP based communications for wireless operators. Though this has yet to materialize in practice, the selection of SIP solidified its role as the industry technology for the future.

Yet, with all of these successes, SIP has had its failures. Two of them in particular need to be called out — its failure to create innovation in telecommunication services, and its failure to facilitate interoperability.

In the initial stages of the development of SIP, all of the participants contributing to it were aligned around a single purpose — to re-imagine telecommunications as an Internet service, and to breathe innovation and new life into a technology (the PSTN) which had been stagnant for a long time. The SIP specification itself included novel ideas that were not possible on the traditional phone network: one could include multimedia content in call setup messages, like vCards or images of the caller, and render those to the callee. This would bring the traditional caller ID service, which simply provided the calling party's name into the new world by enhancing it with multimedia content. SIP also integrated well with the World Wide Web, so that one could reject a call and display a webpage listing other options to the callee, including a link for sending email instead. SIP supported video calling as well as audio calling, and indeed could setup and manage any type of session content— games for example. A demonstration was put together early on in SIP's lifecycle, where a Doom multiplayer game was set up using the protocol. SIP was quickly extended (with work starting in 2001) to include presence and instant messaging, which were just beginning to show up in proprietary desktop applications.

Yet, despite all of the activity around standards that would enable innovative communications experiences, deployments that used these new services did not follow. The vast majority of deployments and SIP products were focused on pure PSTN replacement services. SIP vCards never got deployed. Video eventually got deployed, but only years later. Redirecting to web pages or email were never deployed. SIP-based presence and instant messaging saw deployment, but was limited to a handful of (notable) vendors and enterprise deployments. Overall, it remains arguable if SIP can be tagged as the catalyst for providing the end users a communications experience that is better than the one they receive from the traditional telephone network.

The focus of most deployments has been on *cheaper* and not *better*. While there are varied reasons for this, some long and complicated, we believe that in the end, three rise to prominence. The first is the lowest-common denominator effect caused by standardization and interoperability. SIP was built on the idea of standards and interoperability, with a particular focus on inter-domain and inter-server signaling. In order for a new communications service to get deployed amongst a set of users, this service needs to be deployed by all of the providers servicing those users, and then supported by their clients and the servers that they communicate with. In the traditional model where telecommunication providers service regional user bases, global deployment of a service would require all of the providers in the world to support it. This is possible only through rigorous standardization, and it is a very slow process that does not encourage change and innovation. Even looking within a single service provider, in order to offer a new service to all of their users, all of their clients need to be upgraded, along with all of the servers which support that service. If the service provider had purchased clients and servers from different vendors (all of which interoperate through standards), the service provider will need to wait for its vendors to implement the new feature, and then roll it out once it was ready. Of course, the vendors are loathe to implement a new feature until there is a demand from numerous providers, and even then, until there is a standard that is stable enough to agree upon. The result is a chain of dependencies. Deploying a new service requires standardization to complete, followed by vendor implementation, followed by service provider purchasing, followed by deployment. The slow pace of this process means that features and functionality quickly get locked into the lowest common denominator — the few things which work universally across many different products and service providers. For SIP — this was basic narrowband voice. With the lowest common denominator in place, innovation was stalled.

The second problem is that the customers of SIP were ultimately the traditional telecommunication providers. However, they were not interested in deploying VoIP products that disrupted their own PSTN services. Instead, they were interested in cost reduction and long term evolution. Indeed, due to the slow innovation cycle of traditional telecoms, any plans around innovative services were fundamentally long term. As a result, innovation didn't materialize.

SIP's final failure, ironically, was interoperability. Interoperability can be classified broadly in three different

scenarios: (1) client to server, (2) server to server within a single domain, and (3) inter-domain server-to-server. For the last two cases, SIP has done reasonably well. Basic calling works broadly and has been deployed inter-domain on a wide-scale. Because of the challenges of inter-domain innovation, there was not a lot of demand for anything more than basic voice, and so basic voice evolved to work well. Server to server interoperability within a single domain has been somewhat successful, but challenged. Again, basic voice has worked well, but many providers have run into challenges achieving interoperability with more advanced features. Finally, in the area of client-to-server interoperability, SIP has largely been a failure. There are major gaps in the standards, requiring most vendors to build countless proprietary enhancements to achieve their goals. Configuration and device provisioning were either not specified or underspecified; advanced features like attendant line handling were similarly lacking or SIP provided multiple ways in accomplishing these to the detriment of interoperability. Presence and instant messaging have proved harder to deploy; in particular, it was difficult to separate INVITE/REGISTER as triggers to presence. Furthermore, there was a need to implement a new set of protocols (XCAP [83]) to realize presence on existing systems. All of this raised the implementation barrier to adding presence to existing VoIP systems. The existence of the IETF BLISS[3] (Basic Level of Interoperability of SIP Services) working group stands as a testament to these challenges; BLISS was chartered to facilitate effective feature interoperability. Clearly, SIP has not achieved *plug-n-play* interoperability between a phone and its SIP server.

Why is this? There are ultimately three reasons for it. The first reason is that SIP, as a protocol, became complicated very fast. Because every feature needed standardization to interoperate, standards were produced for all of the needs that anyone had. Literally hundreds of documents were produced. The complexity ran sufficiently high that documents needed to be developed just to explain how to read the rest of the documents. The "Hitchhikers Guide to SIP", RFC5411 [84], has 141 references, and those are just a subset of what a vendor needs to implement. With this escalating complexity, different vendors would implement different subsets of specifications, and of course implement things incorrectly at times, resulting in interoperability failures.

The second reason for interoperability failure was the widespread implementation of proprietary extensions to SIP. Often for expediency, usually for time to market, and sometimes for competitive advantage, vendors implemented features in proprietary ways. SIP made it easy to do this, and the practice became rampant.

The final failure was market demand. Ultimately, vendors respond to the needs of their customers and the market as a whole. Had the market demanded interoperability for features beyond basic voice, such interoperability would have been delivered. But, the demand for new features was not very high, and so they never materialized beyond existing as proof of concepts in academic and industrial research laboratories.

In summary, SIP — like all technologies — has had it successes and failures. It has been an unquestionable success in becoming the lingua-franca for voice calling between vendor products and between domains. It evolved from being a basic rendezvous protocol for establishing up sessions on the MBONE to being the protocol of choice for an all IP-communication network. However, it ultimately fell short on delivering its promises of innovation, or its goals of interoperability for functionality beyond basic voice. We appear now to be at an inflection point for video services. Ten years ago, the technology and networks were simply not ready for large-scale consumer and business deployment of video services. Thus, while setting up video sessions through SIP has always been possible, and indeed, attracted early adopters, it was not able to make a push into enterprise or consumer markets. Of course, much lies ahead, and we may yet see the protocol evolve to support needs and services we have not imagined yet.

[3]See http://datatracker.ietf.org/wg/bliss/charter/

ACKNOWLEDGEMENTS

The authors gratefully acknowledge Henning Schulzrinne for his insights and a close reading of the manuscript. In addition, the authors thank the anonymous referees for their comments that helped improving the quality of the manuscript.

REFERENCES

[1] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: Session Initiation Protocol," RFC 2543 (Proposed Standard), Internet Engineering Task Force, Mar. 1999, obsoleted by RFCs 3261, 3262, 3263, 3264, 3265. [Online]. Available: http://www.ietf.org/rfc/rfc2543.txt

[2] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," RFC 3261 (Proposed Standard), Internet Engineering Task Force, June 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3261.txt

[3] M. Handley, C. Perkins, and E. Whelan, "Session Announcement Protocol," RFC 2974 (Experimental), Internet Engineering Task Force, Oct. 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2974.txt

[4] M. Handley, V. Jacobson, and C. Perkins, "SDP: Session Description Protocol," RFC 4566 (Proposed Standard), Internet Engineering Task Force, July 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4566.txt

[5] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," RFC 2326 (Proposed Standard), Internet Engineering Task Force, Apr. 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2326.txt

[6] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550 (Standard), Internet Engineering Task Force, July 2003, updated by RFCs 5506, 5761, 6051, 6222. [Online]. Available: http://www.ietf.org/rfc/rfc3550.txt

[7] Multiparty Multimedia Session Control Working Group (MMUSIC), Proceedings of the 35th IETF Meeting, March 1996, [Online; accessed August 2011], http://www.ietf.org/proceedings/96mar/area.and.wg.reports/tsv/mmusic/mmusic-minutes-96mar.html.

[8] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616 (Draft Standard), Internet Engineering Task Force, June 1999, updated by RFCs 2817, 5785, 6266. [Online]. Available: http://www.ietf.org/rfc/rfc2616.txt

[9] Multiparty Multimedia Session Control Working Group (MMUSIC), Proceedings of the 37th IETF Meeting, December 1996, [Online; accessed August 2011], http://www.ietf.org/ietf/mmusic/mmusic-minutes-96dec.txt.

[10] "International Telecommunications Union, Telecommunications Standardization Sector: Packet-based Multimedia Communications Systems," ITU-T H.323, June 2006.

[11] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396 (Draft Standard), Internet Engineering Task Force, Aug. 1998, obsoleted by RFC 3986, updated by RFC 2732. [Online]. Available: http://www.ietf.org/rfc/rfc2396.txt

[12] J. Rosenberg and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)," RFC 3264 (Proposed Standard), Internet Engineering Task Force, June 2002, updated by RFC 6157. [Online]. Available: http://www.ietf.org/rfc/rfc3264.txt

[13] A. Niemi, "Session Initiation Protocol (SIP) Extension for Event State Publication," RFC 3903 (Proposed Standard), Internet Engineering Task Force, Oct. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3903.txt

[14] R. Sparks, "The Session Initiation Protocol (SIP) Refer Method," RFC 3515 (Proposed Standard), Internet Engineering Task Force, Apr. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3515.txt

[15] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging," RFC 3428 (Proposed Standard), Internet Engineering Task Force, Dec. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3428.txt

[16] J. Rosenberg and H. Schulzrinne, "Reliability of Provisional Responses in Session Initiation Protocol (SIP)," RFC 3262 (Proposed Standard), Internet Engineering Task Force, June 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3262.txt

[17] C. Holmberg, E. Burger, and H. Kaplan, "Session Initiation Protocol (SIP) INFO Method and Package Framework," RFC 6086 (Proposed Standard), Internet Engineering Task Force, Jan. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6086.txt

[18] J. Rosenberg and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers," RFC 3263 (Proposed Standard), Internet Engineering Task Force, June 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3263.txt

[19] H. Schulzrinne and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control," RFC 3551 (Standard), Internet Engineering Task Force, July 2003, updated by RFC 5761. [Online]. Available: http://www.ietf.org/rfc/rfc3551.txt

[20] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)," RFC 3711 (Proposed Standard), Internet Engineering Task Force, Mar. 2004, updated by RFC 5506. [Online]. Available: http://www.ietf.org/rfc/rfc3711.txt

[21] J. Ott, S. Wenger, N. Sato, C. Burmeister, and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)," RFC 4585 (Proposed Standard), Internet Engineering Task Force, July 2006, updated by RFC 5506. [Online]. Available: http://www.ietf.org/rfc/rfc4585.txt

[22] J. Ott and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)," RFC 5124 (Proposed Standard), Internet Engineering Task Force, Feb. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5124.txt

[23] H. Schulzrinne and T. Taylor, "RTP Payload for DTMF Digits, Telephony Tones, and Telephony Signals," RFC 4733 (Proposed Standard), Internet Engineering Task Force, Dec. 2006, updated by RFCs 4734, 5244. [Online]. Available: http://www.ietf.org/rfc/rfc4733.txt

[24] T. Friedman, R. Caceres, and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)," RFC 3611 (Proposed Standard), Internet Engineering Task Force, Nov. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3611.txt

[25] G. Hellstrom and P. Jones, "RTP Payload for Text Conversation," RFC 4103 (Proposed Standard), Internet Engineering Task Force, June 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4103.txt

[26] J. Rosenberg, "The Session Initiation Protocol (SIP) UPDATE Method," RFC 3311 (Proposed Standard), Internet Engineering Task Force, Oct. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3311.txt

[27] ——, "Obtaining and Using Globally Routable User Agent URIs (GRUUs) in the Session Initiation Protocol (SIP)," RFC 5627 (Proposed Standard), Internet Engineering Task Force, Oct. 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5627.txt

[28] R. Mahy and D. Petrie, "The Session Initiation Protocol (SIP) "Join" Header," RFC 3911 (Proposed Standard), Internet Engineering Task Force, Oct. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3911.txt

[29] R. Mahy, B. Biggs, and R. Dean, "The Session Initiation Protocol (SIP) "Replaces" Header," RFC 3891 (Proposed Standard), Internet Engineering Task Force, Sept. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3891.txt

[30] C. Jennings, R. Mahy, and F. Audet, "Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)," RFC 5626 (Proposed Standard), Internet Engineering Task Force, Oct. 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5626.txt

[31] S. Levy and M. Mohali, "Diversion Indication in SIP," RFC 5806 (Historic), Internet Engineering Task Force, Mar. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5806.txt

[32] M. Barnes, "An Extension to the Session Initiation Protocol (SIP) for Request History Information," RFC 4244 (Proposed Standard), Internet Engineering Task Force, Nov. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4244.txt

[33] C. Jennings, J. Peterson, and M. Watson, "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks," RFC

3325 (Informational), Internet Engineering Task Force, Nov. 2002, updated by RFC 5876. [Online]. Available: http://www.ietf.org/rfc/rfc3325.txt

[34] J. Peterson and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)," RFC 4474 (Proposed Standard), Internet Engineering Task Force, Aug. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4474.txt

[35] D. Willis and B. Hoeneisen, "Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts," RFC 3327 (Proposed Standard), Internet Engineering Task Force, Dec. 2002, updated by RFC 5626. [Online]. Available: http://www.ietf.org/rfc/rfc3327.txt

[36] ——, "Session Initiation Protocol (SIP) Extension Header Field for Service Route Discovery During Registration," RFC 3608 (Proposed Standard), Internet Engineering Task Force, Oct. 2003, updated by RFC 5630. [Online]. Available: http://www.ietf.org/rfc/rfc3608.txt

[37] M. Garcia-Martin, E. Henrikson, and D. Mills, "Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP)," RFC 3455 (Informational), Internet Engineering Task Force, Jan. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3455.txt

[38] ITU-T, "Procedures for real-time group 3 facsimile communication over ip networks," ITU-T, T.38, Sep 2010.

[39] S. Donovan, "The SIP INFO Method," RFC 2976 (Proposed Standard), Internet Engineering Task Force, Oct. 2000, obsoleted by RFC 6086. [Online]. Available: http://www.ietf.org/rfc/rfc2976.txt

[40] E. Burger and M. Dolly, "A Session Initiation Protocol (SIP) Event Package for Key Press Stimulus (KPML)," RFC 4730 (Proposed Standard), Internet Engineering Task Force, Nov. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4730.txt

[41] F. Andreasen, M. Baugher, and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams," RFC 4568 (Proposed Standard), Internet Engineering Task Force, July 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4568.txt

[42] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing, "Session Traversal Utilities for NAT (STUN)," RFC 5389 (Proposed Standard), Internet Engineering Task Force, Oct. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5389.txt

[43] J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols," RFC 5245 (Proposed Standard), Internet Engineering Task Force, Apr. 2010, updated by RFC 6336. [Online]. Available: http://www.ietf.org/rfc/rfc5245.txt

[44] S. Andersen, A. Duric, H. Astrom, R. Hagen, W. Kleijn, and J. Linden, "Internet Low Bit Rate Codec (iLBC)," RFC 3951 (Experimental), Internet Engineering Task Force, Dec. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3951.txt

[45] G. Herlein, J. Valin, A. Heggestad, and A. Moizard, "RTP Payload Format for the Speex Codec," RFC 5574 (Proposed Standard), Internet Engineering Task Force, June 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5574.txt

[46] B. Rosen, H. Schulzrinne, J. Polk, and A. Newton, "Framework for Emergency Calling Using Internet Multimedia," IETF Internet-Draft (work-in-progress), draft-ietf-ecrit-framework-12, October 2010.

[47] B. Rosen and J. Polk, "Best Current Practice for Communications Services in support of Emergency Calling," IETF Internet-Draft (work-in-progress), draft-ietf-ecrit-phonebcp-18, July 2011.

[48] NENA VoIP/Packet Technical Committee, http://www.nena.org/standards/technical/i3-solution, Jun 2011, [Online; accessed September 26, 2011].

[49] J. Peterson, "A Presence-based GEOPRIV Location Object Format," RFC 4119 (Proposed Standard), Internet Engineering Task Force, Dec. 2005, updated by RFCs 5139, 5491. [Online]. Available: http://www.ietf.org/rfc/rfc4119.txt

[50] T. Hardie, A. Newton, H. Schulzrinne, and H. Tschofenig, "LoST: A Location-to-Service Translation Protocol," RFC 5222 (Proposed Standard), Internet Engineering Task Force, Aug. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5222.txt

[51] J. Polk, J. Schnizlein, and M. Linsner, "Dynamic Host Configuration Protocol Option for Coordinate-based Location Configuration Information," RFC 3825 (Proposed Standard), Internet Engineering Task Force, July 2004, obsoleted by RFC 6225. [Online]. Available: http://www.ietf.org/rfc/rfc3825.txt

[52] H. Schulzrinne, "Dynamic Host Configuration Protocol (DHCPv4 and DHCPv6) Option for Civic Addresses Configuration Information," RFC 4776 (Proposed Standard), Internet Engineering Task Force, Nov. 2006, updated by RFC 5774. [Online]. Available: http://www.ietf.org/rfc/rfc4776.txt

[53] M. Barnes, "HTTP-Enabled Location Delivery (HELD)," RFC 5985 (Proposed Standard), Internet Engineering Task Force, Sept. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5985.txt

[54] OMA, "Secure user plane location (SUPL) 1.0," SUPL Enabler 1.0, January 2006.

[55] V. K. Gurbani, E. Burger, T. Anjali, H. Abdelnur, and O. Festor, "The Common Log Format (CLF) for the Session Initiation Protocol (SIP)," IETF Internet-Draft (work-in-progress), draft-ietf-sipclf-problem-statement-07, June 2011.

[56] G. Salgueiro, V. K. Gurbani, and A. Roach, "Format for the Session Initiation Protocol (SIP) Common Log Format (CLF)," IETF Internet-Draft (work-in-progress), draft-ietf-sipclf-format-01, March 2011.

[57] E. Noel and C. R. Johnson, "Novel overload controls for SIP networks," in *Proceedings of 21st International Teletraffic Congress*, September 2009, pp. 1–8.

[58] J. Rosenberg, "Requirements for Management of Overload in the Session Initiation Protocol," RFC 5390 (Informational), Internet Engineering Task Force, Dec. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5390.txt

[59] V. Hilt and I. Widjaja, "Controlling overload in networks of SIP servers," in *IEEE International Conference on Network Protocols*, October 2008.

[60] C. Shen, H. Schulzrinne, and E. Nahum, "Session initiation protocol (SIP) server overload control: Design and evaluation." Berlin, Heidelberg: Springer-Verlag, 2008, pp. 149–173.

[61] E. Noel and C. Johnson, "Initial simulation results that analyze sip based voip networks under overload," in *International Teletraffic Congress*, June 2007.

[62] A. Abdelal and W. Matragi, "Signal-based overload control for sip servers," in *Consumer Communications and Net-*

working Conference (CCNC), 2010 7th IEEE, jan. 2010, pp. 1 –7.

[63] V. Hilt, E. Noel, C. Shen, and A. Abdelal, "Design considerations for Session Initiation Protocol (SIP) overload control," IETF Internet-Draft (work-in-progress), draft-ietf-soc-overload-design-08, July 2011.

[64] "Session Initiation Protocol (SIP) overload control," IETF Internet-Draft (work-in-progress), draft-ietf-soc-overload-control-02, Feb 2011.

[65] M. Homayouni, H. Nemati, V. Azhari, and A. Akbari, "Controlling overload in SIP proxies: An adaptive window based approach using no explicit feedback," in *GLOBECOM 2010, 2010 IEEE Global Telecommunications Conference*, dec. 2010, pp. 1 –5.

[66] Skype, http://www.skype.com/, 2010, [Online; accessed June 2010].

[67] S. A. Baset, J. Reich, J. Janak, P. Kaparek, V. Misra, D. Rubenstein, and H. Schulzrinne, "How Green is IP-Telephony?" in *Proc. of SIGCOMM Green Networking Workshop*, New Delhi, India, August 2010.

[68] C. Shen, E. Nahum, H. Schulzrinne, and C. Wright, "The Impact of TLS on SIP Server Performance," Munich, Germany, August 2010.

[69] K. Singh, "Reliable, Scalable and Interoperable Internet Telephony," Ph.D. dissertation, Columbia University, New York, NY, USA, 2006.

[70] D. A. Bryan, B. Lowekamp, and C. Jennings, "SOSIMPLE: A Serverless, Standards-based, P2P SIP Communication System," in *Proc. of the First International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications*, Orlando, FL, USA, July 2005.

[71] S. A. Baset and H. Schulzrinne, "OpenVoIP: An Open Peer-to-Peer VoIP and IM System," in *Proc. of SIGCOMM (demo)*, Seattle, WA, USA, September 2008. [Online]. Available: http://www1.cs.columbia.edu/ salman/peer/

[72] C. Jennings, J. Rosenberg, and E. Rescorla, "Address Settlement by Peer-to-Peer," IETF Internet-draft (work-in-progress), draft-jennings-p2psip-asp-00, July 2007.

[73] S. A. Baset, H. Schulzrinne, and M. Matuszewski, "Peer-to-Peer Protocol (P2PP)," IETF Internet-draft (work-in-progress), draft-baset-p2psip-p2pp-01, February 2007.

[74] D. Bryan, M. Zangrilli, and B. Lowekamp, "Resource Location and Discovery (RELOAD)," IETF Internet-draft (work-in-progress), draft-bryan-p2psip-reload-00, June 2007.

[75] C. Jennings, B. Lowekamp, E. Rescorla, S. A. Baset, and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol," IETF Internet-Draft (work-in-progress), draft-ietf-p2psip-base-18, August 2011.

[76] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on networking*, vol. 11, no. 1, pp. 17–32, 2003.

[77] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in *IPTPS*, 2002, pp. 53–65.

[78] C. Jennings, B. Lowekamp, E. Rescorla, S. A. Baset, and H. Schulzrinne, "A SIP Usage for RELOAD," IETF Internet-Draft (work-in-progress), draft-ietf-p2psip-sip-06, July 2010.

[79] BitTorrent, http://www.bittorrent.com/, 2010, [Online; accessed June 2010].

[80] J. R. Douceur, "The Sybil Attack," Cambridge, MA, USA, 2002.

[81] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246 (Proposed Standard), Internet Engineering Task Force, Aug. 2008, updated by RFCs 5746, 5878, 6176. [Online]. Available: http://www.ietf.org/rfc/rfc5246.txt

[82] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security," RFC 4347 (Proposed Standard), Internet Engineering Task Force, Apr. 2006, updated by RFC 5746. [Online]. Available: http://www.ietf.org/rfc/rfc4347.txt

[83] J. Rosenberg, "The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)," RFC 4825 (Proposed Standard), Internet Engineering Task Force, May 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4825.txt

[84] ——, "A Hitchhiker's Guide to the Session Initiation Protocol (SIP)," RFC 5411 (Informational), Internet Engineering Task Force, Feb. 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5411.txt

Salman Abdul Baset is a Research Staff Member in IBM T. J. Watson Research Center. He received his B.S. in Computer Systems Engineering from GIK Institute, Pakistan, and his M.S. and Ph.D. in Computer Science from Columbia University. His research is focused on designing, building, and modeling cloud-based and peer-to-peer systems. He is a recipient of 2008 Marconi Young Scholar award for promising research in communication science.

Vijay K. Gurbani is a Distinguished Member of Technical Staff in the Emerging Computing Technologies domain at Bell Laboratories, Alcatel-Lucent. He received his BSc. and MSc. degrees in Computer Science from Bradley University and a Ph.D. degree also in Computer Science from Illinois Institute of Technology. Dr. Gurbani's research interests include distributed systems, networking, signaling protocols and localization techniques in P2P networks.

Alan B. Johnston is a Distinguished Engineer at Avaya, Inc. He received his BE in Electrical Engineering from the University of Melbourne, Australia and a PhD in Electrical Engineering from Lehigh University. Dr. Johnston is an active participant in the Internet Engineering Task Force and a member of the Board of Directors of the SIP Forum. In addition Dr. Johnston is an Adjunct Instructor at Washington University in St. Louis and has written a number of technical books on SIP and VoIP security and a techno thriller novel.

Hadriel Kaplan is Vice President of Technology at Acme Packet. He has been at Acme Packet since 2004, but has been involved in VoIP since the late 90's when he worked at Nortel Networks. Hadriel deals with SIP interoperability issues on a daily basis, directly with numerous SIP Service Providers around the World. He is also an active member of the IETF, has written numerous SIP-related drafts and is a co-author of RFCs, and is on the Board of Directors of the SIP Forum.

Brian Rosen is Senior Director at Neustar, Inc. where he is a systems architect and subject matter expert on public safety. He chairs the North American Emergency Number Association (NENA) Long Term Definition working group, and is the editor of the group's technical standards for Next Generation 9-1-1. Mr. Rosen is also a major contributor to the IETF emergency calling standards, and co-chairs the p2psip, siprec and paws working groups.

Jonathan D. Rosenberg is Chief Technology Strategist for Skype, where he is responsible for Skype's technology directions and overall architecture. Prior to that, he was a Cisco Fellow at Cisco, where he set technology strategy for their enterprise PBX product. Jonathan is the principle author of SIP, the lingua-franca of Voice over IP, and has written many of the standards around it, particularly in the areas of NAT traversal, presence and IM. For his work, Jonathan was named one of the top 100 most innovative young technologists in the world by Technology Review magazine. He received a PhD from Columbia University and his masters and bachelors from MIT.