WHITE PAPER SERIES

SIP

An Introduction

Author: *James Wright, MSc*

"Session Initiation Protocol (SIP) has emerged as the strongest candidate to provide the signalling backbone for the next generation networks and the future of ubiquitous connectivity. This white paper provides an introductory text on SIP and is intended to be read by software practitioners, architects and manages who want an overview on the origins and workings of SIP."

KONNETIC
SIP & IMS .NET development

www.konnetic.com

Introduction

Many of us live in an Internet enriched world where the web and email pervades. That world is dominated by two Internet protocols: Hypertext Transfer Protocol (HTTP), which oversees delivery in the web; and Simple Mail Transfer Protocol (SMTP), which ferries emails around the globe. However, the popularity of individual expression on the web, demand for multimedia on the move, and the desire — and acceptance — for immediate access to people is driving the search for another protocol more suited to supporting these tasks. This is where Session Initiation Protocol (SIP) steps in. It was designed from the bottom up to connect people and devices whenever and wherever they are.

SIP emerged in the late 1990s and has gained traction as a facilitator of instant messaging and VoIP. A decade since inception there are now large numbers of SIP installations embedded in hardware, PC applications, smartphone apps and online services. They support a proliferation of clients. VoIP and telephone gateways have sprung up in homes and offices. High capacity networked SIP proxy and registrar servers manage thousands of user's mobility and localisation requirements. And interactive SIP systems are delivering conferencing, video and audio on demand. Furthermore, the advent of the telecom industry's IP Multimedia Subsystem is likely to see SIP becoming central to the ubiquitous provision of connectivity.

All of this has given a shot in the arm to SIP's rise to prominence. However, SIP is not the answer for all communication problems, and there are still many wrinkles to iron out — robust privacy mechanisms for one.

This paper provides an introductory text to Session Initiation Protocol. It is intended to be read by software practitioners, architects or managers wanting a high-level overview of the protocol, its history, structure, usage and operation. The first two sections look at the origins of SIP and its association with HTTP and SMTP. Following that we describe the standard in more detail and the networked entities involved. Section six provides a description of a simple SIP operation, and section seven looks at reliability within a SIP environment.

History

SIP sprang to life the mid 1990s out of research undertaken by Henning Schulzrinne, Associate Professor of the Department of Computer Science at Columbia University, and Mark Hadley, co-founder of the AT&T Center for Internet Research at ICSI (both were joined at a later stage by Jonathan Rosenberg who worked for DynamicSoft, later integrated into Cisco Systems). The work was supplemental to Schulzrinne's earlier efforts on Real-Time Transport Protocol (defined in RFC2326, RTP is used for transmission of data requiring a high quality of service (QoS) including on-demand media and interactive services). Both protocols were being developed under the auspices of the Internet Engineering Task Force (IETF), the Internet standardization body, and their working group on Multiparty Multimedia Session Control (MMUSIC) who were co-ordinating development of Internet teleconferencing and multimedia communications.

SIP Version 1.0 was submitted to the IETF by MMUSIC as an Internet Draft in 1997. The protocol advanced to proposed standard status in March 1999 and was published shortly after as RFC2543. That document was eventually replaced with a revised standard, RFC3261, and several additional proposed standards: RFC3262 on reliability of provisional responses; RFC3263 on

"SIP is a signalling procedure that manages communication sessions between two-or-more participants"

locating SIP servers; RFC3264 on the offer/answer model; and RFC3265 on specific event notification.

From this small set of protocols, SIP has morphed into a suite of interoperable rules and practices covering a huge number of scenarios. Many still under development. As of the time of writing, identity, security mechanisms, and NAT traversal were active areas of research.

These extensions are necessary because RFC3261's original purpose was tightly focused: to formalise a signalling procedure that manages communication sessions between two-or-more participants. In this context a session was expected to be a lengthy exchange of data between individuals. Existing protocols were not purpose-built for this essential human activity, and so SIP was born to fill the gap. Two other application protocols, SMTP and HTTP, still relatively new at the time, were influential in the design of SIP.

HTTP/SMTP

SIP is an application level protocol — it provides services directly to applications. Thus applications will normally deal directly, and exclusively, with a SIP API (Application Programming Interface) which in turn interacts with the transport layer (home to User Datagram Protocol (UDP) and Transmission Control Protocol (TCP)). The other application level protocols central to Internet communication are HTTP and SMTP, both were designed for client-server interaction: web-client to web-server exchange, and email-client to email-server postings respectively. However they were considered unsuitable for the task the SIP designers set themselves.

Even so, the SIP standard derives much of its behaviour from HTTP — often referring directly to it in RFC3261. Areas of similarity extend to identical text encoding schemes (ANSCII and UTF8); the use of carriage return/line feed separated header fields to organise instructions into messages, some of which are copied straight from HTTP to SIP without annotation; a security infrastructure borrowed from HTTP's authentication architecture; and the request/response message exchange pattern in which Request messages define the operation sought by the client, while Response messages provide the status of that request — this status reporting mechanism even uses identical error codes in some circumstances, for example '404' means 'Address Not Found' in both HTTP and SIP.

Where the two derivate most visibly is concerning resource identification. HTTP uses Uniform Resource Locations such as 'http://www.google.com' to identify documents and web pages for retrieval. Since SIP sessions involve communication resources, URL's were unsuitable. SIP needed an addressing scheme that embedded the identification of resources needed in the reliable transmission of request/response message to individuals. For this, the SIP designers turned to email. SMTP defines the well known email Uniform Resource Indicator (URI) (e.g. 'bob@gmail.com'). Subsequently SIP defined a SIP URI of its own, which contains sufficient information to initiate, fully define and maintain a communication session. The SIP URI (or SIP address), is a similar structure to an email address and typically contains only username and a hostname, such as 'sip:bob@biloxi.com'. Unlike an email URI however, parameter and message header information can be embedded directly into the URI.

"Because of the open nature of the IETF standards, the fact SIP is text based and shares many features with existing specifications, it has been readily understood, extended and implemented"

From SMTP, SIP also took email's use of MIME type descriptions, and uses some SMTP headers such as To, From, Date, and Subject.

Description

Like HTTP and SMTP, SIP is an Internet standard, which means that it specifies a set of instructions that are designed to be combined into a message and sent via Internet protocols such as TCP/UDP and IP. Unlike HTTP/SMTP however, it is a signalling standard: it defines how and where two or more endpoints can exchange a stream of data, but plays no part in the ultimate sending and receiving of the actual media.

Usage

SIP is an off-the-shelf viable and reliable set of rules for managing three broad user requirements:

- Session setup: negotiate the session parameters with the participants.
- Session management: provides for cancelling sessions, adding new participants, modifying of session parameters, and invoking services.
- Mobility: support for service, session, device and personal mobility.

SIP organises and sets up the media exchanges that follow as part of a well-defined session. The session definition normally includes the following:

- Identification: a session ID, subject, an indication of which end points will be used for data transfer and further requests.
- Routing: the path for data transmission.
- User availability: a mechanism to accept or turn down a request.
- Device capabilities: advertisement or requirement of device feature and media capabilities.
- Content: the content type, length, encoding and language used.

Because of the open nature of the IETF standards, the fact SIP is text based and shares many features with existing specifications, it has been readily understood, extended and implemented. An enormous number of functions have been tacked onto the initial standard, notably those for presence (displaying your status like 'away', 'online' etc), event notification and subscription, messaging, privacy, routing, QoS, interworking with the public telephone system, gateway traversal and many more.

The reason for all these extra standards and practices is twofold a) SIP confines itself to defining how sessions should be setup, maintained and managed, nothing else, and b) it is not limited by usage scenario.

Features

This popularity is being driven on by a number of attractive features that SIP embodies:

- User defined extensions can be readily defined enabling service providers to plug-in new services and management information into the protocol without breaking their networks. Legacy applications simply ignore the newer functions.
- The SIP standard is text (UTF8) based, that is, it is human readable. Resulting in simpler to comprehend messages. This makes for easier prototyping, testing and ultimately developing.
- SIP supports MIME type descriptions in a similar way to that commonly found in emails. Thus applications can be associated with SIP and launched automatically.
- No new Internet architectures are required to operate SIP. It uses DNS (taking advantage of the less common NAPTR and SRV records); runs over existing network protocols such as IP, Ethernet and ATM; and uses HTTP

security arrangements. No new services have had to be introduced to support SIP.

• SIP is indifferent to the type of transport used. SIP works just as well over UDP as over TCP, SCTP or any transport layer protocol.

Due in part to the comprehensive session features SIP defines it is becoming the de facto standard for defining and controlling interactive peer-to-peer sessions.

Components

A SIP environment consists of a number of connected entities. These include, User Agents, Proxy servers, Redirect servers, Registrar servers and Back-to-Back User Agents (B2BUA). It is the User Agent that tends to reside on the end user's device. The other entities provide essential support services in many scenarios.

Clients

We naturally associate the concept of client software to the end users. This is even more applicable with SIP as it is indifferent to the device or application the person uses and attempts to abstract away the Internet plumbing and make the person Internet addressable. The User Agent (UA) is the entity typically hosted on client software and associated with end users. It has two modes of operation:

• As a User Agent Client (UAC): Generating and sending requests to servers (which may include a UAS, see next), and receiving responses in return.

• As a User Agent Server (UAS): Receiving and processing requests, and generating responses.

Typically a single UA acts in both capacities.

Servers

SIP servers are instrumental in the location of clients and the efficient and correct routing of SIP messages.

• Proxy Server: These elements are involved in routing the SIP Request to the correct UAS and SIP Responses to the correct UAC. They are the most common type of server in a SIP environment. If an exact address of the recipient is not know at the time of message elaboration the client sends the request to a proxy server which forwards it onto another proxy server closer to the end point or the ultimate recipient itself.

• Redirect Server: A redirect server accepts requests from a client and responds to the client with a new address or different route path to the recipient. They are important in supporting mobility — when a recipient has moved location.

• Registrar: These servers act as current repository of a client's location often utilising a separate Location Server. User Agents register with a Registrar on start up or when the client changes the point of attachment to the network.

• B2BUA: These logical entities act in a dual capacity in that they receive requests like a UAS, process the request further in some manner, then behave as a UAC and forward the processed request on. B2BUA's maintain state between calls and participate in SIP transactions providing tight control over the exchange. A stateful Proxy Server may contain a B2BUA.

Making A Call

The ultimate aim of all these networked components is to make a call to someone and carry on a conversation. Taking a high level view point the following four steps are needed to establish a call, be that VoIP, messaging or conference:

1. A request is issued by the calling party for a session to be arranged. (Invite)

2. The recipients of the request acknowledge receipt of the request and attempt to solicit a response from the end user(s). (Trying/Ringing)

3. The end user confirms (or rejects) the request at this time. (OK/Reject)

4. Session parameters are established and configured in order to engage in the call.

These steps form part of what is known as the Control or Signalling Plane. The actual exchange of interactive data takes place in the Bearer or Transport Plane.

Alice Calling Bob*

To take a specific example, Alice has a SIP VoIP application on her PC. On start-up the VoIP application registers with the Registrar server using a REGISTER SIP message. Alice wants to place a call to Bob on his smartphone. She (or more precisely the UA in the VoIP application) goes through the above four steps. The UA creates an INVITE request and submits it to the proxy that is configured on Alice's PC. The proxy contacts the Registrar to obtain the current location of Bob, and subsequently forwards the invitation directly to him. Bob's Use Agent acknowledges receipt of the INVITE with a Trying response, and further Ringing responses to indicate that Bob's smartphone is attracting his attention. Bob confirms he wishes to engage in a call with Alice by instructing his UA to send an OK back to the proxy for forward delivery to Alice. Alice's UA finalises the exchange with an Acknowledgement (ACK) message. At this point the SIP session (and, incidentally a related logical state, a SIP Dialog) has been established. Furthermore, if they had also swapped Session Description Protocol (SDP) information, the media parameters such as bandwidth, timings and exact media types would also have been defined. This exchange of information occurs completely within the Control Plane.

We have now paved the way for another protocol (such as RTP) to transport the encoded voice data directly between Alice and Bob without further intervention from SIP. This transfer takes place in the Bearer Plane.

At the end of the call, SIP is used to tear down the session. If, for example, it is Bob who ends the call, his UA sends a SIP BYE request and receives a SIP 200 OK response from Alice.

In the above example Alice and Bob carry on a voice conversation. However, this could have easily been a video or Instant Messaging session instead; the procedure would look exactly the same.

Reliability

Recall that Alice's UAC first sends an INVITE request to Bob's UAS. On receipt of the INVITE, Bob's UAS responded with an OK response. Alice is then confident that the INVITE reached its destination. This exchange represents part of a SIP transaction and we say that the INVITE was sent reliably. SIP is said to be a transactional protocol: exchanges between participants occur in a series of verified and independent message exchanges. Specifically, a transaction consists of an initial request and any responses to that request. The responses may include zero or more provisional responses (e.g. Trying or Ringing) and one or more final response (e.g. OK or Reject).

Under favourable conditions Alice's UAC will expect a response in due course. SIP uses timers to manage that transactional wait period. Using these timers Alice may choose to resend the original INVITE or simply give up and terminate the transaction.

In the same example, Bob's OK message was acknowledged by Alice's UAC with an ACK message. Thus the OK is transmitted reliably too. In the case of a transaction in which the original request was an INVITE (known as an INVITE transaction), the exchange must also include an ACK if the final response was not a failure response. If the response was a failure, the ACK is not considered part of the transaction. All other requests engage in what are known as Non-INVITE

** Alice calling Bob is the SIP equivalent of the archetypal 'Hello World' application.*

transactions, and ACK messages are not permitted.

SIP transactions are said to have a client side and a server side. The client side uses a client transaction to send requests. The server side uses server transaction to sends the responses to those requests. These transactions are logical entities embedded within User Agents and stateful Proxies.

In the example above, Alice's UAC was the client side as it initiated the request, and Bob's UAS played the part of the server side, responding to the request.

Because of this transactional activity (as well as incremental message sequences in the CSeq header field) SIP does not require the services of a reliable transport such as TCP and can function dependably using UDP say.

Further Reading

H. Schulzrinne. (2010) Henning Schulzrinne. [Online]. http://www.cs.columbia.edu/~hgs/

A. B. Johnston, SIP: Understanding the Session Initiation Protocol, 3th ed. Boston, MA, USA: Artech House, 2007.

J. Rosenberg et al., "SIP: Session Initiation Protocol," RFC 3261 2002.

IETF. (2010) Multiparty Multimedia Session Control (mmusic). [Online]. http://datatracker.ietf.org/wg/mmusic

IETF. (2010) Session Initiation Protocol (sip). [Online]. http://datatracker.ietf.org/wg/sip/

Konnetic Website. http://www.konnetic.com

Other Papers In the Series

IMS - Technologies and Architecture

http://www.konnetic.com/documents/KonneticIMSTechnologiesAndArchitecture.pdf

IMS - Services and Revenue

http://www.konnetic.com/documents/KonneticIMSServicesAndRevenue.pdf

SIP - Technical Overview

http://www.konnetic.com/documents/KonneticSIPTechnicalOverview.pdf

SDP - Technical Overview

http://www.konnetic.com/documents/KonneticSDPTechnicalOverview.pdf

---

James Wright MSc is the founder of Konnetic, a specialist in providing SIP and IMS based software to the .NET (C#, VB.NET, F# and more) community.