Building Telephony Systems with OpenSIPS
Second Edition

Flavio E. Goncalves
Bogdan-Andrei Iancu

What you will learn from this book
- Learn to prepare and configure a Linux system for OpenSIPS
- Familiarise yourself with the installation and configuration of OpenSIPS
- Understand how to set a domain and create users/extensions
- Configure SIP endpoints and make calls between them
- Make calls to and from the PSTN and create access control lists to authorize calls
- Install a graphical user interface to simplify the task of provisioning user and system information
- Implement an effective billing system with OpenSIPS
- Monitor and troubleshoot OpenSIPS to keep it running smoothly

This book will give you a competitive edge by helping you to create an SIP infrastructure capable of handling tens of thousands of subscribers.

Starting with an introduction to SIP and OpenSIPS, you will begin by installing and configuring OpenSIPS. You will be introduced to the OpenSIPS scripting language and OpenSIPS routing concepts, followed by comprehensive coverage of Subscriber Management. The final chapters of the book are dedicated to troubleshooting tools, SIP security, and advanced scenarios including TCP/TLS support, load balancing, asynchronous processing, and more.

A fictional VoIP provider is used to explain OpenSIPS and by the end of the book you will have a simple but complete system for running a VoIP provider.

Who this book is written for
If you want to understand how to build a SIP provider from scratch using OpenSIPS, then this book is ideal for you. This book will also help readers who were using OpenSER but are now confused by the new OpenSIPS.

Telephony and Linux experience will be helpful to get the most out of this book but are not essential. Prior knowledge of OpenSIPS is not assumed.


Flavio E. Goncalves
Bogdan-Andrei Iancu

$ 44.99 US
£ 28.99 UK

Prices do not include local sales tax or VAT where applicable
In this package, you will find:

- The author's biography
- A preview chapter from the book, Chapter 1 'Introduction to SIP'
- A synopsis of the book’s content
- More information on Building Telephony Systems with OpenSIPS Second Edition
About the Authors

Flavio E. Goncalves was born in 1966 in Brazil. Having a strong interest in computers, he got his first personal computer in 1983, and since then, it has been almost an addiction. He received his degree in engineering in 1989 with a focus on computer-aided designing and manufacturing.

He is also the CTO of SipPulse Routing and Billing Solutions in Brazil—a company dedicated to the implementing of small-to-medium telephone companies, VoIP providers, and large-scale new generation telephony systems. Since 1993, he has participated in a series of certification programs and been certificated as Novell MCNE/MCNI, Microsoft MCSE/MCT, Cisco CCSP/CCNP/CCDP, Asterisk dCAP, and some others.

He started writing about open source software because he thinks that the way certification programs have worked is very good for learners. Some books are written by strictly technical people who sometimes do not have a clear idea on how people learn. He tried to use his 15 years of experience as an instructor to help people learn about the open source telephony software. Together with Bogdan, he created the OpenSIPS boot camp followed by the e-learning program, OpenSIPS eBootcamp.

His experience with networks, protocol analyzers, and IP telephony combined with his teaching experience gave him an edge to write this book. This is the fourth book written by him. The first one was Configuration Guide for Asterisk PBX, by BookSurge Publishing, the second was Building Telephony Systems with OpenSER, by Packt Publishing, and the third was Building Telephony Systems With OpenSIPS 1.6, by Packt Publishing.

As the CTO of SipPulse, Flavio balances his time between family, work, and fun. He is the father of two children and lives in Florianopolis, Brazil—one of the most beautiful places in the world. He dedicates his free time to water sports such as surfing and sailing.
**Bogdan-Andrei Iancu** entered the SIP world in 2001, right after graduating in computer science from the Politehnica University of Bucharest, Romania. He started as a researcher at the FOKUS Fraunhofer Institute, Berlin, Germany. For almost four years, Bogdan accumulated a quick understanding and experience of VoIP/SIP, being involved in research and industry projects and following the evolution of the VoIP world closely.

In 2005, he started his own company, Voice System. The company entered the open source software market by launching the OpenSER/OpensIPS project—a free GPL-SIP proxy implementation. As the CEO of Voice System, Bogdan pushes the company in two directions: developing and supporting.

The OpenSIPS public project (Voice System being the major contributor and sponsor of the project) creates professional solutions and platforms (OpenSIPS-based) for the industry. In other words, Bogdan's interest was to create knowledge (through the work with the project) and to provide the knowledge where needed (embedded in commercial products or raw format as consultancy services). In the effort of sharing the knowledge of the SIP/OpenSIPS project, he started to run the OpenSIPS Bootcamp in 2008 together with Flavio E. Goncalves, which is intensive training dedicated to people who want to learn and get hands-on experience on OpenSIPS from experienced people. Bogdan's main concern is to research and develop new technologies or software for SIP-based VoIP (this is the reason for his strong involvement with the OpenSIPS project) and pack all these cutting-edge technologies as professional solutions for the industry.
Preface

This book will be your companion when working with OpenSIPS using a case study for an Internet Telephony Service Provider (ITSP). With the help of this book, you should be able to build a system that is able to authenticate, route, bill, and monitor VoIP calls. Topics and advanced scenarios such as TCP/TLS support, load balancing, asynchronous processing, and more are discussed in depth in this book. You will create dynamic dialplans, route calls using advanced routing, integrate OpenSIPS with a media server, account calls and generate CDRs, provision the system using a Web GUI, and use tools to monitor and check the health of your server. You will also learn some advanced topics such as support for TLS/TCP and the newest technology called asynchronous callbacks.

By the end of this book, you should be able to build a system that is able to authenticate, route, bill, and monitor VoIP calls. Whenever you are thinking big on telephony, OpenSIPS is your savior and this book is your friend!

What this book covers

*Chapter 1, Introduction to SIP,* introduces you to the SIP server. You will see how to recognize a SIP request and reply according to RFC 3261, identify the mandatory SIP headers, and describe the SIP routing process for initial and sequential requests.

*Chapter 2, Introducing OpenSIPS,* shows you how OpenSIPS is used in the market, the basic architecture of the system, use cases, and the main target market.

*Chapter 3, Installing OpenSIPS,* shows you how to download the OpenSIPS source and its dependencies, compile and install OpenSIPS with MySQL and Radius support, and configure the Linux system to start OpenSIPS at boot time.
Chapter 4, *OpenSIPS Language and Routing Concepts*, introduces you to the OpenSIPS scripting language and OpenSIPS routing concepts. After reading this chapter, you should be able to recognize the OpenSIPS script language, describe its main commands, process initial requests, and drop or route requests.

Chapter 5, *Subscriber Management*, shows you how to manage subscribers in the system using the subscriber, location, group, and address databases. You will learn how to implement a multidomain system that is able to support multitenant implementations.

Chapter 6, *OpenSIPS Control Panel*, demonstrates how to install a web GUI to help with the provisioning of users, dialplan, routes, and other information that is required to run OpenSIPS. You will see how to install, use, configure, and customize the OpenSIPS control panel.

Chapter 7, *Dialplan and Routing*, enables you to integrate OpenSIPS with PSTN through gateways, selecting the best gateway, and failing over automatically if a response code is negative.

Chapter 8, *Managing Dialogs*, shows you how to activate the dialog module, limit the number of simultaneous calls, disconnect hanged calls, impose a maximum duration time for a call, and implement SIP session timers integrated with the dialog module.

Chapter 9, *Accounting*, demonstrates how account calls generate a CDR (Call Detail Record), account correctly forwarded calls, prevent calls without BYE, and add extra fields to the CDR.

Chapter 10, *SIP NAT Traversal*, helps you implement an OpenSIPS solution for clients behind NAT. You will see how to implement OpenSIPS in a data center such as Amazon AWS where all the servers are behind NAT.

Chapter 11, *Implementing SIP Services*, implements services such as call forward, forward on busy, and forward on no answer in cooperation with a media server and SIP phone.

Chapter 12, *Monitoring Tools*, enables you to detect performance issues using the built-in statistics. These include protocol issues using SIP trace, database issues using the benchmark module, script issues using the script trace, and software and hardware issues using GDB.

Chapter 13, *OpenSIPS Security*, shows you how to increase the security of your OpenSIPS installation.
Chapter 14, *Advanced Topics with OpenSIPS 2.1*, covers some advanced topics that can be important for specific installations. Topics such as asynchronous processing, TCP and TLS support, binary replication, and NoSQL integration for clusters are discussed.
Introduction to SIP

Before we dive into OpenSIPS, it is very important to understand some important concepts related to Session Initiation Protocol (SIP). In this chapter, we will cover a brief tutorial regarding the concepts used later in this book. By the end of this chapter, we will have covered the following topics:

• Understanding the basics of SIP and its usage
• Describing the SIP architecture
• Explaining the meaning of its components
• Understanding and comparing main SIP messages
• Interpreting the header fields' processing for the INVITE and REGISTER messages
• Learning how SIP handles identity and privacy
• Covering the Session Description Protocol and Real-Time Protocol briefly

SIP was standardized by Internet Engineering Task Force (IETF) and is described in several documents known as Request for Comments (RFC). The RFC 3261 describes SIP version 2. SIP is an application layer protocol used to establish, modify, and terminate sessions or multimedia calls. These sessions can be audio and video sessions, e-learning, chatting, or screen sharing sessions. It is similar to Hypertext Transfer Protocol (HTTP) and designed to start, keep, and close interactive communication sessions between users. Nowadays, SIP is the most popular protocol used in Internet Telephony Service Providers (ITSPs), IP PBXs, and voice applications.
The SIP protocol supports five features to establish and close multimedia sessions:

- **User location**: Determines the endpoint address used for communication
- **User parameters negotiation**: Determines the media and parameters to be used
- **User availability**: Determines if the user is available or not to establish a session
- **Call establishment**: Establishes parameters for caller and callee and informs about the call progress (such as ringing, busy, or not found) to both the parties
- **Call management**: Facilitates session transfer and closing

The SIP protocol was designed as a part of a multimedia architecture containing other protocols such as **Resource Reservation Protocol (RSVP)**, **Real-Time Protocol (RTP)**, **Real-Time Session Protocol (RTSP)**, **Session Description Protocol (SDP)**, and **Session Announcement Protocol (SAP)**. However, it does not depend on them to work.

**Understanding the SIP architecture**

SIP has borrowed many concepts from the HTTP protocol. It is a text-based protocol and uses the same Digest mechanism for authentication. You will also notice similar error messages such as 404 (Not found) and 301 (Redirect). As a protocol developed by the IETF, it uses an addressing scheme similar to **Simple Mail Transfer Protocol (SMTP)**. The SIP address is just like an e-mail address. Another interesting feature used in SIP proxies are aliases; you can have multiple SIP addresses for a single subscriber such as the following:

- johndoe@sipA.com
- +554845678901@sipA.com
- 45678901@sipA.com

In the SIP architecture, there are user agents and servers. SIP uses a peer-to-peer distributed model with a signaling server. The signaling server only handles the SIP signaling, while the user agent clients and servers handle signaling and media. This is depicted in the following figure:
In the traditional SIP model, a user agent, usually a SIP phone, will start communicating with its SIP proxy, seen here as the outgoing proxy (or its home proxy) to send the call using a message known as INVITE.

The outgoing proxy will see that the call is directed to an outside domain. According to RFC 3263, it will seek the DNS server for the address of the target domain and resolve the IP address. Then, the outgoing proxy will forward the call to the SIP proxy responsible for DomainB.

The incoming proxy will query its location table for the IP address of agentB if its address was inserted in the location table by a previous registration process. It will forward the call to agentB.

After receiving the SIP message, agentB will have all the information required to establish an RTP session (usually audio) with agentA sending a 200 OK response. Once agentA receives the response from agentB, a two-way media can be established. A BYE request message can terminate the session.
Here, you can see the main components of the SIP architecture. The entire SIP signaling flows through the SIP proxy server. On the other hand, the media is transported by the RTP protocol and flows directly from one endpoint to another. Some of the components will be briefly explained in the sequence.

In the preceding image, you can see the following components:

- **UAC (User Agent Client)**: A client or terminal that starts the SIP signaling
- **UAS (User Agent Server)**: A server that responds to the SIP signaling coming from a UAC
- **UA (User Agent)**: A logical entity that can act as both UAC or UAS, such as a SIP endpoint (IP phones, ATAs, softphones, and so on)
- **Proxy Server**: Receives requests from a UA and transfers to another SIP proxy if this specific terminal is not under its domain
- **Redirect Server**: Receives requests and responds to the caller with a message containing data about the destination (302, Moved Temporarily)
- **Registrar Server**: Provides the callee's contact addresses to the proxy and redirect servers

The proxy, redirect, and registrar servers are usually available physically in the same computer and software.
The SIP registration process

The SIP registration process is shown as follows:

The SIP protocol employs a component called Registrar. It is a server that accepts REGISTER requests and saves the information received in these packets on the location server for their managed domains. The SIP protocol has a discovery capacity; in other words, if a user starts a session with another user, the SIP protocol has to discover an existent host where the user can be reached. The discovery process is done (among others) by a Registrar server that receives the request and finds the location to send it. This is based in a location database maintained by the Registrar server per domain. The Registrar server can accept other types of information, not only the client's IP addresses. It can receive other information such as Call Processing Language (CPL) scripts on the server.

Before a telephone can receive calls, it needs to be registered with the location database. In this database, we will have all the phones associated with their respective IP addresses. In our example, you will see the sip user, 8500@opensips.org, registered with the IP address, 200.180.1.1.
RFC 3665 defines best practices to implement a minimum set of functionalities for a SIP IP communications network. In the following table, the flows are defined according to RFC 3665 for registration transactions. According to RFC 3665, there are five basic flows associated with the process of registering a user agent.

<table>
<thead>
<tr>
<th>Message flow</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User Agent</strong></td>
<td><strong>New registration</strong></td>
</tr>
<tr>
<td></td>
<td>REGISTER</td>
</tr>
<tr>
<td></td>
<td>401 Unauthorized</td>
</tr>
<tr>
<td></td>
<td>REGISTER</td>
</tr>
<tr>
<td></td>
<td>200 OK</td>
</tr>
<tr>
<td></td>
<td>A successful new registration: After sending the Register request, the user agent will be challenged against its credentials. We will see this in detail in Chapter 5, Subscriber Management.</td>
</tr>
<tr>
<td><strong>User Agent</strong></td>
<td><strong>Update of Contact List</strong></td>
</tr>
<tr>
<td></td>
<td>REGISTER</td>
</tr>
<tr>
<td></td>
<td>200 OK</td>
</tr>
<tr>
<td></td>
<td>An update of the contact list: As it is not a new registration, the message already contains the Digest and a 401 message won't be sent. To change the contact list, the user agent just needs to send a new register message with the new contact in the CONTACT header field.</td>
</tr>
<tr>
<td><strong>User Agent</strong></td>
<td><strong>Request for current Contact List</strong></td>
</tr>
<tr>
<td></td>
<td>REGISTER</td>
</tr>
<tr>
<td></td>
<td>200 OK</td>
</tr>
<tr>
<td></td>
<td>A request for the current contact list: In this case, the user agent will send the CONTACT header field empty, indicating that the user wishes to query the server for the current contact list. In the 200 OK message, the SIP server will send the current contact list in the CONTACT header field.</td>
</tr>
<tr>
<td><strong>User Agent</strong></td>
<td><strong>Cancellation of the registration</strong></td>
</tr>
<tr>
<td></td>
<td>REGISTER</td>
</tr>
<tr>
<td></td>
<td>200 OK</td>
</tr>
<tr>
<td></td>
<td>The cancellation of a registration: The user agent now sends the message with an EXPIRES header field of 0 and a CONTACT header field configured as * to apply to all the existing contacts.</td>
</tr>
</tbody>
</table>
Types of SIP servers

There are a few different types of SIP servers. Depending on the application, you can use one or all of them in your solution. OpenSIPS can behave as a proxy, redirect, B2BUA, or Registrar server.

The proxy server

In the SIP proxy mode, all SIP signaling goes through the SIP proxy. This behavior will help in processes such as billing and is, by far, the most common choice. The drawback is the overhead caused by the server in the middle of all the SIP communications during the session establishment. Regardless of the SIP server role, the RTP packets will go directly from one endpoint to another even if the server is working as a SIP proxy.
Introduction to SIP

The redirect server

The SIP proxy can operate in the SIP redirect mode. In this mode, the SIP server is very scalable because it doesn't keep the state of the transactions. Just after the initial INVITE, it replies to the UAC with a 302 Moved Temporarily and is removed from the SIP dialog. In this mode, a SIP proxy, even with very few resources, can forward millions of calls per hour. It is normally used when you need high scalability but don't need to bill the calls.

```
① INVITE sip:8500@opensips.org
From: sip:2400@sip.com
To: sip:8500@opensips.org
Call-ID 2400@sip.com
② OK 302 moved temporarily
Contact sip:8500@200.180.4.168
③ INVITE 8500@200.180.4.168
④ OK 200
⑤ ACK 8500@200.180.4.168
```

The B2BUA server

The server can also work as a Back-to-Back User Agent (B2BUA). B2BUAs are normally applied to hide the topology of the network. They are also useful to support buggy clients unable to route SIP requests correctly based on record routing. Many PBX systems such as Asterisk, FreeSwitch, Yate, and others work as B2BUAs.
SIP request messages

There are several types of message requests. SIP is transactional, communicating through requests and replies. The most important types of requests are described in the following table:

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
<th>RFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>Acknowledges an INVITE</td>
<td>RFC 3261</td>
</tr>
<tr>
<td>BYE</td>
<td>Terminates an existing session</td>
<td>RFC 3261</td>
</tr>
<tr>
<td>CANCEL</td>
<td>Cancels a pending registration</td>
<td>RFC 3261</td>
</tr>
<tr>
<td>INFO</td>
<td>Provides mid-call signaling information</td>
<td>RFC 2976</td>
</tr>
<tr>
<td>INVITE</td>
<td>Session establishment</td>
<td>RFC 3261</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>Instant message transport</td>
<td>RFC 3428</td>
</tr>
<tr>
<td>NOTIFY</td>
<td>Sends information after subscribing</td>
<td>RFC 3265</td>
</tr>
<tr>
<td>PRACK</td>
<td>Acknowledges a provisional response</td>
<td>RFC 3262</td>
</tr>
<tr>
<td>PUBLISH</td>
<td>Uploads the status information to the server</td>
<td>RFC 3903</td>
</tr>
<tr>
<td>REFER</td>
<td>Asks another UA to act on Uniform Resource Identifier (URI)</td>
<td>RFC 3515</td>
</tr>
<tr>
<td>REGISTER</td>
<td>Registers the user and updates the location table</td>
<td>RFC 3261</td>
</tr>
<tr>
<td>SUBSCRIBE</td>
<td>Establishes a session to receive future updates</td>
<td>RFC 3265</td>
</tr>
<tr>
<td>UPDATE</td>
<td>Updates a session state information</td>
<td>RFC 3311</td>
</tr>
</tbody>
</table>
Introduction to SIP

Most of the time, you will use REGISTER, INVITE, ACK, BYE, and CANCEL. Some messages are used for other features. For example, INFO is used for Dual-tone Multi-frequency (DTMF) relay and mid-call signaling information. PUBLISH, NOTIFY, and SUBSCRIBE give support to the presence systems. REFER is used for call transfer and MESSAGE for chat applications. Newer requests can appear depending on the protocol standardization process. Responses to these requests are in the text format as in the HTTP protocol. Some of the most important replies are shown as follows:

<table>
<thead>
<tr>
<th>Description</th>
<th>Code</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Informational or provisional response</td>
<td>1XX</td>
<td>100 Trying, 480 Ringing, 181 Call Is Being Forwarded</td>
</tr>
<tr>
<td>Success</td>
<td>2XX</td>
<td>200 OK, 202 Accepted</td>
</tr>
<tr>
<td>Redirect</td>
<td>3XX</td>
<td>300 Multiple-Choices, 308 Alternative Service</td>
</tr>
<tr>
<td>Client Errors</td>
<td>4XX</td>
<td>400 Bad Request, 401 Unauthorized, 402 Payment Required, 403 Forbidden, 404 Not Found, 405 Method Not Allowed, 406 Not acceptable, 407 Proxy Authentication Required, 408 Request Timeout, 409 Conflict, 410 Gone</td>
</tr>
<tr>
<td>Server Errors</td>
<td>5XX</td>
<td>500 Internal Server Error, 501 Not implemented, 502 Bad Gateway</td>
</tr>
<tr>
<td>Global Errors</td>
<td>6XX</td>
<td>600 Busy Everywhere, 603 Decline</td>
</tr>
</tbody>
</table>

The SIP dialog flow

Let's examine this message sequence between two user agents as shown in the following figure. You can see several other flows associated with the session establishment in RFC 3665:
The messages are labeled in sequence. In this example, **userA** uses an IP phone to call another IP phone over the network. To complete the call, two SIP proxies are used.

The **userA** calls **userB** using its SIP identity called the SIP URI. The URI is similar to an e-mail address, such as `sip:userA@sip.com`. A secure SIP URI can be used too, such as `sips:userA@sip.com`. A call made using `sips:` (Secure SIP) will use a secure transport, **Transport Layer Security (TLS)**, between the caller and callee.

The transaction starts with **userA** sending an **INVITE** request addressed to **userB**. The INVITE request contains a certain number of header fields. Header fields are named attributes that provide additional information about the message and include a unique identifier, the destination, and information about the session.

```
INVITE from A->B
INVITE sip:userB@sipB.com SIP/2.0
Via: SIP/2.0/UDP moon.sipA.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: userB <sip:userB@sipB.com>
From: userA <sip:userA@sipA.com>;tag=1234567890
Call-ID: a84b4c7666710@moon.sipA.com
CSeq: 314159 INVITE
Contact: <sip:userB@sun.sipB.com>
Content-Type: application/sdp
Content-Length: 142
(SDP not shown)
```
The first line of the message contains the method name and request URI. The following lines contain a list of header fields. This example contains the minimum set required. The header fields have been described as follows:

- **Method and Request-URI**: In the first line, you have the request URI also referred to as RURI. It contains the current destination of the message and is often manipulated by the proxies to route a request. It is the most important field in a SIP request.

- **Via**: This contains the address to which userA will be waiting to receive responses to this request. It also contains a parameter called branch that identifies this transaction. The Via header defines the last SIP hop as IP, transport, and transaction-specific parameters. Via is used exclusively to route back the replies. Each proxy adds an additional Via header. It is a lot easier for replies to find their route back using the Via header than to go again in the location server or DNS.

- **To**: This contains the name (display name) and SIP URI (that is, sip:userB@sip.com) in the destination originally selected. The To header field is not used to route the packets.

- **From**: This contains the name and SIP URI (that is, sip:userA@sip.com) that indicates the caller ID. This header field has a tag parameter containing a random string that was added to the URI by the IP phone. It is used for the purposes of identification. The tag parameter is used in the To and From fields. It serves as a general mechanism to identify the dialog, which is the combination of the Call-ID along with the two tags, one from each participant in the dialog. Tags can be useful in parallel forking.

- **Call-ID**: This contains a globally unique identifier for this call generated by the combination of a random string and it may contain the hostname or IP address of the UAC. A combination of the To, From, and Call-ID tags fully defines an end-to-end SIP relation known as a SIP dialog.

- **CSeq**: The CSeq or command sequence contains an integer and a method name. The CSeq number is incremented to each new request in a SIP dialog and is a traditional sequence number.

- **Contact**: This contains a SIP URI, which represents a direct route to contact userA, usually composed of a user name and fully qualified domain name (FQDN). It is usual to use the IP address instead of the FQDN in this field. While the Via header field tells the other elements where to send a response, the Contact tells the other elements where to send future requests.

- **Max-Forwards**: This is used to limit the number of allowed hops that a request can make in the path to their final destination. It consists of an integer decremented by each hop.
• **Content-Type**: This contains a body message description.
• **Content-Length**: This contains a byte count of the body message.

Session details such as the media type and codec are not described in SIP. Instead, it uses the **Session Description Protocol (SDP)** (RFC 2327). This SDP message is carried by the SIP message, similar to an e-mail attachment.

The phone does not know the location of userB or the server responsible for domainB. Thus, it sends the INVITE request to the server responsible for the domain, sipA. This address is configured in the phone of userA or can be discovered by DNS. The server sipA.com is also known as the SIP proxy for the domain sipA.com.

The sequence is as follows:

1. In this example, the proxy receives the INVITE request and sends a 100 Trying reply back to userA, indicating that the proxy received INVITE and is working to forward the request. The SIP reply uses a three-digit code followed by a descriptive phrase. This response contains the same To, From, Call-ID, and CSeq header fields and a branch parameter in the header field, Via. This allows for the userA's phone to correlate the INVITE request that is sent.

2. ProxyA locates ProxyB consulting a DNS server (NAPTR and SRV records) to find which server is responsible for the SIP domain sipB and forwards the INVITE request. Before sending the request to proxyA, it adds a Via header field that contains its own address. The INVITE request already has the address of userA in the first Via header field.

3. ProxyB receives the INVITE request and responds with a 100 Trying reply to ProxyA indicating that it is processing the request.

4. ProxyB consults its own location database for userB's address and then it adds another Via header field with its own address to the INVITE request and forwards this to userB's IP address.

5. The userB's phone receives the INVITE request and starts ringing. The phone responds to this condition by sending a 180 Ringing reply.

6. This message is routed back through both the proxies in the reverse direction. Each proxy uses the Via header fields to determine where to send the response and removes its own Via header from the top. As a result, the message 180 Ringing can return to the user without any lookups to DNS or Location Service and without the need for stateful processing. Thus, each proxy sees all the messages resulting from the INVITE request.

7. When userA's phone receives the 180 Ringing message, it starts to ring back in order to signal the user that the call is ringing on the other side. Some phones show this in the display.
8. In this example, userB decides to attend the call. When they pick up the handset, the phone sends a response of 200 OK to indicate that the call was taken. The 200 OK message contains in its body a session description specifying the codecs, ports, and everything pertaining to the session. It uses the SDP protocol for this duty. As a result, an exchange occurs in two phases of messages from A to B (INVITE) and B to A (200 OK) negotiating the resources and capabilities used on the call in a simple offer/response model. If userB does not want to receive the call or is busy, the 200 OK won’t be sent and a message signaling the condition (that is, 486 Busy Here) will be sent instead.

<table>
<thead>
<tr>
<th>Response 200 Ok</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP/2.0 200 OK</td>
</tr>
<tr>
<td>Via: SIP/2.0/UDP sipB.com</td>
</tr>
<tr>
<td>:branch=z9hG4bKnashds8;received=192.0.2.3</td>
</tr>
<tr>
<td>Via: SIP/2.0/UDP moon.sipA.com</td>
</tr>
<tr>
<td>:branch=z9hG4bk77ef4c2312983.1;received=192.0.2.2</td>
</tr>
<tr>
<td>Via: SIP/2.0/UDP phoneA.sipA.com</td>
</tr>
<tr>
<td>:branch=z9hG4bk77basdhd s;received=192.0.2.1</td>
</tr>
<tr>
<td>To: userB <a href="">sip:userB@sipB.com</a>;tag=a6c85cf</td>
</tr>
<tr>
<td>From: userA <a href="">sip:userA@sipA.com</a>;tag=1928301774</td>
</tr>
<tr>
<td>Call-ID: <a href="mailto:a84b4c76e6776@phoneA.sipA.com">a84b4c76e6776@phoneA.sipA.com</a></td>
</tr>
<tr>
<td>CSeq: 314159 INVITE</td>
</tr>
<tr>
<td>Contact: <a href="">sip:userB@192.0.2.4</a></td>
</tr>
<tr>
<td>Content-Type: application/sdp</td>
</tr>
<tr>
<td>Content-Length: 131</td>
</tr>
</tbody>
</table>

The first line contains the response code and a description (OK). The following lines contain the header fields. The Via, To, From, Call-ID, and CSeq fields are copied from the INVITE request and the To tag is attached. There are three Via fields: one added by userA, another by ProxyA, and finally, ProxyB. The SIP phone of userB adds a tag parameter for the To and From headers and will include this tag on all the future requests and responses for this call.

The Contact header field contains the URI by which userB can be contacted directly in its own IP phone.

The Content-Type and Content-Length header fields give some information about the SDP header. The SDP header contains media-related parameters used to establish the RTP session.
After answering the call, the following occurs:

1. The 200 OK message is sent back through both the proxies and received by userA and then the phone stops ringing, indicating that the call was accepted.

2. Finally, userA sends an ACK message to userB's phone confirming the reception of the 200 OK message. When record routing is not involved, the ACK is sent directly from phoneA to phoneB avoiding both the proxies. ACK is the only SIP method that has no reply. The endpoints learn each other's addresses from the CONTACT header fields during the INVITE process. This ends the cycle, INVITE/200 OK/ACK, also known as the SIP three-way handshake.

3. At this moment, the session between both the users starts and they send media packets to each other using a mutually agreed format established by the SDP protocol. Usually, these packets are end to end. During the session, the parties can change the session characteristics issuing a new INVITE request. This is called a reinvite. If the reinvite is not acceptable, a 488 Not Acceptable Here message will be sent, but the session will not fail.

4. At the end of the session, userB disconnects the phone and generates a BYE message. This message is routed directly to userA's SIP phone, bypassing both the proxies.

5. The userA confirms the reception of the BYE message with a 200 OK message ending the session. No ACK is sent. An ACK is sent only for INVITE requests.

In some cases, it can be important for the proxies to stay in the middle of the signaling to see all the messages between the endpoints during the whole session. If the proxy wants to stay in the path after the initial INVITE request, it has to add the Record-Route header field to the request. This information will be received by userB's phone and will send back the message through the proxies with the Record-Route header field included too. Record-routing is used in most scenarios. Without record-routing, it is not possible to account the calls and there is no control of the SIP dialog in the proxy.

The REGISTER request is the way that ProxyB learns the location of userB. When the phone initializes or in regular time intervals, the SIP phoneB sends a REGISTER request to a server on domain sipB known as SIP Registrar. The REGISTER messages associate a URI (userB@sipB.com) to an IP address. This binding is stored in a database in the Location server. Usually the Registrar, Location, and Proxy servers are in the same computer and use the same software such as OpenSIPS. A URI can only be registered by a single device at a certain time.
SIP transactions and dialogs

It is important to understand the difference between a transaction and dialog because we will use this ahead in OpenSIPS scripting. For example, there are attribute value pairs attached to transactions and dialog variables attached to dialogs. If you can't recognize a dialog and variable, it will be hard to configure the SIP server.

A transaction occurs between a user agent client and server and comprises of all the messages from the request to the final response (including all the interim responses). The responses can be provisional, starting with one followed by two digits (for example, 180 Ringing) or final, starting with two followed by two digits (for example, 200 OK). The scope of a transaction is defined by the stack of Via headers of the SIP messages. So, the user agents, after the initial invite, don't need to rely on DNS or location tables to route the messages.

The ACK request is a special case. For positive replies (2XX), the UAC creates a new transaction and generates a new CONTACT header and it can be sent straight to the UAS bypassing the proxy. However, for negative replies, it belongs to the INVITE transaction because it is not possible to create a new transaction without the Contact of the other part. In this case, the request is sent to the same proxy as INVITE.
According to RFC 3261, a dialog represents a peer-to-peer SIP relationship between two user agents that persists for some time. A dialog is identified at each UA with a dialog ID, which consists of a Call-ID value, local tag, and remote tag present in the From and To headers, respectively.

A dialog is a succession of transactions that control the creation, existence, and termination of the dialog. All dialogs do have a transaction to create them and may (or may not) have a transaction to change the dialog (mid-transaction). Additionally, the end-dialog transaction may be missing. (Some dialogs do end based on timeouts rather than on explicit termination.)

According to RFC 3665, there are 11 basic session establishment flows. The list is not meant to be complete but covers the best practices. The first two were already covered in this chapter, *Successful Session Establishment* and *Session Establishment through two Proxies*. Some of them will be seen in *Chapter 11, Implementing SIP Services*.

### Locating the SIP servers

Similar to e-mail servers, you will need to specify which server would serve a specific domain. The location of the SIP servers is described in RFC 3263. The first objective of location is to determine the IP, port, and transport protocol for the server based on the domain name. The second objective is to determine the address of a backup for the first proxy.

To perform these objectives, we will use a Domain Name System, more specifically, **Name Authority Pointer (NAPTR)** and **Service (SRV)** records. NAPTR records are employed to determine the transport protocol. To specify a transport protocol, you should insert the DNS records in the zone file of your DNS server. (Check the documentation of your DNS server on how to do it.) In the following code, we are enabling three protocols for this domain, TLS, TCP, and UDP. If the client supports TLS and UDP, TLS will be chosen because of the defined order in the records:

<table>
<thead>
<tr>
<th>Order</th>
<th>pref</th>
<th>flags</th>
<th>service</th>
<th>regexp</th>
<th>replacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN NAPTR</td>
<td>10</td>
<td>50</td>
<td>&quot;s&quot;</td>
<td>&quot;SIPS+D2T&quot;</td>
<td>_sips._tcp.opensips.org.</td>
</tr>
<tr>
<td>IN NAPTR</td>
<td>20</td>
<td>50</td>
<td>&quot;s&quot;</td>
<td>&quot;SIP+D2T&quot;</td>
<td>_sip._tcp.opensips.org.</td>
</tr>
<tr>
<td>IN NAPTR</td>
<td>30</td>
<td>50</td>
<td>&quot;s&quot;</td>
<td>&quot;SIP+D2U&quot;</td>
<td>_sip._udp.opensips.org.</td>
</tr>
</tbody>
</table>

After selecting the transport protocol, it is time to select the preferred server, which is done as follows:

<table>
<thead>
<tr>
<th>Service</th>
<th>TTL</th>
<th>Class</th>
<th>P/W</th>
<th>Port</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>_sips._tcp.opensips.org.</td>
<td>86400</td>
<td>IN</td>
<td>SRV</td>
<td>0</td>
<td>5 5060 sipA.opensips.org.</td>
</tr>
<tr>
<td>_sips._tcp.opensips.org.</td>
<td>86400</td>
<td>IN</td>
<td>SRV</td>
<td>0</td>
<td>5 5060 sipB.opensips.org.</td>
</tr>
<tr>
<td>_sip._udp.opensips.org.</td>
<td>86400</td>
<td>IN</td>
<td>SRV</td>
<td>0</td>
<td>5 5060 sipA.opensips.org.</td>
</tr>
<tr>
<td>_sip._udp.opensips.org.</td>
<td>86400</td>
<td>IN</td>
<td>SRV</td>
<td>0</td>
<td>5 5060 sipB.opensips.org.</td>
</tr>
</tbody>
</table>
Introduction to SIP

The terms in the preceding code are described as follows:

- **Service**: The symbolic name of the desired service
- **TTL**: The standard DNS time to live field
- **Class**: The standard DNS class field (this is always IN)
- **Priority**: The priority of the target host; a lower value means more preferred
- **Weight**: A relative weight for the records with the same priority
- **Port**: The TCP or UDP port on which the service is to be found
- **Target**: The canonical hostname of the machine providing the service

The configuration of the SRV records is often used to provide failovers and load sharing between the SIP servers. It is one of the easiest ways to get geographical redundancy in a SIP project.

**SIP services**

Beyond making and receiving calls, you can implement a series of SIP services. These services include, but are not limited to, Call Transfers, Call Pickup, Call Hold, Call Forward, and many others. Fortunately, RFC 5359 (SIP services) defines a standard way to accomplish these tasks. Most SIP phones comply with the way SIP services are implemented; however, to make them work, you need to make sure that all the components in the network support some specific RFCs. As an example, the call transfer requires the support of the REFER method defined in RFC 3515 and the Referred-By and Replaces headers defined in RFCs 3891 and 3892, respectively. If you intend to provide PBX-like services using a SIP proxy, you have to make sure that all the components, including phones and gateways, support it. SIP services are implemented in phones, gateways, media servers, and proxies. All the components must collaborate in order to implement each specific service. The following are some of the services defined in RFC 5359:

- Call Hold
- Consultation Hold
- Music on Hold
- Transfer—Unattended
- Transfer—Attended
- Transfer—Instant Messaging
- Call Forwarding—Unconditional
- Call Forwarding—Busy
• Call Forwarding—No Answer
• 3-Way Conference—Third Party Is Added
• 3-Way Conference—Third Party Joins
• Find-Me
• Incoming Call Screening
• Outgoing Call Screening
• Call Park
• Call Pickup
• Automatic Redial
• Click to Dial

It would be counterproductive to describe in detail each service here. Refer to the specified RFC for details.

The SIP identity

SIP servers are often employed to provide telephony services. However, there is a problem where Public Switched Telephone Network (PSTN) does not support SIP addresses containing domains and alphanumeric characters. To identify a caller identity for the PSTN, a few methods were created and applied.

The draft-ietf-sip-privacy-04 document describes the Remote-Party-ID header. While it has never became a standard, it is still quite popular among gateway manufacturers and service providers. See the following example:

Remote-Party-ID: "John" <sip:+554833328560@sip.com>; party=calling; id-type=subscriber; privacy=full; screen=yes

The preceding header sets the caller ID number as +554833328560 and caller name as "John"; it is a subscriber in the proxy, the identity was verified (screen=yes), and the number should not be present in the destination’s terminal (privacy=full). The draft specifies additional features and how to handle privacy requests. For the purposes of this book, Remote-Party-IDs will be used just for caller ID presentation.

The standard way to handle caller IDs and privacy came later in RFC 3325. It defines the P-Asserted-Identity, P-Preferred-Identity, and Privacy headers. See the following example:

P-Asserted-Identity: "John" sip:+554833328560@sip.com
P-Asserted-Identity: tel:+554833328560
To specify the caller ID to be present in the PSTN, you can use these headers. The gateway should match the type of caller ID and privacy used in your proxy. In an OpenSIPS server, you can add headers using the `append_hf` command. It is an extensive RFC and you can check the details in the document itself.

## The RTP protocol

The **Real-Time Protocol (RTP)** is responsible for the real-time transport of data such as audio and video. It was standardized in RFC 3550. It uses UDP as the transport protocol. To be transported, the audio or video has to be packetized by a *codec*. Basically, the protocol allows the specification of the timing and content requirements of the media transmission for the incoming and outgoing packets using the following:

- The sequence number
- Timestamps
- Packet forward without retransmission
- Source identification
- Content identification
- Synchronism

## Codecs

A *codec* is an algorithm capable of encoding or decoding a digital stream. The content described in the RTP protocol is usually encoded by a codec. Each codec has a specific use. Some have compression while others do not. G.711 is still the most popular codec and does not use compression. With 64 Kbps of bandwidth for a single channel, it needs a high-speed network, commonly found in **Local Area Networks (LANs)**. However, in **Wide Area Networks (WANs)**, 64 Kbps can be too expensive to buy for a single channel. Codecs such as G.729 and GSM can compress the voice packets to as low as 8 Kbps, saving a lot of bandwidth. To simplify the way you choose a voice codec, the following table shows the most relevant ones. Bandwidths do not consider the lower protocol layer headers. There are also video codecs where the most relevant ones are the H.264 series and VP8 from Google.
Chapter 1

Codec | Bandwidth | MOS | Env. | When to use
--- | --- | --- | --- | ---
G.711 | 64 Kbps | 4.45 | LAN/WAN | Use it for toll quality and broad support from gateways.
G.729 | 8 Kbps | 4.04 | WAN | Use it to save bandwidth and keep toll quality.
G.722 | 64 Kbps | 4.5 | LAN | Use it for high-definition voice.
OPUS | 6-510 Kbps | — | INTERNET | OPUS is the most sophisticated codec ever created. It spans from a narrowband audio to high-definition music.

There are other codecs such as G.723, GSM, iLBC, and SILK that are slowly losing ground to OPUS. OPUS is the codec adopted for the WebRTC standard. Obviously, you can dig a little more into codec details; there are dozens available, but I truly believe that the four described previously are the relevant choices at the time of this book being written. MOS is the **Mean Opinion Score** and defines the audio quality.

<table>
<thead>
<tr>
<th>MOS</th>
<th>Quality</th>
<th>Impairment</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Excellent</td>
<td>Imperceptible</td>
</tr>
<tr>
<td>4</td>
<td>Good</td>
<td>Perceptible but not annoying</td>
</tr>
<tr>
<td>3</td>
<td>Fair</td>
<td>Slightly annoying</td>
</tr>
<tr>
<td>2</td>
<td>Poor</td>
<td>Annoying</td>
</tr>
<tr>
<td>1</td>
<td>Bad</td>
<td>Very annoying</td>
</tr>
</tbody>
</table>

Source: ITU-T P.800 recommendation

**DTMF-relay**

There are three ways to carry DTMF in VoIP networks: inband as audio tones, named events on RTP as defined in RFC 2833, and signaling using the SIP INFO messages. RFC 2833 describes a method to transmit DTMF as named events in the RTP protocol. It is very important that you use the same method between user agent servers and clients.
Session Description Protocol

The Session Description Protocol (SDP) is described in RFC 4566. It is used to negotiate session parameters between the user agents. Media details, transport addresses, and other media-related information is exchanged between the user agents using the SDP protocol. Normally, the INVITE message contains the SDP offer message, while the 200 OK contains the answer message. These messages are shown in the following figures. You can observe that the GSM codec is offered, but the other phone does not support it. Then it answers with the supported codecs, in this case, G.711 u-law (PCM) and G.729. The rtpmap:101 session is the DTMF relay described in RFC 2833.

| Request-Line: INVITE sip:852098.8.30.36:42989 sip/2.0
| Message Header
| Message body

| Session Description Protocol Version (v): 0
| Owner/Creator, Session Id (a): root 10968 10968 IN IP4 8.8.1.4
| Owner Username: root
| Session Id: 10968
| Session Version: 10968
| Owner Network Type: IN
| Owner Address Type: IP4
| Owner Address: 8.8.1.4
| Session Name (s): session
| Connection Information (c): IN IP4 8.8.1.4
| Time Description, active time (t): 0 0
| Media Description, name and address (m): audio 17412 RTP/AVP 0 3 18 101
| Media Attribute (a): rtpmap:0 PCM/8000
| Media Attribute (a): rtpmap:3 GSM/8000
| Media Attribute (a): rtpmap:101 VAD/8000
| Media Attribute (a): fmt:0-16
| Media Attribute (a): silenceSupp:off - - -

INVITE (SDP Offer)
The SIP protocol and OSI model

It is always important to understand the voice protocols against the OSI model to situate where each protocol fits. The following diagram demonstrates this clearly:
The VoIP provider's big picture

This book was created using the VoIP provider as it is the most common use case. Before we start digging in the SIP proxy, it is important to understand all the components for a VoIP provider solution. A VoIP provider usually consists of several services. The services described here could be installed in a single server or multiple servers depending on the capacity requirements.

In this book, we will cover each one of these components, from left to right, in the chapters ahead. We will use the following picture in all the chapters in order to help you identify where you are:

![Diagram of VoIP provider's big picture]

The SIP proxy

The SIP proxy is the central component of our solution. It is responsible for registering the users and keeping the location database (maps IP to SIP addresses). The entire SIP routing and signaling is handled by the SIP proxy and it is also responsible for end-user services such as call forwarding, white/blacklist, speed dialing, and others. This component never handles the media (RTP packets); most media-related packets are routed directly from the user agent clients, servers, and PSTN gateways.
The user administration and provisioning portal

One important component is the user administration and provisioning portal. In the portal, the user can subscribe to the service and be capable of buying credits, changing passwords, and verifying his account. On the other hand, administrators should be able to remove users, change user credits, and grant and remove privileges. Provisioning is the process used to make it easier for administrators to provide automatic installation of user agents such as IP phones, analog telephony adapters, and SIP phones.

The PSTN gateway

To communicate to the PSTN, a PSTN gateway is usually required, except when you have a SIP trunk. The gateway will interface the PSTN using E1 or T1 trunks. To evaluate a good gateway, check the support of SIP extensions, such as RFC 3325 (Identity), RFC 3515 (REFER), RFC 3891 (Replaces), and RFC 3892 (Referred-by). These protocols will allow unattended transfers behind the SIP proxy; without them in the gateway, it might be impossible to transfer calls.

The media server

The SIP proxy never handles the media. Services such as Interactive Voice Response (IVR), voicemail, conference, or anything related to media should be implemented in a media server. There are many SIP servers fitting this purpose, such as Asterisk (http://www.asterisk.org/), FreeSWITCH (https://freeswitch.org/), Yate (http://yate.ro/), SEMS from IPTEL, and SilkServer for AG projects. The examples in this book will use Asterisk as it is, by far, the most popular.

The media proxy or RTP proxy for NAT traversal

Any SIP provider will have to handle NAT traversal for their customers. The media proxy is an RTP bridge that helps the users behind symmetric firewalls to access the SIP provider. Without them, it won't be possible to service a large share of the user base. You can implement a universal NAT traversal technique using these components. The media proxy can help you in the accounting correction for unfinished SIP dialogs that, for some reason, didn't receive the BYE message.
Accounting and CDR generation

An Authentication, Authorization, and Accounting (AAA) server can be used along with OpenSIPS. FreeRADIUS is a common choice. In several implementations, you can skip RADIUS and use SQL accounting. Some VoIP providers will leverage an existing AAA server while some others will prefer the low overhead MySQL accounting. Beyond accounting, there is CDR generation where the duration of the calls is calculated.

Monitoring tools

Finally, we will need monitoring, troubleshooting, and testing tools to help debug any problems occurring in the SIP server. The first tool is the protocol analyzer and we will see how to use ngrep, Wireshark, and TShark. OpenSIPS has a module called SIP trace, which we will use as well.

Additional references

The best reference for the SIP protocol is RFC 3261. To read the RFCs is a little bit boring and sleepy. (It is very good when you have insomnia.) You can find RFC at http://www.ietf.org/rfc/rfc3261.txt.

The OpenSIPS mailing lists can be found at http://www.opensips.org/Support/MailingLists.

There is a mailing list where you can post questions about SIP called SIP implementors at https://lists.cs.columbia.edu/mailman/listinfo/sip-implementors.

Summary

In this chapter, you learned what the SIP protocol is and its functionality. We saw different SIP components, such as the SIP proxy, SIP Registrar, user agent client, user agent server, and PSTN gateway. We also got acquainted with the SIP architecture and its main messages and processes.

In the next chapter, we will be introduced to OpenSIPS and its basic architecture and components.
Where to buy this book

You can buy Building Telephony Systems with OpenSIPS Second Edition from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.