Project 5: SIP: Session Initiation Protocol

Abrie Greeff
13557343

Overview of how SIP works

Overview
SIP (Session Initiation Protocol) is a protocol that many networking programs use to obtain information about a remote program/host before any actual data transfers occur. It is the initiation of the session, also called a handshake that occurs between any two programs.

The session consists of the sending and receiving of messages between the two programs. These programs each have user agents (UA) that handle the passing of the messages. Each UA consists of a user agent server (UAS) and a user agent client (UAC). When a program wishes to connect to a remote program his UAC will communicate with the remote UAS.

Communication rarely happens directly between the UAS and the UAC. There is one or more servers, called a proxy server, that sits between the UAs and relays the messages between the UAs. These proxy servers have access to a database called a Location Service that they use to resolve any information that may be needed. The Location Service may also be embedded in the proxy servers (see RFC 3261 page 25).

The information that is stored in the Location Service doesn't need to be in the proper format as long as the proxy knows how to handle the information. Information is added to the Location Service by another server called a Registrar. The information inserted must also comply with the format used by the Location Service. The Registrar can also be embedded in the proxy servers (see RFC 3261 page 25).

Messages

There are two types of SIP messages, a Request message and a Response message. Both have the same construct as shown here (see RFC 3261 page 26).

Request line/ Response line
Message Header
(empty)
Optional Body

Rows in the message are terminated with a carriage return line feed. A Request message uses the request line and a Response message the response line.
A request line has the format:

Method (space) Request-URI (space) SIP-version

Method, Request-URI and SIP-version may not contain any spaces otherwise this will make it difficult to parse the request line. The same applies to the response line when it's created.

There are six methods that can be used in the Method field:

REGISTER	Register with the Location Service
INVITE	Initiates a call
ACK	Confirms a final response for a invite
BYE	Terminates a call
CANCEL	Cancels searching and “ringing”
OPTIONS	Queries the remote host's capabilities

The Request-URI has the form sip:piet@someserver.com. Where sip indicates that the SIP protocol is being used, and this is followed by a user's name and the address where he may be found. The SIP-version is currently SIP/2.0.

A response line has the following format:

SIP-version (space) Status-code (space) Reason-Phrase

The SIP-version is again set to SIP/2.0 to indicate the current version. The status code can be a subclass of the following codes:

1xx	Provisional. Continue to process request
2xx	Success. Action successfully understood
3xx	Redirection. Further action needed
4xx	Client error. Server can't fulfil request
5xx	Server error. Server failed to fulfil a request
6xx	Global failure. Request can't be fulfilled anywhere

The most commonly used codes are as follows:

100	Trying
180	Ringing
182	Queued
200	OK
302	Moved temporarily

The Reason-phrase can be used for any miscellaneous info that supplement the Status-code.

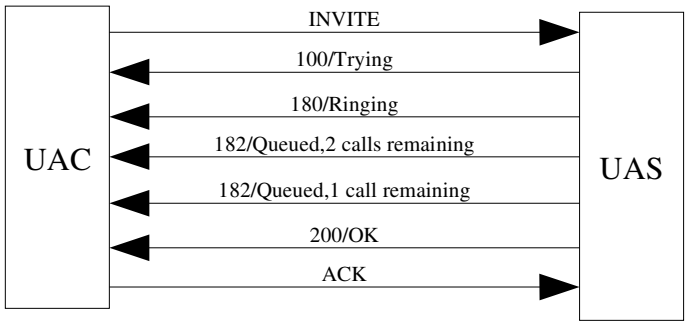
The Header field of the Request and Response messages follow the same syntax of the HTTP/1.1 protocol as defined in RFC 2616. There are six fields that have to be present in the header. They are defined as follows:

Via	The transport for the message
Max-Forwards	Maximum number of hops that the message may take
To	The target of the message
From	The initiator of the message
Call-ID	Identifier for a group of messages
Cseq	Sequence number of message

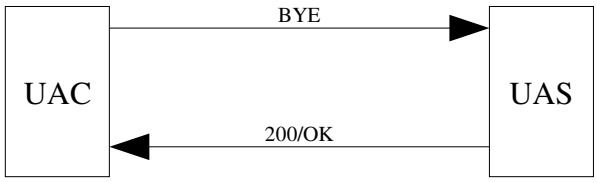
Here is an example from RFC 3261 page 11.

> Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
> Max-Forwards: 70
> To: Bob <sip:bob@biloxi.com>
> From: Alice <sip:alice@atlanta.com>;tag=1928301774
> Call-ID: a84b4c76e66710@pc33.atlanta.com
> CSeq: 314159 INVITE

SIP Session establishment without use of a Proxy Server
This is the order of messages sent between a UAC and a UAS to establish a session between them when direct communication takes place.

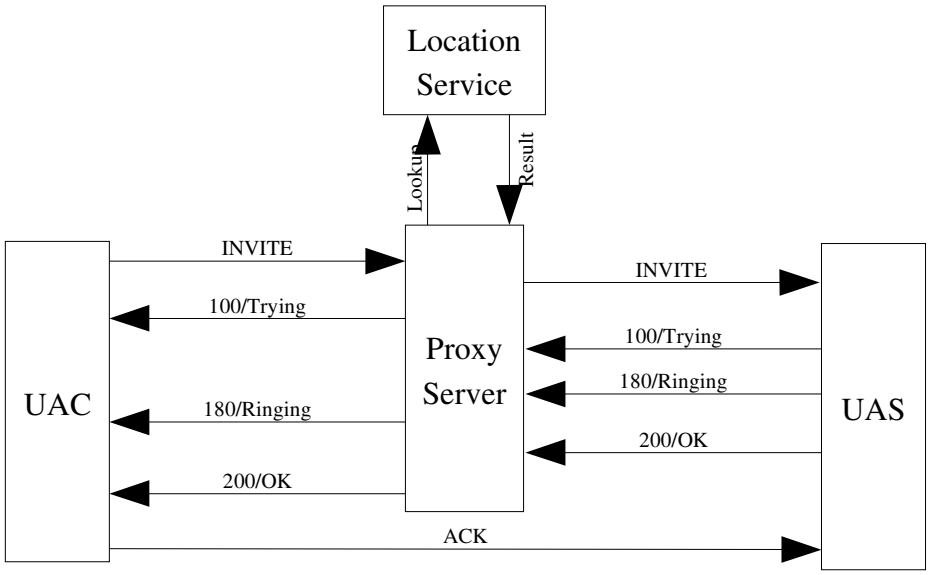


SIP Call termination
When a session is terminated between two programs the following occurs:



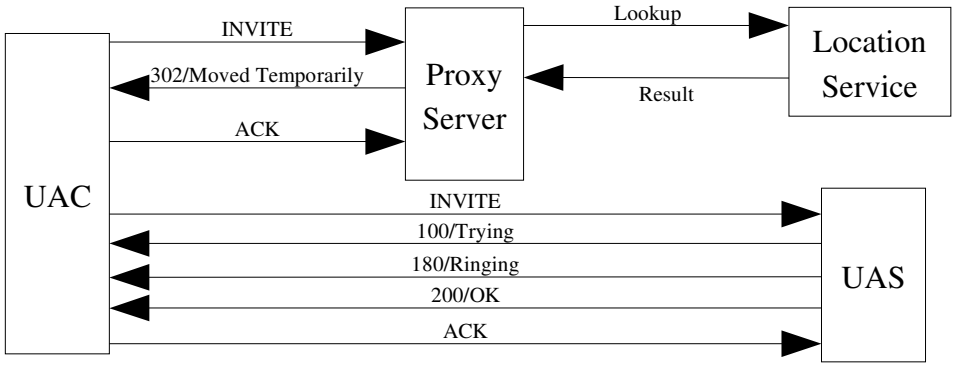
SIP Session establishment when a Proxy Server is present

When the session is established between two UAs who communicate over a proxy server, as is usually the case, the following occurs:



SIP session establishment when a user is on a different address

When a user has changed location from where he was first registered the following happens:



Registration

A user's info is added to the Location Service through the Registrar. The Registrar receives this info from a REGISTER Request message sent to it by a program.

My SIP Implementation

I used UDP as the protocol that I connect my UAs to each other. The Proxy Server always listens on port 5060 (see RFC 3261 page 144). The UAC and UAS ports are generated randomly. The Registrar is embedded into the Proxy Server and the Location Service is a thread in the Proxy Server. All messages sent/received and the sequence in which they are sent/received is RFC 3261 compliant as described in the previous section. When the program is started a REGISTER message is sent to the proxy server to register it self. The URI the program uses to call is the display name as seen in the next figure.

Additional Features Implemented

After the session has been initialized through SIP data transfer can occur. Thus I found it necessary to show that data transfer can happen. To show this I implemented a simple Voice over IP program. This is done by the transfer of audio between the connected programs. Because the Java audio mixer doesn't support synchronized audio in Linux at the moment I only implemented one way sound. The code is written for two way but unfortunately can't be tested. The program that makes the call can listen to audio from the other program's microphone. To enable sound the Audio check box needs to be ticked on both programs in their main interface.

A log of current message events can be seen by selecting the Log check box in the main interface.

Overview of Java classes

gui
This class is the swing gui of the main program

list
This is the object that is added to the Location Service and contains all relevant info about a program.

microphone
This implements the audio system and reads audio from the microphone and sends it over the network to a remote speaker.

ProxyServer
This class is the Proxy Server.

Registrar
This the Registrar and the Location Service thread.

SIP
This class handles all SIP messages by creating and parsing the messages.

speaker
This implements the audio system and receives audio over the network and plays it on the speakers of the system.

UAC
This is the User Agent Client thread of the main program.

UAS
This is the User Agent Server thread of the main program.

VoIP
This is the main program and connects all the different services.

Executing the code

To start a Proxy Server the following must be executed in a console.
java ProxyServer

To start the Voice over IP program the following must be executed in a console.
Java VoIP yourname
You will be prompted for the address of the Proxy Server, enter the address where your proxy is listening or leave it empty if it's the localhost.