Multimediakommunikation über IP

Michael Dienert

4. Februar 2021

Inhaltsverzeichnis

Zusammenfassung

zusammenfassung: t.b.d.

1 Session Initiation Protocol

Ziel von SIP: Ermöglichen jeglicher Multimediakommunikation über das InternetProtocol

- *offener* IETF-Standard ⇒ SIP wird in der RFC 3261 beschrieben
- Schaffung einer gemeinsamen Basis für Kommunikation über IP
- SIP ist modular und damit erweiterbar
- SIP ist im Aufbau ähnlich zu HTTP und funktioniert damit problemlos in IP-Netzen.
- SIP baut lediglich eine Sitzung (Session) zwischen den Teilnehmern auf, die eigentliche Übertragung der Audio-Daten übernimmt bei IP-Telefonie das RealtimeTransportProtocol RTP (2).

1.1 Was kann SIP

Lokalisieren des Gesprächspartners Welche Endgeräte kommen für die Kommunikation in Frage?

Erreichbarkeit des Gesprächspartners Möchte der angerufene Teilnehmer einen Kommunikationskanal aufbauen?

Verwendungsmöglichkeiten Welches Medium wird verwendet und welche Bandbreiten und weitere Parameter (z.B. Latenz) stehen zur Verfügung

Einrichten einer *Session* Rufsignal (Klingeln) und Festlegung der Parameter der Session bei Anrufer und Angerufenem.

Session Verwaltung Übertragung des eigentlichen Audio/Video-Streams, Abbau der Verbindung, Änderung von Parametern während der Sitzung, Start weiterer Dienste

1.2 Grundfunktion von SIP

Im ersten Beispiel (Bild 1) wird die Hauptaufgabe von SIP gezeigt: Lokalisieren eines Gesprächspartners, Signalisieren des Verbindungswunschs. Aushandeln der Session und Abbau der Session am Ende.

Die beiden Teilnehmer heissen Bob und Alice, die Protokollnachrichten von SIP werden mit eine F gefolgt von einer Nummer dargestellt.

Alice telefoniert mit einem Softphone, einer Software die die Audio-Hardware eines Rechners verwendet, Bob hat ein echtes SIP-Telefon.

Ausserdem verwendet das SIP-Trapez noch zwei *SIP-Proxy-Server*. Proxy-Server sind Dienste, die eine Anfrage von einem Benutzer entgegennehmen und dann an Stelle des Benutzers ihrerseits bei einem entfernten Dienst anfragen.

Z.B. nimmt ein http-Proxy http-Requests von einem Browser (http-User-Agent) entgegen und schickt dann anstelle des Browsers einen http-Request an den Ziel-Webserver, empfängt den http-Response und liefert die empfangene Web-Seite wieder an den Browser aus.

Die SIP-Proxies arbeiten sozusagen im Auftag von Bob und Alice.

Alice ruft bei Bob an und verwendet dabei seine SIP URI. Die SIP URI identifiziert hier ein Kommunikationsmittel, so wie eine http-URL einen Webserver identifiziert. URI steht für *Uniform Resource Identifier*, URL für *Uniform Resource Locator*.

SIP-URIs sehen ähnlich aus wie E-Mail-Adressen. Soll TLS verwendet werden, beginnt die URI mit `sips:`:

```
sip:bob@biloxi.com
sip:alice@atlanta.com
sips:bob@biloxi.com
```

SIP ist ähnlich aufgebaut wie HTTP und verwendet ein *request/response* Transaktions-Modell.

Die Transaktion beginnt damit, dass Alice' Softphone einen Invite-Request an `sip:bob@biloxi.com` sendet:

```
INVITE sip:bob@biloxi.com SIP/2.0
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
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142
```

Wie bei http beginnt der Request mit der Methode (INVITE). Auf die request-Zeile folgen mehrere Header-Zeilen. Das gezeigte Beispiel ist ein Minimal-Request.
Die Header-Felder im Detail:

Via: das ist der Host (Adresse) an den Alice die Antwort auf den Request erhalten möchte. Der Branch-Parameter ist eine ID.

To: enthält einen Anzeigenamen ("Bob") und eine SIP- oder SIPS-URI.

From: enthält ebenfals Anzeigenamen und SIP-URI sowie eine Zufallszahl für Identifikationszwecke.

Call-ID: schon wieder eine ID: Zufallszahl@host. Die Kombination aus Call-ID, To und From identifiziert die Peer-to-Peer-Verbindung zwischen Bob und Alice. Das wird als Dialog bezeichnet.

CSeq: das ist einfach eine Sequenz-Nummer, wird bei jedem neuen request innerhalb eines Dialogs hochgezählt.

Contact: das gibt die **direkte** Verbindungsroute zu Alice an. Wichtig ist der Unterschied zu Via: Via: gibt an, wohin der Response zu diesem Request geschickt werden soll, Contact: gibt an, wohin *zukünftige* Requests hingeschickt werden sollen.

Max-Forwards: ist vergleichbar mit dem TTL-Feld aus IPv4

Auffällig ist, dass die Details der Session (Art des Mediums, verwendete Codecs, Sampling-Rate usw.) nicht im Header auftauchen. Diese Daten werden mit dem *Session Description Protocol* (SDP) im Message-Body des SIP-Requests übertragen
Da Alice Softphone nicht weiss, wo Bob oder der SIP-Server in der Domäne `biloxi.com` steht, wird die INVITE-Nachricht an den Server `atlanta.com` geschickt der Anfragen von Alice Softphone beantwortet. Die Adresse bzw. der Hostname dieses Servers muss dabei auf Alice Rechner eingetragen sein.

1.3 Proxy Server

`atlanta.com` im Beispiel ist ein **Proxy Server**, d.h. ein Dienst, der Anfragen entgegen nimmt und diese im **Auftrag** weiterleitet.
Hier im Beispiel empfängt er den INVITE request und sendet eine Meldung `100 (Trying)` zurück, damit Alice sicher weiss, dass der Proxy in ihrem Auftag arbeitet.

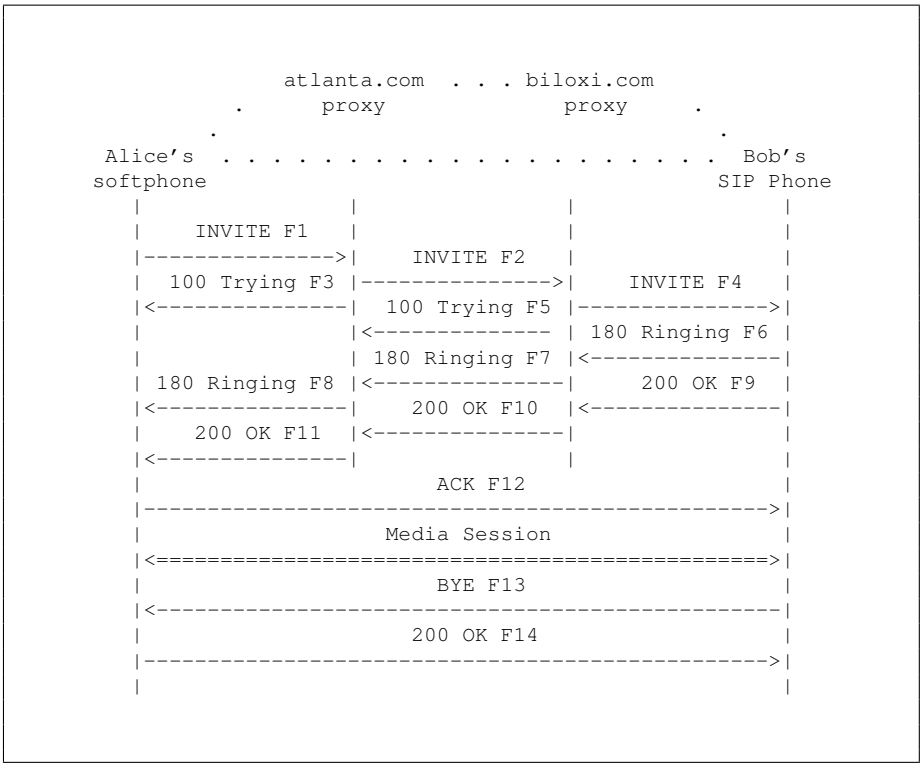

Abbildung 1: Verbindungsaufbau mit dem SIP-Trapez(Quelle: RFC3261)

Allgemein bestehen solche Meldungen aus 3 Ziffern, gefolgt von einer Beschreibung. Ausserdem enthalten sie dieselben To, From, Call-ID, CSeq und Branch Parameter wie das INVITE, so dass Alice Softphone die Meldung dem INVITE zuordnen kann.

Der Proxy atlanta.com lokalisiert nun den Proxy biloxi.com über eine DNS-Abfrage, ähnlich der MX-Mailserver-Namensauflösung. Der Datentyp dieser DNS-Einträge heisst SRV und auch hier gibt es wie bei MX-Recors eine Priorisierung [1] .

Hier ein Beispiel:

```
  dig _sip._udp.sip.voice.google.com SRV

  ;; ANSWER SECTION:
_sip._udp.sip.voice.google.com.       300 IN       SRV       20 1 5060 sip-anycast-2.voice.google.com.
_sip._udp.sip.voice.google.com.       300 IN       SRV       10 1 5060 sip-anycast-1.voice.google.com.
```

`atlanta.com` leitet nun den INVITE-Request von Alice an `biloxi.com` weiter und fügt ihm dabei einen zweiten Via-Header mit seiner eigenen Adresse hinzu.

`biloxi.com` antwortet mit der Meldung `100 Trying`

Der Proxy von Bob (biloxi.com) sucht Bobs aktuelle IP-Adresse aus einer internen Datenbank, fügt dem INVITE ein weiteres Via-Feld hinzu und sendet das INVITE an Bob.

Bobs Telefon klingelt nun und meldet das dem Proxy mit der Meldung `180 Ringing`.

[1] Zusätzlich noch Parameter für die Lastverteilung

Dieses Ringing-Signal wird über die Proxies bis zu Alice Softphone geleitet und Alice bekommt daraufhin ein Signal auf ihren Lautsprecher.

Beim Zurücksenden der Ringing-Meldung kommen nun die Via-Felder zum Einsatz: der Rückweg muss nicht über DNS-Anfragen aufgelöst werden und auch mussten sich die Proxies den Verbindungszustand der hinlaufenden INVITE-Meldung nicht merken, denn die Via-Felder enthalten den jeweils nächsten Knotenpunkt auf dem Rückweg. Bei jedem dieser Hops wird wieder das eigene, oberste Via-Feld vom Stapel entfernt.

Wenn Bob das Gespräch annimmt, wird eine 200 OK Meldung zurückgeschickt. Diese Meldung transportiert im Message-Body ein SDP-Paket, das die wichtigen Parameter von Bobs IP-Telefon enthält (Codec, Sampling-Rate, Wave-Format, usw.).

Hier ein Beispiel für die 200-OK-Meldung:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP server10.biloxi.com
   ;branch=z9hG4bKnashds8;received=192.0.2.3
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com
   ;branch=z9hG4bK77ef4c2312983.1;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com
   ;branch=z9hG4bK776asdhds ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e667l0@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:bob@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131
```

Schlussendlich kann Alice Softphone nun das Gespräch annehmen. Dazu sendet es eine ACK-Meldung an Bob. Hier im Beispiel werden dabei die Proxies umgangen: durch die Contact-Header-Felder haben beide Telefone die IP-Adressen des jeweiligen Gegenüber gelernt.

Nach dem ACK beginnt der eigentliche Transport des Audio/Videostreams.

Legen die Gesprächspartner auf, wird die Verbindung vom Angerufenen (Bob) ausgehend abgebaut.

2 Realtime Transport Protocol

Das *Realtime Transport Protocol* gehört zur Anwendungsschicht: RTP-Pakete werden als Nutzlast von UDP-Datagrammen versendet.

RTP wurde eingeführt, damit Internet-Telefonie-Software *herstellerunabhängig* wird: würde jeder Hersteller seine Audio- oder Video-Streamdaten mit einem eigenen Standard mit UDP transportieren, müssten sich Gesprächspartner beim Telefonieren auf einen einheitlichen Softwarehersteller einigen.

Als Transportprotokoll wird **UDP** und nicht **TCP** verwendet. Bei TCP sorgt das Protokoll dafuer, dass bei der Übertragung nicht ein einzelnes Byte verloren geht. Ist ein TCP-Segment falsch empfangen worden, wird es komplett neu übertragen, was bei einer Anwendung, die einen kontinuierlichen Byte-Strom benötigt zu Aussetzern oder Verzögerungen führen würde.

Bei UDP findet keine Überprüfung und auch keine Wiederholung der Übertragung (Re-Transmission) statt. Werden einzelne Bytes eines Audio- oder Videostreams falsch übertagen, hat man kleine Verluste, z.B. in der Audioqualität. Das ist aber tolerierbar und nicht so störend, wie eine lange Verögerung oder gar längere Aussetzer.

3 WebRTC

Das *Web Real-Time Communication protocol* ist eine offene, standardisierte Javascript-API. Damit lassen sich Javascript-Anwendungen entwickeln, bei denen eine direkte Kommunikation zu anderen Geräten aus einer Webseite heraus aufgebaut wird.

Die Idee dazu ist: bisher konnten HTTP-User-Agents (aka *Web-Browser*) mittels http Webseiten von Servern anfordern:

Request:

```
GET /index.html HTTP/1.1
Host: dt.wara.de
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

Response:

```
HTTP/1.1 200 OK
Date: Mon, 01 Feb 2021 09:48:13 GMT
Server: Apache/2.4.25 (Debian)
Last-Modified: Sun, 17 May 2020 18:43:09 GMT
ETag: "6ec-5a5dc6af33ea6-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 616
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

Mittels Web-RTC kann ein Web-Browser nun auch direkt von einem andern Browser einen Audio/Video-Stream anfragen und wiederum einen Stream zurücksenden (Peer-Connection).