07/09

Anzeige: NEU! STRATO MultiServer
Ein Server, der viele ersetzt!
www.strato.de siehe Seite 2

LINUX MAGAZIN

PDFs konvertieren und editieren

Kostenlose Linux-Tools, die ohne Akrobatik PDF-Dateien auf Linie bringen. S. 56

Fotoalbum 2.0

Mit Python und Fuse ein eigenes Dateisystem programmieren. S. 109

Desktop-Web

Mit Java FX versucht Sun Silverlight und Flash auszustechen S. 98

Big-Bang-Linux

Das Handwerkzeug, das Forscher lieben

- Chemical Development Kit statt Bunsenbrenner S. 32
- Linux in der Raumfahrt S. 37
- Zwei Kameras und ein Set von 3D-Bibliotheken erkennen die Gesten von Personen S. 40
- Deutsche Wetterfrösche und ihre heißen Cluster S. 44

- Nützliche Open-Office-Erweiterungen S. 52
- NAT zu Gunsten von SIP austricksten S. 76
- Nconf, das Nagios-GUI für eilige Admins S. 72

Nagios-Rebellion aus Deutschland: Ethan Galstad verliert nach Fork jede Menge Entwickler. S. 29

www.linux-magazin.de

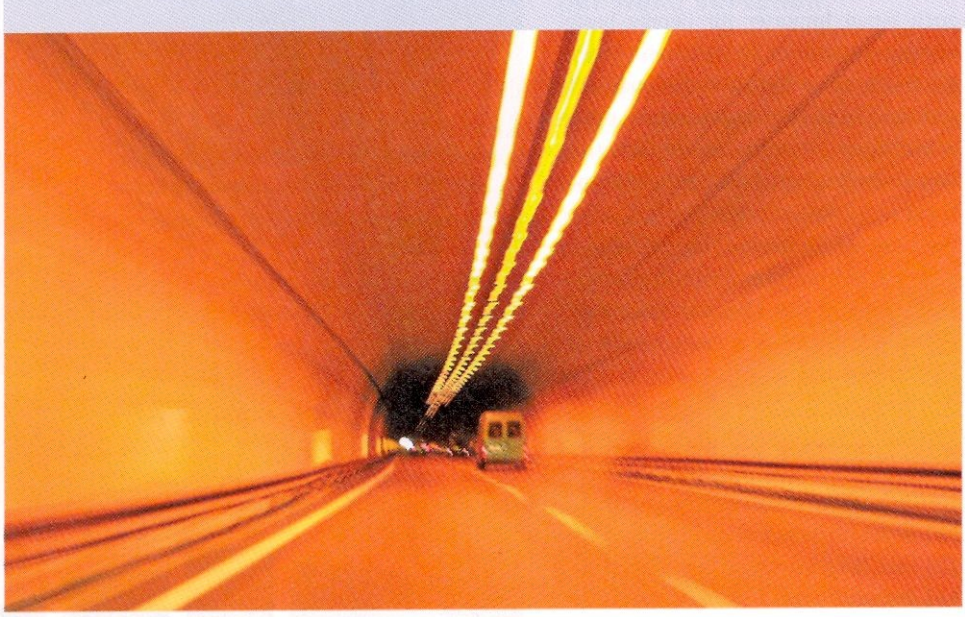
Deutschland	Österreich	Schweiz	Benelux	Spanien	Italien
€ 5,80	€ 6,65	sfr 11,60	€ 6,85	€ 7,85	€ 7,85

4 192587 305800 07

SIP-Telefonie hinter NAT

Telefonieren im Tunnel

Wer für Voice-over-IP-Telefonate auf das offene SIP-Protokoll setzt, stößt an Grenzen, wenn der Client dank NAT nur eine private IP-Adresse hat. SIP-Pakete enthalten nämlich auch im Datenteil Adress- und Port-Informationen, von denen das NAT-Gateway nichts weiß. Michael Hirschbichler



Das Session Initiation Protocol (SIP) zählt neben Skype zu den populärsten Techniken, Voice-over-IP-Verbindungen aufzubauen. Durch seine offene Standardisierung (IETF RFC 3261 [1]) ist es zudem einfach, eigene Anwendungen für dieses Protokoll zu entwickeln.

SIP-Grundlagen

SIP ist nur für die Signalisierung, zum Beispiel beim Gesprächsauf- und -abbau zuständig. Den Transport der eigentlichen Sprachdaten übernimmt das ebenfalls offene Realtime Transfer Protocol (RTP), das RFC 3550 beschreibt [2].

Einen Grundlagenartikel zu SIP hat das Linux-Magazin in Ausgabe 08/2004 veröffentlicht [7].

Listing 1 zeigt den im SIP-RFC als Beispiel verwendeten SIP-»Invite«-Request, der ein VoIP-Telefonat einleitet und aus zwei Teilen besteht: Der erste Teil der Nachricht enthält die SIP-Header, im zweiten Teil – durch eine Leerzeile abgetrennt – folgt das Session Description Protocol (SDP) [3] als SIP-Payload. Dieser SDP-Teil beschreibt den Audio-Stream, enthält also Daten wie den Codec und den offenen Port (im »m«-Header) sowie die IP-Adresse oder den Namen des Hosts.

Die erste Zeile der SIP-Nachricht definiert den Request und die zu kontaktierende URI (Uniform Resource Identifier, ähnlich einer Mailadresse). Der »Via«-Header speichert die einzelnen Hops, über welche die Pakete von Alice zu Bob laufen (Abbildung 1). Für jeden Hop entsteht ein zusätzlicher »Via«-Header. Diese Header machen es möglich, die Antwort über denselben Pfad zurückzusenden.

Der »To«-Header enthält die URI der kontaktierten Person, in der »From«-Zeile steht die öffentliche URI des Absenders. »Call-ID« und »Cseq« sind eindeutige Identifier für den aktuellen Dialog. In »Contact« steht, unter welcher Adresse das Endgerät (User Agent, UA) derzeit auf SIP-Requests lauscht. Registriert der Anwender die URI »alice@atlanta.com« mehrfach auf verschiedenen Endgeräten, so enthält der »Contact«-Header die Daten des Endgeräts, das den Request abgesendet hat – im Beispiel »pc33.atlanta.com«. »Content-Type« und »Content-Length« definieren Typ und Größe des mitgesendeten Payloads. Abbildung 1 zeigt, wie zwei VoIP-Clients ein Gespräch über SIP initiieren: Die Botschaften laufen über mehrere Hops, und es sind mehrere Nachrichten notwendig.

Alice ruft Bob

Der Client mit der URI »alice@atlanta.com« möchte »bob@biloxi.com« anrufen. Er unterstützt mehrere Mediencodecs (erkennbar am SDP-Header »m= ? 0 1 3 99«) und hat einen für Medien offenen Socket »pc33.atlanta.com:3456«. Da Alice nicht weiß, an welchem Host Bob registriert ist, kontaktiert sie entweder zuerst den Proxy von Bob (wie in Abbildung 1, Schritt 1) oder sendet den SIP-Request zuerst an den eigenen Proxy.

Bobs Proxy stellt nun über seine Location-Datenbank den Standort von Bob fest und leitet den Request dorthin weiter (3). Eine alternative, aber unübliche Methode ist die direkte Kontaktaufnahme zwischen Alice und Bob ohne die Hilfe von Proxies. Die ist aber nur möglich, wenn Alice die aktuelle Location-Information von Bobs Client kennt.

Nachdem Alice den Request an den Proxy geschickt hat, erhält sie von ihm die Antwort »100 Trying« (2): Daran erkennt sie, dass der nächste Hop den Request erhalten hat. Trifft dieses »Trying« nicht ein, überträgt Alice erneut – unabhängig davon, ob sie die Message über UDP oder TCP verschickt hat. Hat Bob den Request empfangen, sendet er den Status »180

Ringing« (4), und sein Endgerät gibt ein Signal aus. Diese »Ringing«-Antwort läuft über die in den »Via«-Headern gesammelten Hops von Bob bis zu Alice zurück. Nimmt Bob den Anruf entgegen, sendet er »200 OK« (6). Das »OK« enthält anders als die früheren Antworten einen »Contact«-Header und einen SDP-Payload, der Alice die unterstützten Codecs und den für Medien offenen Socket mitteilt.

Erreicht das »OK« (7) Alice, sendet diese als Bestätigung eine »ACK«-Nachricht (8) direkt an die URI, die Bob im »Contact«-Header mitgesendet hat. Am »ACK« erkennt Bob, dass Alice sein »OK« erfolgreich empfangen hat. Nach diesem Drei-Wege-Handshake hat Alice nun den Medien-Socket von Bob, dieser hat den von Alice (durch den »Invite«-Request), und die Medienübertragung beginnt. Beendet Bob das Gespräch, sendet er einen »Bye«-Request (9) an die URI in Alices »Contact«-Header. Auch dabei lässt er die dazwischenliegenden Stationen aus.

NAT-Problematik

Befindet sich der Anrufer in einem privaten Subnetz hinter einem NAT-Gateway, enthält der »Invite«-Request mehrfach private IP-Adressen (**Listing 2**), die von der Gegenseite aus nicht erreichbar sind. Das Gateway, das in den Paketen die Adressumsetzungen auf Layer 3 (IP) beziehungsweise auf Layer 3 und 4 (TCP/UDP, NAPT: Network Address Port Translation) durchführt, ist von Haus aus nicht in der Lage, IP-Adressen in höheren Layern zu erkennen und zu modifizieren.

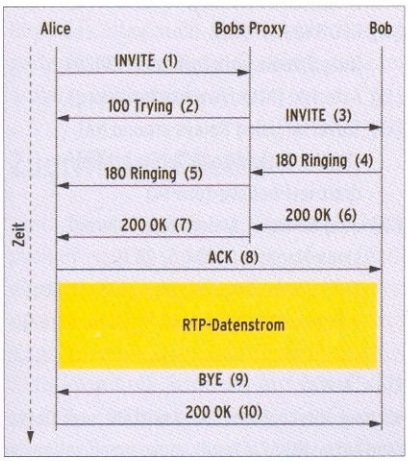


Abbildung 1: Wenn Alice mit dem »Contact«-Header in der Nachricht »200 OK« den SIP-Socket von Bob erhält, kann sie das ACK (8) direkt an Bob senden.

Die am weitesten verbreitete NAPT-Technik, das symmetrische NAPT, legt dynamisch ein Mapping zwischen der internen Source-IP-Adresse und dem Port sowie der öffentlichen Ziel-IP-Adresse und dem Ziel-Port an. Diese Zuordnung bleibt für eine Zeit erhalten, wird aber nach Ablauf einer gewissen Zeitspanne ohne Datenverkehr gelöscht. In dieser Zeit leitet NAPT Antworten von der öffentlichen Adresse auch immer korrekt an die Adresse innerhalb des NAPT weiter. Da aber der SIP-Client Antworten nicht an die Quelladresse, sondern standardmäßig an die im »Via«-Header angegebene sendet, enden diese Nachrichten oft im digitalen Nirwana. Für dieses Problem gibt es mehrere Ansätze.

STUN

Über das klassische STUN-Protokoll (Simple Traversal of UDP through NAT) **[4]** ist es möglich, die eigene öffentliche IP-Adresse und die Port-Nummer festzustellen. Der STUN-Client, der meist in die Telefon-Applikation integriert ist, sendet ein Paket („Binding Discovery") an einen STUN-Server, und dieser antwortet mit einem „Binding-Response"-Paket, das im Protokollattribut »MAPPED-ADDRESS« die Adresse enthält, die das NAT-Gateway nach außen verwendet. Die dadurch erhaltene, öffentlich sichtbare IP-Adresse setzt die Anwendung nun im SIP-Paket an jenen Stellen ein, wo sonst die private Adresse stehen würde.

Diese Technik funktioniert grundsätzlich recht gut – jedoch nicht beim symmetrischen NAPT-Gateway: Durch das Mapping im NAPT-Gateway akzeptiert dieses auf dem via STUN ermittelten eigenen, öffentlichen IP-Socket nur Pakete, die direkt vom STUN-Server kommen. Beim Senden der SIP-Nachricht an den SIP-Proxy im Gateway ist aber ein anderes Mapping entstanden, das nicht mit dem in der SIP-Nachricht übermittelten Daten übereinstimmt.

Symmetric Response

Um Anwendern das eventuell sowieso erfolglose Konfigurieren eines STUN-Servers zu ersparen, erweitern SIP-Provider ihre Proxies um das so genannte Symmetric Response Routing: Die vergleichen dann IP-Adresse und Port aus dem »Via«-Header eines empfangenen »Invite«-Requests mit Source-IP und -Port. Stimmen sie nicht überein, ergänzt der Proxy den »Via«-Header um den »Received«-Parameter und schreibt den »Contact«-Header anschließend so um, dass dieser danach den öffentlich erreichbaren Socket des NAT-Gateways enthält. ▶

Listing 1: »Invite«-Request mit SIP- und SDP-Parts

```
01 INVITE sip:bob@biloxi.com SIP/2.0
02 Via: SIP/2.0/UDP pc33.atlanta.com:5060;
   branch=z9hG4bKnashds8
03 Max-Forwards: 70
04 To: Bob <sip:bob@biloxi.com>
05 From: Alice <sip:alice@atlanta.com>;
   tag=1928301774
06 Call-ID: a84b4c76e66710
07 CSeq: 314159 INVITE
08 Contact: <sip:alice@pc33.atlanta.com:5060>
09 Content-Type: application/sdp
10 Content-Length: 142
11
12   v=0
13   o=alice 53655765 2353687637 IN IP4
   pc33.atlanta.com
14   s=-
15   t=0 0
16   c=IN IP4 pc33.atlanta.com
17   m=audio 3456 RTP/AVP 0 1 3 99
18   a=rtpmap:0 PCMU/8000
```

Listing 2: »Invite«-Request mit privaten IP-Adressen

```
01 INVITE sip:bob@biloxi.com SIP/2.0
02 Via: SIP/2.0/UDP 192.168.1.10:5060;
   branch=z9hG4bKnashds8
03 Max-Forwards: 70
04 To: Bob <sip:bob@biloxi.com>
05 From: Alice <sip:alice@atlanta.com>;
   tag=1928301774
06 Call-ID: a84b4c76e66710
07 CSeq: 314159 INVITE
08 Contact: <sip:alice@192.168.1.10:5060>
09 Content-Type: application/sdp
10 Content-Length: 142
11
12   v=0
13   o=alice 53655765 2353687637 IN IP4
   192.168.1.10
14   s=-
15   t=0 0
16   c=IN IP4 192.168.1.10
17   m=audio 5678 RTP/AVP 0 1 3 99
18   a=rtpmap:0 PCMU/8000
```

Listing 3 zeigt einen »Invite«-Request, den der SIP-Proxy bereits modifiziert hat: 18.17.16.15 ist die öffentliche Adresse des Gateways, 20.22.24.26 und 20.22.24.27 sind die des SIP-Proxies und des Medien-Proxies. Da das NAT-Mapping zu diesem Socket nur für den Proxy des Providers gilt, müssen auch alle zukünftigen Requests des SIP-Dialogs über ihn laufen. Dazu ergänzt der Proxy einen »Record-Route«-Header mit seiner IP-Adresse.

Das nächste Problem bereitet der RTP-Port, der zwar im SDP-Part steht, für den es aber noch kein NAT-Mapping gibt. Hier helfen sich Provider, indem sie einen öffentlichen Medien-Proxy verwenden: Sie ersetzen die IP-Adressen und Ports im SDP-Header durch Adresse und Port des eigenen Medien-Proxies, und zwar sowohl im »Invite«-Request als auch im SDP-Part der folgenden »OK«-Antwort.

Um das NAT-Mapping dauerhaft zu erhalten, setzen die Clients die Zeit bis zur erneuten Registrierung auf einen sehr kurzen Zeitraum von ein bis zwei Minuten herunter. So senden sie immer wieder Registrierungs-Requests und erneuern damit den Eintrag in der NAT-Tabelle. Das gelingt aber nur, wenn der erste Hop im SIP-Request der SIP-Proxy des eigenen Providers ist. Bei dieser Technik laufen nicht nur die Signalisierungs-, sondern auch die Mediendaten über den Provider.

Application Layer Gateway

Unterstützt der Provider kein Header-Rewriting in der oben beschriebenen Form, bietet sich noch das in vielen ADSL-Modems und WLAN-Routern eingebaute SIP-NAT-Helper-Modul an: Da eigentlich nur das Gateway, welches das NAT durchführt, weiß, welcher interne mit welchem externen Socket kommuniziert, ist auch nur dieses Gateway in der Lage, auf dem Application Layer die Adressen zu modifizieren.

Das Application Layer Gateway erkennt SIP-Nachrichten und ersetzt die Adressen und Ports in SDP- und SIP-Headern entsprechend der Mapping-Tabelle. Diese Lösung arbeitet an sich recht zuverlässig. Zu Problemen kann es aber bei mehrstufigen NATs kommen: Der WLAN-Router übersetzt Adressen in der Kommunikation mit dem ADSL-Modem, das ebenfalls NAT durchführt. Das führt oft zu schwer behebbaren Problemen.

Lokaler SIP-Registrar

Einen lokalen SIP-Registrar zu verwenden, bildet einen noch weiter führender Ansatz. Den beherrscht zum Beispiel die aktuelle Fritzbox. Hier kommuniziert der lokale User Agent Client (UAC) nicht direkt mit dem Provider. Stattdessen fungiert der im NAT-Gateway implementierte SIP-Registrar als so genannter „Back-to-Back User Agent", also als ein User Agent, der in eine Richtung die Rolle eines Clients, in die andere Richtung die eines User Agent Servers (UAS) übernimmt (siehe [1], Abschnitt 6). Diese Applikation ist gleichzeitig ein Medienproxy, was zusätzlichen Spielraum für weitere Entwicklungen verschafft. So wäre es möglich, die RTP-Pakete zu recodieren oder in Richtung Internet mit SRTP (Secure Real-time Transport Protocol) zu verschlüsseln. Das gleiche Ergebnis ist auch mit einer lokalen Asterisk-Installation [5] erreichbar.

Weitere Ansätze

Eine Erweiterung des STUN-Ansatzes ist das als IETF-Draft publizierte TURN-Protokoll (Traversal Using Relay NAT) [6]. Es führt STUN weiter, indem es nicht nur die SIP-Signalisierung berücksichtigt, sondern auch eine Lösung für die RTP-Medien einführt. Ein STUN-Server übernimmt dabei zusätzlich die Aufgaben eines RTP-Relays. Diese Relay-Technik ist aber sehr ressourcenintensiv und kann bei falscher Hardware-Dimensionierung zu Verzögerungen und zu Jitter führen, was wiederum die empfundene Gesprächsqualität reduziert.

Eine Gesamtlösung der von NAT-Gateways verursachten Probleme bieten VPNs: Die freie Implementierung OpenVPN baut beispielsweise nicht nur einen virtuellen Kanal zu einem Server auf, sondern verschlüsselt die Daten auch mit TLS. Durch diesen Tunnel erhält der Client eine Adresse aus dem VPN-IP-Adressbereich, welche die Clients dann auch bei der jeweiligen SIP-Signalisierung verwenden.

Fazit

SIP hinter einem NAT zu nutzen, ist schwierig. Es gibt aber mehrere Lösungsansätze. Für die Auswahl eines Ansatzes ist entscheidend, welche Techniken der SIP-Provider und die eingesetzte Hardware bereits unterstützen. (*hge*) ■

Listing 3: Vom SIP-Proxy modifizierter »Invite«-Request

```
01 INVITE sip:bob@biloxi.com SIP/2.0
02 Via: SIP/2.0/UDP 20.22.24.26:5060;
03 Via: SIP/2.0/UDP 192.168.1.10:5060;
04    received=18.17.16.15:25588;branch=
   z9hG4bKnashds8
05 Record-Route: <sip:20.22.24.26;lr>
06 Max-Forwards: 70
07 To: Bob <sip:bob@biloxi.com>
08 From: Alice <sip:alice@atlanta.com>;
   tag=1928301774
09 Call-ID: a84b4c76e66710
10 CSeq: 314159 INVITE
11 Contact: <sip:alice@18.17.16.15:25588>
12 Content-Type: application/sdp
13 Content-Length: 142
14
15   v=0
16   o=alice 53655765 2353687637 IN IP4
     20.22.24.27
17   s=-
18   t=0 0
19   c=IN IP4 20.22.24.27
20   m=audio 20333 RTP/AVP 0 1 3 99
21   a=rtpmap:0 PCMU/8000
```

Infos

[1] RFC 3261 (SIP): [http://rfc-ref.org/RFC-TEXTS/3261/chapter24.html]
[2] RFC 3550 (RTP): [http://tools.ietf.org/html/rfc3550]
[3] RFC 4566 (SDP): [http://tools.ietf.org/html/rfc4566]
[4] RFC 3489 (STUN): [http://tools.ietf.org/html/rfc3489]
[5] Asterisk: [http://www.asterisk.org]
[6] Traversal Using Relays around NAT (TURN), [http://tools.ietf.org/html/draft-ietf-behave-turn-14]
[7] Jörg Reitter: „Anders telefonieren", Linux-Magazin 08/04, S. 28 ff., [http://www.linux-magazin.de/heft_abo/ausgaben/2004/08/anders_telefonieren]

Der Autor

Michael Hirschbichler ist Assistent und Doktorand am Institut für Breitbandkommunikation der Technischen Universität Wien sowie Projektmitarbeiter am Forschungszentrum Telekommunikation Wien und bei der Mobilkom Austria.