TECHNISCHE UNIVERSITÄT CHEMNITZ

Universitätsrechenzentrum

Studienarbeit

AsteriskVoIPErprobung

Autor	Holger Schildt <holger.schildt@informatik.tu-chemnitz.de>
Betreuer	Dr. Ludwig Wolf <ludwig.wolf@hrz.tu-chemnitz.de>
Datum	1. Oktober 2003

Inhaltsverzeichnis

1 Was ist Asterisk

1.1 Motivation

Als es im Jahre 1995 der israelische Firma VocalTec[1] gelang, eine Sprachverbindung zwischen zwei Rechnern aufzubauen, sahen Visionäre darin den Untergang der Telefongesellschaften, die gegen die mächtige und stark wachsende Computerbranche keine Chance mehr hatten. Die Idee ist sehr verlockend: weltweites telefonieren zum Ortstarif, da nur noch die Einwahl zum Internetprovider bezahlt werden muss.
Aber diese Expansion der paketorientierten Sprachvermittlung, im folgenden nur noch *Voice over Internet Protocol* (VoIP) genannt, blieb aus. Als Gründe hierfür sind die meist begrenzten Anbindungen der Endnutzer ans Internet zu nennen, die eine Übermittlung von Sprache in befriedigender Qualität verhindern. Außerdem drängen viele offene und herstellerspezifische Standards auf dem Markt, die Telefonieren über VoIP realisieren sollten, aber in Konkurrenz zueinander stehen. Im weiteren sind die wenigsten Endanwender bereit, für ein Telefongespräch den Rechner zu starten und über Mikrophon und Lautsprecher zu telefonieren.
Im Laufe der Zeit kristallisierte sich aber ein weiterer Einsatzbereich von VoIP heraus: Historisch bedingt existieren innerhalb der meisten Unternehmen zwei Kommunikationsinfrastrukturen. Auf der einen Seite steht die Infrastruktur für die Datenkommunikation (LAN) auf der anderen Seite das Netz der Nebenstellenanlagen mit der PBX-Anlage im Mittelpunkt.
Bei diesen Unternehmen ist das Datennetz in den meisten Fällen in einer Größenordnung dimensioniert, die der technische Umsetzung von VoIP nicht im Weg steht.
Falls sogar mehrere Standorte eines Unternehmens existieren, können diese über das Internet miteinander verbunden werden, falls eine dementsprechend dimensionierte Anbindung ans Internet existiert.
Der Markt für Telekommunikationstechnik stellt sogar mehrere Lösungen bereit, mit denen über die Telefonleitung Kommunikationspartner mit herkömmlichen Telefongeräten angerufen werden können. Es wurde sogar das Problem gelöst, dass die Nutzer nicht über den Computer telefonieren müssen, indem die Hersteller VoIP-Telefone entwickelt haben, die äußerlich den Herkömmlichen sehr ähnlich sind, aber statt einen Anschluss ans Telefonnetz einen Netzwerkanschluss besitzen.
Die Vorteile einer VoIP-Lösung liegen klar auf der Hand: statt einer Telekommunikationsanlage mit dem passenden Netz kann das Datennetzwerk genutzt werden. Das Marktforschungsunternehmen Gartner Group[2] errechnete in diesem Zusammenhang, dass sich bei einer Nutzung von VoIP die Kosten eines Computerarbeitsplatzes um 30% senken lassen können.
Aber auch die Nachteile einer Telekommunikationsanlage, die das Datennetzwerk der Einrichtung nutzt, müssen beleuchtet werden: Eine handelsübliche Telekommunikationsanlage ist ausfallsicherer im Gegensatz zu einem Kommunikationsserver. Ein Ausfall des Servers ist gleichzusetzen mit einem Komplettausfall der gesamten Telekommunikationsinfrastruktur.
Ein weiteres Problem ist die Tatsache, dass sich noch kein VoIP-Standard durchgesetzt hat. Als große Vertreter der VoIP-Protokolle sind SIP und H.323 zu nennen, die von fast allen Hardwareanbietern unterstützt werden. Im weiterem bietet z.B. die Netzwerkfirma Cisco eine komplette Hardwarelösung an, die aber die Protokolle auf ihre eigenen Bedürfnissen anpasst.
An dieser Stelle sind noch einige großen Firmen zu nennen, die auf (Cisco) VoIP-Lösungen

umgestiegen sind: die Deutsche Bahn, das ZDF in Verbindung mit der TV-Sendung *wetten das...*, bei der während der kurzen Sendezeit viele Telefongespräche angenommen werden müssen und natürlich Cisco selbst, wobei alle 60.000 Mitarbeiter weltweit vernetzt wurden.
Anstelle der vielen Standards, die oftmals nicht eingehalten werden, ist noch ein weiteres VoIP-System zu nennen: Asterisk und sein Protokoll IAX.
Asterisk ist eine softwarebasierte Telefonanlage, die zusätzlich die freien VoIP-Standards SIP, H.323 und MGCP unterstützt, Möglichkeiten bietet, das herkömmliche Telefonnetz zu nutzen und viele weitere Telefonanlagenfeatures besitzt. Im folgenden wird Asterisk vorgestellt.

1.2 Features

1.2.1 Umfang der Telefon-Dienstleistungen

Bei dem Softwareprodukt Asterisk handelt es sich um eine digitale Nebenstellenanlage, die einen ähnlichen Leistungsumfang wie analoge Telefonanlagen bereitstellt.
Zu den Fähigkeiten zählen:

- VoiceMail-System, dass von dem Anrufer aufgenommene Nachrichten an den Nutzer weiterleitet:
  - Passwortgeschützt für direkten Zugriff
  - Unterscheidung zwischen *Away* und *Unavailable*-Nachrichten
  - Web Interface für das Abrufen der VoiceMails
  - Benachrichtigung per E-Mail
  - Empfang der VoiceMails per E-Mail
  - Weiterleitung von VoiceMails
- Zeit- und Kostenabrechnung (zu finden im Verzeichnis /var/log/asterisk/cdr-csv)
- VoiceMenu, das Dialoge mit dem Anrufer führt und auf dessen Eingaben reagiert (IVR)
- Flexibles Extension-System
  - Mehre, prioritätsgesteurte Extensions für ein Ziel
  - Mehrschichtige Zugangskontrolle
  - direkter Zugriff auf internes System
- Gruppenkonferenzen
  - nahezu unbegrenzte Anzahl an Konferenzräume
  - Zugangskontrolle
- Möglichkeit, Anrufe in eine Warteschlange zu stellen
- Aufzeichnung von Verbindungsdetails (billing)

- Local Call Agents
- Remote Call Agents
- Bridging zwischen Protokollen
  - Ermöglicht nahtlose Integration der Kommunikationstechnologien[1]
  - stellt umfangreiche Menge von Kommunikationstechnologien unabhängig von dem bevorzugten Dienst des Clienten zur Verfügung.
  - bietet Interoperabilität zwischen VoIP-Systemen

1.2.2 Verbindungs-Features

- Music on Hold
- Music on Transfer
  - Flexibles, mp3-basiertes Audiosystem
  - Kontrolle der Lautstärke
  - Abspielen von zufällig ausgewählten Audio-Dateien
- Call Waiting
- CallerID - Unterstützung
- Blocken von CallerIDs
- Bestimmte CallerIDs warten lassen
- Anrufweiterleitung, falls besetzt (CFB)
- Anrufweiterleitung, falls Anzurufender nicht abnimmt (CFNR)
- Anrufe parken
- *Do Not Disturb*, falls dies vom verwendeten Client unterstützt wird

1.2.3 Interoperabilität von Kommunikationstechnologien

Eine der vielen Stärken des PBX-Systems ist zweifellos das breite Spektrum der Technologien, mit denen eine Verbindung zu dem Asterisksystem aufgebaut werden kann. Dabei erlaubt Asterisk eine Verbindung zwischen VoIP-System und herkömmlichen Telefonsystemen herzustellen. Natürlich ist auch eine Transformation von den unten aufgelisteten VoIP-Techniken in jede anderen möglich, insbesondere in das für Asterisk entwickelte IAX-Protokoll. Im Kapitel 1.2.5 auf Seite 10 werden die Hardwaretechnologien, auf denen die Integration der folgenden Standards basieren, beschrieben.

[1] VoIP, analoge und digitale Hardware

VoIP-Technologien Asterisk unterstützt paketbasierte Telefontechniken (VoIP) wie:

- Voice over Frame Relay (VOFR)
- Session Initiation Protocol (SIP)
- ITU H.323
- UDP-basiertes Inter-Asterisk eXchange (IAX) - Protokoll
- Media Gateway Control Protocol (MGCP)

Analoge Hardware Da auf diesem Sektor nicht viele verschiedene Technologien konkurrieren, ist die Unterstützung nicht allzu breit gefächert. Um genau zu sein, wird nur die „leicht verständliche, alte Telefontechnik“ (POTS) unterstützt, wobei weitere Techniken auch nicht weiter verbreitet oder bekannt sind.

Digitale Hardware Asterisk unterstützt folgende, standardisierte Verfahren. Diese breite Fächerung erlaubt einen globalen Einsatz, da in den meisten Ländern nur wenige Verfahren zur Anbindung ans Internet angeboten werden:

- TDM (Time Division Multiplexing)
- T1[2] /E1 PRI/PRA & RBS (Robbed Bit Signal) Modi
- PRI-Protokolle
  - 4ESS
  - Lucent 5E
  - DMS100
  - National ISDN2
  - EuroISDN
  - BRI (ISDN4Linux)

Eine genaue Auflistung, wie diese Unterstützungen in der Praxis aussehen, zeigt Kapitel 1.2.5 auf Seite 10

[2] Thunk 1-Leitungen sind weit verbreitete Datenleitungen in den USA, die eine Übertragungsgeschwindigkeit von 1,544 MBit/s gewährleistet.

1.2.4 Unterstützte Codec's

- GSM
- G.729
- G.723.1
- Mu-Law
- A-Law
- ADPCM
- iLBC
- LPC10
- MP3 (nur abspielen)

1.2.5 Unterstützte Hardware

Im Groben lassen sich neben der Unterstützung von Sprachpaketen noch zwei Möglichkeiten unterscheiden, wie mit Asterisk (hier in Form von Hardware) kommuniziert werden kann: **Zaptel-** und **Nichtzaptel-Geräte**.

Zaptel-Geräte Diese Art von Geräten werden direkt von der Sponsorfirma Digium[3] vertrieben und dementsprechend gut unterstützt. Diese Steckkarten sind laut Herstellerangaben völlig unkompliziert einzubinden. Ob nun Asterisk an diese im folgenden aufgelisteten Geräte angepasst oder diese Geräte an Asterisk angepasst sind, lässt sich nicht sagen. Aber mit Sicherheit lässt sich sagen, dass ohne das Eine, das Andere nicht existieren würde.

- TE410P: PCI-Karte mit vier E1/T1 Ports
- TDM400P: PCI-Karte mit vier FXS Ports zum Anschluss von analogen Telefonen
- T400P: PCI-Karte mit vier T1/PRI-Interfacen
- T100P: PCI-Karte mit einem T1/PRI-Interface
- E400P: PCI-Karte mit vier E1/PRI Ports
- E100U: PCI-Karte mit einem E1/PRI Port
- X110P: PCI-Karte mit einen FXO-Anschluss

Nichtzaptel-Geräte Im weiterem gibt es noch die **Nichtzaptel-Geräte**, deren Entwicklung nichts mit Asterisk zu tun hat und die hauptsächlich in anderen Softwaresystemen zum Einsatz kommen.

- ISDN4Linux[7]
- OSS[8]/ALSA[9]
- Linux Telephony Interface (LTI)
- Phonejack[10]/Linejack[11]
- (Intel) Dialogic[12] Hardware

1.2.6 Einsatzgebiete

Die letzten Unterkapitel haben gezeigt, dass Asterisk mit einer Vielzahl von Funktionen ausgestattet wurde. Die folgende Auflistung soll noch einmal einen kurzen Überblick darüber geben, welche Anwendungsbereiche mit Asterisk abgedeckt werden können:

- Gateway für heterogene VoIP-Protokolle (MGCP, SIP, IAX, H.323)
- Private Branch eXchange (PBX)
- Server für VoiceMenu-Dienste
- Softswitch
- Konferenzserver
- Rufnummernumsetzung
- Einbindung von *Calling cards* (Telefonguthabenkarten)
- *Predictive dialer*, um ausgehende Anrufe zu tätigen
- Warteschlange für Anrufe
- *Remote offices* für existierende PBX

1.3 Hilfe/Unterstützung

Besonders in der Zeit vor den praktischen Erfahrungen mit dem PBX-System Asterisk sind ausführliche Literaturquellen und eine Möglichkeit, bei Problemen mit Leuten zu kommunizieren, die sich mit Asterisk besser auskennen, unerlässlich. Im folgenden werden alle Quellen, unter anderem auch solche, die für diese Studienarbeit herangezogen wurden, genauer betrachtet.

1.3.1 Handbuch

Als erste Anlaufstelle für Recherchen ist das Handbuch[5] von Mark Spencer, Mack Allision und Christopher Rhodes zu nennen. Bei der Erstellung dieser Studienarbeit lag die Version 2 vom 30. März 2003 vor, die sich zu dieser Zeit noch im Entwicklungsstadium befand.

1.3.2 Mailingliste

Als weitere sehr gute Quelle hat sich das Archiv der Asterisk-Mailingliste[6] erwiesen. Bei der Nutzung dieses Archivs für diese Studienarbeit wird der Zeitpunkt des Erscheinens der Postings berücksichtigt, die sich eventuell auf ältere Asterisk-Versionen beziehen könnten und somit nicht mehr korrekt sind.
Im weiteren ist noch zu erwähnen, dass es sich bei dieser Informationsquelle um eine offizielle Mailingliste handelt, an der sich auch die Entwickler Mark Spencer, Mack Allision und Christopher Rhodes beteiligen. Aus diesem Grund ist die Wahrscheinlichkeit, auf unkorrigierte Fehlinformationen zu treffen, relativ gering.

1.3.3 Internet Relay Chat (IRC)

Falls man auf eine schnelle Antwort angewiesen ist und diese Art der Kommunikation mag, ist man sicherlich gut in dem IRC-Channel #asterisk aufgehoben. Hier werden die aktuellen Probleme von Asterisk diskutiert und man kann jederzeit seine Fragen, in der Hoffnung, dass diese beantwortet werden, stellen.

1.3.4 Kommerzieller Support

Ferner besteht auch noch die Möglichkeit, kommerzielle Hilfsmöglichkeiten zu nutzen. Falls von der „Sponsor“-Firma Digium[3] Zaptel-Hardware erworben wurde, besteht hier die Möglichkeit, kostenlose Unterstützung bei Problemen mit diesen Geräten in Anspruch zu nehmen. Eine darüber hinausgehende Unterstützung, wobei hier z.B. IAX- und SIP-Probleme zu nennen sind, muss natürlich bezahlt werden. Um die Preise hierfür in Erfahrung zu bringen, wird an dieser Stelle auf die Support-Telefonnummer verwiesen, die auf der Homepage des Anbieters[3] zu finden ist.

1.4 Entwickler/Lizenz

Als Vater von Asterisk ist in erster Linie Mark Spencer, der die Urversion schrieb, zu nennen. Im weiteren sind noch Mack Allision und Christopher Rhodes dem engeren Kreis der Entwickler zuzuordnen, die bei der Weiterentwicklung von Asterisk eine große Rolle spielen. Obwohl Asterisk von der Firma *Digium dba Linux Support Services, Inc*[3], bei der Mark Spencer angestellt ist, finanziert wird, steht Asterisk unter der GPL[4]. Digium finanziert Asterisk nicht uneigennützig, da sie Support und Hardware für Asterisk anbietet und anscheinend davon leben kann.

1.5 Anforderungen an Verbindungs- und Verarbeitungsressourcen

Genaue Ressourcenanforderungen sind leider nicht bekannt, da diese von der Anzahl der aktiven Verbindungen abhängig sind.
Der Festplattenbedarf ist von der Anzahl der Nutzer, deren VoiceMails je nach Konfiguration abgespeichert werden können, abhängig. Außerdem sollte bei der Dimensionierung der Festplatte beachtet werden, dass bei einem Selbstbau der Quellen mit H.323-Unterstützung schnell 300MB an temporären Dateien entstehen können.
Als Betriebssystem wird ein Linux mit einer Kernel-Version der 2.4er Generation benötigt.

1.6 Verfügbarkeit

Im Grunde existieren drei Möglichkeiten, mit denen Asterisk beschafft werden kann:

1. CVS
2. Quellen von ftp://ftp.asterisk.org
3. vorcompiliertes Paket

Das Kapitel 2 ab Seite 16 beschreibt detailliert, wie Asterisk über CVS installiert wird. Um die aktuellste Version zu erhalten, sollte diese Methode verwendet werden. Analog können natürlich auch die Quellen vom ftp-Server gezogen werden, was aber weniger komfortabel ist. Aus Gründen der Einfachheit oder im Falle beschränkter Kapazitäten kann natürlich auch Asterisk aus einer vorhanden Paketdatenbank der Linux-Distribution installiert werden.

1.7 Architektur von Asterisk

Die Architektur der PBX-Lösung Asterisk ist relativ einfach gehalten, wobei Asterisk als eine Art Übersetzer von Paket- bzw. Hardwarebasierten Technologien zu Telefonapplikationen wie *call bridging*, *Conferencing* oder anderen zu verstehen ist.
Das „Paket“ Asterisk ist in Abbildung 1 soweit skizziert, dass die Kernkomponenten betrachtet werden können.

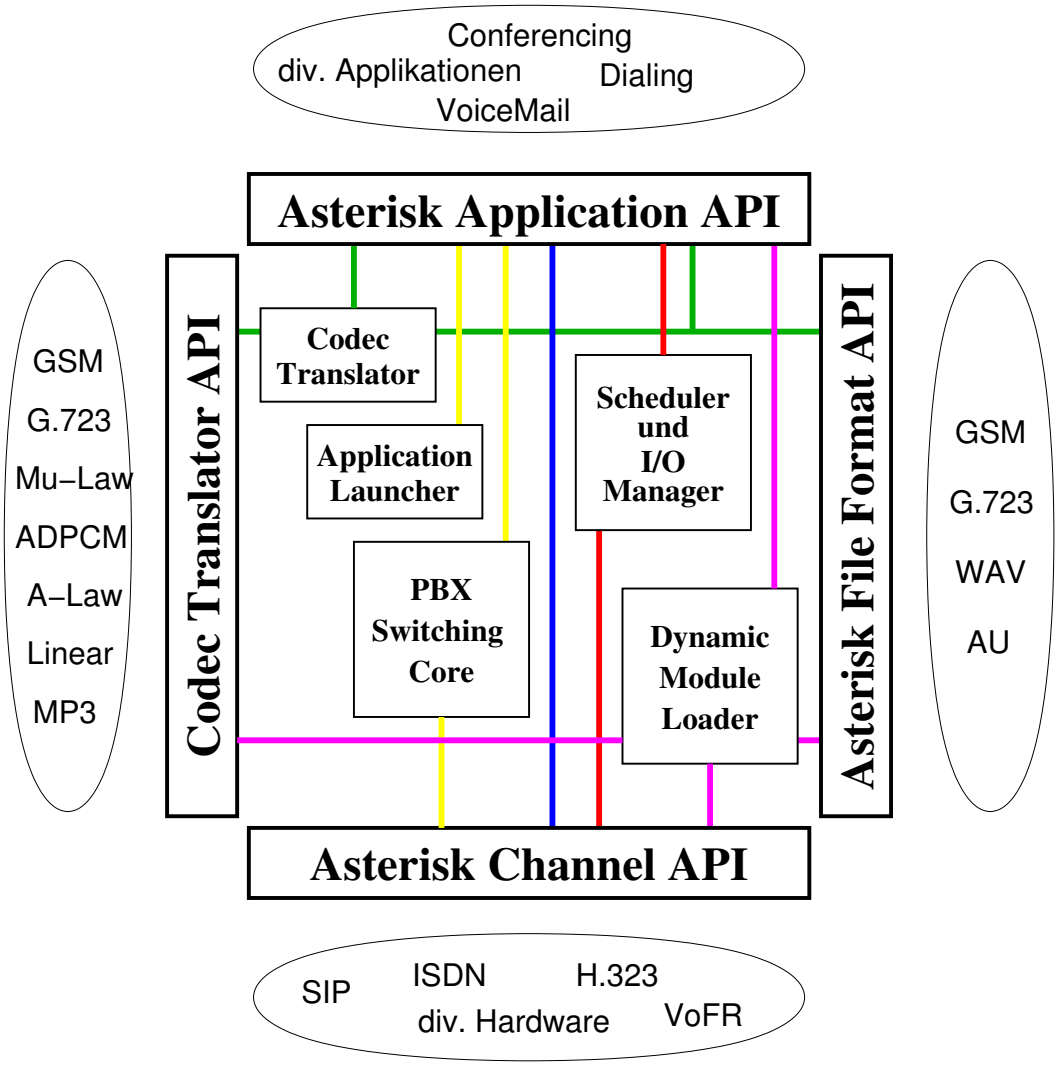

Abbildung 1: Detaillierter Aufbau von Asterisk

Bei einem Start von Asterisk wir als erster Schritt der **Dynamic Module Loader** geladen und initialisiert. Als Module, mit denen Asterisk ohne größere Eingriffe erweitert werden kann, stehen Unterstützungen für Dateiformate, Codecs oder andere zu Verfügung.
Nun ist Asterisk startbereit und Anrufe können von der **Switching Core** über die Schnittstellen (Hardware oder VoIP) angenommen und nach der extensions.conf bearbeitet werden. Falls Anrufe angenommen werden, werden diese an den **Application Launcher** weitergelei-

tet, der die angeschlossenen Telefone klingeln lässt, in dem er z.B die passenden Datenpakete verschickt oder Textnachrichten aufzeichnet (VoiceMails).
Der **Scheduler und I/O Manager** übernimmt dabei die Verwaltung der Applikationen und der Channels (VoIP, ...). Als letztes Element ist der **Codec Translator** zu nennen, der die verschiedenen Codecs zu einen einheitlichen Standard, der intern von allen Applikationen und Dateiformaten verstanden werden kann, übersetzt.

2 Installation

Das Ziel dieses Kapitels besteht darin, eine Anleitung für die unkomplizierte Installation eines kompletten Asterisksystems zu Verfügung zu stellen. Neben einer Asteriskinstallation kann, wie die folgenden Kapitel zeigen, eine Unterstützung für H.323 eingebunden werden. Dies ist aber nur optional und somit empfiehlt es sich, falls keine H.323 Unterstützung benötigt wird, diesen Teil zu übergehen.

2.1 Installation des Asteriskgrundsystems

Um die aktuellste Asteriskversion zu installieren, sollten die Quellen über CVS eingebunden und compiliert werden. Eine bestehende Anbindung ans Internet vorausgesetzt, kann dies mittels

```
[root@voip shol]# cd /usr/src
[root@voip src]# export HOST=pserver:anoncvs@cvs.digium.com
[root@voip src]# export CVSROOT=:$HOST:/usr/cvsroot
[root@voip src]# cvs login - the password is anoncvs.
[root@voip src]# cvs checkout zaptel libpri asterisk
```

in die Wege geleitet werden. Nun sollten sich die jeweiligen Quellen in den Verzeichnissen **zaptel**, **libpri** und **asterisk** befinden. Um einen Abbruch des Compiliervorgangs entgegen zu wirken, sollte sich eine Installation von OpenSSL auf der Zielarchitektur befinden.

2.1.1 Zaptel-Unterstützung

Falls Zaptelkompatible Geräte verwendet werden sollen, muss diese Unterstützung der Asteriskübersetzung zur Verfügung stehen. Im Grunde handelt es sich bei Zaptel-Geräten um Schnittstellenkarten, die dem Asterisksystem einen Anschluss an das Telefonnetz oder einzelnen analogen Endgeräten ermöglichen. Auf die Zaptel-Unterstützung wurde in Kapitel 1.2.5 eingegangen. Um dies zu realisieren, sollten folgende Befehle

```
[root@voip src]# cd zaptel
[root@voip zaptel]# make clean ; make install
```

ausgeführt werden. Diese Unterstützung ist optional.

2.1.2 LibPri-Unterstützung

Dieses von Mark Spencer unter der GNU General Public License gestellte Modul implementiert die Primary Rate ISDN Spezifikation und stellt diese Asterisk zur Verfügung. Falls Asterisk in einem ISDN-System eingesetzt werden soll, muss mittels

```
[root@voip src]# cd libpri
[root@voip libpri]# make clean ; make install
```

diese Unterstützung compiliert und bereitgestellt werden.

2.1.3 Installation des Kern-Programms

Nachdem die beiden oberen Schritte, falls benötigt, ausgeführt wurden, kann nun mit den Befehlen

```
[root@voip src]# cd asterisk
[root@voip asterisk]# make clean ; make install
```

ein lauffähiges Asterisksystem erstellt werden. Dieser Vorgang dauert auf modernen Rechnern zehn bis zwanzig Minuten.
Empfehlenswert ist auch die Erstellung eines Webinterfaces, dass die VoiceMails der Nutzer verwalten kann. Dies geschieht mit dem Befehl:

```
[root@voip asterisk]# make webvmail
```

Die Datei *vmail.cgi* kann von einem beliebigen WWW-Server, der im einfachsten Fall auf dem Asteriskserver läuft, bereitgestellt werden.
Falls anschließend noch die H.323 Unterstützung integriert werden soll, dürfen die Quellen nach einer erfolgreichen Übersetzung nicht gelöscht werden.

2.2 Integration der H.323 Unterstützung

Auf dem ersten Blick stellt sich die Frage, warum die H.323-Unterstützung von externen Quellen eingebunden werden muss, da laut der Asterisk-Homepage[13] diese in Asterisk integriert sein sollte. Leider ist diese aber nicht implementiert, da die H.323-Unterstützung auf das OpenH323-Projekt[14] aufbaut. Die folgende Auflistung skizziert kurz die Probleme, die gegen eine feste Implementierung von H.323 in Asterisk sprechen:

1. Asterisk stützt sich auf die GNU General Public License[15], wogegen OpenH323 der MPL (Mozilla Public License)[16] untersteht.

2. Asterisk ist komplett in der Programmiersprache C geschrieben, OpenH323 wurde komplett in C++ implementiert. Daher ist ein „Einbetten“ von OpenH323 in Asterisk nicht ohne weiteres möglich.

Aus den beiden oben genannten Gründen ist daher eine Asterisk-OH323 Arbeitsgruppe[17] entstanden, die sich zur Aufgabe gemacht haben, eine reibungslose Integration von H.323 zu garantieren. Dieses Projekt wird von Michael Manousos und Dimitris Economou gepflegt und steht ebenfalls unter der GNU General Public License.
Wie aus der *README* des Asterisk-OH323 ersichtlich, wird für die Erstellung des Moduls eine Installation von *OpenH323* und der *Portable Windows Library* für OpenH323 benötigt. Außerdem wird der Einsatz eines H.323-Gatekeepers empfohlen.

2.2.1 Installation der Portable Windows Library

Die PWLib ist eine Klassenbibliothek, die es ermöglicht, viele Arten von Applikationen sowohl unter Linux- als auch unter Windowssystemen auszuführen. Außerdem stellt die PWLib Klassen für I/O-Funktionalitäten, Multi-Threading und weitere zur Verfügung.
Die Library, die in enger Beziehung zu dem OpenH323-Projekt steht, ist auch mit dem üblichen Befehlstupel

```
[root@voip src]# cd pwlib
[root@voip pwlib]# make clean
[root@voip pwlib]# make install
```

denkbar einfach zu installieren. Es sollte eventuell beachtet werden, dass das ca. 1MB große Archiv während des compilierens etwa 70 MB an Speicherplatz benötigt. Außerdem darf das Verzeichnis, in dem sich die pwlib-Quellen befinden, **nicht** nach dem „make install“ gelöscht werden. Diese Quellen werden noch für die folgenden Schritte benötigt.

2.2.2 Installation von OpenH323

Da Asterisk keine H.323 Unterstützung besitzt, muss für das H.323-Plugin eine *OpenH323*-Installation zu Verfügung gestellt werden.
Nach dem Auspacken des Archivs sollte eventuell der Pfad zu dem pwlib-Verzeichnis in dem *Makefile* angepasst werden. Anschließend kann man die Quellen mit dem folgenden Befehlen compilieren und installieren:

```
[root@voip src]# cd openh323
[root@voip openh323]# make clean
[root@voip openh323]# make install
```

Auch die geringe Größe des TAR-Files sollte nicht unterschätzt werden. Während des übersetzens werden ungefähr 200 MB an Dateien angelegt, die nicht direkt nach dem „make install“ gelöscht werden dürfen.

2.2.3 Bauen des Asterisk-OH323

Nun ist es soweit und „Asterisk-OH323“ kann compiliert werden. Nachdem das TAR-File entpackt wurde, sollten die Pfade im Makefile angepasst werden. Nach diesem Schritt kann, wie üblich, mit den Befehlen

```
[root@voip src]# cd asterisk-oh323
[root@voip asterisk-oh323]# make clean
[root@voip asterisk-oh323]# make install
```

die H.323-Unterstützung für Asterisk gebaut werden. Nun sollte in der /etc/ld.so.conf der Zielpfad eingetragen werden (in meinem Fall: /usr/local/lib) und

```
[root@voip etc]# ldconfig
```

aufgerufen werden. Jetzt müsste, wenn alle vorigen Schritte erfolgreich ausgeführt wurden, die H.323-Unterstützung in Asterisk eingebettet sein. Zur Kontrolle sollte sich in /etc/asterisk eine Datei mit dem Namen oh323.conf befinden und eine Reihe oh323-Befehlen unter dem CLI (siehe Kapitel 3) zur Verfügung stehen.

2.2.4 Installieren eines Gatekeepers

Jetzt kann ein beliebiger Gatekeeper aus der Linux-Distribution installiert werden. Zu empfehlen wäre der „GNU Gatekeeper“, aber hier steht der Geschmack des Administrators im Vordergrund. Da in dieser Studienarbeit H.323 eine untergeordnete Rolle spielt, wird an dieser Stelle nicht auf die Konfiguration eingegangen.

3 Command Line Interface

Asterisk bietet die Möglichkeit, über das CLI Statusinformationen auszugeben. Das CLI kann wie eine Art Shell verstanden werden, unter der Befehle eingegeben werden können. Um das CLI aufzurufen, wenn Asterisk noch nicht gestartet sein sollte, empfiehlt sich folgender Befehl:

```
[shol@voip asterisk]$ /usr/sbin/asterisk -c
```

Hierbei wird Asterisk mit dem CLI gestartet. Falls schon Asterisk im Hintergrund läuft, kann mit

```
[shol@voip asterisk]$ /usr/sbin/asterisk -r
```

das *Command Line Interface* des laufenden Systems gestartet werden.
Nun sollte man sich unter dem CLI befinden, welches wie in der folgenden Abbildung auf Eingaben wartet:

```
voip*CLI>
```

Wie unter eine BASH können die Befehle mit der Tabulatortaste vervollständigt und mit der ’Pfeil-Rauf’ Taste wiederholt werden.
Da mittels

```
voip*CLI> help
```

alle Befehle aufgelistet werden, macht es keinen Sinn, jede mögliche Tastatureingabe zu dokumentieren. Daher werden nur die wichtigsten Befehle kurz vorgestellt. Um Informationen zu bestimmten Befehlen zu erhalten, kann auch der Hilfstext für einen bestimmten Befehl angezeigt werden:

```
voip*CLI> help stop now
Usage: stop now
       Shuts down a running Asterisk immediately, hanging up
       all active calls .
voip*CLI>
```

Somit kann man sich über alle möglichen Befehle informierten.
Um diese Übersicht überschaubar zu halten, werden im folgenden IAX und SIP - Befehle gleichgesetzt

Falls für Debugging-Zwecke Informationen über empfangener und gesendeter IAX-Pakete benötigt werden, kann dies über

```
voip*CLI> iax debug
IAX Debugging Enabled
```

aktiviert werden. Analog funktioniert dies wie bei den folgenden Beispielen auch mit SIP.
Um sich weitere IAX (oder analog SIP)-Befehle anzeigen zu lassen, empfiehlt sich ein Aufruf von

```
voip*CLI> help iax show
      iax show cache   Display IAX cached dialplan
   iax show channels   Show active IAX channels
      iax show peers   Show defined IAX peers
   iax show registry   Show IAX registration status
      iax show stats   Display IAX statistics
      iax show users   Show defined IAX users
```

Ein weiter wichtiger Befehl ist das CLI-Kommando *reload*:

```
voip*CLI> reload
    -- Unregistered indication country 'us'
    -- Unregistered indication country 'au'
    -- Unregistered indication country 'fr'
    -- Unregistered indication country 'de'
    -- Unregistered indication country 'nl'
    -- Unregistered indication country 'uk'
    -- Registered indication country 'us'
    -- Registered indication country 'au'
    -- Registered indication country 'fr'
    -- Registered indication country 'de'
    -- Registered indication country 'nl'
    -- Registered indication country 'uk'
    -- Setting default indication country to 'de'
voip*CLI>
```

Falls die Konfigurationsdateien geändert wurden, werden mit einem *Reload* unter anderem diese Änderungen in Asterisk übernommen.

Eine weitere interessante Eingabe ist die Abfrage der Applikationen, die bei einem Zugriff einer Extension in der extensions.conf ausgeführt werden können:

```
voip*CLI> show applications
voip*CLI>
    -= Registered Asterisk Applications =-
         ZapBarge: Barge in (monitor) Zap channel
            Flash: Flashes a Zap Trunk
           MeetMe: Simple MeetMe conference bridge
      MeetMeCount: MeetMe participant count
           ZapRAS: Executes Zaptel ISDN RAS application
      ChanIsAvail: Check if channel is available
        DBdeltree: Delete a family or keytree from the database
            DBdel: Delete a key from the database
            DBput: Store a value in the database
            DBget: Retrieve a value from the database
   PrivacyManager: Require phone number to be entered, if no
                   CallerID sent
      WaitForRing: Wait for Ring Application
  LookupBlacklist: Look up Caller*ID name/number from blacklist
                   database
       SoftHangup: Soft Hangup Application
     Authenticate: Authenticate a user
            Macro: Macro Implementation
        SubString: Save substring digits in a given variable
    LookupCIDName: Look up CallerID Name from local database
         StripLSD: Strip Least Significant Digits
       SetCIDName: Set CallerID Name
         SendDTMF: Sends arbitrary DTMF digits
            Queue: Queue a call for a call queue
         Festival: Say text to the user
      SetCallerID: Set CallerID
         DateTime: Say the date and time
       Zapateller: Block telemarketers with SIT
        Milliwatt: Generate a Constant 1000Hz tone at 0dbm (mu-law)
         GetCPEID: Get ADSI CPE ID
         ADSIProg: Load Asterisk ADSI Scripts into phone
              AGI: Executes an AGI compliant application
             EAGI: Executes an AGI compliant application
             DISA: DISA (Direct Inward System Access)
          SendURL: Send a URL
        SendImage: Send an image file
           Record: Record to a file
             Echo: Echo audio read back to the user
```

Abbildung 2: Auflistung der zur Verfügung stehenden Applikationen, Teil 1

```
          System: Execute a system command
       MP3Player: Play an MP3 file or stream
       Directory: Provide directory of voicemail extensions
   VoiceMailMain: Enter voicemail system
       VoiceMail: Leave a voicemail message
        Playback: Play a file
            Dial: Place an call and connect to the current channel
      AgentLogin: Call agent login
 ParkAndAnnounce: Park and Announce
   ChangeMonitor: Change monitoring filename of a channel
     StopMonitor: Stop monitoring a channel
         Monitor: Monitor a channel
   StopPlaytones: Stop playing a tone list
       Playtones: Play a tone list
      ParkedCall: Answer a parked call
  SetMusicOnHold: Set default Music On Hold class
 WaitMusicOnHold: Wait, playing Music On Hold
     MusicOnHold: Play Music On Hold indefinitely
          GotoIf: Conditional goto
            NoOp: No operation
    SetGlobalVar: Set variable to value
          SetVar: Set variable to value
            Busy: Indicate busy condition and stop
      Congestion: Indicate congestion and stop
         Ringing: Indicate ringing tone
     SetLanguage: Sets user language
          Prefix: Prepend leading digits
        StripMSD: Strip leading digits
            Wait: Waits for some time
      BackGround: Play a file while awaiting extension
 AbsoluteTimeout: Set absolute maximum time of call
 ResponseTimeout: Set maximum timeout awaiting response
    DigitTimeout: Set maximum timeout between digits
          Hangup: Unconditional hangup
            Goto: Goto a particular priority, extension or context
          Answer: Answer a channel if ringing
```

Abbildung 3: Auflistung der zur Verfügung stehenden Applikationen, Teil 2

4 Konfiguration von Asterisk

Dieses Kapitel basiert hauptsächlich auf das Asterisk-Handbuch und in dem Unterkapitel über die extensions.conf auf eigene Erfahrungen. Es beschäftigt sich mit der Konfiguration des Asterisk-VoIP-Systems und stellt die wichtigsten Verzeichnisse und Konfigurationsdateien vor. Im Kapitel 9 befindet sich ein Anwendungsbeispiel für den Einsatz von Asterisk, dessen Konfigurationdateien Beispiele zu den in diesen Kapitel beschriebenen darstellen.

4.1 Wichtige Verzeichnisse

Um mit der Konfiguration zu beginnen, sollten erst einmal die wichtigsten Verzeichnisse lokalisiert werden. Wie bei vielen Linux-Systemen und deren Anwendungssoftware üblich, wurden alle benötigten Dateien in folgende Verzeichnisse gruppiert (diese Auswahl beinhaltet nur die Verzeichnisse, die für die hier vorgestellte Nutzung von Asterisk benötigt werden):

4.1.1 /etc/asterisk

In diesem Verzeichnis befinden sich alle Dateien, die für eine Konfiguration von Asterisk von Bedeutung sind. Die wichtigsten, mit denen ein umfangreicher Einsatz von Asterisk möglich ist, sind:

- iax.conf
- sip.conf
- extensions.conf
- voicemail.conf

Die Dateien werden detailliert in dem Kapitel 4.2 erklärt.

4.1.2 /usr/sbin

In dem Verzeichnis **/usr/sbin** wird während der Installation von Asterisk (siehe Kapitel 2) die ausführbare Binärdatei **asterisk** kopiert. Mit dieser kann Asterisk und/oder das **Command Line Interface** (siehe dafür Kapitel 3) gestartet werden.

4.1.3 /usr/lib/asterisk/modules

In diesem Verzeichnis müssen die nachladbaren Module und Unterstützungen für Applikationen, IP-Protokolle (z.B. IAX, ...) Codecs und Dateiformate (z.B. mp3, wav...) zu finden sein.

4.1.4 /var/lib/asterisk/sounds

Hier können die GSM-Dateien für Ansagen, VoiceMenus und andere Applikationen abgelegt werden. Folgende Dateien und Verzeichnisse haben eine besondere Bedeutung, deren Identifikatoren fest in Asterisk einprogrammiert sind und sich nicht ändern lassen:

digits - Verzeichnis Die in diesem Verzeichnis zu findenden Dateien mit den Namen 0.gsm bis 9.gsm beinhalten jeweils eine Aufnahme der jeweiligen Ziffer. Diese wird z.B. bei Nummern-Ansagen von Asterisk benötigt. Falls man ein individuelles Voicesystem (z.B. VoiceMail) betreibt, sollten diese Dateien jeweils neu aufgenommen werden, um z.B. Differenzen in der Landessprache („Der Nutzer mit der Nummer zero-nine-one ist leider nicht erreichbar“) und Stimme (Normale Ansagen z.B. Frauenstimme, Ziffern mit Männerstimme) zu vermeiden.

Mit vm-beginnende GSM-Dateien Leider existiert bisher in Asterisk keine Möglichkeit, die für ein individuelles VoiceMail-System (Anrufbeantworter) aufgenommen GSM-Dateien in irgendeiner Konfigurationdatei festzulegen. Dieses Problem kann gelöst werden, indem einfach die Inhalte der festgelegten Dateinamen mit eigenen Ansagen überschrieben werden. Folgende Dateien werden für das VoiceMenu-System benötigt:

Dateiname	Sinngemäßer Inhalt
vm-theperson.gsm	Der Teilnehmer mit der Rufnummer
vm-isunavail.gsm	ist im Augenblick nicht verfügbar
vm-isonphone.gsm	telefoniert selber gerade.
vm-intro.gsm	Bitte hinterlassen Sie nach dem Piep-Ton eine Nachricht und legen Sie dann auf
beep.gsm	Ein Piep-Ton

Tabelle 1: Von dem VoiceMail-System benötigte GSM-Dateien

4.1.5 /var/log/asterisk/

Dieses Verzeichnis beinhaltet alle Protokollierungsdateien. In dem Unterverzeichnis *cdr-csv* werden die Zeit- und Kostenabrechnungen der Anrufe abgelegt und die beiden anderen Dateien beinhalten Systemmeldungen und Zustände.

4.2 Ausgewählte Konfigurationsdateien

Die Konfigurationsdateien befinden sich, falls bei der Installation nicht anders vereinbart, im Verzeichnis */etc/asterisk*. Wie schon im Original-Asteriskhandbuch beschrieben, ähnelt der Aufbau der einzelnen Konfigurationsdateien einer win.ini älterer Windowsversionen.

Kommentare werden mit **;** , wie im Beispiel Abbildung 4, eingeleitet.

Gruppierungen die Dateien können in einzelne Abschnitte eingeteilt werden. Wie im Beispiel in Abbildung 4 wird jeder Abschnitt mit einen [*Abschnittname*] definiert, dem die zugehörigen Anweisungen folgen.

Objekte können auch Befehlszeilen sein. Sie werden wie im Beispiel in Abbildung 4 je nach Konfigurationsdatei mit einen = oder => von den Optionen getrennt.

Gliederung Die meisten Konfigurationsdateien besitzen ein [**general**]-Abschnitt, in dem globale Definitionen und Festlegungen eingetragen werden. Analog werden die lokalen Definitionen und Festlegungen in den dementsprechenden [**Abschnitten**], wo diese nur gültig sind, festgelegt. Dies wird in Abbildung 4 verdeutlicht.

```
; dies ist ein Kommentar
;
[general]
globale_option=globaler_wert
;
[Gruppierung]
objekt1 => option1
opjekt2 => option2a,option2b
```

Abbildung 4: Skelett einer Konfigurationsdatei

4.2.1 iax.conf

Die Datei /etc/asterisk/iax.conf besteht aus einem globalen und für jeden eingetragenen IAX-User jeweils einem lokalen Abschnitt. Im globalen Abschnitt werden die Performance-, Server- und Protokolleinstellungen vorgenommen, während in den restlichen Sektionen die einzelnen IAX-Klienten, die auf Asterisk zugreifen dürfen, definiert werden. Diese Datei kann sowohl für IAXv1 als auch IAXv2 genutzt werden. Ein vollständiges Beispiel ist im Kapitel 9.2.1 zu finden.

general-Einstellung

port Auf diesem Port lauscht die IAX-Unterstützung von Asterisk. Hier muss ein numerischer Wert eingetragen werden. Dieser sollte nur in den Fall nicht auf 5036 gesetzt werden, wenn dieser Port von einem anderen Programm belegt wird.

bindaddr Hier kann die IP-Adresse eingetragen werden, an der Asterisk lauschen soll. Falls dieser Eintrag nicht auf die Adresse 0.0.0.0 gesetzt wird, kann bei einem Server mit mehreren Adressen (z.B. bei mehreren Netzwerkkarten) eine Adresse festgelegt werden.

amaflag Kategorisierung der Einträge in den detaillierter Anrufaufzeichnungen. Kann auch auf Klientenseite gesetzt werden. Werte:

- **billing**: Abrechnung
- **documentation**: Dokumentation
- **omit**: Anruf nicht aufzeichnen
- **default**: Defaultwerte des Systems

Die Protokolldateien sind in dem Verzeichnis /var/log/asterisk/cdr-csv zu finden.

accountcode Einstellung des Defaultkontos zum Logging von IAX-Anrufen, die auch Klientendefinition beinhalten können.

bandwidth steuert, welche Codecs benutzt werden sollen. Es werden jedoch nicht bestimmte Codecs angegeben, sondern nur **low**, **medium** oder **high**. Damit werden solche Codecs vermieden, die mit den entsprechenden Bandbreiten nicht gut zusammenarbeiten.

- **high**: high aktiviert alle Codecs und wird nur für Verbindungen mit mindestens 10 Mbps empfohlen.
- **medium**: Bei medium werden *signed linear*, *mu-Law* und *A-law-Codecs* ausgeschlossen. Es werden nur Codecs mit 32kpbs oder weniger, wie z.B MP3, verwendet.

  Diese Einstellung kann bei Breitbandverbindungen genutzt werden.
- **low** Bei low werden die *ADPCM* und *MP3* Formate ausgeschlossen, es bleiben lediglich *G.723.1*, *GSM*, und *LPC10* übrig.

  Dies ist die empfohlene Einstellung, wobei *LPC10* über die Option **disallow** ausgeschlossen werden sollte, da dieser eine schlechte Klangqualität besitzt.

allow verwendet die Codecs, die mit Komma getrennt aufgelistet werden können. Dabei ist auch der Eintrag *all* möglich, der **bandwidth=high** entspricht

disallow schließt die Verwendung der mit dieser Option aufgelistetem Codecs aus.

jitterbuffer Ein- oder ausschalten des Jitterbuffers. Dieser sollte immer angeschaltet sein, es sei denn, die Verbindung erfolgt über ein LAN.
Der Jitterbuffer maximiert die Audioqualität durch Optimierung der Latenzzeiten durch Verwerfen von Paketen. Dieser kann mit passenden Werten feineingestellt werden.

dropcount Maximale Anzahl von Paketen pro Speichereinheit (bzw. 100), die verworfen werden dürfen um die Latenzzeit zu verringern. Gute Werte liegen zwischen 3 und 10.

maxjitterbuffer Maximale Größe des Jitterpuffers.

maxexessjitterbuffer Maximaler Überschuss im Jitterpuffer.
Wird dieser Wert überschritten, wird der Jitterpuffer verkleinert, um die Latenzzeit zu verbessern.

register Mit diesem Eintrag soll sich der Asteriskserver bei einem anderen Asteriskserver registrieren. Dies wird in der Regel nur angewendet, wenn der lokale Server eine dynamische IP besitzt und dem anderen Server mitteilen muss, wie er ihn ausfindig machen kann.
Dazu muss der Server einen dynamischen „Partnereintrag“ (peer entry) mit dem selben Namen und/oder dem *secret* besitzen. Das *secret*-Feld ist optional, falls ein *secret* auf den anderen Server spezifiziert wurde. Bei Verwendung von *RSA* wird der Schlüssel in „[ ]“ an dieser Stelle angegeben. In diesem Fall muss der private Schlüssel des lokalen Servers unter /var/lib/asterisk/keys/<name>.keys liegen und der entfernte Server den öffentlichen Schlüssel besitzen.

tos Angabe des *Type oder Service*-Bits für die IAX-Pakete. Dadurch kann das Routing verbessert werden.

- **lowdelay**: Verzögerung minimieren
- **throughput**: Durchsatz maximieren
- **reliability**: Zuverlässigkeit maximieren
- **mincost**: Verwendung des Pfades mit den niedrigsten Kosten
- **none**: keine Routing flags

Empfohlen wir die Verwendung von *lowdelay*. Viele Router (einschließlich Linux mit 2.4 Kernel, der nicht durch *IPTables* verändert wurde) bevorzugen diese Pakete und somit verbessert sich die Sprachqualität.

User-Einstellungen

peer Mit diesem Eintrag wird der Username festgelegt. So kann sich z.B. GnoPhone an Asterisk authentifizieren und in der extensions.conf (siehe Kapitel 4.2.3) werden diesem Eintrag Telefonnummern zugewiesen.

type Unter welchem Kontext soll diese Entität betrachtet werden?
Entity-Typ des Klienten:

- **peer**: Dieser Klient kann nur angerufen werden, darf aber selber keine Verbindung aufbauen
- **user**: Dieser Klient kann nur anrufen, aber nicht angerufen werden
- **friend**: kann Anrufe senden und empfangen

Im Zweifel sollte **friend** gewählt werden

context Einer oder mehrere Kontexte können für einen User angegeben werden. Damit können die Anrufe des Users in spezielle Kontexte eingeteilt werden.
Kontexte werden von Asterisk genutzt, um Dial-Pläne in logische Einheiten einzuteilen, in denen jeweils Nummern verschieden interpretiert werden können, damit ein eigenes Sicherheitskonzept angewandt werden kann, Unterstützung beim Umschalten (auxillary switch handling) und andere Kontexte mit einbezogen können.
Die meisten User geben den Zugriff auf ihren Defaultkontext frei. Vertrauenswürdigen Usern kann z.B. der Zugriff auf lokale Kontexte gewährt werden.
Wird diese Zeile bei einem Klienten gesetzt, so wird der Kontext aus dem *general*-Abschnitt für den Nutzer überschrieben.

callerid Die vom Nutzer übertragene *CallerID* kann überschrieben werden, damit, falls er unterschiedliche Klientenmaschinen nutzt, immer der gleiche Bezeichner beim Kommunikationspartner angezeigt wird. Die CallerID wird intern benutzt und an das PSTN gesendet, falls eine PRI Verbindung benutzt werden soll um die Anrufe in die „Welt“ zu leiten.
Falls auch die Telefonnummer übertragen werden soll, muss diese expliezit eingetragen werden.
Ist die *CallerID* nicht gesetzt wird nur der lokale Name des Rechners, auf dem der Klient z.B. sein Softphone in Betrieb hat, übertragen.

auth Es kann zwischen verschiedenen Authentifizierungsmethoden gewählt werden, die für die Nutzer zum Registrieren zur Verfügung stehen. Es können mehrere Optionen angegeben werden, die durch Kommas getrennt werden. Falls eine **md5**- oder **plaintext**-Authentifizierung gewählt wird, muss das **secret** definiert werden.
Bei der Authentifizierungsmethode **RSA** müssen die Schlüssel unter **inkeys** spezifiziert werden.
Wird weder ein *secret* noch eine Authentifizierungsoption angegeben, dann ist keine Registrierung möglich.
Dieses *secret* wird über die **secret**-Zeile definiert und sollte niemals (außer vielleicht zum Debugging) als Plaintext angegeben werden.

- **plaintex**: am unsichersten, Server und Klient haben Zugriff auf den Klartext und dieser wird über das Netz gesendet.
- **md5**: beide haben Klartextzugriff, es wird aber ein md5hash über das Netzwerk geschickt
- **RSA**: sicherste Variante, Schlüsselpaar können mittels den beiliegenden Programm *astgenkey* erzeugt werden. Der öffentliche Schlüssel muss per Hand in das entsprechende Verzeichnis auf dem Server übertragen werden.

Soll RSA mit Asterisk verwendet werden, so muss beim Starten auf dem *Command Line Interface* ein *init keys* ausgeführt werden.

inkeys Diese Zeile gibt an, welche Keys zur Authentifizierung des Kommunikationspartners verwendet werden können. Falls der Gegenüber die Challenge mit einem der Schlüssel richtig beantwortet, so kann dessen Authentifizierung akzeptiert werden. Die Schlüssel liegen unter /var/lib/asterisk/keys/<name>.pub und sind öffentliche Schlüssel. Die öffentlichen Schlüssel sind typischerweise nicht DES3 verschlüsselt und deshalb bedarf es gewöhnlich auch keiner Initialisierung.

outkeys Dieser Eintrag ist analog zu dem vorigen Punkt *inkeys* anzusehen, wobei *outkeys* den privaten Schlüssel festlegt.

secret Falls *md5* oder *plaintext* als Authentifikationmethode unter *auth* gewählt wurde, kann mit diesem Eintrag das Passwort zuordnen werden. Falls kein secret angegeben wird, ist keine Authentifizierung möglich.

permit Permit und deny-Regeln werden auf Nutzer angewendet, so dass diese sich nur von bestimmten IP-Adressen an Asterisk anmelden können. Die Regeln werden der Reihe nach verarbeitet, wobei diese Abarbeitung nicht nach einer positiven Prüfung abgebrochen wird. Daher spielt die Reihenfolge eine große Rolle. Es ist auch möglich, Kombinationen von IP-Adressen und Netzmasken anzugeben. Wird keine Regel angegeben wird angenommen, dass sich der zugehöriger Nutzer von überall her verbinden darf.

deny Dieser Eintrag ist analog zu *permit*.

host Die IP-Adresse oder der Hostname des Kommunikationspartner wird mit diesem Eintrag festgelegt. Wird der Eintrag mit dem Wert *dynamic* verwendet, so registriert sich der Gegenüber direkt am Asterisk.

defaultip Bei der Verwendung einer dynamischer Registrierung kann hier trotzdem eine Defaultadresse angegeben werden, falls die Registrierung nicht möglich war oder ein *Time out* aufgetreten ist.

accountcode Wird diese Option innerhalb der Klientendefinition verwendet, so wird der Accountcode nur für diesen Klienten gesetzt.

qualify Gibt an, ob der Gesprächspartner auf „Lebenszeichen“ überprüft werden soll bevor ein Verbindungsversuch unternommen wird.
Wird dies gewünscht, so kontaktiert Asterisk den Gegenüber regelmäßig, bevor er Anrufinformationen weiterleitet. Als Argument wird die Zeit in Millisekunden angegeben, die Asterisk höchstens wartet bevor der Klient als *nicht verfügbar* eingestuft wird.

mailbox Ordnen dem Gegenüber eine Mailbox zu, so das dieser, wenn er sich registriert, alle offenen Nachrichten angezeigt bekommt.

trunk An- oder Ausschalten des ’Trunkings’. Dieser Modus ist die effizientere Methode IAX zu betreiben, falls viele Anrufe verarbeitet werden müssen. Dazu ist es notwendig, ein Zaptel-Interface am Asteriskserver zu betreiben.

4.2.2 sip.conf

In der Datei /etc/asterisk/sip.conf werden alle Einstellungen für SIP eingetragen. Hier existiert wieder ein globalen Abschnitt, der das Verhalten des Servers steuert und Defaultwerte setzt. Außerdem gibt es wieder einen Abschnitt, in dem für jeden Nutzer, der SIP nutzen darf, die persönlichen Einstellungen eingetragen werden.

general-Einstellung

port Der Eintrag der Portnummer ist analog zu Kapitel 4.2.1 auf Seite 27, mit dem Unterschied, dass bei SIP natürlich ein anderer Port gewählt werden muss. In der Praxis hat sich der Defaultwert 5060 durchgesetzt, der durch das Setzen dieses Eintrages überschrieben werden kann.

bindaddr Hier kann die IP-Adresse eingetragen werden, an der Asterisk lauschen soll. Falls dieser Eintrag nicht auf die Adresse 0.0.0.0 gesetzt wird, kann bei einem Server mit mehreren Adressen (z.B. bei mehreren Netzwerkkarten) eine Adresse festgelegt werden.

context Wenn dieser Eintrag gesetzt wird, wird den einzelnen User-Eintragungen dieser Eintrag als Defaultkontext zugewiesen. Dieser Defaulteintrag kann natürlich für jeden Usereintrag überschrieben werden. Die Erklärung des *Context* ist analog zu Kapitel 4.2.1.

allow verwendet die Codecs, die mit Komma getrennt aufgelistet werden können. Dabei ist auch der Eintrag *all* möglich.

disallow schließt die Verwendung der mit dieser Option aufgelistetem Codecs aus.

tos Angabe des *Type of Service*-Bits für die SIP-Pakete. Dadurch kann das Routing verbessert werden.

- **lowdelay**: Verzögerung minimieren
- **throughput**: Durchsatz maximieren
- **reliability**: Zuverlässigkeit maximieren
- **mincost**: Verwendung des Pfades mit den niedrigsten Kosten
- **none**: keine Routing flags

Empfohlen wir die Verwendung von *lowdelay*. Viele Router (einschließlich Linux mit 2.4 Kernel, der nicht durch *IPTables* verändert wurde) bevorzugen diese Pakete und somit verbessert sich die Sprachqualität.

maxexpirey Dieser Wert setzt die maximale, erlaubte Zeitdauer einer Registrationsanfrage, wobei dieser Eintrag in Sekunden anzugegeben ist.

defaultexpirey Der Wert, der dem Eintrag **defaultexpirey** zugewiesen wird, beschreibt die Defaultlänge einer Registrierungsanfrage in Sekunden.

register Mit diesem Eintrag soll sich der Asteriskserver bei einem anderen Asteriskserver registrieren. Dies wird in der Regel nur angewendet, wenn der lokale Server eine dynamische IP besitzt und dem anderen Server mitteilen muss, wie er ihn ausfindig machen kann.
Dazu muss der Server einen dynamischen „Partnereintrag“ (peer entry) mit dem selben Namen und/oder dem *secret* besitzen. Das *secret*-Feld ist optional, falls ein *secret* auf den anderen Server spezifiziert wurde.

User-Einstellungen

type Unter welchem Kontext soll diese Entität betrachtet werden?
Entity-Typ des Klienten:

- **peer**: Dieser Klient kann nur angerufen werden, darf aber selber keine Verbindung aufbauen
- **user**: Dieser Klient kann nur anrufen, aber nicht angerufen werden
- **friend**: kann Anrufe senden und empfangen

Im Zweifel sollte **friend** gewählt werden

host Die IP-Adresse oder der Hostname des Kommunikationspartner wird mit diesem Eintrag festgelegt. Wird der Eintrag mit dem Wert *dynamic* verwendet, so registriert sich der Gegenüber direkt am Asterisk.

defaultip Bei der Verwendung einer dynamischer Registrierung kann hier trotzdem eine Defaultadresse angegeben werden, falls die Registrierung nicht möglich war oder ein *Time out* aufgetreten ist.

callerid Die vom Nutzer übertragene *CallerID* kann (wenn mitgesendet) überschrieben werden, damit, falls er unterschiedliche Klientenmaschinen nutzt, immer der gleiche Bezeichner beim Kommunikationspartner angezeigt wird. Die CallerID wird intern benutzt und an das PSTN gesendet, falls eine PRI Verbindung benutzt werden soll um die Anrufe in die „Welt“ zu leiten.
Falls auch die Telefonnummer übertragen werden soll, muss diese expliezit eingetragen werden.
Ist die *CallerID* nicht gesetzt wird nur der lokale Name des Rechners, auf dem der Klient z.B. sein Softphone in Betrieb hat, übertragen.

context Einer oder mehrere Kontexte können für einen User angegeben werden. Damit können die Anrufe des Users in spezielle Kontexte eingeteilt werden.
Kontexte werden von Asterisk genutzt, um Dial-Pläne in logische Einheiten einzuteilen, in denen jeweils Nummern verschieden interpretiert werden können, damit ein eigenes Sicherheitskonzept angewandt werden kann, Unterstützung beim Umschalten (auxillary switch handling) und andere Kontexte können mit einbezogen werden.
Die meisten User geben den Zugriff auf ihren Defaultkontext frei. Vertrauenswürdigen Usern kann z.B. der Zugriff auf lokale Kontexte gewährt werden.
Wird diese Zeile bei einem Klienten gesetzt, so wird der Kontext aus dem *general*-Abschnitt für den Nutzer überschrieben.

dtmfmode Eine der Fähigkeiten von Asterisk ist das bereitstellen eines VoiceMenus, bei dem über Tasteneingaben bestimmte Befehle/Extensions ausgeführt werden. Dies geschieht über das Tonfrequenzwählsystem DTMF, das folgende in/out-band Sendemodi kennt:

- **inband**: *in-band*
- **rfc2833**: *out-band* nach RFC 2833
- **info**: *out-band* mit INFO-Nachrichten (sehr selten eingesetzt)

Als Default-Wert ist **rfc2833** festgelegt.

mailbox Ordnen dem Gegenüber eine Mailbox zu, so das dieser, wenn er sich registriert, alle offenen Nachrichten angezeigt bekommt.

qualify Gibt an, ob der Gesprächspartner auf „Lebenszeichen“ überprüft werden soll bevor ein Verbindungsversuch unternommen wird.
Wird dies gewünscht, so kontaktiert Asterisk den Gegenüber regelmäßig, bevor er Anrufinformationen weiterleitet. Als Argument wird die Zeit in Millisekunden angegeben, die Asterisk höchstens wartet bevor der Klient als *nicht verfügbar* eingestuft wird.

secret Mit diesem Eintrag wird dem Nutzer ein Passwort zugeordnet. Falls kein secret angegeben wird, ist keine Authentifizierung möglich.

nat Dieser Eintrag kann auf den Wert *yes* oder *no* (oder weggelassen) gesetzt werden und mit dieser Option kann eventuell mit einem Klient hinter einer NAT-Gateway auf einen Server außerhalb des privaten Netzes zugriffen werden.

4.2.3 extensions.conf

Diese Konfigurationsdatei ist die mächtigste im gesamten Asteriskpaket. Im Prinzip können mit ihr komplexe Dial-Pläne realisiert werden. Dies bedeutet, dass bei einer Verbindungsanfrage auf einer bestimmten Telefonnummer bestimmte Aktion ausgeführt werden. So kann

z.B. definiert werden, dass bei dem Anruf einer bestimmten Nummer ein bestimmter Klient angerufen werden soll. In dieser Datei kann ebenfalls festgelegt werden, ob bei dem Anruf zu einer bestimmen Nummer eine mp3-Datei abgespielt, eine Sprachnachricht hinterlassen werden kann, ein Anruf an mehrerer Personen weitergeleitet, oder ein VoiceMenu aufgerufen wird.
Außerdem können auch bestimmte Aktionen in Abhängigkeit von dem Anrufer ausgeführt werden. Dies kann z.B. sinnvoll sein, wenn man nicht mit bestimmten Personen reden will (Stichwort: Ex-Freundin-Regel).
Diese Datei besteht auch aus einen globalen Bereich, der im Gegensatz zu der iax.conf und sip.conf eine untergeordnete Rolle spielt und danach werden die Kontexte definiert.
Im Kapitel 9.2.5 wird eine extensions.conf vorgestellt und in diesem Kapitel werden nur grobe Verfahrensweisen skizziert.
Verallgemeinert kann man sagen, dass eine Zeile der extensions.conf folgendes Aussehen hat:

```
exten => <wert>,<priorität>,<application>[(<args>)]
```

Parallel ist noch folgende Zuweisung möglich, bei der die Argumente nicht in Klammern, sondern mit Kommas getrennt hinter dem Applikation stehen:

```
exten => <ziel>,<priorität>,<application>[,<args>]
```

Jede Extension wird mit **exten** => eingeleitet. Dann folgt das **ziel** (bzw. die Nummer, die gewählt wurde) der Extension, für das diese Regel gilt. Ausserdem werden noch *t*, *i* und *s*-Extensions von Asterisk bereitgestellt, die bei einem *timeout*, im Fall eines *invalid* und eines neuen Anrufes (*start*) ausgeführt werden.
Falls die Extensions nicht bei einem bestimmten Ziel aktiviert werden soll, sondern auf ein Muster anspringen soll, kennt Asterisk folgende Möglichkeiten, die mit einem _ eingeleitet werden:

X : Diese Stelle der gewählten Telefonnummer kann eine 0 bis 9 sein

N : Diese Stelle des *Ziels* kann eine 2 bis 9 sein

[**3-5**] Diese Stelle des *Ziels* kann eine 3, 4 oder 5 sein

. Egal welches Ziel gewählt wurde, diese Extension wird immer verwendet

Im Falle eines *Ziels* von **_1X** wird diese Extension immer ausgeführt, falls die Nummer Zehn bis Neunzehn gewählt wurde.
Falls bei einem Ziel mehrere Extensions ausgelöst werden sollen, bestimmt die **Priorität** die Reihenfolge. Danach wird die **Applikation** mit deren **Argumenten** aufgerufen.
Eine komplette Auflistung der zur Verfügung stehenden Applikationen ist in Abbildung 2 und 3, die auf den Seiten 22 und 23 aufgeführt sind, zu finden.
Im folgendem werden einige kurze Beispiele für Extensions vorgestellt:

Einen Klienten anrufen In der ersten Zeile wird versucht, eine Verbindung mit dem Nutzer *shol* aufzubauen. Kurz darauf besteht die Möglichkeit, ihm eine Nachricht an der Mailbox mit der Nummer '1' zu schicken, falls er sich nicht an Asterisk angemeldet haben sollte.
Dabei ist es egal, dass im Prinzip zwei gegensätzliche Anweisungen ausgeführt werden: Falls der Nutzer nicht angemeldet ist, wird die erste Anweisung übersprungen und falls ein Gespräch zustande kommt, werden die folgenden Anweisungen nicht weiter abgearbeitet.

```
exten => 1,1,Dial(IAX/shol)
exten => 1,2,Voicemail(u1)
exten => 1,102,Voicemail(b1)
```

Der dritte Eintrag erstellt 10 Sekunden nach der Extensionsion mit der Priorität Zwei eine VoiceMail. Dies macht Sinn, falls der Nutzer zwar im Augenblick angemeldet ist, aber nicht abnimmt.

Anruf mehrerer Gesprächspartner Falls man nicht eine einzelne, sondern eine Gruppe von Personen anrufen will, kann dies folgendermaßen gelöst werden:

```
exten => 1,1,Dial(IAX/shol&IAX/shol2)
```

Diese Art des Anrufens ist nicht zu verwechseln mit einer Telefonkonferenz. Bei diesem Beispiel wird das Gespräch mit dem Gesprächspartner geführt, der als erstes den Hörer abnimmt.

Anruf eines Gesprächspartners, der verschiedene Asteriskserver nutzt Falls ein Kommunikationspartner zwischen den Asteriskservern wechselt, da er z.B. des öfteren zwischen mehreren Standorten pendelt, die jeweils einen eigenen Server besitzen, kann auch dieser Nutzer immer mit einer Telefonummer erreicht werden:

```
exten =>  1,1,Dial(IAX/shol)
exten =>  1,2,Dial(IAX2/anderer_server/1)
```

In diesem Beispiel wird bei einem Anruf der Telefonnummer *1* versucht, mit dem angemeldeten Nutzer *shol* eine Kommunikationsverbindung aufzubauen. Falls dies fehlschlägt, da der Nutzer z.B. nicht angemeldet ist, wird auf dem anderen Rechner, dessen vollständiger Netzwerkname durch *anderer_server* repräsentiert wird, nach der Telefonnummer *1* nachgeschlagen und verbunden.

Telefonnummern auf einem anderen Asteriskserver nachschlagen und anrufen
Eine große Stärke von Asterisk ist die Möglichkeit, entfernte Asterisksysteme einzubinden. Dies ist z.B. empfehlenswert, falls die Anzahl der Nutzer so gross ist, dass ein Asteriskserver mit der Verbindungserstellung überfordert ist. Im weiteren ist vorstellbar, dass ein Unternehmen aus zwei geographisch getrennten Standorten besteht und jeder dieser Standorte einen eigenen Asteriskserver besitzt.
Eine sehr unzweckmäßige Möglichkeit, einen beliebigen Nutzer von dem entfernten Server anzurufen, ist eine Definition jeder Nummer des Entfernten auf dem lokalen Server. Ein Beispiel dafür, wie es schon in dem Unterpunkt *Anruf eines Gesprächspartners, der verschiedene Asteriskserver nutzt* erwähnt wurde, wäre:

```
exten =>  <ziel>,<prior>,Dial(IAX2/<rem_server>/<rem_ziel>)
```

Dies ist aber in der Implementierung sehr aufwendig und Fehleranfällig.
Eine viel elegantere Möglichkeit ist das Nachschlagen von Telefonnummern, die nicht auf dem lokalen Asterisksystem definiert sind:

```
switch => IAX2/<nutzername>:<passwd>@<rem_server>/<exten
```

Im Prinzip meldet sich mit diesem Eintrag ein Nutzer, der nur für diesen Zweck auf dem entfernten Rechner angelegt sein sollte, an und leitet die Verbindungsanfragen an den Server weiter.
Mit einer sinnvollen Telefonnummernvergabe, d.h. jede Nummer kommt insgesamt nur einmal vor, kann dieses Problem sehr effizient gelöst werden.

Einbinden von Kontexten Es besteht natürlich aus Gründen der Übersichtlichkeit die Möglichkeit, Gruppen von Anweisungen in Kontexten zusammenzufassen und diese dann wie ein Unterprogramm aufzurufen und somit einzufügen.
Im Prinzip muss man sich dies folgendermaßen vorstellen:

```
[context1]
exten => 11,1,Dial(IAX/kermit)
exten => 12,1,Dial(IAX/gonzo)
exten => 13,1,Dial(IAX/fozzie)

[context2]
exten => 21,1,Dial(IAX/statler)
exten => 22,1,Dial(IAX/waldorf)

[default]
include => context1
include => context2
```

Der **include**-Befehl ist aber noch viel mächtiger, als es auf den ersten Blick den Anschein hat. Es ist nicht nur möglich, bestimmte Kontexte einzufügen, sondern dies auch zeitabhängig zu gestalten. Im folgenden ist die Syntax des *include*-Befehls aufgelistet:

```
include => <context>[|<stunden>|<wtage>|<mtage>|<monate>]
```

Dies ermöglicht einen zeitabhängigen DialPlan, der z.B. Anrufe Nachts oder am Wochenende zu einem VoiceMail-System und zu den Arbeitszeiten direkt zu den Telefon der Klienten weiterleitet.

Extensions nach CallerID des Anrufer ausführen Des öfteren ist es erwünscht, dass eine Extensions in Abhängigkeit zu dem Anrufer ausgeführt wird, der diese Telefonnummer gewählt hat. Dies kann z.B. die beliebte *Anti-Ex-Girlfriend*-Regel sein, die verhindert, dass ein Anruf von der ehemalige bessere Hälfte zugestellt wird. Im folgenden wird ein Beispiel für die Definition vorgestellt:

```
exten => <wert>/<callerid>,<priorität>,<application>[,<args>]
```

Die Definition ist von den vorigen Kapitel bekannt, wobei noch die CallerID des Anrufers berücksicht wird.

Nutzen von Variablen In der extensions.conf ist es auch möglich, sich für öfter benutzte Ausdrücke Variablen zu definieren. Die Syntax für eine Variablenzuweisung ist die folgende:

```
[globals]
varname => ausdruck
```

Nun ist es möglich, diese Variable mit ${varname} zu nutzen. Wichtig ist hierbei, dass die Variablen in dem Kontext [*globals*] definiert werden müssen. Außerdem existieren folgende vordefinierte Variablen:

- **${CONTEXT}**: Der Name des aktuellen Kontextes
- **${EXTEN}**: Die aktuelle Extension
- **${PRIORITY}**: Die aktuelle Priorität
- **${CALLERID}**: Die aktuelle CallerID (Name und Nummer)
- **${CALLERIDNUM}**: Die Nummer der aktuellen CallerID
- **${CALLERIDNAME}**: Der Name in der aktuellen CallerID
- **${RDNIS}**: Die aktuell umgeleitete DNIS

Definition von Makros Es ist gut vorstellbar, dass, sobald z.B. für mehrere Nutzer jeweils mehrere, aber für jeden Nutzer die gleichen Anweisungen in der *extensions.conf* definiert werden, die Übersichtlichkeit darunter leiden kann.
Aus diesem Grund unterstützt Asterisk die Möglichkeit, Makros zu erstellen und diese einzubinden. Im Prinzip wird ein Makro wie ein Unterprogramm mit Übergabe, wie es in den meisten höheren Programmiersprachen bekannt ist, generiert und dann mit diesen Attributen aufgerufen. Im folgenden wird ein kleines Beispiel für eine Makrodefinition vorgestellt:

```
[dial]
exten =>  s,1,Dial(IAX/${ARG1})
exten =>  s,2,Dial(SIP/${ARG1})
```

Wie zu erkennen, werden die Anweisungen, die mit dem *s*-Wert belegt sind, nacheinander abgearbeitet, wobei die Priorität eine grosse Rolle spiele. Insgesamt stehen innerhalb eines Makros folgende vordefinierte Variablen zur Verfügung:

- **${MACRO_EXTEN}**: Der Wert(z.B. Anruf), mit dem das Makro aufgerufen wurde
- **${MACRO_CONTEXT}**: Der Kontext, in dem das Makro aufgerufen wurde
- **${MACRO_PRIORITY}**: Die Priorität der Anweisung, mit dem das Makro aufgerufen wurde
- **${ARG*n*}**: Die Übergabevariablen, die mit *n* durchnummeriert werden.

Das oben vorgestellte Makro läßt sich nun mit dem folgendem Befehl aufrufen:

```
exten => 1,1,Macro(dial,shol)
```

Ein einfaches VoiceMenu Bei einem Anruf der Nummer 99 wechselt Asterisk in den Kontext *voicemenu* und eine Ansage wird abgespielt. Bei einer Übermittlung einer ’1’ oder einer ’2’ über DTMF wird der passende Klient angerufen.

```
[voicemenu]
exten => s,1,Background(ansage)
exten => 1,1,dial(IAX/gonzo)
exten => 2,1,dial(IAX/kermit)

[default]
exten => 99,1,Goto(voicemenu,s,1)
```

4.3 VoiceMail-System

Wie bei der herkömmlichen Kommunikation über ein Telefon kann der angerufene Gesprächspartner nicht in der Nähe seines Telefons sein oder kann schon mit einem anderen Kommunikationspartner telefonieren und kann somit das Gespräch nicht annehmen.
Bei der Telekommunikation, die über Computer stattfindet, besteht noch eine weitere Möglichkeit, weswegen eine Verbindung verhindert werden kann: Der Rechner kann ausgeschaltet sein oder der Anzurufende hat seine Telekommunikationssoftware nicht gestartet und ist somit nicht auf dem Asteriskserver angemeldet.
In den ersten beiden Fällen könnte der nicht erreichte Teilnehmer einfach den Anrufer zurückrufen, da in den meisten Fällen die Telefonnummer übertragen und angezeigt wird, falls dies von der Telefonappikation bzw. Hardwaretelefon unterstützt wird. Dies ist aber nicht möglich, falls der Angerufende nicht angemeldet ist und nicht immer ist ein Rückruf die eleganteste Lösung.
Daher existiert in Asterisk ein VoiceMail-System, bei dem der Anrufer dem gewünschten Gesprächspartner eine Sprachnachricht hinterlassen kann. Im Gegensatz zu einem herkömmlichen Anrufbeantworter werden die Nachrichten direkt auf dem Server und nicht lokal hinterlassen.
Wenn nun eine Nachricht hinterlegt wurde, wird dies, falls es von dem Soft- bzw. Hardphone unterstützt wird, direkt angezeigt. Außerdem wird noch eine EMail an den Nutzer geschickt, wann und von wem er angerufen wurde. Im weiteren besteht noch die Möglichkeit, die Informationen über eine eingetroffene Nachricht an einen Pager zu schicken.
Nun hat der Nutzer zwei Möglichkeiten, an die für ihn hinterlassende Nachricht zu gelangen:

- Die Benachrichtungs-EMail beinhaltet direkt als Anhang die Nachricht in Form einer Audiodatei
- Über ein Webinterface, dass bei der Übersetzung der Asteriskquellen erstellt wurde (siehe Kapitel 2.1.3), können die Nachrichten bereitgestellt werden.

Die Einstellungen, die VoiceMail betreffen, sind in der Konfigurationsdatei *voicemail.conf* in dem Verzeichnis */etc/asterisk* zu finden und werden dort festgelegt.

4.3.1 Konfiguration

Die Datei *voicemail.conf* regelt, vereinfacht gesagt, welcher Nutzer welches Postfach besitzt und an welcher Mailadresse die E-Mail mit der Sprachmitteilung geschickt werden soll.
Wie alle anderen Konfigurationsdateien besitzt auch die /etc/asterisk/voicemail.conf einen globalen Bereich und die Möglichkeit, die restlichen Eintragungen in Kontexten zu unterteilen.

General-Bereich

format Dieser Eintrag wählt das Audioformat aus, mit dem Asterisk die VoiceMails an die Nutzer verschickt. Zu Auswahl stehen folgende Formate:

- **gsm**: GSM-Kodierung
- **wav**: MS WAV-Format, 16 Bit linear
- **WAV**: MS WAV-Format, GSM kodiert
- **g723sf**: G.723.1-kodiert

Ein Beispiel:

```
format=wav
```

fromstring Dieser Eintrag löst den veralteten Eintrag **servermail** ab und ist dafür vorgesehen, in der From-Zeile der zugeschickten Mail einen Absender anzugeben.
Die From-Zeile der E-Mail kann folgendermaßen modifiziert werden:

```
fromstring=Asterisk
```

append Dieser Schalter gibt an, ob in der E-Mail das Audiofile als Attachment verschickt werden soll oder nicht. Dementsprechend lassen sich folgende Werte festlegen:

```
append=yes
```

Bei dieser Einstellung wird an der EMail die hinterlassende Nachricht angefügt.

```
append=no
```

Hier wird der Nutzer nur benachrichtigt, dass er eine Sprachmitteilung erhalten hat.

maxmessage Dieser numerische Wert gibt die maximale Länge einer Sprachmitteilung in Sekunden an.

maxgreet Dieser Eintrag wird ebenfalls mit einen numerischen Wert belegt, der die Länge für die *Besetzt*- und *Nicht verfügbar*-Nachrichten der Nutzer einschränkt. Dieser Wert wird in Sekunden angegeben.

Des öfteren stößt man in der Literatur auf die Einträge für **skipms**, **maxsilence**, **silencethreshold**, **maxlogins**, **pbxskip** und **emailbody**, die aber noch nicht in der zur Studienarbeit zur Verfügung stehenden Asteriskversion implementiert waren. Anwender, die hier auf angepasste Werte angewiesen sind, wird empfohlen, die Quellen des VoiceMenu-Moduls (voicemail.c) anzupassen, manuell zu kompilieren und einzufügen.

Zuweisung der Mailboxen Die Syntax für eine Festlegung ist folgendermaßen definiert:

```
<MailboxNr> => <login>,<Name>,<EMail>[,<pager>,<optionen>]
```

Da dieser Eintrag selbsterklärend ist, muss nicht tiefer darauf eingegangen werden, dass sich mit jeder Zeile die Nummer der Mailbox, die Benutzerkennung (beides in sip.conf oder iax.conf definiert), der vollständige Name und die E-Mail-Adresse, an der die Mitteilung geschickt werden soll, festgelegt wird. Ausserdem besteht noch die Möglichkeit, sich eine Benachrichtung auf den Pager schicken zu lassen. Die im globalen Bereich definierten Optionen können mit Lokalen überschrieben werden, die mit einem | abgetrennt werden.

4.4 Verwaltung dauerhaft festgelegter Variablen

Für den Fall, dass Werte von Variablen nach einem Neustart des Asteriskservers benötigt werden, stellt Asterisk die Datenbankfunktionalität zur Verfügung. Diese Funktionalität ist leider nicht ausreichend dokumentiert, daher wurde die Funktionsweise der folgenden Befehle durch Experimente des Autors dieser Studienarbeit ermittelt.
Alle Befehle können in der *extension.conf* oder direkt unter dem CLI definiert werden, wobei eine parallele Nutzung einiger Befehle nicht immer von Nutzen ist.

Definition Um überhaupt erst einmal auf die Variablen der Datenbank zugreifen zu können, müssen diese erst dort abgelegt werden.
Die Werte können mit der Extension

```
exten => <ziel>,<prior>,DBput(<Gruppe>/<Key>=<wert>)
```

in die Asteriskdatenbank übertragen werden. Wie in dem Beispiel ersichtlich, wird als Identifikator eine Untergruppe und ein *Key*-Name benötigt, um einen *Wert* abzuspeichern.
Analog kann die Variable mit dem CLI-Kommando

```
voip*CLI> database put <gruppe> <key> <wert>
Updated database successfully
```

in der Datenbank abgelegt werden.
Falls die Variablen mit neuen Werten belegt werden sollen, müssen die Variablen erst gelöscht werden.

Anzeige der Werte Eine Anzeigemöglichkeit der definierten Datenbankeinträge in der *extension.conf* macht relativ wenig Sinn und daher ist auf das CLI zurückzugreifen:

```
voip*CLI> database show
/<gruppe>/<key>: <wert>
```

Löschen von Einträgen Es muss unterschieden werden, ob alle Einträge einer *Gruppe* oder nur ein bestimmter *Key* gelöscht werden soll. Ersteres kann mit der Extension

```
exten => <ziel>,<prior>,DBdeltree(<gruppe>)
```

oder dem CLI-Kommando

```
voip*CLI> database deltree <gruppe>
Database entries removed.
```

ausgeführt werden.

Um nur bestimmte Werte einer Gruppe zu löschen, kann entweder die Extension

```
exten => <ziel>,<prior>,DBdel(<gruppe>/<key>)
```

definiert werden oder mittels dem CLI-Kommando

```
voip*CLI> database del <gruppe> <key>
Database entry removed.
```

der Speicher für die Variable freigegeben werden.

Auslesen der Werte Da es relativ wenig Sinn macht, über das CLI die Werte für die Datenbankeinträge zu holen, da diese dort nicht weiter verwendet werden können, sollte diese Funktionalität in der *extensions.conf* genutzt werden. Mittels der Extension

```
exten => <ziel>,<prior>,DBget(<varname>=<gruppe>/<key>)
```

kann der Datenbankeintrag in eine Variable geschrieben werden. Diese Variable kann mit *${varname}* genutzt werden.

5 Klienten

5.1 GnoPhone

Als erste Kommunikationsmöglichkeit ist GnoPhone[18] zu nennen. Diese Software ist eine der wenigen bisher verfügbaren Softphones, die die Kommunikation über das IAX-Protokoll mit Asterisk ermöglichen.
Wegen der fehlenden Dokumentation und der oberflächlichen Homepage[18] des Softwareprojektes ist leider kein Autor ausfindig zu machen. Aber vermutlich ist dies beabsichtigt, da GnoPhone, dessen letzte veröffentliche Version mit der Nummer 0.2.4 vom 13. November 2001 stammt, sehr fehlerhaft ist. Da die Firma Digium GnoPhone unter *Software Products* auf ihrer Homepage verlinkt, ist davon auszugehen, dass Digium bei der Entwicklung beteiligt war. Eine der wenigen Informationen ist der Hinweis, dass es sich bei GnoPhone um freie Software handelt, wobei aber keine Lizenz genannt wird.
Obwohl die im Kapitel 5.1.3 aufgelisteten Fehler nicht ignoriert werden können, ist GnoPhone für IAX-Testreihen ausreichend. Ein weiterer Vorteil von GnoPhone sind die Debuggingmeldungen, die über versendeter und empfangener IAX-Pakete auf der Standardausgabe geschrieben werden.
Für das Betriebssystem Windows ist kein IAX-fähiger Klient bekannt, aber es wurde eine Belohnung von 500$ für die Implementierung eines Softphones ausgesetzt[19].

Abbildung 5: GnoPhone

5.1.1 Installation

Für die Installation von GnoPhone existieren zwei Möglichkeiten:

1. Übersetzung der Quellen
2. Installation eines Binärpaketes über die Linuxdistribution

Die Übersetzung der Quellen ist sehr aufwendig, da eine Installation der GSM- und IAX-Headerdateien benötigt werden (alles zu finden auf der GnoPhone-Homepage[18]). Daher ist die Installation des Binärpaketes die bessere Wahl.

5.1.2 Konfiguration

Unter den Menüpunkt *Preferences*/*Telephone* lassen sich alle Einstellungen festlegen, die für eine Nutzung des Asteriskservers benötigt werden. Das Fenster mit den möglichen Einstellungen ist unter 5.1.2 zu finden. Die Adresse bzw. der Netzwerkname und der Port des Servers

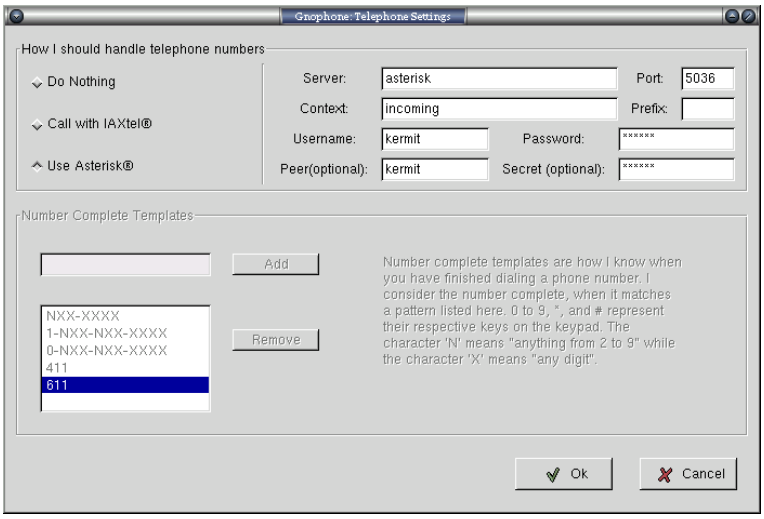

Abbildung 6: Konfigurationsfenser von GnoPhone

erfordert keine Erklärung. Die Einstellungen für den *Context* kann aus der iax.conf übernommen werden. Unerklärlich ist die Tatsache, dass zwischen *Username* und *Peer* unterschieden wird, da dies auch nicht bei dem einzigen IAX-Server Asterisk der Fall ist. Die beiden Werte müssen den gleichen Eintrag besitzen, was auch bei *Password* und *Secret* der Fall ist. Tests haben ergeben, dass der Username und das Passwort für das Anrufen eines Kommunikationspartners erforderlich ist, wogegen Peer und Secret für das Anmelden am Server benötigt werden.

5.1.3 Fehlfunktionen

Bei der *normalen* Verwendung sind folgende Fehlfunktionen oder Verbesserungsmöglichkeiten aufgefallen:

- Keine akustische oder „aufdringliche“ visuelle Meldung über eintreffenden Anrufe
- Obwohl GnoPhone und IAX dies unterstützt, werden die empfangen Bilder des Gesprächspartners teilweise nicht angezeigt.
- Falls Bildübertragung aktiviert, ist es nicht möglich, Textnachrichten zu verschicken und zu empfangen.
- Stürzt oft mit der Fehlermeldung *Read Error* ab, falls der Rechner parallel mit anderen CPU-hungrige Aufgaben genutzt wird.
- Fehlende Dokumentation (noch nicht einmal der Autor ist ersichtlich).
- Die Hälfte der Einträge der Menüleiste sind unbelegt oder führen zum Absturz.
- Keine Meldung, ob VoiceMail hinterlassen wurde.

Die Fehlfunktion sind nicht bei ausführlichen Testreihen, sondern im Alltagsgebrauch aufgetreten.

5.2 tkPhone

Seit die Version 0.1 am 24. Juni 2003 erschienen ist, gibt es ein weiteres IAX-fähiges Softphone: tkPhone[20]. Diese Applikation wird von Reinhard Max gepflegt und steht unter der GPL. tkPhone ist noch relativ jung und daher ist der Funktionsumfang noch sehr eingeschränkt. Aber da es jetzt schon stabiler als GnoPhone läuft, kann in der Zukunft von diesem Tcl-basiertem Tool viel erwartet werden.

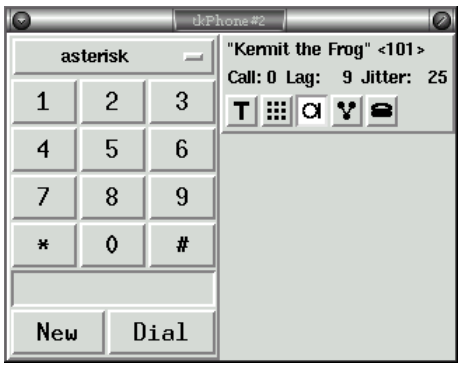

Abbildung 7: tkPhone mit Anruf von *Kermit the Frog*

5.2.1 Verfügbarkeit und Installation

Die Quellen dieses Softwarepakets wurden von dem Autor unter der Adresse `http://max.ddts.net/tclphone-src.tar.gz` bereitgestellt. Leider liegt keine ReadMe-Datei dem Quelltext bei, aber die Übersetzung kann mit den Befehlen *make* und *make install* ausgeführt werden.
Da sich das Programm aber noch in einem frühen Entwicklungsstadium befindet, ist kein *configure*-Script vorhanden, dass die *Makedatei* anpasst. Aus diesem Grund sollten, falls die Übersetzung fehlschlägt, die vorcompilierten Binärpakete aus `http://max.ddts.net/tclphone-bin.tar.gz` verwendet werden.

5.2.2 Konfiguration

Im Gegensatz zu GnoPhone wird bei tkPhone die Konfiguration nicht über ein Fenstermenü, sondern per Konfigurationsdatei vorgenommen. Die Datei *.tkphonerc*, die sich im Homeverzeichnisses des Benutzers befindet, hat folgendes Aussehen:

```
# -*- Tcl -*-
# # configuration file for tkPhone
array set registrations {
VOIP1   user:password@server1/EXT@context
VOIP2   user:password@server2/EXT@context
}
```

Abbildung 8: tkPhone-Konfigurationsdatei

Für die Werte *user*, *password* und *contex* sind die Einträge von *peer=*, *secret=* und *context=* der iax.conf des Asterisk-Servrs zu übernehmen. Für den Wert *server* muss der Netzwerkname des Servers eingetragen werden. Interessanterweise können bei tkPhone mehrere Server eingetragen werden, zwischen denen gewechselt werden kann.

5.2.3 Kinderkrankheiten

Dieser Programm befindet sich noch in der Entwicklung und daher sind die meisten Funktionen, die mit einem IAX-Clienten möglich sind, noch nicht implementiert. Außerdem ist es sehr stark auf die Arbeitsumgebung des Entwicklers angepasst und bei einer Abweichung dieser nicht nutzbar.
tkPhone wäre auch für den Endanwender anwendbar, wenn mindestens folgende Punkte implementiert werden würden:

- akustische und/oder visuelle (in Form eines im Vordergrund springenden Fensters) Benachrichtigung bei dem Empfang einer Verbindungsanfrage
- Neben OSS weiter Standards, Soundhardware einzubinden
- Dokumentation

5.3 kphone

Als weiteres Softphone ist kphone[21] zu nennen, dass zwar nicht das IAX-Protokoll beherrscht, aber die Gruppe der SIP-Klienten repräsentiert. Da in Asterisk die SIP-Unterstützung mit der des IAX-Protokolls gleichzusetzen ist und die IAX-Softphones noch nicht für den Nutzereinsatz geeignet sind, ist Asterisk in Verbindung mit SIP eine gute Alternative.

Abbildung 9: das SIP-Telefon kphone

Das Programm kphone wurde von Billy Biggs geschrieben und von den Mitarbeitern der Firma Wirlab, Pekka Raisio, Jouni Vuorela und Juha Heinanen weiterentwickelt. kPhone steht unter der General Public License(GPL) und besitzt als besondere Features Videoconferencing, ein Adressbuch und die von den IAX-Klienten vermisste Benachrichtigung bei einem eintreffenden Anruf (sowohl akustisch als auch als ein in Vordergrund springende Fenster).

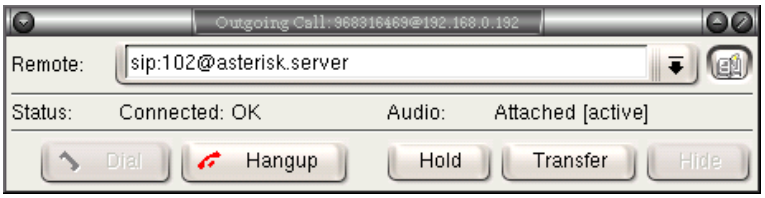

Abbildung 10: kphone-Benachrichtigung bei einem Anruf

5.3.1 Installation

Um dieses Paket zu installieren, müssen sehr viele Softwareabhängigkeiten gelöst werden. So wird z.B. die KDE-library benötigt. Der Anwender, der sehr viel Zeit investieren kann, um diese Abhängigkeiten aufzulösen, kann sich von der kPhone-Homepage[21] die Quellen ziehen und compilieren.
Der empfohlene Weg ist die Verwendung von vorcomplierten Binärpaketen, die für jede Distribution zur Verfügung stehen sollten.

5.3.2 Konfiguration

kPhone wird wie in Abbildung 5.3.2 graphisch konfiguriert. Die Eintragungen sind aus der

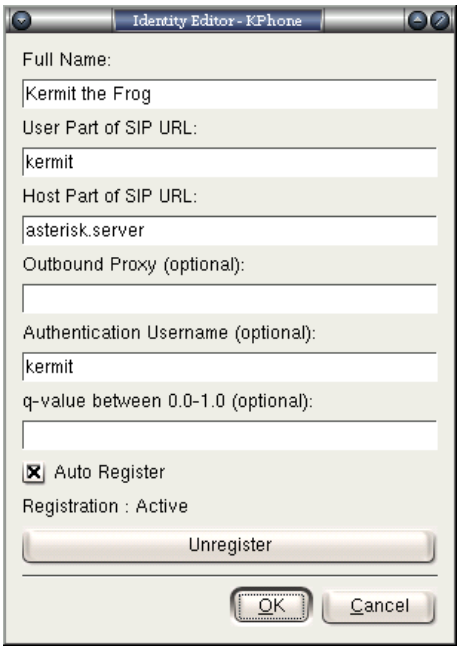

Abbildung 11: Konfiguration von kphone

sip.conf des Asteriskservers zu übernehmen. Die Ausnahme ist der Eintrag des *Host Part*, der den vollständigen Netzwerknamen des Servers trägt und nicht in der sip.conf zu finden ist.

5.4 Snom 100

Das *snom 100* ist ein VoIP-Telefon, dass die Firma *snom* mit einer Preisempfehlung von 299 Euro anbietet. Der Marktpreis liegt zwischen 150 und 200 Euro. Das Gerät besitzt einen 10/100 Mbit Netzwerkanschluss, einen 50MHz Motorola Prozessor und es stehen 4 MB Flash und 16 MB RAM zur Verfügung.
Als Betriebssystem ist ein Linux installiert, dass von *snom* mit einer H.323-, H.450-, SIP- und LPCP-Unterstützung ausgestattet wurde. Wie auch die *snom*-Telefone 105 und 200 sind diese Geräte die einzigen bekannten, die die Möglichkeit besitzen, mit einer IAX-Unterstützung erweitert zu werden. Eine Anleitung hierfür ist unter [23] zu finden.
Die Konfiguration des IP-Telefons ist detailliert in dem mitgelieferten Handbuch beschrieben, wobei das intuitiv zu bedienende Webinterface erwähnt werden muss. Eine Konfiguration ist hierbei über den http-Port 80 mit einem beliebigen Browser möglich. Dieses Interface eignet sich auch sehr gut für Testzwecke, da hierüber auch andere Teilnehmer angerufen werden können.

Abbildung 12: Snom IP-Telefon (Quelle des Bildes: http://www.snom.com)

5.5 Nutzer- und Klientenabhängige Einstellungen

Leider besitzt Asterisk, beziehungsweise IAX, einen groben Mängel: Die Einstellungen über die Nutzer können nur *hart* in den Konfigurationsdateien festgelegt werden.
Es ist nicht möglich, dass ohne Umwege die Nutzer z.B. ihr Passwort ändern oder eine Neuregistierung ausführen können. Falls ein Nutzer sich von verschiedenen Klienten mit verschiedenen Netzwerkanbindungen zu verschiedenen Zeiten anmelden soll, wäre auch eine Anpassung der verwendeten Codecs nach Bandbreite sehr zu empfehlen, aber bisher noch nicht vorgesehen.
Daher wäre eine Nutzerverwaltung, bei der die Nutzer die Änderungen selber vollziehen können, wünschenswert. Dieser Wunsch könnte mit einem Script, z.B. auf *CGI*-Basis, welches die Konfigurationsdateien modifiziert und die Änderungen Asterisk mitteilt, erfüllt werdem.

6 Module

6.1 Was sind Module?

Module ermöglichen einen Entwickler, der das Asterisksystem auf seine Bedürfnisse anpassen will, Asterisk ohne Probleme mit seinen eigenen Quellcode zu erweitern. Im Prinzip sind Module als Zusatzprogramme zu verstehen, die der Entwickler einbinden kann, ohne den Quelltext von Asterisk zu modifizieren.
Ein weiterer Vorteil einer Modulschnittstelle gegenüber die Möglichkeit, den eigenen Quellcode direkt im Asteriskquellcode zu verankern, ist der Zeitaufwand und die Kapazität, die gespart wird, da nur das Modul und nicht das gesamte Asteriskpaket neucompiliert werden muss.
Außerdem erscheinen relativ oft neue Versionen von Asterisk und im Fall eines Updates können die Module ohne Probleme wieder eingebunden werden.
Ein Asteriskmodul muss in der Programmiersprache C implementiert und compiliert in dem Verzeichnis */usr/lib/asterisk/modules* abgelegt werden.
Asterisk kennt vier große Gruppen, denen die Module zugeordnet werden können:

- Applikationen, die bestimmte Aktionen ausführen (z.B. Musik abspielen, wählen, Bilder übertragen)
- Kanäle, über die eine Kommunikation aufgebaut werden kann (z.B. iax, SIP, H.323, OSS, ALSA)
- Codecs für die Verwendung und Manipulation von Audiodatein (z.B. mp3, gsm, A-law)
- Dateiformate, die Asterisk verarbeiten kann (z.B. jpeg, wav, mp3)

In der Konfigurationsdatei */etc/asterisk/modules.conf* können Module aktiviert werden, falls sie nicht schon direkt im Asteriskquellcode eingebunden sind, oder deren Verwendung ausgeschlossen werden.

6.2 Verfügbare Asterisk-Module

Nach einer Asteriskübersetzung stehen über achtzig Module zur Verfügung und daher würde bei einer kompletten Auflistung und Beschreibung die Übersichtlichkeit dieser Studienarbeit sehr darunter leiden.
Aber eine Kurzbeschreibung aller zur Verfügung stehenden Asterisk-Module lässt sich mit dem CLI-Kommando aus der Abbildung 6.2 auflisten:

```
voip*CLI> show modules
```

Abbildung 13: CLI-Kommando, um alle verfügbaren Module aufzulisten

6.3 Aufbau eines Moduls

Im folgendem wird der Aufbau und die Schnittstelle der Module vorgestellt. Diese Informationen stammen aus den Quelltexten der existierenden Module.

6.3.1 Zwingend erforderliche Funktionen

Die Schnittstelle zu der Asteriskapplikation sind die folgenden Funktionen, die in jedem Modul definiert werden müssen:

int load_module(void) In dieser Funktion werden alle Anweisungen geschrieben, die beim Laden des Moduls aufgerufen werden sollen. Diese Anweisungen können z.B. Hardware steuern, andere Funktionen aufrufen, Registrierungen durchführen oder einfach nur normale Anweisungen in der Programmiersprache C beinhalten.
load_module gibt als Rückgabewert eine Null zurück, falls keine Probleme aufgetreten sind.

int unload_module(void) Für ein Entfernen des Moduls aus dem laufenden Asterisksystem wird die Funktion *unload_module* ausgeführt. Die Betriebsmittel, die bei dem Aufruf der Funktion *load_module* reserviert wurden, müssen freigegeben werden und die Registrierungen rückgängig gemacht werden.
Auch hier ist der Rückgabewert Null, falls keine Probleme aufgetreten sind.

int usecount(void) Diese Funktion gibt an, wieviele der durch das Modul bereitgestellten Kanäle genutzt werden. Sie wird von vielen verschiedenen Teilen von Asterisk verwendet. Diese Anzahl muss in dem Modul selbst verwaltet werden.

char *description(void) Der String, der in dieser Funktion definiert wird, beinhaltet eine kurze Beschreibung des Moduls. Diese wird. z.B. von dem CLI-Kommando *show modules* ausgegeben.

char *key(void) Diese Funktion gibt den angegebenen Schlüssel zurück. Durch die Rückgabe des Schlüssels ASTERISK_GPL_KEY wird die Zustimmung zur GPL gegeben, die dieser Schlüssel enthält. Erst danach ist es möglich, ein Modul zu laden.

int reload(void) *Reload* beinhaltet alle Anweisungen, die bei einem Neuladen des Moduls ausgeführt werden müssen. Dies kann z.B. das Neu-einlesen von Konfigurationsdateien oder Änderungen am Signalverhalten sein. In einigen Fällen können dies die Anweisungen aus den Funktionen *unload_module* und *load_module* sein.

6.3.2 Beispiel für ein Modulquellcode

Es folgt ein Listing eines Beispielmoduls. Dieses Beispiel soll nur den Aufbau skizzieren und hat sonst keine weitere Funktionalität. Es ist in der Datei *apps/app_skel.c* in den Asteriskquellen zu finden.

```
/*
 * Asterisk – A telephony toolkit for Linux.
 *
 * Skeleton application
 *
 * Copyright (C) 1999, Mark Spencer
 *
 * Mark Spencer <markster@linux-support.net>
 *
 * This program is free software, distributed under the terms of
 * the GNU General Public License
*/

#include <asterisk/file.h>
#include <asterisk/logger.h>
#include <asterisk/channel.h>
#include <asterisk/pbx.h>
#include <asterisk/module.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

#include <pthread.h>


static char *tdesc = "Trivial skeleton Application";

static char *app = "skel";

STANDARD_LOCAL_USER;

LOCAL_USER_DECL;

static int skel_exec(struct ast_channel *chan, void *data)
{
        int res=0;
        struct localuser *u;
        if (!data) {
                ast_log(LOG_WARNING, "skel requires an argument (filename)\n");
                return -1;
        }
        LOCAL_USER_ADD(u);
        /* Do our thing here */
        LOCAL_USER_REMOVE(u);
        return res;
```

```
}

int unload_module(void)
{
        STANDARD_HANGUP_LOCALUSERS;
        return ast_unregister_application(app);
}

int load_module(void)
{
        return ast_register_application(app, skel_exec);
}

char *description(void)
{
        return tdesc;
}

int usecount(void)
{
        int res;
        STANDARD_USECOUNT(res);
        return res;
}

char *key()
{
        return ASTERISK_GPL_KEY;
}
```

7 Das Inter-Asterisk eXchange Protokoll

Dieses Kapitel beinhaltet eine Beschreibung der Fähigkeiten und erklärt die Funktionsweise des Asterisk-Protokolls IAX. Es existiert bisher keine Dokumentation, die sich mit diesem Thema beschäftigt. Daher wurde als Informationsquelle der Quelltext der Datei *iax.h*, die Debuggingausgaben und Mitschnitte des Netzwerkverkehrs (*sniffing*) herangezogen.

7.1 Warum IAX?

Es stellt sich natürlich die Frage, warum die Entwickler von Asterisk ein neues Protokoll eingeführt haben, obwohl die Protokolle SIP und H.323 so weit verbreitet sind und unterstützt werden. Einer der Hauptgründe, warum sich die Entwickler dazu entschlossen haben, war die Tatsache, dass ein Protokoll benötigt wurde, dass sich besser an die Funktionalität von Asterisk anpassen konnte. Außerdem ist IAX aus folgenden Gründen interessant und einer näheren Betrachtung wert:

Problemlose Kommunikation mit Klienten hinter NAT- und Masquerade-Gateways IAX besitzt die Fähigkeit, mit Servern außerhalb eines privaten Adressbereiches zu kommunizieren, während sich die Maschine innerhalb dieses befindet. Dabei spielt es keine Rolle, ob die Maschine im privaten Adressbereich ein Server oder ein Klient ist und natürlich können auch VoIP-Pakete gesendet, empfangen und weitergeleitet werden.

Hohe Performance bei geringem Protokolloverhead Falls nur eine schmale Bandbreite zur Verfügung steht oder viele Verbindungen auf einem Asterisk-Server bestehen, spielt die ideale Kapazitätsausnutzung von IAX eine große Rolle. IAX benötigt nur 4 Bytes Protokolloverhead, um z.B. Sprach- oder Videopakete zwischen den Kommunikationspartnern auszutauschen.

Zuordnung von Sprachen IAX übermittelt Informationen über die Sprache des Nutzers, die dann das PBX-System für weitere Funktionalitäten nutzen kann.

Flexible Authentifikationsmöglichkeiten IAX ermöglicht die Authentifikation über *plaintext*, *md5* und *RSA*. Die richtige Wahl des Authentifikationsmodus ermöglicht eine sichere Registrierung und eine flexible Sicherheitseinstellungen

Multimedia IAX unterstützt nicht nur die Übermittlung von Sprachpaketen, sondern auch von Video, Bildern, Text und HTML. VoiceMenus können sowohl akustisch als auch visuell gestaltet werden.

Statistiken sammeln und ausnutzen IAX kann Informationen und Statistiken über die Performance des Netzwerks (Latenz- und Jitterzeiten) sammeln und dieses dann für die weitere Kommunikation nutzen.

Übermittlung von Informationen Natürlich werden alle Informationen wie CallerID bei einem Gespräch übermittelt.

Single Socket Design Das Socket-Design von IAX ermöglicht bis zu 32768 Verbindungen zur gleichen Zeit.

7.2 Datagramm-Struktur

Um die Paketgröße so effizient wie möglich auszunutzen, kennt IAX zwei verschiedene Arten von Headern. Einen 12 Byte großen **Fullheader**, der z.B. Steuerfunktionen beinhaltet und einen nur 4 Byte großen **Miniheader**, der ausschließlich für den Transport von Nutzlast vorgesehen ist. Bei einer Übertragung von Nutzlast mit dem Miniheader wird in Abständen von ca. 1 Minute jeweils zu Kontrolle eine Fullheader-Kommunikation ausgeführt. IAX nutzt das *User Datagram Protocol* (UDP) als Transportprotokoll und eine IAX-Einheit ist auf 32768 Bytes beschränkt.
Beide Headerarten werden auf den folgenden Seiten vorgestellt.

7.2.1 Fullheader

Dieser Header überträgt Steuerinformationen, die für beide Kommunikationspartner wichtig sind. Jedes Paket mit einem Fullheader außer ein *ACK* oder *HANGUP* muss mit einem *ACK* bestätigt werden, wobei das *ACK* auf dem Antworten von Sequenznummern basiert. Abbildung 14 skizziert den Aufbau eines zur Steuerung vorgesehenen Datagrammes.

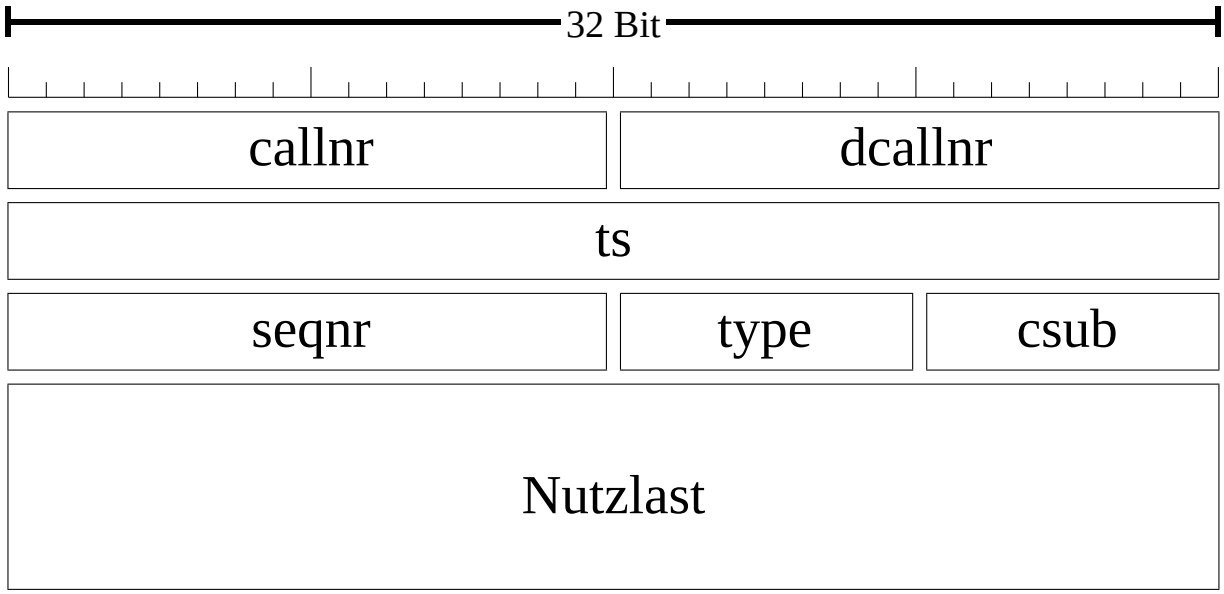

Abbildung 14: Full-Header

Im folgendem werden die Felder des Full-Header kurz vorgestellt:

callnr (Datentyp: short)
Dieser zwei Byte große Wert ordnet der Seite des Senders eine Identifikationsnummer für eine Verbindung zu.

dcallnr (Datentyp : short)
In diesem Feld trägt der Empfänger ab seiner ersten Antwort seine Identifikationsnummer der Verbindung ein. Da bei der ersten Kontaktaufnahme des Senders nicht die Nummer des Empfängers bekannt ist, wird als Default der Wert *FFh* eingetragen

ts (Datentyp: unsigned int)
Der Eintrag des „timestamps“ beinhaltet einen 32 Bit großen Zeitstempel, der als Milisekunde angegeben wird.

seqnr (Datentyp: unsigned short)
Dieses Feld repräsentiert die 2 Byte große Paketnummer. Sie wird aber nur von einem Kommunikationspartner inkrementiert. Im folgendem wird dieser Vorgang am Beispiel der Registrierung eines Klienten an einem Server skizziert:

Sender	Steuerungsfunktion	Sequenznummer
Client	REGAUTH	0
Server	ACK	0
Server	REGAUTH	0
Client	ACK	0
Client	REGREQ	1
Server	ACK	1
Server	REGACK	1
Client	ACK	1

Tabelle 2: Werte für *seqno*

Dieses Verfahren ermöglicht ein 4-Wege-Handshake-Protokoll, wie es so ähnlichen vom dem *Transmission Control Protocol* (TCP) bekannt ist.

type (Datentyp: char) Dieser Eintrag repräsentiert den Wert, von welchem Typ das Paket ist. Es werden zwischen folgenden Frame-Typen unterschieden:

Wert	Kommando	Beschreibung
01	DTMF	Möglichkeit, über das *Dual Tone Multiplexed Frequency*-Verfahren Anweisungen zu verschicken
02	VOICE	Sprachpakete
03	VIDEO	Videoübertragung (Bildtelefonie)
04	CONTROL	Kontrollnachrichten
05	NULL	Für Debugging-Zwecke reserviert
06	IAX	Pakete mit IAX-Anweisungen
07	TEXT	Kommunikation über Tastaturdialoge
08	IMAGE	Übertragung von einzelnen Bildern

Tabelle 3: Werte für *type*

csub (Datentyp: unsigned char) Der Inhalt dieses Feld ist vom dem Frame-Typ, der in dem Feld *type* eingetragen ist, anhängig. In den nächsten beiden Unterkapiteln werden die *type*-Einträge für IAX- und Kontrollkommandos aufgelistet. Falls ein *DTMF*-Datagramm genutzt wird, werden die Einträge dieses Feldes je nach dem zu übertragenden Wert nach ASCII codiert (0 = 30h). Bei den restlichen *type*-Einträgen kann leider keine genau Aussage getroffen werden, da diesen *ctype*-Werten weder in Dokumentation noch in den Quelltexten Bezeichner zugeordnet sind. Teilweise wird aber der selbe Eintrag wie im vorigen Feld eingetragen (z.B. *type*=02 und *csub*=02 bei *VOICE*-Übertragung).

csub für IAX Dieses ein Oktett große Feld beinhaltet, welche von den insgesamt 29 Steuerungsfunktionen dieses Paket bezweckt. In Tabelle 4 werden alle möglichen Feldinhalte kurz skizziert:

Wert	Kommando	Beschreibung
01	NEW	Einleitung einer Verbindung
02	PING	Test auf Erreichbarkeit
03	PONG	Antwort auf Erreichbarkeit
04	ACK	ACK. Bestätigt fast jedes Datagramm mit Full-Header
05	HANGUP	Auflegen
06	REJECT	Abweisen einer Verbindung
07	ACCEPT	„Abnehmen des Hörers“
08	AUTHREQ	Authentifizierungs-Anfrage
09	AUTHREP	Authentifizierungs-Antwort
0a	INVAL	besetzt
0b	LAGRQ	Sinngemäß: Server möchte überprüfen, ob Client aktiv
0c	LAGRP	Sinngemäß: Client möchte überprüfen, ob Server aktiv
0d	REGREQ	Registrations-Anfrage
0e	REGAUTH	Authentifikationsdaten für Registrierung benötigt
0f	REGACK	Registration akzeptiert
10	REGREJ	Registration abgelehnt
11	REGREL	*Force release* einer Registration
12	VNAK	Wird schon Sprache vor dem ersten gültigern Voicepacket gesendet, so wird dies als Antwort gesendet
13	DPREQ	Anfrage nach Status eines Dialplan-Eintrags
14	DPREP	Antwort Status eines Dialplan-Eintrags
15	DIAL	„Anrufen“
16	TXREQ	Transfer-Anfrage (Request)
17	TXCNT	Transfer-Verbindung (Connect)
18	TXACC	akzeptierter Transfer (Accepted)
19	TXREADY	Transfer bereit (ready)
1a	TXREL	Transfer release
1b	TXREJ	Transfer abgelehnt (reject)
1c	QUELCH	Audio/Video Verbindung unterbrechen
1d	UNQUELCH	Audio/Video Verbindung wieder aufnehmen

Tabelle 4: *csub*-Werte für IAX-Pakete

csub für Control Falls im *type*-Feld der Wert 04 für „CONTROL“ eingetragen wurde, stehen die folgenden 7 Werte für das Feld *csub* zur Verfügung:

Wert	Kommando	Beschreibung
01	HANGUP	Auflegen
02	RING	Klingeln
03	RINGING	Information, dass Telefon klingelt
04	ANSWER	Gespräch wurde angenommen
05	BUSY	Besetzt
06	TKOFFHK	
07	OFFHOOK	

Tabelle 5: *csub*-Werte für Control-Pakete

7.2.2 Miniheader

Um nicht unnötigen Ballast bei der Kommunikation zwischen Endgeräten mitzuführen, haben die Entwickler von IAX einen minimalen Header eingeführt. Ein Paket mit einem Miniheader transportiert die Nutzdaten, die keine Steuerinformationen beinhalten müssen. Da der Header nur eine Größe von 4 Bytes besitzt, können mehr Nutzdaten z.B. in Form von *RTP*-Datagrammen transportiert werden. In Abbildung 15 ist der Aufbau eines Headers skizziert:

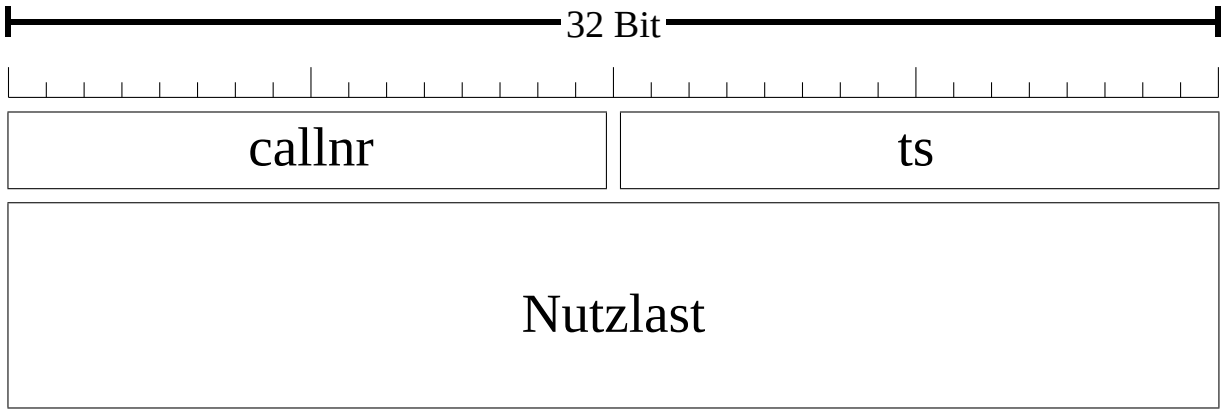

Abbildung 15: Mini-Header

Die Einträge sind analog zu Kapitel 7.2.1 ab Seite 60 zu betrachten, wobei der Zeitstempel auf 16 Bit beschränkt ist.

7.3 Verbindungsaufbau

Dieses Kapitel skizziert den Zeitablauf der Kommunikation über IAX. Das Rechnersymbol, dass in den Skizzen verwendet wird, charakterisiert ein Asterisk-System und das Telefon ein Endgerät. Als Endgeräte kommen Hardware-Telefone, wie z.B. das *Snom 100,*ein gewöhnliches analoges Telefon, dass über eine Zaptel-Karte betrieben wird oder ein Softphone, das in der Regel aus einem Rechner mit Mikrophon und Lautsprecher/Kopfhörer besteht, zum Einsatz.
Da in dem IAX-Protokoll mehrere Arten von Paketen existieren, die über das **type**-Feld bestimmt werden, werden die Verbindungen mit einem Vollheader mit folgenden Pfeilarten symbolisiert:

***type*-Bezeichnung**	***type*-Wert**	**Pfeilart**
IAX-Pakete	06	- - - - - - -
CONTROL-Pakete	04	· · · · · · ·
VOICE-Pakete	02	- · - · - · -
DTMF	01	- · · · · · -

Tabelle 6: Zuordnung Pfeil-*type*-Werte

Verbindungen, die über keinen Vollheader verfügen, werden mit einem Pfeil mit einer durchgezogenen Linie dargestellt. Als Beispiel sind die in dem RTP-Format kodierten Sprachpaketen zu nennen.
Die Pakete, die mit einem Vollheader eingeleitet werden, beinhalten in der Nutzlast die Informationen, die übertragen werden soll. Dabei werden den verwendeten Anweisungen mit **=** Werte zugeordnet. Falls mehrere Anweisungen übertragen werden sollen, werden diese mit **;** voneinander getrennt.

7.3.1 Registrierung

Bei einer Anmeldung an dem Asterisk-Server muss sich der Nutzer zunächst authentifizieren. Auch bei einer erfolgreichen Anmeldung wird in regelmäßigen Abständen die folgende Authentifikations-Abfrage wiederholt.

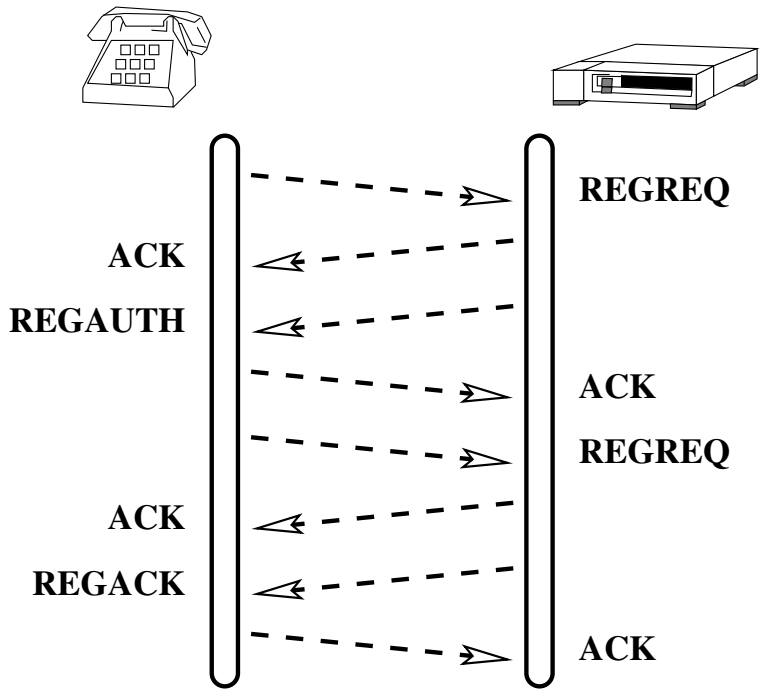

Abbildung 16: Zeitablauf einer Registrierung

ACK (*Type*: IAX)
Dieses Paket bestätigt jedes empfangende Paket mit einem Vollheader, indem es an den Sender geschickt wird. Somit erkennt der Sender des ursprünglichen Paketes, dass sein Paket bei dem Empfänger angekommen ist. Die Bestätigung erfolgt auf der Basis der Sequenznummer und nur andere *ACKs* und *HANGUPs* müssen nicht bestätigt werden.

Erstes REGREQ (*Type*: IAX)
Mit diesem Paket fordert der Klient eine Registrierung an. Als wichtige Nutzlast übermittelt der Klient den Nutzernamen, der auf dem Asterisk-Server eingetragen ist. Dies geschieht mit dem Nutzlasteintrag **peer=*nuzternam*e**.

REGAUTH (*Type*: IAX)
Nachdem der Asterisk-Server von dem *REGREQ*-Paket die Registierungsanfrage interpretiert hat, bestätigt er den eingetragenen Nutzernamen mittels **peer=*nutzername*** und überträgt die Authentifizierungsmethode, bei der als Beispiel **methods=plaintext** zu nennen ist, an den Klienten.

Zweites REGREQ (*Type*: IAX)
Nachdem der Klient nun über den Empfang des *REGAUTH*-Paketes vom Server die Authentifizierungsmethode erfahren hat, schickt er ein zweites *REGREQ*-Paket, dass im Unterschied zu dem vorigen um das Passwort, falls es sich bei der Authentifizierungsmethode um **plaintext** oder **md5** handelt, erweitert ist. Dabei wird das vorige *REGAUTH* mit dem Eintrag **secret=*passwort*** ergänzt.

REGACK (*Type*: IAX)
Nachdem die Authentifizierung überprüft und akzeptiert wurde, weißt der Server dem Klient einen freien UDP-Port zu. Anschließend bestätigt der Server die erfolgreiche Registrierung mit einem *REGACK*-Paket. Dieses beinhaltet wieder einen **peer**-Eintrag mit dem Nutzernamen. Außerdem übermittelt der Server die IP-Adresse des Clienten und die Portnummer, mit dem dieser mit dem Asterisk-Server kommunizieren kann. Dies geschieht mit den Einträgen **yourip=*ip.ad.dres.se*** und **yourport=*portnummer***.
Falls die Registrierung abgelehnt wurde, wird statt des *REGACK*-Paket ein *REGREJ* verschickt. Dieses beinhaltet in der Nutzlast einen String mit dem Grund der Ablehnung. Bei einem falschen Passwort kann der Eintrag **Registration Refused** lauten.

7.3.2 Sprachkommunikation mit einem Server

Da es eventuell Sinn machen kann, einfach nur Informationen (z.B. in Form von mp3s) abzuhören oder falls eine VoiceMail für einen Gesprächspartner hinterlegt werden soll, wird im folgendem ein Verbingungsaufbau zu einem Asterisk-Server dargestellt.

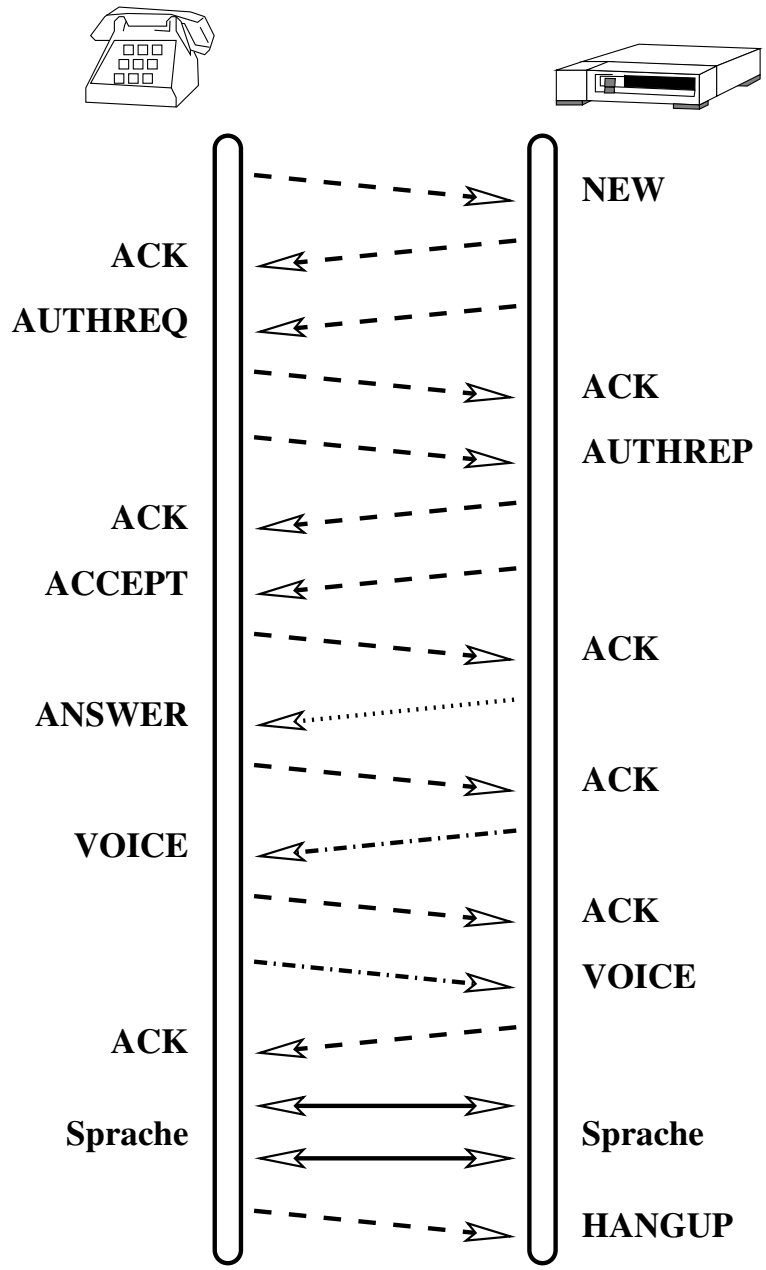

Abbildung 17: Kommunikation mit einem Server

ACK (*Type*: IAX)
Dieses Paket bestätigt jedes empfangende Paket mit einem Vollheader, indem es an den Sender geschickt wird. Somit erkennt der Sender des ursprünglichen Paketes, dass sein Paket bei dem Empfänger angekommen ist. Die Bestätigung erfolgt auf der Basis der Sequenznummer und nur andere *ACKs* und *HANGUPs* müssen nicht bestätigt werden.

NEW (*Type*: IAX)
Im Prinzip kann das *NEW*-Paket neben einer Verbindungsanfrage als Erinnerung an dem Server angesehen werden. Die Anfrage wird durch einen String in der Nutzlast des Paketes mit der gewählten Nummer dargestellt, der die Form **exten=*nummer*** besitzt. Mit diesem Eintrag kann der Asterisk-Server die passende Verbindung herraussuchen und später aufbauen.
Außerdem wird in diesem Paket u.a. noch der Name, der Kontext, die CallerID und die Sprache (falls dies vom Klient unterstützt wird) des registrierten Nutzers übertragen. Dies geschieht in der bekannten Form
callerid=*angezeigte_ID*;context=*context*;username=*nutzername*;language=;.
Das *language*-Feld wird hierbei nicht belegt, da dies nicht von dem verwendeten Softphone unterstützt wird.

AUTHREQ (*Type*: IAX)
Da Asterisk sich wirklich sicher sein muss, dass dieser Klient auch wirklich der ist, für den er sich ausgibt, leitet er mit diesem Paket eine neue Authentifizierung ein.
Die Nutzlast beinhaltet bei diesem Vorgang den Nutzernamen und die Authentifizierungsmethode des Clients, von dem Asterisk annimmt, dass dieser die Anfrage gestartet hat. Die Daten werden wie in den vorigen Paketen mit den Einträgen **methods=*auth_methode*** und **username=*username*** übertragen.

AUTHREP (*Type*: IAX)
Da im vorigen *AUTHREQ*-Paket (Die ACK's werden nicht berücksichtigt) der Asterisk-Server eine Authentifizierungsanfrage mit den Nutzernamen und Methode gestartet hat, muss der Client auf diese antworten und seine Identität beweisen. Falls die Methode *plaintext* oder *md5* verwendet wurde, kann dies mit der Übermittlung des Passwortes geschehen.
Dies geschieht mit dem Nutzlastinhalt **secret=*passwort***.

ACCEPT (*Type*: IAX)
Falls die Authentifizierung akzeptiert wurde, bestätigt dies der Asterisk-Server mit einem *ACCEPT*-Paket. Dieses Paket kann als ein besonderes *ACK*-Paket, das aber mit einem 'richtigen' *ACK* bestätigt werden muss, angesehen werden.

ANSWER (*Type*: CONTROL)
Dieses Paket übermittelt dem Client die Information, dass der Kommunikationspartner die Verbingunsanfrage angenommen hat. Dieses Paket besitzt als Nutzlast nur eine Reihe von *7Ehs*, die zum Auffüllen des Paketes verwendet werden.

VOICE (*Type*: VOICE)
Mit diesem Paket können die beiden Kommunikationspartner aushandeln, aus welcher Art die Datenübertragung besteht. Dieses Paket wird versendet, falls Sprachpakete versendet werden sollen.
Analog hierzu können auch *VIDEO (03)*-, *TEXT (07)*- oder *IMAGE (03)*-Pakete für die jeweiligen Übertragungen versendet werden.
Die Nutzung dieser Kommunikationsmöglichkeiten muss aber von dem verwendete Softphone unterstützt werden.

HANGUP (*Type*: IAX)
Dieses Paket dient dazu, dem Gesprächspartner ein „Auflegen des Telefonhörers“ zu übermitteln. Dieses Paket beinhaltet in der Nutzlast ein **Goodbye** und es muss nicht, aber kann mit einen *ACK* bestätigt werden.

7.3.3 Sprachkommunikation über einen Server

Den meisten Menschen finden es ziemlich uninteressant, wie in Kapitel 7.3.2 ab Seite 67 mit einer Maschine zu reden und sind der Meinung, dass eine Telefonanlage den Zweck erfüllen soll, mit einen anderen menschlichen Partner zu kommunizieren. Hierfür muss der Anrufer ebenfalls eine Verbindung zu Asterisk aufbauen, der wiederum den Anzurufenden kontaktiert.

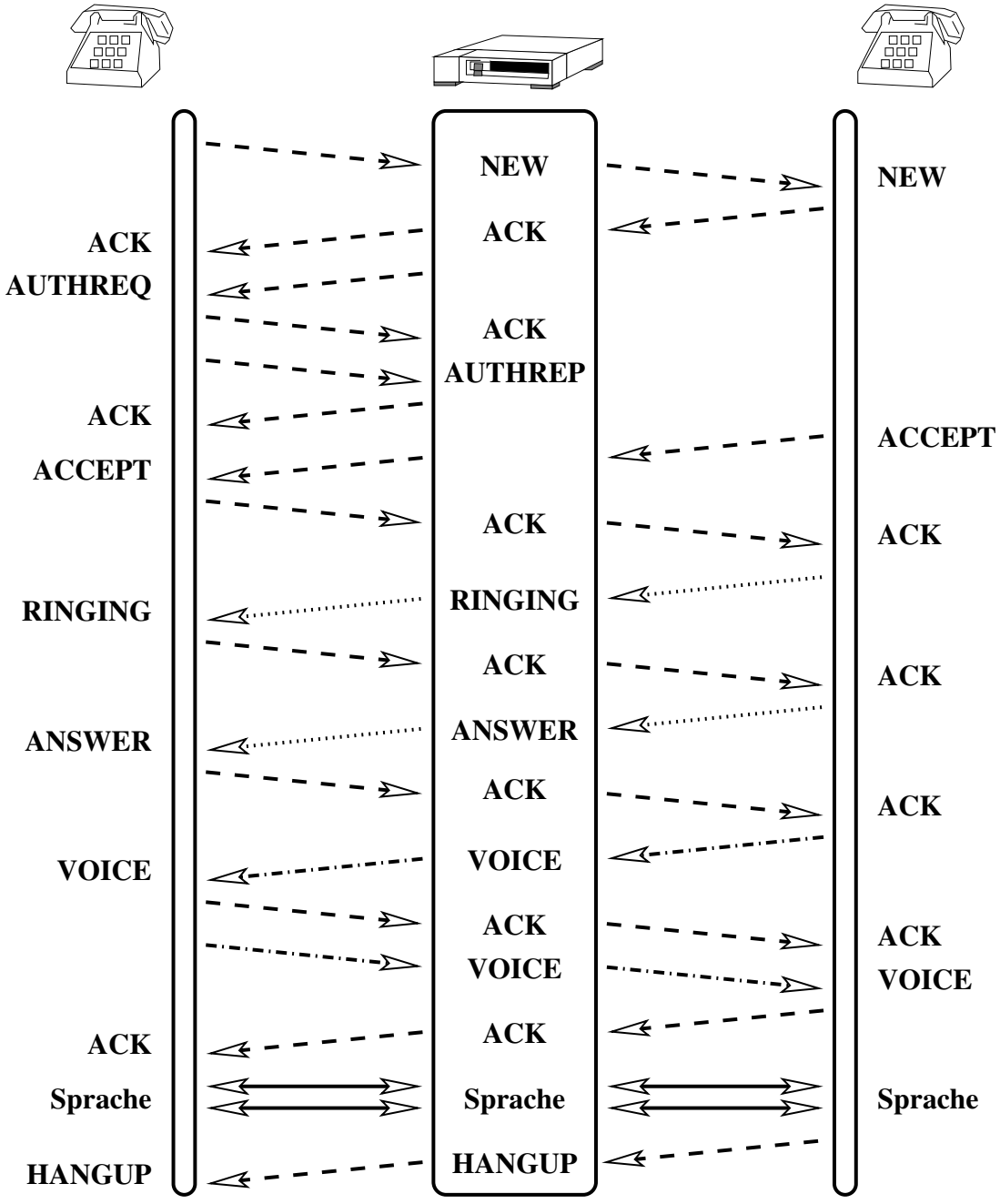

Abbildung 18: Gespräch zweier Kommunikationspartner

Der Inhalt der Pakete ist analog zu dem im Kapitel 7.3.2 auf Seite 67 zu betrachten, wobei der Asterisk-Server die Pakete nur weiterleitet. Der einzige Unterschied besteht darin, dass sich nur der Anrufer neu authentifizieren muss und dies von dem Anzurufenden nicht benötigt wird.

7.3.4 Statusabfragen während einer Datenübertragung mit dem Mini-Header

Wie in Kapitel 7.2 auf Seite 59 angekündigt, werden während einer Datenübertragung mit dem Miniheader in regelmäßigen Abständen kurze Dialoge mit Fullheadern geführt.

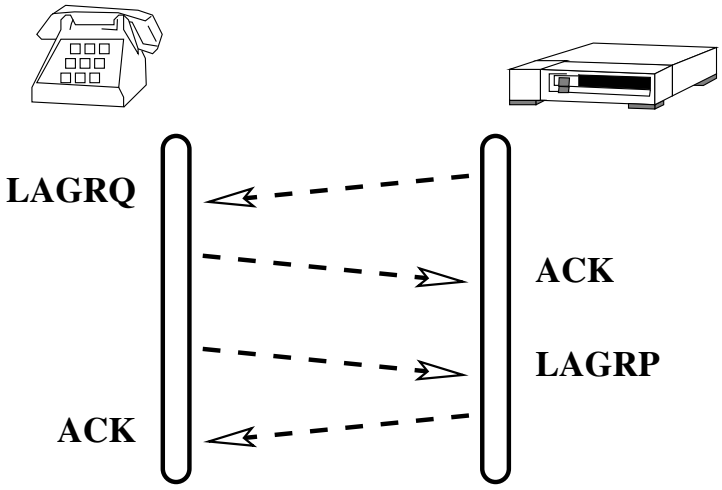

Abbildung 19: Statusabfragen

ACK (*Type*: IAX)
Dieses Paket bestätigt jedes empfangende Paket mit einem Vollheader, indem es an den Sender geschickt wird. Somit erkennt der Sender des ursprünglichen Paketes, dass sein Paket bei dem Empfänger angekommen ist. Die Bestätigung erfolgt auf der Basis der Sequenznummer und nur andere *ACKs* und *HANGUPs* müssen nicht bestätigt werden.

LAGRQ (*Type*: IAX)
Mit diesem Paket überprüft der Server, ob der Klient während z.B. eines Gespräches noch aktiv ist. Falls die Verbindung nicht unterbrochen wurde, wird der Klient diesen „Request“ mit einem „Response“ dass durch das Paket *LAGRP* symbolisiert wird, beantworten.
Die Nutzlast dieses Paketes dient nur dem Zweck des Platzfüllens.

LAGRP (*Type*: IAX)
Dieses Paket beantwortet das *LAGRQ* und dient dazu, dem Asterisk-Server die Aktivität des Klienten zu übermitteln.

7.3.5 Kommunikation mit einem VoiceMenu

Das folgende Szenario skizziert die Verbindung zu einen VoiceMenu, dessen Auswahlmöglichkeiten über DTMF-Tasteneingaben realisiert werden.

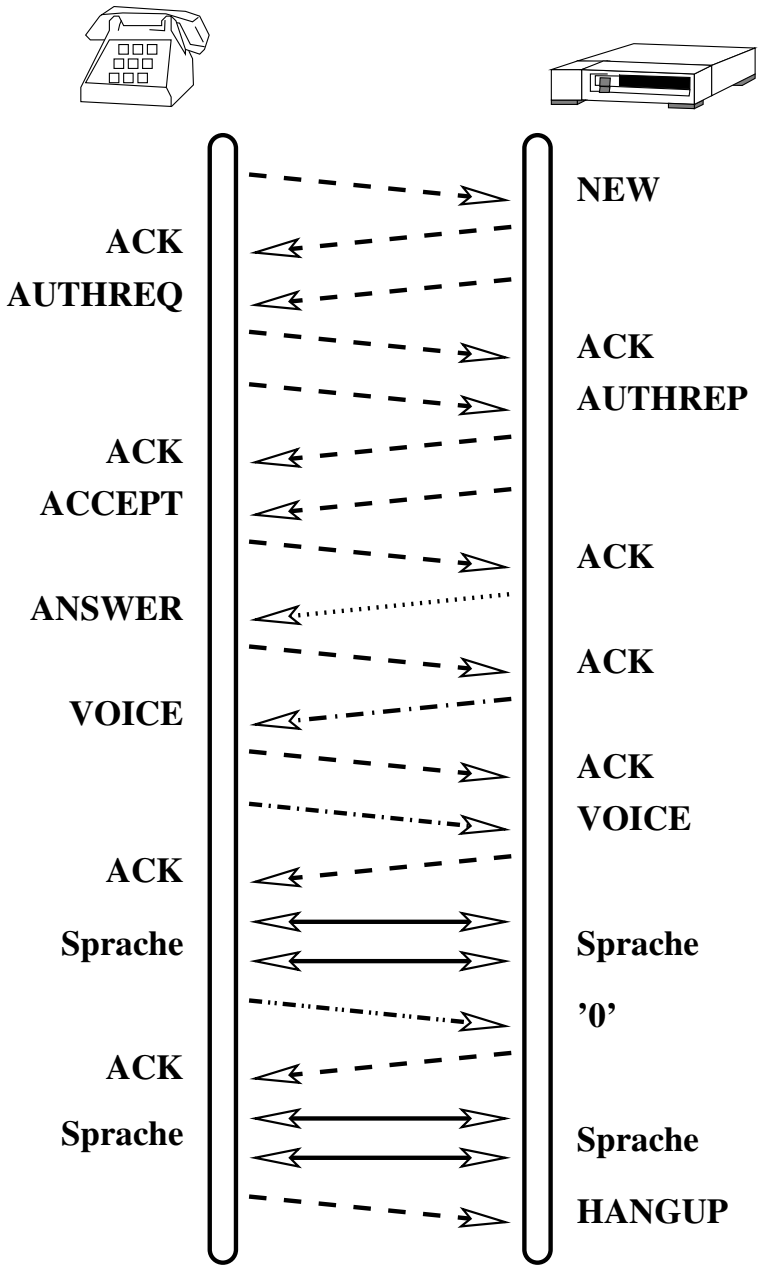

Abbildung 20: Kommunikation mit einem Voice-Menu mit DTMF-Eingabe

ACK (*Type*: IAX)
Dieses Paket bestätigt jedes empfangende Paket mit einem Vollheader, indem es an den Sender geschickt wird. Somit erkennt der Sender des ursprünglichen Paketes, dass sein Paket bei dem Empfänger angekommen ist. Die Bestätigung erfolgt auf der Basis der Sequenznummer und nur andere *ACKs* und *HANGUPs* müssen nicht bestätigt werden.

NEW (*Type*: IAX)
Im Prinzip kann das *NEW*-Paket neben einer Verbindungsanfrage als Erinnerung an dem Server angesehen werden. Die Anfrage wird durch einen String in der Nutzlast des Paketes mit der gewählten Nummer dargestellt, der die Form **exten=*nummer*** besitzt. Mit diesem Eintrag kann der Asterisk-Server die passende Verbindung herraussuchen und später aufbauen.
Außerdem wird in diesem Paket u.a. noch der Name, der Kontext, die CallerID und die Sprache (falls dies vom Klient unterstützt wird) des registrierten Nutzers übertragen. Dies geschieht in der bekannten Form
callerid=*angezeigte_ID*;context=*context*;username=*nutzername*;language=;.
Das *language*-Feld wird hierbei nicht belegt, da dies nicht von dem verwendeten Softphone unterstützt wird.

AUTHREQ (*Type*: IAX)
Da Asterisk sich wirklich sicher sein muss, dass dieser Klient auch wirklich der ist, für den er sich ausgibt, leitet er mit diesem Paket eine neue Authentifizierung ein.
Die Nutzlast beinhaltet bei diesem Vorgang den Nutzernamen und die Authentifizierungsmethode des Clients, von dem Asterisk annimmt, dass dieser die Anfrage gestartet hat. Die Daten werden wie in den vorigen Paketen mit den Einträgen **methods=*auth_methode*** und **username=*username*** übertragen.

AUTHREP (*Type*: IAX)
Da im vorigen *AUTHREQ*-Paket (Die ACK's werden nicht berücksichtigt) der Asterisk-Server eine Authentifizierungsanfrage mit den Nutzernamen und Methode gestartet hat, muss der Client auf diese antworten und seine Identität beweisen. Falls die Methode *plaintext* oder *md5* verwendet wurde, kann dies mit der Übermittlung des Passwortes geschehen.
Dies geschieht mit dem Nutzlastinhalt **secret=*passwort***.

ACCEPT (*Type*: IAX)
Falls die Authentifizierung akzeptiert wurde, bestätigt dies der Asterisk-Server mit einem *ACCEPT*-Paket. Dieses Paket kann als ein besonderes *ACK*-Paket, das aber mit einem 'richtigen' *ACK* bestätigt werden muss, angesehen werden.

ANSWER (*Type*: CONTROL)
Dieses Paket übermittelt dem Client die Information, dass der Kommunikationspartner die Verbingunsanfrage angenommen hat. Dieses Paket besitzt als Nutzlast nur eine Reihe von *7Ehs*, die zum Auffüllen des Paketes verwendet werden.

VOICE (*Type*: VOICE)
Mit diesem Paket können die beiden Kommunikationspartner aushandeln, aus welcher Art die Datenübertragung besteht. Dieses Paket wird versendet, falls Sprachpakete versendet werden sollen.
Analog hierzu können auch *VIDEO (03)*-, *TEXT (07)*- oder *IMAGE (03)*-Pakete für die jeweiligen Übertragungen versendet werden.
Die Nutzung dieser Kommunikationsmöglichkeiten muss aber von dem verwendete Softphone unterstützt werden.

DTMF-Eingabe *Type*: DTMF
Dieses Paket besitzt keine Nutzlast und beinhaltet als *csub*-Eintrag im Header den Wert der Eingabe. Dieser Wert, der in der Grafik als Beispiel mit dem Wert **'0'** belegt ist, wird in ASCII kodiert und in das dementsprechende (symbolische) Feld eingetragen.

HANGUP (*Type*: IAX)
Dieses Paket dient dazu, dem Gesprächspartner ein „Auflegen des Telefonhörers“ zu übermitteln. Dieses Paket beinhaltet in der Nutzlast ein **Goodbye** und es muss nicht, aber kann mit einen *ACK* bestätigt werden.

7.4 IAX hinter Masquerade-Gateways

Mittels IAX läßt sich eine Kommunikationsverbindung zwischen zwei Partnern in verschiedenen Netzen herstellen, die durch einem Masquerade-Gateway getrennt sind. Dies ist eindeutig eine Stärke von Asterisk, da diese Funktionalität von den weiter verbreiteten Signalisierungsprotokollen SIP und H.323 nicht geboten wird.
Aus diesem Grund stellt sich die Frage, warum dies den großen Protokollen, hinter denen sehr viele Entwickler stehen, so große Probleme bereitet und IAX diese Funktionalität ohne bekannte Einschränkungen bereitstellt.
Als erstes muss grob beleuchtet werden, wie z.B. SIP funktioniert: Es wird über einen definierten Port (in diesem Fall 5060) eine Verbindung aufgebaut. Ist dies geschehen, wird für den Medienstrom (z.B. RTP) ein weiterer Port ausgehandelt. Dieser Port ist nicht fest definiert und diese Aushandlung bereitet der Masqueradearchitektur in den meisten Fällen Probleme.
IAX umgeht dieses Problem, indem die gesamte Kommunikation nur über einen Kanal stattfindet. Sowohl für die Übertragung der Steuerbefehle als auch der Medienströme wird der selbe Port verwendet.
Im Gegensatz zu den meisten Signalisierungsprotokollen baut der IAX-Klient eine UDP-Verbindung, über die die gesamte Kommunikation stattfindet, auf und diese bleibt die ganze Zeit bestehen. Es wird nicht nach der Registierung diese Verbindung beendet und bei einem Anruf eine neue aufgebaut. Ein Aufbau einer direkten, neuen Verbindung von einem externen (ausserhalb des privaten Netzes) Endgerät zu einem Klienten in dem Privaten stellt für die Masqueradearchitekur eine große Hürde da.
Ein weiterer Grund, der Masquerade erleichtert, ist die Tatsache, dass in dem IAX-Datagramm keine IP-Adressen (z.B. *Source*- und *Destination Address*) eingetragen werden. Im Gegensatz zu SIP oder H.323 muss die Masqueradefunktionalität des Gateways diese Adressen im IAX-Header nicht verändern.

8 Unterstützung von SIP und H.323

Es existieren sehr viele standardisierte Protokolle, von denen sich hauptsächlich H.323, MGCP und SIP von der restlichen Masse abgesetzt haben. Dazu kommen noch viele Entwickler, die einfach eines der offenen Protokolle für Ihre Zwecke weiterentwickelt haben.
Es ist vielleicht ein sehr gutes Verkaufsargument z.B. des Netzwerkgeräteanbieters Cisco, das ihr erweitertes Protokoll den Andereren weit überlegen ist. Dies ist für die Festlegung eines Standards nicht sehr hilfreich.
Falls ein Hersteller zusätzlich noch eins oder mehrere der standardisieren VoIP-Protokolle unterstützt, dann nicht in vollen Umfang, um „Ihr erweitertes“ Protokoll weiter in den Vordergrund zu stellen. So macht es in den Augen der Kunden Sinn, die Geräte, die die standardisierten Protokolle nutzen, gegen die des Herstellers zu ersetzen.
Obwohl Asterisk das Protokoll IAX favorisiert, geht es einen anderen Weg: Weder den einschlägigen Quellen (z.B. Mailingliste) noch dem Autor dieser Studienarbeit, der SIP und IAX gleichwertig verwendet, ist ein geringerer Funktionsumfang der anderen freien VoIP-Protokoll bekannt.
Der einzige bekannte Nachteil ist die nicht direkte Integration von H.323, deren Gründe in Kapitel 2.2 beleuchtet wurden.
Der Unterschied zwischen einem praktischen Einsatz von SIP und IAX ist die Tatsache, dass die fast identischen Nutzereintragunen für IAX in die Konfigurationsdatei *iax.conf* und für SIP in die *sip.conf* eingetragen werden. Als weiterer Unterschied muss für das *Session Initiation Protocol* in der extensions.conf in der dementsprechenden Regel IAX durch das Schlüsselwort SIP ersetzt werden.
Das einzige Defizit ist die bessere Nutzerverwaltung der meisten SIP-Registrare, wobei aber der Fehler bei Asterisk zu suchen ist. Bis auf diesen Nachteil sind keine Probleme und Einschränkungen in einer heterogenen VoIP-Lösung bekannt.

9 Ein Anwendungsbeispiel

9.1 Aufgabenstellung

Die aus 31 Mitgliedern bestehende Organisation **Muppet-Show**[24] hat beschlossen, sich eine neue Telefonanlage anzuschaffen. Da die Organisation aber aus zwei räumlich getrennten Standorten besteht, die jeweils breitbandig ans Internet angeschlossen ist, bietet sich eine paketorientierte Kommunikationstechnologie an.
Aus diesem Grund soll mit zwei Asterisksservern (VoIP-I und VoIP-2) eine Telekommunikationsanlage aufgebaut werden, die durch Hard- und Softphones in beliebiger Mischung ergänzt werden können. Ein Teil der Mitglieder darf sich nur an den jeweiligen Server seines Standortes anmelden, wobei einige Nutzer, die des öfteren zwischen den Standorten pendeln, sich an beiden anmelden dürfen. Dabei sollen sie auf jedem Server immer mit der gleichen Nummer erreichbar sein.
Da einige Mitglieder der Organisation öfter zusammenarbeiten, sollen diese auch jeweils unter einer Gruppenrufnummer zu erreichen sein. Diese sind im Kapitel im Abschnitt *Sammelrufnummern* im Kapitel 9.1.1 aufgelistet.
Als Beispiel dafür sind die Schweine (Miss Piggy, Andy und Randy Pig) zu nennen, die in dieser Organisation im Bereitschaftsdienst arbeiten. Da während den Öffnungszeiten[3] viele Verbindungsanfragen an den Bereitschaftsdienst auftreten können, sollen die Anrufe gerecht an die Mitarbeiter verteilt werden. Dazu werden die Anrufe an den nächsten Mitarbeiter weitergeleitet, falls der aktuell Angerufene gerade telefoniert oder das Gespräch nicht annimmt. Der Mitarbeiter des Bereitschaftsdienstes, der die letzte Anfrage bearbeitet hat, soll in dieser Reihenfolge der letzte sein, der angerufen wird.
Bei einem Anruf außerhalb der Öffnungszeiten soll eine VoiceMail an den Administrator (shol) gesendet werden.
Da die Nutzer nicht nur an einem Arbeitsplatz arbeiten, soll auch gewährleistet werden, dass sich diese auch von verschiedenen Rechnern anmelden dürfen. Dabei muss auch die Nutzung eines Rechners, der sich hinter einer NAT-Firewall befindet, möglich sein.
Im weiteren soll ein kleines VoiceMenu auf dem Server VoIP-I und zwei Endgeräte, die nur Verbindungen aufbauen, aber nicht annehmen dürfen, realisiert werden.
In Abbildung 9.1 auf der nächsten Seite ist die Netztopologie der beiden Standorte grob skizziert. Da die Netzwerk- und IP-Adressen eine untergeordnete Rolle spielen, werden diese nicht weiter berücksichtigt.

[3] Mo-Do: 8:00 Uhr bis 16:00 Uhr, Fr: 8:00 Uhr bis 13:00 Uhr

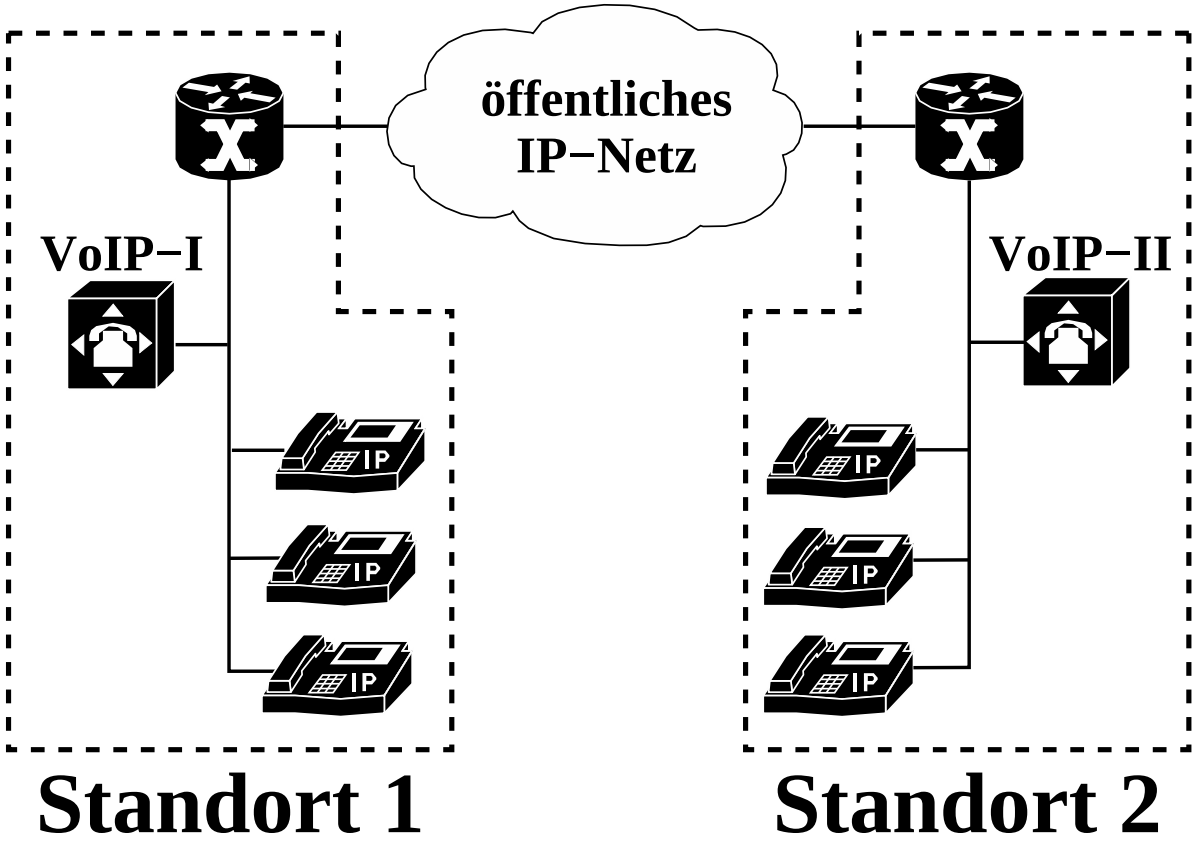

Abbildung 21: Netztopologie ohne Adressen

9.1.1 Rufnummernplan

Nutzerverzeichnis Im folgenden werden alle Nutzer aufgelistet, die eine eigene Rufnummer erhalten sollen:

Rufnummer	Name	Nutzername	VoIP-1	VoIP-II	IAX	SIP
101	Kermit the Frog	kermit	X		X	X
102	Fozzie Bear	fozzie	X		X	X
103	Rizzo the Rat	rizzo	X		X	
104	Rowlf	rowlf	X			X
105	Beaker	beak	X		X	
106	Robin the Frog	robin	X			X
107	The Swedish Chef	chef	X		X	
108	Floyd Pepper	floyd	X			X
109	Zoot	zoot	X		X	
110	Link Hogthrob	link	X			X
111	Clifford	cliff	X		X	
112	Bobo the Bear	bobo	X			X
113	Andy Pig	andy	X		X	
114	Randy Pig	randy	X		X	
115	Scooter	scoot		X		X
116	Miss Piggy	piggy		X	X	X
117	Gonzo	gonzo		X	X	X
118	Animal	ani		X	X	
119	Dr. Bunsen Honeydew	honey		X		X
120	Statler	stat		X	X	X
121	Waldorf	wald		X	X	X
122	Sam the Eagle	sam		X	X	
123	Dr. Teeth	teeth		X		X
124	Janice	jan		X	X	
125	Dr. Julius T. Strangepork	jst		X		X
126	Bean Bunny	bunny		X	X	
127	Johnny Fiamma	fiamma	X	X		X
128	Sal	sal	X	X	X	
129	Seymour	sey	X	X		X
130	Pepe	pepe	X	X	X	
131	Sweetums	sweet	X	X		X

Tabelle 7: Nutzerverzeichnis

Sammelrufnummern Da einige Nutzer im Team zusammenarbeiten und diese über eine Rufnummer erreicht werden sollen, ergibt sich folgende Rufnummernzuweisung:

Rufnummer	Nutzernamen	VoIP-I	VoIP-II
2	Bereitschaftsdienst	X	
201	Statler und Waldorf		X
202	Johnny Fiamma und Sal	X	X
203	Seymour und Pepe	X	X
204	Andy und Randy Pig	X	

Tabelle 8: Sammelrufnummern

Weitere Telefonnummern Einige der Rufnummern haben besondere Funktionen und sind nicht „normalen“ Nutzern zugeordnet.

Rufnummer	Name bzw. Bezeichnung	Nutzername	VoIP-I	VoIP-II	IAX	SIP
0	VoiceMenu		X			
1	VoiceMail an Admin		X			
11	Holger Schildt	shol	X	X	X	X
12	Dr. Ludwig Wolf	lwo	X	X	X	X
13	Snom IP-Telephone	snom	X			X
99	Telefonzelle 2	box2		X	X	

Tabelle 9: Rufnummernverzeichnis ohne Nutzer

Die Endgeräte, die keine Verbindungen annehmen dürfen, werden im folgenden *Telefonzelle* genannt. Dabei wird nur die Grundfunktionalität, Anrufe anzunehmen, betrachtet, obwohl herkömmliche Telefonzellen auch Gespräche annehmen können.
Der oben aufgelisteten Telefonzelle muss aber eine Telefonnummer zugeordnet werden, da von ihr nur das VoiceMenu angerufen werden darf und das Routing auf der CallerID mit dieser Nummer beruht.
Diese Nummer wird nur für interne Zwecke genutzt und die Telefonzelle ist von anderen Teilnehmern nicht anrufbar.

Nutzer ohne eigene Telefonnummern Nicht jeder Nutzer/jedes Objekt muss eine eigene Telefonnummer zugewiesen bekommen. Beispiele dafür sind Telefonzellen, die nicht angerufen werden dürfen und somit keine Telefonnummer benötigen (Ausnahme: siehe *Telefonzelle 2* im vorigem Unterpunkt).

Bezeichnung	Nutzername	Servername
„Telefonzelle 1“	box1	VoIP-I
Asteriskserver „VoIP1“	voipI	VoIP-II
Asteriskserver „VoIP2“	voipII	VoIP-I

Tabelle 10: Nutzer ohne zugeordneter Rufnummer

VoiceMail Da für jeden Nutzer, der VoiceMails empfangen darf, eine Mailadresse zugeordnet sein muss, dürfen aus verwaltungstechnischen Gründen nur folgende Nutzer dieses Asterisk-Feature nutzen:

Nutzername	EMail-Adresse
shol	shol@hrz.tu-chemnitz.de
lwo	lwo@hrz.tu-chemnitz.de

Tabelle 11: VoiceMail-Nutzer

Falls mit der Telefonnummer *1* (Einstellig, da leichter vom Benutzer zu merken) dem Administrator eine EMail hinterlassen werden soll, wird diese an *shol@hrz.tu-chemnitz.de* geschickt.

9.2 Lösung

9.2.1 iax.conf

Aufteilung der Benutzernamen auf den Servern VoIP-I und VoIP-II

Nutzername	***type***	**Kontext**	**Authentifizierungsmethode**
shol	friend	incoming	md5
lwo	friend	incoming	md5
box1	peer	incoming	md5
voip2	friend	local	plaintext
kermit	friend	incoming	md5
fozzie	friend	incoming	md5
rizzo	friend	incoming	md5
beak	friend	incoming	md5
chef	friend	incoming	md5
zoot	friend	incoming	md5
cliff	friend	incoming	md5
andy	friend	incoming	md5
randy	friend	incoming	md5
sal	friend	incoming	md5
pepe	friend	incoming	md5

Tabelle 12: IAX-Nutzer für den Server VoIP-I

Nutzername	***type***	**Kontext**	**Authentifizierungsmethode**
shol	friend	incoming	md5
lwo	friend	incoming	md5
box2	peer	incoming	md5
voip1	friend	local	plaintext
piggy	friend	incoming	md5
gonzo	friend	incoming	md5
ani	friend	incoming	md5
stat	friend	incoming	md5
wald	friend	incoming	md5
sam	friend	incoming	md5
jan	friend	incoming	md5
bunny	friend	incoming	md5
sal	friend	incoming	md5
pepe	friend	incoming	md5

Tabelle 13: IAX-Nutzer für den Server VoIP-II

Auszug aus der iax.conf des Asteriskservers VoIP-I

```
[general]
bandwidth=high
disallow=lpc10
tos=lowdelay

[shol]
type=friend
context=incoming
callerid="Holger Schildt" <11>
mailbox=11
auth=md5
secret=shol
host=dynamic

[lwo]
type=friend
context=incoming
callerid="Dr. Ludwig Wolf" <12>
auth=md5
secret=lwo
host=dynamic

[box1]
type=peer
context=incoming
auth=md5
secret=box1
host=dynamic

[voipII]
type=friend
context=local
auth=plaintext
secret=voipII
host=dynamic
defaultip=<ip.des.anderen.askeriskservers>

[kermit]
type=friend
context=incoming
callerid="Kermit the Frog" <101>
auth=md5
secret=kermit
host=dynamic

;Analog zu dem Nutzer 'kermit' muessen auch die Benutzer
;fozzie, rizzo, beak, chef, zoot, cliff, andy, randy, sal, pepe
;definiert werden
```

Abbildung 22: iax.conf des VoIP-I

Auszug aus der iax.conf des Asteriskservers VoIP-II

```
[general]
bandwidth=high
disallow=lpc10
amaflag=billing

[shol]
type=friend
callerid="Holger Schildt" <11>
context=incoming
auth=md5
secret=shol
host=dynamic

[lwo]
type=friend
context=incoming
callerid="Dr. Ludwig Wolf" <12>
auth=md5
secret=lwo
host=dynamic

[box2]
type=user
context=incoming
callerid="box2" <99>
auth=md5
secret=box2
host=dynamic

[voipI]
type=friend
context=local
auth=plaintext
secret=voipI
host=dynamic
defaultip=134.109.132.89

[piggy]
type=friend
context=incoming
callerid="Miss Piggy" <116>
auth=md5
secret=piggy
host=dynamic

;Analog zu dem Nutzer 'Piggy' muessen auch die Benutzer
;gonzo, ani, stat, wald, sam, jan, bunny, sal, pepe
;definiert werden
```

Abbildung 23: iax.conf des VoIP-II

9.2.2 sip.conf

Aufteilung der Benutzernamen auf den Servern VoIP-I und VoIP-II

Nutzername	***type***	**Kontext**
shol	friend	incoming
lwo	friend	incoming
snom	friend	incoming
box1	peer	incoming
kermit	friend	incoming
fozzie	friend	incoming
rowlf	friend	incoming
robin	friend	incoming
floyd	friend	incoming
link	friend	incoming
bobo	friend	incoming
fiamma	friend	incoming
sey	friend	incoming
sweet	friend	incoming

Tabelle 14: SIP-Nutzer für den Server VoIP-I

Nutzername	***type***	**Kontext**
shol	friend	incoming
lwo	friend	incoming
box2	peer	incoming
scoot	friend	incoming
piggy	friend	incoming
gonzo	friend	incoming
honey	friend	incoming
stat	friend	incoming
wald	friend	incoming
teeth	friend	incoming
jst	friend	incoming
fiamma	friend	incoming
sey	friend	incoming
sweet	friend	incoming

Tabelle 15: SIP-Nutzer für den Server VoIP-II

Auszug aus der sip.conf des Asteriskservers VoIP-I

```
[general]
port = 5060
bindaddr = 0.0.0.0
context = incoming
disallow = gsm
allow=ulaw
allow=alaw

[shol]
type=friend
secret=shol
host=dynamic
qualify=200
callerid="Holger Schildt" <11>

[lwo]
type=friend
callerid="Dr. Ludwig Wolf" <12>
secret=lwo
host=dynamic
qualify=200

[snom]
type=friend
callerid="Snom IP-Telefon" <13>
secret=snom
host=dynamic
qualify=200

[box1]
type=peer
secret=box1
host=dynamic
qualify=200

[kermit]
type=friend
callerid="Kermit The Frog" <101>
secret=kermit
host=dynamic
qualify=200

;Analog zu dem Nutzer 'kermit' muessen auch die Benutzer
;fozzie, rowlf, robin, floyd, link, bobo, fiamma, sey, sweet
;definiert werden
```

Abbildung 24: sip.conf des VoIP-I

Auszug aus der sip.conf des Asteriskservers VoIP-II

```
[general]
port = 5060
bindaddr = 0.0.0.0
disallow = gsm
allow=ulaw,alaw
context = incoming

[shol]
type=friend
secret=shol
host=dynamic
qualify=200
callerid="Holger Schildt" <11>

[lwo]
type=friend
secret=lwo
host=dynamic
qualify=200
callerid="Dr. Ludwig Wolf" <12>

[box2]
type=peer
secret=box2
host=dynamic
qualify=200

[piggy]
type=friend
secret=piggy
host=dynamic
qualify=200
callerid="Miss Piggy" <116>

;Analog zu dem Nutzer 'Piggy' muessen auch die Benutzer
;scoot, gonzo, honey, stat, wald, teeth, jst, fiamma, sey, sweet
;definiert werden
```

Abbildung 25: sip.conf des VoIP-II

9.2.3 VoiceMenu

Abbildung 26 skizziert die Ansagen, die abgespielt und die möglichen Aktionen die in dem VoiceMenu ausgeführt werden. Das VoiceMenu wird als Kontext in Abbildung 27 definiert und muss in der *extensions.conf* des Asteriskservers VoIP-I eingefügt werden (siehe Kapitel 9.2.5). Da dieses VoiceMenu nur die Fähigkeiten des Asterisk demonstrieren soll, ist es im Umfang sehr beschränkt.

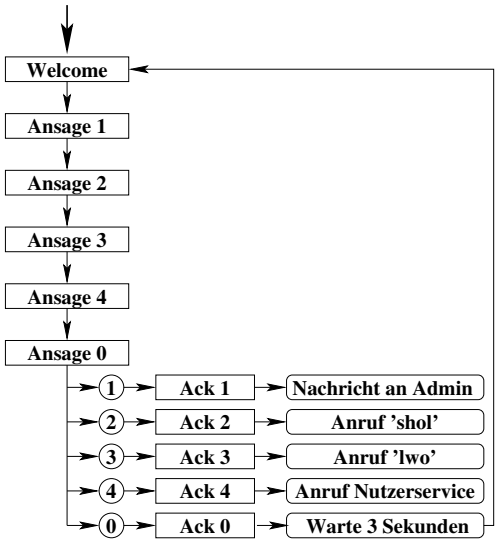

Abbildung 26: Skizzierung VoiceMenu auf VoIP-I

In Tabelle 16 werden den Ansagen aus Abbildung 26 Dateinamen zugeordnet. Im weiteren wird der Text, mit dem die Dateien aufgenommen werden, aufgeführt.

Ansage	Dateiname	Inhalt
Welcome	welcome.gsm	Willkommen auf dem VoiceMenu des Asteriskservers VoIP-I
Ansage 1	ansage1.gsm	Drücken Sie die '1', um eine Sprachnachricht dem Administrator zu hinterlassen
Ansage 2	ansage2.gsm	Drücken Sie die '2', um mit dem Nutzer *shol* verbunden zu werden
Ansage 3	ansage3.gsm	Drücken Sie die '3', um mit dem Nutzer *lwo* verbunden zu werden
Ansage 4	ansage4.gsm	Drücken Sie die '4', um mit dem Bereitschaftsdienst verbunden zu werden
Ansage 0	ansage0.gsm	Drücken Sie die '0', um das VoiceMenu nochmal zu hören
Ack 1	ack1.gsm	Sie haben die Taste '1' gedrückt
Ack 2	ack2.gsm	Sie haben die Taste '2' gedrückt
Ack 3	ack3.gsm	Sie haben die Taste '3' gedrückt
Ack 4	ack4.gsm	Sie haben die Taste '4' gedrückt
Ack 0	ack0.gsm	Sie haben die Taste '0' gedrückt
Invalid	invalid.gsm	Ihr Eingabe ist ungültig

Tabelle 16: Dateinamen für das VoiceMenu

```
[mainmenu]
exten => s,1,BackGround(welcome)
exten => s,2,BackGround(ansage1)
exten => s,3,BackGround(ansage2)
exten => s,4,BackGround(ansage3)
exten => s,5,BackGround(ansage4)
exten => s,6,BackGround(ansage0)


exten => 1,1,BackGround(ack1)
exten => 1,2,VoiceMail,1
exten => 1,3,Hangup

exten => 2,1,BackGround(ack2)
exten => 2,2,Dial(IAX/shol)
exten => 2,3,Dial(SIP/shol)
exten => 2,4,Dial(IAX2/${REMOTE}/11)
exten => 2,5,Hangup

exten => 3,1,BackGround(ack3)
exten => 3,2,Dial(IAX/lwo)
exten => 3,3,Dial(SIP/lwo)
exten => 3,4,Dial(IAX2/${REMOTE}/12)
exten => 3,5,Hangup

exten => 4,1,BackGround(ack1)
exten => 4,2,Dial(IAX2/${REMOTE}/3)
exten => 4,3,Hangup

exten => 0,1,BackGround(ack0)
exten => 0,2,Goto(mainmenu,s,1)

exten => i,1,Playback,invalid
```

Abbildung 27: Kontext für VoiceMenu des Asteriskservers VoIP-I

9.2.4 Phone Hunting Group

Wie in der Aufgabenstellung gefordert, sollen die ankommende Anrufe gerecht zwischen den Mitgliedern der Nutzerverwaltung verteilt werden. Bei dieser Liste mit den drei Mitgliedern Miss Piggy, Randy und Andy Pig soll erreicht werden, dass der Mitarbeiter, der den letzten Anruf angenommen hat, als Letzter einen neuen Anruf zugestellt bekommt.
Für diese Funktionalität, die der Netzwerkgerätehersteller Cisco als *ephone hunting group* bezeichnet, gibt es in dem PBX-System Asterisk keine fertige Implementierung. Nach einer Anfrage auf der Asteriskmailingliste wurde vorgeschlagen, dass die Datenbankfunktionalität genutzt werden kann, um dieses Ziel zu erfüllen. Die folgende Lösung dieses Problems setzt genau diesen Ansatz um.
Sicherlich wäre eine Implementation eines Moduls hierfür die elegantere Lösung, aber weniger geeignet, die mitgelieferte Asteriskfunktionalität auszuprobieren und zu verstehen.
Ein großer Nachteil der Lösung ist der Umfang, den sie bei einer hohen Mitarbeiteranzahl annehmen kann. Dies hat zwar keine Nachteile in der Performance, macht aber die Lösung unübersichtlicher.
Da die Extensionregeln in Abbildung 28 zu einem Kontext zusammengefasst sind, lässt sich diese ohne Probleme in der *extensions.conf* des VoIP-1 Asterisksserver im Kapitel 9.2.5 einbinden.
Als erster Schritt wird die Anfangsreihenfolge definiert, indem den Datenbankeinträgen Im *NS/1st*, *NS/2nd* und *NS/3rd* die Nutzernamen für den ersten, zweiten und dritten Mitarbeiter des Bereitschaftsdientes zugewiesen wird. In diesem Fall sind das die Nutzer *andy*, *randy* und *piggy*. Miss Piggy registriert sich nicht direkt an dem Asteriskserver VoIP-I, sondern auf dem Server, der in der *extensions.conf* mit der Variable *${remote}* symbolisiert wird.
Hier werden die Werte initialisiert, wenn der Nutzer mit der CallerID 11 (in diesem Fall *shol*) die Telefonnummer 99 anruft.

```
[bereitschaftsdienst]
;werte initialisieren
exten => 99/11,1,DBdeltree(NS)
exten => 99/11,2,DBput(NS/1st=IAX/andy)
exten => 99/11,3,DBput(NS/2nd=IAX/randy)
exten => 99/11,4,DBput(NS/3rd=IAX2/${REMOTE}/116)
exten => 99/11,5,Hangup

exten => 2,1,DBget(erster=NS/1st)
exten => 2,2,DBget(zweiter=NS/2nd)
exten => 2,3,DBget(dritter=NS/3rd)

exten => 2,4,DBdeltree(NS)
exten => 2,5,DBput(NS/1st=${zweiter})
exten => 2,6,DBput(NS/2nd=${dritter})
exten => 2,7,DBput(NS/3rd=${erster})
exten => 2,8,Dial(${erster})

exten => 2,9,DBdeltree(NS)
exten => 2,10,DBput(NS/1st=${dritter})
exten => 2,11,DBput(NS/2nd=${erster})
exten => 2,12,DBput(NS/3rd=${zweiter})
exten => 2,13,Dial(${zweiter})

exten => 2,14,DBdeltree(NS)
exten => 2,15,DBput(NS/1st=${erster})
exten => 2,16,DBput(NS/2nd=${zweiter})
exten => 2,17,DBput(NS/3rd=${dritter})
exten => 2,18,Dial(${dritter})
```

Abbildung 28: Kontext aus extensions.conf für Phone Hunting Group (VoIP-I)

Analog kann dies auch mit den folgenden Befehlen unter dem CLI geschehen:

```
voip*CLI> database deltree NS
Database entries removed.
voip*CLI> database put NS 1st IAX/andy
Updated database successfully
voip*CLI> database put NS 2nd IAX/randy
Updated database successfully
voip*CLI> database put NS 3rd IAX2/${REMOTE}/116
Updated database successfully
```

Abbildung 29: Initialisieren der Phone Hunting Group-Werte über das CLI

Die initialisierte Datenbank bleibt auch nach einem Beenden des Asterisk-Prozesses bestehen und somit muss die Initialisierung nur einmal durchgeführt werden.

Nun beginnt die eigentliche Verwaltung des Bereitschaftsdienstes. Im Prinzip werden die drei Variablen, die bestimmen, welcher der Mitarbeiter als erstes, als zweites und als drittes angerufen wird, mit den passenden Werten belegt. Dafür werden die Werte mit dem Befehl *DBput* aus der Datenbank geholt.
Nun wird die Datenbank gelöscht, damit sie mit neuen Werten belegt werden kann. Der nächste Mitarbeiter, der angerufen werden soll, wird auf der letzten (*dritter*) Stelle gesetzt. Falls er das Gespräch annimmt, wird er bei dem nächsten Anruf erst angewählt, falls die anderen beiden Mitarbeiter nicht das Gespräch annehmen können. Falls er nicht erreichbar ist, wird die Datenbank wieder überschrieben und mit dem nächsten Mitarbeiter werden die Schritte analog wiederholt.

9.2.5 extensions.conf

Auszug der extension.conf des Asteriskservers VoIP-I

```
[general]
static=yes
writeprotect=no

[globals]
REMOTE => voipI:voipI@<netzwerknamen.des.voipII>
HUNT=IAX/kermit,IAX/fozzie

[mainmenu]
;siehe Unterkapitel 'VoiceMenu'

[bereitschaftsdient]
;siehe Unterkapitel 'phone hunting group'

[adminmail]
exten => s,1,VoiceMail,1

[local]
include => bereitschaftsdienst|8:00-16:00|Mon-Thu
include => bereitschaftsdienst|8:00-13:00|Fri
include => adminmail

exten =>   0,1,Goto(mainmenu,s,1)
exten =>   1,1,VoiceMail,1

;Nutzer mit VoiceMail
exten =>  11,1,Dial(IAX/shol)
exten =>  11,2,Dial(SIP/shol)
exten =>  11,3,Dial(IAX2/${REMOTE}/11)
exten =>  11,4,VoiceMail(u11)
exten =>  11,102,VoiceMail(b11)

;Nutzer mit VoiceMail
exten =>  12,1,Dial(IAX/lwo)
exten =>  12,2,Dial(SIP/lw0)
exten =>  12,3,Dial(IAX2/${REMOTE}/12)
exten =>  12,4,VoiceMail,u12
exten =>  12,102,VoiceMail,b12

;'normale' Benutzer
exten =>  13,1,Dial(SIP/snom)
exten => 101,1,Dial(IAX/kermit&SIP/kermit)
exten => 102,1,Dial(IAX/fozzie&SIP/fozzie)
exten => 103,1,Dial(IAX/rizzo)
exten => 104,1,Dial(SIP/rowlf)
exten => 105,1,Dial(IAX/beak)
```

Abbildung 30: extensions.conf des VoIP-I, Teil 1

```
exten => 106,1,Dial(SIP/robin)
exten => 107,1,Dial(IAX/chef)
exten => 108,1,Dial(SIP/floyd)
exten => 109,1,Dial(IAX/zoot)
exten => 110,1,Dial(SIP/link)
exten => 111,1,Dial(IAX/cliff)
exten => 112,1,Dial(SIP/bobo)
exten => 113,1,Dial(IAX/andy)
exten => 114,1,Dial(IAX/randy)

;diese Nutzer koennen sich auf beiden Asteriskservern anmelden
exten => 127,1,Dial(SIP/fiamma)
exten => 127,2,Dial(IAX2/${REMOTE}/127)
exten => 128,1,Dial(IAX/sal)
exten => 128,2,Dial(IAX2/${REMOTE}/128)
exten => 129,1,Dial(SIP/sey)
exten => 129,2,Dial(IAX2/${REMOTE}/129)
exten => 130,1,Dial(IAX/pepe)
exten => 130,2,Dial(IAX2/${REMOTE}/130)
exten => 131,1,Dial(SIP/sweet)
exten => 131,2,Dial(IAX/${REMOTE}/131)

;weitere Sammelrufnummern
exten => 202,1,Dial(SIP/fiamma&IAX/sal)
exten => 202,2,Dial(IAX2${REMOTE}/127&IAX2/${REMOTE}/128)
exten => 203,1,Dial(SIP/sey&IAX/pepe)
exten => 203,2,Dial(IAX2/${REMOTE}/129&IAX2/${REMOTE}/130)
exten => 204,1,Dial(IAX/andy&IAX/randy)

[incoming]
include => local
switch => IAX2/${REMOTE}/local
```

Abbildung 31: extensions.conf des VoIP-I, Teil 2

Auszug der extension.conf des Asteriskservers VoIP-II

```
[general]
static=yes
writeprotect=no

[globals]
REMOTE => voipII:voipII@voip.hrz.tu-chemnitz.de

[local]
exten =>  11,1,Dial(IAX/shol)
exten =>  11,2,Dial(SIP/shol)
exten =>  11,3,Dial(IAX2/${REMOTE}/11)
```

Abbildung 32: extensions.conf des VoIP-II, Teil 1

```
exten =>  12,1,Dial(IAX/lwo)
exten =>  12,2,Dial(SIP/lw0)
exten =>  12,3,Dial(IAX2/${REMOTE}/12)

exten => 115,1,Dial(SIP/scoot)
exten => 116,1,Dial(IAX/piggy&SIP/piggy)
exten => 117,1,Dial(IAX/gonzo&SIP/gonzo)
exten => 118,1,Dial(IAX/ani)
exten => 119,1,Dial(SIP/honey)
exten => 120,1,Dial(IAX/stat&SIP/stat)
exten => 121,1,Dial(IAX/wald&SIP/wald)
exten => 122,1,Dial(IAX/sam)
exten => 123,1,Dial(SIP/teeth)
exten => 124,1,Dial(IAX/jan)
exten => 125,1,Dial(SIP/jst)
exten => 126,1,Dial(IAX/bunny)

;diese nutzer sind auf beiden maschinen zuhause
exten => 127,1,Dial(SIP/fiamma)
exten => 127,2,Dial(IAX2/${REMOTE}/127)
exten => 128,1,Dial(IAX/sal)
exten => 128,2,Dial(IAX2/${REMOTE}/128)
exten => 129,1,Dial(SIP/sey)
exten => 129,2,Dial(IAX2/${REMOTE}/129)
exten => 130,1,Dial(IAX/pepe)
exten => 130,2,Dial(IAX2/${REMOTE}/130)
exten => 131,1,Dial(SIP/sweet)
exten => 131,2,Dial(IAX/${REMOTE}/131)

;gruppen
exten => 201,1,Dial(IAX/wald&SIP/wald&IAX/stat&IAX/stat)
exten => 202,1,Dial(SIP/fiamma&IAX/sal)
exten => 202,2,Dial(IAX2${REMOTE}/127&IAX2/${REMOTE}/128)
exten => 203,1,Dial(SIP/sey&IAX/pepe)
exten => 203,2,Dial(IAX2/${REMOTE}/129&IAX2/${REMOTE}/130)

[incoming]
switch => IAX2/voipII:voipII@voip.hrz.tu-chemnitz.de/local
include => local

exten => _./99,1,Dial(IAX2/${REMOTE}/0)

exten => t,1,answer
exten => t,2,Playback,na-low
exten => t,3,Hangup
```

Abbildung 33: extensions.conf des VoIP-II, Teil 2

9.2.6 voicemail.conf

Zuweisung der EMailadressen an Nutzerkonnten

Auf Grunde des geringeren Administrationsaufwandes ist nur der Asteriskserver **VoIP-I** in der Lage, EMails an den Nutzern zu schicken. Dies ergibt folgende Zuweisungen:

Nutzername	Mailbox-Nummer	EMailadresse
shol (als Admin)	1	shol@hrz.tu-chemnitz.de
shol	11	shol@hrz.tu-chemnitz.de
lwo	12	lwo@hrz.tu-chemnitz,de

Tabelle 17: VoiceMail-Einstellungen

voicemail.conf des Asteriskservers VoIP-I

```
[general]
format=wav
serveremail=asterisk
attach=yes
fromstring=Asterisk PBX

[default]
 1 => shol,Administrator ,shol@hrz.tu-chemnitz.de
11 => shol,Holger Schildt,shol@hrz.tu-chemnitz.de
12 =>  lwo,Dr Ludwig Wolf,lwo@hrz.tu-chemnitz.de
```

Abbildung 34: voicemail.conf des VoIP-I

10 Fazit

10.1 Asterisk als PBX-System

Wie schon in dem Kapitel 1.2.6 genannt, sind die Einsatzbereiche für ein Asterisk-System fast unbegrenzt. Für Asterisk spricht u.a. die Offenheit bei der Auswahl des Signalisierungsprotokolls. Es stehen neben dem systemeigenen Protokoll IAX die standardisierten Protokolle SIP, H.323 und MGCP zur Verfügung, wobei die H.323-Unterstützung explizit eingebunden werden muss. Es werden die Funktionalitäten der offenen VoIP-Protokolle, die für eine reibungslose Nutzung benötigt werden, von Asterisk bereitgestellt.
Wo bei anderen Serverimplementierungen mehrere unabhängige Dienste betrieben werden müssen, die für die Registrierung, Anrufverarbeitung, Protokollumsetzung und die weiteren kleineren Spielereien zuständig sind, besteht Asterisk nur aus einem ausführbarem Programm. Dies macht die Konfiguration, die nur in dieses laufende Programm übernommen werden muss, sehr einfach.
Diese Konfiguration geht, nachdem die Grundprinzipien verstanden wurden, sehr einfach von statten. Die Syntax ist in allen Dateien identisch und die Festlegung, welches Protokoll ein eingetragener Nutzer verwenden soll, ist nur davon abhängig, in welcher Datei dieser eingetragen wird.
Außerdem lässt sich Asterisk sehr gut erweitern, indem andere Server mit genutzt werden können. Falls die weiteren VoIP-Server aus Asteriskmaschinen bestehen, können die auf dem lokalen Asteriskserver unbekannten Telefonnummern einfach auf den entfernten nachgeschlagen und angerufen werden. Bei einer eindeutigen Vergabe der Telefonnummern kann mit einem sehr geringen Konfigurationsaufwand ein großes Netz aus Asteriskservern betrieben werden.
Natürlich kann als Ziel eines Anrufs nicht nur ein dem Asteriskserver bekannter Nutzer angerufen werden, sondern auch z.B. SIP-Adressen. Dies ermöglicht eine Kooperation mit anderen VoIP-Architekturen.
Als ein sehr großer Nachteil ist die noch nicht ausgereifte Nutzerverwaltung anzusehen. Die Angaben zu den Nutzern müssen fest in den Konfigurationsdateien eingetragen werden. Dies verhindert eine eigenständige Änderung von z.B. Passwörtern oder eine Neuregistrierung der Nutzer. Eine provisorische Abhilfe könnte z.B. ein CGI-basiertes Script sein, dass die Konfigurationsdateien verändert und dann die Änderungen an Asterisk übergibt. Aber eine elegantere, direkt in Asterisk implementierte Lösung wäre sehr wünschenswert.
Bis auf dieses Defizit ist Asterisk aus dem oben genannten Gründen eine sehr gute Alternative zu den anderen verfügbaren, teilweise kommerziellen Systemen. Da auch sehr viele Soft- und Hardphones die unterstützten Protokolle verwenden, steht einem Einsatz des PBX-Systems Asterisk nichts mehr im Weg.

10.2 IAX in Verbindung mit Asterisk

Das VoIP-Protokoll IAX ist ein Protokoll, das speziell auf die Fähigkeiten von Asterisk angepasst wurde. Die Vorteile liegen in dem geringen Umfang des Paketheaders, der bei einer Sprach- oder Videokommunikation nur vier Bytes groß ist.
Mit Hilfe von IAX kann ein Kommunikationskanal zwischen zwei Endgeräten, die durch ein NAT-Gateway getrennt sind, aufgebaut werden. Außerdem stehen mehrere Authentifikationsmechanismen zur Verfügung. Weitere Vorteile sind im Kapitel 7 auf Seite 58 aufgelistet.
Leider sind die bisher verfügbaren Softphones noch nicht genügend ausgereift. Daher ist mit ihnen ein Einsatz von IAX noch nicht in vollem Umfang möglich.
Zwar laufen die Hardwaretelefone der Firma *Snom* mit einer IAX-Unterstützung stabil, aber weitere zuverlässige Hardphones sind nicht bekannt.
Aus diesem Grund ist, solange keine stabilen Softphones existieren, von einem Einsatz von IAX in Verbindung mit Endgeräten abzuraten.
Ein Grund dieses Problems ist in der geringen Verbreitung von IAX zu suchen, da nur die wenigstens Entwickler Bedarf an einer Implementation von IAX-Softphones sehen.
Da aber IAX an Asterisk angepasst ist, sollte dieses Protokoll für eine Kommunikation zwischen mehreren Asteriskservern verwendet werden.
Auch der Einsatz eines lokalen Asteriskservers auf dem Klientenrechner innerhalb eines privaten Netzwerkes wäre denkbar. Hier könnte sich der Nutzer über einen beliebigen VoIP-Protokoll mit einem beliebigen Softphone eine Verbindung zu dem lokalen Asteriskserver aufbauen, der dann den Nutzer an einen zentralen Asteriskserver anmeldet. Mit diesem Ansatz könnte man das Problem umgehen, dass bei den anderen standardisierten Protokollen mit NAT-Gateways besteht.

Abbildungsverzeichnis

Tabellenverzeichnis

Syntaxübersicht

Literatur

[1] VocalTec
(http://www.vocaltec.com/)

[2] Gartner Group (http://www.gartnergroup.com)
Die Information über die Umfrage stammt aus der Zeitschrift c't, Heise Verlag, Ausgabe 16/01

[3] Digium - A Linux Telephony Company
(http://www.digium.com)

[4] GNU General Public License
(http://www.gnu.org/copyleft/gpl.html)

[5] Asterisk Handbook - second draft
(http://www.digium.com/handbook-draft.pdf)

[6] Asterisk Mailing List
(http://www.asteriskpbx.com/index.php?menu=support#mailing_list)

[7] ISDN4Linux
(http://www.isdn4linux.de)

[8] OSS
(http://www.opensound.com)

[9] ALSA: Advanced Linux Sound Architecture
(http://www.alsa-project.org)

[10] QuickNet: PhoneJack
(http://www.quicknet.net/products/ipj.htm)

[11] QuickNet: LineJack
(http://www.quicknet.net/products/ilj.htm)

[12] Intel Dialogic Hardware
(http://www.intel.com/network/csp/Trans/dialogic.htm?iid=sr+dialogic\&)

[13] Asterisk The Open Source Linux PBX
http://www.asteriskpbx.com

[14] The OpenH323 project
http://www.openh323.org

[15] The GNU General Public License (GPL)
http://www.opensource.org/licenses/gpl-license.html

[16] Mozilla Public License (MPL)
http://www.mozilla.org/MPL/MPL-1.0.html

[17] H.323 support for ASTERISK PBX using the OpenH323 library (asterisk-oh323)
http://www.inaccessnetworks.com/projects/asterisk-oh323

[18] GnoPhone Homepage
http://www.gnophone.com/

[19] Reward offered for Windows IAX Client
http://phone.nq.net/reward.html

[20] tclPhone / tkPhone
http://sourceforge.net/projects/tel

[21] Wirlab kphone
http://www.wirlab.net/kphone

[22] snom technology AG
http://www.snom.com

[23] IAX-Unterstützung für das snom-Telefon
ftp://ftp.asterisk.org/pub/snom/snomphone_boot_instructions.txt

[24] Muppet Profiles
http://www.muppets.com/profiles/profiles.htm

Abkürzungsverzeichnis

CFB call forwarding busy

CFNR call forwarding no reply

CLI Command Line Interface

DNIS Dialed Number Identification Service

DTMF Dual Ton Multi Frequency

iLBC Internet Low Bandwith Codec - IETF draft (13,9kbit/s)

IVR Interactive Voice Response

LPC10 Linear Predictive Coding

PBX Private Branch Exchange

POTS Plain Old Telephone Service

PRI Primary Rate Interface

PSTN Public Switched Telephone Network - das öffentliche Telefonnetz

RTP Real-Time Transport Protocol

VoIP Voice over Internet Protocol