Studienarbeit

VoIP Telefon-Gateway

Albert-Ludwigs-Universität Freiburg
Fakultät für Angewandte Wissenschaften
Lehrstuhl für Kommunikationssysteme
Prof. Dr. G. Schneider

Timo Zauner, Matrikel Nr.: 1129564
zauner@informatik.uni-freiburg.de

19. Januar 2005

Inhaltsverzeichnis

Kapitel 1

Einleitung

Mit der Verbreitung und stark steigenden Leistung von IP-Netzwerken bietet es sich an, auch klassische Kommunikationswege, wie Telefonie über diese zu realisieren. So kann beispielsweise aus einem festen ISDN-Anschluss, der durch einen Telefon-Gateway mit einem LAN und WLAN verbunden wird, eine Telefonanlage mit festen und mobilen Endgeräten eingerichtet werden. Dies ist besonders interessant für Organisationen und Firmen mit einer schon vorhandenen Netzwerk-Infrastruktur. In dieser Studienarbeit möchte ich die vorhandenen Protokolle und Ansätze zur Implementierung einer solchen Telefonanlage besprechen. Dabei stellt sich zuerst die Frage, welches VoIP-Protokoll für die Kommunikation benutzt werden soll. Die beiden populärsten Protokolle sind H.323 und SIP, die ich beide im zweiten Kapitel kurz erläutern möchte. Für die Implementierung des Telefon-Gateways möchte ich die beiden Linux-Projekte PBX4Linux und Asterisk vorstellen. Beide sind frei erhältlich und bieten sich somit für den Aufbau einer kostengünstigen, auf VoIP basierenden Telefonanlage an. In Kapitel 4 werde ich die Praxistauglichkeit der beiden Projekte testen, indem ich sie für eine Beispielsanwendung im Rechenzentrum konfiguriere. Als Testkriterien werde ich dabei Unterstützung von VoIP-Protokollen, die benötigte Hardware, die Dokumentation, die Installation, die Konfiguration (insbesondere für mobile Clienten) und die Qualität bei VoIP-Gesprächen der beiden Projekte unter die Lupe nehmen.

Kapitel 2

Grundlagen

2.1 Voice over IP

Voice over IP (VoIP) steht für das Telefonieren über ein Computernetzwerk auf der Grundlage des Internet Protokolls (IP). Auf IP möchte ich an dieser Stelle jedoch nicht weiter eingehen. Ausführliche Informationen dazu finden sich in der Fachliteratur (siehe z. B. [3]). Der wesentliche Unterschied von VoIP zur klassischen Telefonie besteht darin, dass bei VoIP keine reservierte Leitung für die Dauer eines Gesprächs zur Verfügung gestellt wird. Bei VoIP werden die Daten als einzelne IP-Pakete verschickt, die auf nicht festgelegtem Weg an ihr Ziel gelangen. Man spricht auch von einer verbindungslosen Vermittlung. Der Vorteil hiervon ist, dass so die Infrastruktur, also das Netzwerk, mit anderen Diensten geteilt werden kann, und nicht mehr nur für die Telefonie reserviert ist wie im klassischen Telefonnetz.

Um die Kommunikation in einer verbindungslosen Vermittlung zu ermöglichen, müssen vorher ein paar Probleme bedacht werden. Bei der verbindungslosen Vermittlung erfolgt der Versand der Daten nur anhand der Zieladresse eines IP-Pakets. Es wird vorher nicht sichergestellt, ob der Empfänger bereit ist, die Pakete tatsächlich anzunehmen. Außerdem gibt es bei der verbindungslosen Vermittlung kein Quality of Service, d.h. die Verzögerungszeiten der Pakete sind sehr variabel und schwanken von Paket zu Paket innerhalb einer Kommunikation. Um die Sprachübertragung für die verbindungslose Vermittlung trotzdem zu ermöglichen, werden Protokolle benötigt, die sowohl den Auf- und Abbau von Verbindungen steuern, als auch den eigentlichen Versand der Nutzdaten.

2.1.1 Auf- und Abbau von Verbindungen

Um die Kommunikation in einem verbindungslosen Netzwerk zu unterstützen, werden Protokolle benötigt, die Verbindungen (sogenannte Sessions) aufbauen, steuern und wieder beenden. Dazu muss beispielsweise festgelegt werden, wie die Teilnehmer adressiert werden können, welche Art von Kommunikation erwünscht ist und welches Transportprotokoll verwendet wird. Für diese Aufgabe wurden unterschiedliche Protokolle entwickelt. In Kapitel 2.2 bzw. 2.3 werde ich näher auf die beiden bekanntesten Protokolle H.323 und SIP eingehen.

2.1.2 Übertragung der Nutzdaten

Nachdem eine Verbindung zwischen zwei (oder mehr) Teilnehmern aufgebaut ist, müssen die eigentlichen multimedialen Nutzdaten übertragen werden. Dazu wird sowohl bei H.323 als auch bei SIP das Realtime Transport Protocol (RTP) verwendet. RTP ermöglicht den Transport von Audio- und Videodaten in Paketform und sorgt für die Synchronität der Dienste. Es muss also Probleme von Paketüberholungen, Paketverzögerungen und Jitter bis hin zu Paketverlusten, wie sie in verbindungslosen Netzen vorkommen, beim Empfänger ausgleichen. Dazu erhält jedes RTP-Paket eine fortlaufende Nummer (Sequence Number) und einen Zeitstempel (Timestamp). Weiterhin ist in im RTP-Header eine eindeutige Identifikation des Senders und Empfängers enthalten. Das Realtime Control Protocol (RTCP) steuert die eigentliche Datenübertragung. Es gibt dem Sender in regelmäßigen Abständen eine Rückmeldung über die Qualität der Übertragung und sorgt für die „Lippensynchronität“ bei der Übertragung von Audio- und Videodaten. Um Laufzeitprobleme zu vermeiden, basieren beide Protokolle auf UDP als Transportprotokoll (siehe Abb. 2.1). Mehr Informationen zu RTP und RTCP und die genaue Spezifikation können im RFC 1889 [6] nachgelesen werden.

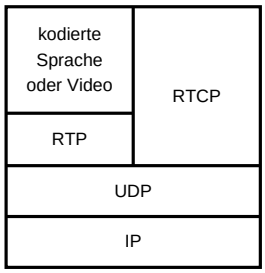

Abbildung 2.1: RTP- und RTCP-Protokollstack

2.2 H.323

Das H.323-Protokoll wurde 1996 von der ITU (International Telecommunications Union) eingeführt und stellt eine wichtige Spezifikation für die Übertragung multimedialer Daten im Internet dar. Es definiert die Ausrüstung und Dienste zur paketvermittelten Kommunikation über verbindungslose Netzwerke und ist ausgerichtet auf Netze, die kein Quality of Service bieten. Dazu baut es auf einer ganzen Reihe von Protokollen zur Signalisierung, Registration, Zugangskontrolle, Transport und Codec-Wahl auf. Die wichtigsten Funktionen und Komponenten von H.323 werden im folgenden kurz beschrieben.

2.2.1 Architektur

Im H.323-Standard sind für die Kommunikation die folgenden Komponenten definiert: Terminal, Gatekeeper, Gateway und Multipoint Controller Einheiten (siehe Abb. 2.2).

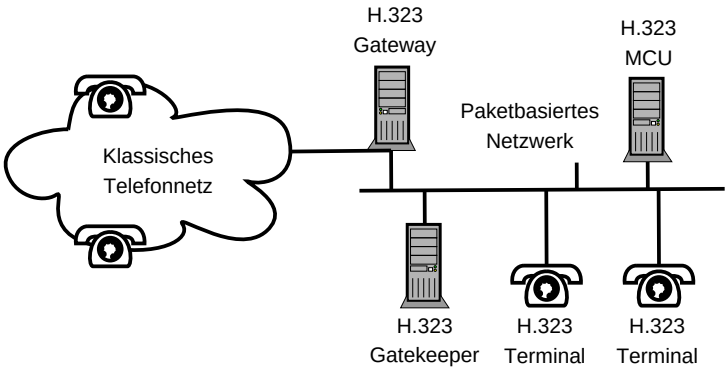

Abbildung 2.2: H.323-Komponenten

- **Terminals** sind Endgeräte, die verschiedene Formen von Audio, Video und reiner Datenkommunikation umfassen können (vom einfachen Telefon bis zum Multimedia-PC).

- **Gatekeeper** verwalten die verschiedenen Terminals und leisten außerdem administrative Dienste wie beispielsweise Adressübersetzungen und die Teilnehmerverwaltung. Bei einer Anbindung an ein klassisches Telefonnetz über einen Gateway muss ein Gatekeeper vorhanden sein, der die eingehenden Telefonnummern in IP-Adressen übersetzt.

- **Gateways** bilden die verschiedenen Netzübergänge und passen Nutz- und Signalisierungsinformationen entsprechend an. Sie ermöglichen beispielsweise die Anbindung eines H.323-Netzes an das klassische Telefonnetz, indem sie Protokollaufrufe übersetzen, die Melodienformate umwandeln, den Informationsaustausch steuern und die Einleitung und die Beendigung eines Anrufs auf beiden Seiten der Verbindung durchführen.

- **Multipoint Controller Einheiten** (MCU) unterstützen Mehrpunktverbindungen, d.h. bei Konferenzen zwischen drei oder mehr Terminals baut jedes Terminal die Verbindung über die MCU auf.

Gateway, Gatekeeper und MCU sind voneinander unabhängige Komponenten, die je nach Bedarf eingesetzt werden können. Selbstverständlich können auch alle Komponenten auf einem Rechner implementiert werden.

2.2.2 Verbindungssteuerung

Die Verbindungssteuerung bei H.323 stellt alle notwendigen Nachrichten bereit, um eine Verbindung zwischen H.323-Terminals herzustellen. In einem Netz mit Gatekeeper werden alle Nachrichten über diesen geführt. Falls in einem Netz kein Gatekeeper vorhanden ist, können die Nachrichten auch direkt von einem Terminal zum anderen geschickt werden. Die Verbindungssteuerung bei H.323 ist in folgende zwei Komponenten unterteilt:

- **H.225 (RAS - Registration, Admission, Status)** dient zum Austausch von Registrierungs-, Zugangs- und Statusinformationen zwischen H.323-Terminals und Gatekeeper. Es folgt eine kurze Erläuterung der wichtigsten RAS-Nachrichten:

  - **RegistrationRequest (RRQ)** – Registrierungsaufforderung eines Terminals oder eines Gateways an einen Gatekeeper. Der Gatekeeper antwortet entweder mit einer Bestätigung (RCF) oder mit einer Verweigerung (RCJ).
  - **AdmissionRequest (ARQ)** – Zugangsaufforderung zu einem Netz von einem Terminal an einen Gatekeeper. Der Gatekeeper antwortet entweder mit einer Bestätigung (ACF) oder mit einer Verweigerung (ARJ).
  - **InfoRequest (IRQ)** – Aufforderung einer Statusinformation des Gatekeepers an ein Terminal. Das Terminal antwortet mit einem InfoRequestResponse (IRR).

- **H.225.0 (Call Control)** wird zur Einleitung und Beendigung eines Anrufs benutzt und kennt folgende Nachrichentypen:
  - **SETUP** – Verbindungswunsch von einem H.323-Terminal an ein anderes.
  - **CALL PROCEEDING** – Verbindungswunsch wurde vom angerufenen H.323-Terminal erhalten.
  - **ALERTING** – Das angerufene H.323-Terminal „klingelt“.
  - **CONNECT** – Anruf wurde vom angerufenen H.323-Terminal angenommen.
  - **ACK** – Positive Bestätigung.

Eine komplette Übersicht aller H.225 und H.225.0 Nachrichten und deren genaue Spezifikation findet sich im H.225.0-Standard der ITU. In Abbildung 2.3 ist der Ablauf eines H.323-Verbindungsaufbau zwischen zwei Terminals über einen Gatekeeper zu sehen. Weitere Beispiele für komplexere H.323-Verbindungen finden sich in [2] oder in [5]. Nach einem erfolgreichen Verbin-

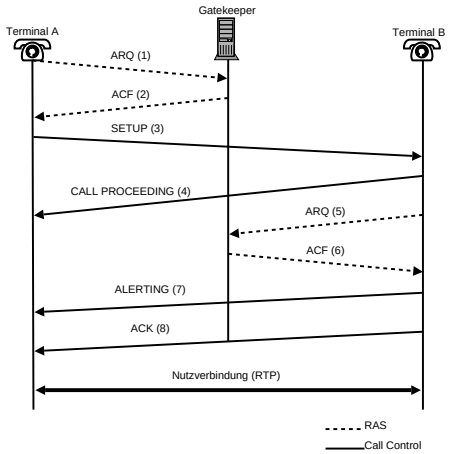

Abbildung 2.3: Verbindungsaufbau mit H.323

dungsaufbau übernehmen RTP und RTCP die Übertragung der eigentlichen Nutzdaten der Verbindung.

2.3 SIP

Das Session Initiation Protocol (SIP) ist ein Netzwerkprotokoll zum Aufbau einer Kommunikationsverbindung (Session) zwischen zwei und mehr Teilnehmern. Entwickelt wurde das Protokoll von einer Arbeitsgruppe der IETF, die

die Spezifikationen 1999 veröffentlichten. Die aktuelle Version des Protokolls mit allen Definitionen kann im RFC 3261 [4] nachgelesen werden. SIP ist ein Protokoll für das Internet und ähnelt in seiner Form dem HTTP Protokoll. Der Nachrichtenaustausch erfolgt immer zwischen einem Client und einem Server, wobei der Client Anforderungen (Request) stellt, die der Server beantwortet (Response). Die Rollen von Client und Server sind nicht festgelegt und können durchaus im Verlauf einer Session wechseln. Der Nachrichten- und Status-Austausch basiert wie bei HTTP ebenfalls auf ASCII-Code. Zur Adressierung von Benutzern wird bei SIP die sogenannte SIP URI benutzt, welcher in ihrem Format „sip:userID@domain“ einer Email Adresse gleicht. Dies bietet Vorteil, dass Benutzern dieselbe Email und SIP Adresse zugewiesen werden kann und es sich somit nur noch eine Adresse gemerkt werden muss, unter welcher der Benutzer erreichbar ist. Außerdem kann dieselbe Adresse unabhängig vom benutzten Gerät beibehalten werden, was insbesondere sinnvoll für mobile Benutzer ist.

2.3.1 Nachrichten

Eine SIP-Nachricht ist entweder eine Anforderung von einem Client an einen Server oder eine Antwort des Servers an den Client. SIP-Nachrichten sind textbasiert und bestehen immer aus einem Header und einem Body. Der SIP-Header enthält die Verbindungsparameter und Dienstinformationen, wobei viele Felder einfach von HTTP übernommen und nur ein paar neue Felder wie Call-ID und Via hinzugefügt worden sind. Der SIP-Body enthält die Beschreibung der RTP-Verbindung wie z. B. den verwendeten Audio- bzw. Video-Codec sowie dessen Parameter, IP-Adressen, TCP-Portnummern. Ein typischer SIP-Header könnte folgendermaßen aussehen:

```
INVITE sip:homer@rz.uni-freiburg.de SIP/2.0
Via SIP/2.0/UDP 132.230.4.50
From sip:bart@rz.uni-freiburg.de
To sip:homer@rz.uni-freiburg.de
Call-ID: a2e3a@rz.uni-freiburg.de
```

Die INVITE Zeile enthält die benutzte SIP Version. Immer wenn eine SIP-Nachricht eine SIP-Komponente durchläuft, wird eine Via Zeile angefügt, welche die IP-Adresse der SIP-Komponente enthält. Die erste Via Zeile wird immer von dem Gerät gesetzt, welches die Nachricht verschickt. In den nächsten beiden Zeilen stehen ähnlich wie bei Email Adressen von wem die Nachricht verschickt wurde und an wen sie adressiert ist. In der letzten Zeile steht die Call-ID, welche den Anruf eindeutig identifiziert.

Methoden Eine Methode ist eine Nachricht von einem Client an den Server. SIP kennt folgende sechs Methoden:

- **REGISTER** – Übermittelt Standortinformationen über einen Benutzer.
- **INVITE** – Lädt einen Kommunikationspartner beispielsweise zu einem Telefongespräch ein.
- **ACK** – Positive Bestätigung.
- **CANCEL** – Unterbricht die Suche für einen Benutzer.
- **BYE** – Beendet eine bestehende Kommunikation.
- **OPTIONS** – Kann Informationen zu den Eigenschaften von Endsystemen erfragen und bereitstellen, ohne eine Verbindung aufzubauen.

Statusinformationen Die Statusinformation ist eine Rückmeldung des Servers (Response) über den Status der Bearbeitung einer Anforderung. Sie wird dargestellt als eine dreistellige Zahl und eine kurze Beschreibung, die für den Menschen einfach zu verstehen ist. So wird beispielsweise die erfolgreiche Bearbeitung einer Anforderung immer mit der Statusinformation „200 OK“ quittiert. Die Statusinformationen sind in folgende sechs Grundtypen eingeteilt:

- **1xx: Informational** – Die Anforderung wurde empfangen und wird bearbeitet.
- **2xx: Success** – Die Aktion wurde erfolgreich empfangen und bearbeitet.
- **3xx: Redirection** – Weitere Aktionen müssen durchgeführt werden, um den Request vollständig zu bearbeiten.
- **4xx: Client Error** – Anforderung kann nicht bearbeitet werden (z. B. falsche Syntax).
- **5xx: Server Error** – Der Server konnte eine (wahrscheinlich) gültige Anforderung nicht bearbeiten.
- **6xx: Global Error** – Die Anforderung konnte von keinem Server bearbeitet werden.

Ein einfaches Beispiel des Zusammenspiels der Methoden und Statusinformationen bei SIP wird in Abbildung 2.4 dargestellt. Es wird eine direkte Verbindung von einem SIP User-Agent zu einem anderen aufgebaut wird. Weitere Beispiele für komplexere SIP-Verbindungen sollen an dieser Stelle nicht ausgeführt werden und können in der Fachliteratur (siehe z. B. [2] und [5]) nachgelesen werden.

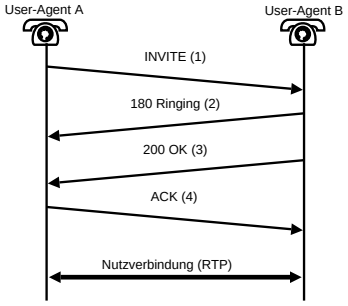

Abbildung 2.4: Einfacher Verbindungsaufbau mit SIP

Nach dem erfolgreichen Verbindungsaufbau übernimmt RTP die Übertragung der Nutzdaten der Kommunikation.

2.3.2 Architektur

Vergleichbar mit H.323 sind bei SIP ebenfalls vier Komponenten definiert: User-Agent, Registrar, Proxy Server und Redirect Server. Die Aufgaben dieser Komponenten soll im folgenden Abschnitt kurz erläutert werden.

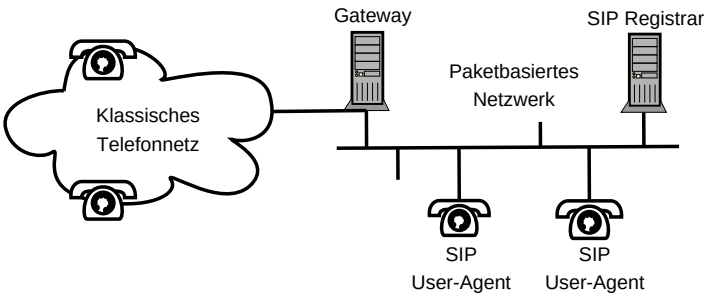

Abbildung 2.5: SIP Komponenten

- **User-Agents** sind die Endpunkte einer Verbindung, beispielsweise ein Telefon, PC oder ein PDA.

- **Registrar** nehmen bei SIP die Position der zentralen Schaltstelle ein. Jeder User-Agent schickt bei der Anmeldung und in regelmäßigen Abständen eine *REGISTER*-Nachricht and den Registrar seiner Domain. Diese Nachricht beinhaltet unter anderem die eigene SIP-Adresse (z. B. `sip:max@domain.de`) und die IP-Adresse des User-Agents, welche der Registrar als sogenanntes „Binding“ in seinem Location Service speichert. Ein Benutzer kann sich dabei auch mehrmals mit derselben Adresse registrieren, beispielsweise für sein SIP-Telefon am Arbeitsplatz und zu Hause.

- **Proxy Server** helfen beim Auffinden eines Teilnehmers. Er kann als Server und als Client agieren und im Namen anderer Clients Anfragen starten. Dies macht er unter anderem, um die Bindings von SIP-Adressen bei einem Registrar zu erfahren. Sobald er die IP-Adresse eines Empfängers bestimmt hat, leitet er die Verbindungsanfrage an dessen User Agent weiter, falls dieser sich innerhalb seiner Domain befindet. Ansonsten reicht er die Anfrage an den Proxy der anderen Domain weiter.

- **Redirect Server** kommen vor allem zum Einsatz, um die Last auf dem Proxy Server zu verringern. Ein Redirect Server gibt die Routing Informationen direkt an den Anfragenden zurück. Sie ermöglicht dem Proxy auch, die Einladung zu einem Gespräch an den Client in einer anderen Domain weiterzuleiten. Desweiteren versorgt er den anfragenden Proxy mit Informationen über die nächsten Stationen, die zu passieren sind. Danach kann der den zuständigen Proxy Server oder User Agent direkt ansprechen.

Registrar, Proxy und Redirect Server sind unabhängige Komponenten und können alle auf demselben Rechner implementiert sein. Anhand des folgenden Beispiels (siehe Abb. 2.6) aus [3] soll das Zusammenspiel der einzelnen SIP-Komponenten verdeutlicht werden. In diesem Beispiel möchte Bart (`sip:bart@ub.uni-freiburg.de`) eine SIP Verbindung mit Homer (`sip:homer@informatik.uni-freiburg.de`) aufbauen.

1. Bart sendet eine INVITE Nachricht für Homer an den SIP Proxy `ub.uni-freiburg.de`.

2. Der Proxy leitet die Nachricht weiter den SIP Registrar `informatik.uni-freiburg.de`.

3. Da Homer nicht mehr beim Registrar `informatik.uni-freiburg.de` angemeldet ist, sendet dieser ein sogenanntes „Redirect“ als Antwort,

welches besagt, dass Homer jetzt unter der Adresse `sip:homer@rz.uni-freiburg` erreichbar sein müsste.

4. Der Proxy sendet jetzt die INVITE Nachricht an `rz.uni-freiburg`.

5. Der Registrar findet in seinen Bindings die IP Adresse, über die `homer@rz.uni-freiburg` erreichbar ist und leitet die INVITE Nachricht an dessen SIP User-Agent weiter.

6.-8. Die SIP Antwort „200 OK“ wird über die Registrar und Proxy an den User-Agent von Bart gesendet.

9. Die Nutzverbindung für die multimedialen Daten wird direkt zwischen den beiden User-Agents aufgebaut.

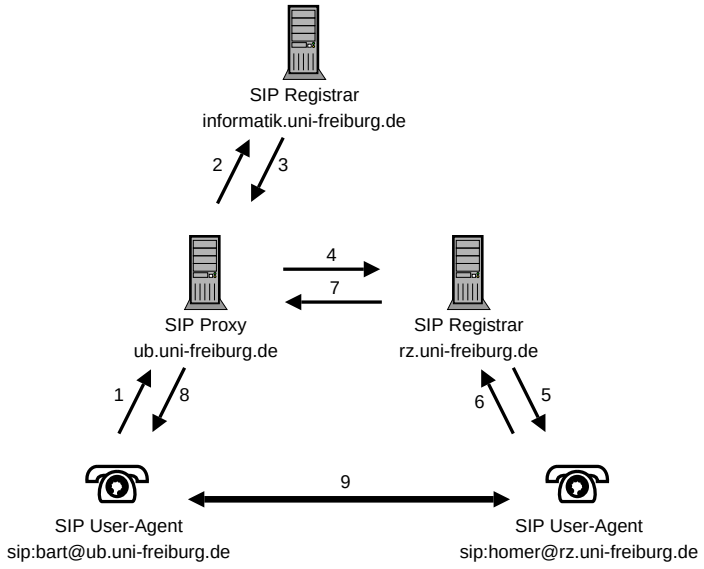

Abbildung 2.6: Zusammenspiel der SIP-Komponenten bei einem Verbindungsaufbau

Kapitel 3

Open-Source-Telefonanlagen für Linux

3.1 Überblick

Es existiert mittlerweile eine Vielzahl von Projekten, die an Open-Source-Telefonanlagen für Linux arbeiten. In dieser Studienarbeit möchte ich zwei davon vorstellen. Zum ersten ein eher kleines Projekt namens PBX4Linux, welches H.323 unterstützt und zum zweiten Asterisk, die wohl bekannteste Telefonanlage unter Linux, die sowohl H.323 als auch SIP und ein eigenes VoIP-Protokoll IAX unterstützt.

3.2 PBX4Linux

PBX4Linux ist eine frei erhältliche Linux-Software, mit der ein normaler PC mit einer ISDN-Karte und einem Internetanschluss in eine sehr flexible Telefonanlage verwandelt werden kann. Beispielsweise kann die PBX (Private Branch Exchange) per VoIP Gespräche von H.323-Geräten in einem lokalen Netzwerk an Gegenstellen im Telefon-Festnetz vermitteln. Dies war ursprünglich von dem Entwickler Andreas Eversberg gar nicht so gedacht. Sein Ziel bei der Entwicklung von PBX4Linux war es nur, mit einem normalen ISDN-Telefon per H.323 zu telefonieren können. Eine ausführliche Anleitung dazu findet sich unter [8]. Nach und nach ist dann aus dem Projekt eine vollwertige, eigenständige Telefonanlage entstanden. SIP-Unterstützung ist bisher bei PBX4Linux leider noch nicht implementiert, ist aber für die Zukunft geplant. Die folgende Ausführung befasst sich mit der Funktionsweise der Software und der Konfiguration als H.323-Gateway.

3.2.1 Ablauf eines Anrufs

Wenn ein Anruf bei der PBX ankommt, dann schaut sich die PBX zuerst an, ob es sich bei dem Anruf um einen internen (von einer Nebenstelle) oder um einen externen Anruf handelt und konsultiert dann die entsprechende Konfigurationsdatei. Dort wird überprüft, ob die gewählte Nummer angerufen werden darf und wenn ja, ob dabei eine bestimmte Aktion ausgeführt werden soll. H.323-Anrufe werden gesondert behandelt: Bei einem eingehenden H.323-Anruf wird zuerst in der Datei `h323_gateway.conf` überprüft, ob der IP-Adresse des anrufenden H.323-Terminals eine Nebenstelle zugeordnet ist. Wenn dies der Fall ist, dann handelt es sich bei dem H.323-Anruf um einen normalen internen Anruf und es wird die Datei `numbering_int.conf` für weitere Aktionen konsultiert. Falls der IP-Adresse keine Nebenstelle zugeordnet ist, dann wird der Anruf wie ein externer Anruf behandelt.

3.2.2 Nebenstellen

Für jedes Telefon (auch softwarebasierte Telefone), das ein Teil der Telefonanlage sein soll, muss in der PBX eine Nebenstelle definiert sein. Die Nebenstelle ist die Telefonnummer, mit der das angeschlossene Telefon innerhalb der Telefonanlage erreichbar ist. Sie muss also für jedes Gerät eindeutig sein. Bei PBX4Linux können die Nebenstellen einfach mit dem Kommando `genextension` generiert werden. Das Kommando legt dabei für jede Nebenstelle ein neues Verzeichnis `/usr/local/pbx/extensions/<Nebenstelle>` an. Dieses enthält eine Konfigurationsdatei, eine Log-Datei und ein Telefonbuch, in dem Kurzwahlnummern für die Nebenstelle definiert werden können.

Kommandosyntax `genextension`

```
genextension EXTENSION PORT CALLERID
```

- `EXTENSION` – ist die Nummer der Nebenstelle, z. B. 200. Sie kann im Grunde genommen beliebig gewählt werden, muss jedoch für jedes angeschlossene Gerät eindeutig sein. ist. Beispielsweise leitet 1 den Anruf an den ersten internen Port der PBX weiter.

- `CALLERID` – definiert die Nummer, die angezeigt werden soll, wenn Anrufe nach außen gemacht werden. Dies ist normalerweise die Rufnummer des ISDN-Anschlusses, der die die PBX mit dem Festnetz verbindet.

3.2.3 Konfiguration

Konfiguration der Anrufs-Dateien Je nach Art des eingehenden Anrufs (intern bzw. extern) konsultiert die PBX die entsprechende Anrufs-Datei `numbering_int.conf` bzw. `numbering_ext.conf`. In diesen Dateien wird festgelegt, ob die gewählte Nummer von intern (extern) erreichbar ist und welche Aktion von der PBX bei der Wahl dieser Nummer durchgeführt werden soll. Einträge sind in beiden Konfigurationsdateien im selbem Format:

```
Präfix/Nebenstelle  Aktion  [ Parameter [...]]>
```

Dabei stehen zahlreiche Aktionen zur Verfügung. Beispielsweise verbindet die Aktion `internal` den Anruf mit der gewählten Nebenstelle. Es kann aber auch bei der Wahl einer bestimmten Nummer eine Audio-Datei abgespielt oder die Mailbox abgehört werden. Eine Übersicht aller verfügbaren Aktionen ist in der Datei `numbering_int.conf` aufgelistet und ausführliche Erklärungen dazu finden sich in der PBX4Linux-Dokumentation. Im folgenden werde ich kurz zwei Beispielskonfigurationen der Konfigurationsdateien erläutern.

`numbering_int.conf`

```
0       outdial
200     internal    200
201     h323        192.168.1.13#
```

Wenn eine 0 von einem Telefon der Telefonanlage gewählt wird, dann bekommt die Nebenstelle ein Amt wie bei einer normalen ISDN-Telefonanlage und es können Anrufe in das externe Telefonnetz geführt werden. Bei der Wahl der Nummer 200 wird der Anruf an die interne Nebenstelle 200 vermittelt. In der letzten Zeile wird die Nummer 201 als Kurzwahl für die IP-Adresse eines H.323-Terminals definiert. Hierbei ist wichtig, dass bei Angabe einer IP-Adresse die Zeile immer mit einem "#"-Zeichen beendet werden muss. Alternativ hierzu kann eine H.323-Adresse auch in der Form `userID@domain` angegeben werden.

`numbering_ext.conf`

```
0       internal    200
1       external    076112345
2       h323        192.168.1.13#
```

Angenommen die Rufnummer des ISDN-Anschlusses, mit dem die PBX verbunden ist, sei 1234. In der Datei `numbering_ext.conf` können jetzt Durchwahlen für die Nebenstellen definiert werden, so dass man mit der an die Stammnummer angefügten Durchwahl mit einer Nebenstelle verbunden wird. Durchwahlen müssen allerdings vom ISDN-Anschluss unterstützt werden, d. h. es wird ein sogenannter Anlagenanschluss (Point-to-Point) benötigt. Durch die obige Konfiguration werden dann alle eingehenden Anrufe mit der Nummer 1234-0 an die interne Nebenstelle 200 und alle Anrufe an die Nummer 1234-1 werden an die externe Nummer 076112345 weitergeleitet. In der letzten Zeile wird definiert, dass Anrufe an die Nummer 1234-2 über VoIP an ein H.323-Terminal mit der IP-Adresse 192.168.1.13 weitergeleitet werden.

Konfiguration als H.323-Gateway Um PBX4Linux als H.323-Gateway zu konfigurieren, so dass Gespräche von H.323-Terminals ins klassische Telefonnetz aufgebaut werden können, muss die Datei `h323_gateway.conf` konfiguriert werden. Dort muss für jedes H.323-Terminal, das den Gateway benutzen soll, dessen IP-Adresse einer Nebenstelle zugeordnet werden. Verbindungen von H.323-Terminals, deren IP-Adressen nicht in der Datei `h323_gateway.conf` eingetragen sind, werden von der PBX wie externe Anrufe behandelt. In der folgenden Beispielskonfiguration sollen von dem H.323-Terminal mit der IP-Adresse 192.168.1.14 Telefongespräche ins Telefonnetz geführt werden dürfen. Es bekommt also die Nebenstelle 202 zugewiesen. Die Option `connect` bewirkt, dass die PBX den H.323-Anruf sofort entgegen nimmt und den Rufaufbau ins Festnetz hörbar macht.

```
h323_gateway.conf

192.168.1.14   200   [connect]
```

3.2.4 Kommandos

Im Folgenden möchte ich kurz die wichtigsten Kommandos zur Bedienung der PBX vorstellen. Eine Übersicht über alle Kommandos findet sich in der PBX4Linux Dokumentation.

- `pbx start` – startet PBX4Linux.
- `pbx state` – startet PBX4Linux und zeigt eine Übersicht aller Ports und aller aktiven Anrufe.
- `pbx query` – zeigt eine Übersicht aller ISDN-Karten und den zugeordneten Ports.

3.3 Asterisk

Asterisk ist eine sehr umfangreiche, software-basierte Telefonanlage für GNU/-Linux. Das Programm wurde ursprünglich von Mark Spencer von Digium, Inc. geschrieben und wird mittlerweile als Open Source Projekt weiterentwickelt. Asterisk unterstützt die VoIP-Protokolle SIP, H.323 und das eigene Protokoll IAX (Inter Asterisk Exchange) und kann mit nahezu allen Standard-Telefonie-Geräten mit Hilfe von relativ preisgünstiger Hardware kommunizieren.

3.3.1 Ablauf eines Anrufs

Ein Anruf kommt immer über einen bestimmten Kanal herein und ist an eine „gewählte Nummer“ gerichtet. Asterisk schaut sich zuerst an, über welchen Kanal der Anruf hereinkommt und liest dann die entsprechende Konfigurationsdatei. Beispielsweise wird für eingehende SIP-Anrufen, die Datei `sip.conf` konsultiert. In dieser ist definiert, an welchen Kontext in der Datei `extensions.conf` der Anruf gesendet werden soll. Ein Kontext enthält mindestens einen Nummernvergleich, welcher bei einer Übereinstimmung mit der gewählten Nummer eine Anwendung ausführt.

3.3.2 Kontext und Nummernvergleiche

Ein Kontext ist ein zentrale Konfigurationseinheit von Asterisk, und ist nichts anderes als eine Ansammlung von Nummernvergleichen („extension match“). Alle Kontexte werden in der Datei `extensions.conf` definiert. Innerhalb eines Kontextes gibt es mindestens einen Nummernvergleich mit dem die gewählte Nummer (gespeicherte in der Asterisk-Variable `$EXTEN`) verglichen wird. Falls es für dieselbe Nummer mehrere Nummernvergleiche gibt, dann werden diese in aufsteigender Reihenfolge ihrer Priorität bearbeitet. Ein Nummernvergleich hat die folgende Form:

```
exten => Vergleichsnummer,Priorität,Anwendung
```

Die Vergleichsnummer kann auch Jokerzeichen enthalten. Allerdings muss sie dann mit einem "_"-Zeichen beginnen. Beispiele für Jokerzeichen sind `N` für die Zahlen von 2-9, `X` für irgendeine Zahl, `.` für eine beliebige Anzahl von Zahlen und natürlich eine Vielzahl von regulären Ausdrücken. Ausführlichere Informationen zu Jokerzeichen in Asterisk gibt es im Asterisk Handbuch.

3.3.3 Anwendungen

Wenn passende Nummernvergleiche für `$EXTEN` gefunden wurden, dann werden die zugehörigen Anwendungen in der Reihenfolge ausgeführt, in der sie durch die Prioritäten eingeordnet wurden. Eine Übersicht aller verfügbaren Anwendungen erhält man mit dem Befehl `show applications`. Danach kann mit `show application xy` eine Erklärung der Anwendung `xy` angezeigt werden.

Eine häufig benutzte Anwendung ist Dial. Dial ruft einen entfernten Kanal an, und verbindet den Eingangskanal mit dem entfernten Kanal, falls dieser antwortet. Die verbundenen Kanäle werden wieder getrennt, wenn eine Seite auflegt. Dial beendet sich dann mit einem „Non-Zero“ Status und die Prioritätenliste wird nicht mehr weiter abgearbeitet, da der Anruf beendet wurde. Falls der Anruf innerhalb einer festgelegten Zeit nicht beantwortet wird, dann beendet sich Dial und die Anwendung mit der nächsthöheren Priorität wird ausgeführt.

Beispiel Im folgenden Beispiel werde ich den Ablauf eines Anrufs, der an den Kontext `default` geschickt wird, erläutern.

```
[default]
exten => _0.,1,Wait,1
exten => _0.,2,Dial(CAPI/${EXTEN:1}, 20)
exten => _0.,3,Congestion
exten => _0.,4,Hangup
```

Die Vergleichsnummer `_0.` steht für alle Nummern, die mit einer `0` beginnen. Wenn also die Nummer des eingehenden Anrufs `$EXTEN` mit einer `0` beginnt, dann wird nach einer Wartezeit von 1 Sekunde der CAPI-Kanal mit der gewählten Nummer ohne die erste Ziffer `0` angerufen (mit `${EXTEN:1}` wird die erste Stelle der angerufenen Nummer abgeschnitten). So wird also die Amtsholung bei einer normalen ISDN-Telefonanlage simuliert. Falls die Gegenstelle den Anruf annimmt, werden die Kanäle verbunden und die Teilnehmer können miteinander sprechen. Andernfalls wird nach 20 Sekunden die nächste Anwendung ausgeführt, die den Besetztton („Congestion“) abspielt bevor der Anruf in der letzten Zeile beendet („Hangup“) wird.

3.3.4 Kommandos

Asterisk wird mit normalerweise mit dem Kommando `safe_asterisk` im Hintergrund gestartet. Das Kommando `asterisk -r` stellt eine Verbindung

zu einem laufenden Asterisk Prozess her und startet eine konsolenähnliches Interface. Danach kann in der Kommandozeile mit `?` eine komplette Liste der Befehle angezeigt werden. Es folgt ein kurzer Überblick über die wichtigsten Befehle:

- `reload` - Neustart von Asterisk, wobei alle Konfigurationsdateien neu geladen werden.
- `show dialplan` - Zeigt den kompletten Dialplan mit allen Kontexten und Nummernvergleichen.
- `sip show peers` - Listet alle registrierten SIP-Clienten an.
- `stop gracefully` - Beendet Asterisk nachdem alle Anrufe beendet wurden.
- `stop now` - Beendet Asterisk und alle aktiven Anrufe.

Kapitel 4

Beispielanwendung im Rechenzentrum

4.1 Situation

Im Rechenzentrum der Universität Freiburg wird momentan eine herkömmliche ISDN-Telefonanlage benutzt. Es können nur Endgeräte vom selben Hersteller an die Anlage angeschlossen werden und daher ist beispielsweise der Einsatz von mobilen Endgeräten relativ teuer. Ausserdem müssen in der jetztigen Situation im Rechenzentrum zwei Netze parallel verwaltet werden, die beide der Kommunikation dienen. Es stellt sich also die Frage, ob eine VoIP-Telefonanlage eine Alternative zu der bisher benutzten ISDN-Telefonanlage darstellt. Die Netzwerk-Infrastruktur ist im Rechenzentrum sehr gut ausgebaut und über das WLAN könnten relativ einfach mobile Endgeräte wie Laptops oder PDAs mit der Telefonanlage verbunden werden.

In diesem Kapitel möchte ich also die Tauglichkeit und Funktionalität der im vorigen Kapitel vorgestellten Linux-basierten Telefonanlagen mit dem folgenden Versuchsaufbau testen (siehe Abb. 4.1). Der Rechner mit der IP-Adresse 132.230.4.98 soll dabei als Telefon-Gateway dienen, der Gespräche von VoIP-Clienten ins Festnetz weiterleitet und umgekehrt.
Als Betriebssystem ist auf dem Rechner SuSE Linux 9.1 installiert. Um eine Verbindung mit den VoIP-Clienten herstellen zu können, ist er mit mit dem Local Area Network (LAN) 132.230.4.0/24 verbunden. Ausserdem wurde eine AVM Fritzcard PCI eingebaut, die den Rechner über den ISDN-Anschluss mit der Rufnummer 2025020 mit dem Telefonnetz verbindet. An die Telefonanlage sollen zwei VoIP-Clienten angeschlossen werden, der erste hat die feste IP-Adresse 132.230.4.99 und befindet sich im LAN. Der zweite VoIP-Client

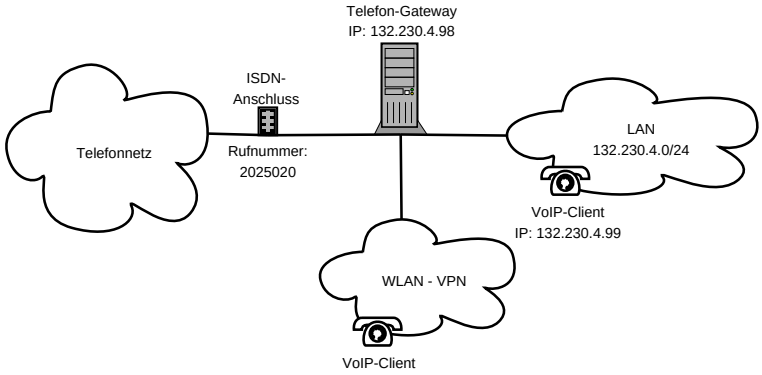

Abbildung 4.1: Versuchsaufbau

befindet sich im WLAN und ist durch einen IPSec-Tunnel mit dem Virtual Private Network (VPN) für mobile Geräte verbunden.

4.2 Implementierung mit PBX4Linux

Als erstes soll der oben beschriebene Versuchsaufbau mit PBX4Linux implementiert werden. Vorher müssen jedoch erst einmal die Software und die benötigten Treiber installiert werden. Da alle Komponenten einzeln installiert und kompiliert werden müssen ist diese Arbeit recht mühselig. Im nächsten Paragraph werde ich erläutern, was dabei zu beachten ist, bevor ich dann die eigentliche Konfiguration der PBX erklären werde.

4.2.1 Software und Treiber

Die Installation der Software und der Treiber ist bei PBX4Linux relativ aufwendig. Alle Quellcode-Pakete der Software, die benötigten Treiber und die Dokumentation zur Installation können über die Webseite `http://isdn.jolly.de` heruntergeladen werden. PBX4Linux verwendet nicht die üblichen ISDN-Treiber wie ISDN4Linux oder CAPI, sondern basiert auf den neuen mISDN-Treibern, die für den Kernel 2.6 entwickelt wurde. Diese sind im offiziellen Linux-Kernel noch nicht enthalten und der Kernel muss daher zuerst gepatcht werden. Dazu benutzt man am Besten das Skript `std2kern` des mISDN-Pakets, welches den Quellcode der Treiber nach `/usr/src/linux` kopiert. Die mISDN-Treiber können jetzt bei der Kernel Konfiguration in der Gruppe „Device Drivers/ ISDN Subsystem“ als Module ausgewählt (siehe Abb. 4.2) werden. Hierbei ist zu beachten, dass man neben dem Treiber für die eigene ISDN-Karte, auch das Modul „Digital Audio Processing“ aus-

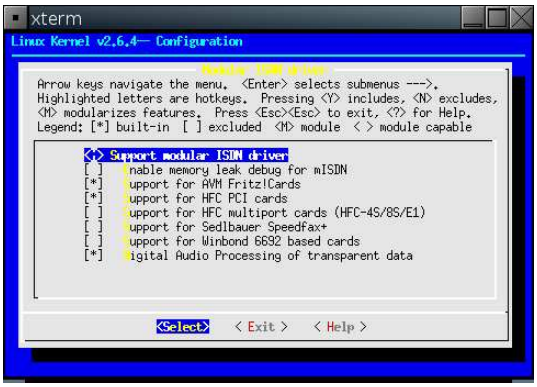

Abbildung 4.2: Auswahl der mISDN-Treiber bei der Kernel-Konfiguration

gewählt werden muss. Die Option „Enable memory leak debug for mISDN“ darf hingegen nicht ausgewählt sein [8]. Danach können die Module wie gewohnt kompiliert werden.

Nachdem der Kernel vorbereitet wurde, muss als nächstes die Bibliothek mISDNuser übersetzt werden. Dazu muss zuerst ein symbolischer Link mit `ln -s /usr/src/linux/include/linux/mISDNif.h /usr/include/linux` gesetzt werden, damit der Compiler die Header-Dateien aus dem Kernel unter `/usr/include/linux` findet. Die Kompilation wird dann wie gewohnt mit dem Kommando `make` im Verzeichnis mISDNuser gestartet. Zum Abschluss der Installation müssen jetzt noch die eigentlichen Programme für PBX4Linux übersetzt werden. Hierbei muss die Option `WITH_H323` im Makefile gesetzt werden, um die PBX mit H.323-Unterstützung zu kompilieren. Nachdem die Programme mit einem `make` im Verzeichnis `pbx4linux` erfolgreich kompiliert wurden, können sie mit `make install` installiert werden.

Um die mISDN-Treiber Module einfach in den Kernel zu laden und nach Gebrauch wieder zu entfernen, kann mit dem Befehl `genrc` ein RC-Skript namens `mISDN` erstellt werden. Wenn die Module bei einem Aufruf von `mISDN start` erfolgreich geladen werden, kann mit dem Aufruf von `pbx query` die Belegung der benutzten ISDN-Ports angezeigt werden lassen. Die erhaltenen Portzuordnungen müssen dann in der Konfigurationsdatei `options.conf` für die Parametern `te_if` bzw. `nt_if` eingetragen werden. Der Kernel und die PBX sind jetzt für einen erfolgreichen Einsatz vorbereitet.

Erzeugen der Nebenstellen Nachdem alle Vorbereitungen des Kernels und der PBX erfolgreich beendet wurden, muss im nächsten Schritt für jedes

Gerät, das an die Telefonanlage angeschlossen wird, eine Nebenstelle mit dem Kommando `genextension` erzeugt werden. Für den Versuchsaufbau (siehe Abb. 4.1) sollen zwei H.323-Terminals an die Telefonanlage angeschlossen werden. Das erste Terminal im LAN bekommt die Nebenstelle 200 und das zweite im WLAN bekommt die Nebenstelle 201 zugewiesen. Bei ausgehenden Anrufen wird für beide Nebenstellen die Rufnummer des ISDN-Anschlusses 2025020 übermittelt.

```
# genextension 200 1 2025020
# genextension 201 1 2025020
```

4.2.2 Konfiguration

Alle Konfigurationsdateien für PBX4Linux befinden sich im Verzeichnis `/usr/local/pbx`. Da PBX4Linux leider keine dynamische IP-Adressen unterstützt, wird die IP-Adresse 132.230.129.201 des mobilen H.323-Terminals im VPN fest in die Konfigurationsdateien eingetragen.

1. `numbering_int.conf`:

   ```
   0 outdial
   200 h323 132.230.4.99#
   201 h323 132.230.129.201#
   ```

   In der Datei `numbering_int.conf` werden alle Nummern definiert, die von einer Nebenstelle der Telefonanlage angerufen werden können. In diesem Fall wird die Nummer 0 als Vorwahl für Anrufe ins Telefonnetz, die 200 als Kurzwahl-Nummer für das H.323-Terminal im LAN und die 201 als Kurzwahl-Nummer für das Terminal im WLAN definiert.

2. `numbering_ext.conf`:

   ```
   2025020 h323 132.230.4.99#
   ```

   In der Datei `numbering_ext.conf` werden die Nummern festgelegt, die für externe Anrufe erreichbar sind. Da der verfügbare ISDN-Anschluss beim Versuchsaufbau kein Anlagenanschluss ist, können nur externe Anrufe an die Nummer 2025020 von der PBX entgegengenommen werden. Diese sollen an das H.323-Terminal mit der IP-Adresse 132.230.4.98 im LAN weiterleitet werden.

3. `h323_gateway.conf`:

```
132.230.4.99 200 connect
132.230.129.201 201 connect
```

Damit die Anrufe von den H.323-Terminals wie interne Anrufe behandelt werden und auch Anrufe in das Telefon-Festnetz weitergeleitet werden, müssen den IP-Adressen der H.323-Terminals in der Datei `h323_gateway.conf` noch Nebenstellen zugewiesen werden. Leider unterstützt PBX4Linux weder dynamische IP-Adressen noch DNS-Hostnamen H.323-Terminals. Dies stellt insbesondere ein Problem für das mobile H.323-Terminal im WLAN des Versuchsaufbaus dar, da dieses beim Aufbau des VPN-Tunnels eine dynamische IP-Adresse erhält. Um dieses Problem zu umgehen, habe ich zu Testzwecken die dynamische IP-Adresse fest in die Konfigurationsdatei eingetragen.

Damit ist die Konfiguration von PBX4Linux als H.323-Gateway auch schon abgeschlossen. Zum Schluss müssen nur noch die H.323-Terminals für den Einsatz mit der PBX konfiguriert werden.

Konfiguration der H.323-Terminals Als H.323-Terminals können beispielsweise GnomeMeeting unter Linux und NetMeeting unter Windows benutzt werden. Um die H.323-Terminals bei der PBX anzumelden, muss nur noch die IP-Adresse der PBX 132.230.4.98 als Gateway eingetragen werden (siehe Abb. 4.3). Danach kann von den H.323-Terminals aus wie gewohnt telefoniert werden.

Abbildung 4.3: Konfiguration von GnomeMeeting

4.3 Implementierung mit Asterisk

Die Konfiguration von Asterisk als Telefon-Gateway, so dass Gespräche von angeschlossenen SIP-Clienten ins Festnetz weitergeleitet werden und umgekehrt, ist ebenfalls relativ einfach.

4.3.1 Software und Treiber

Asterisk läuft unter Linux ab der Kernel-Version 2.4.x, OpenBSD, FreeBSD und Mac OS X. Die Software wird als Open Source Projekt entwickelt und die meisten Linux-Distributionen liefern Asterisk schon als vorgefertigtes Paket mit, welches ganz einfach über den jeweiligen Paket-Manager installiert werden kann. Alternativ kann der Quellcode von Asterisk auch auf der Asterisk Homepage (`http://www.asterisk.org`) heruntergeladen und dann selbst kompiliert werden. Zum Kompilieren der Pakets, einfach in das neue Verzeichnis `asterisk` wechseln, und mit den Befehlen `make` und `make install` die Software kompilieren und installieren.

Asterisk unterstützt standardmäßig nur die alten ISDN4linux-Treiber des Linux Kernels. Da bei diesen Treibern einige ISDN-Funktionen nicht implementiert sind, sollte das zusätzliche Treiberpaket chan_capi installiert werden, welches die neueren CAPI-Treiber unterstützt. Die Treiber stehen auf `http://www.junghanns.net/asterisk/` zum Download bereit. Um diese zu nutzen, braucht man eine capi-fähige ISDN-Karte und die dazugehörigen CAPI-Treiber des Kernels, welche im Kernel als Module kompiliert werden müssen (siehe Abb. 4.4). Die Module können dann mit dem Befehl `capiinit start` in den Kernel geladen werden und mit dem Befehl `capiinfo` kann eine Übersicht der installierten CAPI-Controller angezeigt werden.

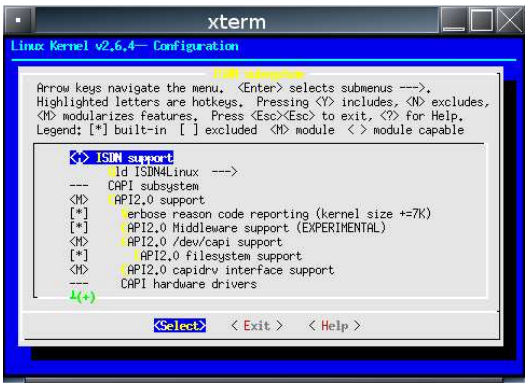

Abbildung 4.4: Auswahl der CAPI-Treiber bei der Kernel-Konfiguration

Eine Beschreibung der Installation von chan_capi findet sich in der mitgelieferten Datei `INSTALL`. Nach einer erfolgreichen Installation müssen noch folgende Einträge in der Asterisk-Konfigurationsdatei `/etc/asterisk/modules.conf` gemacht werden:

```
load => res_features.so
load => chan_capi.so
```

Und in der Sektion global:

```
chan_capi.so=yes
```

Zum Testen der Installation, den Asterisk-Server mit `safe_asterisk` und eine Asterisk-Konsole mit `asterisk -r` starten und in der Asterisk-Konsole den Befehl `capi info` ausführen. Es sollten dann die vorhandenen ISDN-Kanäle angezeigt werden.

4.3.2 Konfiguration

Zuerst müssen in den Dateien `sip.conf` und `capi.conf` die Benutzer-Konten angelegt werden, in denen auch definiert wird, an welchen Kontext eingehende Anrufe dieser Kanäle gesendet werden sollen. Danach wird in der Datei `extensions.conf` festgelegt, welche Anwendungen Asterisk für den jeweiligen Kontext ausführen soll. Alle Konfigurationsdateien für Asterisk befinden sich im Verzeichnis `/etc/asterisk`.

1. `sip.conf`:

```
[general]
 port = 5060
 bindaddr = 0.0.0.0
 context = bogon-calls

[200]
 type=friend
 username=200
 secret=200pw
 host=dynamic
 context=from-sip

[201]
 type=friend
 username=201
```

```
 secret=201pw
 host=dynamic
 context=from-sip
```

Es werden zwei Benutzer-Konten angelegt, deren Anrufe an den Kontext `from-sip` in der Datei `extensions.conf` weitergelegt werden. Beide Benutzer-Konten werden für SIP-Clienten mit dynamischer IP-Adresse eingerichtet, d. h. die Benutzer können sich unabhängig von benutztem Gerät und IP-Adresse am Asterisk-Server anmelden.

2. `capi.conf`:

```
[general]
 nationalprefix=0
 internationalprefix=0049
 rxgain=0.8
 txgain=0.8

[interfaces]
 msn=2025020
 incomingmsn=2025020
 controller=1
 softdtmf=1
 context=from-capi
 devices=1
```

In der Datei `capi.conf` muss die Rufnummer des ISDN-Anschlusses 2025020 eingetragen werden und es wird definiert, dass alle Anrufe, die über die ISDN-Karte bei Asterisk ankommen, an den Kontext `from-capi` weitergeleitet werden.

3. `extenstions.conf`:

```
[from-sip]
exten => 200,1,DIAL(SIP/200,30)
exten => 200,2,Hangup
exten => 201,1,DIAL(SIP/201,30)
exten => 201,2,Hangup
exten => _0.,1,Wait,1
exten => _0.,2,Dial(CAPI/@2025020:${EXTEN:1})
exten => _0.,3,Congestion
exten => _0.,4,Hangup
```

```
[from-capi]
exten => s,1,Wait,1
exten => s,2,Dial(SIP/200,60)
exten => s,3,Hangup
```

Für alle eingehenden SIP-Anrufe an die Nummer 200 bzw. 201 wird der Anruf mit dem entsprechenden SIP-Client verbunden. Mit der Vorwahl einer 0 können Anrufe von den SIP-Clienten über den ISDN-Anschluss ins Telefon-Festnetz gemacht werden. Im Kontext `from-capi` wird festgelegt, dass alle eingehenden Anrufe über die ISDN-Karte an den SIP-Clienten 200 weitergeleitet werden.

Damit ist Asterisk auch schon fertig für den Einsatz als Telefon-Gateway. Zum Schluss müssen nur noch die SIP-Clienten für die Anmeldung bei Asterisk konfiguriert werden.

Konfiguration der SIP-Clienten Als SIP-Client können beispielsweise KPhone oder LinPhone unter Linux und Xten X-Lite unter Windows benutzt werden. Damit sich die SIP-Clienten bei Asterisk anzumelden können, müssen sie noch mit den entsprechenden Benutzerdaten konfiguriert werden (siehe Abb. 4.5). Danach können von den SIP-Clienten aus Gespräche ins Telefon-Festnetz oder zum anderen SIP-Clienten geführt werden.

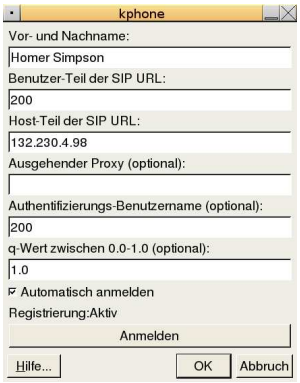

Abbildung 4.5: Konfiguration von KPhone

4.4 Ergebnis des Versuchsaufbaus

PBX4Linux ist ein relativ junges Projekt und unterstützt bisher nur das H.323-Protokoll. Für die Verbindung der PBX mit dem Telefonnetz reicht

eine einfache, passive ISDN-Karte (z. B. AVM Fritzcard) aus. Es wird also keine teure Spezialhardware für die PBX benötigt. Die Dokumentation von PBX4Linux ist recht ausführlich, allerdings sind die Erläuterungen darin oft missverständlich. Daher fand ich es relativ schwierig, die Funktionsweise und das Zusammenspiel der verschiedenen Komponenten zu verstehen. Es existiert auch eine Mailing-Liste für PBX4Linux, die bei Problemen sehr hilfreich ist. Die Installation von PBX4Linux ist nicht trivial und relativ zeitaufwändig. Eine gewisse Erfahrung im Umgang mit Linux und dem Kompilieren von Programmen ist auf jeden Fall von Vorteil. Ein großes Problem ergibt sich bei der Definition der mobilen Clienten, die den H.323-Gateway benutzen dürfen. In den Konfigurationsdateien von PBX4Linux können nämlich nur feste IP-Adressen für H.323-Terminals eingetragen werden. Da mobile Geräte, die über einen VPN-Tunnel mit dem Rechenzentrum verbunden sind, eine dynamische IP-Adresse erhalten, konnten diese nur über die feste Eintragung der dynamischen Adresse die PBX als Gateway benutzen. Wenn PBX4Linux und die benötigten Treiber einmal installiert und konfiguriert sind, dann läuft die Telefonanlage stabil und es können Gespräche von H.323-Terminals ins Festnetz und umgekehrt in einer relativ guten Qualität geführt werden. „Allerdings ist der VoIP-Gateway bei PBX4Linux zu langsam“ (Andreas Eversberg), was sich durch eine Verzögerung der Stimmen bei H.323-Telefonaten bemerkbar macht.

Asterisk ist eine sehr ausgereifte und umfangreiche Telefonanlage für Linux, die beide gängigen VoIP-Protokolle SIP und H.323 und das eigene Protokoll IAX unterstützt. Für die Verbindung mit dem ISDN-Anschluss reicht wie bei PBX4Linux eine einfache passive ISDN-Karte (z. B. AVM Fritzcard) aus. Die Asterisk Dokumentation ist sehr umfangreich und es gibt zahlreiche Anleitungen und Foren im Internet, mit deren Hilfe man eine Lösung für nahezu jedes erdenkliche Problem finden kann. Die Installation der Software ist unproblematisch und es existieren schon vorgefertigte Pakete für viele gängige Linux-Distributionen (Debian, Gentoo, SuSE 9.2), was die Installation nochmals erheblich erleichtert. Die Konfiguration von Asterisk als VoIP-Gateway ist recht einfach. Die Anmeldung von mobilen SIP-Clients, die durch einen VPN-Tunnel mit Asterisk verbunden sind, stellt ebenfalls kein Problem dar, da die SIP-Benutzerkonten für dynamische IP-Adressen eingerichtet werden können. Asterisk läuft sehr stabil und die Qualität von VoIP-Telefonaten über den Gateway ist ebenfalls sehr gut, wobei hier relativ grosse Unterschiede zwischen den verschiedenen SIP-Clienten festgestellt werden konnte. Xten X-Lite unter Windows hat hierbei am Besten abgeschnitten, da es eine sehr klare und natürliche Sprachübertragung ohne Störgeräusche hatte. Danach danach kam LinPhone unter Linux, welches ebenfalls eine relativ gute

Sprachübertragung hatte. Am schlechtesten hat KPhone abgeschnitten, welches Probleme mit den Soundkarten-Treibern hatte und somit die Sprache nur sehr schlecht und mit starken Störgeräuschen übermitteln konnte.

Alles in allem ist Asterisk für professionelle Anwendungen mit Sicherheit die bessere Wahl. Asterisk unterstützt beide gängigen VoIP-Protokolle SIP und H.323, hat eine sehr gute und ausführliche Dokumentation, ist recht einfach zu konfigurieren und unterstützt dynamische IP-Adressen, was insbesondere für die Kommunikation mit mobilen Clienten sehr wichtig ist.

Kapitel 5

Zusammenfassung

VoIP ist mittlerweile eine ausgereifte Technologie, die durchaus mit den klassischen Telefonsystemen konkurrieren kann. Durch die gut ausgebauten Netzwerk-Infrastrukturen in vielen Organisationen und Firmen, ist eine Integration von VoIP relativ einfach zu realisieren. Als VoIP-Protokoll ist SIP nach der Meinung vieler Experten das Protokoll der Zukunft. Dies liegt wohl vor allem daran, dass SIP ein Protokoll aus dem Internet ist, HTTP stark ähnelt und somit einen einfachen Einstieg ermöglicht.

Frei erhältliche VoIP-Telefonanlagen für Linux wie PBX4Linux oder Asterisk bieten eine Vielzahl von Funktionen, die durchaus mit den Funktionen einer klassischen Telefonanlage mithalten können. Ein ISDN-Anschluss, ein Rechner mit einer ISDN-Karte und ein Netzwerk-Anschluss reichen aus, um mit der Software einen Telefon-Gateway zu realisieren, der Gespräche vom klassischen Telefonnetz zu VoIP-Clienten weiterleitet und umgekehrt. Asterisk hat sich dabei als die bessere Lösung herausgestellt, vor allem da PBX4Linux dynamische IP-Adressen, die beispielsweise für den Einsatz mobiler Endgeräte gebraucht werden würde, leider nicht unterstützt. Auf die Themen Sicherheit und Verfügbarkeit der vorgestellten Systeme wurde in dieser Studienarbeit nicht eingegangen. Sie sollten jedoch vor jedem Einsatz einer VoIP-Telefonanlage genaustens durchdacht werden.

VoIP hat noch einen großen Vorteil gegenüber der klassischen Telefonie, auf den in dieser Ausarbeitung nicht eingegangen wurde und das ist die neue Dimension der Erreichbarkeit. Bei VoIP ist jeder Benutzer unabhängig vom benutzten Telefonie-Gerät und Ort, an dem er sich gerade befindet, über nur nur noch eine Adresse ständig erreichbar. So ist man bei der Arbeit am PC direkt über diesen erreichbar. Wenn man sich am PC abmeldet werden die Gespräche direkt an einen PDA weitergeleitet und nachdem man sich in ei-

nem anderem Gebäude oder Zimmer wieder an einem PC anmeldet, werden die Gespräche wieder an diesen geleitet. Es bleibt noch die Frage, ob man vorhandene andere Adressen und Nummern, wie beispielsweise die Email Adresse, Handy- und Festnetznummer ebenfalls über nur noch eine Adresse zusammenführen kann. Auch hierfür gibt es schon Lösungsansätze wie beispielsweise ENUM (tElephone NUmber Mapping), spezifiziert im RFC 2916 [1]. Bei ENUM registrieren Benutzer einfach eine Telefonnummer als Domain die alle Adressen und Nummern unter denen sie erreichbar sein möchten, enthält. So entsteht ein globales Verzeichnis, in dem alle Telefonnummern und Adressen über URIs verlinkt sind.

Die Zeichen für VoIP stehen also sehr gut und man liest täglich neue Nachrichten über grosse Unternehmen, die ihre Telefonsysteme auf VoIP umstellen möchten. In der Zukunft werden wahrscheinlich nach und nach alle klassischen Telefonsysteme durch VoIP abgelöst werden und irgendwann wird vielleicht die gesamte Kommunikation über nur noch ein einziges Netz - das Internet - geführt werden.

Literaturverzeichnis

[1] P. Faltstrom. RFC 2916 - E.164 number and DNS. `http://www.faqs.org/rfcs/rfc2916.html`, September 2000.

[2] Andreas Hitzig. Verbindung schaffen - Protokolle im Überblick: H.323 und SIP. *iX*, 9, 2004.

[3] James F. Kurose and Keith W. Ross. *Computer Networking*. Addison Wesley, third edition, 2005.

[4] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. RFC 3261 - SIP: Session Initiation Protocol. `http://www.faqs.org/rfcs/rfc3261.html`, June 2002.

[5] Stephan Rupp, Gerd Siegmund, and Wolfgang Lautenschlager. *SIP - Multimedial Dienste im Internet*. dpunkt.verlag, 2002.

[6] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RFC 1889 - RTP: A Transport Protocol for Real-Time Applications. `http://www.faqs.org/rfcs/rfc1889.html`, January 1996.

[7] John Todd. Asterisk: A Bare-Bones VoIP Example. `http://www.onlamp.com/pub/a/onlamp/2003/07/03/asterisk.html`, March 2003.

[8] Andreas Eversberg und Karsten Violka. Tux vermittelt - Linux als Telefonanlage mit VoIP. *c't*, 9, 2004.