

Fachhochschule Köln
University of Applied Sciences Cologne

07

Fakultät für Informations-, Medien- und
Elektrotechnik, Studiengang Elektrotechnik

Institut für Nachrichtentechnik
Fachgebiet Datennetze

Diplomarbeit

Thema: Entwicklung eines einfach konfigurierbaren
Telekommunikationssystems „VoIP on CD“

Student :	Rolf Winterscheidt
Matrikelnummer:	010 461794
Referent :	Prof. Dr. Andreas Grebe
Koreferent :	Prof. Dr. Uwe Dettmar
Abgabedatum :	31. Oktober 2005

Hiermit versichere ich, dass ich die Diplomarbeit selbstständig angefertigt und keine anderen als die angegebenen und bei Zitaten kenntlich gemachten Quellen und Hilfsmittel benutzt habe.

Rolf Winterscheidt

1	Einleitung und Aufgabenstellung	6
2	Grundlagen	7
2.1	VoIP im Gegensatz zur klassischen Telefonie	8
2.1.1	Variable Verwendung der Kanäle	8
2.1.2	Nomadische Nutzung	8
2.1.3	Ausfallsicherheit	8
2.1.4	Kosten gegenüber einer klassischen TK-Anlage	8
2.2	Mean Opinion Score (MOS)	8
2.3	Codecs	9
2.4	Bandbreitenbedarf / Datenvolumen	9
2.5	Übertragungsschwierigkeiten	10
2.5.1	Paketverluste	10
2.5.2	Latenz (Laufzeit)	10
2.5.3	Jitter	10
2.6	Protokolle im Vergleich: SIP und IAX2	10
2.6.1	SIP-Protokoll (Session Initiation Protocol)	11
2.6.2	IAX2-Protokoll (Inter Asterisk eXchange)	11
2.7	Abhörsicherheit von VoIP	12
2.8	Quality of Service (QoS)	12
3	Analyse und Bewertung alternativer VoIP-Systeme als Basistechnologie	12
3.1	SER (SIP Express Router)	12
3.2	Cisco Callmanager	13
3.3	Innovaphone VoIP-Systeme	13
3.4	Skype	13
3.5	Swyxware	13
3.6	Ahead Sippstar Telefonanlage	14
3.7	Asterisk	14
3.8	Auswahl eines geeigneten Systems	14
3.8.1	Gewichtung von OpenSource	15
3.8.2	Entscheidung SER oder Asterisk	15
4	Konzeption und Implementierung	15
4.1	Grundkonzept	15
4.1.1	Hardware	16
4.1.1.1	Nutzung von ISDN-Karten	17
4.1.2	Betriebssystem	17
4.1.3	Datenbank	17
4.1.4	Webserver	18
4.1.5	Asterisk	18
4.2	Installationsschritte	19
4.3	Automatisches Softwareupdate bei Installation	20
4.4	Konfiguration Debian-Grundsystem	20
4.4.1	Debian-CD remastern	20
4.4.1.1	Isolinux / Syslinux	20
4.4.1.2	Bootparameter	21
4.4.1.3	Voreinstellungsdatei (preseed)	21
4.4.1.4	Mkisofs	22
4.4.1.5	CD brennen	22

4.4.2	Installiertes System / Softwareinstallation	23
4.4.2.1	Installationskript voiponcd.sh	23
4.4.2.2	Kernelversion	23
4.4.2.3	Filesystem ext3	23
4.4.3	Implementierung MySQL	23
4.4.4	Implementierung Apache Webserver	23
4.5	Konfigurationsschritte Asterisk	23
4.5.1	Betriebsart	24
4.5.2	SIP-Konfiguration (sip.conf)	24
4.5.3	SIP-Datenbank (zu sip.conf)	25
4.5.4	IAX2-Konfiguration (iax.conf)	26
4.5.5	Dialplankonfiguration (extensions.conf)	26
4.5.6	Dialplan-Datenbank (zu extensions.conf)	27
4.5.7	extconfig.conf	28
4.5.8	res_mysql.conf	28
4.5.9	cdr_mysql.conf	28
4.5.10	Astcc-Skript (Prepaid)	28
4.5.10.1	sub load_config()	29
4.5.10.2	sub connect_db()	29
4.5.10.3	sub msleep()	29
4.5.10.4	sub savedata()	29
4.5.10.5	sub getcard()	29
4.5.10.6	sub savecdr()	30
4.5.10.7	sub mystreamfile()	30
4.5.10.8	sub mysaynumber()	30
4.5.10.9	sub tell_time()	30
4.5.10.10	sub saycost()	30
4.5.10.11	sub mysayfloat()	30
4.5.10.12	sub getphone()	30
4.5.10.13	sub trytrunk()	30
4.5.10.14	sub calccost()	30
4.5.10.15	sub checkinuse()	30
4.5.10.16	sub checkexpired()	31
4.5.10.17	sub setfirstuse()	31
4.5.10.18	sub setexpiration()	31
4.5.10.19	sub setinuse()	31
4.5.10.20	Hauptteil	31
4.6	Erstellung Weboberfläche	31
4.6.1	Cascading Stylesheets	33
4.6.2	Authentisierung durch Sessions	33
4.6.3	Administrationsoberfläche	33
4.6.3.1	Übersicht (index.php)	36
4.6.3.2	Userverwaltung (show_customer.php)	36
4.6.3.3	Anruflisten (all_calls.php)	36
4.6.3.4	Administratorenverwaltung (show_admins.php)	36
4.6.3.5	Rufnummernpool (numberpool.php)	37
4.6.3.6	Tarife und Routing (new_rates.php)	37
4.6.3.7	Tarifgruppen/Tarifmodelle (new_brands.php)	37
4.6.3.8	Carrierverwaltung / Providerverwaltung (new_carrier.php)	37
4.6.3.9	Systemkonfiguration (systemconfig.php)	37
4.6.4	Useroberfläche	37
4.6.4.1	Home (index.php)	38
4.6.4.2	Tarife (rates.php)	38
4.6.4.3	Mein Account (myaccount.php)	38
4.6.4.4	Telefonbuch (phonebook.php)	38
4.6.4.5	Anrufliste (callist.php)	38
4.6.4.6	X-Lite Anleitung (xlite.php)	38
4.7	Installation von VoIP on CD mit der fertiggestellten CD	39
5	Kosten	42

5.1	Anschaffungskosten Hardware und Einrichtung.....	42
5.2	Betriebskosten.....	42
6	Ergebnis / Fazit	43
6.1	SOHO-System (1-Server-System ohne Satelliten).....	43
6.2	Business-System (Mehrserver-System mit Satelliten).....	44
6.3	Provider-System (Anbieter von VoIP-Services)	45
6.4	Weltweit verteiltes System	45
7	To Do-Liste.....	46
8	Quellenverzeichnis.....	46

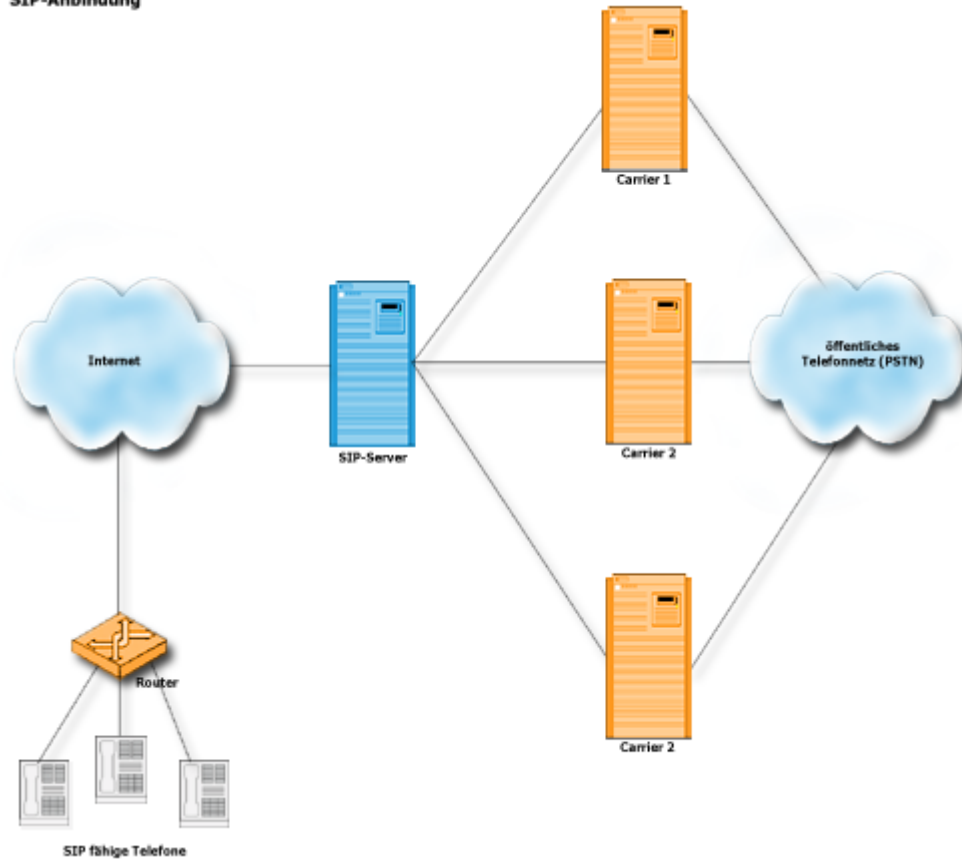
1 Einleitung und Aufgabenstellung

Herkömmliche Telefonanlagen basieren meist auf proprietärer Technik. Das betrifft die Hardwareseite, wie aber auch die Software. Die im Zuge immer weiterschreitender, flächendeckender Verteilung von Breitbandanschlüssen zunehmende Voice over IP-Technik schafft Möglichkeiten, die die typischen Büro-Telefonanlagen überfordern. Ein neues Konzept baut auf eine OpenSource-Technik auf, die in Verbindung mit Standard-PC-Hardware ein leistungsfähiges Gesamtsystem ergibt, das neben der IP-Telefonie auch eine Anbindung an die klassische Telefonie erlaubt. Das Telefonieren über Datenleitungen ist nicht neu, denn z.B. große Firmen nutzen diese Technik bereits seit längerer Zeit, um Verbindungen zwischen Satelliten (Zweigstellen) und dem Stammhaus herzustellen. Auch große Carrier wie die DTAG nutzen die IP-Telefonie (meist noch mit dem H.323-Protokoll), um kostengünstige Verbindungen zwischen weit entfernten Hauptvermittlungsstellen (HVstn) zu realisieren.

Es gibt viele Wege, um an der IP-Telefonie zu partizipieren. Einige sind sehr kostenintensiv, andere sehr umständlich und aufwändig einzurichten und wiederum andere geben nicht die notwendige Flexibilität, die sich ein anspruchsvoller Nutzer wünscht. Eine zufriedenstellende Lösung, die einfach konfigurierbar, d.h. ohne erst eine Skriptsprache oder dergleichen lernen zu müssen, und darüber hinaus noch kostengünstig realisierbar ist, existiert offenbar nicht. Da eine solche Lösung aber von vielen benötigt wird, erscheint es sinnvoll, die beste frei erhältliche Lösung als Basis für eine eigene Weiterentwicklung zu nutzen. Um dabei möglichst viele Anwendungsgebiete abdecken zu können, macht es Sinn, diese Software unter vielen Anwendungsaspekten zu sehen, d.h. im privaten Umfeld, um mehrere IP-Telefone über einen Account an einen Provider anschließen zu können oder im Büro, um dort eine kleine Firma mit deren IP-Telefonen an den Provider anzuschließen und evtl. auch Zweigstellen (Satelliten) damit anbinden zu können. Eine Art Königsdisziplin ist hierbei der Versuch, selbst einen Provider nachzubilden, also angeschlossene Firmen und Privatpersonen in gewissen Grenzen auch abrechnen zu können. Legt man die Messlatte auf die höchste Stufe, also eine Softwaresuite zu entwerfen, die es mit wenigen Handgriffen möglich macht, selbst zum Provider zu werden, sind häufig auch die einfacheren Aufgaben fast automatisch gelöst. Eine Firma mit einigen Telefonen ist schließlich auch in einer etwas anderen Art und Weise ein Provider, wenngleich auch die einzelnen Teilnehmer keinen Vertrag mit dem „Provider“ eingehen und die Gespräche auch nicht auf Minutenbasis abgerechnet werden. Sollte also ein Weg gefunden werden, ein Providersystem zu entwerfen, könnten davon auch fast alle anderen IP-Telefonierer in spe profitieren. Das System wäre also gedanklich übertragbar.

Ein Beispiel einer möglichen Platzierung eines als Providersystem eingesetzten SIP-Servers (in blau dargestellt) ist nachfolgender Grafik zu entnehmen. IP-Telefone von Kunden könnten sich mit einem bestehenden DSL-Anschluss über das Internet am SIP-Server (dem Registrar) anmelden. Kommende bzw. gehende Gespräche aus bzw. in das Festnetz würden mittels der Carrieranbindungen (z.B. an DTAG, Telefonica, Versatel, Netcologne, Colt, 3u usw.) realisiert. Reine IP-Telefonate könnte der Server selbst handeln. In einer Firma ließe sich ein System als zentraler SIP-Server einsetzen, an die sich alle Unternehmenstelefone anmelden. Der Server selbst benötigte dann über einen Provider nur einen einzigen Account (mit Einschränkungen), da er selbst das Management über die SIP-Gespräche führen könnte. Auch wäre die Firewall-Einstellung deutlich einfacher zu handhaben, da das System quasi als eine Art SIP-Proxy agieren würde (streng genommen ist ein SIP-Proxy jedoch etwas weniger komplex und dient mehr der reinen Weiterleitung).

SIP-Anbindung



Grafik 1.1: Einsatz von „VoIP on CD“ als Providersystem

Um der Einfachheit eines solchen Systems wirklich gerecht zu werden, würde es kaum Voraussetzungen erwarten, am besten auch kein installiertes Betriebssystem. Eine CD, die auf einer frisch formatierten Festplatte völlig selbstständig alles Notwendige inkl. Betriebssystem installieren würde, wäre eine optimale Lösung. Ein solches SIP-System auf CD, dem die Bezeichnung „VoIP on CD“ gegeben wurde, soll hier entwickelt werden. Technische Laien sollen in die Lage versetzt werden, vernetzte Telefonanlagen für kleine und mittelständische Unternehmen zu installieren. Die Mindestanforderungen umfassen folgende Punkte:

- flexibel einsetzbar
- in weiten Teilen frei konfigurierbar
- Möglichkeit der Fernwartung
- vergleichsweise kostengünstig
- stabil
- integrierte Mailbox
- Weboberfläche zum Verwalten der Benutzer etc.
- LCR (Anbindung mehrerer Carrier zum PSTN)

Nach Analyse der verfügbaren technischen und wirtschaftlichen Alternativen soll das für diesen Zweck beste vorhandene System ausgewählt und weiterentwickelt werden. Hierbei soll eine Oberfläche geschaffen werden, die einfach zu bedienen ist und hierzu keine tiefgehende Computererfahrung oder gar Programmierkenntnisse voraussetzt. Im Folgenden werden die wesentlichen Eigenschaften dieses Systems beschrieben, ohne die Grundlagen der Voice over IP-Technik und der Netzwerktechnik zu wiederholen.

2 Grundlagen

Voice over IP, kurz VoIP, baut auf einer völlig anderen Struktur auf als die herkömmliche Telefonie. Während das Protokoll H.323 von der „Telefonwelt“ kommt und die „Datenwelt“ einbezieht, so ist es z.B. bei SIP oder IAX2 genau anders herum. Letztere Protokolle sind deutlich besser für die typischen Schwierigkeiten in Datennetzen gerüstet, wie z.B. NAT (was bei der Umsetzung von lokalen Adressräumen auf eine einzige öffentliche IP-Adresse korrekterweise eher mit PAT, also der Port Address Translation,

bezeichnet wird). H.323 konnte sich in Netzen etablieren, die möglichst ohne zwischengeschaltete Firewalls oder NAT/PAT auskommen. Doch im Zuge der großen Verbreitung von Breitbandanschlüssen kommt gerade NAT eine wichtige Bedeutung zu. Somit wird SIP und neuerdings auch IAX2 in der IP-Telefonie über Breitbandanschlüsse immer populärer.

2.1 VoIP im Gegensatz zur klassischen Telefonie

Eine herkömmliche TK-Anlage bedient Endgeräte, die am selben Standort genutzt werden. Ein Teilnehmer, der z.B. in einem Homeoffice arbeitet, kann Gespräche für ihn meist nur über eine (kostenpflichtige) Rufumleitung auf seinen Heim-Anschluss annehmen. Eine Steuerung der Rufumleitung ist i.d.R. nicht vom Homeoffice aus möglich. Ebenso verhält es sich mit der Abrechnung und Auswertung getätigter abgehender Anrufe. Diese werden dem Anschluss des Homeoffice belastet, eine Auswertung (wichtig z.B. für Callcenter) ist hier nur schlecht möglich. Allein an diesem Beispiel wird deutlich, dass Bedarf an neuartigen Lösungswegen besteht, die es auch einem Endanwender ermöglichen, sich in der Telekommunikationswelt frei zu entfalten.

2.1.1 Variable Verwendung der Kanäle

Eine TK-Anlage mit 2 (S0) oder auch 30 Kanälen (S2m) wird keine weiteren Gespräche ins Festnetz terminieren können, wenn die vorher festgelegte Anzahl an gleichzeitigen Gesprächen überschritten wird. Da die Struktur bei VoIP nicht auf dedizierte Kanäle ausgerichtet ist, sondern ein gemeinsames Medium nutzt kann hier bei Verwendung eines stärker komprimierenden Codecs (z.B. GSM statt G.711) unter Verlust der hohen Sprachqualität weiterer Raum für Gespräche geschaffen werden. Das bietet sich bei kurzzeitigen Engpässen an.

2.1.2 Nomadische Nutzung

Durch Voice over IP ist es erstmals auch für Endanwender möglich, den Standort eines Telefonanschlusses aus dem Büro kurzfristig zu ändern. Ein Geschäftsreisender telefoniert vom Hotel aus mit einem Internetzugang über die TK-Anlage (VoIP) des heimischen Büros mit seiner eigenen Durchwahl (Rufnummernübermittlung) und ist auch darüber erreichbar. Auch die zugehörige Callstatistik ist, da in der Tat über die eigene TK-Anlage telefoniert wurde, vorhanden. Ein Callcenter benötigt somit nicht zwingend eigene Flächen, sondern nur noch eine VoIP-TK-Anlage und Heimarbeitsplätze, die mit einer DSL-Anbindung versehen werden.

2.1.3 Ausfallsicherheit

Die klassische Telefonie ist als sehr stabil anzusehen. Durch den langen Reifeprozess konnten die Hersteller von TK-Equipment viele Detailverbesserungen einfließen lassen. Doch auch in der IT, auf der die VoIP-Anlagen basieren, ist Ausfallsicherheit kein Fremdwort. Gerade durch die Digitalisierung ist es möglich, redundante Anlagen wie auch Leitungen bereitzustellen, die der klassischen Telefonie durchaus ebenbürtig, wenn nicht sogar überlegen sind. Im Gegensatz zu einem herkömmlichen analogen Telefonanschluss, bei dem keine eigene Stromversorgung notwendig ist und sich die Ausfallmöglichkeit auf die Hardware, die Leitung und den Provider aufteilt, existieren in der IP-Telefonie mehrere Komponenten, die zu einem Ausfall beitragen können. Dabei ist die reine Internetanbindung an sich erfahrungsgemäß ebenso stabil, wie ein Telefonanschluss. Die bei VoIP größere Anzahl an Komponenten, die einen Ausfall herbeiführen können, wie z.B. Router, Stromausfall, Software im Telefon, ergeben dabei auch eine höhere Ausfallwahrscheinlichkeit, die sich jedoch in der Praxis bislang nicht zeigt.

2.1.4 Kosten gegenüber einer klassischen TK-Anlage

Intelligente TK-Anlagen für 100 Nutzer können, je nach Verwendungszweck mehr als EUR 100000 kosten (Erfahrungswert in einem Callcenter). Durch die VoIP-Technik reicht für diese Nutzerzahl ein handelsüblicher PC aus. Bei der Installation eines solchen Systems werden zur Sicherheit meist zwei PCs redundant ausgelegt. Da in vielen Fällen ohnehin ein lokales Netzwerk verwendet wird, ist es möglich, einige Teile der Infrastruktur mit zu nutzen, d.h. bei einer Neuinstallation müssen lediglich Netzkabel verlegt werden, über die auch die Telefonie stattfindet. Eine separate Verlegung von Telefonkabeln ist somit nicht notwendig.

2.2 Mean Opinion Score (MOS)

Einem Menschen, der telefoniert, ist es normalerweise ziemlich egal, mit welchen technischen Raffinessen sein Gespräch ermöglicht wurde. Er möchte einfach nur telefonieren und somit ordnet er die Gesprächsqualität keinen technischen Kriterien unter. Genau so wurde damals die MOS, was frei und

sinngemäß übersetzt etwa „Durchschnittliche Beurteilung“ heißt, auch gebildet [mos]: Eine Gruppe von ausgewählten Männern und Frauen hörte sich bestimmte Gespräche an und vergab Noten über die Gesprächsqualität, die dann über die Gruppe je Gespräch gemittelt wurde. Allem Anschein nach ist dieses Verfahren dem im Amateurfunk üblichen R-Wert (Radio-Wert) gleichzusetzen, der das gleiche Ziel verfolgt. Eine Skala von 1 bis 5 gibt dem Gegenüber den subjektiven Eindruck der hörbaren Übertragungsqualität wider, dabei steigt die Qualität mit dem Wert. Heutige Computersysteme können unter Berücksichtigung von bestimmten Parametern den MOS-Wert auch automatisch bestimmen. Ein Maximalwert von ca. 4,1 bis 4,3 lässt sich mit dem G.711-Codec erzielen, ein komprimierender Codec wie G.729 oder GSM liegt unter dem wichtigen Wert von 4, der wie eine Schranke zwischen „Mobilfunk-Qualität“ und „ISDN-Qualität“ zu sehen ist. Von einer hohen Qualität spricht man also nur bei Werten über 4. Ein Wert von 5 ist bei Übertragungsstrecken nicht zu erreichen. Die untere Grenze ist erreicht, wenn das Gegenüber gerade noch hörbar ist, ein Informationsaustausch ist hier nur noch sehr begrenzt möglich. Folgenden Qualitäten können die entsprechenden Werte so zugeordnet werden:

Wert	Qualität
1	mangelhaft
2	mäßig
3	ordentlich
4	gut
5	exzellent

Tabelle 2.1: MOS-Wert zur Bestimmung der Gesprächsqualität

2.3 Codecs

Die Sprachqualität verhält sich proportional zur benötigten Bandbreite, d.h. eine höhere Sprachqualität benötigt in aller Regel eine höhere Bandbreite, was nicht zuletzt auch höhere Kosten mit sich bringt. Folgende Tabelle nach [codecs] zeigt die gängigsten Codecs in Verbindung mit der Bitrate, wobei für die Übertragung die Rate auf dem Ethernet ausschlaggebend ist, d.h. die Bitrate des Codecs zzgl. Overhead wie TCP/IP-Header usw.

Codec	Bitrate Codec [kbit/s]	Bitrate Ethernet [kbit/s]
G.711 (frei)	64	88
G.722	64	88
G.723	6,4	22
G.726-32	32	55
G.728	16	32
G.729	8	32
ILBC (frei)	13,3	27
GSM	13	35
Speex (frei)	2 bis 44	variabel

Tabelle 2.2: Bitraten gängiger Codecs im Vergleich

2.4 Bandbreitenbedarf / Datenvolumen

Die Kommunikation wird meistens mit dem G.711-Codec stattfinden. Dieser hat die höchste Qualität und benötigt somit auch mit über 80 kbit/s (Brutto) die größte Bandbreite. Heutige Internetanschlüsse lassen selbst für den Privatkunden noch teilweise mehr als 5 Gespräche gleichzeitig zu (z.B. Netcologne mit über 600 kbit/s Upstream). Ein Gespräch von einer Stunde produziert bei Annahme von 88 kbit/s je Richtung eine Datenmenge von knapp 39 MB, also insgesamt rund 78 MB. Bei durchschnittlich täglich einer Stunde Telefonie ergibt sich somit ein monatliches Telefonievolumen von etwa 2,3 GB, womit für die Internettelefonie nicht zwingend eine Flatrate vorausgesetzt werden muss, sondern durchaus auch Volumentarife hier nutzbar sind.

Aus Providersicht stellen selbst 100 gleichzeitige Gespräche mit insgesamt rund 17 Mbit/s keine große Hürde dar. Da eine Software-TK-Anlage bei G.711 keinen Komprimierungsaufwand hat, ist auch die CPU-Last in diesem Fall im Gegensatz zu den komprimierten Codecs nahezu vernachlässigbar.

2.5 Übertragungsschwierigkeiten

Neben der unterschiedlichen Gesprächsqualität der Codecs und der beschriebenen Ausfallsicherheit existieren weitere Faktoren, die die Gesprächsgüte beeinflussen. Diese sind nachfolgend aufgeführt.

2.5.1 Paketverluste

Da es bei den Sprachdaten nicht darauf ankommt, dass alle Daten den Empfänger erreichen, sondern mehr, wie schnell sie dort ankommen, werden diese per UDP verschickt. Eine Wiederholung, wie es bei TCP vorkommen würde, wird in einem Sprachstrom ohnehin keine positive Auswirkung haben. Kommen die Pakete nicht in der richtigen Reihenfolge beim Empfänger an (ggfs. durch verschiedene Routen), werden diese verworfen und zählen damit auch zu den Paketverlusten, obwohl sie das Ziel erreicht haben. Je nach Codec ist ein Verlust von bis zu 10% der Pakete (iLBC) noch ohne größere Kommunikationsstörung hinnehmbar. In der Regel werden die Paketverluste im Internet weit darunter liegen. In lokalen Netzen sind die Verluste, sofern die Netze nicht ausgelastet sind, vernachlässigbar.

2.5.2 Latenz (Laufzeit)

Ein Paket, das von einer Quelle zu einem Ziel transportiert wird, kommt nicht ohne eine Verzögerung (Laufzeit) dort an. Viele Faktoren, wie z.B. die Bandbreite, Netzwerkauslastung, Router auf dem Weg zum Ziel, verzögern das Paket immer mehr. Mittels eines Traceroutes lassen sich Verzögerungen von Hop zu Hop anschaulich darstellen. Die maximale Latenz sollte ungefähr 150 ms nicht überschreiten, da sonst die entstehende hörbare Verzögerung zwischen den sprechenden Teilnehmern als störend empfunden wird [itu].

2.5.3 Jitter

Die zeitliche Schwankung zwischen dem Empfang von Datenpaketen ist als Jitter definiert. Um diese Schwankung wieder kompensieren zu können, werden Buffer verwendet. Durch das Zurückhalten von Paketen wird jedoch die Laufzeit erhöht, so dass die Größe des Buffers möglichst klein gehalten wird. Folgende Grafik veranschaulicht dies. Hierbei zeigt die gestrichelte Linie den direkten Fluss der Sprachpakete und der Pfad mit der durchgehenden Linie den „Umweg“ über den Buffer.

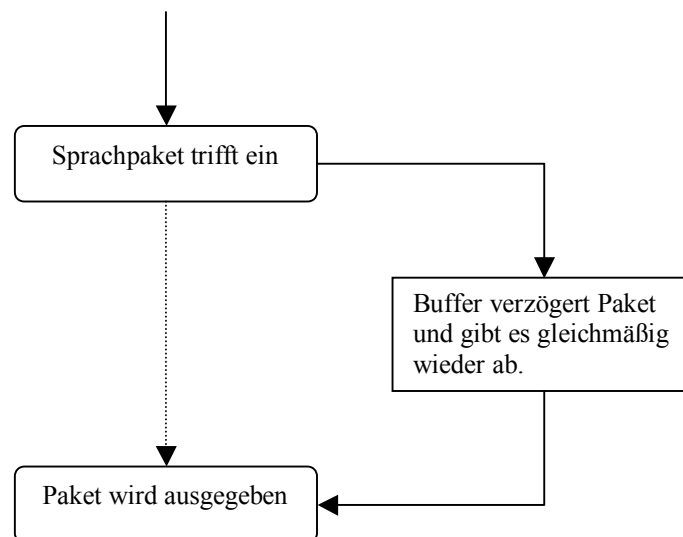


Diagramm 2.1: Buffer zur Kompensierung von Jitter

2.6 Protokolle im Vergleich: SIP und IAX2

Da H.323 in der Internet-Telefonie eine untergeordnete Rolle spielt, lassen sich nur noch das standardisierte SIP und das von der Community entwickelte IAX in der Version 2 sinnvoll vergleichen [IAXvsSIP]. Während SIP sich auf breiter Basis durchgesetzt hat und in sehr vielen Geräten implementiert ist, holt IAX2 in der IP-Telefonie wegen der überzeugenderen Struktur auf. Grundsätzlich ist hier vom Protokoll her sicherlich IAX2 eine bessere Wahl, doch ist SIP sehr viel mehr verbreitet und kann als Basis für mehrere Dienste wie einem Messenger oder Videoübertragung herangezogen werden. Somit trumpft IAX2 nur in der

IP-Telefonie auf. Fast alle IP-Telefone und Adapter sind SIP-fähig, doch es befinden sich derzeit kaum IAX2-kompatible Geräte auf dem Markt.

	SIP	IAX2
Firewall- /NAT-Freundlichkeit	Relativ gut: 1 Steuer-, 1 RTP-Port je Gesprächskanal, also min. 3 pro Telefonat.	Sehr gut: Immer nur 1 Port, inkl. Steuerung und Gesprächskanäle.
Brutto-Bandbreitenrelation	Mit 1 Steuerkanal und je 1 RTP-Kanal mäßige Bandbreitenausnutzung.	Sehr gute Bandbreitenausnutzung, da alles in einem Kanal geführt wird. Mit zunehmender Anzahl an Gesprächskanälen noch besseres Verhältnis.
Flexibilität der übertragenen Datentypen	Sehr hoch, da SIP nur die Sessions verwaltet, nicht aber den eigentlichen Inhalt (außer Messenger, dort schon).	Schlechte Flexibilität, da sehr auf Sprach-Telefonie ausgelegt. Videoübertragung auch begrenzt möglich, jedoch sonst nichts.
Verfügbar in Endgeräten	Fast überall.	Kaum, da kein offizieller Standard, stark auf die Software-PBX Asterisk fixiert.

Tabelle 2.3: Vergleich SIP zu IAX2

2.6.1 SIP-Protokoll (Session Initiation Protocol)

Das SIP-Protokoll, seit 1996 von der IETF entwickelt und erstmals 1999 in der RFC2543 und in der aktualisierten Version in 2002 als RFC3261 (Anhänge bis RFC3265) veröffentlicht, hat der IP-Telefonie einen großen Schub gegeben. Während bei H.323 nur mit Aufwand Firewall-Regeln erstellt werden konnten, benötigt SIP lediglich Port 5060 (meist UDP) und mindestens einen RTP-Port (UDP). Damit wurde der Weg zur Internettelefonie auch für DSL-Nutzer frei. Dabei ist das SIP-Protokoll deutlich vielseitiger konzipiert als nur zur Telefonie. Wie der Name schon andeutet, werden mit SIP Sessions verwaltet. Dabei bedeutet Session in diesem Fall eine Sitzung (Verbindung) zwischen Teilnehmern. Hierbei kann das Protokoll nicht nur Sprache, sondern auch Bilder, Video oder Textnachrichten transportieren. Die ersten Messenger (Programme für Kurznachrichten zwischen Netzwerknutzern), die dieses Protokoll verwenden, existieren bereits [sipmess]. SIP benötigt eine Client/Server-Architektur, d.h. der User Agent (UA) registriert sich an einem SIP-Proxy Server (Registrar). An diesem Server hat der Client, ähnlich wie bei Mails, unabhängig vom tatsächlichen Aufenthaltsort im Netz, immer eine feste Adresse wie z.B. 2001@sip.rowi.net. Ein SIP-Telefon kann ebenso im heimischen Büro verbunden sein wie in einem Hotel in Tokio. Ein Anruf an diese Adresse wird somit immer zu diesem Telefon führen, sofern es eine Verbindung zum Internet hat und sich so am SIP-Proxy anmelden kann.

Um Sprache übertragen zu können, reicht das SIP-Protokoll allein nicht aus. Da SIP nur die Sitzung managed, d.h. z.B. eine Rufsignalisierung vornimmt, den Anfang und das Ende des Gesprächs kennzeichnet und die Registrierung am SIP-Proxy vornimmt, wird es für einen Messenger zwar alles Notwendige bereithalten, doch fehlt hierbei die Möglichkeit für die Übertragung der Sprache. Somit wird SIP meistens (Ausnahme: Messenger) ergänzt durch andere Protokolle wie dem Realtime Transport Protocol (RTP, RFC1889), um Echtzeitübertragung inkl. Quality of Service (QoS) zu ermöglichen, dem Realtime Streaming Protocol (RTSP, RFC2326), um Streams wie Livemusik zu kontrollieren oder dem Session Description Protokoll (SDP, RFC2327), um Multimedia-Sitzungen zu beschreiben.

2.6.2 IAX2-Protokoll (Inter Asterisk eXchange)

Im Gegensatz zu SIP ist IAX2 nicht in einem RFC definiert, sondern basiert auf einer gemeinsamen Arbeit der Community um die Softwaretelefonanlage Asterisk [IAX]. Die Abkürzung IAX bedeutet Inter-Asterisk-Exchange und steht folglich für die Verbindung von Asterisk-Systemen untereinander. Während bei SIP noch jeweils ein separater Port für die Sprache notwendig ist, wird bei IAX2, dem erweiterten Protokoll zu IAX, nur ein einziger UDP Port (4569) für Daten und Sprache benötigt. Der Einsatz bei privaten Netzen hinter einem Gateway (NAT bzw. eher PAT) wird deutlich erleichtert. Das bei SIP bekannte Problem, dass die Sprachströme (RTP) nicht durchgeleitet werden, verschwinden hier. Auch ist ein STUN-Server nicht mehr notwendig, der bei SIP dem User Agent die öffentliche IP-Adresse mitteilt. Auf das RTP-Protokoll wird auch verzichtet, da durch die Verwendung nur eines Ports sämtliche Informationen wie Steuerkommandos und Sprachdaten in einem Paket vorhanden sein müssen. Das IAX2-Protokoll erhebt nicht den Anspruch, so flexibel einsetzbar zu sein wie SIP. Videodaten können zwar auch hiermit versendet werden, doch liegt das

Hauptaugenmerk auf der Übermittlung von Sprache. Durch die Vorzüge bei NAT/PAT eignet sich das Protokoll auch für den Endnutzer mit einem Telefon. Einige Hersteller haben es daher schon in Telefone integriert. Der größte Vorteil liegt aber darin, dass auch mehrere unabhängige Gespräche über den gleichen, einzigen, Port geschickt werden können. Während bei SIP für jeden Gesprächskanal ein neuer RTP-Port geöffnet wird, bleibt bei IAX2 alles auf einem Port, was auch firewallseitig sehr angenehm ist. Durch das Zusammenfügen sämtlicher Informationen ist auch die Gesamtbandbreite im Gegensatz zu SIP bei einer größeren Anzahl an Gesprächskanälen kleiner.

2.7 Abhörsicherheit von VoIP

Mit RFC3711 existiert ein Dokument für SRTP (Secure RTP), das als Basis für einen verschlüsselten Sprachstrom dienen kann. Auf Sourceforge.net wird bereits eine Library dazu entwickelt [SRTP]. Zusammen mit SIPS, einem Synonym für SIP over SSL, mit dem sich der Protokollstrom verschlüsseln lässt, kommen auch einige Telefone wie das Snom 360 oder das Sipura SPA-841 auf den Markt, die diese Technik in die Praxis umsetzen. Da die Ver- und Entschlüsselung auf der Providerseite deutlich mehr Rechenleistung benötigt als die bislang unverschlüsselte Variante, können dort auf einem System deutlich weniger Calls pro Server gehandhabt werden. Daher werden die sicheren Verfahren von Providern nur zögernd umgesetzt. Sofern zwischen den VoIP on CD-Systemen VPNs bestehen, ist eine Abhörsicherheit auf der Durchleitungsstrecke zwischen den Netzen ohnehin gegeben. Zu beachten ist jedoch grundsätzlich, dass durch die Verschlüsselungsarbeit die Laufzeit der RTP-Pakete zunimmt und es so bei hoher Last zu hörbaren Verzögerungen kommen kann.

2.8 Quality of Service (QoS)

Die Type of Service (ToS) basierte QoS hält mittlerweile auch in vielen Homeoffice-Routern wie auch IP-Telefonen und Adaptern Einzug. Somit stellt das lokale Netz bei Verwendung neuerer Adapter selbst bei schmalbandigeren DSL-Anbindungen kaum eine Hürde für die Sprachpakete dar. Meist sind nicht einmal weitere Einstellungen notwendig, weil die Telefone in ihren Paketen das ToS-Feld auf "Reliability" oder "Throughput" setzen und die Router dieses auswerten. Da im Internet die Abschnitte nur streckenweise verwaltet werden, kann keine Garantie dafür gegeben werden, dass der ToS überall berücksichtigt wird. Vielmehr ist derzeit noch davon auszugehen, dass dies nur im eigenen LAN geschieht, wenn überhaupt.

3 Analyse und Bewertung alternativer VoIP-Systeme als Basistechnologie

Es existieren viele verschiedene Systeme, die auf Computertechnik basieren und um die Gunst der Nutzer buhlen. Teilweise entwickeln sich aus einer Basistechnologie heraus unterschiedliche Zweige von Systemarten, die für die ein oder andere Umgebung optimiert sind. In diesem Fall liegt das Hauptaugenmerk auf die im Punkt Aufgabenstellung beschriebenen Eigenschaften. Eine gewisse Flexibilität bieten die meisten Systeme an, doch offenbart sich meist erst auf den zweiten Blick, welches System auch für die angedachten Funktionen frei erweiterbar ist. Im Folgenden werden die wichtigsten derzeit verfügbaren flexiblen Telefonie-Systeme verglichen.

3.1 SER (SIP Express Router)

Der SIP Express Router, kurz SER genannt, entstammt dem Projekt iptel.org, das vom Fraunhofer Institut initiiert wurde. Erstellt für das Routing von sehr vielen gleichzeitigen Calls ist der Kern des Programms bewusst sehr schmal gehalten. Der Fokus von SER liegt deutlich auf der Geschwindigkeit des Routings und weniger in der Menge der Features. Obwohl SER über die Konfigurationsdatei ser.cfg viele Gestaltungsmöglichkeiten lässt, ist dieses Programm eher dafür geeignet, bei sehr großen Callvolumina den Prozessor nur wenig zu beanspruchen. Tiefergehende Features wie z.B. eine Voicebox sind mit diesem System nicht realisierbar. Da der ursprüngliche SER kaum weiterentwickelt wurde, entstand das Projekt OpenSER.

Fazit:

- Auf handelsüblicher PC-Hardware installierbar
- Verwendet ausschließlich SIP-Standard
- Reines SIP-Routing-System, kaum weitere Funktionen
- OpenSource
- Einfach per ser.cfg anpassbar
- Entwicklergemeinde im Internet ansprechbar
- Ungeeignet als Voicemail-System

- Sehr schnelles Routing der Daten, sehr geringe Prozessorlast
- Bewältigt großes Volumen an gleichzeitigen Verbindungen

3.2 Cisco Callmanager

Als Vorreiter in Sachen IP-Telefonie und durch langjährige Erfahrung im Netzwerkbereich hat Cisco schon recht früh ein System erstellt, das sich mittlerweile im Bereich der professionellen Telekommunikation etabliert hat. Da der Callmanager im IOS eines Cisco-Routers verankert wird, ist ein Betrieb auf einem herkömmlichen PC-System nicht möglich. Die Lizenzkosten betragen je nach Ausrüstung mehrere tausend Euro. Durch den Closed Source Charakter der Software sind Änderungen nur im vom Hersteller angedachten Bereich möglich, zudem schränkt der Router im Gegensatz zu einem PC-System die Möglichkeiten deutlich ein, da sich hier nur begrenzt Erweiterungskarten einsetzen lassen. Bei einem PC-System ist es dagegen möglich, z.B. Hardwareerweiterungen über USB oder die internen PCI-Steckplätze (oder ISA sofern noch vorhanden) zu integrieren.

Fazit:

- Benötigt als Plattform einen Cisco-Router (ab 1700er Serie)
- Kostenintensiv
- Kein OpenSource-System
- Nur aufwändig an eigene Wünsche anpassbar

3.3 Innovaphone VoIP-Systeme

Das VoIP-System von Innovaphone basiert vollständig auf dem älteren H.323-Standard, was ein Durchleiten durch Firewalls im Gegensatz zu SIP oder IAX deutlich behindert. Das Closed Source System, sowie die höheren Anschaffungskosten, schaffen kaum Spielraum für eigene Anpassungen.

Fazit:

- Verwendet nur den älteren H.323-Standard
- Kostenintensiv
- Kein OpenSource-System
- Nur aufwändig an eigene Wünsche anpassbar

3.4 Skype

Da Skype komplett auf einem proprietären Protokoll basiert, existieren zu der hauseigenen Software keine weiteren Clients (insbesondere Hardwaretelefone), mit denen eine Telefonie möglich wäre. Ein Routing in das Festnetz geschieht nur über Skype selbst. Eine IP-Telefonanlage würde sich mit Skype allein schon nicht realisieren lassen, weil sich sämtliche Clients an den betreibereigenen Servern anmelden müssen.

Fazit:

- Proprietäres System
- Keine IP-Telefone nutzbar, nur Softclients
- Kostenlose interne Telefonie, jedoch nur im Skype-Netzwerk
- Festnetztelefonie nur über Skype, Alternativcarrier nicht möglich

3.5 Swyxware

Aufbauend auf einen Microsoft Windows Server und als Closed Source konzipiert, basiert auch Swyxware auf dem älteren H.323 Standard. Eine Erweiterung ist auch hier nur in den vom Hersteller gesteckten Grenzen möglich. Bei möglichen Erweiterungen wie z.B. zusätzliche Teilnehmer, sind weitere Lizenzgebühren zu entrichten.

Fazit:

- Proprietäres System
- Verwendet nur den älteren H.323-Standard
- Nur IP-Telefone von Swyx nutzbar
- Kostenintensiv
- Kein OpenSource-System
- Max. 30 Kanäle gleichzeitig nutzbar
- Keine Anbindung an VoIP-Carrier vorgesehen

3.6 Ahead Sippstar Telefonanlage

Der Softwarehersteller Nero hat mit Sippstar eine Telefonanlage entwickelt, die das SIP-Protokoll zur Übertragung nutzt und auch Zweigstellen (Satelliten) bedienen kann. Als kommerzielles Produkt mit einem Preis von bis zu EUR 3000 ist es ebenfalls als Closed Source konzipiert, was tiefgreifende Änderungen vom Hersteller abhängig macht.

Fazit:

- Relativ kostengünstig
- Setzt auf den SIP-Standard
- Kein OpenSource
- Hohe Systemvoraussetzungen (Pentium4, 2GHz)
- Keine eigenen Erweiterungen möglich

3.7 Asterisk

Das von Mark Spencer entwickelte System ist OpenSource und unterstützt SIP, IAX2, H.323 und weitere Protokolle. Es läuft auf handelsüblichen PC-Systemen und ist frei erweiterbar. Eine ständig wachsende Gemeinschaft von Entwicklern erweitert das Grundsystem stetig. Eine freie Programmierung durch Verwendung von Datenbanken (Realtime Modus) und beliebige Programmiersprachen ist möglich. Per IAX2 oder SIP lassen sich Satelliten in ein Hauptsystem einbinden oder eine zweigstellenübergreifende Kommunikation herstellen.

Fazit:

- Auf handelsüblicher PC-Hardware installierbar
- Verwendet als Protokoll SIP, IAX2 und weitere Protokolle
- Viele Einsatzmöglichkeiten wie Voicemail, LCR, ACD uvm.
- OpenSource
- Relativ einfach anpassbar
- Entwicklergemeinde im Internet ansprechbar
- Bei Verwendung nur eines Codecs relativ geringe Prozessorlast

3.8 Auswahl eines geeigneten Systems

Die Auswahl eines geeigneten Grundsystems für die spätere Weiterentwicklung richtet sich nach verschiedenen Faktoren. Sofern ein Standard-PC genutzt werden kann, sind Erweiterungen durch Speicher-upgrades, eine größere Festplatte, einen schnelleren Prozessor usw. möglich. Durch die immense Verbreitung von SIP kann kaum ein anderes Protokoll genutzt werden, wenn die Kompatibilität sichergestellt werden soll. Auch die freie Erweiterbarkeit und die bereits implementierten Features sind ein wichtiger Entscheidungsgrund. Nicht zuletzt ist auch die Stabilität ein äußerst wichtiges Kriterium. Um die verschiedenen Entscheidungsfaktoren der Systeme gegenüberzustellen, sind diese nachfolgend tabellarisch aufgeführt:

	Standard-PC-Hardware	OpenSource	SIP-Protokoll	Komplette PBX inkl. Voicemail etc.	Erweiterbar (Programmierung)
SER (Sip Express Router)	Ja	Ja	Ja	Nein, nur rudimentär	Ja
Cisco Callmanager	Nein, Cisco	Nein	Ja	Ja	Nein
Innovaphone VoIP-Systeme	Ja	Nein	Ja	Ja	Nein
Skype	Ja	Nein	Nein, proprietär	Nein	Nein
Swyxware	Ja	Nein	Ja	Ja	Nein
Ahead Sippstar	Ja	Nein	Ja	Ja	Nein
Asterisk	Ja	Ja	Ja	Ja	Ja

Tabelle 3.1: Vergleich gängiger VoIP-Systeme

3.8.1 Gewichtung von OpenSource

Um auch langfristig eigene Entwicklungen in die zu erstellende Software-TK-Anlage einfließen lassen zu können, stellt sich zuerst die Frage, ob und wie der OpenSource-Charakter einer Basissoftware gewichtet werden soll. OpenSource erlaubt es, ein Teil der Gemeinschaft zu sein, die das Gesamtsystem entwickelt und so eigene Ideen einfließen zu lassen. Ein Closed Source-System ist meist nur, wenn überhaupt, in festgelegten Grenzen erweiterbar. Die Hersteller dieser Systeme agieren nach wirtschaftlichen Kriterien, d.h. Erweiterungen oder Support sind, neben der Basissoftware, oft kostenpflichtig. Bei OpenSource liegt normalerweise kein kommerzielles Interesse vor, da vom Grundsatz her der Quellcode für jedermann frei nutzbar ist. Sofern sich also zum kommerziellen Angebot ein zumindest ebenbürtiges OpenSource-System finden lässt, ist dieses eine bessere Wahl. Dabei spielt nicht nur der Kaufpreis eine Rolle, wie der ewige Kampf zwischen den Betriebssystemen Linux (Open Source) und Windows (kommerziell) eindrucksvoll zeigt. Hier besetzt jedes der Systeme den dafür am besten geeigneten Platz. Man kann also nicht prinzipiell sagen, dass Open Source immer die bessere Wahl ist. Wichtiger ist vielmehr die Flexibilität unter den Gesichtspunkten eines Total Cost of Ownership (TCO), also den gesamten Betriebskosten. SER wie auch Asterisk sind im Gegensatz zu den anderen Systemen frei verfügbar und dabei auch stabil und sehr flexibel. SER wurde bereits erfolgreich in sehr großen Umgebungen mit rund 40000 Telefonen eingesetzt (Vonage, USA), Asterisk beinhaltet mit derzeit rund 154 internen Kommandos genügend Spielraum für fast beliebige Erweiterungen schon auf dieser Ebene und hat sich seit einigen Jahren auch in zahlreichen freiberuflichen Projekten des Autors bewährt. Die kommerziellen Systeme sind somit nicht flexibler oder stabiler als die OpenSource-Pendants, so dass hier auch unter Einbezug aller beschriebenen Kriterien den OpenSource-Varianten der Vorzug gegeben wird.

3.8.2 Entscheidung SER oder Asterisk

SER (SIP Express Router) wie auch Asterisk sind sehr leistungsfähige Systeme, die jedoch unterschiedliche Zwecke verfolgen. Während SER darauf bedacht ist, große Mengen an SIP-Calls zu verarbeiten, ist Asterisk eine All-in-one-Lösung und bringt neben dem SIP-Transfer auch viele weitere Funktionen wie Konferenzen (optional) und eine Voicebox mit. Mit dem erweiterten „Realtime“-Modus, mit dem sich das Gros der Konfigurationsdateien auf eine Datenbank abbilden lässt, sind gute Voraussetzungen geschaffen, um per Weboberfläche mittels PHP-Skripten eine komplette Steuerung zu realisieren. Auch die vielen Funktionen wie Voicemail, PIN-Eingaben, IVR und vieles mehr, das durch Skripte in beliebiger Art und Weise gesteuert werden kann, sind ein guter Ausgangspunkt für das VoIP on CD-System. Die große Programmierergemeinschaft, die stetige Weiterentwicklung und die vielen verfügbaren Protokolle und Funktionen machen Asterisk im Vergleich zu den anderen Systemen zu einem besonders leistungsfähigen Grundsystem, das mit vertretbarem Aufwand äußerst individuell anpassbar ist. Auch für eine spätere Weiterentwicklung bringt Asterisk schon heute Funktionen mit, die nur noch genutzt werden müssen. Einen SER würde man hier ggfs. zusammen mit Asterisk in einer sehr großen Umgebung (mehr als 1000 aktive Nutzer) einsetzen, um den Traffic der Signalisierungspakete abzufangen, da SER deutlich effektiver im SIP-Bereich ist, dafür aber kaum mit Features wie z.B. einer Voicebox aufwartet. Diese Calls lassen sich jedoch über einen SER zu einem Asterisk durchschleusen. Auch SER ist OpenSource und somit frei verfügbar. Da Asterisk in einer Standard-PC-Lösung zwar ein kleineres Gesprächsvolumen als SER verwalten kann, damit aber immer noch etwa 100 gleichzeitige Gespräche möglich sind (G.711, keine transcodierung) und es dazu sehr viele Features schon in der Grundausstattung mitbringt, wird für den weiteren Aufbau ein Asterisk-System als Basis verwendet.

4 Konzeption und Implementierung

Da Asterisk als Basissystem allein für die gewünschten Zwecke nicht ausreicht, wird ein Konzept erstellt, anhand dessen der Aufbau des Gesamtsystems VoIP on CD geschieht. Das reine Asterisk-System ist weder webbasiert steuerbar noch kann es eine Datenbank verwenden. Ein Betriebssystem auf der CD und natürlich ein Installer sind wichtig, um völlig unabhängig von bereits installierter Software zu sein. Somit reicht ein PC mit einer leeren Festplatte aus. In den folgenden Unterpunkten wird das Konzept vorgestellt und die Umsetzung beschrieben.

4.1 Grundkonzept

Neben dem eigentlichen Asterisk, das einen Linux-PC voraussetzt, sind noch weitere Programmpakete notwendig, um die gewünschte Funktionsweise zu erreichen. Die Daten werden in einer MySQL-Datenbank gespeichert, die Bedienung über die Weboberfläche übernimmt der Apache-Webserver, der über PHP-Skripte auch auf die Datenbank zugreifen kann. Asterisk ist im Realtime-Modus auch in der Lage, auf die Datenbank zuzugreifen, weshalb diese auch gleichzeitig die Kommunikationsschnittstelle zwischen Asterisk und Weboberfläche bildet. Folgendes Diagramm verdeutlicht dies:

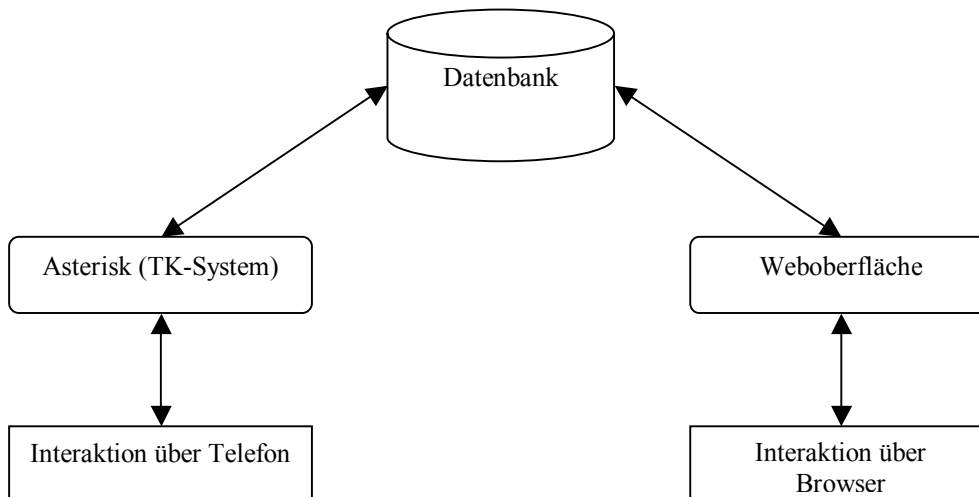


Diagramm 4.1: Kommunikationsbeziehungen

Zur administrativen Bedienung des Systems empfiehlt es sich, keinen eigenen Client zu programmieren, der auf den PCs installiert werden muss und zudem für jedes Betriebssystem anzupassen ist. Eine Administration über einen Webserver ist hier sinnvoller, zumal den meisten Benutzern die Bedienung eines Browsers vertraut ist. Ein weiterer Vorteil ist dabei die Betriebssystemunabhängigkeit des Client-Rechners. Ein Webserver selbst reicht hier freilich nicht aus, die Erstellung einer kompletten Websteuerung unter PHP ist viel Arbeit, doch gibt es bislang keine zufrieden stellenden Alternativen. Die Zugangsdaten für Telefone, die über die Weboberfläche (rechte Seite) in die Datenbank eingepflegt werden, nutzt Asterisk (linke Seite) sofort für die Authentifizierung, d.h. die Anmeldung eines Telefons ist unmittelbar nach dem Anlegen über die Webschnittstelle von Asterisk möglich. Ein Reload oder gar ein Neustart von Asterisk ist im Gegensatz zur Verwendung von Dateien (also ohne Realtime-Modus) nicht notwendig. Durch diesen modularen Aufbau können Komponenten redundant ausgelegt werden, so z.B. die Datenbank: Per MySQL-Replication kann eine logische Datenbank auf mehrere physikalische Systeme verteilt werden. Bei Ausfall eines Systems stehen die Daten nach wie vor zur Verfügung. Auf die zu einer vollständigen Redundanz in der Realität notwendigen Schritte wie ggfs. Round-Robin-DNS oder Einsatz von Loadbalancern wird hier nicht weiter eingegangen. Noch einfacher verhält es sich mit der Weboberfläche. Da sämtliche Daten in der Datenbank gespeichert sind, reicht eine statische Spiegelung des Webservers, um hier eine Redundanz zu schaffen. Auch das Asterisk-System bezieht sämtliche Daten von der Datenbank, somit darf auch dieses System statisch gespiegelt werden. Statisch spiegeln heisst in diesem Fall einfach nur, dass die Dateien von einem PC zu einem anderen z.B. per scp oder sftp übertragen werden und danach nicht mehr abgeglichen werden müssen. Ist die Datenbank nicht mit Asterisk und Webserver auf einem einzigen PC, sondern auf einem anderen System oder Systemverbund bei einer Replication, sollte die Übertragung der Daten zu und von der Datenbank über einen verschlüsselten Tunnel erfolgen, da ansonsten die Daten im Klartext über das Netz laufen würden. Auch die Verbindung zwischen Telefon und Asterisk ist über das SIP-Protokoll unverschlüsselt. Hier ist eine Lösung jedoch ungleich aufwändiger, wenn man hier davon ausgeht, dass sich ein Telefon bei einem Endanwender befindet, der dieses in einem lokalen Netz ohne Möglichkeiten zu einem VPN aufgestellt hat, wie es bei einer typischen Provider/Kunden-Konstellation ist. Zudem ist es in der Realität kaum möglich, zu jedem Kunden ein VPN zu schalten. Daher existieren Verfahren wie SIPS und SRTP, um verschlüsselte Anbindungen von Telefonen an einen Registrar, in diesem Fall den Asterisk-Server, zu ermöglichen. Diese Technik befindet sich jedoch erst in der Aufbauphase. Im Folgenden wird näher auf die einzelnen Module wie Datenbank, Webserver und Asterisk, sowie der gemeinsamen Basis, dem Betriebssystem, eingegangen.

4.1.1 Hardware

Als Hardware eignet sich z.B. ein Mini-ITX-System mit Standard-PC-Komponenten. Ein VIA C3 Prozessor (Nehemia) mit 1 GHz reicht für diese Zwecke aus, hat eine recht geringe Leistungsaufnahme und ist kompakt aufgebaut. Für größere Umgebungen (Provider) rentiert sich der Einsatz eines 19"-Systems, das meist eine qualitativ höherwertige Backplane (Mainboard) beinhaltet. Getestet wurde eine solche Konfiguration u.a. auf einem Systemverbund mit je 1024 MB RAM und einem RAID5-Array mit einer Gesamtkapazität von 48 GB nutzbarem Festplattenplatz.

4.1.1.1 Nutzung von ISDN-Karten

Besonders bei Verwendung von Asterisk kommt schnell der Gedanke auf, eine ISDN-Karte als Gateway zwischen VoIP- und PSTN-Welt im gleichen System zu nutzen. Prinzipiell ist dies auch mit VoIP on CD möglich, der Asterisk wird aus den Sourcen heraus kompiliert – eine gute Ausgangsbasis, um ISDN-Karten einzubinden. Dennoch werden in diesem System in der Basiskonfiguration keine ISDN-Karten berücksichtigt. Die Gründe darin liegen in der Stabilität der einzelnen Verfahren zur Umsetzung, die der Übersichtlichkeit halber in einer Tabelle gegenübergestellt werden. Die Spalte Stabilität in der Tabelle gibt die Erfahrung des Autors in größeren Asterisk-Installationen (bei Providern) wie auch Erfahrungsberichte anderer Nutzer in diversen Foren wider. Es ist wünschenswert, dass diese Tabelle bald keine Gültigkeit mehr besitzt und ISDN-Karten deutlich stabiler unter Asterisk funktionieren.

Karte / Verfahren	ISDN-Kanäle	Stabilität / Störungen
AVM A1 (passiv, PCI oder ISA)	2 (1 x S0)	Unter CAPI schlechte Sprachqualität, grundsätzlich aber relativ instabil, d.h. min. 1 Störung innerhalb weniger Tage.
AVM C4 (aktiv, PCI)	8 (4 x S0)	Karte entkoppelt sich ca. wöchentlich, keine Erreichbarkeit, reload rekonnetiert den Treiber.
Digium TE 410 P	120 (4 x S2m)	Stabil über mehrere Monate, sofern Kernelheader korrekt eingebunden werden.
CologneChip, HFC-Karte	2 (1 x S0)	Mit HFC-Treiber halbwegs stabil, doch Sprachausfälle möglicherweise durch Taktstörungen.

Tabelle 4.1: Vergleich der Stabilität gängiger ISDN-Karten in einem Asterisk-System

Es zeigt sich, dass lediglich die Digium-Karte in kommerziellen Umgebungen sinnvoll einsetzbar ist, es sei denn, Störungen werden in Kauf genommen. Schade ist hierbei, dass die HFC-Karte im Dauertest nicht so stabil ist, wie ursprünglich angenommen, denn mit etwas Arbeit lässt sie sich auch im NT-Modus betreiben, so dass an diese auch ein ISDN-Telefon oder eine ISDN-Telefonanlage angeschlossen werden kann und sie somit gut als Gateway für eine ganze Telefonanlage dienen könnte. Ggfs. wird sich die Stabilität mit der Weiterentwicklung von Asterisk und den Treibern der Karten noch verbessern. Die Digium-Karte wird für die meisten Installationen zu teuer und zu groß geraten sein, da sie mit weit über 1000 Euro und den 4 S2m-Anschlüssen eher für große Installationen in Betracht kommt, bei denen die zur Verfügung gestellten 120 Kanäle auch genutzt werden.

4.1.2 Betriebssystem

Bevor die einzelnen Dienste installiert werden können, wird ein Betriebssystem benötigt. Asterisk läuft zwar prinzipiell unter Windows, doch wird hier eine Linux-Emulation verwendet. Sinnvoller für alle Komponenten ist daher der Einsatz von Linux. Es existieren unzählige Linux-Varianten. Einige sind mehr auf den Endanwender bezogen und helfen bei der Installation der grafischen Benutzeroberfläche (z.B. SuSE), andere hingegen sind etwas aufwändiger zu betreiben, dafür aber schlank. Für erfahrene Anwender, die eher Server-Systeme aufsetzen und die Kommandozeile schätzen, ist Debian eine gute Wahl, da hier ein Minimal-system aufgesetzt werden kann, das sehr einfach per apt-get erweiterbar ist. Auch ist das Update hier beispielhaft gelöst. Eine Installation wird meist von CD nur minimalistisch durchgeführt, so dass eine Internetverbindung besteht. Alle weiteren Pakete werden in aktueller Version von einem der Debian-Mirrors geholt (z.B. per ftp oder http). Auch mit einer älteren Installations-CD werden hier immer die aktuellen Pakete verwendet. Um nicht Gefahr zu laufen, ein unausgereiftes Paket zu erhalten, unterteilt sich der Bereich der Pakete in oldstable, stable, testing und unstable. Mit den Paketen aus dem stable Bereich, die hier verwendet werden, ist ein sicherer und möglichst störungsarmer Betrieb gewährleistet. Auf die neuesten Funktionen muss jedoch dabei verzichtet werden, da diese nur allmählich von unstable in testing und dann in stable eingeordnet werden. Asterisk selbst kann nicht aus den Debian-Quellen bezogen werden, da die Paketvariante keinen Realtime Modus zulässt, was die Verwendung eines MySQL-Servers zur Speicherung der Konfigurationsdaten unmöglich machen würde.

4.1.3 Datenbank

Auch wenn die Datenbank eine zentrale Rolle spielt, werden hier nur die Standardfunktionen benötigt. Stored procedures, wie sie in großen Datenbanken wie Oracle vorhanden sind, benötigt das System nicht. Insgesamt

werden für eine typische Datenbankanwendung nur wenige Daten gespeichert und transferiert. Asterisk erlaubt neben Textdateien den Einsatz einer MySQL- wie auch einer PostgreSQL-Datenbank. Durch die jahrelangen Erfahrungen des Autors und die sehr weite Verbreitung von MySQL wird dieser Datenbank der Vorzug gegeben. Da pro Telefon bei der Registrierung (alle 2 bis 60 Minuten), sowie bei jedem Gesprächsauf- und Abbau ein Datenbankzugriff mit wenigen Datensätzen notwendig ist, kann eine MySQL-Datenbank von der Kapazität her mindestens 100000 Telefone verwalten, legt man die positiven Erfahrungen des Autors mit einer getesteten Datenmenge von 1000000 Datensätzen pro Stunde zugrunde. Somit wird der Engpass nicht bei der Datenbank liegen. Auch ist die besprochene Replikation in diesem Fall weniger eine Idee zur Lastverteilung, als mehr eine zusätzliche Datensicherheit.

4.1.4 Webserver

Als Webserver bieten sich nicht viele reelle Alternativen. Es gibt zwar unterschiedliche Webserver wie thttpd, Roxen, Boa, Jigsaw oder auch den Netscape Webserver, doch gibt es keinen triftigen Grund, auf den millionenfach installierten und bewährten Apache Webserver zu verzichten, der zwar einige Fähigkeiten mitbringt, die hier nicht benötigt werden, damit etwas größer ausfällt als notwendig. Laut netcraft.com ist der Apache-Server bei weltweit 70% der Webserver im Einsatz. Nicht zuletzt hat der Autor jahrelange Erfahrung mit diesem Webserver und vertraut der bekannten Stabilität dieses Softwarepaketes. Den Apache gibt es in der Version 1.3 und 2.0 wobei letztere noch als Beta-Version zu sehen ist, auch wenn sie schon recht stabil ist. Da schon die Version 1.3 sämtliche benötigten Funktionen bereitstellt, wird diese hier verwendet.

4.1.5 Weboberfläche

Da der Webserver selbst nur eine Grundlage dafür bildet, mit der Datenbank und damit indirekt mit dem Asterisk zu kommunizieren, aber selbst keine fertige Weboberfläche mitbringt, ist es notwendig, diese zu erstellen. Nach der Installation des Webserver ist nicht mehr als eine „Hallo Welt“-Seite zu sehen. Eine Kommunikation mit der Datenbank wie auch die komplette Interaktion mit dem Benutzer muss programmiert werden. CGI-Skripte, wie sie vor Jahren üblich waren, sind heute nicht mehr flexibel genug. Einzig PHP ist hier eine sinnvolle Wahl, die alle notwendigen Funktionen beinhaltet, d.h. Interaktion mit der Datenbank, Schreiben von Dateien und dynamische Ausgabe der Texte. Eine strukturierte Programmierung unter PHP ist möglich, so findet sich ein anderer PHP-Programmierer schnell zurecht. Neben dem Einpflegen der Daten für Asterisk, d.h. Parameter für Telefone, Voicebox, Routen usw., muss hier auch eine sichere Schnittstelle zum (Web-)User geschaffen werden. Mit Blickwinkel auf ein Providersystem werden Accounts für User und Administratoren benötigt. Bei einfacheren Installationen reicht es dann aus, nur einen Administratoraccount zu nutzen, die Useroberfläche bleibt dann ungenutzt. Um Missbrauch zu unterbinden ist eine https-Schnittstelle sinnvoll, da hier die Eingabe von Username und Passwort verschlüsselt zum Webserver übertragen wird. Ausserdem ist es notwendig, ein Verfahren zu programmieren, mit dem neben der Authentifizierung beim Login auch eine fortlaufende Identifizierung möglich ist, da das http-Protokoll verbindungslos ist und für jede angeklickte Seite eine neue Authentifizierung notwendig wäre. Eine Lösung hierzu findet sich z.B. in der Verwendung von Sessions in PHP.

4.1.6 Asterisk

Die Entscheidung für Asterisk ist zwar schon gefallen, nicht aber die Art, wie diese Software genutzt wird. Der Realtime-Modus ist für selbst kompilierte Pakete per Default eingeschaltet. Fertige Versionen von Asterisk werden durch den Beta-Status von Realtime nicht damit versehen, was jedoch verwundert, da dieser Modus schon seit rund einem Jahr vom Autor genutzt und als recht stabil angesehen wird. Offenbar will die Release-Gruppe um Asterisk, dass der Realtime-Modus nur von erfahrenen Asterisk-Anwendern eingesetzt wird, da ansonsten möglicherweise die Support-Mailinglisten bei einer Fehlkonfiguration überflutet werden könnten. Da es für die Anbindung von Asterisk an eine Datenbank keine Alternativen zum Realtime-Modus gibt, wird dieser trotz Beta-Kennzeichnung verwendet. Beruhend auf den Erfahrungen des Autors, wird dieses Konstrukt dennoch zuverlässig funktionieren.

4.2 Installationsschritte

Welche Schritte im Einzelnen notwendig sind, um die Teile auf einem System zusammenzusetzen, zeigt folgendes Diagramm. Hier wird eine manuelle Installation aller Teile nachgebildet.

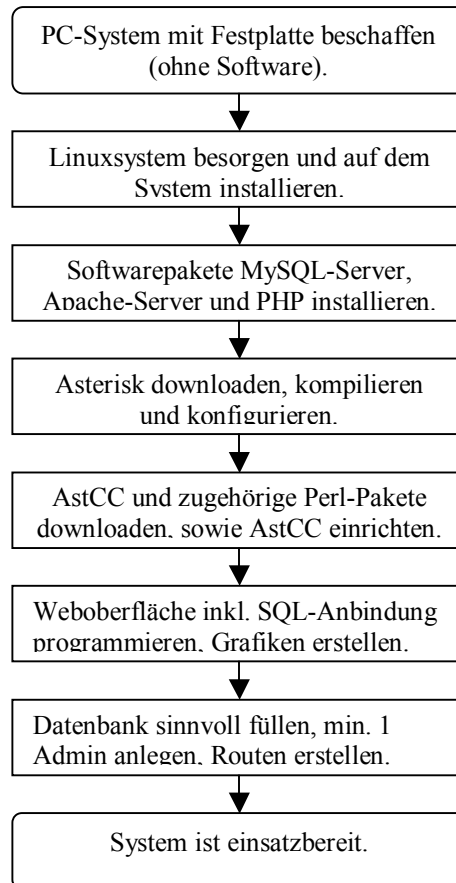


Diagramm 4.2: Schritte einer manuellen Installation analog zu VoIP on CD

Es ist zu erkennen, dass einige Schritte durchgeführt werden müssen, um ein einsatzbereites System zu erhalten. Unter Linux lassen sich diese Schritte durch Skripte automatisieren, sofern ein PC die Daten aufnehmen kann, d.h. von einer bootfähigen, fertigen CD installiert wird. Da das Betriebssystem auch installiert werden muss, ist eine normale Linux-Installations-CD so anzupassen, dass neben dem customizing auch die zusätzliche Software für Voice over IP nebst benötigten anderen Paketen installiert werden kann. Eine Linux-CD lässt sich nicht wie ein Skript einfach ändern, hierzu ist es notwendig, diese zu remastern, d.h. eine neue CD zu erstellen, die neben dem Bootsektor und Bootloader auch den Kernel und die grundlegende Software wie z.B. Treiber für die Netzwerkkarte enthält. Die einzelnen Schritte können, wie in der Grafik dargestellt, modulweise abgearbeitet werden, was auch eine mögliche Fehlersuche vereinfacht. So wird im ersten Schritt ein Basis-Linux installiert, das neben der Festplatte auch eine Netzwerkkarte ansprechen kann, um dann weitere (aktuelle) Software über das Netzwerk von einem Debian-Server zu erhalten. Sind alle Pakete wie Datenbank und Webserver eingerichtet, installiert das Hauptskript das VoIP-System, d.h. es holt sich weitere benötigte Pakete von einem anderen Server, der speziell für die VoIP on CD bereitgestellt wurde (<http://www.voiponcd.de> bzw. <http://voiponcd.rowi.net>) und bereitet diese für den Einsatz vor. Nachdem auch die Weboberfläche (PHP-Skripte) heruntergeladen und installiert wurde, muss auch der Datenbank ein erster sinnvoll definierter Inhalt zugewiesen werden, was über einen Datenbankimport mit einem vorher als Referenzwert abgespeicherten Datenbankinhalt geschieht. Hierbei wird auch ein Administrationsaccount angelegt, ohne den die komplette Benutzung der Weboberfläche nicht möglich wäre. Anschließend kann das System komplett über die Weboberfläche genutzt werden, da zahlreiche Skripte z.B. ein Einpflegen von Daten und ein anschließendes Reload von Asterisk für den Nutzer transparent bewerkstelligen.

4.3 Automatisches Softwareupdate bei Installation

Software auf einer CD hat immer den Nachteil, dass sie recht schnell veraltet. Beim Brennen wird der aktuelle Stand der Software auf die CD gebrannt. Notwendige Updates, insbesondere sicherheitsrelevante Patches, wie aber auch neue Funktionen können natürlich erst zum Zeitpunkt der Behebung bzw. Erstellung zur Verfügung gestellt werden. Um mit einer solchen CD dennoch immer die aktuellsten Pakete zu erhalten, werden bei Debian Linux nach der Grundinstallation die notwendigen Pakete aus dem Internet geholt. Eine Fehlerbehebung in den Paketen ist somit auch nach dem Brennen der CD möglich. Jede Installation wird damit immer auf dem Stand des Installationszeitpunktes und nicht des Zeitpunktes der Erstellung der CD sein. Angeregt durch die tägliche Weiterentwicklung des Asterisk-Systems wird dieser Ansatz auch für diese Programmteile weiter verfolgt. Dazu existiert ein Server im Internet, von dem automatisch bei der Installation über <http://voiponcd.rowi.net> die aktuelle Version der Software heruntergeladen wird. Bei Ausfall dieses Rechners kann der DNS unabhängig von den Daten auf der CD auf ein anderes System geschwenkt werden. Sollte die Last auf dem System zu hoch werden, kann diese per Round Robin Verfahren (mehrere Server im DNS zur gleichen Subdomain) auf mehrere Systeme verteilt werden.

4.4 Konfiguration Debian-Grundsystem

Ein erheblicher Teil der Konfiguration liegt in der Erstellung der Debian-CD, die als Basis des Asterisk dient. Immerhin ist es hier notwendig, ein komplettes Linux-System inkl. Bootloader, Kernel etc. auf einer bootfähigen CD zu erstellen. Auch wenn schon mehrere Distributionen existieren, sind die Informationen zum Erstellen einer solchen schwer zu finden, ist doch eine eigene Linux-Version normalerweise nicht notwendig.

4.4.1 Debian-CD remastern

Die Installations-CD der Debian Distribution beinhaltet sehr viele Konfigurationsmöglichkeiten. Da in diesem Fall nur ein Grundsystem inkl. Asterisk, MySQL, Apache, PHP und einigen Hilfsprogrammen installiert werden soll, ist es notwendig, eine eigene CD zu mastern. Damit ergibt sich die Gelegenheit, die CD gleich in allen Belangen anzupassen, d.h. nicht benötigte Pakete zu entfernen und ein eigenes Boot-Logo zu installieren [exss].

4.4.1.1 Isolinux / Syslinux

Um ein bootfähiges Medium herzustellen, wird ein Bootloader wie isolinux oder syslinux benötigt. Für VoIP on CD wurde isolinux verwendet, weil dieses nicht den Umweg über die DOS-Emulation von Diskettenimages nimmt, sondern direkt von einer ISO9660/El Torito CD-ROM im „no emulation“ Modus bootet.

4.4.1.1.1 Konfigurationsdatei isolinux.cfg

Die Parameter in dieser Konfigurationsdatei geben die Position des Kernels auf der CD an, definieren F-Tasten, bei deren Druck z.B. eine Hilfeseite erscheint und mehr. Alles, was hier definiert wird, steuert unmittelbar nach dem Booten von der CD das Verhalten der Software.

4.4.1.1.2 Anzeige und Auswahlmenü bei CD-Boot

Wenn die fertiggestellte CD gebootet hat, erscheint ein Auswahlmenü, das sich mit den Funktionstasten steuern lässt. Die dort angezeigten Texte werden zuvor in der Datei isolinux.cfg bestimmten Textdateien zugewiesen. Folgende Funktionen lassen sich bei der Bildschirmausgabe der Texte realisieren:

<FF> = <Strg-L> = ASCII 12	Bildschirm löschen (wie Shell-Befehl clear)
<SI> = <Strg-O> = ASCII 15	Vorder- und Hintergrundfarbe setzen (<SI><Hintergrund><Vordergrund>)
	0 = schwarz 8 = dunkelgrau
	1 = dunkelblau 9 = hellblau
	3 = dunkelcyan b = hellcyan
	4 = dunkelrot c = hellrot
	5 = dunkelviolett d = hellviolett
	6 = braun e = gelb
	7 = hellgrau f = weiß
<CAN> = <Strg-X> = ASCII 24	In Grafikmodus schalten und Grafik anzeigen (<CAN>image.lss)
 = <Strg-Y> = ASCII 25	Zurück in den Textmodus schalten
<SUB> = <Strg-Z> = ASCII 26	END of File-Kennzeichnung

4.4.1.1.3 Eigener Splashscreen

Sofort nach dem Booten der CD wird neben dem Text auch eine Grafik angezeigt. Eine Standardgrafik im BMP-Format lässt sich mit Hilfe einiger Programme zum LSS-Format konvertieren, das isolinux dafür benötigt. Folgende Schritte sind dazu notwendig:

- Grafik in einem BMP-Format (z.B. mit mspaint) mit einer Farbpalette von 16 Farben erstellen.
- Paket netpbm auf einem Linuxrechner installieren.
- Paket imagemagick installieren (wird für convert benötigt).
- `convert splash.bmp splash.png`
- `LANG="C" pngtopnm splash.png | ppmtolss16 > splash.lss`

splash.pnm ist die Datei, die in `./isolinux/` hineinkopiert wird. Das explizite Setzen der Variable LANG hat den Hintergrund, dass das Konvertierungstool nicht mit bestimmten Zeichensätzen zusammen arbeitet.

4.4.1.2 Bootparameter

Da das Ziel der CD bekannt ist, lassen sich viele Fragen, die bei einer normalen Installation eines Debian-Systems gestellt werden, automatisch beantworten. Dem Anwender werden somit Entscheidungen abgenommen und die Einfachheit des Gesamtsystems so unterstrichen. In der Konfigurationsdatei `isolinux.cfg` wird hierzu im Append-Parameter des Kernels der Pfad zum Preseed-File (`preseed/file=/cdrom/preseed.cfg`) angegeben, womit die komplette Zeile folgendermaßen aussieht:

```
APPEND      DEBCONF_PRIORITY=critical  languagechooser/language-name=German
initrd=/install/initrd.gz             ramdisk_size=10240             root=/dev/rd/0
devfs=mount,dall rw preseed/file=/cdrom/preseed.cfg
```

Auf dieser CD werden lediglich die Sprache, Festplattenpartitionierung und das Root-Passwort abgefragt. Die Partitionierung der Festplatte kann statt mit einer einfachen Bestätigung mit der Enter-Taste auf Wunsch auch manuell vorgenommen werden. Bei Verwendung von Kernelparametern ist darauf zu achten, dass bei Verwendung eines Kernels kleiner als 2.6.9 insgesamt nicht mehr als 8 Parameter entstehen, da die 2.4er Kernel weitere Parameter ignorieren und ältere 2.6er Kernel einen Kernelpanic hervorrufen.

4.4.1.3 Voreinstellungsdatei (preseed)

Da bei einer gewöhnlichen Debian-Installation die gewünschte Einsatzweise des Linuxsystems nicht bekannt ist, werden dem Anwender einige Fragen gestellt, die ein unerfahrener Nutzer möglicherweise nicht beantworten kann. Mit Hilfe der Voreinstellungsdatei können Antworten so vorgegeben werden, dass die Frage nicht mehr am Bildschirm erscheint. Das ist für dieses System sinnvoll, da genau definiert ist, welchen Zweck das System letztlich verfolgt. Anpassungen an das Umfeld des Benutzers werden im Anschluss an die Installation vorgenommen.

Nach einer Standardinstallation des Debian-Systems lassen sich die beantworteten Fragen in dem für die Voreinstellungsdatei benötigten Format folgendermaßen ausgeben, sofern das Paket `debconf-utils` installiert ist:

```
debconf-get-selections -installer >preseed.txt
debconf-get-selections >>preseed.txt
```

Es zeigt sich, dass die Einträge nicht in chronologischer Reihenfolge in die Datei (hier: `preseed.txt`) geschrieben werden, zudem sind viele Einträge nicht notwendig, wenn als Bootparameter in der Append-Zeile die `DEBCONF_PRIORITY` auf `critical` gesetzt wird und somit das System insgesamt weniger Fragen stellt.

Auszug aus einer automatisch generierten Voreinstellungsdatei:

```
# dummy template for preseeding unavailable questions
d-i      tzconfig/choose_country_zone_single      string      true
# dummy template for preseeding unavailable questions
d-i      console-data/keymap/qwertz/german/standard/keymap      string
# Integrität einer weiteren CD-ROM prüfen?
d-i      cdrom-checker/nextcd      boolean      false
# Internal use
```

```

d-i      debian-installer/consolidated string  kbd=lat0-sun16(iso15)
# Choose a country, territory or area:
d-i      countrychooser/shortlist-zh_CN  select
# Choose a country, territory or area:
d-i      countrychooser/shortlist-sv     select
# kernel needs a link in /boot/ (linux only)
d-i      debian-installer/kernel/linux/link_in_boot      boolean false
# Internal use

```

Um die Übersichtlichkeit zu erhöhen, ist eine manuelle Nacharbeit notwendig. Nach dem Entfernen der unnötigen Einträge und sortieren nach zugehörigen Zeilen erhält man folgende Voreinstellungsdatei (Auszug). Leerzeilen in der Datei führen dazu, dass sämtliche Inhalte nicht akzeptiert werden.

```

##### Festplatte automatisch partitionieren
d-i      partman-auto/choose_recipe      select  Alle Dateien auf eine
Partition (für Anfänger empf.)
# Automatisch neues Label schreiben
d-i      partman/confirm_write_new_label boolean true

```

Da eine eigene Voreinstellungsdatei wegen der bis dato spärlichen Dokumentation etliche Versuche erfordert, bis alle Parameter den Vorstellungen entsprechend funktionieren, empfiehlt es sich, in der Append-Zeile des Kernels den Parameter `preseed/file=/cdrom/preseed.cfg` auf `preseed/url=http://<ip>/preseed.cfg` anzupassen. Mit der gebrannten CD können dann über einen externen Webserver verschiedene Versionen der Preseed-Datei getestet werden.

Mit dem folgenden Befehl lassen sich die Eingaben der `preseed.cfg` prüfen.

```
debconf-set-selections -v -c preseed.cfg
```

Insbesondere Leerzeichen am Anfang und Ende eines Kommandos, die in einem Texteditor wie z.B. dem `vi` nicht sofort auffallen, sind hierbei gut erkennbar. Nachfolgend ist ein Auszug der Ausgabe des Kommandos aufgeführt.

```

info: Trying to set 'partman/confirm' [boolean] to 'true'
info: Loading answer for 'partman/confirm'
info: Trying to set 'tzconfig/gmt' [boolean] to 'false'
info: Loading answer for 'tzconfig/gmt'
info: Trying to set 'tzconfig/choose_country_zone_multiple' [select] to
'Europe/Berlin'
info: Loading answer for 'tzconfig/choose_country_zone_multiple'
info: Trying to set 'tzconfig/choose_country_zone_single' [boolean] to
'true'
info: Loading answer for 'tzconfig/choose_country_zone_single'

```

4.4.1.4 Mkisofs

Aus den vorbereiteten Daten auf der Festplatte des Rechners, auf dem die CD zusammengestellt wird, erstellt `mkisofs` das fertige ISO-Image. Ein Beispiel einer Anwendung des Befehls ist hier wiedergegeben (nähere Infos zur Nutzung mit `man mkisofs` in einer Linux Shell):

```

mkisofs -V "VoIPonCD 1.0" -o ../rowi-beta.iso -quiet -no-emul-boot -boot-
load-size 4 -boot-info-table -b isolinux/isolinux.bin -c
isolinux/boot.cat -r .

```

4.4.1.5 CD brennen

Beim Brennen der CD, z.B. mit Nero muss darauf geachtet werden, die Session abzuschließen, da ansonsten z.B. bei erneutem Brennen einer CD-RW eine Session angehängen und der Bootloader vom alten System beim Booten genutzt wird. Das kann zu nicht erklärbaren Phänomenen führen. Bei einem einmaligen Brennen auf einem frischen Rohling kommt es natürlich nicht zu diesen Effekten.

4.4.2 Installiertes System / Softwareinstallation

Die gebrannte CD ist lediglich ein Teil des Gesamtsystems, da hiermit nur das Basissystem installiert wird und weitere Pakete aktuell aus dem Internet heruntergeladen werden, das betrifft nicht nur Debian-Pakete wie Apache oder den MySQL-Server, sondern insbesondere auch die VoIP on CD-Pakete. Trotz der schnellen Weiterentwicklung von Asterisk bleiben die gebrannten CDs up to date.

4.4.2.1 Installationskript voiponcd.sh

Nach Abschluss der Grundinstallation wird das Skript voiponcd.sh von <http://voiponcd.rowi.net/voiponcd.php> heruntergeladen und ausgeführt. Hiermit beginnt die eigentliche Installation des VoIP on CD-Systems. Da selbst die Installationsroutine von einem externen Server heruntergeladen wird, sind Änderungen auch nach dem Brennen von CDs möglich, was besonders für sicherheitsrelevante Bugfixes wichtig ist. Typische Serverdienste wie Apache, PHP und MySQL werden als stable Version per apt-get von ftp.de.debian.org geholt.

4.4.2.2 Kernelversion

Als Kernel wird der „alte“ Linuxkernel 2.4.27 genutzt. Ein Kernel der Version 2.6, insbesondere ab 2.6.9 enthält zwar prinzipiell mehr Features, doch sind diese für das hier verwendete System nicht relevant. Daher wird auf einen Kernel zurückgegriffen, der seit längerer Zeit sehr stabil funktioniert und für diese Anwendung keine Nachteile birgt.

4.4.2.3 Filesystem ext3

Die Journaling Filesysteme wie ReiserFS und ext3 haben die älteren mittlerweile fast ganz abgelöst. Im Falle eines Fehlers in der Datenstruktur besteht hier schon durch die Verteilung der Index-Informationen eine deutlich höhere Wahrscheinlichkeit der vollständigen Datenwiederherstellung als bei den alten Filesystemen. Durch Praxiserfahrungen mit ext3 und ReiserFS wird hier ganz bewusst aufgrund mehrfacher negativer Erfahrungen mit ReiserFS das ext3-System vorgezogen.

4.4.3 Implementierung MySQL

Die aktuell stabile Version des MySQL-Datenbank-Servers wird einfach per apt-get bei der Installation vom Debian-Server geholt. In den Defaulteinstellungen ist für User root in der Datenbank kein Passwort vergeben, was der Einfachheit halber beibehalten wurde, aber keinen gängigen Sicherheitsrichtlinien entspricht. Ein Zugriff auf die Datenbank ist allerdings ohnehin nur für auf dem System angemeldete User möglich. In späteren Versionen von VoIP on CD wird sich dies ändern.

4.4.4 Implementierung Apache Webserver

Auch der Apache Webserver wird als stabiles Paket per apt-get vom Debian-Server heruntergeladen. Nach wie vor wird die Version 1.3 verwendet, auch wenn Version 2.0 schon seit längerer Zeit existiert und mittlerweile stabil läuft. Sobald das Debian-Team die neuere Version als stabil genug erachtet, wird diese bei einer Neuinstallation von VoIPonCD verwendet. Notwendige weitere Pakete für den Webserver wie PHP4 oder die MySQL-Anbindung werden ebenfalls per apt-get geladen, was den Installationsaufwand auf ein Minimum reduziert.

4.5 Konfigurationsschritte Asterisk

Die am 29. August 2005 vorgestellte Version 1.2.0beta ermöglicht es erstmals, den Realtime-Modus zu nutzen, ohne auf den tagesaktuellen und von der Stabilität tagesabhängigen CVS-Snapshot zurückgreifen zu müssen. Trotz der Bezeichnung „beta“ wurde von der Entwicklungsgemeinde um Asterisk sehr viel Aufwand getrieben, um diese ganz neue Version sehr stabil zu halten. Da eigene Tests die Stabilität bestätigen, wird diese neue Version, trotz des Anhangs „beta“ im Namen mit Stand Oktober 2005 über die CD installiert. Sofern Bugfixes notwendig werden oder neue, stabilere Versionen erhältlich sind, werden diese der Installationsroutine der bereits gebrannten CDs über das Internet zugänglich gemacht und bei neuen Installationen berücksichtigt. Wie bereits beschrieben, kann Asterisk in der CVS-Version und in der regulären Version ab 1.2 die Konfigurationsparameter auch aus einer Datenbank einlesen, was den Vorteil hat, dass eine Weboberfläche die geänderten Daten einfach in die Datenbank schreibt, ohne einen Reload von Asterisk veranlassen zu müssen. Aber auch hier gibt es Ausnahmen, wie z.B. bei neuen Kontexten oder neuen Carriern. Diese werden in einzelnen Textdateien auf dem Server gespeichert. Mit der Version 1.2 von Asterisk ist zwar Realtime vollständig implementiert, doch kommt auch jetzt noch vorerst die Konfiguration nicht ganz ohne Konfigurationsdateien aus.

4.5.1 Betriebsart

Asterisk lässt sich neben dem normalen Modus auch in der Betriebsart Realtime verwenden. Hierbei werden Daten in einer Datenbank abgelegt und bei Änderung dieser Daten im Gegensatz zum Normalbetrieb, bei dem ein Reload erforderlich ist, sofort aktualisiert.

4.5.2 SIP-Konfiguration (sip.conf)

Die Datei sip.conf nimmt die Einträge für User und Provider auf, die im SIP-Standard mit dem System kommunizieren. Hierbei werden ausgehende und ankommende Gespräche getrennt behandelt. Für ankommende Gespräche muss eine Registrierung beim Provider vorgenommen werden, die üblicherweise im Stundentakt oder in kürzeren Intervallen erneuert wird. Diese Registrierung kann in derzeitigen Versionen von Asterisk nicht in die Datenbank ausgelagert werden, sondern muss in Konfigurationsdateien Platz finden. Um die automatisierte Verarbeitung dieser Textdateien so angenehm wie möglich zu machen, sind diese hier je Registrierung in eine separate Datei eingeflossen. Somit kann eine Registrierung bei einem Provider dauerhaft durch Anlegen der zugehörigen Datei und anschließendem Reload von Asterisk erfolgen. Ein fehleranfälliges, automatisiertes Cut and Paste innerhalb einer Konfigurationsdatei ist damit nicht mehr notwendig. Ein wiederholtes Löschen einer Datei, um den Provider wieder zu entfernen, das schlicht fehlschlägt, ist weniger fehlerträchtig wie ein erneutes Löschen einer Zeile in der Konfigurationsdatei, sofern nicht umfangreiche Prüfungen vorgenommen werden.

Beispiel für die Verwendung von Includes innerhalb der sip.conf (hier für ankommende Gespräche genutzt):

```
/etc/asterisk/sip-register/provider1      ; Datei für Provider 1
/etc/asterisk/sip-register/provider2      ; Datei für Provider 2
/etc/asterisk/sip-register/provider3      ; usw..
```

Der Inhalt einer Include-Datei sieht folgendermaßen aus:

```
;Kommentarzeile zum Provider, ggfs. Prüfziffern
register => username:passwort@provider.example.com/Ziel-Extension
```

Die Ziel-Extension gibt hier an, auf welcher internen Rufnummer ankommende Rufe des Providers gelegt werden. Dabei ist eine Extension nicht zwingend mit einer Zielrufnummer (Telefon) gleichzusetzen. Vielmehr kann eine Extension als Platzhalter für beliebige, auch variable, Ziele definiert werden. Ein Block mit 1000 Rufnummern, z.B. für Mehrwertdienste (0180-Rufnummern) wird im Testsystem auf eine Extension gelegt. Benötigt ein Kunde eine dieser Mehrwertnummern, so wird diese einzelne Nummer auf den Anschluss dieses Teilnehmers gelegt.

Die vollständige Konfigurationsdatei sip.conf sieht folgendermaßen aus. Die letzte Zeile ist hierbei nicht auskommentiert, sondern ein Befehl zum Laden der o.g. externen Dateien.

```
[general]
context=sippool      ; Defaultkontext für ankommende Gespräche.
bindaddr=0.0.0.0     ; IP-Adresse, an die der Asterisk gebunden
                    ; wird (0.0.0.0 bindet an alle Interfaces).
srvlookup=yes        ; Wenn auf yes gesetzt, werden DNS-Abfragen zu
                    ; SRV-Records bei ausgehenden Gesprächen getätigt.
defaultexpiry=600   ; Zeitintervall (in s), in der sich z.B. ein Telefon
                    ; neu registrieren muss.
disallow=all         ; Ähnlich einer Firewall: Zuerst alle Codecs verbieten...
allow=ulaw            ; ...um sie dann nacheinander freizugeben.
allow=alaw            ; Auch hierbei spielt die Position der freizugebenen Codecs
                    ; eine Rolle (Priorität)!
allow=gsm             ; GSM hat zwar die beste Kompression, doch auch die
                    ; schlechteste Qualität.
language=de          ; Per Default wird die Sprache auf deutsch gesetzt, diese
                    ; kann später z.B. für die Ansagen ausgewertet werden.
tos=lowdelay          ; Der Type of Service kann hier definiert werden, um z.B.
                    ; Routern ein Flag im Datenpaket mitzugeben, um die Sprachströme
                    ; gegenüber anderen Datenpaketen zu priorisieren.
                    ; Neben dem Wert lowdelay hat sich auch reliability bewährt, dies
                    ; ist jedoch abhängig von den Routern und deren Einstellungen.
maxexpiry=3600       ; Endgeräte können normalerweise selbst bestimmen, in welchen
                    ; Zyklen sie sich neu registrieren. Hier besteht die Möglichkeit,
                    ; diese Zeit zu begrenzen (in s).
rtcachefriends=yes   ; Für die Message Waiting Signalisierung (MWI) bei Verwendung des
                    ; Realtime-Modus notwendig, um die User aus der DB zu cachen.
```

```

nat=yes                ; Per Default wird hier davon ausgegangen, dass sich angeschlossene
                        ; Endgeräte hinter einem NAT (PAT) Gateway befinden und sich somit
                        ; die lokale IP-Adresse des Endgerätes von der zum VoIP-Server
                        ; sichtbaren IP (meist öffentliche IP) unterscheidet.

; Sipuser ankommend, alle Register liegen im Verzeichnis /etc/asterisk/sip-register/
; in separaten Dateien (eine pro Register)
#include /etc/asterisk/sip-register/*

```

4.5.3 SIP-Datenbank (zu sip.conf)

Sinnvollerweise werden die User (Telefone) nicht in Dateien gespeichert. Die Datenbank bietet sich hier wesentlich besser an. Der Name „sipfriends“ wird in der Datei extconfig.conf mit der Zeile

```
sipusers => mysql,asterisk,sipfriends
```

definiert. Die Tabelle muss natürlich dann unter diesem Namen angelegt werden.

Feld	Bedeutung
Id	Automatisch vergebene, fortlaufende ID.
Name	Username , hier: interne Rufnummer.
Accountcode	Kartenummer des Kontos, macht es möglich, mehrere Telefone auf eine Kostenstelle laufen zu lassen. Hier jedoch gleich der Rufnummer.
Amaflags	Feature für das Accounting, hiermit kann ein Account z.B. grundsätzlich von Gebühren befreit werden.
Callgroup	Gruppenzugehörigkeit eines Telefons (wird hier noch nicht genutzt).
CallerID	Caller ID im Format „Vorname Nachname <Rufnummer>“.
Canreinvite	Direkte Kommunikation mit Ziel erlauben (Vorsicht: Kann ein Billing unmöglich machen!).
Context	Der Kontext, in dem sich der User befindet.
DefaultIP	Sollte die Registrierung abgelaufen sein, d.h. das Telefon ist am Server nicht mehr angemeldet, so werden ankommende Anrufe blind an diese IP geleitet. Hier kann z.B. eine DnyDNS-Adresse des Telefons eingetragen werden.
DTMFmode	Es gibt bei SIP mehrere Möglichkeiten der Übertragung von DTMF (MFV) Tönen. Sinnvoll ist hier die Verwendung nach RFC2833.
Fromuser	Bleibt bei Usern leer, bei Anbindung an einen anderen Provider kommt hier der Username rein.
Fromdomain	Bleibt bei Usern leer. Manche Provider erwarten bei einer Anbindung, dass dort die Domain des Providers eingetragen wird (z.B. sipgate.de)
Host	Hier wird der Typ des Hosts eingetragen, auch bei festen IPs reicht hier der Wert „dynamic“.
Insecure	Bleibt bei Usern leer, bei Anbindungen an andere Provider funktioniert das Peering mit dem Wert „very“, was in der Tat bedenklich stimmt. Die Authentifizierung geschieht hier nur mit (im Klartext übertragenen) Username und Passwort. Derzeit existiert jedoch kein schönerer Weg.
Language	Werden mehrere Sprachen unterstützt, kann hier die Usersprache eingegeben werden.
Mailbox	Dem User kann eine beliebige Mailbox zugewiesen werden, da meist aber ein User seine eigene Mailbox erhält, ist dieser Wert gleich dem Accountcode oder der Rufnummer.
Md5secret	Das Passwort kann hier auch MD5-verschlüsselt abgelegt werden (ungetestet).

Nat	Bei NAT/PAT-Paketen ist die im SIP-Header mitgelieferte IP-Adresse nicht gleich der öffentlichen IP. Mit NAT=yes wird nur die externe IP verwendet.
Permit	- keine Infos zu diesem Feld vorhanden -
Deny	- keine Infos zu diesem Feld vorhanden -
Mask	- keine Infos zu diesem Feld vorhanden -
Pickupgroup	Telefone können verbunden werden und von anderen Telefonen die Anrufe beim Klingeln holen. Hier wird die Gruppe von Telefonen definiert.
Port	Hier wird automatisch der Port des angemeldeten Telefons gespeichert.
Qualify	- keine Infos zu diesem Feld vorhanden -
RestrictID	- keine Infos zu diesem Feld vorhanden -
RTPtimeout	- keine Infos zu diesem Feld vorhanden -
RTPholdtimeout	- keine Infos zu diesem Feld vorhanden -
Secret	Passwort für das Telefon.
Type	User: Ankommende Anrufe, Peer: Gehende Anrufe, Friend: beides.
Username	Der Name des Telefons, hier: Rufnummer.
Disallow	All: Ersteinmal alle Codecs verbieten, um sie später einzeln freizugeben.
Allow	Bestimmte Codecs freigeben: G.711, GSM, iLBC etc.
Musiconhold	Jeder User kann seine eigene Wartemusik speichern.
Regseconds	Zeitpunkt in Epoch-Sekunden (seit 1970), der die nächste Registrierung angibt. Liegt dieser Wert in der Vergangenheit, so ist das Telefon offline.
Ipaddr	IP-Adresse des Telefons, bei NAT: Externe IP.
Regexten	- keine Infos zu diesem Feld vorhanden -
Cancallforward	Info, ob das Telefon per SIP-Kommando weiterleiten darf.

Tabelle 4.2: Erläuterung der Datenbankfelder von Tabelle sipfriends

4.5.4 IAX2-Konfiguration (iax.conf)

Zur Kopplung von Asterisk-Servern mit Satelliten (Zweigstellen etc.) empfiehlt sich das IAX2-Protokoll, da es im Gegensatz zu SIP in beliebige Kontexte springen kann, recht firewallfreundlich durch Verwendung nur eines Ports ist und zudem wesentlich weniger Probleme mit NAT bereitet. Die Konfigurationsparameter können auch hierbei in der Datenbank abgelegt werden. Eine Verwendung von IAX2 findet in dieser Version allerdings noch nicht statt, was jedoch nur zeitliche Gründe hat. Technisch gesehen macht IAX besonders bei der Provideranbindung Sinn. Die Konfiguration ist recht analog zu SIP.

4.5.5 Dialplankonfiguration (extensions.conf)

Einen Großteil der Asterisk-Konfiguration macht die extensions.conf aus, denn hier werden die Dialpläne abgelegt, d.h. jeder Anruf, ob Inbound oder Outbound, durchläuft den Dialplan. Bei neueren Asterisk-Versionen ist es nicht mehr notwendig, bei Verwendung des Realtime-Modus sämtliche Teile der Konfiguration in die extensions.conf abzulegen, so dass diese Datei nur noch die Verweise auf die Familien der Extensions beinhalten muss.

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

[sippool]                ; Kontext für den Dialplan
switch => Realtime/@sippool ; Verweis auf die Familie sippool in der Datenbank
                           ; Die Zuordnung Familie => Tabelle geschieht in
                           ; der Datei extconfig.conf
include => parkedcalls    ; Optional ist es möglich, die Rufnummer 700 (Block ab 700)
                           ; einzubinden, um ein Parken von Anrufen zu ermöglichen.
                           ; Die Extension parkedcalls ist hierbei ein Schlüsselwort
                           ; und liegt hier nicht als definierte Extension vor.
```

4.5.6 Dialplan-Datenbank (zu extensions.conf)

Die Einträge in der Tabelle Dialplan in der Datenbank asterisk steuern die Wege der Anrufe. So fängt ein ankommender Anruf immer in Priorität 1 an und läuft, bis auf eine Ausnahme, jeweils um einen Schritt weiter. Jeder Eintrag in der Datenbank ist dabei einer Priorität, also einem Schritt zugeordnet. In einer Priorität kann eine Aktion erfolgen, d.h. ein Dial-Kommando, ein Aufruf eines AGI-Skripts, die VoiceMail, eine Ansage oder ein anderes der derzeit insgesamt 154 Applikationen, die Asterisk beinhaltet. Ist Priorität n abgearbeitet, wird die Priorität n+1 angesprochen. Diese muss definiert sein, ein Sprung bei Fehlen einer Priorität ist derzeit nicht in Asterisk vorgesehen und resultiert in einem Fehler, die Abarbeitung weiterer Prioritäten ist dann nicht mehr gegeben. Somit muss genau darauf geachtet werden, keine Ziffer in den Prioritäten auszulassen. Ein Kommando in einer Priorität, das nicht mehr genutzt wird, kann damit nicht einfach aus der Datenbank gelöscht werden, da dies eine Lücke darstellen würde. Um nicht alle folgenden Prioritäten um eins subtrahieren zu müssen, kann ein NoOp an diese Stelle eingefügt werden, das als „No Operation“ ein guter Platzhalter ist.

Struktur der Tabelle dialplan in der Datenbank asterisk:

Field	Type	Null	Key	Default	Extra
context	varchar(20)		PRI	sippool	
exten	varchar(64)		PRI		
priority	int(2)		PRI	1	
app	varchar(20)				
appdata	varchar(255)	YES		NULL	
descr	text	YES		NULL	
flags	int(1)			0	

Ein Dialplan für die Rufnummer 7001, bei dem die VoiceMail für besetzt und nicht erreichbar eingerichtet ist, sieht folgendermaßen aus:

context	exten	priority	app	appdata
sippool	7001	1	Dial	SIP/\${EXTEN} 20
sippool	7001	2	VoiceMail	u\${EXTEN}
sippool	7001	3	Hangup	NULL
sippool	7001	102	VoiceMail	b\${EXTEN}
sippool	7001	103	Hangup	NULL

In Priorität 1 wird 20 Sekunden lang versucht, über das SIP-Protokoll den User \${EXTEN} zu erreichen. Dabei ist in der Variable \${EXTEN} immer der Wert der aktuellen angerufenen Nummer (Extension) gespeichert, so wie es auch im Feld exten der Datenbank steht, in diesem Fall also die 7001. Statt der Variable dürfte hier durchaus auch der feste Wert „SIP/7001|20“ mit dem gleichen Resultat eingetragen werden. Wird der Ruf nicht innerhalb der 20 Sekunden angenommen, so springt der Dialplan auf Priorität 2, mit der die Voicemail mit dem Parameter u\${EXTEN} aufgerufen wird. Der Buchstabe „u“ vor der Extension veranlasst die Voicemail dazu, einen Nichterreichbarkeitstext auszugeben (unavailable). In Priorität 3 wird schliesslich das Gespräch beendet. Der Sonderfall in den Dialplanprioritäten fällt mit der Priorität 102 auf, die scheinbar schon wegen der großen Lücke zwischen 3 und 102 nie angesprochen wird. Initiiert wird der Sonderfall vom Dial-Kommando, das bei einem Besetztsignal nicht auf die Priorität n+1, sondern n+101 springt. Der Parameter b\${EXTEN} für die Voicemail auf Priorität 102 hat somit den Hintergrund, dass dem Anrufer dort eine Besetztansage (busy) ausgegeben wird. Auch hier beendet in der nächsten Priorität ein Hangup das Gespräch. Die Dialpläne funktionieren in diesem Fall auch ohne das explizite Hangup am Ende, doch wird ansonsten ein Fehler in das Log geschrieben, da der Dialplan unkontrolliert beendet wurde. Nicht nur daher ist die Variante mit Hangup am Ende des Gesprächs sauberer. Der Dialplan kann insgesamt beliebig komplex werden, es sind Bedingungen möglich, DTMF-Eingaben können z.B. zur Passwortabfrage getätigt werden. Nicht zuletzt ist es möglich, eigene Skripte zu starten, die mit Asterisk von der Shell-Ebene aus kommunizieren. Ein solches Skript findet hier in Form des Perl-Skriptes astcc.agi Anwendung.

4.5.7 extconfig.conf

Die Datei extconfig.conf verknüpft Familien mit Datenbanken und Tabellen. Hierbei gilt folgende Syntax:

Familie => Datenbanktyp, Datenbankname, Tabellename

Als Datenbanktyp sind zulässig: mysql, pgsql und odbc. In diesem System wird eine MySQL-Datenbank verwendet. Ähnlich den Extensions in der Dialplankonfiguration existieren auch hier einige vordefinierte Familien: Die Familien sippeers und sipusers definieren die angeschlossenen Endgeräte wie auch die Carrier, an die im Festnetz terminiert wird, voicemail hingegen steht für das Asterisk-interne Sprachmailboxsystem. Username, Passwörter und der Host der MySQL-Datenbank finden sich in der Datei res_mysql.conf.

```
[settings]
;Usertabelle (aus sip.conf)
sippeers => mysql,asterisk,sipfriends
sipusers => mysql,asterisk,sipfriends

;Dialplantabelle (aus extensions.conf)
sippool => mysql,asterisk,dialplan

;Voicemail
voicemail => mysql,asterisk,voicemail_users
```

4.5.8 res_mysql.conf

In dieser Datei werden die "Low-Level"-Daten für die Datenbank abgelegt, d.h. IP-Adresse der Datenbank, Username und Passwort, zudem noch der Port oder, falls die Verbindung über einen lokalen Socket geleitet wird, die Datei hierzu angegeben. Sofern die Datenbank auf dem gleichen System liegt (dbhost = localhost), verwendet Asterisk aus Gründen der Geschwindigkeit automatisch den Socket. Die Einstellungen für dbname und dbport werden in dem Fall ignoriert.

```
[general]
dbhost = localhost
dbname = asterisk
dbuser = asterisk
dbpass = mySecret
dbport = 3306
dbsock = /var/run/mysqld/mysqld.sock
```

4.5.9 cdr_mysql.conf

Die Call Detail Records (CDRs) müssen nicht zwingend in der gleichen Datenbank lagern wie die Konfigurationsdateien. Besonders bei sehr großen Callvolumina und damit verbundenem hohen Aufkommen an CDRs kann es durchaus nützlich sein, diese auf ein anderes Datenbanksystem zu speichern. Daher existiert analog zu res_mysql.conf auch diese Datei für die Logdaten. Zusätzlich existiert jedoch eine Zeile, in der das Userfield eingeschaltet werden kann. Damit ist es möglich, in einem weiteren Feld der CDRs noch eigene Informationen zu speichern.

```
[global]
hostname=localhost
dbname=asterisk
table=cdr
password=mySecret
user=asterisk
port=3306
sock=/var/run/mysqld/mysqld.sock
userfield=1
```

4.5.10 Astcc-Skript (Prepaid)

Wird das System als Provider genutzt, ist eine Gesprächsabrechnung der Nutzer notwendig. Um das System nicht zu komplex zu gestalten, sind nur die Basisfunktionen enthalten, d.h. die Gespräche werden im Sekundentakt nach der Destination aus der Tariftabelle abgerechnet und dem Prepaid-Konto belastet. Sämtliche Gespräche, die zu einem Ziel mit einer definierten Vorwahl in der Tariftabelle geleitet werden,

kosten den dort abgelegten Minutenpreis. Um dieses Feature zu nutzen, sind die Tarife jeweils von der Voreinstellung 0,00 Cent auf beliebige Werte zu setzen. Damit ist bei der Voreinstellung ein Guthaben auf den Prepaid-Karten per Default nicht notwendig, da keine Destination kostenpflichtig ist.

Das Skript `/var/lib/asterisk/agi-bin/astcc.agi [astcc]` wird bei einem möglicherweise kostenpflichtigen Gespräch zu einer Destination über die Konfigurationsdatei `/etc/asterisk/extensions.conf` aufgerufen. Die Bedingung zur Unterscheidung von internen und externen Anrufen wird hier auf Rufnummern mit mehr als 4 Ziffern gelegt, was nur dann funktioniert, wenn die internen Rufnummern auch tatsächlich nicht mehr als 4 Ziffern haben. Notrufe über 110 und 112 wurden hier bewusst vernachlässigt, da noch nicht alle Provider diese Funktionalität unterstützen, die Lokalisierung aufwändig ist und zumindest derzeit für solche Fälle noch ein anderes Telefon präsent sein sollte. Eine Weiterleitung dieser Nummern zu einem Provider ist jedoch mit wenig Aufwand zu realisieren (z.B. Destination 11 im Dialplan und explizites Routing der 3-stelligen Nummern in der `extensions.conf`).

Das Astcc-Skript ist lediglich eine AGI-Datei (`astcc.agi`, abgeleitet von CGI-Skript), benötigt neben zwei Konfigurationsdateien (`/var/lib/astcc/astcc-config.conf` und `users-astcc-config.conf`) noch Perl-Module, die unter Debian per `apt-get` geholt werden können (`libdbd-mysql`). Alternativ lässt es sich auch per CPAN einbinden:

```
shell> perl -MCPAN -e shell
cpan> install DBI
cpan> install DBD:mysql
```

Bei der Installation ist es einfacher, das aktuelle `libdbd-mysql` per `apt-get` zu holen, da dies automatisch möglich ist und immer die aktuelle stabile Version erreicht wird. Das originale Astcc-Skript enthält einige Fehler, die in der hier vorliegenden Version gefixed wurden. Nachfolgend werden die Routinen des für VoIP on CD angepassten Astcc-Skripts erklärt.

4.5.10.1 sub load_config()

Mit dem Aufruf dieser Subroutine wird die Konfigurationsdatei mit den darin enthaltenen Defaultwerten in den Variablen eingelesen. Die in der Datei `/var/lib/astcc/astcc-config.conf` enthaltenen Variablen setzen Werte wie Datenbankname, Username und Passwort für die Datenbank usw.

4.5.10.2 sub connect_db()

Diese Routine stellt die Verbindung zur MySQL-Datenbank her und übergibt in `$dbh` den Handler der Datenbank. Falls bereits ein Handler existiert, wird die Datenbankanbindung wieder geschlossen.

4.5.10.3 sub msleep()

Das in `msleep` enthaltene `select`-Kommando ist normalerweise für das File-Handling zuständig. Hier wird jedoch lediglich der 4. Parameter (Timeout) genutzt und so eine Möglichkeit geschaffen, eine Wartefunktion zu realisieren, die als einzigen Parameter Millisekunden übergeben bekommt [`msleep`].

4.5.10.4 sub savedata()

Diese Unterprozedur hat gleich zwei Funktionen. Einmal speichert sie bei Übergabe geeigneter Parameter die Information, ob eine bestimmte Prepaidkarte (Account) gerade genutzt wird, um ggfs. eine weitere gleichzeitige Nutzung zu unterbinden, was aber das hier angepasste Astcc-Skript nicht weiter auswertet. Die zweite Funktion besteht darin, nach erfolgreichem, kostenpflichtigen Telefonat das verbleibende Guthaben der Karte in die Datenbank zurück zu schreiben.

4.5.10.5 sub getcard()

Zu einer von Asterisk übergebenen Kartenummer (Account) werden hier zugehörige Parameter wie z.B. das Restguthaben oder die Information, ob die Karte noch gültig ist (wird hier nicht verwendet) aus der Datenbank ausgelesen und zurückgegeben.

4.5.10.6 sub savecdr()

Zu jedem erfolgreichen oder auch erfolglosen Gespräch wird ein Datensatz (CDR, Call Detail Record) zusammen mit sämtlichen Parametern wie Callstatus (angenommen, besetzt, abgebrochen, keine Antwort, Gassenbesetzt), Zeitstempel und Dauer gespeichert.

4.5.10.7 sub mystreamfile()

Als Parameter erwartet diese Subroutine den Namen einer Datei in /var/lib/asterisk/sounds/ (falls die Localizer-Funktion genutzt wurde, finden sich die Sounds z.B. in deutsch in /var/lib/asterisk/sounds/de/) und gibt dieses Soundfile aus. Hiermit kann z.B. eine Ansage für das verbleibende Guthaben (mit mysaynumber) oder ein Grund für das Nichtzustandekommen eines Gesprächs angesagt werden.

4.5.10.8 sub mysaynumber()

Diese Routine arbeitet ähnlich zu mystreamfile(), erwartet als Parameter jedoch einen numerischen Wert und gibt diesen gesprochen aus.

4.5.10.9 sub tell_time()

Die Bezeichnung tell_time für diese Prozedur ist etwas irreführend. Neben dem Setzen von im weiteren Verlauf der Abarbeitung benötigten Variablen, wird hier unter Berücksichtigung von Schwellwerten des Quiet-Levels (ähnlich einem Verbose-Level, wird am Anfang des Skriptes gesetzt) das verbleibende Guthaben angesagt. Eine Trennung von gesetzten Variablen und der Ansage in zwei Prozeduren ist hier zumindest in einer späteren Version sinnvoll.

4.5.10.10 sub saycost()

Wie der Name schon sagt, hört der Anrufer (A-Teilnehmer) abhängig vom Quiet-Level eine je nach Detailgrad eingestellte Ansage zu den Verbindungskosten.

4.5.10.11 sub mysayfloat()

Mysayfloat wird für die Ansage von Fließkommazahlen, wie sie in diesem Fall bei den Kosten pro Minute vorkommen können, genutzt und sagt somit als Unterfunktion von saycost() die Kosten pro Minute der aktuellen Verbindung an. Auch eine mögliche Verbindungsgebühr ist hier berücksichtigt, die jedoch in der derzeitigen Version von VoIP on CD noch nicht genutzt wird.

4.5.10.12 sub getphone()

Durch den Vergleich des Anfangs der gewählten Rufnummer mit den gespeicherten Destinationen wird die Route über eine Datenbankabfrage ausgewählt. Da hier die gespeicherten Länderpräfixe (wie z.B. 49 für Deutschland) mit der gewählten Nummer verglichen werden, die Abfrage also genau umgekehrt zu einer gewöhnlichen SQL-Abfrage mittels SELECT .. LIKE gestellt wird, kommt eine Konstellation von RLIKE und concat zum Einsatz, um dies zu realisieren.

4.5.10.13 sub trytrunk()

Nachdem die Übertragungstechnik (SIP, IAX2, ZAP (ISDN) oder Local) wie auch der Pfad (meist identisch zum Namen) aus der Datenbank ermittelt wurde, gibt diese Subroutine abhängig von der Technik an den als Parameter übergebenen Trunk (Verbindung zu einem Carrier wie z.B. Sipgate) den passenden Dial-Befehl heraus.

4.5.10.14 sub calccost()

Die Zusammenfassung der Kosten für ein gerade geführtes Gespräch findet in dieser Subroutine statt. Neben einer evtl. vorhandenen Verbindungsgebühr (wird hier nicht genutzt) zählt die Routine auch die Minutenpreise für die Dauer des Gesprächs zusammen.

4.5.10.15 sub checkinuse()

Sofern die Prepaid-Karten nicht mehrfach gleichzeitig genutzt werden sollen, ermittelt diese Routine die dafür notwendigen Bedingungen und gibt ggfs. eine Sprachmeldung aus, in der mitgeteilt wird, dass auf einem Account nur ein Gespräch gleichzeitig stattfinden darf. Sinnvoll ist ein solches Vorgehen dann, wenn mit einem Betrugsversuch zu rechnen ist, da beim Start der Gespräche die Zeit für ein einziges Gespräch zur

ausgewählten Destination für das Restguthaben ausgerechnet wird und ein Anrufer somit bei gleichzeitigen Telefonaten sein Prepaid-Konto in den negativen Bereich bringen kann, was dem Prepaid-Gedanken an sich widerspricht.

4.5.10.16 sub checkexpired()

Sollte die Prepaid-Karte eine Gültigkeitsdauer eingestellt haben, so prüft diese Routine, ob die Karte noch gültig, d.h. der Gültigkeitszeitraum noch nicht überschritten ist. Andernfalls wird unter Berücksichtigung des Quiet-Levels ggfs. eine dementsprechende Ansage ausgegeben und der Anrufversuch beendet.

4.5.10.17 sub setfirstuse()

Bei Verwendung der Gültigkeitsdauer einer Karte startet die Zählung auf Wunsch des Betreibers erst nach Beginn des ersten Gesprächs, also erst, wenn die Karte auch tatsächlich genutzt wird. Hierzu wird in der Datenbank die Startzeit gespeichert. Anschliessend wird die Subroutine setexpiration() aufgerufen.

4.5.10.18 sub setexpiration()

Wünscht der Betreiber den Anfang der Zählung einer Ablaufdauer einer Karte erst nachdem diese das erste Mal genutzt wurde, schreibt diese Routine die Ablaufzeit in die Datenbank.

4.5.10.19 sub setinuse()

Diese Routine erhält als Parameter eine Kartenummer (Accountnummer) und ein inuse-Flag und setzt oder löscht die Variable inuse im Array der Karte.

4.5.10.20 Hauptteil

Im Hauptteil des Astcc-Skripts werden die Subroutinen nach Bedarf eingesetzt. Nach der Prüfung auf eine hergestellte Datenbankverbindung ist es möglich, zur Authentisierung einen Pin-Code einzugeben, sofern kein Account (keine Prepaid-Karte) ermittelt werden kann. Diese Funktionalität wird hier jedoch nicht genutzt, da immer ein Account übergeben wird. Sollte der Account über ausreichend Guthaben verfügen und die Gültigkeitsdauer noch nicht überschritten sein, wird die zulässige Höchstdauer abhängig vom Guthaben ausgerechnet und die für die ermittelte Route zu nutzenden Trunks angesprungen. Sofern der erste Trunk keinen Erfolg bringt, wird der nächste ausprobiert, sofern dieser definiert ist. Bricht das Gespräch ab oder kommt es nicht zustande, so wird der CDR (Call Detail Record) geschrieben. Bei einem erfolgreichen Gespräch findet zusätzlich ein Update des neuen verbleibenden Gesprächsguthabens statt.

4.6 Erstellung Weboberfläche

Die Interaktion des Administrators und ggfs. auch der User mit dem System geschieht über eine Weboberfläche, deren Handhabung besonders für Laien angenehmer und einfacher ist, als die Kommunikation über eine Kommandozeile. Alle wesentlichen Punkte sind hierüber einstellbar. Zur Strukturierung der Seiten werden Tabellen eingesetzt, da Frames wegen der schlechteren Indexierung durch Suchmaschinen und der fehlenden Möglichkeit, direkte Deeplinks einzusetzen, kein optimales Ergebnis erzeugt hätten.

Die in PHP4 geschriebenen Skripte folgen einer einheitlichen, objektorientierten Struktur. Separiert vom hauptsächlichen Inhalt (Body) werden am Anfang und am Ende die separaten Dateien für Header und Footer eingelesen. Um nicht mehrere Includes für Authorisierung, Navigation etc. an den Anfang einer jeden Datei stellen zu müssen, wird eine Initialisierungsdatei im Header-Include selbst eingebunden, so reicht es aus, Header und Footer zu nutzen. In der Init-Include-Datei werden die Variablen und Funktionen für Authentisierung, Darstellungshilfen oder Konstanten z.B. für die Datenbankanbindung eingelesen.

Die Kommunikation mit Asterisk geschieht über die MySQL-Datenbank. Dort werden User verwaltet oder Stati abgerufen. Lediglich für das Anlegen neuer Carrier wird in eine Datei geschrieben.

Eine einheitliche Programmierweise im EVA-Prinzip nach folgendem Schema gibt Programmierern die Gelegenheit, sich zügig in den Quellcode einzuarbeiten. Bedingungen werden, soweit mehr als zwei mögliche Fälle auftreten können, als CASE-Anweisung geschrieben, ansonsten einfacher als IF-Bedingung.

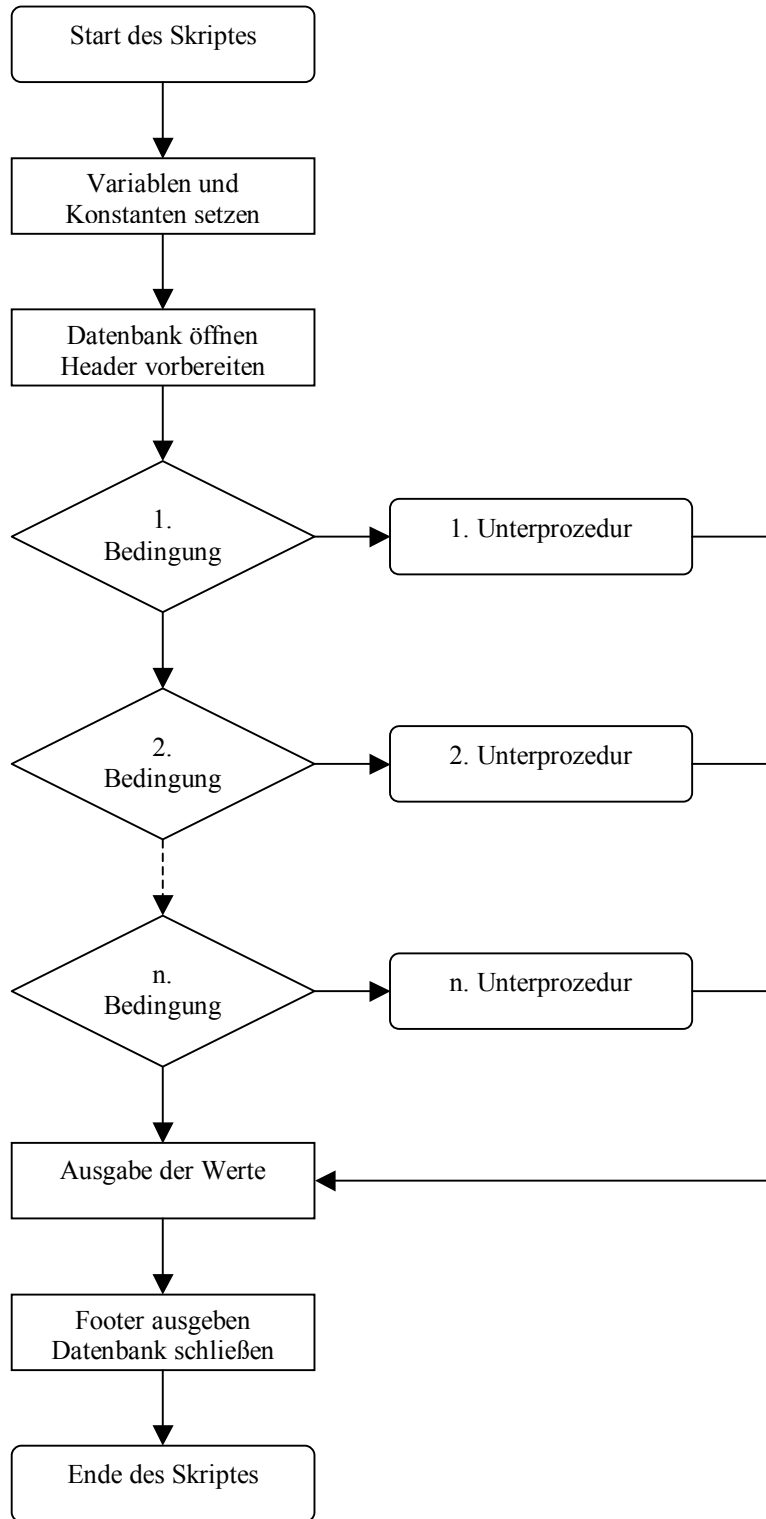


Diagramm 4.3: Struktogramm der Weboberfläche

4.6.1 Cascading Stylesheets

Die Formatierung der Ausgabe geschieht über Cascading Stylesheets, kurz CSS [css]. Hiermit ist es im Gegensatz zu reinem HTML möglich, sämtlichen Elementen der HTML-Ausgabe beliebige Formatierungsparameter wie z.B. Schriftgröße, Abstand zum nächsten Absatz oder spezifische Hintergrundbilder mitzugeben. Die Informationen können im HTML-Code enthalten sein, hier sind sie jedoch aufgrund der lesbareren Struktur in separate Dateien ausgelagert. Allein durch die Änderung der CSS-Dateien lässt sich die Ausgabe (Positionierung, Farben etc.) völlig umstellen. Sinnvoll ist diese Möglichkeit z.B. (ab CSS Version 2) beim Drucken von Texten der Webseite, da hier auch die Navigation ausgeblendet werden kann. Alle gängigen Browser unterstützen CSS. Eingebunden wird die CSS-Datei im Header mit folgender Zeile:

```
<link rel="stylesheet" type="text/css" href="inc/css/main.css">
```

Der Inhalt für die CSS-Datei kann beliebig komplex werden. Im folgenden wird ein Beispiel für einen CSS-Eintrag beschrieben.

```
.navi_links {
  text-decoration:none;
  color:#000000;
  font-weight:bold;
  font-style :normal ;
  font-size :11px ;
  font-family:Tahoma,Arial,Helvetica;
}
```

Wird die Klasse `navi_links` für die formatierte HTML-Ausgabe genutzt, so gelten hier folgende Parameter:

- Keine "Text-Dekoration", d.h. Links werden nicht unterstrichen, besuchte Links werden nicht andersfarbig markiert.
- Farbe wird auf #000000 gesetzt, d.h. sie ist schwarz.
- Text wird in Fettschrift ausgegeben.
- Der Textstyle ist auf normal gesetzt.
- Die Texthöhe beträgt 11 Pixel.
- Folgende Zeichensätze sollten, falls verfügbar verwendet werden: Tahoma, Arial, Helvetica. Sofern der Font Tahoma auf dem Client-PC nicht installiert ist, wird auf Arial ausgewichen usw.

4.6.2 Authentisierung durch Sessions

Nach Eingabe eines gültigen Paares von Username und Passwort, wird durch das Include `auth.inc.php` eine Session-Variable in der Datenbank gespeichert, die der weiteren Identifizierung dient, da diese bei jedem Seitenaufruf über ein Cookie in dem Array `$_SESSIONS` übergeben wird [sessions]. Der Inhalt der Session-Variable wird dabei immer mit dem Datenbankfeld `websession` in der Benutzerdatenbank verglichen. So ist auch ein Ausloggen einfach durch Löschen der Session möglich.

4.6.3 Administrationsoberfläche

Die Weboberfläche für den Administrator wird über folgende URL aufgerufen:

```
http://<IP oder Name des Servers>/admin
```

Der Default-Username lautet „admin“ und das zugehörige Passwort ist „rowi.net“. Diese Kombination aus Username und Passwort ist als Einstieg vorgegeben und für alle VoIP on CD-Installationen gleich. Daher sollte dieser Administratoraccount sofort nach dem Anlegen eines neuen Accounts gelöscht oder zumindest das Passwort dazu geändert werden, um unauthorisierte Zugriffe zu vermeiden.

Die Struktur der Adminoberfläche ist wie folgt aufgebaut:

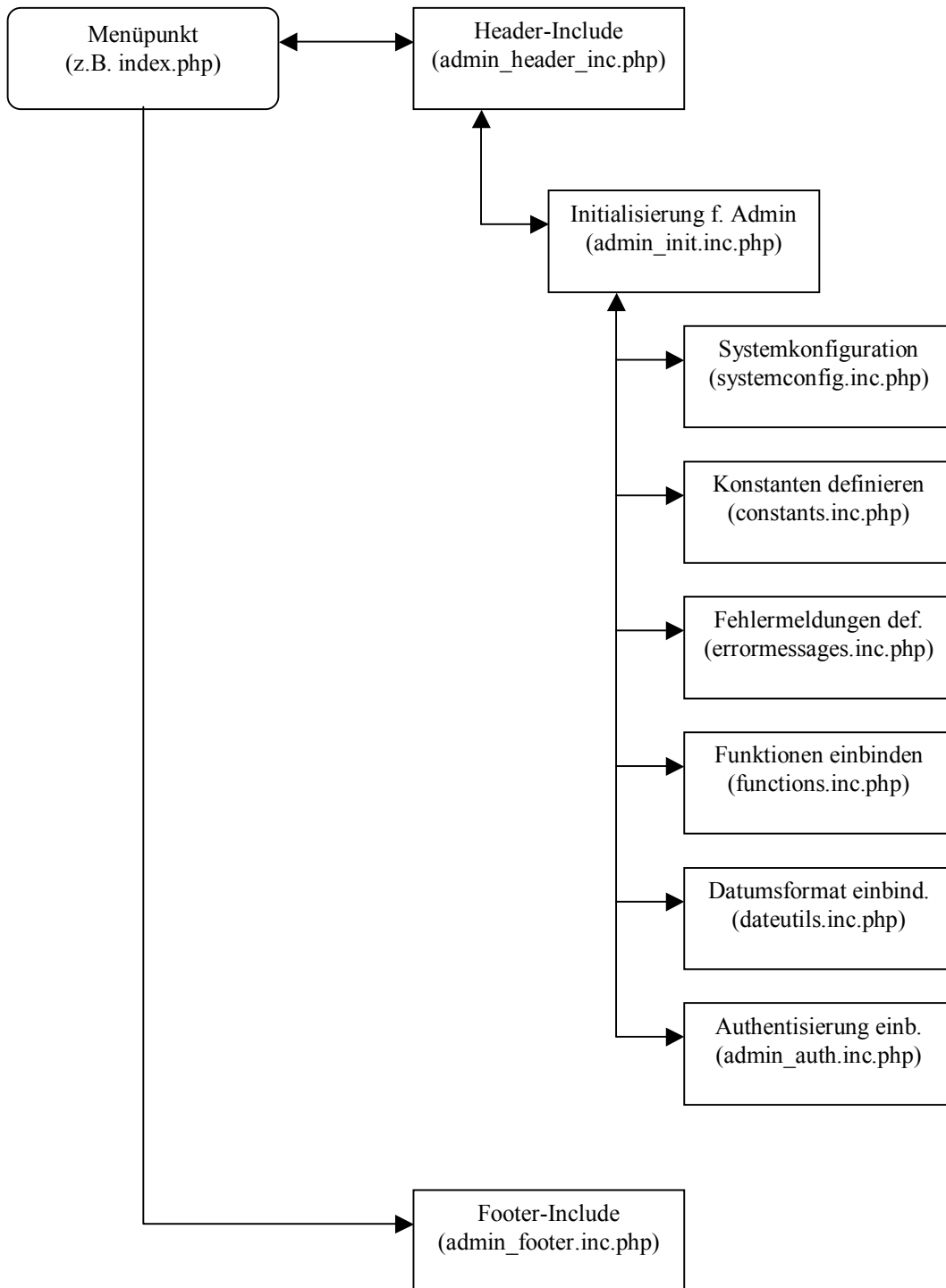


Diagramm 4.4: Struktogramm der Adminoberfläche

Innerhalb des Menüpunktes selbst sieht die Struktur wie folgt aus:

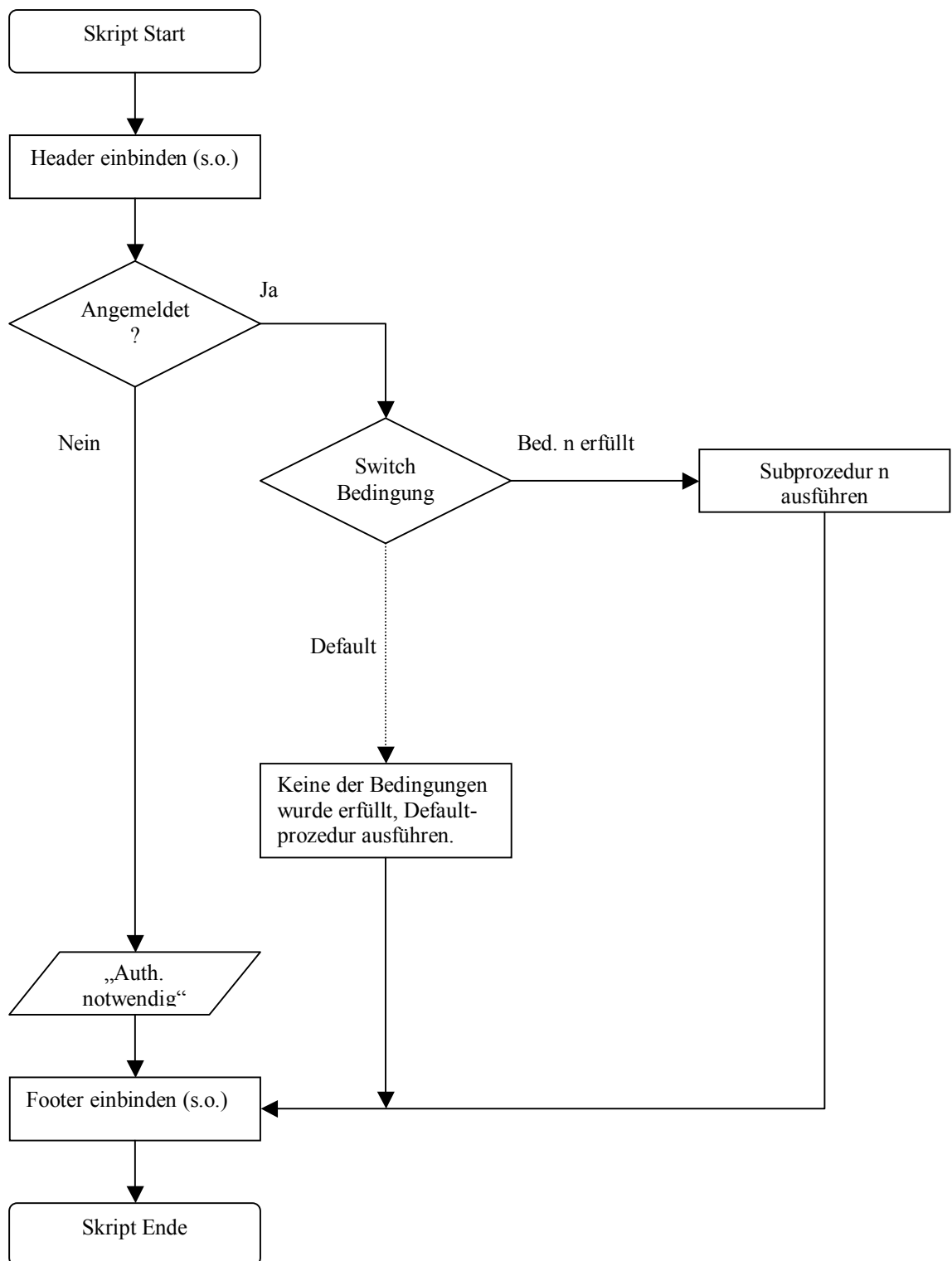
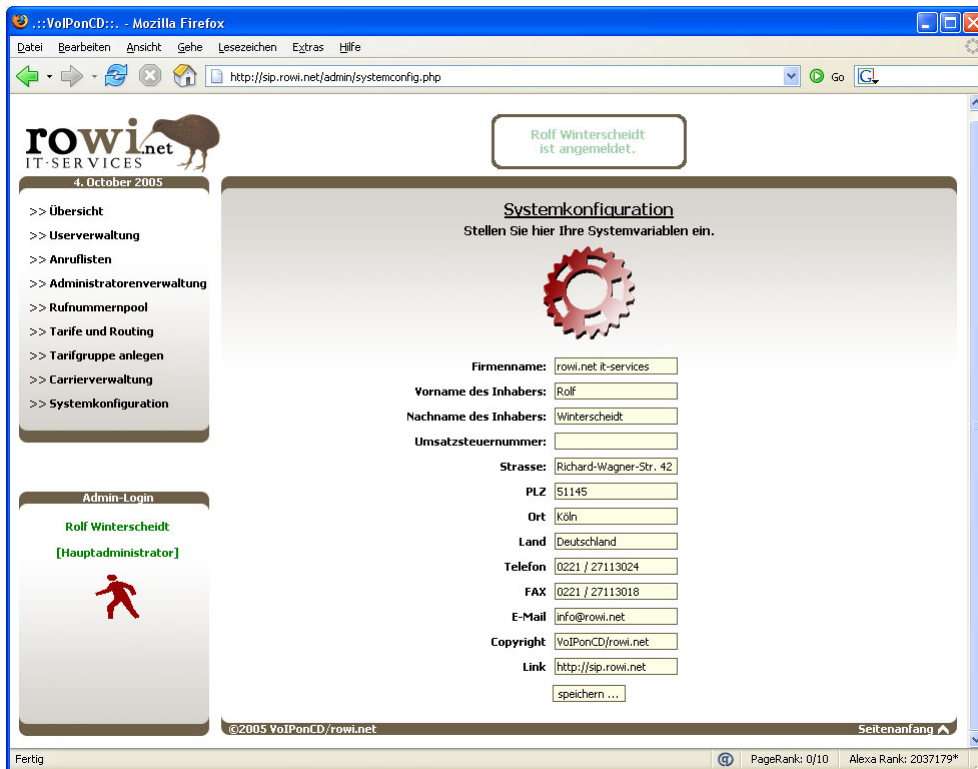


Diagramm 4.5: Struktogramm des Menüpunktes

Screenshot der Adminoberfläche:



Grafik 4.1: Screenshot der Adminoberfläche

4.6.3.1 Übersicht (index.php)

In der Übersicht / Statistik wird, wie der Name schon sagt, eine Statistik über sämtliche Gespräche und User angezeigt. Neben der Anzahl der registrierten Nutzer ist es interessant zu wissen, wie viele Telefone derzeit angemeldet sind, ggfs. welche Kosten (für die User) angefallen sind und für das Debugging, wann das letzte Gespräch stattfand oder welche Gespräche derzeit aktiv sind.

4.6.3.2 Userverwaltung (show_customer.php)

In der Userverwaltung ist es möglich, einen einzelnen User zu editieren, zu löschen oder das Guthaben für den Account aufzuladen. Auch ist eine Anzeige für den Onlinestatus des Telefons und die gesamten SIP-Daten zur Anmeldung sichtbar. Ein Kundensupport ist daher über eine einzige Seite möglich, ein Blättern über mehrere Seiten entfällt. Neues Guthaben muss manuell aufgeladen werden, ein komplettes Billing ist derzeit noch nicht implementiert.

4.6.3.3 Anruflisten (all_calls.php)

Auch die Anruflisten dienen als Hilfsmittel zu einer Störungsbeseitigung. Zu jedem bislang erfolgten Telefonat wird ein Datensatz inkl. Status ausgegeben (neueste zuerst). Werden hier z.B. für ein bestimmtes Ziel (Destination, z.B. Deutschland Mobilfunk) immer ein Besetzt ausgegeben, ist es naheliegend, dass die Route nicht in Ordnung ist. Eine Automatisierung dieser Tests wäre ein sinnvoller Schritt zur nächsten Version von VoIP on CD. So könnte bei überdurchschnittlicher Häufung von Besetzt-Fällen zu einem bestimmten Ziel eine Mail an den Admin gesendet werden.

4.6.3.4 Administratorenverwaltung (show_admins.php)

Die Administratoren werden in zwei Gruppen unterteilt: Administrator und Hauptadministrator. Letzterer kann neben der normalen Userverwaltung auch grundlegende Parameter wie URL des Systems, Name des Inhabers oder Steuernummer ändern, die für den normalen Betrieb nicht mehr angefasst werden müssen. In dieser Sektion lassen sich Administratoren wie auch Hauptadministratoren anlegen. Ein normaler Administrator ist auf die Userverwaltung und die Anruflisten beschränkt und kann z.B. keinen Hauptadministrator anlegen.

4.6.3.5 Rufnummernpool (numberpool.php)

Um einem User eine Rufnummer zuweisen zu können, muss diese erst einmal in den Rufnummernpool eingetragen werden. Normalerweise erhält ein Provider einen oder mehrere Rufnummernblöcke, die ihm zur freien Verfügung stehen. Diese Blöcke lassen sich bei der Bundesnetzagentur (ehem. RegTP) bestellen. Ein Routing findet damit noch nicht statt, dies muss gesondert eingerichtet werden. Im kleinen Rahmen können hier Nebenstellennummern vergeben werden, also z.B. 7000 bis 7099, um 100 Rufnummern einzugeben. Auch mehrere Blöcke sind möglich. In dieser Konfiguration ist es sinnvoll, die Blöcke nicht mehr als 4-stellig zu definieren, da in der extensions.conf alle Ziele, die länger als 4 Stellen sind, nach Extern geroutet werden.

4.6.3.6 Tarife und Routing (new_rates.php)

Es gibt insgesamt über 1000 verschieden tarifierte Destinationen. So kostet die Destination 49 (Deutschland Festnetz) deutlich weniger als 4915 bis 4917 (Deutschland Mobilfunk). Jeder Destination kann hier ein eigener Minutenpreis zugeordnet werden. Zur Vereinfachung sind nicht alle 1000 Tarife in der Datenbank aufgeführt, sondern sehr einfache Ziele wie "0" für nationale und "00" für internationale Gespräche. Zur Verdeutlichung findet sich auch die Destination "017" als Repräsentant für Mobilfunkgespräche wieder. Ein eingegebener VK-Preis (Verkaufspreis) wird gleichzeitig als Abrechnungsgrundlage für Gespräche herangezogen, während der EK-Preis (Einkaufspreis) derzeit nur eine informative Angabe ist, aber nicht weiter ausgewertet wird. Nach der Installation ist als Provider "provider1" eingetragen, der jedoch nicht existiert. Sobald jedoch unter dem Menüpunkt Carrierverwaltung ein Provider eingegeben wird, stellt das System sämtliche Destinationen auf diesen Provider um, so dass alle ausgehenden Telefonate über diesen geführt werden. Ebenso verhält es sich, wenn der vorletzte Provider gelöscht wird und nur noch ein einziger Provider im System existiert. Routen lassen sich über maximal drei Provider einstellen, d.h. reagiert der erste Provider nicht, wird der zweite genutzt. Reagiert auch dieser nicht, wird der dritte Provider genutzt. Hierbei wird davon ausgegangen, dass spätestens der 3. Provider eine funktionsfähige Verbindung ins Festnetz herstellen kann. Routen können jedoch auch zu Satelliten (Zweigstellen) hergestellt werden. Auch in diesem Fall kann diesem "Provider", also dem Satelliten eine Route zugewiesen werden. Es ist kein Muß, dass drei Provider definiert werden. Bei reinen internen Telefonaten nur über den VoIP on CD-Server ist kein Provider erforderlich, da dieser nur eine Querverbindung in ein anderes Netz herstellen würde.

4.6.3.7 Tarifgruppen/Tarifmodelle (new_brands.php)

Mit Tarifmodellen lassen sich Grundgebühren (z.B. monatliche Grundgebühr) oder aber Verbindungsentgelte einrichten. So können z.B. die ersten 60 Sekunden eines Gesprächs unabhängig von der Dauer berechnet werden, danach gilt der Sekundentakt. Jeder User kann einem bestimmten Tarifmodell zugeordnet werden. Dieses Verfahren ist hier jedoch nur vom Prinzip her verwirklicht, wird aber nicht aktiv genutzt.

4.6.3.8 Carrierverwaltung / Providerverwaltung (new_carrier.php)

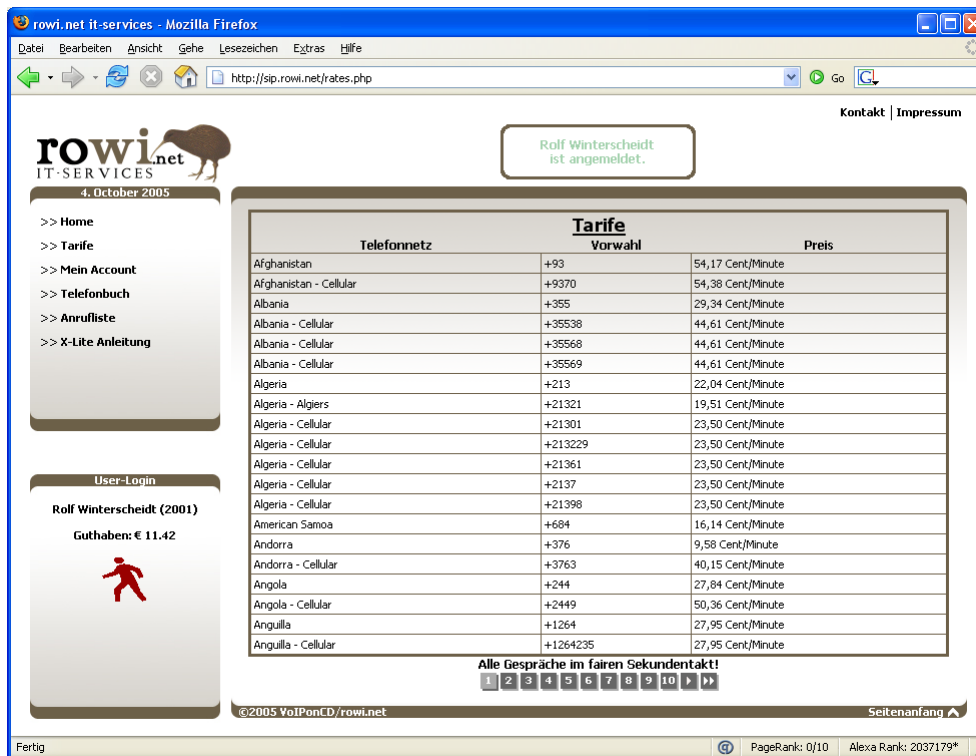
Ein neuer Carrier bzw. Provider wie z.B. Sipgate, Nikotel oder Wodatel lässt sich mit der Carrierverwaltung einrichten. Hierbei wird Benutzername, Passwort, Carrierhost, ein Kommentar (optional) und eine Weiterleitungsrufnummer eingegeben. Die Weiterleitungsrufnummer bestimmt, welcher internen Rufnummer Anrufe aus dem Festnetz zugeordnet werden sollen.

4.6.3.9 Systemkonfiguration (systemconfig.php)

Einem Hauptadministrator wird es hier besonders im Hinblick auf die Nutzung von VoIP on CD als Providersystem ermöglicht, den Firmennamen, Umsatzsteuernummer usw. einzugeben. Diese Werte werden z.B. im Impressum angezeigt.

4.6.4 Useroberfläche

Soll die CD als Providersystem eingesetzt werden, so kann den Benutzern ermöglicht werden, die Useroberfläche zu bedienen. Wie bei einem Providersystem üblich, kann sich ein User neu registrieren und so einen Account anlegen und kostenfrei intern telefonieren (auch extern, sofern die Destinationen auf 0 Cent Minutengebühr gesetzt wurden). Sämtliche Einstellmöglichkeiten existieren aber auch in der Administrations-Oberfläche. Die Struktur der Useroberfläche ist gleich der im Adminbereich.



Grafik 4.2: Ausschnitt aus der Useroberfläche

4.6.4.1 Home (index.php)

Die Startseite sollte die wesentlichen Vorzüge des Providers kurz und knapp erläutern. Das kann z.B. ein besonders günstiger Minutenpreis, eine im Vergleich bessere Sprachqualität oder der Wegfall der Grundgebühr sein. Dieses Alleinstellungsmerkmal wird hier hervorgehoben.

4.6.4.2 Tarife (rates.php)

Entscheidend für die Nutzung des Providers ist auch die Tarifliste. Da es über 1000 verschiedene Ziele gibt, werden die Tarife in der Datenbank gespeichert (Datenbank astcc, Tabelle rates). Eine Form der Nutzung der Tariftabelle ist die Ausgabe für den User, wobei hier natürlich nur der Verkaufspreis angezeigt wird.

4.6.4.3 Mein Account (myaccount.php)

Unter diesem Menüpunkt hat der User die Möglichkeit, sämtliche relevanten Daten seines Account anzeigen und editieren zu können. Somit kann dieser z.B. bei Umzug, Namensänderung oder Änderung der E-Mail-Adresse seine Daten selbst korrigieren, was den Administrationsaufwand minimiert. Auch die Daten des Telefonaccounts sowie der Anmeldestatus des Telefons werden angezeigt, was auch für die Fehlersuche wertvoll ist.

4.6.4.4 Telefonbuch (phonebook.php)

Neben anderen Telefonen lassen sich auch interne Rufnummern anwählen, unter denen bestimmte Informationen verfügbar sind, die dann ggfs. angesagt werden. So existieren derzeit Rufnummern für die Guthabenansage, die Möglichkeit, die Wartemusik probe zu hören oder die Voicebox abzuhören.

4.6.4.5 Anrufliste (callist.php)

Um die geführten Gespräche nachvollziehen zu können, existiert die Anrufliste. Bei einem höheren Gesprächsaufkommen kann diese recht unübersichtlich groß werden, so dass hier eine Unterteilung nach Monaten durchgeführt wird.

4.6.4.6 X-Lite Anleitung (xlite.php)

Zum Anschluss an einen SIP-Provider bietet sich das derzeit noch kostenfreie Programm X-Lite von Xten Networks an. Um dem User die Konfiguration möglichst einfach zu machen, werden in dieser Anleitung

nicht nur Beispieldaten verwendet, sondern, sofern der User angemeldet ist, seine persönlichen Daten in der Anleitung angezeigt.

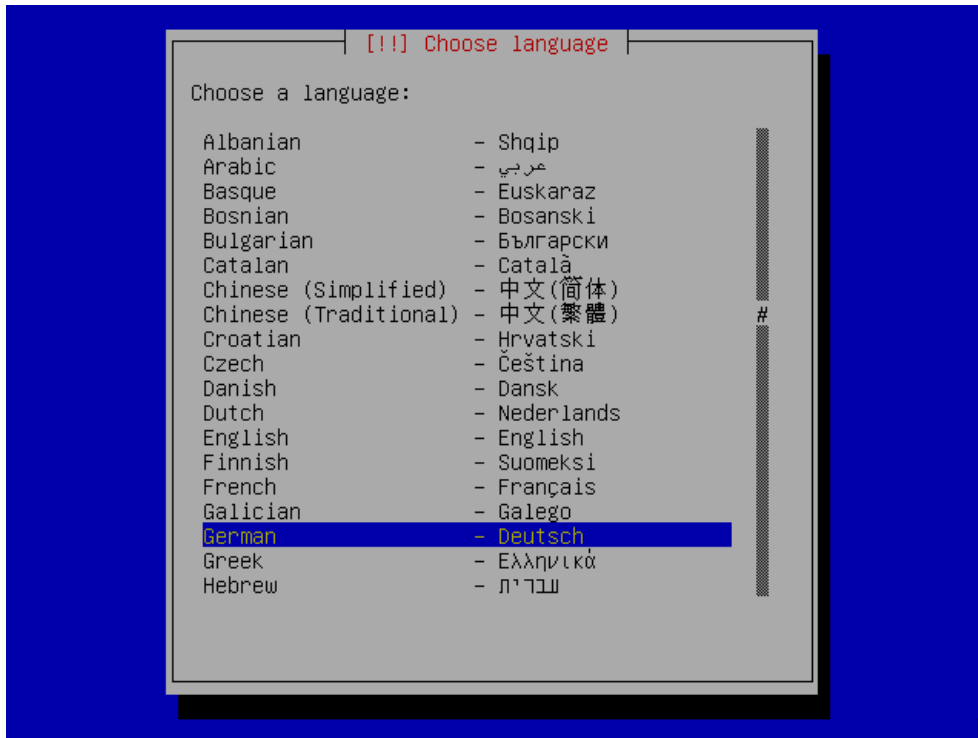
4.7 Installation von VoIP on CD mit der fertiggestellten CD

Aus Anwendersicht ist kein größeres PC- oder gar Linuxwissen notwendig. Grundlagen zu VoIP sind zwar sinnvoll, aber für die Einrichtung und den Betrieb des VoIP on CD-Systems nicht zwingend erforderlich. Benötigt wird ein PC (min. 500 MHz) mit einer angeschlossenen Internetanbindung (Ethernet) und einer Festplatte (min. ca. 1 GB) ohne installiertes Betriebssystem. Der PC muss von CD booten können, was aber auf fast alle Systeme zutrifft. Bootet der PC nicht von CD, ist im BIOS die Bootreihenfolge zu ändern. Im Folgenden wird davon ausgegangen, dass die IP-Adresse, Gateway usw. von einem DHCP-Server zur Verfügung gestellt wird, andernfalls werden diese Parameter noch abgefragt. Soll ein vorhandener DHCP-Server nicht genutzt werden, so ist beim Start der Installation der Netzwerkstecker zu entfernen. Dieser muss aber spätestens nach dem Auswurf der CD wieder angeschlossen sein. Nach dem Einlegen der CD und Anschalten des PCs ist folgender Bildschirminhalt zu sehen.



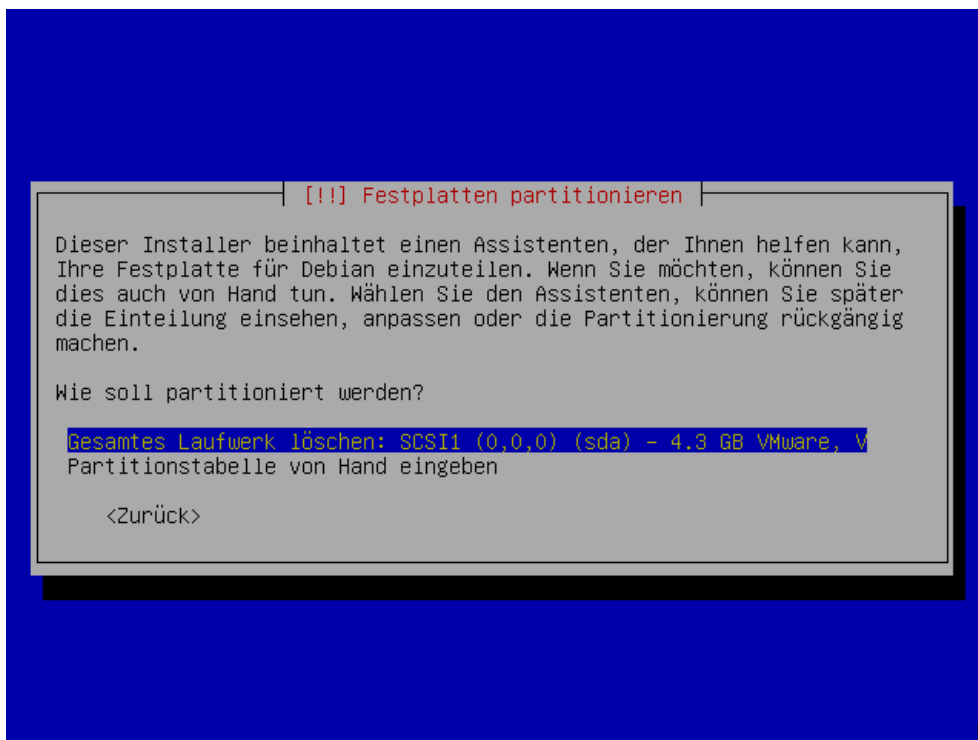
Grafik 4.3: System nach dem Booten von CD

Mit den Funktionstasten lassen sich weitere Informationen anzeigen, die Enter-Taste startet den Installationsvorgang. Kurz darauf fragt das System nach der Spracheinstellung, um anschliessend dem User weitere Informationen in seiner Landessprache zu geben und z.B. die Keymap der Tastatur anzupassen.



Grafik 4.4: Auswahl der Sprache

Nachdem einige Hilfsprogramme von CD geladen wurden, startet die Partitionierung der Festplatte. Hier ist es zwar möglich, selbst zu partitionieren, doch reicht es aus, die Voreinstellung durch Druck auf die Enter-Taste zu übernehmen. Neben einer Hauptpartition legt die Routine noch die obligatorische Swap-Partition an, die abhängig von der Größe des Hauptspeichers und der Festplatte automatisch gewählt wird.

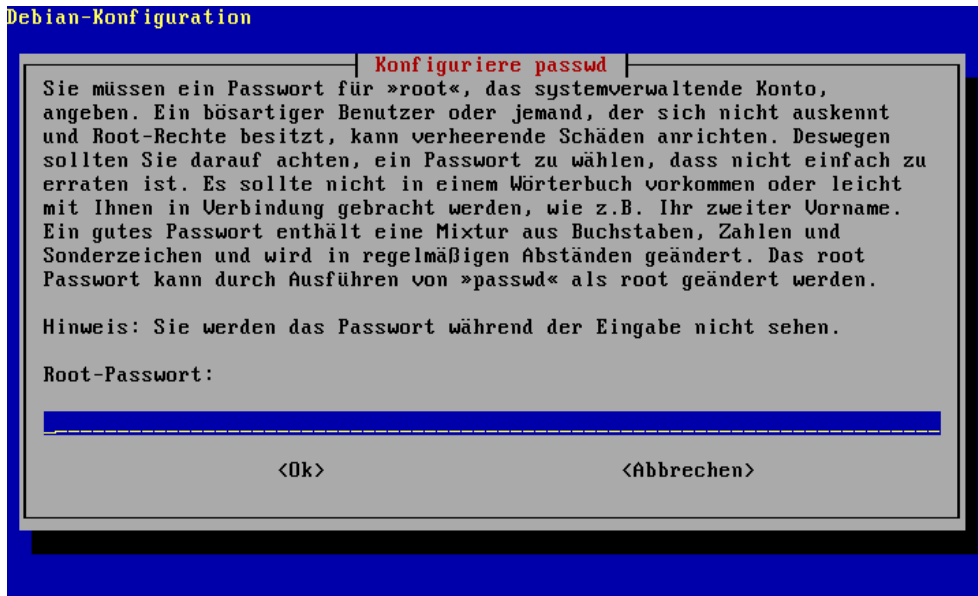


Grafik 4.5: Partitionierung der Festplatte

Sind die wichtigsten Programmteile installiert, wird die CD ausgeworfen und das System bootet neu. Bewußt wird hier auf eine Meldung zur Entnahme der CD und anschließender Bestätigung verzichtet, da Slot-In CD-Laufwerke oder jene in Notebooks oder Mini-ITX-Systemen nach dem automatischen Auswurf nicht wieder

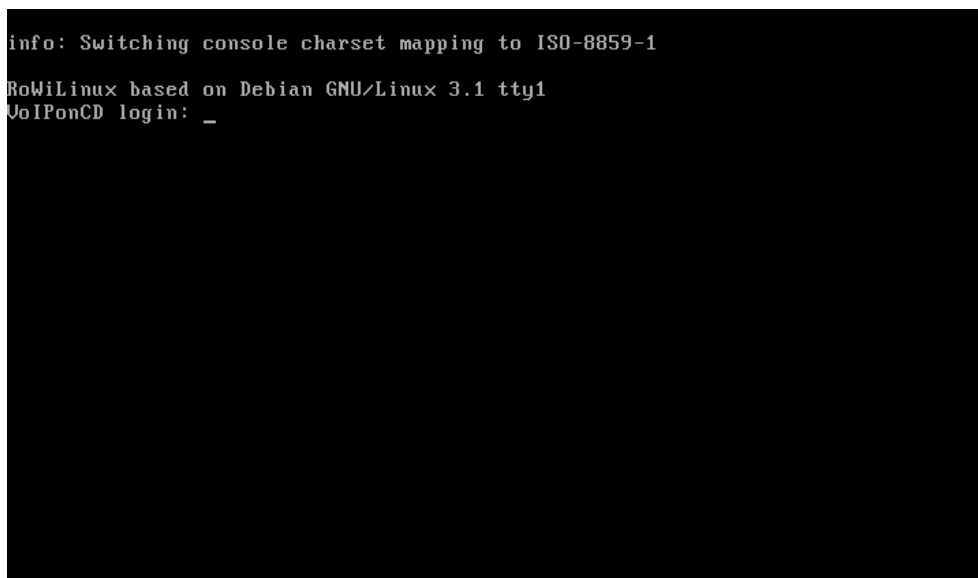
eingezogen werden und auch der Reboot völlig ohne Interaktion stattfinden kann. Bei herkömmlichen Laufwerken wird die CD beim Booten meist wieder eingezogen und endet am Bootprompt. In diesem Fall ist die CD manuell herauszunehmen und die Enter-Taste zu betätigen.

Nach kurzer Zeit wird das Root-Passwort des Systems eingegeben. Hier sollte die übliche Sorgfalt für Passwörter angewendet werden. Die Eingabe erfolgt nur einmal, daher ist es hier besonders wichtig, die Buchstaben korrekt einzugeben, da das System ansonsten später über die Konsole nicht mehr erreichbar ist.



Grafik 4.6: Eingabe des Root-Passwortes

Nun beginnt die eigentliche Installation der Softwarepakete, was je nach Leistung des PCs weitere rund 30 Minuten in Anspruch nimmt. Ist die Installation abgeschlossen, erscheint ein Boot-Prompt. Auf der Konsolenebene sind nun keine Aktionen mehr erforderlich, der Monitor für diesen Server kann nun entfernt werden.



Grafik 4.7: Konsole von VoIP on CD nach fertiggestellter Installation

Die Administration des Systems geschieht über einen Webbrowser an die ggfs. per DHCP vergebene IP-Adresse im Unterverzeichnis admin, also z.B. <http://192.168.1.42/admin>.

5 Kosten

Telefonieren innerhalb des Internet ist kostenlos, heißt es. Zumindest ein VoIP-Provider wird dem vehement widersprechen, was faktisch auch völlig korrekt ist.

5.1 Anschaffungskosten Hardware und Einrichtung

Ein kleiner Server ist schon für rund EUR 500 zu haben, zusammen mit der CD ergibt sich ein im Gegensatz zu herkömmlichen TK-Anlagen sehr günstiger Einstieg in die Telefonie. Wird gar ein ausgemusterter PC verwendet, so ergibt sich ein unschlagbares Preis/Leistungsverhältnis. Die Installation kann auch vom unbedarften Anwender selbst vorgenommen werden, Kosten für die Installation entstehen so auch nicht. Die Preise für IP-Telefone unterscheiden sich gegenüber vergleichbaren ISDN-Telefonen kaum. Sofern eine bestehende Netzwerkverbindung mitgenutzt werden kann, entstehen auch hier keine Kosten.

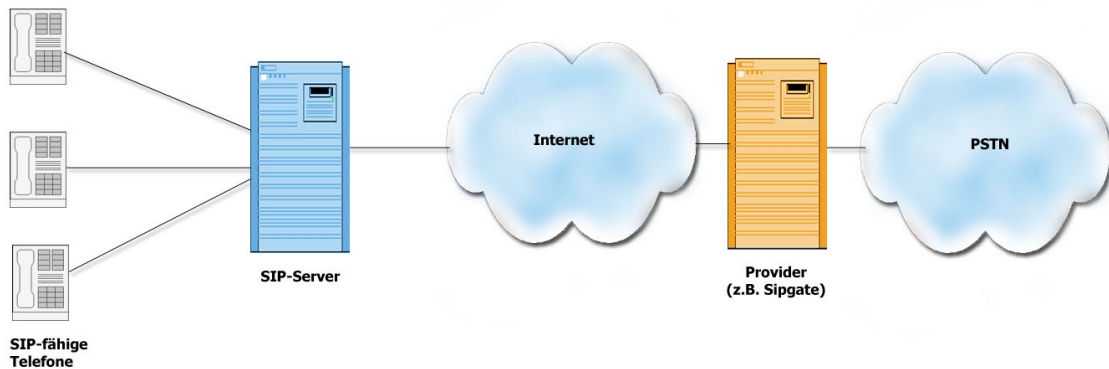
5.2 Betriebskosten

Auf lange Sicht sind neben den Anschaffungskosten die laufenden Kosten wichtiger. Die Befreiung vom Zwang, den vertraglich festgelegten Telefonanbieter nutzen zu müssen, eröffnet die Möglichkeit, das jeweils günstigste Angebot wahrnehmen zu können. Funktioniert das bei der T-Com auch per Call-by-Call, so müssen die kleineren Provider wie z.B. Netcologne das nicht zulassen. Über einen DSL-Anschluss sind hier vorerst keine Einschränkungen der Telefonie zu erwarten, da das Monopol für DSL-Anschlüsse nicht mehr bei einem Provider liegt. Neben den reinen Minutenpreisen ins Festnetz müssen auch Kosten für Strom und Wartung einkalkuliert werden, die jedoch hier fast vernachlässigbar niedrig gehalten werden können. Kostet eine Minute Internettelefonie über eine DSL-Flatrate bei Vernachlässigung der Stromkosten für das Telefon keine Gebühren, ist es z.B. bei Providern anders, da hier eine Flatrate unwirtschaftlich teuer wird. Kostet das GB Traffic EUR 0,19, so fallen pro Minute Telefonie rund 0,03 Cent an reinen Gebühren für Traffic an. Was für eine Minute noch vernachlässigbar wenig wirkt, sollte bei 1000000 Minuten (Providergröße) mit EUR 300 an Kosten schon in die Kalkulation einfließen.

6 Ergebnis / Fazit

Die CD lässt sich in mehrfacher Hinsicht nutzen: Ein minimalistisches Linux ist hiermit schnell und ohne mühsame Installationsabfragen aufgesetzt. Die zweite Stufe kommt auch mit einem bestehenden Linux zurecht, so dass VoIP on CD auch auf einem gemieteten Linuxsystem einsetzbar ist. Im Folgenden wird für verschiedene Einsatzweisen ein Fazit gezogen.

6.1 SOHO-System (1-Server-System ohne Satelliten)

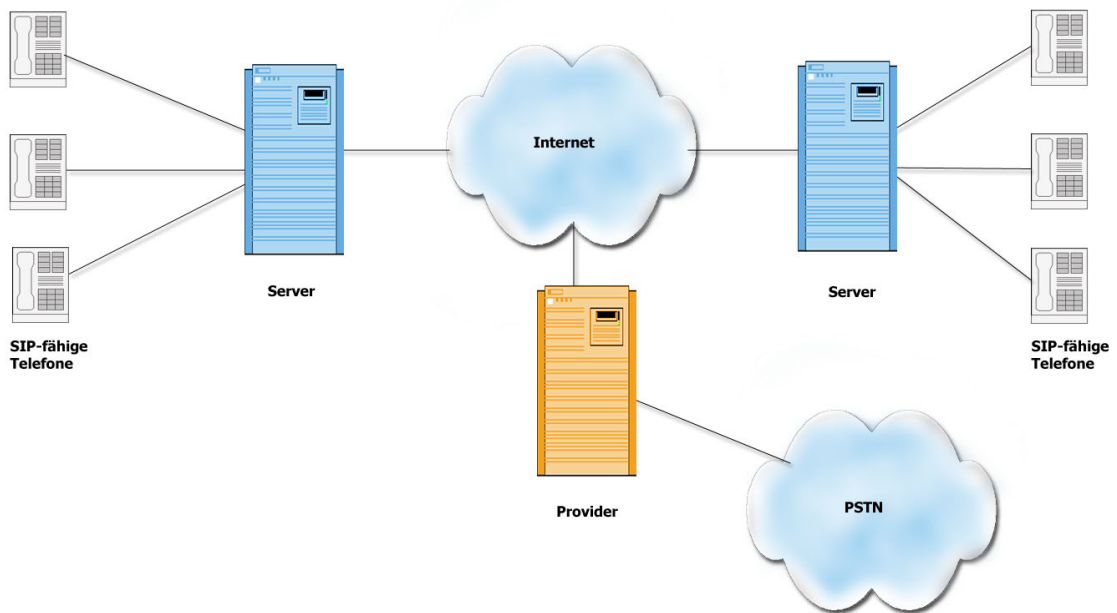


Grafik 5.1: VoIP on CD als SOHO-System

Als SOHO-System wird hier ein Einsatz bei „Powerusern“ im privaten Umfeld oder aber in kleineren Betrieben definiert. Hier kann die CD so eingesetzt werden, dass fast beliebig viele Telefone über eine einzige SIP-Anbindung über das Internet an einen Provider angeschlossen werden. Prinzipiell könnten sich die Telefone auch direkt am Provider anmelden, doch würde es durch das NAT/PAT schnell zu Problemen kommen, wie sich in der Praxis herausgestellt hat. Somit sollte prinzipiell bei Verwendung mehrerer Telefone ein Proxy wie z.B. VoIP on CD schon im lokalen Netz zwischengeschaltet werden. Doch die Proxy-Funktionalität ist natürlich nicht der einzige Vorteil. Es ist anzunehmen, dass die Telefone sich auch untereinander erreichen sollen, d.h. dass von einem Telefon aus ein anderes im gleichen lokalen Netz angerufen wird. Grundsätzlich kommt eine solche Verbindung über einen Registrar, d.h. SIP-Server zustande, es sei denn, es wird explizit die IP-Adresse des anderen Telefons angewählt. Mit einem SIP-Server im lokalen Netz bleibt die Verbindung lokal, gäbe es den Server nicht, so müssten auch hausinterne Gespräche über die Internetverbindung laufen und würden die Anbindung unnötig belasten. Nicht zuletzt ist auch die Verteilung eines ankommenden Anrufs auf die Telefone möglich, eine Rufnummer (z.B. von Sipgate) kann jedem der Telefone zugewiesen werden, auch ein Anrufbeantworter ist hier unabhängig vom Provider auch für interne Telefonate möglich.

Schon seit der Entwicklungsphase wird VoIP on CD erfolgreich so über einen 6 Mbit/s-DSL-Anschluss eingesetzt. Insgesamt 3 Telefone sind heute noch hierüber ständig online. Die Tarifstruktur ist auf 0 Cent pro Minute gesetzt, so dass hier kein Billing stattfindet, was auch nicht notwendig ist. Insgesamt gab es während der gesamten Dauer von etwa 3 Monaten bis heute keinen einzigen Ausfall.

6.2 Business-System (Mehrserver-System mit Satelliten)

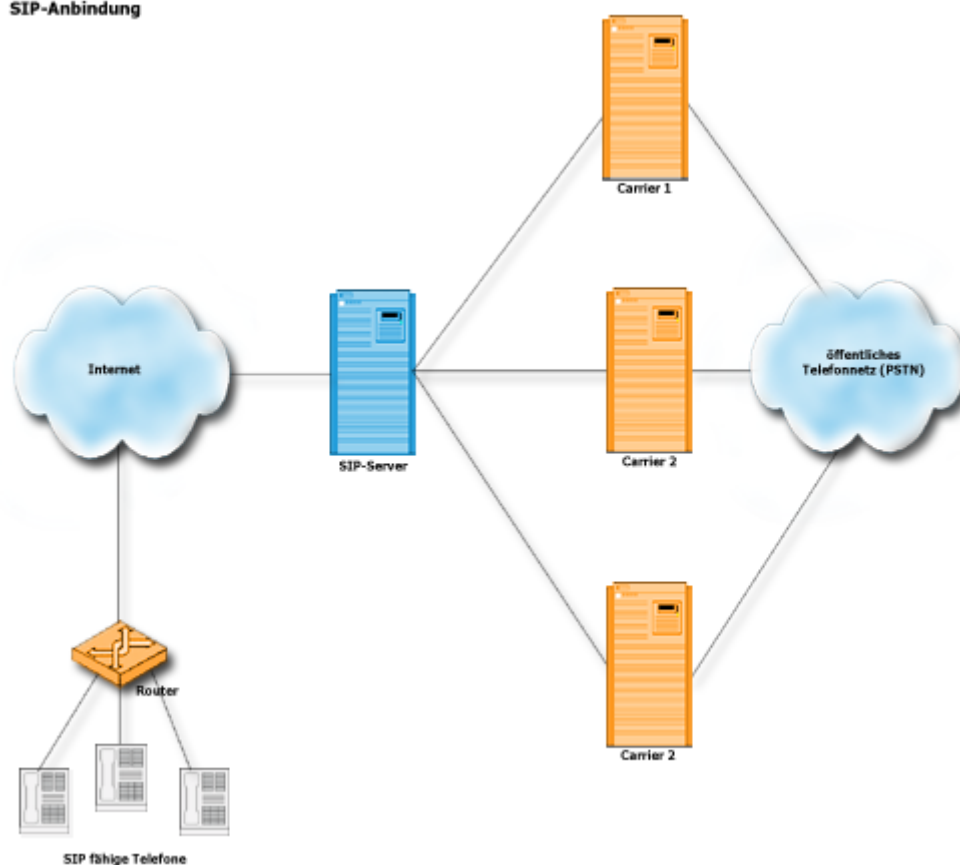


Grafik 5.2: VoIP on CD als Business-System

Durch das freie Routing ist eine direkte Route (Vorwahl 0001) zu einem anderen System, das ebenfalls mit VoIP on CD läuft, problemlos möglich. Die derzeit von Providern nicht genutzte Ziffernkombination 000 kann somit für ein Interconnect durch Anhang von z.B. zwei Ziffern mit bis zu 100 VoIP on CD Systemen verwendet werden, ohne das normale Routing ins Festnetz zu stören. Im Test gab es auch hier keine Probleme.

6.3 Provider-System (Anbieter von VoIP-Services)

SIP-Anbindung



Grafik 5.3: VoIP on CD als Provider-System

Auch als Provider-System lässt sich VoIP on CD einsetzen. Hier kommt erstmals das Billing (reines Prepaid, kein komplettes Billing) zum Einsatz, das verschiedenen Routen unterschiedliche Kosten zuweisen kann. Somit sind Gespräche (allerdings derzeit nur im Sekundentakt) abrechenbar. Aufladungen auf die Accounts geschehen (noch) händisch durch den Administrator. Neben den Kosten können jeder Route bis zu 3 Carrier zugewiesen werden, was dann Sinn macht, wenn ein Carrier keine gehenden Anrufe mehr zulässt (Ausfall des Servers etc.). Ein solches Konstrukt war die ursprüngliche Idee für die Erstellung von VoIP on CD und läuft bereits im produktiven Einsatz auf sip.rowi.net und diente gleichzeitig als Basis für eines der SIP-Systeme von Wodatel.com.

6.4 Weltweit verteiltes System

Bedingt durch die Anforderungen von Wodatel Inc. wurde VoIP on CD so konzipiert, dass es (in etwas abgeänderter Form) so einsetzbar ist, dass auf bereitgestellten Linux-Servern auf der ganzen Welt mit einem Einzeiler auf der Kommandozeile ein kompletter SIP-Server unter Asterisk eingerichtet werden kann. Dieser Einzeiler sieht in etwa so aus:

```
wget voiponcd.rowi.net/scripts.php -O script.sh && /bin/bash ./script.sh
```

Ein Skript wird über http per wget geholt und anschliessend ausgeführt. Standardisierte Systeme sind hierbei wichtig, um auch noch bei 100 Systemen den Überblick zu behalten. Fällt ein System aus, können andere Server die Last übernehmen, bis das ursprüngliche System wiederhergestellt oder ersetzt wird. Da auf solchen Servern keine Daten gespeichert werden, sondern diese rein per VPN mit einem Datenbankcluster, der ebenfalls im Internet verteilt aufgestellt wurde, kommunizieren, verliert der Provider keine wichtigen Daten (wie z.B. Billingdaten etc.), wenn ein Gerät komplett ausfällt. Hier werden somit nur sehr kleine Festplatten (effektiv unter 1 GB) benötigt.

7 To Do-Liste

Leider konnte nicht alles, was angedacht war, in der Diplomarbeit umgesetzt werden. Ausschlaggebend war hier allein die zur Verfügung stehende Zeit. Eine Weiterentwicklung von VoIP on CD wird es schon allein aufgrund der bisherigen positiven Resonanz geben. Da das System zum Teil erst bei der Installation vom VoIP on CD-Server geholt wird, könnten bereits einige dieser Punkte bereinigt sein, wenn VoIP on CD installiert wird, die CD altert somit von den Daten her kaum.

Überblick über die offenen Punkte der VoIP on CD mit Stand Oktober 2005:

- MySQL-User ist derzeit ohne Passwort.
- Kein HTTPS, damit keine sichere Übertragung von Username und Passwort.
- Wenn User Passwort vergisst, wird kein neues generiert.
- Kein Mailserver konfiguriert, um Mailbeachtigungen der VoiceMail zu schicken.
- Letzter Admin kann sich selbst löschen und so das System von der Weboberfläche nicht mehr administrierbar machen.

8 Quellenverzeichnis

[astcc]	http://asterisk.gnuinter.net/files/asterisk-perl-0.08.tar.gz
[codecs]	http://www.nwlab.net/art/voip/sip-telefone.html
[css]	http://de.selfhtml.org/css/
[exss]	Jan Exss, http://jan.exss.de/debian-cd-sarge.html
[IAX]	http://www.cornfed.com/iax.pdf
[IAXvsSIP]	http://www.voip-info.org/wiki/view/IAX+versus+SIP
[itu]	http://www.voip-info.org/wiki/view/QoS
[mos]	http://www.testyourvoip.com/faq.html#2
[msleep]	http://perldoc.perl.org/functions/select.html
[RFC????]	http://www.faqs.org/rfcs/rfc????.html
[sessions]	http://de3.php.net/manual/de/ref.session.php
[sipmess]	http://www.iptel.org/ietf55/use_msn.html
[SRTP]	http://srtp.sourceforge.net/srtp.html

Danke

Viele Menschen begleiteten mich während der Diplomarbeit. Sie bauten mich auf, gaben mir Tipps oder waren einfach für mich da. Und dafür möchte ich mich bedanken.

Prof. Dr Grebe und Prof. Dr. Dettmar

Wie könnte eine Diplomarbeit ohne Professoren stattfinden, also fange ich hier mal an. Bei Prof. Dettmar hatte ich meine letzte Vorlesung. Diese hat mir so viel Spass gemacht, dass ich die Klausur gleich zweimal hintereinander schrieb. Asche auf mein Haupt, ich hatte einfach viel zu viele andere Dinge gemacht, als mich richtig auf die erste Klausur zu konzentrieren, dafür wurde die zweite eine der besten. Schade aber, dass es solche Professoren nicht in großer Anzahl vor 13 Jahren gegeben hat, dann hätte das Studium vielleicht sogar Spass machen können. Das meine ich wirklich so. Noch während der Vorbereitung auf die letzte Klausur (deren Vorlesung schon weit zurück liegt), hörte ich von einem Professor Grebe, der „da was mit IP-Telefonie macht“. Das wäre doch eine gute Wahl zur Diplomarbeit, passt es doch vom Thema sehr gut. Ein Glück, dass Prof. Grebe trotz hoher Auslastung ein Herz für VoIPer hatte und sich meinen Diplomvorschlag anhörte. Nach einigen interessanten Gesprächen stand das Gerüst und es konnte losgehen. Es wäre schön, wenn ich den Kontakt auch nach der Diplomarbeit nicht ganz verlieren würde.

Holger Sesterhenn

Holger lernte ich kennen, als das Studium schon fast beendet war. Er hatte noch eine Klausur zu schreiben, ich zwei. Er konnte pauken wie nichts und hat mich gut mitgezogen. Nun ist er fertig und mit diesen Zeilen ziehe ich ihm nach. Ich hoffe, dass wir uns jetzt, wo das Studium hinter uns liegt, nicht aus den Augen verlieren.

Thomas Latsch

Ich bin wohl berühmt für meine spontanen Hilferufe. Thomas ist immer an vorderster Front, immer zur Stelle, wenn er helfen kann. Trotz anstrengendem Tag hat er an einem Abend mit zufallenden Augen die Diplomarbeit komplett durchgelesen, damit sie am übernächsten Tag korrigiert in den Druck gehen konnte. Das nenne ich Freundschaft, Hut ab.

Wodatel Inc.

Bei mir ist das Gefühl, das Studium nun endlich zu beenden etwa so, als würden andere Menschen die Rente erreichen. Da ist ein Gefühl von Leere, es kommen Fragen auf wie „Was mache ich nun?“. Da stand ich, wenn gerade nicht beruflich, dann vom Studium her unter Strom. Und auf einmal sollten meine Großaufträge wie auch das Studium beendet sein. Wolfgang Uelpenich und Daniel Könnner konnten sich das Elend nicht ansehen und nahmen mich in ihre Mitte. Somit habe ich auch nach dem Studium immer genug Stress (natürlich nur von positiver Art), so wie ich es gewohnt bin. Thank you, boys!

Mark Spencer

Ohne diesen Namen wäre die Diplomarbeit nie entstanden, er ist der Ersteller der Asterisk-Software. Ohne ihn wären tausende von VoIP-Entwicklern noch deutlich hinter dem, wo sie jetzt sind. Ich gehe nicht davon aus, dass es ohne ihn heute schon eine gleichwertige Software geben würde. Hierdurch habe ich auch viele Aufträge für die Einrichtung von Asterisk bei verschiedenen VoIP-Anbietern erhalten, Mark hat mir sozusagen mein monatliches Einkommen gesichert.

Meine Frau

Fast zum Schluss nenne ich nun meine Frau, wo Sie mich aber doch am meisten unterstützt hat. Jeden Tag musste sie einen etwas grummeligen Ehemann ertragen und das fing schon um 6 Uhr an, liess aber erst nach 22 Uhr nach. Da sie in dieser Zeit nicht die Scheidung eingereicht hat, sollte unsere Ehe auf jeden Fall die nächsten 100 Jahre Bestand haben. Ein Glück habe ich mit ihr die richtige Frau gefunden...

Alle anderen...

Man sagt, Informationen im Internet altern. Andersherum ist es wohl richtig. Hätte ich die Diplomarbeit als HTML auf eine Webseite gelegt, könnte ich die, die ich jetzt vergessen habe, einfach nachpflegen. Doch durch den Transfer auf ein nicht flüchtiges Medium, nämlich Papier, muss ich mich hier schon festlegen. Vielen Dank auch an jene, die mir jetzt gerade nicht einfallen, was sicherlich kein Zeichen dafür ist, dass sie mir weniger wichtig sind!

Rolf Winterscheidt, rolf@rowi.net