

### 3.1.5 Ressourcenbedarf

In Abhängigkeit von der Art des ausgewählten Codecs und des eingesetzten Transportmechanismus (z.B. RTP-Transport im Falle der Nutzung von SIP vs. IAX2 zur Koppelung von Asterisk-Knoten) ergibt sich ein unterschiedlicher Anspruch an die CPU und ein deutlich variierender Bandbreitenbedarf, der bei der Planung und Auslegung einer Asterisk-Lösung beachtet werden sollte.

*Der Anspruch von Asterisk an die CPU hängt stark von den verwendeten Codecs ab.*

Zu beachten ist insbesondere der durch die unterschiedlichen Paket-Header bedingte Overhead, der gegenüber der Rohdatenrate der verwendeten Codecs nicht zu vernachlässigen ist. Dies gilt vor allem beim Vergleich des in der Regel als Referenz anzunehmenden G.711-Codecs mit 64 kbps vs. eines höher komprimierenden Codecs wie z.B. G.729. Hier ist das letztlich resultierende Verhältnis der entstehenden Datenrate deutlich verschieden vom zunächst sehr vorteilhaft erscheinenden Verhältnis der Rohdatenraten.

*Die benutzte Bandbreite hängt vom Codec und vom Paket-Overhead ab.*

Wir verweisen hier auf die Darstellung zu den einzelnen Codecs im Abschnitt 1.3.1 im Grundlagenkapitel.

## 3.2 Installation

Asterisk liegt heute neben der Quellversion auch als fertig vorbereitete CD oder als brennbares ISO-Image vor. Allerdings sind diese CDs aus unterschiedlichen Quellen immer nur an eine bestimmte Hardwareumgebung optimal angepasst.

Es ist daher für den etwas ambitionierten Anwender in jedem Fall von Vorteil, Asterisk angepasst an seine persönlichen Bedürfnisse neu aus Quellen zu installieren und genau mit den Modulen zu versehen, die in seiner Umgebung am sinnvollsten sind.

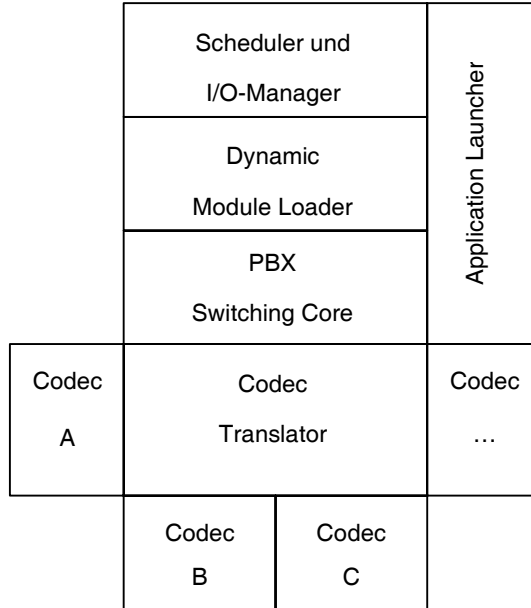
*Ein optimal an die eigenen Wünsche angepasstes Asterisk-System lässt sich einfach aus den Quellen selbst zusammenstellen.*

Allerdings sind hierzu einige Kenntnisse und Vorbereitungen notwendig, da Asterisk als Open-Source-Entwicklung derzeit immer noch stark im Fluss ist und noch sehr wenig Unterstützung für eine automatische oder selbst laufende Installation bietet.

### 3.2.1 Architektur

Die Architektur von Asterisk ist relativ einfach und überschaubar. Dennoch handelt es sich bei Asterisk um ein komplexes System, dessen interner Aufbau einigermaßen verstanden sein sollte, bevor man sich mit der Umsetzung beschäftigt.

**Abb. 3-1**  
Modulararchitektur  
von Asterisk



Kern des Systems und erstes Modul beim Start ist der »Dynamic Module Loader«, der die einzelnen für den Ablauf von Asterisk notwendigen Module Zug um Zug in den Speicher lädt und initialisiert.

Das Modul »Scheduler und I/O-Manager« ist für die Verwaltung der Applikationen, den Betrieb der Sprachkanäle und für zeitgesteuerte Abläufe verantwortlich.

*Asterisk ist modular  
aufgebaut.*

Die eigentlichen Aufgaben in der Vermittlung von Rufen übernimmt das Modul »PBX Switching Core«. Dieses Modul bedient eingehende Rufe und leitet sie nach den in der Konfiguration festgelegten Regeln weiter. Hier findet die Abbildung von Rufnummern auf Anschlüsse statt.

Sollen Telefone gerufen werden oder Anrufe an andere Softwaremodule wie Voicemail weitergeleitet werden, übernimmt das der »Application Launcher«.

Der »Codec Translator« ist für die Umrechnung von unterschiedlich kodierten Sprachpaketen zuständig.

Der Asterisk-Kern kommuniziert mit der Außenwelt über vier unterschiedliche Schnittstellen. Für die Kommunikation mit Codec-Modulen ist das »CODEC Translator API« zuständig. Die Kommunikation mit den Schnittstellenkarten und ihren Treibern findet mit den Regeln des »Asterisk Channel API« statt. Zusatzmodule, auch von dritter Seite oder selbst geschrieben, können über das »Asterisk Application API« mit den Modulen im Kern kommunizieren. Alle Dateizu-

griffe von Asterisk laufen über eine im »Asterisk File Format API« definierte Schnittstelle ab.

### 3.2.2 Basis

Um auf Linux ein Asterisk-System installieren zu können, benötigt man die üblichen Entwicklungswerkzeuge und Softwarepakete. Asterisk selbst ist in C geschrieben, während zum Beispiel die Implementierung von H.323 in C++ vorliegt. Man benötigt also beide Compiler und die zugehörigen Werkzeuge.

Will man direkt auf die aktuellsten Quellen von Asterisk zugreifen, so muss man dazu das Programmpaket *Subversion* installieren. *Subversion* ist ein Versionskontrollsystem, das von Digium für die Verwaltung der Asterisk-Quellen eingesetzt wird.

Bei Asterisk wird zur Erstellung der Programmdokumentation das Tool *doxygen* verwendet, man muss es installieren, um auf die vollständige Programmdokumentation der Quelldateien zugreifen zu können.

Ein weiteres von Asterisk vorausgesetztes Softwarepaket ist *OpenSSL*, hiervon werden verschiedene Bibliotheken benötigt. Am sinnvollsten ist die Installation des Paketes *openssl-devel*, das alle benötigten Teile enthält.

Weiterhin werden von der Asterisk-Dokumentation die folgenden Bibliotheken und Pakete genannt, wobei allerdings nicht alle für die beim Erscheinen des Buches vorliegenden Version 1.2.3 von Asterisk wirklich gebraucht werden:

- *libtermcap-devel* (ist in der aktuellen Version nicht notwendig),
- *ncurses-devel*,
- *readline-devel* (ist in der aktuellen Version nicht notwendig),
- *postgresql-devel* (für die Ablage von CDR-Datensätzen in einer Datenbank),
- *bison* oder *yacc* (ist in der aktuellen Version nicht notwendig),
- *newt* (ist in der aktuellen Version nicht notwendig),
- *zlib-devel*,
- *mpg123* (nur für das Abspielen von Musik in Wartepausen),
- *kernel source* (ist in der aktuellen Version nicht immer notwendig).

*Asterisk braucht nur einige Werkzeuge, die im Linux-Umfeld üblich sind.*

*Je nach verwendeter Asterisk-Version sind unterschiedliche Zusatzbibliotheken notwendig.*

### 3.2.3 Software

Die Software für Asterisk kann man sich auf zwei Wegen beschaffen, man kann entweder eine Reihe von Tarballs per FTP laden oder die Dateien direkt aus dem SVN-(Subversion)-Server von Digium laden.

*Der Asterisk-Quellcode wird von Digium bereitgestellt.*

Sinnvollerweise erzeugt man ein eigenes Quellverzeichnis für Asterisk und lädt die benötigten Dateien dorthin.

```
#cd /usr/src
#mkdir asterisk
#cd asterisk
```

Jetzt kann man die benötigten Pakete laden.

Zuvor sollte man sich mit einem Blick auf die zugehörige Website (<http://www.asterisk.org/download>) über die aktuellen Versionen informieren. Es lohnt sich, regelmäßig nach aktualisierten Versionen zu suchen, da Asterisk derzeit noch relativ schnell und stark weiter entwickelt wird. Die in den folgenden Beispielen vorkommenden Versionsnummern sollen nur als Beispiele dienen und sind sicher schon wenige Wochen nach Drucklegung des Buches überholt. Benötigt wird in jedem Fall das Paket *asterisk*.

```
#wget http://ftp.digium.com/pub/asterisk/asterisk-1.2.8.tar.gz
```

Je nach eingesetzter Hardware sind noch *libpri* für ISDN-Multiplex-Verbindungen (Primärmultiplexanschlüsse) und *zaptel* für die Unterstützung von Zaptel-Karten (analoge Telefone) notwendig.

```
#wget http://ftp.digium.com/pub/libpri/libpri-1.2.3.tar.gz
#wget http://ftp.digium.com/pub/zaptel/zaptel-1.2.6.tar.gz
```

Das Paket *libiax* für das IAX-Protokoll als Bibliothek benötigt nur der Entwickler von eigener Software, die über IAX mit einem Asterisk-Server kommunizieren soll.

```
#wget http://ftp.digium.com/pub/libiax/iax-0.2.2.tar.gz
```

Die Auswahl der Module erfolgt nach den persönlichen Bedürfnissen.

Im Paket *asterisk-addons* finden sich eine Reihe von Zusatzmodulen, die einen Basis-Server erweitern können, so zum Beispiel um eine Einbindung in eine MySQL-Datenbank zur Verwaltung der Gesprächsdaten.

Im Paket *asterisk-sounds* befindet sich eine Reihe von zusätzlichen Sprachaufnahmen, allerdings nur in Englisch, mit denen man seine Ansagen in Asterisk gestalten kann.

```
#wget http://ftp.digium.com/pub/asterisk/asterisk-addons-1.2.3.tar.gz
#wget http://ftp.digium.com/pub/asterisk/asterisk-sounds-1.2.1.tar.gz
```

Ansagen in Deutsch sind im Netz verfügbar.

Ein sehr praktisches Zusatzmodul kommt von der Stadt Pforzheim. Dort wurde im Rahmen eines internen Asterisk-Projekts eine umfangreiche Sammlung von Sprachaufzeichnungen erstellt und unter der GPL-Lizenz der Asterisk-Gemeinde zur Verfügung gestellt. Da im Originalpaket von Asterisk nur Aufzeichnungen auf Englisch mitkommen, ist dieses Paket eine große Hilfe.

```
#wget http://www.stadt-pforzheim.de/asterisk/dateien/
                                ast_prompts_de_v2_0.tar.gz
```

Die einzelnen Pakete kann man jetzt entpacken, auf Unterverzeichnisse verteilen, die nicht mehr benötigten Dateien entfernen und zur Vereinfachung einige Links setzen:

```
#for i in *.tar.gz ; do tar xzpf $i ; done
#rm *.gz
#ln -s asterisk-1.2.8 asterisk
#ln -s asterisk-addons-1.2.3 asterisk-addons
#ln -s asterisk-sounds-1.2.1 asterisk-sounds
#ln -s iax-0.2.2 libiax
#ln -s libpri-1.2.3 libpri
#ln -s zaptel-1.2.6 zaptel
#ln -s ast_prompts_de_v2_0 de-sounds
```

Alternativ kann man sich die Dateien auch direkt vom SVN-Server holen:

```
#svn checkout http://svn.digium.com/svn/asterisk/trunk asterisk
#svn checkout http://svn.digium.com/svn/zaptel/trunk zaptel
#svn checkout http://svn.digium.com/svn/libpri/trunk libpri
```

Allerdings ist bei dieser Vorgehensweise besondere Vorsicht und große Sorgfalt beim weiteren Verwenden der Software angesagt. Bei den direkt aus dem SVN-Server ausgelesenen aktuellsten Versionen handelt es sich nicht um ausreichend getestete und sicher lauffähige Module, man kann hier auch schon auf Fehler bei der Übersetzung oder Fehlfunktionen später beim Betrieb stoßen. Hier sollte sich nur der Anwender bewegen, der genau weiß, auf was er sich einlässt.

*Asterisk aus dem aktuellen Entwicklungsbaum ist nicht immer stabil.*

Für den nicht ganz so ambitionierten Anwender empfiehlt es sich, seine Quellen gezielt aus einer aktuellen, aber schon etwas weiter fortgeschrittenen Version des Entwicklungsbaumes zu holen (hier als Beispiel die Release-Version 1.2):

```
#svn checkout http://svn.digium.com/svn/asterisk/branches/
                                1.2 asterisk
#svn checkout http://svn.digium.com/svn/zaptel/branches/1.2 zaptel
#svn checkout http://svn.digium.com/svn/libpri/branches/1.2 libpri
```

Und schon kann es mit dem Übersetzen der Einzelteile losgehen, zuerst werden die Interfacebibliotheken vorbereitet, wobei die *zaptel*-Treiber-Software nur dann benötigt wird, wenn man auch Geräte aus dieser Klasse einsetzen will. Die *libpri*-Bibliotheken werden in jedem Falle gebraucht.

Hat man keine Zaptel-Karte, so sollte man dennoch den Treiber *ztdummy* übersetzen und laden. Er wird von Asterisk als interne Zeit-

referenz für unterschiedliche Funktionen, wie zum Beispiel die Konferenzschaltung, benötigt.

Dazu muss man bei einem Linux-Kernel vor Version 2.6 in der Datei Makefile die Zeile mit »ztdummy« suchen und das Zeichen »#« davor entfernen. Bei Kernel ab der Version 2.6 wird dies automatisch vom Makefile erkannt und der passende *ztdummy* wird generiert.

```
#cd zaptel
#vi Makefile
#make
#make install
```

Damit beim Systemstart die richtigen Module geladen werden, muss jetzt in der Datei */etc/sysconfig/zaptel* die Zeile *ZAPTEL\_MODULES=""* um »ztdummy« ergänzt werden. Dann kann man den *zaptel*-Treiber mit

```
#/etc/rc.d/zaptel start
```

in den Speicher laden.

Jetzt kommt noch die Bibliothek für ISDN-PRI-Anschlüsse:

*Je nach eigener  
Konfiguration müssen  
nicht alle Bibliotheken  
übersetzt werden.*

```
#cd ../libpri
#make
#make install
```

Dann kann man sich an das Hauptprogramm wagen:

```
#cd asterisk
#make
```

Hier werden in vielen Fällen Fehler auftreten. Meist fehlen für das erfolgreiche Übersetzen und Binden eine oder mehrere Bibliotheken und Funktionen, die von Asterisk in der aktuellen Version gerade benötigt werden.

Eine Fehlermeldung kann zum Beispiel so aussehen:

```
checking for tgetent in -lncurses... no
figure: error: termcap support not found
make: *** [editline/libedit.a] Error 1
```

*Fehler bei der Übersetzung  
sind kein großes Problem.*

Zur Lösung muss man das jeweilige Paket (der oben gezeigte Fehler wurde durch das Nachladen der SuSE-Pakete *termcap*, *ncurses* und *ncurses-devel* behoben) installieren und kann dann den make-Lauf wiederholen. Falls man nicht beim ersten Mal alle fehlenden Pakete nachladen konnte, muss man diesen Vorgang beliebig oft wiederholen.

Ist man bei der Übersetzung erfolgreich, so erscheint am Ende des Protokolls ein Hinweis auf den nächsten Schritt:

```
+----- Asterisk Build Complete -----+
+ Asterisk has successfully been built, but +
+ cannot be run before being installed by +
+ running:                                +
+                                          +
+      make install                        +
+-----+

```

Das »make install« platziert die Hauptkomponenten von Asterisk in die dafür vorgesehenen Verzeichnisse. Mit »make samples« kann man die mitgelieferten Beispielkonfigurationen auspacken und in die passenden Verzeichnisse kopieren. Allerdings ist hierbei Vorsicht geboten, da eventuell bereits existierende Konfigurationsdaten überschrieben werden, vorher existierende Daten werden in Dateien mit der Erweiterung `.old` verschoben.

Taucht bei der Installation von Asterisk eine Fehlermeldung bezüglich fehlendem oder falschem Modul `mpg123` auf, so kann man diese am schnellsten durch ein Laden des entsprechenden und wirklich für Asterisk passenden Moduls aus den Asterisk-Quellen beheben:

```
#make mpg123
```

Beschwert sich Asterisk beim Installieren über falsche oder unpassende Module in seinem Verzeichnis, so kann dies zwei Ursachen haben: Entweder hat man bereits eine ältere Version von Asterisk selbst installiert oder sie wurde mit der Distribution aufgespielt. Dann sollte man am einfachsten das gesamte Verzeichnis `/usr/lib/asterisk/modules` löschen und die Installation mit »make install« erneut aufrufen. Beschwert sich Asterisk nur über einzelne Module (z.B. `chan_capi.so`), so hat man eventuell ein nicht zur Grundausstattung von Asterisk gehörendes Modul bereits übersetzt und geladen. Diese Meldungen kann man dann getrost ignorieren.

Will man auf die vollständige Programmdokumentation zugreifen, so kann man diese mit dem Kommando »make progdocs« erzeugen. Voraussetzung für ein Gelingen ist die Installation des dazu notwendigen Paketes `doxygen`, mit dessen Hilfe aus den Quelldateien die Dokumentationszeilen extrahiert werden. Das Ergebnis findet man unterhalb der aktuellen Arbeitsdatei im Verzeichnis `doc/apilhtml`.

Um die zusätzlichen Sprachdateien zu installieren, wechselt man zuerst in das Verzeichnis `asterisk-sounds` und kann dort mit »make install« eine große Auswahl an englischen Sprachansagen für Asterisk bereitstellen.

```
#cd asterisk-sounds
#make install
```

*Die Übersetzung und Installation von Asterisk folgt den für Linux üblichen Pfaden.*

*Alte Module können stören und müssen entfernt werden.*

Anschließend kann man die Auswahl englischer Texte um deutsche Ansagen erweitern. Für dieses Paket gibt es noch kein automatisches `make`, hier ist Handarbeit angesagt.

```
#cd de-sounds/var/lib/asterisk/sounds
#cp -R * /var/lib/asterisk/sounds
```

*Deutsche Ansagen erleichtern den Umgang mit Asterisk für die Benutzer.*

Insgesamt hat man jetzt knapp über 2.000 Tonschnipsel installiert, die man an passender Stelle als Ansage oder Steuertext verwenden kann.

Die anderen zusätzlichen Pakete sollten nur bei Bedarf gebaut und installiert werden.

Ist man mit dem Übersetzen und Installieren fertig, so zeigt ein erster Aufruf von *asterisk*, ob die wesentlichen Teile vorhanden sind und funktionieren.

```
#/usr/sbin/asterisk -c
```

Asterisk sollte sich jetzt auf dem Bildschirm melden:

```
Asterisk 1.2.7.1, Copyright (C) 1999 - 2006 Digium, Inc. and others.
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY;
type 'show warranty' for details.
This is free software, with components licensed under
the GNU General Public License version 2 and other
licenses; you are welcome to redistribute it under
certain conditions. Type 'show license' for details.
=====
[ Booting...Sep  4 15:37:06
.....
..... ]
Asterisk Ready.
*CLI>
```

Mit diesem Aufruf startet Asterisk im Dialogmodus. Asterisk meldet sich mit dem internen Kommando-Prompt und erwartet Anweisungen.

```
*CLI>help
```

*Asterisk kann von der Kommandozeile aus bedient werden.*

Das Kommando *help* zeigt die verfügbaren Befehle und mit »*help* Kommandoname« erhält man einen Hilfstext zum jeweiligen Befehl.

```
*CLI>show applications
```

Hiermit kann man die zur Verfügung stehenden Erweiterungsmodule auflisten lassen.

```
*CLI>stop now
```

Mit dem Kommando *stop* wird Asterisk wieder beendet.

Für den Kommandomodus von Asterisk gelten die üblichen Abkürzungstasten. Die TAB-Taste vervollständigt ein teilweise eingegebenes Kommando. Die Pfeil-oben-Taste blättert zu den vorher eingegebenen Kommandos zurück.

### 3.2.4 Start und Konfiguration

Asterisk wird wie fast alle Unix-Services über Dateien im Unterverzeichnis */etc* konfiguriert. Asterisk bietet allerdings zusätzlich vielfältige Steuerungsmöglichkeiten über das Kommandozeileninterface (siehe ausführliche Beschreibung in [28] und [12]).

Asterisk fasst seine Konfigurationsdateien in einem Unterverzeichnis zusammen, üblicherweise */etc/asterisk*. Man darf sich von der großen Zahl an Dateien in diesem Unterverzeichnis nicht erschrecken lassen, für den Anfang reichen einige zentrale Einstellungen.

Asterisk sucht Konfigurationsdateien üblicherweise in */etc/asterisk*.

Das Asterisk-Hauptprogramm liegt üblicherweise unter */usr/sbin/asterisk* und kann von dort aus als Dämon im Hintergrund (mit */usr/sbin/asterisk*) oder als Dialoginterface (*/usr/sbin/asterisk -r* falls schon ein Asterisk-Prozess im Hintergrund läuft oder mit */usr/sbin/asterisk -c* ohne Dämon) im Vordergrund gestartet werden.

Im Verzeichnis */usr/lib/asterisk/modules* werden bei der Installation alle nachladbaren Module von Asterisk abgelegt. Sieht man in dieses Verzeichnis hinein, so erkennt man an der großen Zahl von Modulen schon die Vielfältigkeit der Anwendungen von Asterisk.

Im Verzeichnis */var/lib/asterisk* werden weiter für den Betrieb notwendige Daten abgelegt. So findet man im Unterverzeichnis *sounds* die für verschiedene Ansagen benötigten Textschnipsel. Die Dateinamen und Unterverzeichnisse wie *sounds/digits* sind größtenteils fest in Asterisk einkompiliert und lassen sich nicht von außen ändern. Will man hier eigene Texte verwenden, so kann man die entsprechenden Dateien einfach ersetzen. Weitere Informationen und Hinweise für passende deutsche Sprachschnipsel finden sich in Abschnitt 3.2.3.

Asterisk nutzt zur Laufzeit Dateien im Baum unterhalb von */var/lib/asterisk* für viele Zwecke.

In weiteren Unterverzeichnissen von */var/lib/asterisk* finden sich Dateien mit Firmware zum Laden intelligenter Adapterkarten (Unterverzeichnis *firmware*), digitale Zertifikate für die Authentisierung von gesicherten Verbindungen (unter *keys*) oder Musik im Format mp3 zur Einspielung in Warteschleifen (Verzeichnis *mohmp3*).

Protokollierungsdaten werden von Asterisk im Verzeichnis */var/log/asterisk* angelegt. Neben allgemeinen Log-Daten (in */var/log/asterisk/messages*) finden sich dort im Unterverzeichnis *cdr-csv* alle Dateien für die Zeit- oder Kostenabrechnungen.

Asterisk legt Log-Daten und Abrechnungsdaten in */var/log/asterisk* ab.