Konzepte zur Vernetzung analoger Sprach-Relais unter Linux

NILS PRAUSE, DO6NP
Menkestr. 6
49076 Osnabrück
E-Mail: do6np@amateurfunk-osnabrueck.de
Web: www.nils-prause.de

JANN TRASCHEWSKI, DG8NGN
Drosselstr. 1
90513 Zirndorf
E-Mail: jann@gmx.de
Web: www.qsl.net/dg8ngn

23. März 2010

Inhaltsverzeichnis

1 Einführung

Wer sich einmal mit der Frage beschäftigt hat, wie Relais-Steuerungen zu Beginn des "Relais-Zeitalters" Anfang der 1970er-Jahre realisiert wurden, wird unwillkührlich Respekt vor den damaligen Erbauern dieser Steuerungen bekommen. Die Zeit von Schlitzpappscheiben und Fototransistoren liegt jedoch bereits ein Weilchen zurück. Bei den meisten Umsetzern sind heute Steuerungen auf Basis verschiedenster Controller-Typen im Einsatz, deren Funktionalität sich jedoch häufig sehr ähnelt.

Im Punkto Vernetzung von Relais wurden ebenfalls früh erste Erfahrungen gesammelt. Hierbei wurden verschiedene Standorte zumeist mittels Linkstrecken im 70- und später auch 3-cm-Band miteinander verbunden. Große Reichweiten waren auf diesem Weg jedoch nicht zu erzielen. Die Vernetzung von Relais mit Hilfe des so genannten Monitor-Verfahrens bot hier ein neues Konzept, das auch in einigen Projekten, etwa im Ruhrgebiet, realisiert wurde [Monitorverfahren]. Dieses Konzept stellt jedoch besondere Anforderungen an Hardware und Topographie und ist überdies nur dort realisierbar, wo eine direkte (Sicht)-Verbindung zwischen den Umsetzern möglich ist.

Die Vernetzung mittels Internet über EchoLink [EchoLink] und vereinzelt auch IRLP [IRLP] ist bekannt, birgt aber eine große Zahl Unzulänglichkeiten. So ist etwa der bei EchoLink verwendete GSM 6.10 Codec äußerst schmalbandig, was dem Einsatz in Ländern mit weniger entwickelter IT-Infrastruktur geschuldet ist, und durch die Tatsache, dass sich hier verschiedenste PC-Nutzer, Simplex-Links und Relais miteinander vernetzen können, kann eine hohe Linkqualität nicht gewährleistet werden [ELWiki]. IRLP wiederum stellt ganz bestimmte Anforderungen sowohl an Hard- und Software als auch an Operationsmodi und hat bislang in Deutschland keine hohe Durchdringung erfahren. Auch "private Netze", bei denen sich nur ganz bestimmte Umsetzer durch Tastendruck miteinander verbinden lassen, können über diese Systeme nicht realisiert werden.

Eine Übersicht herkömmlicher Vernetzungsmöglichkeiten bietet unter anderem ein entsprechender Artikel von Jann Traschewski, DG8NGN, aus dem Jahr 2004 [Relaiskopplung].

Neue Möglichkeiten bieten hier moderne Software-Konzepte auf Linux-Basis. Dieser Artikel möchte die beiden Systeme SvxLink und Asterisk/app_rpt vorstellen und Möglichkeiten aufzeigen, wie die Attraktivität "herkömmlicher" Analog-Relais durch Softwarelösungen

erhöht werden kann und wie Relais mittels TCP/IP-Protokoll leicht miteinander vernetzt werden können.

Besondere Attraktivität könnten diese Systeme nicht zuletzt auch im Rahmen des so genannten HAMNET entfalten (vgl. hierzu den Beitrag von Jann Traschewski in diesem Band), da hier erstmals flächendeckend eine ausreichend schnelle, "amateurfunkinterne" und internetunabhängige Vernetzung gewährleistet werden könnte. Aufgrund des Umfangs wird dieser Artikel jedoch Embedded-Technologien nur am Rande behandeln.

Nicht detailliert eingegangen werden kann aufgrund des Umfangs auf das ebenfalls zum SvxLink-Programmpaket gehörende QTel, welches einen EchoLink-Client für die QT-Oberfläche darstellt. Ebenfalls nur am Rande dokumentiert werden können allgemeine Hinweise zu Asterisk sowie weitere Channel-Treiber für Amateurfunkanwendungen wie etwa chan_dstar [chandstar].

2 SvxLink

2.1 Einführung

Bei SvxLink (gesprochen: SwixLink) handelt es sich um eine Open-Source-Entwicklung auf Linux-Basis von Tobias Blomberg, SM0SVX, welche die Verbindung von PC und Transceiver via Soundkarte und serieller Schnittstelle nutzt, um darüber verschiedenste Audio-Anwendungen zu realisieren.

Mit SvxLink können sowohl einfache Simplex-Lösungen (EchoLink-Einstieg, Papagei...) als auch komplexere Szenarien (etwa Relais mit mehreren Sendern und Empfängern) realisiert werden. Im Folgenden wird dieser Artikel ausschließlich die für Repeater relevanten Funktionen beleuchten, wobei viele dieser Funktionen so oder ähnlich auch im Simplex-Betrieb eingesetzt werden könnten.

SvxLink wird bereits seit dem Jahr 2003 programmiert und hat eine kontinuierliche Entwicklungsgeschichte hinter sich, wobei häufig Entwickler- und ein bis zweimal jährlich auch stabile Versionen herausgegeben werden. Lag die Entwicklung bis etwa 2008 beinahe zu 100% in den Händen von SM0SVX, so haben sich mittlerweile einige weitere, vor allem deutsche, Entwickler gefunden, die das Projekt mit einer größeren Zahl von Änderungen und Erweiterungen vorangebracht haben.

2.2 Funktionsbeschreibung

Um die Funktionalität des Systems besser verstehen zu können, soll zunächst dessen Aufbau dokumentiert werden.

Das Programmpaket SvxLink gliedert sich in vier Ebenen:

- Core
- Modul(e)
- Logic
- Konfiguration

2.2.1 Der Core

Der Core ist das Hauptprogramm. Es handelt sich dabei um eine native Konsolenanwendung, die vereinfacht gesagt die Kommunikation zwischen PC und Transceiver steuert. Darüber hinaus stellt sie Basisfunktionen wie etwa das Aussenden einer Kennung oder die Rauschsperren-Erkennung (COS) zur Verfügung. Der Core wird also immer benötigt und stellt das Zentrum des Systems dar. Gleichzeitig ist er auch das Bindeglied zu den Modulen, die die eigentlichen Programmfunktionen übernehmen.

2.2.2 Die Module

Rund um den Core kann SvxLink mit Modulen um nahezu beliebige Funktionen erweitert werden. Die Module sind dabei eigenständige Teilprogramme, welche auf die Basisfunktionen des Core wie beispielsweise das Abspielen einer Audio-Datei oder das Warten auf ein DTMF-Kommando zurückgreifen. Umgekehrt werden die Module bei Programmstart initialisiert und in den Speicher geladen. Nach einer Aktivierung durch den Benutzer springt SvxLink dann in die entsprechenden Funktionen des Moduls.

Ein solches Modul kann dabei entweder als reines Skript in der Sprache Tcl [Tcl] oder als dynamische C++-Bibliothek konzipiert werden. Doch auch wer nicht selbst programmieren kann, erhält mit den SvxLink beiliegenden Modulen bereits einen überaus interessanten "Werkzeugkasten" zur Realisierung verschiedenster Anwendungen. Darüber hinaus lassen sich im Internet diverse Module von Drittentwicklern herunterladen, mit denen das eigene System erweitert werden kann.

- `EchoLink` - Das Modul stellt eine nahezu vollwertige Alternative zum EchoLink-Programm von K1RFD dar und bietet darüber hinaus einige Optionen, die unter Windows nur mit dem Zusatz-Skript EchoLinkPlus von DK6XH und DC4FS [ELPlus] realisiert werden können. So ist es beispielsweise problemlos möglich, die Sprachausgaben selbst aufzuzeichnen und auch Funktionen wie das automatische Auftasten eines entfernten Repeaters können leicht implementiert werden. Das Modul kann sich darüber hinaus mit einem APRS-Server verbinden und eine Positionsbake ausgeben, die auch den aktuellen Verbindungsstatus beinhaltet.

- `TclVoiceMail` - ... stellt eine kleine Sprach-Mailbox zur Verfügung. Diese kann Nachrichten für zuvor registrierte Nutzer aufzeichnen und diese nach einem Login des Empfängers mit Benutzernummer und Passwort wiedergeben. Je nach Wunsch kann die Nachricht anschließend gelöscht oder wiederum beantwortet werden. Eine Information, dass Nachrichten vorliegen, kann bei vorhandenem E-Mail-System (Sendmail [Sendmail] oder ein entsprechender Wrapper) direkt per E-Mail an den Empfänger gesendet werden. Außerdem kann SvxLink auch zur vollen Stunde eine Information hierüber ausgeben, etwa: "Es ist jetzt 15.00 Uhr. Nachrichten für DL1ABC." Die Nutzer müssen in einer kleinen Datenbank im Textformat angelegt und ihnen muss eine Benutzernummer zugewiesen werden. Eine Registrierung per Sprache oder DTMF ist bislang nicht möglich.

- `Help` - Das Modul nimmt in der Philosophie von SM0SVX eine zentrale Rolle ein. Nach seiner Überzeugung werden Benutzer, die sich mit der Bedienung von SvxLink nicht auskennen, dieses verwenden, um generelle bzw. modulbezogene Hilfe abzurufen. Wir können nach den auf dem eigenen Umsetzer gemachten Erfahrungen diese Einschätzung nicht teilen. Dennoch kann dieses Modul bei richtiger Anwendung durchaus eine Hilfe sein, um die Bedienung des Relais zu erlernen.

- `Parrot` - Das Modul erlaubt es dem User, sich seine eigene Aussendung auf Knopfdruck wiederholen zu lassen.

- `DTMFRepeater` - ... erlaubt die Prüfung von DTMF-Signalen, in dem jeweils der zuletzt gesendete DTMF-Ton als Zahl angesagt wird. So kann leicht überprüft werden, warum z.B. der Verbindungsaufbau mit einer EchoLink-Gegenstation regelmäßig misslingt.

- `MetarInfo` - Dieses noch relativ junge Modul ermöglicht es Wetterbeobachtungen und Prognosen von mehr als 4.000 Flughäfen (METAR-Daten [METAR]) auszugeben. Hierzu muss lediglich der ICAO-Code [ICAO] des nächstgelegenen Flughafens bekannt sein.

- `PropagationMonitor` -todo-

Wie zuvor bereits angesprochen, ist es für Entwickler mit Kenntnissen in den Sprachen C++ und / oder Tcl möglich, relativ leicht eigene Module zu entwickeln. Genau wie die bereits vorgestellten können diese Module durch einfache Funktionsaufrufe die Basisoperationen des Core nutzen und so leicht in Interaktion mit dem Benutzer treten. Denkbar und bereits realisiert sind beispielsweise Ansagen von Wetterstationsdaten, Sturmvorhersagen oder Staumeldungen. Denkbar wären aber auch komplexere Funktionen wie ein Service-Modul zur Fernwartung oder Verbindungen zu anderen Netzwerken wie etwa IRLP. Durch die freie Verfügbarkeit der Quelltexte steht darüber hinaus auch der Anpassung vorhandener Module an die eigenen Wünsche und Bedürfnisse nichts im Wege. Autoren eigener oder modifizierter Versionen vorhandener Module sind ausdrücklich aufgerufen, diese der Nutzergemeinschaft zur Verfügung zu stellen.

2.2.3 Logic

Ein dritter wichtiger Aspekt, der bislang nur am Rande angesprochen wurde, ist die Logik oder - genauer gesagt - sind die Logiken.

Mit Logic ist bei SvxLink das gemeint, was man bei herkömmlichen Umsetzern als Ablaufsteuerung bezeichnen würde. In ihr wird anhand von so genannten Events festgelegt, was passieren soll, wenn ein bestimmter Zustand oder ein bestimmtes Ereignis eintritt. Durch diese Struktur wird dem Stationsbetreiber ein überaus mächtiges Werkzeug in die Hand gegeben, mit dessen Hilfe er das Verhalten des Systems nahezu beliebig steuern kann. Zusammen mit der Konfiguration kann so eine äußerst flexible Ablaufsteuerung entwickelt werden. Im Allgemeinen müssen die Logiken für einen reibungslosen Betrieb zunächst nicht verändert werden; für eine Individualisierung der eigenen Station ist das Verständnis der Funktionsweise dieser Dateien jedoch essenziell.

Logiken sind grundsätzlich in Tcl abgefasst, so dass hier alle Funktionen der Skriptsprache inklusiv externer Aufrufe eingesetzt werden können.

Jede Logic besteht aus der Haupt-Datei logic.tcl, die alle zentralen Behandlungsroutinen enthält. Darüber hinaus existieren die beiden Dateien SimplexLogic.tcl und RepeaterLogic.tcl, die je nach konfiguriertem Betriebsmodus (s. Kapitel 2.2.4) vom Programm aufgerufen werden. In ihnen bestehen nochmals alle Funktionen, die bereits in der logic.tcl vorhanden sind. Im Standardfall wird hier einfach auf die entsprechende Prozedur aus der logic.tcl referenziert; es ist jedoch auch möglich, an dieser Stelle eine eigene einzufügen.

Die Datei events.tcl schließlich stellt Prozeduren bereit, die die Funktionsaufrufe in den zuvor genannten Logic-Dateien vereinfachen sollen ("Hilfsfunktionen").[1]

[1] Da Tcl streng genommen nur Prozeduren ("proc") kennt, es sich hierbei nach heutigem Sprachgebrauch aber um Funktionen handelt, wird der Begriff in diesem Kontext synonym verwendet.

Erklärt sei dies an einem Beispiel:

Nehmen wir an, der Benutzer auf der Eingabe hat ein falsches DTMF-Kommando gesendet. Daraufhin ruft der Core die Funktion "unknown_command" der Logic auf, um dort den Fehler weiter zu behandeln. Da unser System als Repeater konfiguriert ist, findet hier der Namensraum [Namespace] "RepeaterLogic" Anwendung, hier ist also die Datei RepeaterLogic.tcl angesprochen.

Listing 1: Ausschnitt aus RepeaterLogic.tcl

```
proc unknown_command {cmd} {
        Logic::unknown_command $cmd;
}
```

Das "Logic::" zeigt an, dass hier auf den Namensraum der logic.tcl verwiesen wird. In dieser findet sich ebenfalls eine Prozedur des Namens "unknown_command", an die der Parameter "$cmd" übergeben wird. Diese sieht dann wie folgt aus:

Listing 2: Ausschnitt aus logic.tcl

```
proc unknown_command {cmd} {
        spellWord $cmd;
        playMsg "Core" "unknown_command";
}
```

Hier wird also zunächst über "SpellWord" die fehlerhafte Eingabe wiederholt und dann mittels "playMsg" eine Audio-Datei mit der Fehlermeldung ausgegeben. Der erste Parameter ist dabei das Unterverzeichnis der Audio-Dateien, in dem sich die abzuspielende Datei befindet, der zweite bestimmt den Dateinamen ohne Dateiendung.

Da es eigentlich egal ist, in welchem Modus das System läuft, falsche Eingabe bleibt schließlich falsche Eingabe, wurde in diesem Beispiel von der RepeaterLogic.tcl aus lediglich die gleichnamige Funktion aus der logic.tcl referenziert. Dass man in der RepeaterLogic.tcl anstelle des bloßen Aufrufs der Funktion aus der logic.tcl auch eine abweichende Behandlungsroutine für die Fehleingabe hätte definieren können, sollte inzwischen klar geworden sein.

Kommen wir noch einmal zurück zum Befehl "playMsg" aus der logic.tcl. Dieser ist wiederum eine Funktion der events.tcl, die wir bislang noch nicht behandelt haben.

Listing 3: Ausschnitt aus events.tcl

```
proc playMsg {context msg} {
        global basedir;
        set candidates [glob -nocomplain
                "$basedir/$context/$msg.{wav,raw,gsm}"
                "$basedir/Default/$msg.{wav,raw,gsm}"];
        if { [llength $candidates] > 0 } {
                playFile [lindex $candidates 0];
        } else {
                puts "*** WARNING: Could not find audio
                clip \"$msg\" in context \"$context\"";
        }
}
```

Im Prinzip passiert hier nichts anderes, als dass der Funktionsaufruf für die logic.tcl vereinfacht und darüber hinaus geprüft wird, ob die angegebene Datei auch tatsächlich existiert. Am Ende wird die Funktion `PlayFile` aufgerufen, die direkt vom Core zur Verfügung gestellt wird und die Ausgabe einer Audio-Datei erlaubt. Diese Funktion hätte natürlich auch direkt in der logic.tcl oder gar RepeaterLogic.tcl aufgerufen werden können; da das Abspielen von Audio-Dateien jedoch häufig benötigt wird, macht es Sinn, sich hier zur Vereinfachung einer Hilfsfunktion zu bedienen.

Auch wenn dieses Schema zunächst ein wenig unübersichtlich wirken mag, weil Funktionen an verschiedenen Stellen definiert werden können, hat das vorangegangene Beispiel hoffentlich dazu beigetragen, die Mächtigkeit des Logic-Systems zu verdeutlichen.

Im Allgemeinen sollte gelten, dass solche Funktionen, die unabhängig vom Betriebsmodus sind, in der logic.tcl untergebracht werden, während modusspezifische Funktionen in SimplexLogic.tcl bzw. RepeaterLogic.tcl zu notieren sind.

2.2.4 Konfiguration

SvxLink wird mittels einer Hauptkonfigurationsdatei gesteuert, die standardmäßig als /etc/svxlink.conf abgespeichert wird. Diese ist in einem üblichen Textformat aufgebaut und in mehrere Sektionen unterteilt.

In der "GLOBAL"-Sektion wird zunächst der Betriebsmodus - Simplex oder Repeater - ausgewählt, welcher zuvor ja bereits eine Rolle spielte und über die zu nutzende Logic entscheidet.

In der "RepeaterLogic" werden dann unter anderem RX und TX definiert. Für jeden Sender und Empfänger werden dabei wiederum eigene Sektionen festgelegt. Jede Sektion kann dabei einen der folgenden Typen enthalten:

- TYPE=Local - lokal angeschlossenes Gerät (RX/TX)
- TYPE=Net - per TCP/IP angebundener Rechner mit installiertem remotetrx, an dem wiederum ein lokales Gerät angeschlossen ist (RX/TX)
- TYPE=Voter - System zum Betrieb mehrerer Empfänger (nur RX)

Pro SvxLink können theoretisch beliebig viele Sender und Empfänger definiert werden.

Der Voter bietet eine Art Diversity-Funktion, bei der durch eine Analyse aller eingehenden Empfängersignale versucht wird, das jeweils beste Empfangsergebnis herauszufinden und dieses dann an den / die angeschlossenen Sender weiterzugeben. Es muss jedoch gesagt werden, dass diese Funktion derzeit nur rudimentäre Prüfungen durchführt und hier sicher noch Verbesserungspotenzial besteht.

Soll der Voter eingesetzt werden, muss der RX ein Signal vom Diskriminatorausgang (ohne Squelch oder andere Veränderungen) bereitstellen und es müssen die Variablen SIGLEV_SLOPE und SIGLEV_OFFSET in der RX-Sektion richtig gesetzt werden. Um die benötigten Werte zu ermitteln, existiert das Zusatzprogramm siglevdetcal, für das eine eigene Beschreibung existiert [siglevdetcal].

Über so genannte "Link'"-Sektionen können außerdem mehrere Logiken verbunden werden. So ist es beispielsweise möglich, mit Hilfe einer Konfiguration gleichzeitig einen Simplex-Link und einen Repeater zu definieren - entsprechende Hardware vorausgesetzt. Über diese Sektion könnten beide dann mittels eines Kommandos verbunden werden. Hierüber kann auch eine Verlinkung mehrerer Repeater realisiert werden (vgl. Kapitel 2.6).

Die "Macros"-Sektion bietet eine Möglichkeit, Befehle zu vereinfachen. Makros können aufgerufen werden, in dem auf der Eingabe der Buchstabe "D" gefolgt von einem mehrstelligen Kommando eingegeben wird. So kann durch das Makro "49=EchoLink:49069#" beispielsweise erreicht werden, dass das DTMF-Kommando "D49" eine EchoLink-Verbindung zu DB0OSN-R mit der Nodenummer 49069 hergestellt wird (das Nummernzeichen dient hier als Anzeige des Kommandoendes, s. auch Kapitel 2.3).

2.3 Grundlagen der Bedienung

Wie bereits mehrfach angesprochen, können sowohl die Module als auch die Grundfunktionen von SvxLink in großem Maße konfiguriert und angepasst werden. Es ist daher nicht möglich, die Bedienung für jede Situation zu beschreiben. Ausgehend von einer Standardkonfiguration können jedoch folgende allgemeine Hinweise gegeben werden:

Im Grundzustand ist der Umsetzer geschlossen. Dieser kann je nach Konfiguration mittels PTT, 1750-Hz-Tonruf, CTCSS oder (zusätzlich) mittels des DTMF-Kommandos "*" geöffnet werden. Danach kann der Umsetzer für eine zu definierende Zeitspanne nachlaufen, anschließend kann er für eine weitere zu konfigurierende Periode durch Drücken der PTT erneut aufgetastet werden.

Im normalen Sprechbetrieb läuft SvxLink ohne ein aktiviertes Modul. Das Aktivieren eines Moduls geschieht durch die Eingabe der Ziffern 0 bis 9 auf der DTMF-Tastatur, wobei die Nummer eines Moduls in der zugehörigen Konfigurationsdatei festgelegt wird und jede Ziffer nur einmal vorkommen darf. Im Standardfall muss dieses und jedes weitere Kommando mit einer Raute abgeschlossen werden. Der SysOp hat jedoch die Möglichkeit, anstelle des Abschlusses mit der Raute einen Timeout zu setzen, nach dessen Ablauf das Kommando als beendet gilt oder das Loslassen der PTT als Kommandoende zu definieren. Die Raute sollte jedoch auch funktionieren wenn sie nicht benötigt wird und kann vor allem bei verrauschten Signalen eine Hilfe sein. Nach Aufruf eines Moduls hängt die weitere Bedienung von dessen Kommandosatz ab.

Das Help-Modul nimmt, wie bereits dargestellt, eine zentrale Rolle ein. So steht in jedem Modul Hilfe zur Verfügung, die Auskunft über die wichtigsten Kommandos gibt. Hilfe kann in der Regel mit dem Kommando "0" abgerufen werden.

2.4 Inbetriebnahme

2.4.1 Vorüberlegungen

Zunächst sind natürlich die üblichen Vorüberlegungen zum Aufbau einer Relaisstelle zu tätigen, auf die an dieser Stelle jedoch nicht eingegangen werden soll.

Transceiver, die als geeignet für ein Relais eingestuft werden, sollten auch alle für SvxLink benötigten Anschlüsse zur Verfügung stellen. Die Autoren haben die besten Erfahrungen mit Betriebs- bzw. Bündelfunkgeräten der Firma Motorola gemacht, was jedoch andere Transceiver natürlich nicht ausschließt.

Zur Verbindung mit dem PC wird ein Interface benötigt. Gemäß der Philosophie von SM0SVX soll SvxLink mit einem minimalen Aufwand an Hardware auskommen. Daher kann hier ein einfaches Interface mit galvanischer Trennung und seriellem Anschluss aufgebaut werden, wie es etwa Tim Miller, WB0RXX, auf seiner Website beschreibt [WB0RXX]. SvxLink verfügt zwar auch über die Möglichkeit, das Audiosignal mittels VOX zu ermitteln, dies scheint jedoch maximal für einen ersten Test oder einen Simplex-Link, nicht jedoch für ein Relais eine adäquate Methode zu sein. Stattdessen sollte auf jeden Fall entweder das Audio- und das Squelch-Signal zur Verfügung stehen oder man überlässt SvxLink das Diskreminatorsignal und somit auch die Signalerkennung. In letzterem Fall sollten die Hinweise zum Voter in Kapitel 2.2.4 unbedingt beachtet werden. Sollen mehrere Empfänger eingesetzt werden, ist immer ein Diskriminatorausgang notwendig.

Legt man Wert auf einen externen DTMF-Dekoder, wobei die internen Dekoder-Routinen von SvxLink relativ zuverlässig arbeiten, so existiert hierfür ein Interface von S54S, dessen Dekoder von SvxLink angesprochen werden kann [S54S]. Mit den externen Dekodern anderer

bekannter Interfaces wie etwa Digi-1 kann SvxLink hingegen nicht zusammenarbeiten.

Für EchoLink sowie ggf. die Anbindung externer Sender und Empfänger ist natürlich weiterhin eine ausreichend schnelle Internetverbindung notwendig. Für EchoLink muss dabei die Möglichkeit beachtet werden, eingehende Verbindungen auf den Ports 5198 und 5199 freizugeben, die Anwendung eines Proxies kann hier nicht empfohlen werden. Für die Verbindung mit entfernten TRX ist neben der Portfreigabe auch auf eine genügende Bandbreite zu achten, im Falle hoher Qualität können dies mehr als 128 kBit/s sein. Der Einsatz des GSM-Codecs ist ebenfalls möglich, reduziert jedoch natürlich nicht nur die Datenbandbreite sondern auch die Audioqualität.

2.4.2 Voraussetzungen

Grundsätzlich kann SvxLink auf jeder Hardwareplattform betrieben werden, für die ein Linux-Derivat existiert und die über die weiteren Voraussetzungen (s.u.) verfügt. Entwickelt wird das Programm auf einem X86-PC, weshalb diese Architektur die bevorzugte sein dürfte. Ein Einsatz beispielsweise in einer PowerPC- oder sogar Embedded-Umgebung ist jedoch ebenfalls durchaus vorstellbar und in Ansätzen bereits realisiert.

Hinsichtlich der Leistungsfähigkeit des Systems ist SvxLink relativ anspruchslos. Ein Pentium I oder II mit 128 bzw. 256 MB RAM wurden bereits erfolgreich eingesetzt. Anstelle einer Festplatte kann zur Reduzierung des Energieverbrauchs beispielsweise auch ein Compact-Flash-Kartenleser für den IDE-Anschluss zum Einsatz kommen. Die Einrichtung in virtuellen Umgebungen wurde von mir ebenfalls bereits erfolgreich getestet; da es sich hier jedoch um eine Echtzeitanwendung handelt, rate ich von dieser Lösung grundsätzlich ab.

Folgende weitere Mindestvoraussetzungen sollte ein System erfüllen, um für SvxLink geeignet zu sein:

- Systemarchitektur, auf der Linux arbeitet und ein GCC-Compiler zur Verfügung steht (x86-PC bevorzugt)
- Soundkarte, die zur Wiedergabe von Stereosignalen geeignet ist und von ALSA [ALSA] unterstützt wird bzw. für die Treiber angeboten werden. [2]
- Serieller Port oder USB-Seriell-Adapter, der vom verwendeten Linux unterstützt wird bzw. für den Treiber kompiliert werden können.
- Für EchoLink, Metar, PropagationMonitor: Internetverbindung mit möglichst geringer Latenz (DSL, ISDN, notfalls auch G3/UMTS)

Wie bereits ausgeführt, basiert SvxLink auf Linux, weshalb dieses als Betriebssystem natürlich Voraussetzung ist. Welche Distribution eingesetzt wird, hängt hauptsächlich von Wissen, Vorkenntnissen und Neigungen des Anwenders ab.

Der Autor SM0SVX entwickelt SvxLink unter Fedora Core [Fedora]. Daher stellt er auf der offiziellen SvxLink-Webseite [SvxLink] RPM-Pakete für diese Plattform zur Verfügung. Ebenfalls fertige Pakete herunterladen können die Benutzer von Debian [Debian], diese können in aller Regel auch unter Ubuntu [Ubuntu] eingesetzt werden, welches in wesentlichen Teilen auf Debian basiert.

Die in Deutschland verbreitete Linux-Distribution openSUSE [openSUSE] wird zwar nicht direkt unterstützt; Tim Fischer, DG7GT, macht sich jedoch gemeinsam mit Jan-Simon Möller, DL9PF, die Mühe, eine umfangreiche Paketquelle rund um den Amateurfunk für die diversen openSUSE-Versionen zu pflegen, in welcher auch die jeweils aktuelle Version von SvxLink als fertiges Paket zu finden sein sollte [SUSEHAM]. Nach unseren Erfahrungen läuft SvxLink unter openSUSE 10.x und 11.x mit ALSA und OSS-Emulation [ALSA] sehr stabil und störungsfrei.

[2] Es hat sich gezeigt, dass an dieser Stelle der Preis der Soundkarte nicht auf deren Eignung für SvxLink schließen lässt. Oftmals sind einfache Karten ohne 5.1-Sound - auch on-Board-Lösungen - teureren Produkten vorzuziehen.

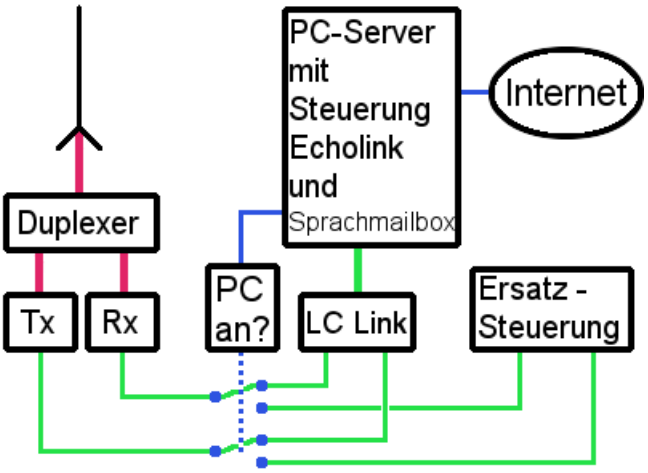

Abbildung 1: Skizze des Relais-Aufbaus bei DB0OSN

2.4.3 Installation

Wie bereits im vorangegangenen Kapitel ausgeführt, kann SvxLink am einfachsten mittels der verfügbaren RPM oder DEB-Pakete installiert werden. Ist dies nicht möglich, so steht auf der offiziellen Webseite [SvxLink] - wie bei Linux-Programmen üblich - natürlich auch der Quellcode zum Selberkompilieren zur Verfügung. Auf die genauere Prozedur kann in diesem Skript aufgrund des Umfangs jedoch leider nicht eingegangen werden, vergleiche hierzu einschlägige Linux-Literatur und Einführungswerke. SvxLink benötigt außerdem diverse Systembibliotheken. Welche dies sind und welche Versionen jeweils benötigt werden, kann auf der SvxLink-Webseite [SvxLink] unter dem Menüpunkt “Installation Instructions” nachgelesen werden.

2.4.4 Erste Inbetriebnahme

- Die Basiskonfiguration wird wie bereits in Kapitel 2.2.4 eingeführt in der Datei /etc/svxlink.conf vorgenommen.
- Die Konfigurationen der einzelnen Module, sofern diese über eine eigene verfügen, werden in /etc/svxlink.d/ abgelegt. Der Name wird dabei aus dem Wort “Module”, gefolgt vom Modulnamen und der Endung .conf gebildet.
- SvxLink legt seine Dateien in /usr/share/svxlink/ ab. Relevant ist hier eigentlich nur das Verzeichnis sounds/. Wichtig und im Anfang sicher etwas verwirrend ist, dass sich hier neben den Audio-Dateien auch die Logics verbergen.

2.5 Einsatzbeispiele

2.5.1 Relais DB0OSN Osnabrück

Bei DB0OSN in Osnabrück wird SvxLink seit Januar 2007 konsequent für die gesamte Steuerung eingesetzt. Die Abbildung 1 auf Seite 5 soll Einsatz und Aufbau der Station zeigen und so ein mögliches Einsatzszenario verdeutlichen:

Es ist zu sehen, dass das System im Prinzip mit einem minimalen Aufwand an Hard- und Software auskommt. Als Transceiver werden hier Motorola-Funkgeräte vom Typ GM 900 eingesetzt. Diese werden über ein typisches Audio-Interface mit dem PC verbunden. Das Squelch-Signal wird hierbei an der seriellen Buchse der Transceiver abgenommen. Der PC ist über ein Netzwerk mit dem Internet verbunden. Da sich das Relais direkt an einem Universitäts-Standort befindet, besteht eine dauerhafte Verbindung mit statischer IP-Adresse. Die Wartung von SvxLink wird zum überwiegenden Teil über Internet und SSH erledigt. Der Standort muss nur dann aufgesucht werden, wenn Modifikationen oder Reparaturen an der Hardware notwendig werden.

SvxLink spricht standardmäßig Englisch. Darüber hinaus entspricht die Ablaufsteuerung der Repeater-Funktionen nicht dem in Deutschland gebräuchlichen und zum Teil auch gesetzlich festgelegten Schema, wonach etwa alle 10 Min. im Betrieb die Kennung ausgegeben werden muss. Wir haben daher sowohl deutsche Sprachdateien erstellt als auch die Logiken entsprechend angepasst. Das Relais verhält sich nun wie folgt:

- Das Relais wird mit 1750-Hz-Tonruf aufgetastet.
- Nach Auftastung bleibt der Sender für 30 Sekunden getastet. Anschließend kann das Relais für weitere 30 Sekunden mittels PTT wieder geöffnet werden.
- Bei Auftasten des Relais wird eine Kennung ausgegeben, wobei Sprache und CW gemischt werden. Hier wird ein Zähler gestartet, der von nun an alle 10 Minuten die Kennung in CW ausgibt, bis das Relais wieder abfällt. Während ein Signal auf der Eingabe anliegt, wird die Lautstärke der Kennung durch eine Konfigurationseinstellung in der svxlink.conf um 6 dB gegenüber normal abgesenkt, um hörbar zu bleiben ohne das Signal des Benutzers zu beeinträchtigen.
- Zur vollen Stunde wird eine Kennung mit anschließender Ansage der Uhrzeit und ggf. vorhandener Mailboxnachrichten ausgegeben.

Wer keine Nachlaufzeit des Trägers mag, kann diese leicht in der Konfiguration deaktivieren und muss hierfür die Logic nicht bearbeiten. In diesem Fall müsste jedoch der Zähler für die Kennung leicht modifiziert werden, damit trotzdem alle 10 Minuten eine Kennung ausgegeben wird.

Das fertige deutsche Sprachpaket mit Logiken kann von unserer Webseite heruntergeladen werden [IAO].

Um bei PC-Abschaltungen oder möglichen Ausfällen über eine Backup-Steuerung zu verfügen, wurde eine minimale Relais-Steuerung auf herkömmlicher Controller-Basis entwickelt, welche die Grundfunktionen (Sprachbetrieb und Kennung) auch bei deaktiviertem SvxLink gewährleistet. Um beide Steuerungen zu synchronisieren und die Backup-Steuerung im Notfall schnell aktivieren zu können, wurde ein kleiner Watchdog entwickelt, welcher vom PC in regelmäßigen Abständen “getriggert” wird. Unterbleibt dieses Ansprechen, schaltet der Watchdog auf die Backup-Steuerung um und lässt diese so lange aktiv, bis er vom PC erneut angesprochen wird. Nicht in der Grafik dargestellt sind zwei unterbrechungsfreie Stromversorgungen, welche einen Betrieb auch bei Stromausfall aufrecht erhalten können. Eine USV ist hierbei so mit dem Server verbunden, dass sie diesen bei Auftreten eines Stromausfalls automatisch herunter fährt, um den Verbrauch zu reduzieren. Anschließend kann über die Backup-Steuerung und die Transceiver, die an der zweiten USV angeschlossen sind, unterbrechungsfrei weitergearbeitet werden.

2.5.2 Niederländisches “Poldermodell”

In den Niederlanden bestand das Problem, dass aufgrund der Topographie an der Küste keine hohen Relais-Standorte zur Verfügung stehen, so dass ein Relais niemals eine große Reichweite erzielen konnte. Gleichzeitig waren die einzelnen Regionen durch Deich- und Wehranlagen so durchschnitten, dass es schwierig war, Kontakt über diese “Grenzen” hinweg zu halten.

Hier wurde auf Basis von SvxLink ein Relais-Netzwerk aufgebaut, dass nach letzten Informationen fünf Umsetzer beinhaltet. Diese decken jeweils eine kleinere Fläche ab, an den Grenzen dieser Flächen muss der Benutzer einen Frequenzwechsel durchführen und kann dann sein QSO nahezu unterbrechungsfrei fortführen. Wichtig für dieses Konzept ist allerdings ein zentraler Server, der die Verbindungen der einzelnen Relais annimmt und die Informationen sternförmig verteilt (vgl. Kapitel 2.6).

Weitere Informationen in niederländischer Sprache findet man auf der Webseite des Repeaters Lelystad, der mit den Umsetzern in Ylst, Emmeloord, Almere und Soest verbunden ist [Holtrop].

2.5.3 Minimales System auf Embedded-Basis

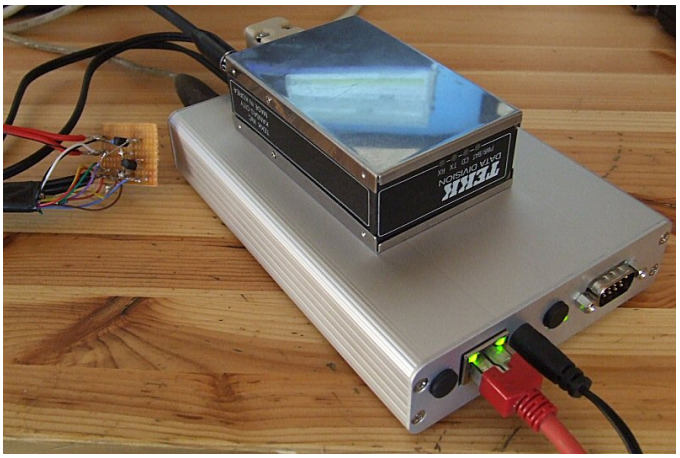

Abbildung 2: Beispiel eines minimalen Systems mit SvxLink

Die Grafik soll verdeutlichen, wie klein und stromsparend ein tatsächlich funktionsfähiges System mit SvxLink sein kann. Es ist zu sehen, dass auch mit wenig Ressourcen ein effektives und vor allem sparsames System aufgebaut werden kann, welches etwa auch an DFMG-Standorten kein Problem darstellt.

2.6 Das Konzept abgesetzter Sender und Empfänger

Anders als etwa Asterisk/app_rpt erlaubt SvxLink derzeit noch keine Koppelung mehrerer Umsetzer in einem privaten Netzwerk - dies kann allenfalls über einen Umweg realisiert werden, in dem mehrere Logiken über eine “Link”-Sektion zusammengeschaltet werden (vgl. Kapitel 2.2.4). Stattdessen verfolgt SvxLink ein Konzept, bei dem externe Sender und Empfänger mit Hilfe eines Zusatzprogramms an SvxLink angelinkt werden können. Hierbei muss auf dem entfernten System das Programm remotetrx [remotetrx] laufen, welches Bestandteil des SvxLink-Programmpakets ist, so dass dieses auf dem entfernten System ebenfalls installiert werden muss. Anstelle der svxlink.conf kommt jedoch hier die remotetrx.conf [remoteconf] zum Einsatz, in der neben den Einstellungen für die angeschlossene Hardware auch die Verbindungsparameter definiert werden müssen. Ist dies geschehen, kann der

entfernte Sender oder Empfänger in der svxlink.conf des Hauptsystems eingetragen werden, so dass der SvxLink-Server über die TCP/IP-Verbindung auf das entfernte System zugreifen kann. Die Verbindung wird also immer von SvxLink zu remotetrx und nicht umgekehrt aufgebaut.

Es ist möglich, ein oder mehrere lokale mit ein oder mehreren entfernten Sendern und Empfängern zu koppeln. Sobald mehrere Empfänger ins Spiel kommen, wird in aller Regel auch der Voter benötigt werden, welcher den Empfänger mit dem jeweils besten Nutzsignal auswählen soll und in Kapitel 2.2.4 näher erläutert wird.

Sofern das remotetrx-System nicht über eine statische IP-Adresse verfügt, sollte ein dynamischer DNS-Dienst wie etwa dyndns.org eingesetzt werden. Auf diese Weise kann erreicht werden, dass das System trotz wechselnder IP immer unter dem selben Domainnamen erreichbar ist.

2.7 Die Community

Abschließend soll noch auf die Community hingewiesen werden, die sich weltweit rund um SvxLink gebildet hat. Obwohl das Programm in Deutschland zum Zeitpunkt der Erstellung dieses Skripts nur von relativ wenigen Stationen eingesetzt wird, sind einige deutsche OM mittlerweile stark in die Weiterentwicklung involviert und haben bereits mit zahlreichen Patches zur Verbesserung beigetragen. Aber auch im europäischen Ausland sowie den USA hat SvxLink bereits zahlreiche Freunde gefunden, die aktiv in der Community mitarbeiten und Einsteigern gerne helfen. Auch der Programmautor Tobias Blomberg, SM0SVX steht häufig mit Rat und Tat zur Seite; allerdings beantwortet er sehr ungern private E-Mails sondern verweist stattdessen auf die zugehörige Mailingliste [SvxMailinglist]. Zu dieser kann man sich mit einer leeren E-Mail an svxlink-devel@lists.sourceforge.net anmelden. Darüber hinaus existiert ein Archiv[SvxMLArchiv], in welchem alle bisher in der Mailinglist veröffentlichten Beiträge nachgelesen werden können.

2.8 Kritik und Ausblick

SvxLink hat sich bislang immer als sehr stabiles System auf den uns bekannten Umsetzern bewährt. Dies beinhaltet, dass SM0SVX von den Quelltexten, die ihm zur Integration in SvxLink zugeschickt werden, auch ein gewisses Maß an Code-Qualität erwartet. Auch ist es ratsam, sich zuvor seinen Sourcecode anzuschauen und die eigene Syntax daran zu orientieren.

Das Konzept von SvxLink zum Betrieb abgesetzter Transceiver ist interessant, stößt jedoch dort an seine Grenzen, wo es darum geht, mehrere Relais miteinander zu vernetzen. Hier besteht aus unserer Sicht Handlungsbedarf, dessen Ergebnis die Möglichkeit zur Vernetzung zweier SvxLink-Systeme sein sollte.

Wie im folgenden Abschnitt 3 noch gezeigt werden wird, ist Asterisk nicht nur eine äußerst flexible Telefonanlagen-Software sondern bietet aufgrund der hohen Verbreitung sowie der Modularität auch interessante Möglichkeiten zur Vernetzung mit neuen Systemen wie IRLP oder D-Star. Es sollte daher überlegt werden, in wie weit SvxLink mit Asterisk verbunden werden kann. Einen ersten Ansatz könnte hier das verbreitete IAX-Protokoll bieten, welches extra zur Kommunikation mit Asterisk entwickelt wurde und im folgenden ebenfalls noch eine Rolle spielen wird.

Interessant erscheint uns darüber hinaus auch eine Funktion zum Monitoren des Systems sowie eine Möglichkeit der Fernwartung, etwa über ein Webinterface oder auch ein Modul, dass auf DTMF-Kommandos reagiert.

Wie bei allen Open-Source-Anwendungen gilt wohl auch hier: Es gibt nichts gutes außer man tut (programmiert) es.

3 Asterisk/app_rpt

3.1 Einführung

Asterisk ist als softwarebasierte Telefonieanlage für Linux bekannt. Im Heimbereich kommt die Software, vom Anwender oft unbemerkt, in Form einer aktuellen FritzBox ins Haus und stellt die Internettelefoniefunktion bereit. Das Softwaremodul app_rpt der Kategorie “Applications” stellt die Repeaterfunktionalität zur Verfügung.

Bei der Softwarekonzeptionierung wurde von Anfang an auch an die Integrierung von Funkanwendungen gedacht. Jim Dixon, WB6NIL, hat dies zur Bedingung für seine Mitarbeit an Asterisk gemacht, die zusammen mit Mark Spencer (Firma Digium), dem Ur-Vater von Asterisk, auch gefruchtet hat. Jim hat zusammen mit Steve Rodgers, WA6ZFT, die Entwicklung des Moduls app_rpt gestartet. Zur Anbindung von Funkhardware wurde zunächst die “Quad Radio PCI card” von Steves Firma QRVC Communications genutzt.

Der genaue Verlauf des Asterisk/app_rpt-Projekts kann auf der Webseite von Steve Lampereur, KB9MWR, nachgelesen werden [KB9MW].

3.2 Voraussetzungen

3.2.1 Erforderliche Hardware

Interface Der breite Einsatz von Asterisk/app_rpt wird durch den kostengünstigen USB-Sound-IC CM-108 von C-Media möglich. Steve Henke, W9SH, hat mit seinen Erfahrungen in DSP-Programmierung für Asterisk ein Modul der Kategorie “Channels” für diesen Sound-Chip entwickelt. Es ermöglicht das Zusammenspiel mit Asterisk/app_rpt.

Soundkarten mit dem CM-108-IC gibt es im Internet zu finden. Auch die neueren Modelle CM108AH, CM109 und CM119 funktionieren. Die Soundkarten müssen modifiziert werden, da zumindest eine der drei Steuerungsleitungen des ICs herausgeführt und als PTT-Leitung verwendet werden muss. Entsprechende Beschreibungen sind verfügbar [USBFOB].

Persönlich empfehlen wir den Einsatz des sogenannten URI (Universal-Radio-Interface) von DMK Engineering. Das Interface enthält den CM-108-IC und ist auf den Einsatz von Amateurfunkanwendungen zugeschnitten. Das Datenblatt mit weiteren Informationen ist auf der Homepage von DMK Engineering zu finden [DMK]. Für Funkamateure ist der Preis von $99.95 auf $69.95 herabgesetzt. Es empfiehlt sich eine Sammelbestellung zu machen, da der Versand mit $36.00 sehr teuer ist. Bei einer Bestellung von drei Stück hat auch noch der Zoll zugeschlagen.

Das Interface eignet sich auch für andere Softwareprodukte wie TheLinkbox oder SvxLink. Spezielle Unterstützung mit (drei IO-Leitungen) wurde auf Empfehlung auch in “PCRepeatercontroller” von G4KLX eingebaut.

Transceiver Prinzipiell ist jeder TRX für den Einsatz von Asterisk/app_rpt geeignet. Volle Softwarefunktionalität steht erst unter Verwendung des Diskriminatorausgangs und Modulatoreingangs und dem Software-DSP für die Trägererkennung zur Verfügung (ansonsten wird eine extra Leitung für “Carrier Detect” benötigt). Richtig Spaß macht das System allerdings erst im Duplexmodus. Als Relais kann das System all seine Funktionen optimal entfalten.

Abbildung 3: USB-Soundkarteninterface von DMK Engineering

Computer Als Voraussetzung wird ein Intel Pentium III 800 MHz oder besser, 256 MB RAM, 4.5 GB HDD und Netzwerkkarte genannt. Das System ist echtzeitkritisch, weshalb vom Einsatz in einer virtuellen Umgebung abgeraten wird. Das USB-Interface sollte möglichst direkt an einen USB 2.0 Anschluss am PC und nicht über einen USB-Hub angeschlossen werden.

Netzwerk Die nötige Bandbreite der Internetverbindung hängt vom verwendeten Sprachcodec ab. Unter Verwendung des GSM-Codecs wie bei EchoLink kommt man mit Modemgeschwindigkeit aus. Ohne Internetverbindung können Repeater auch in einem geschlossenen Netzwerk wie dem HAMNET miteinander Verbindung aufnehmen.

3.2.2 Erforderliche Software

Plug 'n Play-Variante Auf der Homepage des Allstarlinknetzwerks [ALLSTAR] steht ein ISO-Image für die vollautomatisierte Installation von Asterisk/app_rpt zur Verfügung (Vorsicht: Die komplette Festplatte wird dabei gelöscht). Die Software installiert das Linux-Betriebssystem CentOS, lädt alle notwendigen Daten aus dem Internet herunter und compiliert die Software.

Installation unter Debian Für Debian-basierte Linux-Systeme hat Stephen Brown, K1LNX, Installationsskripte auf seinem Wiki abgelegt [K1LNX].

Installation auf einem bestehenden Asterisk-System Dies ist nicht möglich. Hierzu ein Zitat aus der FAQ:

> Q: Why did we fork Asterisk?
>
> A: Asterisk is undergoing a significant amount of change at Digium. Because of this, and the long time it takes to get patches approved, we decided to drive a stake in the ground and archive the Asterisk and Zaptel/Dahdi sources on our repository with the required changes to support the latest version of app_rpt. If the need arises, we may fetch a later version of Asterisk and Dahdi from Digium and make the necessary changes to it to support app_rpt (a formward port). Back pointing changes to support app_rpt into the Digium source tree at this point does not seem likely at least at this point in time.

3.3 Beschreibung der Software

3.3.1 Features

- Die Software ist ein "open source"-Projekt und ist unter der GPL lizensiert. Das heißt, dass der Sourcecde vorliegt und beliebige Änderungen zur Erweiterung der Funktionalität gemacht werden können. Diese können wieder an die Gemeinschaft zurückfließen.
- Es können mehrere Interfaces und Transceiver mit einer Softwareinstallation angesteuert werden.
- Es gibt keine zentrale Instanz zur Autorisierung von Verbindungen. Man kann das Allstarlinknetzwerk verwenden, aber auch ein eigenes komplett abgeschlossenes Netzwerk aufbauen.
- Alle Sicherheits-Features von Asterisk können genutzt werden (Public/Private keys, IP-Adressfilterung) und/oder das Authentifizierungverfahren von Allstarlink.
- DTMF-Kommandos können über einen entfernten Knoten auf dem lokalen Repeater ausgeführt werden.
- app_rpt ist ein Repeater-Controller und nicht nur ein Sprachgateway.
- app_rpt und Asterisk können auf einem Embedded-PC mit 13.8V-Spannungsversorgung mit der Limey-Linux-Distribution und einer Compact-Flash-Karte betrieben werden.
- Sowohl die Einwahl ins Telefonnetz als auch eingehende Verbindungen aus dem Telefonnetz sind über Voice-over-IP oder traditionelle Telefonleitungen (ISDN/analog) machbar.
- Zugriff auf fernbedienbare Transceiver (Kurzwelle, VHF/UHF) mit individueller Authentifizierung.
- Ad-Hoc-Verbindungen sind möglich. Eine beliebige Anzahl von Knoten können miteinander verbunden sein. Limitierend ist nur die verfügbare Bandbreite.
- IAX2/ADPCM (g726aal2-Sprachcodec) nutzt 55 Kilobit/s Bandbreite. Optional kann GSM mit 35 Kilobit/s genutzt werden.
- Interoperabilität mit Echolink und IRLP
- Steuerung und Zugriff auf den Repeater über Einwahlverbindungen
- Remote Console über die Windowssoftware iaxRpt
- Vollduplex-Fähigkeit

3.3.2 Konfiguration

Ein guter Start, um sich in das Themengebiet einzuarbeiten, ist die Webseite zum Projekt [apprpt]. Leider ist die Bedienung der Seite etwas ungewöhnlich, so dass man sich nicht gleich zurecht findet. Die gewohnte Navigation findet man auf der rechten Seite in Form eines aufklappbaren Baumes. Starten sollte man allerdings mit der Leiste ganz oben. Unter "Knowledge Base" gibt es eine gute Einführung zum System und die offizielle Dokumentation.

Das "ACID System Administrators Manual" stellt unter anderem die wichtigsten Konfigurationsdateien vor:

/etc/asterisk/usbradio.conf	USB-Soundkartenkonfiguration
/etc/asterisk/rpt.conf	Repeater-Konfiguration
/etc/asterisk/iax.conf	IAX (inter-asterisk-exchange) Konfiguration
/etc/asterisk/extensions.conf	Asterisk-Dialplan

Die einzelnen Konfigurationsabschnitte werden “stanzas” genannt und mit eckigen Klammern gekennzeichnet. In diesen Abschnitten gibt es Schlüssel und Werte (“key/values”).

Die USB-Soundkartenkonfiguration ist selbsterklärend. Zur Dialplankonfiguration empfiehlt es sich, sich in die Grundlagen zu Asterisk einzulesen [Asteriskbuch]. Für die initiale Konfiguration muss die Datei aber nicht geändert werden. Die IAX-Konfiguration wurde bei der Installation bereits durch ein Skript entsprechend angepasst.

Ein paar wertvolle Hinweise möchte ich zur Repeater-Konfiguration geben. Im Abschnitt “Nodenummer” verweisen die folgenden Zeilen auf weitere Abschnitte in der gleichen Datei (rpt.conf):

```
;controlstates=controlstates
; system control state list
;scheduler=schedule
; scheduler entries
;functions=functions
; Repeater Function Table
phone_functions=functions
; Repeater Function Table
link_functions=functions
; Link Function Table
telemetry=telemetry
; Telemetry descriptions
wait_times=wait-times
; Wait times
```

Besonders wichtig ist der Verweis zum Abschnitt “functions”. Aktuell sind alle Funktionen für Nutzer, die den Repeater über Telefon, Funk oder über einen anderen Knoten steuern, im Abschnitt “functions” definiert. Bei Bedarf kann hier an andere Abschnitte gesprungen werden.

Im Abschnitt “functions” werden die DTMF-Ziffern mit der wahren Funktion verlinkt. Das Schlüssel-Wertepaar “984=ilink,7” bedeutet hier, dass die DTMF-Zeichenfolge “984” die Funktion “ilink” mit dem Argument “7” aufruft. Ein User kann mit dem DTMF-Zeichen “*” Funktionen aufrufen. Der User tippt also “*984” in sein Funkgerät, um die Funktion “ilink,7” aufzurufen.

Die Beschreibung der meisten Funktionen findet sich am Anfang des Sourcecodes von app_rpt in der Datei /usr/src/app_rpt/asterisk/apps/app_rpt.c. In diesem Fall sind zwar einige Funktionen von “ilink” definiert, aber nicht das Argument “7”. Eine kurze Suche im Sourcecode ergab:

```
case 7:
/* Identify last node which keyed us up */
rpt_telem_select(myrpt,command_source,mylink);
        rpt_telemetry(myrpt, LASTNODEKEY, NULL);
        break;
```

Wer möchte, kann “ilink,7” oben im Kommentarabschnitt hinzufügen und einen Patch von app_rpt.c schicken :)

3.3.3 Standardisierte User-Befehle

Im Allstarlinknetzwerk gibt es standardisierte Nutzerbefehle, die nicht geändert werden sollten. Es handelt sich um folgende Kommandos:

Befehl	Funktion
1<node>	Verbindung trennen
2<node>	Verbindung im Monitormodus herstellen
3<node>	Verbindung herstellen (Transceive-Mode)
4<node>	Befehlsmodus auf einem entfernten Knoten aktivieren
7	Status der Verbindungen abrufen

Die Nodenummer <node> baut sich wie folgt auf:

- **Allstarlink:** DTMF der 4- oder 5-stelligen Allstarlinknodenummer (die erste Ziffer ist immer eine 2)
- **Echolink:** DTMF 3, nachfolgend DTMF der sechsstelligen EchoLink-Nummer (4- oder 5-stellige Knoten werden mit führenden Nullen eingegeben)
- **IRLP:** DTMF 4, nachfolgend DTMF der vierstelligen IRLP-Nummer

Beispiele: Knoten mit DB0FUE (EchoLink #8510) verbinden: DTMF *33008510
Knoten von DB0RDT (IRLP #5822) trennen: DTMF *145822
Knoten mit DB0FOX (Allstarlink #2397) verbinden: DTMF *32397

Bemerkungen:

- Die Nodenummer “0” ist eine Abkürzung für die zuletzt verwendete Nodenummer.
- Mit dem Monitormodus kann auf einem Knoten zugehört werden, ohne dass Audio dorthin übertragen wird.
- Den Befehlsmodus kann man nur aktivieren, wenn der entsprechende Knoten bereits verbunden ist.

Die Bedienung ist damit recht komplex. Die Funktionalitätsvielfalt erfordert eine logische Strukturierung der Nutzerbefehle. Kurzbefehle lassen sich über Makros realisieren.

3.4 Praxisbeispiel DB0VOX und DB0FOX

3.4.1 Historie

An DB0VOX “Fernmeldeturm Nürnberg” war jahrelang ein selbstentwickeltes Sprachmailboxsystem mit Repeaterfunktion im Einsatz. Das in Delphi programmierte “Amateur Voice Network System” (AmVoNeS) für Windows hat im Laufe der Zeit einen großen Funktionszuwachs erfahren. Zuletzt wurde die Emulation eines IRLP-Boards des “Internet Radio Link Projekt” über die parallele Schnittstelle fertig gestellt. Ein zweiter PC auf Linux-Basis hat für die Anbindung an IRLP und EchoLink gesorgt.

Aufgrund von notwendigen Stromsparmaßnahmen wurde das Relais nach DB0FHN “Hochschule Nürnberg” verlegt. Da der Einzugsbereich massiv eingeschränkt wurde, haben wir uns Gedanken gemacht, wie man das Relais unter Beibehaltung des Funktionumfangs an DB0VOX mit stromsparender Technik wieder reaktivieren könnte. Eine erste Überlegung war der Einsatz von SvxLink mit abgesetztem TRX, die aber aufgrund der Emigration beteiligter Projektmitglieder wieder verworfen wurde.

Letztendlich haben wir uns zur Stilllegung des AmVoNeS-Systems an DB0FHN entschieden und das Relais an DB0VOX mit Embedded-Hardware unter Einsatz von Asterisk/app_rpt wieder reaktiviert. Langfristiges Ziel ist die Implementierung der noch fehlenden Funktionen des alten Systems.

3.4.2 Aufbau und erste Inbetriebnahme

Es kommen zwei Motorola GM340 Mobilfunkgeräte zum Einsatz. Durch Aktivierung der Option “Data Channel enabled” werden der Diskriminatorausgang und der Modulatoreingang aktiviert. Vom USB-Interface gehen also vier Leitungen (TX, RX, PTT und Gnd) zu den Funkgeräten.

Die Beschreibung zur exakten Einstellung der Audiopegel ist im Administrator Guide Kapitel “Adjusting Audio Levels” zu finden. Zunächst

startet man das “Command Line Interface” von Asterisk mit asterisk −r auf der Linux-Console.

Das Rauschen wird mit “radio tune rxnoise” gemessen. Danach legt man am Empfänger einen 1kHz-Sinus mit 3kHz Hub an und tippt “radio tune rxnoise”. Zur Subtonerkennung legt man einen CTCSS-Ton mit 0,6kHz Hub an und tippt “radio tune rxtone”. Die Einstellungen werden mit “radio tune save” gespeichert.

Bevor der Sendezweig eingestellt wird, wird der Subton mit “radio tune txtone 0” deaktiviert. Danach wird mit “radio tune txvoice <wert zwischen 0 und 999>” die Aussendung gestartet und der entsprechende Wert mit einem Hub von 3kHz gesucht. Mit “radio tune txtone <wert zwischen 0 und 999>” sucht man den geeigneten Wert für den Subton mit einem Hub von 0,6kHz. Die Einstellungen werden mit “radio tune save” gespeichert.

Alle Werte werden in /etc/asterisk/usbradio_tune_usb.conf gespeichert. Für erste Versuche können die Werte auch manuell gesetzt werden. Die aktuellen Einstellungen können auch jederzeit abgerufen werden. Hier am Beispiel von DB0VOX:

```
db0vox*CLI> radio tune rxsquelch
Current Signal Strength is 711
Current Squelch setting is 800
db0vox*CLI> radio tune txvoice
Current txvoice setting on Channel A is 400
Tone output starting on channel usb...
Tone output ending on channel usb...
db0vox*CLI> radio tune txtone
Current Tx CTCSS modulation setting = 215
```

3.4.3 Konfiguration und Betrieb

Zunächst muss die usbradio.conf an unser Hardwaresetup angepasst werden. Mit der Option “data channel enabeld” haben wir “flat audio”:

```
rxdemod=flat
; input type from radio: no,speaker,flat
; flat − Use RX audio from discriminato
; (before de−emphasis)
txprelim=yes
; Audio processing on left output channel:
; no,yes
; yes − Audio is pre−emphasized and limited.
; Suitable for direct connection
; to an FM modulator
```

Mit “flat audio” können wir die Rauschsperre über Software realisieren:

```
carrierfrom=dsp
; no,usb,usbinvert,dsp,vox
; dsp − from RX noise using dsp techniques
```

Auch die Erkennung von Subtönen ist damit möglich:

```
ctcssfrom=dsp
; no,usb,dsp
; dsp − CTCSS decoding using RX audio in DSP.
```

Da der Audiopfad stets durch den PC verläuft, ist die Ausgabe vom Repeater in etwa 100ms zeitverzögert. Dadurch ist es möglich, das Nachrauschen beim Schließen der Rauschsperre am Repeater auf dem Sendezweig und die Eingabe von DTMF-Zeichn zu unterdrücken. Das Nachrauschen beim Funkgerät des Nutzers kann durch Einsatz von Subtönen ausgeblendet werden. Der Repeater wird entsprechend konfiguriert:

```
txctcssdefault=88.5
; default tx ctcss freq, any frequency permitted
txtoctype=notone
; Transmit tone control type: no,phase,notone
; notone − encode CTCSS and stop sending
; tone before unkeying TX
```

Der Nutzer muss nun den CTCSS-Auswerter auf 88,5Hz einstellen. Da der Repeater kurz vor dem Abschalten des Senders den Subton deaktiviert, wird vor dem Abfallen des Signals beim Nutzer der Audiokanal stumm geschaltet.

Auf DB0VOX sind viele verschiedene Nutzergruppen QRV. Wir haben uns darauf geeinigt, dass jede Gruppe einen eigenen Subton zum Rufen anderer Teilnehmer der Gruppe nutzt. Nach dem Ruf wird der Subton wieder deaktiviert oder auf den Standardsubton des Repeaters geschaltet. Dies ermöglicht es am Repeater erreichbar zu bleiben, wenn normale Gespräche ausgeblendet werden müssen. Entsprechende Unterstützung wurde erst kürzlich in die Software eingebaut:

```
rxctcssfreqs=67.0,71.9,74.4,77.0,79.7,82.5,85.4,
88.5,91.5,94.8,97.4,100.0,103.5,107.2,110.9,
114.8,118.8,123.0,127.3,131.8,136.5,141.3,146.2,
151.4,156.7,162.2,167.9,173.8,179.9,186.2,192.8,
203.5,210.7,218.1,225.7,233.6,241.8,250.3
; rx ctcss freqs in floating point. must be in
; table
txctcssfreqs=67.0,71.9,74.4,77.0,79.7,82.5,85.4,
88.5,91.5,94.8,97.4,100.0,103.5,107.2,110.9,
114.8,118.8,123.0,127.3,131.8,136.5,141.3,146.2,
151.4,156.7,162.2,167.9,173.8,179.9,186.2,192.8,
203.5,210.7,218.1,225.7,233.6,241.8,250.3
; tx ctcss freqs, any frequency permitted
rxctcssoverride=1
; Set to 1 or yes to start out in carrier
; squelch mode
```

Der bemannte Crossband-Betrieb über DB0VOX ist ausdrücklich erlaubt, wenn der Nutzer zum Beispiel durch Absicherung mit CTCSS dafür sorgt, dass keine Störungen übertragen werden. Die Ausgabefrequenz von DB0VOX wird dabei vom Crossband-Betreiber auf eine andere Frequenz und zurück umgesetzt. Der Repeater muss dazu ohne Nachlaufzeit konfiguriert werden. Entsprechende Änderungen wurden in rpt.conf gemacht:

```
hangtime=0
; squelch tail hang time (in ms) (optional)
unkeywait=200
; Time to wait after unkey before sending CT's
; and link
```

Vor einigen Jahren bin ich nach Augsburg umgezogen und konnte den Repeater in Nürnberg nicht mehr direkt arbeiten. Der Betrieb über einen lokalen EchoLink-Einstieg war oft nicht möglich, da der Repeater in Nürnberg bereits lokal in Benutzung war. Konnte eine Verbindung hergestellt werden, waren die Sprechpausen in Nürnberg oft viel zu kurz, so dass ein Gespräch mit drei oder mehr Teilnehmern nur mit strikter Mikrofonübergabe möglich war.

Nachdem in Augsburg ein gutes QTH für einen weiteren Repeater ausgemacht werden konnte, ist nun neben DB0VOX (FMT Nürnberg) auch DB0FOX (BLLV Wohnheim Augsburg) mit Asterisk/app_rpt on air. Beide Repeater wurden permanent über das Allstarlinknetz miteinander verlinkt. Die Audioqualität und Latenz der Verbindung hängt von den verwendeten Sprach-Codecs ab. Im Abschnitt “radio” der Datei iax.conf wird die Priorität entsprechend festgelegt:

```
allow=ulaw
allow=g726aal2
allow=gsm
```

Trotz zweier Wireless-LAN-Strecken in Nürnberg und Augsburg ist die Round-Trip-Time (RTT) zwischen den Knoten im Schnitt nur 25ms. Beide Repeater arbeiten mit statischen IP-Adressen aus dem Internet und können daher die Verbindung permanent ohne Neuaufbau halten. Im praktischen Betrieb kann ein Endnutzer nicht mehr unterscheiden, ob ein Nutzer gerade über DB0VOX oder DB0FOX spricht. Ein rudimentärer Latenztest kann durch einfaches Heraufzählen mit Mikrofonübergabe zwischen Nutzern beider Systeme durchgeführt werden. Eine Verzögerung konnte nicht bemerkt werden.

Eine gute Unterscheidung von wo ein Signal kam, kann durch Modifzierung der Tonsequenz nach einem Sprachdurchgang gemacht werden. Bricht die Internetverbindung ganz zusammen, kann auch eine extra Sequenz für den unverbundenen Zustand definiert werden. Der Abschnitt "telemetry" in rpt.conf lässt alle Freiheiten zu:

```
remotetx=|t(633,0,50,1000)(0,0,80,0)
        (1209,0,50,1000);
```

Das "t" steht für eine Tonsequenz. Ein Ton wird in Klammern definiert. Die ersten beiden Werte stehen für einzelne Töne in Hz (mit zwei Tönen können DTMF-Töne definiert werden), der dritte Wert für die Länge in Millisekunden und der vierte Wert für die Amplitude. Hier haben wir zwei Einzeltöne getrennt durch eine Pause von 80ms. Man könnte hier die Morsezeichen "A" für Augsburg und "N" für Nürnberg definieren.

Wie bereits erwähnt, kann das System nur im Vollduplexmodus wie auf einem Repeatersystem voll ausgeschöpft werden. Zwei solche Repeatersysteme lassen sogar den Vollduplex-Betrieb zu, wenn der Nutzer zwei Funkgeräte hat. So habe ich bereits ein Vollduplex-QSO mit Jim Dixon, WB6NIL, geführt. Die Vollduplex-Fähigkeit wird für viele Funktionen benötigt. Verbindet man sich mit einem Knoten im Monitormodus könnte man sonst bei einem laufenden QSO die Verbindung nicht mehr trennen.

Alle Audiosignale werden in dem System gemultiplext. Ein weiterer Knoten kann sich jederzeit mit dem System verbinden. Eine Verbindungsansage kann in ein laufendes QSO eingemischt werden (Amplitude konfigurierbar). Ein OM von einem entfernten Knoten kann sich jederzeit bemerkbar machen, wenn ein QSO läuft.

Im Befehlsmodus kann man einen entfernten Knoten komplett fernsteuern. Ich nutze diesen Modus, um eine 5-Ton-Folge auf einen entfernten Knoten abzusetzen. Die Definition eines solchen Befehls kann im entsprechenden Abschnitt ("link_functions=functions") erledigt werden. Im Abschnitt "functions" gibt es daher folgenden Eintrag:

```
882=cop,48,!1160/70,!2400/70,
        !1400/70,!1270/70,!1830/70
```

Ein Nutzer kann also mit "*882" den Befehl "cop" mit Argument "48" aufrufen. Eine Beschreibung ist auf der Webseite in den Howtos unter "Transmitting Tone Sequences" zu finden.

Neben der Unterstützung des Allstarlinknetzwerkes ist die EchoLink-Untersützung gleich mit an Bord. Sämtliche Ansagen vom System (inkl. der Repeater-ID) werden lokal gehalten und nicht über EchoLink-Verbindungen übertragen. Verbundene Allstarlinkknoten tauschen neue Linkinformationen über das IAX-Protokoll aus.

Die Unterstützung des IRLP-Netzwerks wird über ein extra Skript installiert. Dazu ist eine IRLP-Nummer nötig. Evtl. können von der Hardware entbundene IRLP-Nummern bezogen werden. Weitere Informationen inkl. der Nutzungsbedingungen im IRLP-Netzwerk sind auf der IRLP-Webseite nachzulesen [IRLP].

Der Zugriff vom und in das Telefonnetz ist an DB0VOX über einen SIP-Provider gelöst. Im Abschnitt der verfügbaren Funktionen der rpt.conf kann die Funktion "Autopatch" aktiviert werden. Sie verweist per default an den Kontext "radio" der Datei extensions.conf (Dialplan). Dort wird die ausgehende Route über den SIP-Provider festgelegt. Für eingehende Anrufe wird ein SIP-Account in der Datei sip.conf definiert. Kommt ein Anruf über SIP herein, so wird der entsprechend definierte Kontext in der extensions.conf aufgerufen. Dort ist eine Ansage der Nodenummer implementiert, bevor der Anruf auf den Repeater durchgeschaltet wird.

Es gibt vier verschiedene Modi, wie eine Telefonverbindung mit dem Repeater zusammengeschaltet werden kann:

- Im "Dumb"-Modus wird der Sender permanent getastet. Der Schlüssel "dphone_functions" in rpt.conf zeigt dabei auf den Abschnitt mit den verfügbaren DTMF-Kommandos.
- Im normalen Modus wird der Sender nicht getastet. Der Schlüssel "phone_functions" in rpt.conf zeigt dabei auf den Abschnitt mit den verfügbaren DTMF-Kommandos. Darunter sollte ein Befehl sein, der Sendertastung ermöglicht (Befehl "cop" Argument "6").
- Im "Simple"-Modus wird der Sender mit "*" getastet und mit "#" deaktiviert. Es stehen keine weiteren DTMF-Kommandos zur Verfügung.
- Der "VOX"-Modus ist identich zum normalen Modus mit dem Unterschied, dass der Sender automatisch mit Erkennung von Sprache getastet wird.

Der entfernte Zugriff auf das Repeatersystem ist auch mit der Windows-Software iaxRpt möglich. DTMF-Sequenzen werden dabei als Zusatzinformation im IAX-Protokoll übertragen.

Asterisk selbst kann über das "Command Line Interface" (CLI) aus der Ferne gesteuert werden. Auf der Linuxebene kann man das CLI mit asterisk r aufrufen. Mit zweifachem Drücken von "Tab" werden die verfügbaren Befehle angezeigt. Die Befehle "radio" und "rpt" sind für uns von besonderem Interesse. Stellt man den Befehlen ein "help" voran, wird entsprechende Hilfe angezeigt:

```
db0vox*CLI> help radio
radio active  Change commanded device
radio key  Simulate Rx Signal Present
radio set debug  Radio Debug
radio set debug off  Radio Debug
radio set xdebug  Radio set xpmr debug level
radio tune  Radio Tune
radio unkey  Simulate Rx Signal Lusb
db0vox*CLI> help rpt
rpt cmd  Execute a DTMF function
rpt debug level  Enable app_rpt debugging
rpt dump  Dump app_rpt structs for debugging
rpt fun  Execute a DTMF function
rpt fun1  Execute a DTMF function
rpt irlpplay  Play Back an Audio File Locally
rpt localnodes  Dump list of local node numbers
rpt localplay  Play Back an Audio File Locally
rpt lstats  Dump link statistics
rpt nodes  Dump node list
rpt playback  Play Back an Audio File Globally
rpt reload  Reload app_rpt config
rpt restart  Restart app_rpt
rpt stats  Dump node statistics
```

Das sieht schieße aus.

Möchte man, dass sein Knoten auf der Statuswebseite vom Allstarlinkprojekt gelistet wird, so werden in rpt.conf entsprechende Zeilen unter "Status Reporting" auskommentiert. Außerdem ist darauf zu achten, dass das Skript /etc/rc.d/rc.updatenodelist im Hintergrund gestartet ist. Der eigene Knoten wird dann auf http://stats.allstarlink.org aufgelistet. Eine grafische Auswertung gibt es auch.

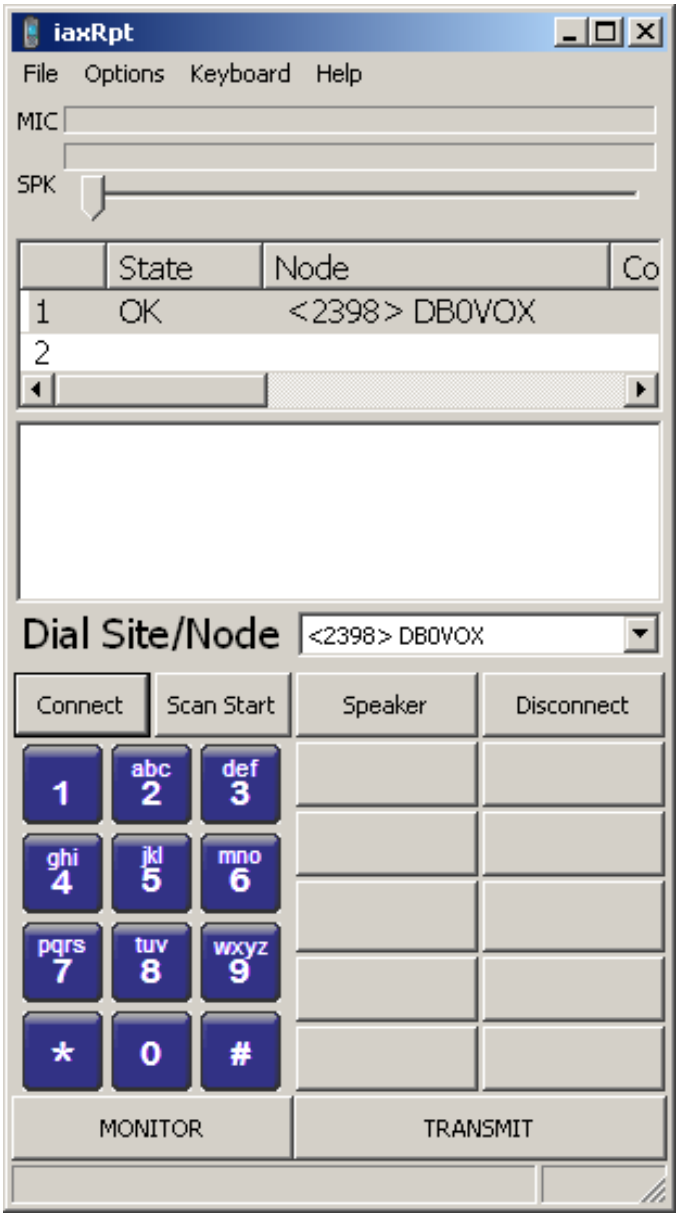

Abbildung 4: iaxRpt-Software mit Verbindung zu DB0VOX

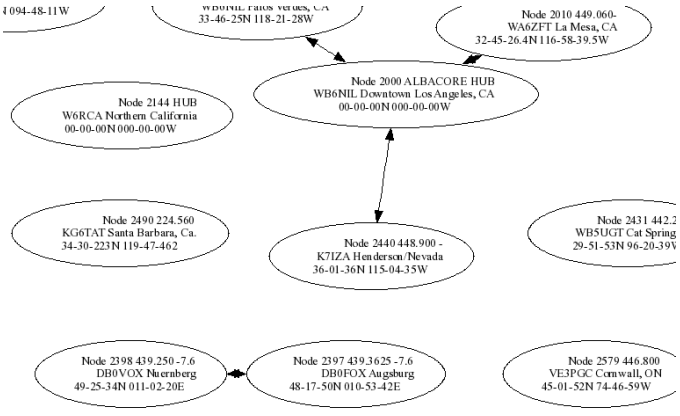

Abbildung 5: Ausschnitt der grafischen Statusauswertung von Allstarlink

Die derzeit im Netzwerk erreichbaren Knoten können auch im System mit

less /var/lib/asterisk/rpt_extnodes

abgerufen werden:

```
1 [extnodes]
2 2000=radio@67.215.233.178/2000,67.215.233.178
3 2001=radio@64.118.102.138:4569/2001,64.118.102.138
4 2002=radio@64.118.103.70:4569/2002,64.118.103.70
5 ...
```

3.5 Development und Community

Das Asterisk/app_rpt-Projekt stellt eine für Entwickler geeignete Umgebung zur Verfügung. Neben der offiziellen Homepage [apprpt] erfolgt der Zugriff auf den Sourcecode über Subversion [RptSvn]. Fehler können direkt im Bugtracking-System [Mantis] gemeldet werden. Auf der Mailingliste [RptMailinglist] können ebenso Probleme wie auch Feature-Requests gemeldet werden. Auch ein Archiv der Nachrichten ist vorhanden.

Die Anzahl der Anwender hält sich bislang noch in Grenzen, aber die Community wird sicher noch wachsen, wenn der Bekanntheitsgrad größer wird. Ich persönlich habe nur gute Erfahrungen mit dem System gemacht. Selbst eine komplette Neuinstallation ist in kurzer Zeit möglich, da nur ein paar Konfigurationsfiles das gesamte System beschreiben. Dazu kann unter /usr/src/app_rpt/asterisk einfach

make uninstall-all

und unter /usr/src/app_rpt/zaptel einfach

make uninstall-modules

und

make uninstall-hotplug

gesagt werden.

Einige Featurerequests, die ich auf der Mailingliste gestellt hatte, waren innerhalb weniger Stunden bearbeitet und der neue Code stand über Subversion zur Verfügung.

4 Ausblick

Open-Source-Projekte eignen sich hervorragend zur Erweiterung mit neuen Funktionalitäten. Die nachfolgenden Ideen sind plattformunabhängig realisierbar und sollen auf Machbarkeit mit allen Systemen geprüft werden. Auch die Interaktion zwischen verschiedenen Systemen könnte dabei zum Erfolg führen. Es wurde zum Beispiel bereits über den Einsatz von SvxLink oder TheLinkBox als Repeater-System in Interaktion mit Asterisk zur Nutzung dessen Möglichkeiten diskutiert. Eine Antwort könnte Asterisk/app_svxlink oder Asterisk/chan_svxlink sein (s. auch Kapitel 2.8.

Das folgende Kapitel ist daher als Anregung zu verstehen, welche Erweiterungen machbar wären und wie es gehen könnte. Gerade interessierte Entwickler sind aufgerufen, sich Gedanken über diese oder darüber hinausgehende Entwicklungen zu machen und diese in die Tat umzusetzen. Die Plattform Linux und der gemeinsame Kommunikationsansatz bieten hier noch viele Möglichkeiten, die auf ihre Realisierung warten.

EchoLink Generell wäre es von Vorteil, wenn alle Systeme den gleichen Minimalbefehlssatz zur Nutzung des Echolinknetzwerks implementieren würden. Die Bedienbarkeit für den Gelegenheitsnutzer sollte trotz Kritik am "EchoLink-Standard" an vorderer Stelle stehen. Die Bedienung von Asterisk/app_rpt ist in dieser Hinsicht noch nicht kompatibel. SvxLink kann zwar so konfiguriert werden, dass Nodenummern wie von EchoLink gewohnt eingegeben werden können, die Statusabfrage geschieht hier jedoch beispielsweise mit "1" anstatt "08".

Auch das Feature, ein Zielrufzeichen im SMS-Format eingeben zu können, wäre wünschenswert. Während dies bei SvxLink bereits in Teilen möglich ist, sieht app_rpt diese Möglichkeit noch nicht vor. Die alte Lösung mit AmVoNeS beherrschte es und hat sich als sehr nützlich beim täglichen Umgang mit dem Repeater-System erwiesen. Als Gelegenheitsnutzer merkt man sich eher das Repeater-Rufzeichen als dessen EchoLink-Nummer.

Sprachmailbox Gemessen an Sprachmailbox-Systemen wie AmVoNeS verfügt keines der beiden vorgestellten Programme bereits über eine vergleichbare Funktionsvielfalt. Durch das vorgestellte Modul TclVoiceMail kann SvxLink immerhin bereits als Mailbox für begrenzte Nutzerkreise eingesetzt werden. Asterisk besitzt zwar von sich aus schon eine mächtige Sprachmailbox, aber der Bedienkomfort und Möglichkeiten im Zusammenhang mit app_rpt sind noch eingeschränkt. Das händische Einpflegen von Nutzerkonten ist sehr aufwendig und sollte durch ein einfaches System (z.B. Eingabe via SMS-Format über Funk oder über eine Webseite) ersetzt werden. Durch die Vernetzbarkeit der Repeater wäre auch eine zentrale Mailbox für Funkamateure vorstellbar oder es könnte ein Sprachmailbox-Netzwerk aufgebaut werden, in welchem dann die Nachrichten - vergleichbar dem Packet-Radio-Mailboxnetz mit seinem Store and Forward - verteilt werden könnten.

An DB0VOX war auch das Aufsprechen von Verkehrsmeldungen und die Nutzung von öffentlichen Nachrichtenbrettern möglich. Dies ist bislang bei keinem der Systeme vorgesehen.

Benachrichtigungen über neue Nachrichten wurden über Packet-Radio, E-Mail, SMS, Funkruf, APRS und ICQ realisiert. Auch hier könnten neben E-Mail, das bereits verfügbar ist, noch weitere Dienste realisiert werden.

Schnittstellen Neben der TCP/IP-Schnittstelle wäre eine Packet-Radio-Anbindung eine weitere Möglichkeit das System anzusprechen. Bei Packet-Radio handelt es sich um ein abgeschlossenes Netzwerk für Funkamateure, weshalb im Gegensatz zum Internet eine Autorisierung entfallen kann. Der Zugang kann zur individuellen Nutzereinstellung oder auch vom Betreiber zur Wartung genutzt werden.

Der Zugang via Internet könnte über ein Webinterface oder Konsoleninterface (Telnet/SSH) realisiert werden. Zusätzlich wäre hier ein Autorisierungsverfahren zu implementieren.

Vernetzung Neben dem EchoLink-, IRLP- und Allstarlinknetzwerk könnte auch eine Schnittstelle zu den digitalen Verfahren wie D-Star, APCO25, MotoTRBO oder NXDN hergestellt werden.

Für Asterisk ist dies in einem ersten Ansatz mit dem Channel-Treiber chan_dstar für diese Betriebsart bereits realisiert, app_rpt macht hiervon jedoch noch keinen echten Gebrauch. Grundsätzlich eignet sich Asterisk aufgrund seiner modularen Struktur, die bereits auf verschiedenste Channel-Treiber z.B. für analoge, ISDN-Karten oder auch das für app_rpt benötigte Interface setzt, hierfür bestens geeignet. SvxLink kann aufgrund seiner Eigenständigkeit bislang mit keinem der vorgenannten Netze kommunizieren, hier erscheint eine Vernetzung mit Asterisk z.B. über das IAX-Protokoll als eine optimale Symbiose.

Tools Die klassische Papageifunktion zur Kontrolle der eigenen Aussendung ist in den meisten Systemen bereits implementiert. Meist ist auch eine Funktion eingebaut, die eingegebene DTMF-Zeichen dekodiert und in Sprache ansagt. Die Aussendung von Telemetriedaten (z.B. Temperatur oder Batteriespannung) ist mit Asterisk/app_rpt möglich. Für SvxLink könnten solche Funktionen ebenfalls leicht realisiert werden.

Sprachzugänge Die Telefoneinwahl auf dem Repeater aus dem Telefonnetz muss gegen unautorisierten Gebrauch geschützt werden. Denkbar ist es, dass der Nutzer seine Telefonnummern im System hinterlegt und diese bei einem Anruf vom System geprüft wird (Caller-ID-Auswertung). Akzeptierte Rufnummern für ausgehende Anrufe könnten ebenfalls hinterlegt werden.

Telefonanlagen von Heimanwendern (z.B. die Fritzbox) unterstützen bereits Internettelefonie. Der Repeater könnte dabei "Internettelefonie-Provider" spielen, wodurch der Repeater direkt über das SIP-Protokoll erreicht werden könnte. Anders herum können Repeaternutzer direkt den Funkamateur anrufen. Auch hier gilt es sicherzustellen, dass nur Funkamateure ein Gespräch entgegennehmen.

Die Implementierung eines Zugangs über Skype ist prinzipiell ebenfalls möglich. Ein entsprechendes Modul (kostenpflichtig) ist für Asterisk verfügbar. Andere Lösungen sind uns bisher nicht bekannt.

In D-Star können Endnutzer über ihr Rufzeichen ebenfalls direkt adressiert werden. Das Potential des Callsign-Routings könnte hier voll ausgeschöpft werden. Ein DV-Dongle bzw. AMBE-Chip am Repeaterstandort kann für die notwendige Transcodierung sorgen.

Der Audiozugang über das Packet-Radio-Netz könnte mit Flextalk realisiert werden (2400 Baud Voice Codec).

Audiostreaming, Sprachsynthese und Spracherkennung Ein Audiostream eines Repeaters, der über das Internet abgerufen werden kann, ist ein oft gewünschtes Feature. Leider wurde diese Möglichkeit nicht direkt in EchoLink implementiert (Monitor-Modus), so dass sich Endnutzer häufig auf Repeater nur zum Zuhören verbinden und dadurch eine Ansage auf Funk generieren. Zur Realisierung eines Audiostreams wird häufig das Ausgabesignal hardwaretechnisch abgezweigt und in eine zweite Soundkarte wieder eingespeist. Das USB-Interface von DMK Engineering für Asterisk/app_rpt hat zwar einen zweiten Ausgang, der über die usbradio.conf sogar speziell konfiguriert werden kann. Wünschenswert wäre jedoch eine rein softwarebasierte Lösung zum Abgreifen des ausgehenden Audios zur Soundkarte. Mit Hilfe der Advanced Linux Sound Architecture [ALSA] scheint es möglich zu sein, in der Datei /etc/asound.conf ein entsprechendes Capture-Device als virtuelles Gerät zu definieren, welches dann mit einem geeigneten Streaming-Client wie etwa ezstream abgegriffen werden könnte, hierzu haben wir jedoch noch keine ausreichenden Tests durchgeführt. Asterisk/app_rpt unterstützt ALSA bereits seit langem, der aktuelle Entwicklungszweig (Subversion) von SvxLink enthält seit kurzem ebenfalls eine native ALSA-Unterstützung. Noch einfacher wäre jedoch natürlich ein direktes Feature in beiden Programmen, das solche "Hilfslösungen" überflüssig machen würde.

Die Sprachsynthese kann in Form einer Text-to-Speech-Engine (TTS) erfolgen und ermöglicht ein riesiges Spektrum zur Nutzung. Vor einigen Jahren hat Jann zum Beispiel im Urlaub den Audiostream von DB0VOX über das Internet abgehört und über den Telnetzugang mit Hilfe der TTS mit der Heimat kommuniziert. Aber auch abseits dieser Spielereien gibt es handfeste Einsatzszenarien: Durch eine TTS wird es wesentlich einfacher möglich, Dateien für verschiedene Landessprache zu erstellen. Sofern eine TTS, wie bei SvxLink, unterstützt wird, können auch leicht beliebige Textdateien zur Laufzeit z.B. aus dem Internet geladen und vorgelesen werden. Leider ist das Angebot an guten deutschen Sprachausgaben für Linux äußerst eingeschränkt. Als frei

erhältliche deutsche Sprachausgabe bietet sich das bereits etwas in die Jahre gekommene MBROLA [MBROLA] an, welches qualitativ jedoch nicht mit den für Windows oder Mac verfügbaren Sprachen mithalten kann. Eine Alternative könnte hier das kostenpflichtige Voxin [Voxin] sein, das ein in der Windows-Welt bekanntes System portiert. Die Firma AMOOMA [AMOOMA] bietet darüber hinaus ein für Asterisk konzipiertes Online-Tool an, das kurze Texte in deutsche Sprache übersetzt und dabei eine ordentliche Qualität liefert. Auch für SvxLink können die dort erstellten Dateien leicht eingesetzt werden, wenn diese z.B. mit SOX in das passende Format (dabei ist auf korrekte Samplingrate zu achten) konvertiert werden.

Auch Spracherkennung ist ein großes Thema. Die CPU kann beim Spracherkennungsprozess stark belastet werden. Eine kurzzeitige Aktivierung (zum Beispiel über 1750 Hz) könnte die Zeit der CPU-Belastung verkürzen. Eine nette Alternative ist es den zu erkennenden Abschnitt zu digitalisieren, über das Internet an einen Spracherkennungsdienstleister zu schicken und das zurück gelieferte Ergebnis lokal auszuwerten. Die Verbindungssteuerung zu anderen Repeatersystemen wäre damit stark vereinfacht.

Ein besonders interessantes Thema ist die Erkennung von Funkgeräten anhand des individuellen Einschwingverhaltens beim Betätigen der Sendetaste. Evtl. könnten Endnutzer ihre Funkgeräte im System registrieren und personenbezogene Funktionen nutzen. Darüber hinaus könnte dies ein nützliches Instrument sein, um Störer zu identifizieren und auch eine versehentliche Aussendung einem Rufzeichen zuzuordnen, so dass man den Sendenden über seine fehlerhaft arbeitende Station schnell in Kenntnis setzen könnte.

5 Literatur

[ALLSTAR] , 15.03.2010

[ALSA] Advanced Linux Sound Architecture. In: http://alsa.opensrc.org, 01.03.2010

[AMOOMA] AMOOMA GMBH: Voicepromts. In: http://www.amooma.de/voiceprompts, 01.03.2010

[apprpt] Asterisk/app_rpt - The fusion of 2-way radio systems and VOIP. In: http://app-rpt.qrvc.com, 01.03.2010

[Asteriskbuch] Stefan Wintermeyer: Asterisk 1.4 + 1.6: Installation, Programmierung und Betrieb. 1. Aufl.; Addison-Wesley Longman Verlag 2009. In: http://www.das-asterisk-buch.de, 15.03.2010

[chandstar] SCOTT, K4KLF: rptDir. In: http://rtpdir.weebly.com/index.html, 01.03.2010

[Debian] Debian GNU/Linux. In: http://www.debian.org, 01.03.2010

[DMK] DMK Engineering: Offizielle Webseite. In: http://www.dmkeng.com, 15.03.2010

[EchoLink] JONATHAN TAYLOR / SYNERGENICS, LLC.: EchoLink. In: http://www.echolink.org, 01.03.2010

[ELPlus] HERMANN BÖHM / RÜDIGER STENZEL EchoLink+. In: http://www.db0xw.de, 01.03.2010

[ELWiki] EchoLink bei Wikipedia. In: http://de.wikipedia.org/wiki/Echolink, 01.03.2010

[Fedora] FEDORA-PROJEKT: Fedora Core. In: http://fedoraproject.org, 01.03.2010

[Holtrop] FOEKE HOLTROP: Repeater Lelystad. In: http://www.familieholtrop.nl/site/index.php?option=com_content&task=view&id=40&Itemid=83, 01.03.2010

[IAO] INTERESSENGEMEINSCHAFT AMATEURFUNK OSNABRÜCK: Dienste für SysOps. In: http://www.amateurfunk-osnabrueck.de/taxonomy/term/36, 15.03.2010

[ICAO] Erklärung des International Civil Aviation Organization Code (ICAO). In: http://de.wikipedia.org/wiki/International_Civil_Aviation_Organization, 01.03.2010

[IRLP] IRLP - Internet Radio Linking Project. In: http://www.irlp.net, 01.03.2010

[K1LNX] STEPHEN BROWN: Wiki. In: http://www.k1lnx.net/wiki/index.php?title=App_RPT/TIARA, 01.03.2010

[KB9MW] Steve Lampereur, KB9MWR: http://www.qsl.net/kb9mwr/projects/voip/plan.html#asterisk, 15.03.2010

[Mantis] app_rpt Bugtracking. In: http://mantis.qrvc.com, 15.03.2010

[MBROLA] The MBROLA Project. In: http://tcts.fpms.ac.be/synthesis, 01.03.2010

[METAR] Erklärung zu METAR bei Wikipedia. In: http://de.wikipedia.org/wiki/METAR, 01.03.2010

[Monitorverfahren] BERNHARD KLAUCKE: Neuartige Relaiskopplung zwischen DF0HHH und DB0KB. In: CQ DL 6/2002, S. 412

[Namespace] FRIEDRICH W. HESSE (HRSG.): e-teaching.org - Namensraum. In: http://www.e-teaching.org/glossar/namensraum, 01.03.2010

[openSUSE] openSUSE. In: http://de.opensuse.org/Willkommen_auf_openSUSE.org, 01.03.2010

[Relaiskopplung] JANN TRASCHEWSKI: Relaiskopplungsmethoden und Beispiele. Nürnberg 2004. In: http://db0fhn.efi.fh-nuernberg.de/doku.php?id=dg8ngn:publications:relaiskopplung, 01.03.2010

[remoteconf] TOBIAS BLOMBERG: Manpage of remotetrx.conf - Configuration file for the SvxLink remote transceiver server. In: http://svxlink.sourceforge.net/man/man5/remotetrx.conf.5.html, 01.03.2010

[remotetrx] TOBIAS BLOMBERG: Manpage of remotetrx - The SvxLink remote transceiver application. In: http://svxlink.sourceforge.net/man/man1/remotetrx.1.html, 01.03.2010

[RptMailinglist] app_rpt Mailinglist. In: http://qrvc.com/mailman/listinfo/app_rpt-users, 15.03.2010

[RptSvn] app_rpt Subversion. In: http://qrvc.com/viewvc/projects/allstar/?root=svn, 15.03.2010

[S54S] ALEKSANDER STARE: Transceiver to PC/SvxLink Echolink GW interface. In: http://lea.hamradio.si/ s57nan/ham_radio/svx_intf/svx_intf.pdf, 01.03.2010

[Sendmail] THE SENDMAIL CONSORTIUM: Sendmail. In: http://www.sendmail.org, 01.03.2010

[siglevdetcal] TOBIAS BLOMBERG: siglevdetcal. In: http://svxlink.sourceforge.net/man/man1/siglevdetcal.1.html, 01.03.2010

[SUSEHAM] TIM FISCHER, JAN-SIMON MÖLLER: HAM Radio Repository. In: http://software.opensuse.org/download/hamradio, 01.03.2010

[SvxLink] TOBIAS BLOMBERG: SvxLink - Offizielle Webseite. In: http://svxlink.sourceforge.net, 01.03.2010

[SvxMailinglist] SvxLink - Offizielle Mailinglist. In: http://sourceforge.net/mailarchive/forum.php?forum_name=svxlink-devel, 01.03.2010

[SvxMLArchiv] SvxLink - Archiv der Mailinglist. In: http://sourceforge.net/mailarchive/forum.php?forum_name=svxlink-devel, 01.03.2010

[Tcl] Tcl (Tool Command Language). In: http://www.tcl.tk, 01.03.2010

[Ubuntu] Ubuntu Linux. In: http://www.ubuntu.com, 01.03.2010

[USBFOB] Beschreibung von Sound-Interfaces für Asterisk/app_rpt. In: http://images.qrvc.com/usbfob.pdf sowie http://www.qsl.net/kb9mwr/projects/voip/usbfob-119.pdf, 15.03.2010

[Voxin] ORALUX.ORG ASSOCIATION: Voxin. In http://voxin.oralux.net, 01.03.2010

[WB0RXX] TIM MILLER: Svxlink - Ham And Computer Experimenters of Montevideo. In: http://www.hacem.org/svxlink.html, 01.03.2010