DB0OVN Kompendium

Handbuch

AXUDP-Gateways im Hamnet nutzen
Konfiguration unter Linux

20.02.2011 Christoph, DK2CRN

Inhaltsverzeichnis

1. Einleitung

Um über das Hamnet ins klassische PR-Netz einsteigen zu können, sind bei einigen Hamnet-Servern sogenannte AXUDP-Gateways eingerichtet worden. Über diese Gateways können ganz normale AX.25-Verbindungen zum PR-Netz aufgebaut werden. Diese Anleitung soll das Einrichten einer solchen AXUDP-Verbindung unter Linux zeigen.

2. Vorraussetzungen

Grundsätzlich muss natürlich bereits eine Verbindung zum Hamnet bestehen.
Für den Funkamateur gibt drei Möglichkeiten, das HAMNET zu erreichen:

- Man muss in Reichweite eines 13/6cm-WLAN-Einstieges sein (zur Zeit in Neuss nicht verfügbar). Zusätzlich braucht man spezielle WLAN-Hardware, die Bandbreiten von 5 MHz und Frequenzen im Amateurfunkband unterstützt. Beispiel für solch eine passende WLAN-Hardware ist die NanoStation von Ubiquiti (http://www.ubnt.com/nanostation). Eine Anleitung dazu wird folgen.
- Über einen sogenannten PPTP-Tunnel via Internet. Man registriert sich beim Sysop von DB0OVN (DK2CRN) und loggt sich über einen PPTP-Zugang bei DB0OVN ins HAMNET ein. Hierfür existiert die separate Anleitung „*pptp_zugang_db0ovn.pdf*“.
- Man loggt sich über den High-Speed Digi DB0DSP ein. Der Digi DB0DSP bietet einen 70cm-Usereinstieg an, der mit einer Baudrate von 102k4 Baud arbeitet. Man benötigt einen speziellen 70cm Daten-Transceiver (Bausatz DL2ZBN/DL8AAU) sowie ein schnelles PR-Modem (TNC7multi, Baycom-USB-Modem, TNC4e). Unter Windows wird das Flexnet32-Softwarepaket mit IPOVER-Netzwerkschnittstelle gebraucht. Unter Linux werden die AX.25-Libs, -Tools und -Ultilities dazu benötigt. Siehe dazu auch die Anleitung „*pr_hamnet_zugang_db0dsp.pdf*“.

Um unter Linux Packet-Radio machen zu können, muss der Linux-Kernel das AX.25-Protokoll unterstützen und den passenden BPQ-Treiber bereitstellen. In den meisten Distributionen ist das der Fall. Man kann diese Unterstützung mit dem modprobe-Befehl testen.
Wir öffnen eine Textshell und geben folgende Befehle nacheinander ein (wir brauchen dabei das root-Passwort):

```
dk2crn@dk2crn:~> sudo /sbin/modprobe ax25
dk2crn@dk2crn:~> sudo /sbin/modprobe bpqether
```

Wurden alle Befehle ohne Fehlermeldungen ausgeführt (mögliche Fehlermeldung: `FATAL: Module ax25 not found.`), unterstützt der Kernel AX.25 und kennt den Treiber für BPQ-Netzwerke. Damit ist die erste wichtige Voraussetzung für PR erfüllt.

Zusätzlich benötigt man folgende Pakete:

Paket	Version	Erklärung
libax25	0.0.12	Systemdateien (Libraries), auf die andere AX.25-Programme zugreifen
ax25-tools	0.0.10	Hilfsprogramme, die mit dem AX.25-Kernel kommunizieren
ax25-apps	0.0.8	Weitere nützliche AX.25-Programme
linkt	0.8.rc3	Ein PR-Terminalprogramm für den KDE-Desktop

Diese Pakete werden bei einigen Distributionen fertig mitgeliefert. Diese können mit den Installationsprogrammen der Distributionen einfach installiert werden.

Aktuelle Bin-Pakete für Opensuse-Distributionen gibt's bei <http://download.opensuse.org/repositories/hamradio/>

Etwas ältere, aber dennoch brauchbare Bin-Pakete für Debian und Ubuntu gibt's bei <http://dg7gt.osth.de/Debian/>

Aktuelle Sourcen findet man bei <http://www.linux-ax25.org/pub/>.
Die Sourcen für das Terminalprogramm „LinKT" findet man bei DB0OVN unter <ftp://44.225.48.67/pub/packet/linux/linkt/linkt-0.8rc3_patch.tar.bz2>
Hinweis: Source-Pakete müssen vor der Installation kompiliert werden. Hierfür brauchen sie weitere Entwicklungspakete und einen Compiler. Wenn man keinerlei Erfahrungen mit dem Kompilieren von Programmen hat, sollte man möglichst die Bin-Pakete der Distributionen verwenden.

3. Kompilieren und Installieren von Source-Paketen

Wenn die distributionsspezifischen Bin-Pakete installiert werden, kann dieses Kapitel übersprungen werden.

Dieses Kapitel zeigt das Kompilieren und Installieren von distributionsunabhängigen Source-Paketen.
Für das Kompilieren wird zunächst eine Entwicklungsumgebung gebraucht. Folgende Pakete müssen dazu mindestens installiert sein:

Paket	Erklärung
gcc	C-Compiler
gcc33	C-Compiler Version 3.3 (für LinKT-Sourcen)
gcc-c++	C++-Compiler
gcc-c++33	C++-Compiler Version 3.3 (für LinKT-Sourcen)
libtool	Tool zum Erstellen von gemeinsam benutzten Systemdateien
automake	Tool erstellt Makefiles.in-Scripte aus Makefiles.am-Dateien
autoconf	Tool zum Konfigurieren von Sourcen und Erstellen von makefiles

Paket	Erklärung
cvs	Tool zum Herunterladen von aktuellen Sourcen
glib2-devel	Sammlung von C-Bibliotheken
glibc-devel	Standard C-Bibliotheken
fltk-devel	C++ GUI-Toolkit für X-Window
qt3-devel	QT3-Bibliothek (für LinKT-Sourcen)
readline-devel	Bibliothek wird von einigen Programmen gebraucht
kde3libs-devel	KDE3-Bibliothek (für LinKT-Sourcen)

Zunächst werden folgende Sourcepakete heruntergeladen:
<http://www.linux-ax25.org/pub/ax25-apps/ax25-apps-0.0.8-rc2.tar.gz>
<http://www.linux-ax25.org/pub/ax25-tools/ax25-tools-0.0.10-rc2.tar.gz>
<http://www.linux-ax25.org/pub/libax25/libax25-0.0.12-rc2.tar.gz>

Bei LinKT sind die Originalsourcen nicht mehr so ohne weiteres unter den aktuellen Distributionen kompilierbar, daher habe ich bei DB0OVN ein kompilierbares Source-Paket hinterlegt:
<ftp://44.225.48.67/pub/packet/linux/linkt/linkt-0.8rc3_patch.tar.bz2>

Man öffnet eine Textshell und geht in das Verzeichnis, wo die geraden heruntergeladen Source-Pakete abgespeichert wurden.

3.1 libax25

Beginnen wir mit dem Paket „libax25“, es muss zunächst ausgepackt werden:

```
~> tar xfz libax25-0.0.12-rc2.tar.gz
~> cd libax25-0.0.12-rc2
```

Nur erfolgt das Konfigurieren und Kompilieren des Source-Paketes:

```
~/libax25-0.0.12-rc2> autoreconf --install --force
~/libax25-0.0.12-rc2> ./configure --exec_prefix=/usr --sysconfdir=/etc
--localstatedir=/var
~/libax25-0.0.12-rc2> make
```

Wenn alles ohne Error-Meldung kompiliert wurde, kann die Software installiert werden. Dazu braucht man allerdings root-Rechte, dies erlangt man mit dem „sudo“-Befehl. Das Passwort sollte allerdings bekannt sein.

```
~/libax25-0.0.12-rc2> sudo make install
```

Damit ist die Installation des libax25-Pakets abgeschlossen.

3.2 ax25-tools

Das Kompilieren und Installieren ist ähnlich dem des libax25-Pakets, daher hier nur eine Kurzfassung:

```
~> tar xfz ax25-tools-0.0.10-rc2.tar.gz
~> cd ax25-tools-0.0.10-rc2
~/ax25-tools-0.0.10-rc2> autoreconf --install --force
~/ax25-tools-0.0.10-rc2> ./configure --prefix=/usr --sysconfdir=/etc
--localstatedir=/var
~/ax25-tools-0.0.10-rc2> make
~/ax25-tools-0.0.10-rc2> sudo make install
~/ax25-tools-0.0.10-rc2> sudo mkdir /var/ax25
~/ax25-tools-0.0.10-rc2> sudo mkdir /var/ax25/ax25rtd
```

3.3 ax25-apps

Das Kompilieren und Installieren ist ähnlich dem des libax25-Pakets, daher hier nur eine Kurzfassung:

```
~> tar xfz ax25-apps-0.0.8-rc2.tar.gz
~> cd ax25-apps-0.0.8-rc2
~/ax25-apps-0.0.8-rc2> autoreconf --install --force
~/ax25-apps-0.0.8-rc2> ./configure --prefix=/usr --sysconfdir=/etc
--localstatedir=/var
~/ax25-apps-0.0.8-rc2> make
~/ax25-apps-0.0.8-rc2> sudo make install
```

3.4 LinKT

Das PR-Programm LinKT ist schon etwas älter und braucht daher die alten KDE3- und QT3-Bibliotheken. Bei manchen Distributionen kann das zu Problemen führen. Auch gibt es Probleme beim Kompilieren unter 64bit-Systemen.
Versuche daher nach Möglichkeit passende Bin-Pakete für LinKT zu installieren. Dort sind die Probleme durch entsprechende Patches bereits ausgebügelt worden.

Das Paket „linkt“ muss zunächst ausgepackt werden:

```
~> tar xfj linkt-0.8rc3_patch.tar.bz2
~> cd linkt-0.8rc3_patch
```

Wir müssen durch das Setzen von Umgebungsvariablen dafür sorgen, dass beim Kompilieren der Compiler Version 3.3 verwendet wird. Außerdem sagen wir dem Source-Paket, wo es die Libaries findet (Verzeichnisnamen eventuell auf das eigene System anpassen):

```
~/linkt-0.8rc3_patch> export CC=gcc-3.3
~/linkt-0.8rc3_patch> export CXX=g++-3.3
~/linkt-0.8rc3_patch> export QTDIR=/usr/lib/qt3
~/linkt-0.8rc3_patch> export KDEDIR=/opt/kde3
```

Nun wird das Paket konfiguriert:

```
~/linkt-0.8rc3_patch> make -f admin/Makefile.common
~/linkt-0.8rc3_patch> ./configure
```

Das Paket wird kompiliert:

```
~/linkt-0.8rc3_patch> make
```

Wenn keine error-Meldungen auftauchen, kann man „linkt“ nun mit root-Rechten installieren:

```
~/linkt-0.8rc3_patch> sudo make install
```

4. Konfiguration

Wenn alle Pakete installiert wurden, kann mit der Konfiguration gestartet werden. Das AX.25-System unter Linux besitzt keine graphischen Konfigurationstools, wie z.B. Flexnet32 unter Windows. Alles wird über Konfigurationsdateien und Startscripte gesteuert. Für die Erstellung der Script- und Konfigurationsdateien kann man einen einfachen Texteditor einsetzen (**fett** hervorgehobene Abschnitte müssen angepasst werden, *kursive* Textteile geben wichtige Hinweise).

4.1 Die Konfigurationsdateien

Für die Einrichtung der AXUDP-Verbindung müssen zwei Konfigurationsdateien angelegt werden:

Konfigurationsdatei	Erklärung
/etc/ax25/axports	Definiert die Ports des AX.25-Kernels
/etc/ax25/ax25ipd.conf	Konfigurationsdatei für das Programm ax25ipd

Beispieldateien müssten bereits vorhanden sein. Hier ein Beispiel, wie die Datei /etc/ax25/axports aussehen könnte:

```
# /etc/ax25/axports
#
# The format of this file is:
#
# name callsign speed paclen window description
#
bpq   dk2crn-13   38400 255   7     AXUDP-Gateway
#
#end of axports
```

- ***bpq*** ist der Portname unter dem später Anwenderprogramme den AX.25-Kernel ansprechen können
- ***dk2crn-13*** ist das ARP-Call oder die „MAC-Adresse“, die für die Netzwerkschnittstelle gebraucht wird. Bitte unbedingt das eigene Call angeben!

- ***38400*** ist die Schnittstellenbaudrate, sie spielt hier aber keine Rolle.
- ***255*** ist die maximale Datengröße eines PR-Frames in Bytes. Der Maximalwert liegt bei 255 Bytes. Den Wert nicht ändern.
- ***7*** gibt an, wie viele PR-Frames ohne Bestätigung der Gegenstation gesendet werden können. Der Maximalwert liegt bei 7 Frames. Den Wert ändern.
- ***AXUDP-Gateway*** ist nur eine Schnittstellenbeschreibung, der Text ist beliebig.

Die Datei ax25ipd.conf ist da schon etwas umfangreicher. Diese Konfigurationsdatei steuert das Programm „ax25ipd“, welches die AXUDP-Verbindungen aufbaut. Das Programm „ax25ipd“ ist Routing-fähig, daher können mehrere AXUDP-Links angegeben werden (in unserem Beispiel sind es die Links nach DB0OVN, DB0IUZ-4, DB0RES und OE2XZR-15).

```
# ax25ipd configuration file for station dk2crn.ampr.org
#
# auf diesem Port wird gehört, der eigene UDP-Port
socket udp 93
#
# Modus, in dem ax25ipd arbeitet (tnc oder digi)
mode tnc
#
# Name des bpqether-Netzwerk-Ports - die Verbindung zum lokalem AX25-Kernel
device axipd
#
# Baudrate, bei bpqether wird sie ignoriert
speed 1
#
# loglevel 0 schaltet das Logging ab
loglevel 0
#
# definiert den AXUDP-Partner
# route <Zielcall> <Ziel-IP-Adresse> [Parameter]
# Folgende Parameter sind möglich:
#         b       - Broadscast-Pakete dürfen über diese Route gesendet werden
#         d       - Diese Route ist die Standardroute
#         udp     - verwende das UDP-Protokoll (Standard: tcp)
#         udp 98  - die Gegenstation hört auf UDP-Port 98 (Standard: 93)
route db0ovn 44.225.48.67 bd udp 8093
route db0iuz-4 44.225.52.20 udp 8093
route db0res 44.225.28.20 udp 8093
route oe2xzr-15 44.143.40.16 udp 98
#
#end of ax25ipd.conf
```

Sämtliche Gateways sind nur über das Hamnet erreichbar, Verbindungen aus dem Internet werden blockiert. Derzeit sind folgende Gateways verfügbar:

Digipeater	UDP-Port	IP-Adresse	Eigener UDP-Port
DB0RES-0	8093	44.225.28.20	egal
DB0IUZ-4	8093	44.225.52.20	egal
DB0OVN-0	8093	44.225.48.67	egal

Digipeater	UDP-Port	IP-Adresse	Eigener UDP-Port
DB0GOS-0*	8093	44.225.29.20	egal
OE2XZR-15	98	44.143.40.16	93

*: noch nicht in Betrieb / im Aufbau

4.2 Das Startscript

Nachdem nun die Konfigurationsdateien geschrieben wurden, fehlt nur noch ein Startscript, welches das AX.25-System aktiviert. Unter Linux werden Startscripte normalerweise beim Hochfahren des Linux-System gestartet.
Man kann diese Scripte natürlich auch im laufenden Betrieb starten, allerdings benötigt man dafür - wie kann es anders sein – root-Rechte ;)

Zunächst müssen wir uns ein Startscript bauen. Wir öffnen einen Texteditor und geben folgendes ein (wichtig: Variablen anpassen !!):

```
#!/bin/sh
#
# AX25-Kernel Anbindung an AXUDP
# 12.10.2010 DK2CRN
#
### BEGIN INIT INFO
# Provides:          ax25
# Required-Start:    $network
# Should-Start: network
# Required-Stop:
# Should-Stop:
# Default-Start:     2 3 5
# Default-Stop:
# Short-Description: ax25 starts AXUDP-Gateway
# Description:       Activate AX.25-Kerneldevice BPQ for AXUDP Gateway
### END INIT INFO
#
# Der Name muss mit der name-Spalte in der /etc/ax25/axports übereinstimmen
AXPORT=bpq
#
# Das ARP-Call muss mit der callsign-Spalte in der /etc/ax25/axports
# übereinstimmen
ARP_CALL=DK2CRN-13
#
# BPQ_PIPE muss mit dem "device" Eintrag in der /etc/ax25/ax25ipd.conf
# übereinstimmen
BPQ_PIPE=axipd

# Die IP-Adresse ist für das AX.25-Netz. Wenn sie nicht gebraucht wird,
# brauchen hier keine Änderungen gemacht zu werden.
IP_ADRESS=192.168.2.205
NETROUTE=192.168.2.0
NETMASK=255.255.255.0
#
#
PARDIR=/proc/sys/net
#
```

```
#
case "$1" in
    start)
       echo "Initiating AX.25-Kernel"
       #
       # Lädt die Kernel-Treiber
       modprobe ax25
       modprobe bpqether
       #
       echo "AX.25-BPQ-Port : $AXPORT, IP-Adress: $IP_ADRESS"
       #
       # startet den AXUDP-Daemon ax25ipd, der dabei das Loop-Netzwerk-
       # Device "axipd" erstellt
       /usr/sbin/ax25ipd -c /etc/ax25/ax25ipd.conf
       #
       # drei Sekunden warten
       sleep 3
       #
       # ermittelt die BPQ-Schnittstelle, die mit dem Loop-Netzwerk-Device
       # "axipd" kommuniziert. Die Liste /proc/net/bpqether enthält die
       # Netzwerk-Zuordnungen
       if [ -f /proc/net/bpqether ]
       then
           PORT=$(cat /proc/net/bpqether | grep $BPQ_PIPE | head -c4)
       else
           echo "BPQ-Pipe konnte nicht erstellt werden, Abbruch!"
           exit 1
       fi
       #
       # BPQ-Schnittstelle wird konfiguriert
       /sbin/ifconfig $PORT hw ax25 $ARP_CALL $IP_ADRESS mtu 1296 up
       #
       # Konfiguriert die AX.25-Schnittstelle bpq
       # Aktiviert den VC-Modus (für TCP over AX.25-Verbindungen)
       echo 1 > $PARDIR/ax25/$PORT/ip_default_mode        #Mode VC ein
       # Gibt an nach wie vielen ms Inaktivität ein Disconnect erfolgt
       # 0 = kein Timeout, 360000 = 6 min Timeout
       echo 0 > $PARDIR/ax25/$PORT/idle_timeout
       # Wartezeit in ms, die auf ein weiteres Paket gewartet wird,
       # bevor eine Empfangsbestätigung gesendet wird
       echo 300 > $PARDIR/ax25/$PORT/t2_timeout
       # Wartezeit in ms, bis ein unbestätigtes Paket nochmal gesendet wird.
       echo 1000 > $PARDIR/ax25/$PORT/t1_timeout
       # Zeit in ms, bis eine Verbindung überprüft wird (Polling). Dieser
       # Wert sollte nicht verändert werden
       echo 175000 > $PARDIR/ax25/$PORT/t3_timeout
       #
       ;;

    stop)
       echo "Shutdown AX.25-Kernel"

       AX25IPDPID=`ps ax | grep -e "/usr/sbin/ax25ipd -c /etc/ax25/ax25ipd.conf"
| grep -v grep | wc -c`

       if [ $AX25IPDPID -gt 0 ]; then
           PID=`ps ax | grep -e "/usr/sbin/ax25ipd -c /etc/ax25/ax25ipd.conf" |
grep -v grep | awk '{ print $1}'`
           kill $PID
```

```
            echo "ax25ipd $PID killed..."
        else
            echo "no ax25ipd found..."
        fi

        sleep 5
        rmmod bpqether
        #
        ;;

    restart)
        $0 stop
        /bin/sleep 2
        $0 start
        #
        ;;

    *)
        echo "Usage: $0 {start|stop|restart}"
        exit 1
        #
        ;;
esac
exit 0
```

Wir nennen das Script „ax25“.
Nachdem wir das Script geschrieben haben, muss es erstmal ausführbar gemacht werden, dies macht man besten in einer Textshell:

```
~> chmod a+x ax25
```

Nun kann man das Script testen. Für den Start des Scriptes sind root-Rechte notwendig:

```
~> sudo ./ax25 start
Initiating AX.25-Kernel
AX.25-BPQ-Port : bpq, IP-Adress: 192.168.2.205
```

Wenn keine Fehlermeldungen beim Starten des Scriptes angezeigt wurden, müsste man eine funktionierende AX.25-Kernel Schnittstelle testen können. Zunächst überprüfen wir dazu die Netzwerkschnittstellen:

```
~> /sbin/ifconfig | grep -A 3 bpq
bpq1      Link encap:AMPR AX.25  Hardware Adresse DK2CRN-13
          inet Adresse:192.168.2.205  Maske:255.255.255.0
          UP RUNNING  MTU:1296  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0

~> /sbin/ifconfig | grep -A 3 axipd
axipd     Link encap:Ethernet  Hardware Adresse C6:61:BD:E2:1F:FD
          inet6 Adresse: fe80::c461:bdff:fee2:1ffd/64
Gültigkeitsbereich:Verbindung
          UP BROADCAST RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
```

Werden diese beiden Netzwerkports angezeigt, können wir das Gateway testen. **Man**

sollte natürlich darauf achten, dass eine direkte Verbindung zum 44er Hamnet-Netzwerk besteht (siehe 2.Vorraussetzungen).

```
~> call -s dk2crn-7 bpq db0ovn
```

Es müsste sofort eine Verbindung mit DB0OVN aufgebaut werden.
Hinweise: „-s dk2crn-7“ ist das eigene Rufzeichen (mycall), „bpq“ ist der AX.25-Port und „db0ovn“ ist das Zielcall (Destination).
Das Programm „call“ beendet sich automatisch, sobald die Verbindung beendet wird. Man kann die Verbindung durch die Eingabe von „~.“ beenden. Solange noch keine Verbindung aufgebaut wurde, kann das Programm „call“ auch mir den Tasten „Strg + C“ beendet werden.

Um den AX.25-Port wieder zu schließen gibt man folgendes ein:

```
~> sudo ./ax25 stop
```

Wenn nun alles Tests erfolgreich abgeschlossen worden sind, kann das Script richtig installiert werden. Dafür sind natürlich root-Rechte erforderlich:

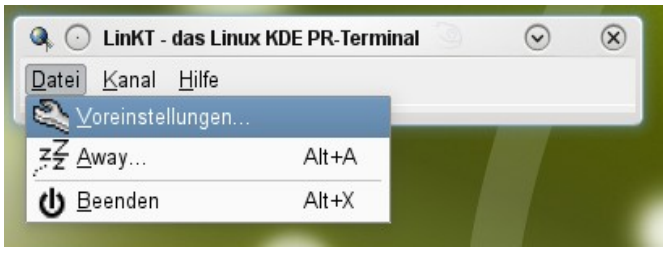
```
~> sudo cp -dp ax25 /etc/init.d/.
~> cd /etc/init.d
```

Man muss nun das Script für den Systemstart aktivieren. Dies lässt sich unter Opensuse mit Yast über das Menü System → Systemdienste (Runlevel) einrichten.
Bei Debian/Ubuntu gibt es dafür das Tool „rcconf“.
Beide Tools, „yast“ und „rcconf“ müssen unter root-Rechten gestartet werden.

5. LinKT

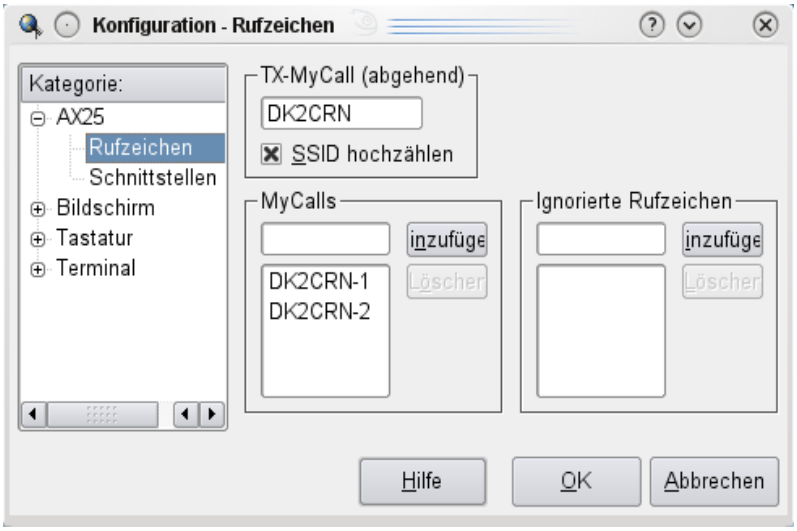
Das Programm „linkt“ kann in einer Textshell auf der Desktopoberfäche gestartet werden. Häufig wird nach der Installation auch ein Programmsymbol angeboten. Man findet es meist im K-Menü unter „Programme → Internet → Hamradio“.

Nachdem LinKT gestartet wurde, müssen wir noch einige Einstellungen vornehmen. Man erreicht die Einstellungen über das Menü „Datei → Voreinstellungen“:

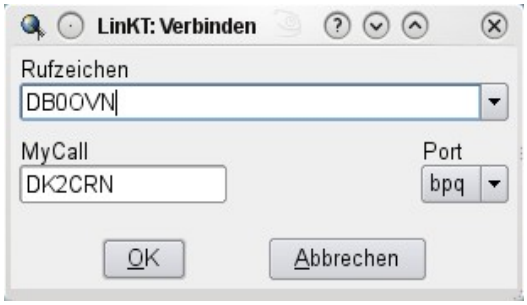


Im Einstellfenster wählen wir in der Kategorie „AX25 → Schnittstellen“ den AX.25-Port aus. In unserem Falle ist das der Port „bpq“.

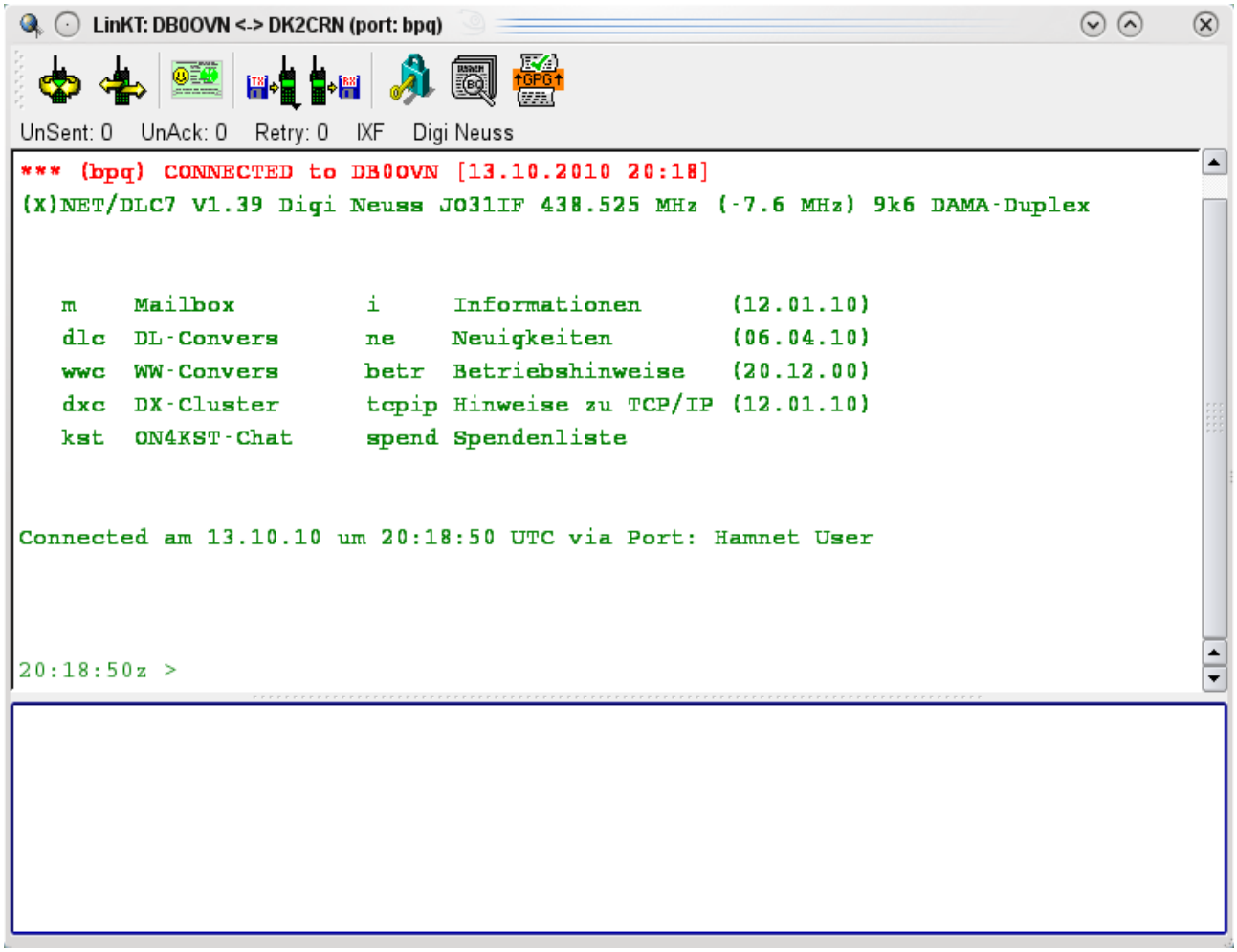
Anschließend definieren wir noch unsere Rufzeichen in der Kategorie „AX25 → Rufzeichen“.



Jetzt sind wir bereit für den ersten Conncetversuch. Über das Hauptmenü „Kanal → Verbinden“ öffnen wir das Verbindungsfenster. Hier wird das Zielrufzeichen und das eigene Rufzeichen angegeben. Der Portname muss auf das AXUDP-Gateway zeigen (hier „bpq“).



Den folgenden Bildschirm sollten sie erfolgreichen Connect sehen:



Das war's.

Viel Spaß mit PR im Hamnet!

73 de Christoph, DK2CRN