Linux Amateur Radio AX.25 HOWTO

Jeff Tranter, VE3ICH

tranter@pobox.com

Linux è forse l'unico sistema operativo che offre un supporto nativo standard per il protocollo AX.25, utilizzato in tutto il mondo dai radioamatori per il packet radio. Questo documento ha lo scopo di spiegare come installare e configurare questo supporto. Traduzione a cura di Nico Alberti (nico@TOGLIMI.langhirano.it) e revisione a cura di Michele Ferritto (m.ferritto@TOGLIMI.virgilio.it).

Sommario

1. Introduzione

Quella radioamatoriale è un'attività non commerciale e senza scopo di lucro che viene praticata da hobbisti in tutto il mondo. I radioamatori sono titolari di una licenza, rilasciata dalle autorità governative, che dà loro diritto di utilizzare porzioni dello spettro radio per attività non commerciali e senza scopo di lucro, come radiocomunicazioni personali, sperimentazione tecnica e fornitura di servizi di pubblica utilità. Il Packet Radio è un particolare modo di trasmissione digitale che utilizza i protocolli di rete per fornire comunicazioni tra computer.

Questo documento era originariamente un'appendice dell'HAM-HOWTO, ma è diventato troppo grande per essere gestito in quel modo. Questo documento descrive come installare e configurare il supporto nativo di AX.25, NET/ROM e Rose per Linux. Vengono descritte alcune tipiche configurazioni che possono essere usate come modelli di partenza.

L'implementazione in Linux dei protocolli di rete per radioamatori è molto flessibile, ma per coloro che non hanno particolare familiarità con questo sistema operativo il processo di configurazione può apparire complesso e laborioso; infatti occorre un po' di tempo per capire tutto l'insieme. Configurare il proprio sistema può apparire un'operazione molto difficile se non ci si è prima documentati sul funzionamento di Linux in generale, del resto non si può pretendere di passare a Linux da un altro sistema operativo senza prima documentarsi su Linux stesso.

1.1. Modifiche rispetto alla versione precedente

- Questo documento ha un nuovo curatore.
- Il documento è stato convertito nel formato DocBook SGML. La maggior parte delle informazioni tabellari sono state convertite in tabelle.
- Il documento è stato rilasciato sotto licenza GNU FDL.
- Sono state aggiunte nuove informazioni sui driver per Baycom, YAM, 6PACK e soundmodem, in user mode.
- È stata aggiunta una sezione sull'APRS
- Sono stati effettuati diversi altri aggiornamenti poiché l'ultima versione di questo documento risaliva al 1997. Probabilmente ci sono ancora molti errori e diverse informazioni non più valide.

1.2. Dove reperire nuove versioni di questo documento

La fonte migliore è da un archivio del Linux Documentation Project. In particolare il Linux Documentation Project gestisce un server Web nel quale questo documento vi appare come the AX25-HOWTO (http://www.linuxdoc.org/HOWTO/AX25-HOWTO.html). Questo documento è inoltre disponibile in vari formati presso il Linux Documentation Project (http://www.linuxdoc.org).

[NdT: la traduzione italiana è disponibile presso il sito dell'Italian Linux Documentation Project, reperibile all'indirizzo http://it.tldp.org (oppure http://ildp.pluto.it)]

È sempre possibile contattarmi, ma dato che passo le nuove versioni del documento direttamente al coordinatore del LDP, è probabile che io non sia in grado di darvi versioni complete più aggiornate di quella presente nell'archivio

1.3. Documenti correlati

C’è parecchia documentazione che tratta del networking in Linux in generale e che raccomando calorosamente di leggere poiché sarà di grande aiuto nello sforzo di capire più a fondo l’argomento.

- Linux Networking HOWTO (http://www.linuxdoc.org/HOWTO/Net-HOWTO/index.html)
- Linux Ethernet HOWTO (http://www.linuxdoc.org/HOWTO/Ethernet-HOWTO.html)
- Linux Firewall and Proxy Server HOWTO (http://www.linuxdoc.org/HOWTO/Firewall-HOWTO.html)
- Linux 2.4 Advanced Routing HOWTO (http://www.linuxdoc.org/HOWTO/Adv-Routing-HOWTO.html)
- NET/ROM-Node mini-Howto (http://www.linuxdoc.org/HOWTO/mini/NET/ROM-Node.html)

È possibile che incontriate riferimenti a un documento chiamato HAM HOWTO. Questo documento è obsoleto ed è stato sostituito dall’Hamsoft Linux Ham Radio Applications and Utilities Database (http://radio.linux.org.au/). Ulteriori informazioni di carattere generale si possono trovare all’interno di altri Linux HOWTO (http://www.linuxdoc.org/HOWTO/HOWTO-INDEX/index.html).

2. Linux e i protocolli per Packet Radio

Il protocollo *AX.25* offre la possibilità di lavorare o meno in modo connesso, ed è usato sia da solo in collegamenti punto-punto, che come trasporto per altri protocolli come il TCP/IP e NET/ROM

La sua struttura è simile all’AX.25 livello 2, con alcune estensioni che lo rendono più adatto all’ambito radioamatoriale.

Il protocollo NET/ROM rappresenta un tentativo di realizzare un protocollo di rete completo e usa AX.25 al livello più basso come protocollo dati. Presenta uno strato di rete che è una forma adattata di AX.25 e offre il routing dinamico e l’alias dei nodi.

Il protocollo Rose fu concepito ed implementato da Tom Moulton W2VY come un’implementazione del livello di pacchetto X.25 ed è stato progettato per operare col protocollo AX.25 come protocollo a livello dati. Esso fornisce anche un layer di rete. L’indirizzamento in Rose è costituito da un numero a 10 cifre. Le prime quattro rappresentano il Codice Identificativo dei Dati di rete (Data Network Identification Code) (DNIC) come indicato dalla Raccomandazione CCIT X.121 Appendice B. Maggiori informazioni sul protocollo Rose si possono trovare sul Web server RATS (http://www.rats.org/).

Alan Cox è stato lo sviluppatore del primo supporto AX.25 nel kernel di Linux, successivamente preso in carico da Jonathon Naylor (mailto:g4klx@g4klx.demon.co.uk) che ha aggiunto anche il supporto per NET/ROM e Rose. Il supporto per il protocollo DAMA è stato sviluppato da Joerg (mailto:jreuter@poboxes.com), DL1BKE, mentre il supporto per il Baycom e il SoundModem è stato aggiunto da Thomas Sailer (mailto:sailer@ife.ee.ethz.ch). Il software specifico per AX.25 è ora sviluppato e mantenuto da un piccolo gruppo di sviluppatori su SourceForge (http://www.sourceforge.net).

Il codice sviluppato per Linux supporta i TNC (Terminal Node Controllers) in modo KISS o basati su 6PACK, l’Ottawa PI card, la Gracilis PacketTwin card e le altre schede SCC basate su Z8530 attraverso il driver generico SCC, diversi modem Baycom seriali e paralleli e i modem YAM con interfaccia seriale. Il driver soundmodem di Thomas Sailer permette l’utilizzo delle SoundBlaster e delle schede sonore basate sul chipset Crystal; inoltre il suo più recente soundmodem, che funziona in user-mode, utilizza i driver standard del kernel, per cui dovrebbe funzionare con ogni scheda audio supportata da Linux.

I programmi utente sono costituiti da un semplice PMS (Personal Message System), un beacon, un programma a linea di comando per effettuare connessioni, `listen` (un esempio su come catturare tutti i frame AX.25 a livello di interfaccia) e programmi per configurare il protocollo NET/ROM. È incluso anche un programma tipo server AX.25 per gestire ed instradare connessioni AX.25 e un demone NET/ROM che svolge la maggior parte del lavoro di supporto per questo protocollo.

Ci sono anche programmi di utilità che supportano l'APRS, fornendo funzionalità quali il digipeating e il gateway verso Internet.

2.1. Come tutto si combina assieme

Quella in Linux è un'implementazione dell AX.25 del tutto nuova. Sebbene assomigli in molti modi a quella di NOS, BPQ o altre implementazioni AX.25, non è uguale a nessuna di queste. L'AX.25 di Linux è in grado di essere configurato in modo tale da poter comportarsi praticamente come le altre implementazioni, ma il processo di configurazione è del tutto diverso.

Per aiutarvi a capire a cosa occorra pensare mentre si effettua la configurazione, questa sezione descrive alcune delle caratteristiche strutturali dell'implementazione AX.25 e come questa si inserisce nel contesto dell'intera struttura di Linux.

Diagramma semplificato dei livelli dei protocolli di rete

```
 _____________________________________________
|         |           |             |         |
| AF_AX25 |AF_NET/ROM |   AF_INET   | AF_ROSE |
|=========|===========|=============|=========|
|         |           |             |         |
|         |           |    TCP/IP   |         |
|         |           |________     |         |
|         |   NET/ROM          |    | ROSE    |
|         |____________________|____|_________|
|            AX.25                            |
|_____________________________________________|
```

Questo schema illustra con chiarezza come NET/ROM, Rose e TCP/IP lavorino sopra l'AX.25, ma anche che, ognuno di questi, sia considerato come un diverso protocollo a livello applicazione. I nomi che iniziano con '_' sono quelli dati alla '*Address Family*' di ognuno dei seguenti protocolli nei programmi che ne fanno uso. La cosa importante che va messa in luce in questo caso, è l'implicita dipendenza dalla configurazione dei dispositivi AX.25, che si riflette in quella dei dispositivi NET/ROM, ROSE o TCP/IP.

Diagramma dei moduli software dell'implementazione di rete in Linux

```
 ____________________________________________________________________________
|         |           |                        ||          |                |
| User    | Programmi |   call        node     ||  Daemons | ax25d  mheardd |
|         |           |   pms         mheard   ||          | inetd  NET/ROMd |
|_________|___________|________________________||__________|________________|
|         | Sockets   |open(), close(), listen(), read(), write(), connect()|
|         |           |_____________________________________________________|
|         |           |   AF_AX25   |  AF_NET/ROM |   AF_ROSE   |  AF_INET  |
|         |___________|_____________|_____________|_____________|___________|
```

```
|Kernel    | Protocolli|    AX.25      |    NET/ROM    |     ROSE      | IP/TCP/UDP|
|          |___________|_______________|_______________|_______________|___________|
|          | Devices   |   ax0,ax1     |   nr0,nr1     | rose0,rose1   | eth0,ppp0 |
|          |___________|_______________|_______________|_______________|___________|
|          | Drivers   | Kiss   PI2   PacketTwin   SCC   BPQ         | slip ppp  |
|          |           |     Soundmodem      Baycom                  | ethernet  |
|__________|___________|_____________________________________________|___________|
|Hardware  | PI2 Card, PacketTwin Card, SCC card, Serial port, Ethernet Card |
|__________|_______________________________________________________________________|
```

Questo diagramma è un po' più generale di quello precedente e vuole mostrare la relazione che intercorre tra le applicazioni utente, il kernel e l'hardware. In particolare si nota il rapporto esistente tra le interfacce di programmazione delle applicazioni (API) a livello di Socket, i moduli effettivamente relativi ai vari protocolli, i dispositivi di rete del kernel e i driver. Ogni cosa in questo diagramma dipende da ciò che è presente sotto di lui, quindi in generale le configurazioni devono essere fatte dal basso verso l'alto. Se, per esempio, si vuole far funzionare il programma *call*, occorre anche configurare l'hardware, poi assicurarsi che il kernel abbia l'opportuno driver, che sia stato creato l'opportuno dispositivo di rete, che il kernel includa il protocollo desiderato e che a sua volta, esponga un interfaccia di programmazione che possa essere utilizzata dal programma *call*. La stesura di questo documento ricalca a grandi linee quest'ordine.

3. I componenti software per AX.25/NET/ROM/Rose

Il software per AX.25 è costituito da tre componenti: il kernel, gli strumenti di configurazione di rete e i programmi di utilità.

Il supporto del protocollo AX.25 è divenuto piuttosto stabile a partire dai kernel della serie 2.2. In questo documento si suppone che si stia usando il kernel più recente che, alla data di stesura, è il 2.4.9

> **Nota:** Le versioni dei software indicate in questo documento sono le più recenti alla data di stesura, ma sono soggette ad aggiornamenti, per cui si prega di controllare la presenza di nuove versioni al momento di scaricarseli.

3.1. Dove reperire il kernel, gli strumenti e i programmi di utilità

3.1.1. I sorgenti del kernel

I sorgenti del kernel si possono trovare su `www.kernel.org` e `ftp.kernel.org`. Il kernel 2.4.9 può essere scaricato da ftp://ftp.kernel.org/pub/linux/kernel/v2.4/linux-2.4.9.tar.gz.

3.1.2. Gli strumenti di rete

L'ultima versione standard degli strumenti di rete di Linux supporta l'AX.25 e NET/ROM e si può trovare su http://www.tazenda.demon.co.uk/phil/net-tools.

La versione più recente del pacchetto ipchains si trova su http://netfilter.filewatcher.org/ipchains (http://netfilter.filewatcher.org/ipchains/).

> **Nota:** Di solito, comunque, non è necessario scaricarsi ed installare questi programmi poiché ogni distribuzione recente dovrebbe includerli.

3.1.3. Le utilità AX.25

Le vecchie ax25-utils, che funzionavano con i kernel della serie 2.0 e 2.1 sono ormai obsolete e sono state sostituite da un nuovo pacchetto che si trova su SourceForge (http://sourceforge.net) all'indirizzo http://sourceforge.net/projects/hams.

Il software è diviso in tre pacchetti: le librerie, gli trumenti e le applicazioni. Alla data di stesura di questo documento le versioni più recenti sono le seguanti:

- ftp://hams.sourceforge.net/pub/hams/ax25/libax25-0.0.7.tar.gz
- ftp://hams.sourceforge.net/pub/hams/ax25/ax25-tools-0.0.6.tar.gz
- ftp://hams.sourceforge.net/pub/hams/ax25/ax25-apps-0.0.4.tar.gz

3.1.4. I programmi di utilità per l'APRS

Se si vuole utilizzare l'APRS, si può scaricare aprsd (http://sourceforge.net/projects/aprsd/) e aprsdigi (http://www.users.cloud9.net/~alan/ham/aprs/):

- http://prdownloads.sourceforge.net/aprsd/aprsd-2.1.4.tar.gz
- http://www.users.cloud9.net/~alan/ham/aprs/aprsdigi-2.0-pre3.tar.gz

4. Installazione del software per AX.25, NET/ROM e Rose

Per installare correttamente il supporto per AX.25 sulla vostra macchina Linux, occorre configurare ed installare un kernel appropriato e poi installare le utilità AX.25

> **Suggerimento:** Invece che compilare ed installare il software direttamente dai sorgenti, può essere preferibile installarsi i pacchetti binari già pronti. Sono infatti presenti pacchetti in formato deb e RPM in diversi siti tra cui http://www.debian.org e http://rpmfind.net nei quali una ricerca con la stringa 'ax25' può restituire il pacchetto richiesto. Incidentalmente la distribuzione Debian è considerata da molti la più "Amateur Radio friendly", e fornisce molte applicazioni già preparate sotto forma di pacchetto Debian (.deb) (uno dei fondatori del progetto, è un radioamatore)

4.1. La compilazione del kernel

Se avete già familiarità col processo di compilazione del Kernel potete saltare questa sezione; state ben attenti comunque, a selezionare le opzioni appropriate, al momento della compilazione. Se non siete ancora pratici continuate a leggere. Può essere consigliabile anche una lettura del Linux Kernel HOWTO (http://www.linuxdoc.org/HOWTO/Kernel-HOWTO.html).

La posizione dove comunemente si decomprimono i sorgenti del kernel è la directory `/usr/src`, nella quale viene creata una sottodirectory chiamata `linux`. Per eseguire questa operazione, occorre essere loggati come `root` ed eseguire una sequenza di comandi simile alla seguente:

```
# cd /usr/src
# mv linux linux.old
# tar xzvf linux-2.4.9.tar.gz
# cd linux
```

Dopo aver decompresso il sorgente del kernel ed applicato l'aggiornamento, occorre eseguire lo script di configurazione e scegliere le opzioni che permettono al kernel di adattarsi al vostro hardware, e le funzionalità che volete che siano implementate nel kernel stesso. Per fare ciò usate il comando:

```
# make menuconfig
```

Se siete sotto X, è possibile avere un'interfaccia grafica usando:

```
# make xconfig
```

È possibile anche usare:

```
# make config
```

Verrà descritto il metodo di configurazione a menu (menuconfig) perché è più comodo e semplice nella scelta delle opzioni, ma potete usare anche l'altro, se vi trovate più a vostro agio.

In entrambi i casi vi verranno proposte una serie di opzioni alle quali dovete rispondere 'Y' (sì) o 'N' (no) (potreste anche voler rispondere 'M' se siete intenzionati ad usare i moduli del kernel, ma per semplicità supponiamo che non lo siate, modificate le scelte in modo appropriato nel caso sceglieste di usare un kernel di tipo modulare).

Le opzioni più importanti per la parte di configurazione relativa all'AX.25 sono:

```
Code maturity level options  --->
    [*] Prompt for development and/or incomplete code/drivers
    ...
General setup  --->
    ...
    [*] Networking support
    ...
Networking options  --->
    <*> UNIX domain sockets
    ...
    [*] TCP/IP networking
```

```
    ...
    [?] IP: tunneling
    ...
Amateur Radio Support --->
    --- Packet Radio protocols
    [*]   Amateur Radio AX.25 Level 2 protocol
    [?]     AX.25 DAMA Slave support
    [?]     Amateur Radio NET/ROM protocol
    [?]     Amateur Radio X.25 PLP (Rose)
    AX.25 network device drivers  --->
    [?] Serial port KISS driver
    [?] Serial port 6PACK driver
    [?] BPQ Ethernet driver
    [?] High-speed (DMA) SCC driver for AX.25
    [?] Z8530 SCC driver
    [?] BAYCOM ser12 fullduplex driver for AX.25
    [?] BAYCOM ser12 halfduplex driver for AX.25
    [?] BAYCOM picpar and par96 driver for AX.25
    [?] BAYCOM epp driver for AX.25
    [?] Soundcard modem driver
    [?]   soundmodem support for Soundblaster and compatible cards
    [?]   soundmodem support for WSS and Crystal cards
    [?]   soundmodem support for 1200 baud AFSK modulation
    [?]   soundmodem support for 2400 baud AFSK modulation (7.3728MHz crystal)
    [?]   soundmodem support for 2400 baud AFSK modulation (8MHz crystal)
    [?]   soundmodem support for 2666 baud AFSK modulation
    [?]   soundmodem support for 4800 baud HAPN-1 modulation
    [?]   soundmodem support for 4800 baud PSK modulation
    [?]   soundmodem support for 9600 baud FSK G3RUH modulation
    <?> YAM driver for AX.25
```

Le opzioni che ho indicato con '`*` sono quelle alle quali si *deve* rispondere 'Y'. Il resto dipende da che hardware avete e quali altre opzioni volete includere. Alcune di queste saranno descritte più avanti in dettaglio, per cui se non sapete ancora che funzionalità implementare, andate avanti nella lettura e ritornate su questo argomento più tardi.

Dopo aver completato la configurazione del kernel dovreste essere in grado di compilarlo senza problemi con i seguenti comandi:

```
# make dep
# make clean
# make zImage
```

Assicuratevi di spostare il file del kernel `arch/i386/boot/zImage` nel posto in cui lo ritenete più opportuno, di editare il vostro file `/etc/lilo.conf` e rieseguire *lilo* per essere sicuri che il nuovo boot di Linux avvenga con nuovo kernel appena compilato

4.1.1. A proposito dei moduli del Kernel

Compilare i driver come moduli del kernel può essere una soluzione utile se si ha intenzione di utilizzare l'AX.25 soltanto occasionalmente e si vuole essere in grado di caricarlo o scaricarlo a richiesta in modo da risparmiare risorse di sistema. Qualcuno, però, ha riscontrato problemi a far funzionare i driver modularizzati in quanto più complicati da configurare. In caso si voglia compilare uno o più driver come modulo, occorre anche dare i seguenti comandi:

```
# make modules
# make modules_install
```

per installare i moduli nella posizione appropriata.

Occorrerà anche aggiungere alcune voci nel file `/etc/modules.conf` in modo da assicurarsi che il programma *kerneld* sappia dove localizzare i moduli del kernel. Occorrerebbe aggiungere/modificare:

```
alias net-pf-3     ax25
alias net-pf-6     NET/ROM
alias net-pf-11    rose
alias tty-ldisc-1  slip
alias tty-ldisc-3  ppp
alias tty-ldisc-5  mkiss
alias bc0          baycom
alias nr0          NET/ROM
alias pi0a         pi2
alias pt0a         pt
alias scc0         optoscc    (o uno degli altri driver scc)
alias sm0          soundmodem
alias tunl0        newtunnel
alias char-major-4 serial
alias char-major-5 serial
alias char-major-6 lp
```

> **Suggerimento:** Nei sistemi Linux basati su Debian queste voci dovrebbero andare nel file `/etc/modutils/aliases` e poi andrebbe lanciato il comando `/sbin/update-modules`.

4.2. Le librerie, i programmi e gli strumenti AX.25

Una volta compilato il nuovo kernel e fatto il reboot con questo, occorre compilare ed installare i programmi, le librerie e gli strumenti per gestire il protocollo AX.25.

Per compilare ed installare libax25 occorre dare una serie di comandi simili a questi:

```
# cd /usr/src
# tar xzvf libax25-0.0.7.tar.gz
# cd libax25-0.0.7
```

```
# ./configure --exec_prefix=/usr --sysconfdir=/etc --localstatedir=/var
# make
# make install
```

> **Suggerimento:** Gli argomenti del comando `configure` fanno sì che i file vengano installati nelle posizioni "standard" sotto la directory `/usr` nelle sottodirectory `bin`, `sbin`, `etc` e `man`. Infatti, dando il comando `configure` senza opzioni, tutti i file verranno messi sotto `/usr/local`. Questo può portare al fatto di avere i file di configurazione sia sotto `/usr` che `/usr/local`. Se si vuole essere sicuri che ciò non succeda si può creare il link simbolico `/usr/local/etc/ax25` che punta a `/etc/ax25` nella primissima fase dell processo d'installazione, in modo da non doversene preoccupare.

Se questa è la prima installazione sulla vostra macchina, si può dare anche il comando:

```
# make installconf
```

per installare anche alcuni file di esempio dentro la directory `/etc/ax25/` dai quali partire per creare la propria configurazione.

A questo punto si può effettuare l'installazione degli strumenti AX.25 in questo modo:

```
# cd /usr/src
# tar xzvf ax25-tools-0.0.6.tar.gz
# cd ax25-tools-0.0.6
# ./configure --prefix=/usr  --sysconfdir=/etc --localstatedir=/var
# make
# make install
# make installconf (se si vogliono installare i file di configurazione)
```

E finalmente installare le applicazioni AX.25:

```
# cd /usr/src
# tar xzvf ax25-apps-0.0.4.tar.gz
# cd ax25-apps-0.0.4
# ./configure --prefix=/usr  --sysconfdir=/etc --localstatedir=/var
# make
# make install
# make installconf (se si vogliono installare i file di configurazione)
```

Se si ricevono messaggi d'errore del tipo:

```
gcc -Wall -Wstrict-prototypes -O2 -I../lib -c call.c
call.c: In function 'statline':
call.c:268: warning: implicit declaration of function 'attron'
call.c:268: 'A_REVERSE' undeclared (first use this function)
```

```
call.c:268: (Each undeclared identifier is reported only once
call.c:268: for each function it appears in.)
```

allora bisogna assicurarsi di avere il pacchetto *ncurses* correttamente installato nel proprio sistema. Lo script di configurazione tenta di localizzare il pacchetto nelle posizioni più comuni, ma alcune distribuzioni la installano in modo errato, e per questa ragione lo script non riesce a trovarlo.

5. Due parole, prima di cominciare, su nominativi, indirizzi e simili

Ogni porta AX.25 e NET/ROM sul vostro sistema deve avere un nominativo/ssid associato ad essa. Questi sono inseriti nei file di configurazione che saranno descritti in dettaglio tra poco.

Alcune implementazioni AX.25 come NOS e BPQ permettono di configurare lo stesso nominativo/ssid sulla stessa porta AX.25 e NET/ROM. Per ragioni tecniche un po' complesse, Linux non lo consente; questo, alla fine, non è un grosso problema, come potrebbe sembrare a prima vista.

Occorre dunque tenere a mente le seguenti cose, mentre si configura il proprio sistema:

1. Ogni porta AX.25 e NET/ROM deve essere configurata con un singolo nominativo/ssid.
2. Il TCP/IP usa il nominativo/ssid della porta usata per ricevere o trasmettere dati, cioè quella configurata per l'interfaccia AX.25 al punto 1.
3. Il NET/ROM usa il nominativo/ssid specificato nel proprio file di configurazione, ma esso viene usato solo quando si parla con un altra stazione NET/ROM; questo *non* è il nominativo/ssid che useranno gli utenti AX.25 che intendono usare il vostro 'nodo' NET/ROM. Ne parleremo più diffusamente tra un po'.
4. Il Rose usa di default il nominativo/ssid delle porte AX.25, eccetto quando il nominativo per Rose sia stato espressamente settato col comando *rsparms*. In questo caso il Rose utilizzerà il nominativo/ssid scelto per tutte le porte
5. Altri programmi, come '*ax25d*' possono usare ogni nominativo/ssid per ascoltare, e ciò può essere duplicato su diverse porte
6. Se si effettua un attenta operazione di routing, si può usare, se si vuole, lo stesso indirizzo IP su tutte le porte

5.1. Cosa sono tutte queste T1, T2, N2 eccetera?

Non tutte le implementazioni AX.25 sono quelle di un TNC2. Linux usa una nomenclatura che si differenzia in alcune parti da quella usata da chi ha fatto packet solamente con un TNC. La tabella che segue dovrebbe essere d'aiuto per interpretare ognuna delle variabili di configurazione, in modo da poterne capire il significato quando se ne parlerà di nuovo più avanti.

Linux	TAPR TNC	Descrizione

Linux	TAPR TNC	Descrizione
T1	FRACK	Tempo di attesa prima di ritrasmettere un frame non confermato (senza acknowledge).
T2	RESPTIME	Tempo minimo di attesa di ricezione di un altro frame prima dell'invio della conferma.
T3	CHECK	Periodo di attesa prima di inviare un segnale che controlli se il collegamento è ancora attivo.
N2	RETRY	Quante volte ritrasmettere un frame prima di considerare interrotta la connessione.
Idle	Periodo in cui la connessione può essere inattiva prima di interromperla	
Window	MAXFRAME	Numero massimo di frame trasmessi senza attendere la conferma di ricezione.

5.2. Parametri configurabili durante l'attività

Il kernel consente di modificare diversi parametri "al volo". Se si controlla con attenzione la directory `/proc/sys/net/` si possono notare diversi file con nomi che descrivono diversi parametri della configurazione della rete. Ognuno dei file nella directory `/proc/sys/net/ax25/` rappresenta una porta AX.25 configurata. Il nome del file si riferisce al nome della porta.

La struttura dei file in `/proc/sys/net/ax25/`*`portname`*`/` è la seguente:

Nome file	Significato	Valore	Default
ip_default_mode	Modalità IP di default	0=DG 1=VC	0
ax25_default_mode	Modalità AX.25 di default	0=Normale 1=Esteso	0
backoff_type	Backoff	0=Lineare 1=Esponenziale	1
connect_mode	Modalità connessa	0=No 1=Yes	1
standard_window_size	Dimensione della finestra standard	1 .. 7	2
extended_window_size	Dimensione della finestra estesa	1 .. 63	32
t1_timeout	T1 Timeout	1s .. 30s	10s
t2_timeout	T2 Timeout	1s .. 20s	3s
t3_timeout	T3 Timeout	0s .. 3600s	300s
idle_timeout	Idle Timeout	0m or greater	20m
maximum_retry_count	N2	1 .. 31	10

Nome file	Significato	Valore	Default
maximum_packet_length	Lunghezza del frame AX.25	1 .. 512	256

Nella tabella T1, T2 e T3 sono dati in secondi, mentre quello di Idle è in minuti. Si noti, però, che i valori usati nell'interfaccia sysctl sono dati in unità interne, dove il tempo in secondi è moltiplicato per 10 in modo da avere una risoluzione di 1/10 di secondo. Ponendo pari a zero i valori che lo permettono (come T3 e Idle) li si disabilita.

La struttura dei file in `/proc/sys/net/NET/ROM/` è la seguente:

Nome File	Significato	Values	Default
default_path_quality			10
link_fails_count			2
network_ttl_initialiser			16
obsolescence_count_initialiser			6
routing_control			1
transport_acknowledge_delay			50
transport_busy_delay			1800
transport_maximum_tries			3
transport_requested_window_size			4
transport_timeout			1200

La struttura dei file in `/proc/sys/net/rose/` è la seguente:

Nome file	Significato	Valori	Default
acknowledge_hold_back_timeout			50
call_request_timeout			2000
clear_request_timeout			1800
link_fail_timeout			1200
maximum_virtual_circuits			50
reset_request_timeout			1800
restart_request_timeout			1800
routing_control			1
window_size			3

Per modificare un parametro, tutto ciò che occorre è scrivere il valore desiderato nel file stesso; ad esempio per controllare e modificare la grandezza della finestra ROSE, si può usare ad esempio:

```
# cat /proc/sys/net/rose/window_size
3
# echo 4 >/proc/sys/net/rose/window_size
```

```
# cat /proc/sys/net/rose/window_size
4
```

6. Configurazione di una porta AX.25

Ogni applicazione che fa uso del protocollo AX.25 legge un file di configurazione per sapere i parametri delle varie porte AX.25 attivate sulla macchina Linux. Il file in questione è `/etc/ax25/axports` e occorre che vi sia una voce per ognuna delle porte attive.

6.1. Creazione del dispositivo di rete AX.25

Il dispositivo di rete è ciò che viene manipolato quando si usa il comando '*ifconfig*'. È l'oggetto attraverso il quale il kernel di Linux spedisce e riceve i dati. Quasi sempre un dispositivo di rete ha una porta fisica ad esso associata, ma vi sono casi in cui ciò non è necessario. Il dispositivo di rete fa riferimento direttamente a un driver di dispositivo.

Nel codice AX.25 di Linux ci sono diversi driver. Il più comune è probabilmente il driver KISS, ma ci sono anche i driver SCC, Baycom e SoundModem.

Ogni driver, quando viene lanciato, crea un dispositivo di rete

6.1.1. Creazione di un dispositivo KISS

Opzioni di compilazione del Kernel:

```
Amateur Radio support  --->
    [*] Amateur Radio support
    --- Packet Radio protocols
    <*>   Amateur Radio AX.25 Level 2 protocol
    ...
    AX.25 network device drivers  --->
    --- AX.25 network device drivers
    <*> Serial port KISS driver
    ...
```

Probabilmente la configurazione più comune è quella con un TNC KISS collegato al pc tramite una porta seriale. Visto che occorre partire col TNC in modo KISS, dopo averlo connesso alla porta seriale, lo si può configurare con programmi di comunicazione come *minicom* o *seyon*.

Per creare un dispositivo KISS si usa il programma *kissattach*. Nella sua forma più semplice si può usare questo programma come segue:

```
# /usr/sbin/kissattach /dev/ttyS0 radio 44.135.96.242
# kissparms -p radio -t 100 -s 100 -r 25
```

Il comando *kissattach* crea un dispositivo di rete di tipo KISS. I dispositivi di questo sono indicati dal nome 'ax[0-9]'. La prima volta che si usa *kissattach* viene creato 'ax0', la seconda 'ax1' e così via. Ogni dispositivo KISS ha associata una porta seriale.

Il comando *kissparms* permette di modificare i parametri di un dispositivo KISS.

In particolare nell'esempio presentato, viene creato un dispositivo di rete KISS usando il dispositivo della porta seriale '/dev/ttyS0' e la voce nel file /etc/ax25/axports con una porta chiamata 'radio' che viene configurata inoltre con un *txdelay* e uno *slottime* di 100 millisecondi, nonché con un valore di *ppersist* pari a 25.

Per maggiori informazioni potete far riferimento alle pagine *man*

6.1.1.1. Configurazione dei TNC Dual Port

L'utilità *mkiss* inclusa nel pacchetto ax25-utils permette di usare entrambi i modem di un TNC dual port. La configurazione è piuttosto semplice e consiste nel prendere un singolo dispositivo seriale connesso ad un singolo TNC multiporta e facendolo apparire come diversi dispositivi connessi ognuno ad un TNC single port. Questa operazione va fatta *prima* di ogni configurazione della parte AX.25. I dispositivi sui quali si effettua la configurazione AX.25 sono interfacce pseudo-TTY, (/dev/ttyq*), e non gli effettivi dispositivi seriali. Questi pseudo-dispositivi TTY creano una specie di tunnel detto pipe attraverso il quale possono parlarsi i programmi che devono colloquiare con i dispositivi TTY. Ogni pipe è composta da una parte master e da una parte slave. La parte master è in genere chiamata '/dev/ptyq*', mentre quella slave è chiamata '/dev/ttyq*'. C'è una relazione uno a uno tra master e slave, quindi, ad esempio, /dev/ptyq0 è la parte master di una pipe che ha /dev/ttyq0 come slave. Occorre aprire per prima la parte master di una pipe. *mkiss* sfrutta questo meccanismo per dividere una singola porta seriale in diversi dispositivi.

Esempio: nel caso di un TNC dual port connesso alla porta seriale /dev/ttyS0 a 9600 bps, i comandi:

```
# /usr/sbin/mkiss -s 9600 /dev/ttyS0 /dev/ptyq0 /dev/ptyq1
# /usr/sbin/kissattach /dev/ttyq0 port1 44.135.96.242
# /usr/sbin/kissattach /dev/ttyq1 port2 44.135.96.242
```

creeranno due pseudo-dispositivi tty ognuno delle quali si comporta come un TNC single port. A questo punto si può usare /dev/ttyq0 e /dev/ttyq1 come se fossero dei normali dispositivi seriali con un TNC connesso. Questo significa che si dovrà usare *kissattach* per ognuna di essi, come descritto nell'esempio sopra per le porte AX.25 chiamate port1 e port2. Non si deve usare *kissattach* sulla effettiva porta seriale vera poiché viene usata dal programma *mkiss*.

Il programma *mkiss* ha diversi argomenti opzionali che possono essere usati e che sono:

-c

: permette l'aggiunta di un byte di checksum per ogni frame KISS; la maggior parte delle implementazioni non supporta questa opzione, che è invece presente in quella di G8BPG.

-s <velocità>

: modifica la velocità della porta.

-h

abilita l'handshaking hardware sulla porta seriale; è disabilitata di default poiché quest'opzione non è; supportata dalla maggior parte delle implementazioni KISS.

-l

abilita il logging delle informazioni nel file log di *syslog*.

6.1.2. Creazione di un dispositivo 6PACK

Opzioni di compilazione del kernel:

```
Amateur Radio support  --->
    [*] Amateur Radio support
    --- Packet Radio protocols
    <*>   Amateur Radio AX.25 Level 2 protocol
    ...
    AX.25 network device drivers  --->
    --- AX.25 network device drivers
    ...
    <*> Serial port 6PACK driver
    ...
```

6PACK è un protocollo supportato da alcuni TNC come alternativa al KISS. Viene utilizzato in modo simile al driver KISS, usando il comando `slattach` al posto di `kissattach`.

Un mini HOWTO sul driver 6PACK è incluso nei sorgenti del kernel nel file `/usr/src/linux/Documentation/networking/6pack.txt`.

6.1.3. Creazione di un dispositivo Baycom

Opzioni di compilazione del Kernel:

```
Amateur Radio support  --->
    [*] Amateur Radio support
    --- Packet Radio protocols
    <*>   Amateur Radio AX.25 Level 2 protocol
    ...
    AX.25 network device drivers  --->
    --- AX.25 network device drivers
    ...
    <?> BAYCOM ser12 fullduplex driver for AX.25
    <?> BAYCOM ser12 halfduplex driver for AX.25
    <?> BAYCOM picpar and par96 driver for AX.25
    <?> BAYCOM epp driver for AX.25
    ...
```

Thomas Sailer (mailto:sailer@ife.ee.ethz.ch), nonostante si ritenesse che non avrebbe funzionato bene, ha sviluppato il supporto Linux per i modem Baycom. Il suo driver supporta il modem seriale `Ser12` e i modem paralleli `Par96` e `PicPar`. Maggiori informazioni sui modem possono essere reperite presso il Sito web dedicato al Baycom (http://www.baycom.de/).

Il primo passo da compiere è quello di determinare gli indirizzi di I/O della porta seriale o parallela alla quale è connesso il Baycom. Una volta fatto, si possono usare queste informazioni per configurare il driver.

Il driver Baycom crea, quando viene configurato, dei dispositivi di rete chiamati: `bc0`, `bc1`, `bc2` ecc.

L'utilità *sethdlc* permette di configurare il driver con questi parametri; nel caso si abbia un solo modem Baycom installato, si possono specificare anche nella linea di comando di *insmod* al caricamento del modulo di controllo del Baycom

Seguono delle semplici dimostrazioni di esempio. Viene disabilitato il driver seriale per COM1 (per evitare conflitti, visto che accede alla stessa porta fisica del Baycom) e configura il driver Ser12 per un Baycom connesso a COM1 con il rilevamento software di portante (DCD) attivato:

```
# setserial /dev/ttyS0 uart none
# insmod hdlcdrv
# insmod baycom mode="ser12*" iobase=0x3f8 irq=4
```

Installa un Baycom Parallelo Par96 su LPT1 usando il rilevamento DCD hardware:

```
# insmod hdlcdrv
# insmod baycom mode="par96" iobase=0x378 irq=7 options=0
```

Questo modo di configurare il driver per Baycom in realtà non è il preferibile, visto che l'utilità *sethdlc* funziona con la stessa facilità con uno o con più dispositivi connessi.

Le pagine *man* di *sethdlc* contengono tutti i dettagli relativi a questo comando, tuttavia si forniscono un paio di esempi per illustrare gli aspetti più importanti di questo tipo di configurazione. Gli esempi presuppongono che sia già stato caricato il modulo per il supporto del Baycom coi comandi:

```
# insmod hdlcdrv
# insmod baycom
```

o che il kernel sia stato compilato col supporto Baycom al suo interno.

Configurazione del driver `bc0` come modem Baycom parallelo su LPT1 con DCD software:

```
# sethdlc -p -i bc0 mode par96 io 0x378 irq 7
```

Configurazione del driver `bc1` come modem Baycom seriale su COM1 :

```
# sethdlc -p -i bc1 mode "ser12*" io 0x3f8 irq 4
```

6.1.4. Configurazione dei parametri di accesso al canale AX.25

I parametri di accesso al canale AX.25 sono equivalenti ai parametri KISS ppersist, txdelay e slottime. La loro modifica si effettua ancora una volta col comando *sethdlc*.

Ancora una volta, le pagine man di *sethdlc* sono la fonte più completa di informazioni, ma un altro paio di esempi non fanno certo male:

Configurazione del dispositivo `bc0` con TxDelay di 200 mS, SlotTime di 100 mS, PPersist di 40 in half duplex:

```
# sethdlc -i bc0 -a txd 200 slot 100 ppersist 40 half
```

Si noti che i valori di temporizzazione sono in millisecondi.

6.1.4.1. Configurazione del Kernel AX.25 per l'uso con un modem Baycom

Il driver Baycom crea dei dispositivi di rete standard che il codice del Kernel AX.25 è in grado di sfruttare. La configurazione è sostanzialmente la stessa di quella per una scheda PI o PacketTwin.

Il primo passo è quello di configurare il dispositivo con un nominativo AX.25. L'utilità *ifconfig* può essere utile allo scopo.

```
# /sbin/ifconfig bc0 hw ax25 VK2KTJ-15 up
```

assegna al dispositivo Baycom `bc0` il nominativo `VK2KTJ-15`. In alternativa è possibile usare il comando *axparms*, ma occorre pure *ifconfig* per attivarlo.

```
# ifconfig bc0 up
# axparms -setcall bc0 vk2ktj-15
```

Il passo successivo è quello di creare una voce nel file `/etc/ax25/axports` come si farebbe per ogni altro dispositivo. La voce nel file `axports` è associata col dispositivo di rete configurato per il nominativo ad esso legato. La voce nel file axports che ha il nominativo con il quale si è configurato il dispositivo BayCom è quella che verrà usata per riferircisi.

A questo punto si userà il nuovo dispositivo AX.25 come ogni altro; lo si potrà configurare per l'uso col TCP/IP, aggiungerlo ad ax25d e veicolarci sopra NET/ROM e Rose a proprio piacimento.

6.1.5. Creazione di un dispositivo SoundModem

Opzioni di compilazione del Kernel:

```
Amateur Radio support  --->
    [*] Amateur Radio support
```

```
--- Packet Radio protocols
<*>   Amateur Radio AX.25 Level 2 protocol
...
AX.25 network device drivers  --->
--- AX.25 network device drivers
...
<*> Soundcard modem driver
[?]   soundmodem support for Soundblaster and compatible cards
[?]   soundmodem support for WSS and Crystal cards
[?]   soundmodem support for 1200 baud AFSK modulation
[?]   soundmodem support for 2400 baud AFSK modulation (7.3728MHz crystal)
[?]   soundmodem support for 2400 baud AFSK modulation (8MHz crystal)
[?]   soundmodem support for 2666 baud AFSK modulation
[?]   soundmodem support for 4800 baud HAPN-1 modulation
[?]   soundmodem support for 4800 baud PSK modulation
[?]   soundmodem support for 9600 baud FSK G3RUH modulation
...
```

Thomas Sailer ha sviluppato un driver per il kernel che permette l'uso come modem della scheda audio la quale, una volta connessa direttamente alla radio, può essere usata per fare packet. L'autore raccomanda di usare per lo meno un 486DX2/66 con questo sistema, poiché tutta la parte di elaborazione del segnale digitale egrave; compiuta dalla CPU del calcolatore.

Attualmente il driver emula i modem 1200 bps AFSK, 4800 HAPN e 9600 FSK (compatibile G3RUH). Le uniche schede audio attualmente supportate sono quelle compatibili SoundBlaster e WindowsSoundSystem. Se si possiede una scheda audio di tipo diverso, si può provare ad usare il soundmodem in modalità utente (user-mode) descritta più avanti in questo documento.

L'uso delle schede audio richiede della circuiteria addizionale in grado di pilotare il Push-To-Talk; le informazioni per realizzare tale dispositivo sono disponibili nell' apposita pagina (http://www.baycom.org/~tom/pcf/ptt_circ/ptt.html) sul sito web di Thomas Salier. Ci sono alcune opzioni possibili: rilevare l'uscita dalla scheda audio, o usare l'uscita da una porta parallela, seriale, o MIDI. Sul sito di Thomas sono presenti esempi di realizzazione di circuiti per ognuno di questi metodi.

Quando viene configurato, il driver SoundModem crea dei dispositivi di rete chiamati: `sm0`, `sm1`, `sm2` eccetera.

> **Nota:** Il driver SoundModem usa le stesse risorse del driver sonoro di Linux, per cui se si vuole usare il modem, occorre accertarsi che il driver sonoro non sia installato. Si possono naturalmente compilare entrambi i driver come moduli, in modo da inserirli e rimuoverli a piacimento.

6.1.5.1. Configurazione della scheda audio

Il driver SoundModem non inizializza la scheda audio. A questo scopo, il pacchetto ax25-utils include un'utilità chiamata '*setcrystal*' che può essere usata per le schede audio basate sul chipset Crystal. Se si possiede una scheda basata su un'altro chipset occorre usare qualche altro programma per inizializzarla. La sintassi dell'utilità è piuttosto semplice:

```
setcrystal [-w wssio] [-s sbio] [-f synthio] [-i irq] [-d dma] [-c dma2]
```

Quindi, per esempio, volendo configurare una SoundBlaster all'indirizzo base di i/o 0x388, irq 10 e dma 1 si userà :

```
# setcrystal -s 0x388 -i 10 -d 1
```

Per configurare una scheda Window Sound System all'indirizzo base di i/o 0x534, irq 5 e dma 3 si userà:

```
# setcrystal -w 0x534 -i 5 -d 3
```

Il parametro `[-f synthio]` serve per impostare l'indirizzo del sintetizzatore, mentre il parametro `[-c dma2]` serve per impostare il secondo canale DMA in modo da avere la possibilità di operare il full duplex.

6.1.5.2. Configurazione del driver Soundmodem

Una volta configurata la scheda audio, occorre configurare il driver, fornendogli informazioni sulla posizione della scheda audio e su che tipo di modem si intende emulare.

L'utilità *sethdlc* permette di configurare il driver con questi parametri. Se si ha solo una scheda audio installata, i parametri si possono specificare anche dal comando *insmod* al momento del caricamento del modulo soundmodem.

Ad esempio, una semplice configurazione con una scheda SoundBlaster configurata come negli esempi precedenti che emula un modem a 1200 bps:

```
# insmod hdlcdrv
# insmod soundmodem mode="sbc:afsk1200" iobase=0x220 irq=5 dma=1
```

Questo non è in realtà il modo più indicato per fare questa operazione. L'utilità *sethdlc* si usa con la stessa facilità con un dispositivo così come con diversi.

Le pagine man di *sethdlc* contengono tutte le informazioni richieste, però un paio di esempi possono illustrare gli aspetti più importanti della configurazione di questa utilità. Negli esempi seguenti si suppone che i moduli Soundmodem siano già stati caricati usando:

```
# insmod hdlcdrv
# insmod soundmodem
```

o che il kernel sia già stato compilato con questo tipo di supporto.

Configurazione del driver per il supporto della scheda Windows SoundSystem configurata come negli esempi precedenti e impostata in modo da emulare un modem G3RUH 9600 compatibile come dispositivo `sm0` usando una porta parallela all'indirizzo 0x378 per pilotare il Push-To-Talk:

```
# sethdlc -p -i sm0 mode wss:fsk9600 io 0x534 irq 5 dma 3 pario 0x378
```

Configurazione del driver per supportare la scheda Soundblaster configurata come negli esempi precedenti per emulare un modem HAPN a 4800 bps come dispositivo `sm1` usando la porta seriale all'indirizzo 0x2f8 per pilotare il Push-To-Talk:

```
# sethdlc -p -i sm1 mode sbc:hapn4800 io 0x388 irq 10 dma 1 serio 0x2f8
```

Configurazione del driver per supportare la scheda SoundBlaster configurata come negli esempi precedenti e impostata in modo da emulare un modem 1200 bps AFSK come dispositivo `sm1`usando la porta seriale all'indirizzo 0x2f8 per pilotare il Push-To-Talk della radio:

```
# sethdlc -p -i sm1 mode sbc:afsk1200 io 0x388 irq 10 dma 1 serio 0x2f8
```

6.1.5.3. Configurazione dei parametri di accesso al canale AX.25

I parametri di accesso al canale AX.25 sono equivalenti ai parametri KISS ppersist, txdelay e slottime. La loro modifica si effettua ancora una volta col comando *sethdlc*.

Ancora una volta le pagine man di *sethdlc* sono la fonte di informazioni più completa, tuttavia un paio di esempi in più non fanno male:

Configurazione del dispositivo `sm0` con TxDelay di 100 mS, SlotTime di 50mS, PPersist di 128 e full duplex:

```
# sethdlc -i sm0 -a txd 100 slot 50 ppersist 128 full
```

Si noti che le temporizzazioni sono espresse in millisecondi.

6.1.5.4. Messa a punto del driver e del livello audio

Per ogni modem radio è molto importante che i livelli audio siano impostati correttamente e il SoundModem non fa eccezione. Thomas Sailer ha per questo scritto alcune utilità che facilitano questo compito, chiamate *smdiag* e *smmixer*.

smdiag

: fornisce due tipi di visualizzazione del segnale in ingresso: come oscilloscopio e con un diagramma ad occhio.

smmixer

: permette di effettuare la modifica del livello audio in ricezione ed in trasmissione.

Per lanciare l'utilità *smdiag* per il dispositivo SoundModem `sm0` in modalità 'diagramma ad occhio' si usa il comando:

```
# smdiag -i sm0 -e
```

Per lanciare l'utilità *smmixer* per il dispositivo SoundModem `sm0` si usa il comando:

```
# smmixer -i sm0
```

6.1.5.5. Configurazione del Kernel AX.25 per l'uso con SoundModem

Il driver Soundmodem crea dei dispositivi di rete standard che il codice del Kernel AX.25 è in grado di sfruttare. La configurazione è sostanzialmente la stessa di quella per una scheda PI o PacketTwin.

Il primo passo è quello di configurare il dispositivo con un nominativo AX.25. l'utilità *ifconfig* può essere utile allo scopo.

```
# /sbin/ifconfig sm0 hw ax25 VK2KTJ-15 up
```

assegna al dispositivo SoundModem `sm0` il nominativo AX.25 `VK2KTJ-15`. In alternativa sarebbe possibile usare il comando *axparms*, ma occorre pure *ifconfig* per attivarlo.

```
# ifconfig sm0 up
# axparms -setcall sm0 vk2ktj-15
```

Il passo successivo è quello di creare una voce nel file `/etc/ax25/axports` come si farebbe per ogni altro dispositivo. La voce nel file `axports` è associata col dispositivo di rete configurato per il nominativo ad esso legato; verrè usata quella col nominativo assegnato al dispositivo SoundModem.

A questo punto si userè il nuovo dispositivo AX.25 come ogni altro; lo si potrà; configurare per l'uso col TCP/IP, aggiungerlo a ax25d e usarci sopra NET/ROM o ROSE a proprio piacimento.

6.1.6. Creazione di un dispositivo Soundmodem in modo utente

Opzioni di compilazione del kernel: non applicabile

Thomas Sailer ha scritto un driver Soundmodem che funziona in modo utente usando i driver audio del kernel, per cui dovrebbe funzionare con ogni scheda audio che Linux supporta.

Il driver è composto dal programma `soundmodem`. Il programma grafico `soundmodemconfig` consente di configurare e testare il driver soundmodem. Così come per il supporto del suono , occorre il driver AX.25 mkiss del kernel.

Il software e la relativa documentazione possono essere scaricati da http://www.baycom.org/~tom/ham/soundmodem (http://www.baycom.org/~tom/ham/soundmodem/).

6.1.7. Creazione di un dispositivo YAM

Opzioni di compilazione del Kernel:

```
Amateur Radio support  --->
    [*] Amateur Radio support
    --- Packet Radio protocols
    <*>   Amateur Radio AX.25 Level 2 protocol
    ...
    AX.25 network device drivers  --->
    --- AX.25 network device drivers
    ...
    <?> YAM driver for AX.25
    ...
```

YAM (Yet Another Modem) è un modem a 9600 baud progettato da Nico Palermo. Le informazioni per il suo driver Linux si possono trovare su http://www.teaser.fr/~frible/yam.html mentre informazioni generali sul modem si possono trovare su http://www.microlet.com/yam/

6.1.8. Creazione di un dispositivo per la scheda PI

Opzioni di compilazione del Kernel:

```
General setup  --->
    [*] Networking support
Network device support  --->
    [*] Network device support
    ...
    [*] Radio network interfaces
    [*] Ottawa PI and PI/2 support for AX.25
```

Il driver per schede PI crea dei dispositivi chiamati '`pi[0-9][ab]`'. La prima scheda PI sarà indicata come '`pi0`', la seconda '`pi1`', eccetera. Le lettere '`a`' e '`b`' si riferiscono rispettivamente alla prima e alla seconda interfaccia fisica delle schede PI. Se si è compilato il Kernel in modo da includere il driver per la scheda e questa è stata riconosciuta dal sistema in modo esatto, si può; usare il seguente comando per configurare il dispositivo di rete:

```
# /sbin/ifconfig pi0a hw ax25 VK2KTJ-15 up
```

Questo comando configura ed attiva la prima porta della prima scheda PI assegnandole il nominativo `VK2KTJ-15`. Per usare il dispositivo, tutto ciò che serve a questo punto è inserire una voce nel file `/etc/ax25/axports` col medesimo nominativo/ssid.

Il driver per la scheda PI è stato scritto da David Perry (mailto:dp@hydra.carleton.edu).

6.1.9. Creazione di un dispositivo PacketTwin

Opzioni di compilazione del Kernel:

```
General setup  --->
    [*] Networking support
Network device support  --->
    [*] Network device support
    ...
    [*] Radio network interfaces
    [*] Gracilis PackeTwin support for AX.25
```

Il driver per la scheda PacketTwin crea i dispositivi '`pt[0-9][ab]`'; la prima scheda PacketTwin localizzata nel calcolatore sarà indicata come '`pt0`', la seconda '`pt1`' e così via, mentre '`a`' e '`b`' fanno riferimento alla prima e alla seconda interfaccia fisica della scheda PacketTwin. Una volta compilato il kernel con incluso il driver per la scheda PacketTwin, se quest'ultima è stata correttamente riconosciuta, si può usare il seguente comando per configurare il dispositivo di rete:

```
# /sbin/ifconfig pt0a hw ax25 VK2KTJ-15 up
```

Questo comando configura e attiva la prima porta della prima scheda PacketTwin col nominativo `VK2KTJ-15`. Per usare il dispositivo, tutto ciò che occorre fare è configurare una voce nel proprio file `/etc/ax25/axports` con il medesimo nominativo/ssid.

Il driver per schede PacketTwin è stato scritto da Craig Small (mailto:csmall@triode.apana.org.au), VK2XLZ.

6.1.10. Creazione di un dispositivo generico SCC

Opzioni di compilazione del Kernel:

```
General setup  --->
    [*] Networking support
Network device support  --->
    [*] Network device support
    ...
    [*] Radio network interfaces
    [*] Z8530 SCC KISS emulation driver for AX.25
```

Joerg Reuter (mailto:jreuter@poboxes.com), DL1BKE, ha sviluppato il supporto generico per le schede basate sullo SCC Z8530 . Il suo driver permette il supporto di diversi tipi di queste schede e offre una modalità di utilizzo simile a quella di un TNC KISS.

6.1.10.1. Reperire e compilare i programmi di configurazione

Sebbene il driver sia incluso nella distribuzione standard del Kernel, Joerg Reuter distribuisce versioni più recenti del suo driver assieme ad una collezione di programmi che ne aiutano la configurazione.

I programmi di configurazione possono essere scaricati da pagina web di Joerg (http://www.qsl.net/dl1bke), ftp://db0bm.automation.fh-aachen.de/incoming/dl1bke, ftp://insl1.etec.uni-karlsruhe.de/pub/hamradio/linux/z8530, ftp://ftp.ucsd.edu/hamradio/packet/tcpip/linux, o ftp://ftp.ucsd.edu/hamradio/packet/tcpip/incoming.

Troverete diverse versioni; va scelta quella che maggiormente si adatta alla versione di Kernel che si intende usare. `z8530drv-2.4a.dl1bke.tar.gz` per i kernel 2.0.* e `z8530drv-utils-3.0.tar.gz` per i kernel 2.1.6 o successivi.

I seguenti comandi sono quelli che ho usato per compilare ed installare il pacchetto per la versione 2.0.30 del kernel:

```
# cd /usr/src
# gzip -dc z8530drv-2.4a.dl1bke.tar.gz | tar xvpofz -
# cd z8530drv
# make clean
# make dep
# make module         # Se volete avere il driver compilato come modulo
# make for_kernel     # Se volete avere il driver incluso staticamente nel kernel
# make install
```

Dopo aver completato quest'operazione dovreste avere tre nuovi programmi installati nella vostra directory `/sbin`: *gencfg*, *sccinit* e *sccstat*. Saranno questi a permettervi di configurare il driver per la scheda.

Ci si troverà anche ad avere un un gruppo di dispositivi speciali nella directory `/dev` chiamati `scc0`-`scc7`; questi saranno usati più avanti e saranno i dispositivi 'KISS' da utilizzare.

Se scegliete l'opzione 'make for_kernel' occorre ricompilare il kernel. Per far sì che venga incluso il supporto per il driver z8530 bisogna rispondere '`Y`' alla domanda '`Z8530 SCC kiss emulation driver for AX.25`' che viene fatta dalla procedura di configurazione del kernel, durante il '`make config`'.

Se invece avete fatto 'make module', occorre che il nuovo file `scc.o` sia messo nella directory `/lib/modules`, ma non occorre ricompilare il kernel. Si ricordi di usare il comando *insmod* per caricare il modulo prima di provare a configurarlo.

6.1.10.2. Configurazione del driver per la propria scheda

Il driver per scheda SCC z8530 è stato concepito per garantire la massima flessibilità e per supportare il maggior numero di schede. Questa flessibilità porta però come conseguenza una configurazione piuttosto impegnativa.

Nel pacchetto c'è una esauriente documentazione che va letta in caso di difficoltà. In particolare, maggiori informazioni si possono trovare in `doc/scc_eng.doc` o `doc/scc_ger.doc`. In questo documento vengono ripresi i concetti fondamentali, ma si invita a consultare i file indicati nel caso si richieda un livello di dettaglio maggiore.

Il file di configurazione principale è chiamato `/etc/z8530drv.conf` e viene letto dal programma *sccinit*, Questo file è diviso in due parti: configurazione dei parametri hardware e configurazione del canale. Una volta adattato opportunamente questo file alle vostre esigenze, basta aggiungere:

```
# sccinit
```

nel file `rc` che configura la rete, e il driver sarè inizializzato nel modo indicato dal file di configurazione. Questa operazione va fatta prima di provare ad usare il driver.

6.1.10.2.1. Configurazione dei parametri hardware

La prima sezione è divisa in due sottoparti, ognuna delle quali rappresenta la configurazione per un chip 8530 ed è costituita da una lista di parole chiave e valori associati. In questo file si possono specificare fino a quattro chip SCC di default. Il valore `#define MAXSCC 4` nel file `scc.c` può essere aumentato se si desidera il supporto per un numero maggiore di chip.

Le parole chiave possibili e i relativi argomenti sono:

chip

: la parola chiave `chip` serve per separare le sottosezioni. Non usa argomenti.

data_a

: questa keyword specifica l'indirizzo della porta dati per il canale 'A' dello z8530. L'argomento è un numero esadecimale, ad esempio 0x300

ctrl_a

: specifica l'indirizzo della porta di controllo per il canale 'A' dello z8530. L'argomento è un numero esadecimale, ad esempio 0x304

data_b

: specifica l'indirizzo della porta dati per il canale 'B' dello z8530. L'argomento è un numero esadecimale, ad esempio 0x301

ctrl_b

: specifica l'indirizzo della porta di controllo per il canale 'B' dello z8530. L'argomento è un numero esadecimale, ad esempio 0x305

irq

: specifica l'IRQ (l'interrupt) usato dalla scheda SCC 8530 descritta in questa sottosezione. L'argomento è un intero, ad esempio 5

pclock

: specifica la frequenza del clock al pin PCLK dell'8530. L'argomento è un intero che indica la frequenza in Hz, ed è fissato di default a 4915200 se si omette questa parola chiave.

board

: specifica il tipo particolare di scheda SCC 8530. L'argomento è una stringa alfanumerica che può assumere i seguenti valori:

PA0HZP

: scheda SCC PA0HZP

EAGLE

: Scheda Eagle

PC100

scheda DRSI PC100 SCC

PRIMUS

scheda PRIMUS-PC (DG9BL)

BAYCOM

scheda BayCom (U)SCC

escc

questa parola chiave è opzionale; viene usata per abilitare il supporto per i chip Extended SCC (ESCC) come l'8580, 85180 o 85280. L'argomento è una stringa di caratteri i cui valori ammessi sono 'yes' oppure 'no', che è il valore didefault.

vector

questa parola chiave è opzionale; specifica l'indirizzo del 'vector latch' (conosciuto pure come "intack port") per le schede PA0HZP. Ci può essere solo un vector latch per tutti i chip e il valore di default è 0

special

questa parola chiave è opzionale; specifica l'indirizzo del registro per funzioni speciali presente su diverse schede SCC. Il valore di default è 0

option

questa parola chiave è opzionale e il suo valore di default è 0

Seguono alcuni configurazioni d'esempio per le schede più comuni:

BayCom USCC

```
chip    1
data_a  0x300
ctrl_a  0x304
data_b  0x301
ctrl_b  0x305
irq     5
board   BAYCOM
#
# SCC chip 2
#
chip    2
data_a  0x302
ctrl_a  0x306
data_b  0x303
ctrl_b  0x307
board   BAYCOM
```

scheda SCC PA0HZP

```
chip 1
data_a 0x153
data_b 0x151
ctrl_a 0x152
ctrl_b 0x150
irq 9
pclock 4915200
board PA0HZP
vector 0x168
escc no
#
#
#
chip 2
data_a 0x157
data_b 0x155
ctrl_a 0x156
ctrl_b 0x154
irq 9
pclock 4915200
board PA0HZP
vector 0x168
escc no
```

Scheda SCC DRSI

```
chip 1
data_a 0x303
data_b 0x301
ctrl_a 0x302
ctrl_b 0x300
irq 7
pclock 4915200
board DRSI
escc no
```

Se con NOS si ha già una configurazione funzionante per la vostra scheda, si può usare il comando *gencfg* per convertire i comandi del driver NOS di PE1CHL in una formato adatto al file di configurazione del driver z8530.

Per usare *gencfg* basta invocare il comando con gli stessi parametri usati per il driver di PE1CHL in NET/NOS. Per esempio il comando:

```
# gencfg 2 0x150 4 2 0 1 0x168 9 4915200
```

genererà un abbozzo di configurazione per la scheda OptoSCC.

6.1.10.3. Configurazione del canale

La sezione di configurazione del canale è quella nella quale si specificano tutti gli altri parametri associati con la porta che si sta configurando. Anche questa è divisa in due sottosezioni che rappresentano ciascuna una porta logica, quindi, visto che ogni scheda SCC 8530 ne supporta due, saranno presenti due sottosezioni per ogni sottosezione nella parte dei parametri hardware.

Queste parole chiave sono scritte anche nel file `/etc/z8530drv.conf` e devono apparire *dopo* la sezione dei parametri hardware.

L'ordine in cui queste appaiono è molto importante, ma quello in cui sono presentate qui dovrebbe andare bene. Le parole chiave e i relativi argomenti sono:

device

questa parola chiave deve apparire all'inizio della definizione della porta e specifica il nome del dispositivo file speciale al quale si applica il resto dei parametri di configurazione; ad esempio `/dev/scc0`

speed

questa parola chiave specifica la velocità dell'interfaccia in bit al secondo: ad esempio `1200`

clock

questa parola chiave specifica da dove recuperare il clock per i dati. I valori consentiti sono:

dpll

normali operazioni in half duplex

external

il modem fornisce il proprio clock per le operazioni di RX/TX

divider

usa il divider full duplex se installato.

mode

>questa parola chiave specifica il tipo di codifica da adottare per la rappresentazione dei dati. I valori consentiti sono: `nrzi` oppure `nrz`

rxbuffers

specifica il numero di buffer di ricezione per cui allocare memoria. L'argomento è un intero, ad esempio 8.

txbuffers

specifica il numero di buffer di trasmissione per cui allocare memoria. L'argomento è un intero, ad esempio 8.

bufsize

specifica la dimensione dei buffer di ricezione e trasmissione. L'argomento è in byte e rappresenta la lunghezza totale del frame, compreso l'header AX.25 (non solo la parte dati). Questa parola chiave è opzionale e il suo valore di default è `384` txdelay

rappresenta il parametro KISS di ritardo nella trasmissione; l'argomento è un intero ed esprime una grandezza in millisecondi.

persist

rappresenta il parametro KISS di persist e il suo argomento è un intero.

slot

rappresenta il parametro KISS di tempo di slot (slot time); l'argomento è un intero ed esprime una grandezza in millisecondi.

tail

rappresenta il parametro KISS di tail; il suo argomento è un intero ed esprime una grandezza in millisecondi.

fulldup

rappresenta il flag KISS che indica se la trasmissione è in full duplex; l'argomento è un intero e i valori ammessi sono: `1`==Full Duplex, `0`==Half Duplex.

wait

rappresenta il parametro KISS di tempo di wait; l'argomento è un intero ed esprime una grandezza in millisecondi.

min

rappresenta il parametro KISS 'min'; l'argomento è un intero ed esprime una grandezza in secondi.

maxkey

rappresenta il parametro KISS indicante il tempo massimo di keyup; l'argomento è un intero ed esprime una grandezza in secondi.

idle

rappresenta il parametro KISS di tempo di idle; l'argomento è un intero ed esprime una grandezza in secondi.

maxdef

rappresenta il parametro KISS 'maxdef'; l'argomento è un intero.

group

rappresenta il parametro KISS di valore di gruppo; l'argomento è un intero.

txoff

rappresenta il parametro KISS di tempo di 'txoff'; l'argomento è un intero ed esprime una grandezza in millisecondi.

softdcd

rappresenta il parametro KISS 'softdcd'; l'argomento è un intero.

slip

rappresenta il flag KISS 'slip'; l'argomento è un intero.

6.1.10.4. Uso del driver

Per usare il driver basta semplicemente utilizzare i dispositivi `/dev/scc*` come se si avesse un dispositivo seriale tty con un TNC KISS connesso ad esso. Per esempio, per configurare la parte di networking del kernel di Linux per l'uso di una scheda SCC, si può fare in questo modo:

```
# kissattach -s 4800 /dev/scc0 VK2KTJ
```

Si può anche usare NOS per effettuare la configurazione nello stesso modo. Da JNOS, per esempio, si può fare:

```
attach asy scc0 0 ax25 scc0 256 256 4800
```

6.1.10.5. I programmi sccstat e sccparam

Per la diagnosi di eventuali problemi, si può usare il programma *sccstat* per mostrare la configurazione corrente di un dispositivo SCC. Per usarlo si può dare il comando:

```
# sccstat /dev/scc0
```

in questo modo verranno mostrati un sacco di informazioni sulla configurazione e sul comportamento della porta SCC `/dev/scc0`.

Il comando *sccparam* permette di cambiare o modificare la configurazione dopo il boot del sistema. La sua sintassi è molto simile al comando NOS `param`, quindi, ad esempio, per settare a 100 ms il valore di `txtail` si puo fare:

```
# sccparam /dev/scc0 txtail 0x8
```

6.1.11. Creazione di un dispositivo ethernet BPQ

Opzioni di compilazione del Kernel:

```
General setup  --->
    [*] Networking support
Network device support  --->
    [*] Network device support
    ...
    [*] Radio network interfaces
```

```
    [*] BPQ Ethernet driver for AX.25
```

Linux supporta la compatibilità Ethernet BPQ. Questo permette di trasportare il protocollo AX.25 su una LAN Ethernet e di connettere sulla rete locale la macchina Linux con altre macchine BPQ.

I dispositivi di rete BPQ hanno nome ’`bpq[0-9]`’. Il dispositivo ’`bpq0`’ è associato con ’`eth0`’, ’`bpq1`’ con ’`eth1`’ e così via.

La configurazione è abbastanza immediata. Prima di tutto occorre che il kernel sia stato compilato per supportare una scheda Ethernet standard e che si sia già verificato il corretto funzionamento di quest’ultima con Linux. Per maggiori informazioni su queste operazioni si può fare riferimento all’Ethernet-HOWTO (Ethernet-HOWTO.html)

Per configurare il supporto BPQ occorre dotare di un nominativo AX.25 il dispositivo Ethernet col seguente comando:

```
# /sbin/ifconfig bpq0 hw ax25 vk2ktj-14 up
```

Ancora una volta si ricordi che il nominativo specificato deve essere uguale a quello presente nel file `/etc/ax25/axports` che si intende usare per questa porta.

6.1.12. Configurazione del nodo BPQ per il colloquio con il supporto AX.25 di Linux

A differenza dell’implementazione standard di BPQ Ethernet che usa normalmente un indirizzamento multicast, in Linux si adotta il normale indirizzamento broadcast; il file NET.CFG per il driver BPQ ODI dovrebbe perciò essere modificato per assomigliare a questo:

```
LINK SUPPORT

        MAX STACKS 1
        MAX BOARDS 1

LINK DRIVER E2000                    ; o altre MLID che si adattino
     ; alla vostra scheda

        INT 10                       ;
        PORT 300                     ; per adattarsi alla vostra scheda

        FRAME ETHERNET_II

        PROTOCOL BPQ 8FF ETHERNET_II ; richiesto per BPQ - può cambiare PID

BPQPARAMS                            ; opzionale - richiesto solo se
                                     ; non si vuole usare l’indirizzo
                                     ; di default

        ETH_ADDR  FF:FF:FF:FF:FF:FF  ; indirizzo di default della scheda
```

6.2. Creazione del file `/etc/ax25/axports`

`/etc/ax25/axports` è un semplice file di testo, da creare con un editor. Il suo formato è il seguente:

```
portname  callsign  baudrate  paclen  window  description
```

dove:

portname
: è il nome che viene dato alla porta.

callsign
: è il nominativo AX.25 che si vuole assegnare alla porta.

baudrate
: è la velocità con la quale si vuol far comunicare la porta col proprio TNC.

paclen
: è la grandezza massima del pacchetto che si vuole che la porta usi per le trasmissioni AX.25.

window
: è il parametro di finestra AX.25 (K), cioè il valore di `MAXFRAME` di molti TNC.

description
: è la descrizione testuale della porta.

La mia è la seguente:

```
radio    VK2KTJ-15       4800        256     2       4800bps 144.800 MHz
ether    VK2KTJ-14       10000000    256     2       BPQ/ethernet device
```

Si ricordi di usare un nominativo/ssid univoco per ogni porta AX.25 che si crea. Occorre creare, dunque, una voce per ogni dispositivo AX.25 che si vuole utilizzare (KISS, SCC, PI, PT, Baycom o SoundModem che sia). Ogni voce descrive un singolo dispositivo di rete AX.25 che sono associati alle porte attraverso il nominativo/ssid, per questo esso deve essere univoco.

6.3. Configurazione del routing AX.25

È possibile configurare l'elenco dei digipeater da utilizzare per raggiungere un host specifico, operazione che risulta utile sia per le operazioni AX.25 che basate su IP. Il comando *axparms* permette di effettuare questa operazione; le pagine *man* sono sempre la fonte migliore di informazioni su questo comando, ma un semplice esempio può essere:

```
# /usr/sbin/axparms -route add radio VK2XLZ VK2SUT
```

Questo comando indica che il percorso per `VK2XLZ` deve avvenire tramite il digipeater `VK2SUT` sulla porta AX.25 chiamata `radio`.

7. Configurazione di un'interfaccia AX.25 per TCP/IP

È molto semplice configurare una porta AX.25 per il trasporto del TCP/IP. Se si ha un'interfaccia KISS ci sono due metodi per configurare un indirizzo IP. Il comando *kissattach* possiede un opzione che permette di configurare l'indirizzo IP; ma il metodo più convenzionale che usa il comando *ifconfig* funzionerà per tutti i tipi d'interfaccia.

Quindi, modificando l'esempio fatto per il KISS:

```
# /usr/sbin/kissattach -i 44.136.8.5 -m 512 /dev/ttyS0 radio
# /sbin/route add -net 44.136.8.0 netmask 255.255.255.0 ax0
# /sbin/route add default ax0
```

si crea un'interfaccia AX.25 con un indirizzo IP `44.136.8.5` e una `MTU` di `512` byte. Comunque occorrerà sempre usare *ifconfig* per configurare, se necessario, gli altri parametri.

Se si hanno altri tipi di interfaccia occorre usare sempre *ifconfig* per configurare l'indirizzo IP, i dettagli di netmask per la porta e aggiungere un percorso (route) attraverso la porta stessa, così come si fa per ogni altra interfaccia TCP/IP. L'esempio che segue è riferito ad un dispositivo per una scheda PI, ma funziona altrettanto bene per ogni altro dispositivo di rete AX.25:

```
# /sbin/ifconfig pi0a 44.136.8.5 netmask 255.255.255.0 up
# /sbin/ifconfig pi0a broadcast 44.136.8.255 mtu 512
# /sbin/route add -net 44.136.8.0 netmask 255.255.255.0 pi0a
# /sbin/route add default pi0a
```

I comandi visti sopra sono tipici delle configurazioni a cui sono abituati gli utenti di NOS o dei suoi derivati, o di ogni altro software TCP/IP. Si noti che il percorso tipico (default route) può non essere richiesto nella propria configurazione, se ci sono altri dispositivi configurati.

Per testare il tutto, si provi un ping o un telnet verso un host locale.

```
# ping -i 5 44.136.8.58
```

Si noti l'uso dell'argomento '`-i 5`' per *ping* per mandare gli impulsi ogni 5 secondi, invece che ogni secondo come scelta predefinita.

8. Configurazione di una porta NET/ROM

Il protocollo NET/ROM usa e si appoggia alle porte AX.25 create in precedenza; per configurarlo su un'interfaccia AX.25 occorre modificare due file: uno descrive l'interfaccia NET/ROM, e l'altro quali porte AX.25 verranno usate per trasportare questo protocollo. Si possono configurare più porte NET/ROM, ognuna col proprio nominativo e alias, usando la stessa procedura.

8.1. Configurazione di `/etc/ax25/nrports`

Il primo file è `/etc/ax25/nrports`. Questo file descrive le porte NET/ROM pressapoco nello stesso modo in cui `/etc/ax25/axports` descrive le porte AX.25. Ogni dispositivo NET/ROM che si vuole creare deve avere una voce di descrizione all'interno del file `/etc/ax25/nrports`. Normalmente una macchina Linux avrà configurato un unico dispositivo NET/ROM che usa un certo numero delle porte AX.25 definite, ma in alcune situazioni, come ad esempio nei BBS, si potrebbero volere diversi alias NET/ROM.

Questo file è formattato nel seguente modo:

```
name callsign  alias  paclen   description
```

Dove:

name

è il nome con cui si fa riferimento alla porta.

callsign

è il nominativo che verrà usato dal traffico NET/ROM di questa porta. Si noti che questo *non* è l'indirizzo al quale si connettono gli utenti per avere un accesso ad un interfaccia di tipo *node* . (Il programma node è descritto più avanti). Questo nominativo/ssid dovrebbe essere unico e non dovrebbe apparire in alcun altro punto dei file `/etc/ax25/axports` o `/etc/ax25/nrports`.

alias

è l'alias NET/ROM assegnato a questa porta.

paclen

è la dimensione massima dei frame NET/ROM trasmessi dalla porta.

description

è una descrizione libera della porta.

Un tipico esempio potrebbe essere il seguente:

```
netrom VK2KTJ-9        LINUX   236     Linux Switch Port
```

In questo modo viene creata una porta NET/ROM, conosciuta dal resto della rete NET/ROM come 'LINUX:VK2KTJ-9'.

Questo programma viene usato da programmi come *call*.

8.2. Configurazione di /etc/ax25/nrbroadcast

Il secondo file di configurazione è /etc/ax25/nrbroadcast. Questo file può contenere diverse voci; normalmente ci dovrebbe essere una voce per ogni porta AX.25 su cui si vuole consentire il traffico NET/ROM.

Questo file ha il seguente formato:

```
axport min_obs def_qual worst_qual verbose
```

Dove:

axport

: è il nome della porta ricavato dal file /etc/ax25/axports. Se per una porta non è presente una voce nel file /etc/ax25/nrbroadcasts significa che da questa non sarà instradato traffico NET/ROM, così come saranno ignorati i broadcast NET/ROM per quella porta.

min_obs

: è il valore minimo di obsolescenza per la porta.

def_qual

: è il valore di default della qualità per la porta.

worst_qual

: è il peggior valore di qualità consentito per la porta; ogni route al di sotto di questo livello sarà ignorata.

verbose

: è un flag che indica se da questa porta avvengono broadcast del routing NET/ROM completi, o solo di annuncio per il nodo stesso.

Un esempio può essere il seguente:

```
radio    1       200      100         1
```

8.3. Creazione del dispositivo di rete NET/ROM

Una volta pronti i due file, occorre creare il dispositivo NET/ROM con un metodo molto simile a quello usato per creare i dispositivi AX.25.In questo caso si usa il comando *nrattach*, che funziona pressapoco nello stesso modo di *axattach*, ad eccezione del fatto che crea dei dispositivi di rete NET/ROM chiamati 'nr[0-9]'.. Anche in questo caso, per primo verrà creato il dispositivo '`nr0`', poi '`nr1`' eccetera. Per creare il dispositivo per la porta NET/ROM definita in precedenza si userà:

```
# nrattach netrom
```

Questo comando inizializzerà il dispositivo NET/ROM (`nr0`) chiamato `netrom` configurato nel modo definito dai parametri del file `/etc/ax25/nrports`.

8.4. Lancio del demone NET/ROM

Il kernel di Linux gestisce tutto il protocollo NET/ROM, ad eccezione di alcune funzioni. Il demone NET/ROM gestisce le tabelle di instradamento (routing tables) e genera la trasmissione del routing NET/ROM. Il demone NET/ROM viene lanciato dal comando:

```
# /usr/sbin/netromd -i
```

A questo punto il file `/proc/net/nr_neigh` dovrebbe cominciare a riempirsi di informazioni relative ai nodi NET/ROM adiacenti.

Si ricordi di mettere il comando `/usr/sbin/netromd` nei propri file *rc*, in modo che il demone venga lanciato ogni volta che si fa ripartire il sistema.

8.5. Configurazione del routing NET/ROM.

Volendo configurare degli instradamenti NET/ROM statici per degli host specifici si può usare il comando *nrparms*; ancora una volta si rimanda alle pagine *man* relative, non prima di proporre un esempio come il seguente:

```
# /usr/sbin/nrparms -nodes VK2XLZ-10 + #MINTO 120 5 radio VK2SUT-9
```

Questo comando abilita una route NET/ROM per `#MINTO:VK2XLZ-10` attraverso `VK2SUT-9` sulla porta AX.25 chiamata '`radio`'.

Si possono creare manualmente voci per nuovi host vicini usando sempre il comando *nrparms*. Ad esempio:

```
# /usr/sbin/nrparms -routes radio VK2SUT-9 + 120
```

questo comando crea `VK2SUT-9` come nodo NET/ROM adiacente con qualità `120`; questa voce sarà statica e quindi non sarà cancellata automaticamente.

9. Configurazione di un interfaccia NET/ROM per TCP/IP

Il processo di configurazione di un interfaccia NET/ROM per TCP/IP è quasi del tutto identico a quella di un interfaccia AX.25 per TCP/IP.

Anche in questo caso occorre specificare indirizzo ip e mtu nella linea di comando di *nrattach*, o usare i comandi *ifconfig* e *route*, ma occorre aggiungere manualmente le voci *arp* per gli host verso i quali si vuole creare una route, poiché non c'èegrave; nessun meccanismo che permetta alla propria macchina di sapere quale indirizzo NET/ROM usare per raggiungere un particolare host IP.

Per questo, per creare un dispositivo `nr0` con indirizzo IP `44.136.8.5`, mtu di `512` e configurato nel modo descritto nel file `/etc/ax25/nrports` per la porta NET/ROM chiamata `NET/ROM`, si userà il comando:

```
# /usr/sbin/nrattach -i 44.136.8.5 -m 512 NET/ROM
# route add 44.136.8.5 nr0
```

oppure una cosa simile alla seguente sequenza di comandi:

```
# /usr/sbin/nrattach NET/ROM
# ifconfig nr0 44.136.8.5 netmask 255.255.255.0 hw NET/ROM VK2KTJ-9
# route add 44.136.8.5 nr0
```

Per ogni host IP che si voglia raggiungere via NET/ROM occorre dunque indicare route e arp. Per raggiungere un host con indirizzo IP `44.136.80.4` all'indirizzo NET/ROM `BBS:VK3BBS` tramite un nodo NET/ROM con nominativo `VK2SUT-0` si useranno i seguenti comandi:

```
# route add 44.136.80.4 nr0
# arp -t NET/ROM -s 44.136.80.4 vk2sut-0
# nrparms -nodes vk3bbs + BBS 120 6 sl0 vk2sut-0
```

Gli argomenti di *nrparms*, '`120`' e '`6`' sono rispettivamente i valori NET/ROM di *quality* e *obsolescence count* per la route.

10. Configurazione di una porta Rose

Il protocollo packet Rose è simile al livello tre delle specifiche X.25. Il suo supporto nel kernel di Linux è una versione *modificata* dell'Implementazione Rose FPAC (http://fpac.lmi.ecp.fr/f1oat/f1oat.html).

Il protocollo ROSE usa le porte AX.25 che sono presenti nel sistema e si appoggia a questo protocollo. Per configurare il ROSE occorre creare un file di configurazione che descrive le porte ROSE che si vogliono creare; per ogni porta la procedura è la stessa.

10.1. Configurazione di `/etc/ax25/rsports`

Il file nel quale si configurano le interfacce ROSE è `/etc/ax25/rsports`. In esso vengono descritte le porte ROSE più o meno nello stesso modo in cui il file `/etc/ax25/axports` descrive le porte AX.25.

Questo file è formattato nel seguente modo:

```
name  address  description
```

Dove:

name
: è il nome con il quale si fa riferimento alla porta

address
: è l'indirizzo ROSE a dieci cifre che si vuole assegnare alla porta.

description
: è una descrizione libera della porta.

Un esempio potrebbe essere il seguente:

```
rose  5050294760  Porta Rose
```

Si noti che, a meno di specificare diversamente, il ROSE usa il nominativo/ssid di default configurato su ogni porta AX.25.

Per configurare un nominativo/ssid da far usare al ROSE su ogni porta, si usa il comando *rsparms* come segue:

```
# /usr/sbin/rsparms -call VK2KTJ-10
```

Questo esempio fa sì che la macchina Linux usi il nominativo `VK2KTJ-10` in tutte le porte AX.25 configurate per traffico ROSE.

10.2. Creazione di un dispositivo di rete ROSE

Una volta creato il file `/etc/ax25/rsports` si possono creare i dispositivi ROSE allo stesso modo in cui sono stati creati i dispositivi AX.25. In questo caso si usa il comando *rsattach*, che crea i dispositivi di rete chiamati '`rose[0-5]`'. La prima volta viene creato '`rose0`', la seconda '`rose1`' e così via. Ad esempio:

```
# rsattach rose
```

Questo comando inizializza il dispositivo ROSE (`rose0`) configurato nel modo descritto nel file `/etc/ax25/rsports` per la porta chiamata '`rose`'.

10.3. Configurazione del routing ROSE

Attualmente il protocollo ROSE supporta solo l'instradamento statico. L'utilità *rsparms* permette di configurare la tabella di routing ROSE per Linux.

Ad esempio:

```
# rsparms -nodes add 5050295502 radio vk2xlz
```

aggiunge una route al nodo ROSE `5050295502` attraverso una porta AX.25 chiamata '`radio`' nel file `/etc/ax25/axports`, per un host col nominativo `VK2XLZ`.

Le route possono essere specificate con una maschera per includere diverse destinazioni in un unica voce ROSE. La sintassi è la seguente:

```
# rsparms -nodes add 5050295502/4 radio vk2xlz
```

che è identica all'esempio precedente, ad eccezione del fatto che si applica ad ogni destinazione che ha un indirizzo che inizia con le quattro cifre `5050`. Una forma alternativa per questo comando è:

```
# rsparms -nodes add 5050/4 radio vk2xlz
```

che è probabilmente quella meno ambigua

11. Come effettuare chiamate AX.25/NET/ROM/Rose

Una volta configurate ed attivate le interfacce AX.25, NET/ROM e Rose, si è in grado di effettuare collegamenti di prova.

Nelle AX25 Utilities è incluso un programma chiamato '*call*' che è un terminale per AX.25, NET/ROM e Rose.

Una semplice chiamata AX.25 sarà del tipo:

```
/usr/bin/call radio VK2DAY via VK2SUT
```

Una semplice chiamata NET/ROM ad un nodo con alias `SUNBBS` sarà:

```
/usr/bin/call NET/ROM SUNBBS
```

Mentre una semplice chiamata Rose verso `HEARD` al nodo `5050882960` sarà fatta nel seguente modo:

```
/usr/bin/call rose HEARD 5050882960
```

Nota: occorre dire al programma *call* su quale porta si vuole effettuare la chiamata, poiché lo stesso nodo di destinazione potrebbe essere raggiungibile tramite diverse porte configurate.

call è un programma terminale a linea di comando per effettuare chiamate AX.25. Riconosce linee che iniziano con '~' come comandi. Il comando '`~.`' chiude il collegamento.

Si faccia riferimento alle pagine man relative in `/usr/man` per maggiori informazioni.

12. Configurare Linux per accettare connessioni Packet

Linux è un sistema operativo molto potente e offre un elevato grado di flessibilità nella sua configurazione. Questa flessibilità porta come effetto collaterale una relativa complicatezza nell'operazione di configurazione. Quando si configura una macchina Linux per accettare connessioni AX.25, NET/ROM o Rose occorre farsi diverse domande. La più importante è "cosa voglio che gli utenti vedano quando mi connetto?". Molti hanno sviluppato belle applicazioncine che possono offrire servizi agli utenti che si connettono; alcune sono semplici come il programma *pms*, altre più complesse come *node* (entrambi presenti nelle utilità AX25). Alternativamente si potrebbe dare agli utenti un account sulla macchina, oppure potreste aver scritto un vostro programma, come un database personalizzato, o un gioco al quale volete che gli utenti si connettano. Qualunque soluzione scegliate, dovete informare il software AX.25 su quale programma lanciare quando accetta connessioni AX.25.

Il demone *ax25d* è simile a *inetd*, che viene comunemente usato per accettare e gestire le connessioni TCP/IP su macchine unix. Il compito di *ax25d* è quello di raccogliere e gestire i tentativi di connessione AX.25. Quando ne sente uno, controlla un file di configurazione per stabilire quale programma lanciare e connetterlo al chiamante. Poiché *ax25d* è lo strumento standard per accettare connessioni AX.25, NET/ROM e Rose, verrà descritto come configurarlo.

12.1. Creazione del file `/etc/ax25/ax25d.conf`

Questo è il file di configurazione per il demone AX.25 *ax25d*, che gestisce le connessioni entranti AX.25, NET/ROM e ROSE.

Ad una prima occhiata può apparire un po' criptico, ma presto si nota come in pratica sia molto semplice, avendo l'accortezza di evitare una piccola trappola.

Il formato generale del file `ax25d.conf` è il seguente:

```
# Questo è un commento ed è ignorato dal progamma ax25d.
[nome_porta] || <nome_porta> || {nome_porta}
<peer1>    window T1 T2 T3 idle N2 <mode> <uid> <cmd> <cmd-name> <arguments>
<peer2>    window T1 T2 T3 idle N2 <mode> <uid> <cmd> <cmd-name> <arguments>
parameters window T1 T2 T3 idle N2 <mode>
<peer3>    window T1 T2 T3 idle N2 <mode> <uid> <cmd> <cmd-name> <arguments>
   ...
default    window T1 T2 T3 idle N2 <mode> <uid> <cmd> <cmd-name> <arguments>
```

Dove:

#

all'inizio di una riga indica che questa è un commento e deve essere ignorato da *ax25d*.

<nome_porta>

è il nome della porta AX.25, NET/ROM o ROSE come specificato nei file `/etc/ax25/axports`, `/etc/ax25/nrports` e `/etc/ax25/rsports`. Il nome della porta è circondato da parentesi quadre '`[]`' se è una porta AX.25, da parentesi acute '<>' se NET/ROM, o graffe '`{}`' se Rose. Esiste una forma alternativa per questo campo, che consiste nel far precedere il nome della porta da '`callsign/ssid via`' per indicare che si vogliono accettare chiamate al nominativo/ssid attraverso quest'interfaccia. L'esempio dovrebbe rendere più chiaro il tutto.

<peer>

è il nominativo del nodo a cui si applica questa particolare configurazione. Se non si specifica un SSID, questa configurazione sarà applicata a tutti i SSID del nominativo.

window

è il parametro AX.25 "Window" (K) conosciuto anche come MAXFRAME che si applica in questa configurazione.

T1

è il valore di tempo di ritrasmissione del frame (T1) espresso in mezzi secondi.

T2

è il tempo, espresso in secondi, che il software AX.25 attende per un altro frame in ingresso, prima di mandare una risposta.

T3

è il tempo di inattività espresso in secondi, prima che il software AX.25 interrompa la sessione.

idle

è il valore di idle espresso in secondi.

N2

il numero di ritrasmissioni consecutive che possono essere fatte prima di interrompere la connessione.

<mode>

fornisce un meccanismo per stabilire alcuni tipi di permessi. I modi sono abilitati o disabilitati fornendo una combinazione di caratteri, ognuna rappresentante un permesso. I caratteri possono essere scritti sia in maiuscolo che in minuscolo, in un unico blocco, senza spazi.

u/U

UTMP - attualmente non supportato.

v/V

Validate call - attualmente non supportato.

q/Q

Quiet - Non viene fatto il log della connessione

n/N

check NET/ROM Neighbour - Attualmente non supportato.

d/D

Disallow Digipeaters - La connessione dev'essere diretta, non effettuata tramite digipeater.

l/L

Lockout - Non permette la connessione.

*/0

marker - Mette un marker, nessun modo settato.

<uid>

è l'utente con il quale viene eseguito il programma per gestire la connessione

<cmd>

è il path completo del comando da lanciare, senza specificare argomenti.

<cmd-name>

è il nome del programma lanciato, così come deve apparire in un comando *ps* as (normalmente è lo stesso valore di <cmd> privo del path).

<arguments>

sono gli argomenti da passare a <:cmd> quando viene lanciato. Possono essere passate utili informazioni con i seguenti token:

%d

Nome della porta sulla quale si è ricevuta la connessione.

%U

Nominativo AX.25 della stazione collegata senza ssid e scritto in maiuscolo.

%u

Nominativo AX.25 della stazione collegata senza ssid e scritto in minuscolo.

%S

Nominativo AX.25 della stazione collegata con ssid e scritto in maiuscolo.

%s

Nominativo AX.25 della stazione collegata con ssid e scritto in minuscolo.

%P

Nominativo AX.25 del nodo dal quale è venuta la chiamata, senza ssid e in maiuscolo.

%p

Nominativo AX.25 del nodo dal quale è venuta la chiamata, senza ssid e in minuscolo.

%R

Nominativo AX.25 del nodo dal quale è venuta la chiamata, con ssid e in maiuscolo.

%r

Nominativo AX.25 del nodo dal quale è venuta la chiamata, con ssid e in minuscolo.

Occorre una sezione scritta nel formato visto sopra per ogni interfaccia AX.25, NET/ROM o ROSE dalla quale si vogliano accettare connessioni.

Nel paragrafo ci sono due tipi speciali di righe: uno inizia con la stringa '`parameters`' e l'altro con la stringa '`default`' (sì, c'è differenza). Queste servono per funzioni speciali.

Lo scopo delle linee '`default`' dovrebbe essere ovvio; queste regolano il comportamento di quelle connessioni per le quali non sono specificate regole specifiche. In assenza di una regola di '`default`', ogni connessione che non è riconducibile ad una regola definita, sarà rifiutata e il chiamante disconnesso senza alcun messaggio.

La riga '`parameters`' assolve un compito più sottile, qui sta la trappola menzionata precedentemente. In ogni campo di ogni definizione per una regola di connessione, si può usare il carattere '*' per dire 'usa i valori di default'. La linea '`parameters`' è quella che definisce questi valori. Lo stesso kernel ha alcuni valori di default che possono essere usati se non li si specifica con la linea '`parameters`' La trappola è che questi valori di default si applicano *solo* alle regole *sotto* la linea '`parameters`'. Si possono avere diverse linee '`parameters`' per interfaccia in modo da creare gruppi di configurazioni di default. È importante notare che '`parameters`' non permette di settare i valori dei campi '`uid`' o '`command`'.

12.2. Un semplice esempio di file `ax25d.conf`

Ok, a questo punto si impone un esempio chiarificatore:

```
# ax25d.conf for VK2KTJ - 02/03/97
# Questo file di configurazione usa le porte definite precedentemente

# <peer> Win T1  T2  T3  idl N2 <mode> <uid> <exec> <argv[0]>[<args....>]

[VK2KTJ-0 via radio]
parameters 1    10  *  *  *   *   *
VK2XLZ     *     *  *  *  *   *   *    root  /usr/sbin/axspawn axspawn %u +
VK2DAY     *     *  *  *  *   *   *    root  /usr/sbin/axspawn axspawn %u +
NOCALL     *     *  *  *  *   *   L
default    1    10  5 100 180 5   *    root  /usr/sbin/pms pms -a -o vk2ktj

[VK2KTJ-1 via radio]
default    *     *     *   *   *   0    root /usr/sbin/node node
```

```
<NET/ROM>
parameters 1    10  *  *  *   *   *
NOCALL     *     *  *  *  *   *   L
default    *     *  *  *  *   *   0        root /usr/sbin/node node

{VK2KTJ-0 via rose}
parameters 1    10  *  *  *   *   *
VK2XLZ     *     *  *  *  *   *   *    root  /usr/sbin/axspawn axspawn %u +
VK2DAY     *     *  *  *  *   *   *    root  /usr/sbin/axspawn axspawn %u +
NOCALL     *     *  *  *  *   *   L
default    1    10  5 100 180 5   *    root  /usr/sbin/pms pms -a -o vk2ktj

{VK2KTJ-1 via rose}
default    *     *     *  *   *   0    root /usr/sbin/node node radio
```

Questo esempio dice che chiunque voglia connettersi alla stazione ’`VK2KTJ-0`’ ascoltato sulla porta AX.25 chiamata ’`radio`’ dovrà sottostare alle seguenti regole:

Chiunque abbia il nominativo col valore ’NOCALL’ non viene fatto entrare; si noti l’uso del modo ’L’.

La linea `parameters` modifica due parametri rispetto a quelli di default del kernel (Window e T1) e verrà lanciato il programma */usr/sbin/axspawn* per loro. Ogni istanza di */usr/sbin/axspawn* lanciata in questo modo apparirà come *axspawn* nell’output di un comando *ps*. Le due linee successive danno le regole per due stazioni che riceveranno questi permessi.

L’ultima linea nel paragrafo costituisce la regola che sarà applicata a tutti gli altri (compresi VK2XLZ e VK2DAY se usano un SSID diverso da -1). Questa definizione imposta implicitamente tutti i parametri e fa sì che il programma *pms* sia lanciato con un argomento che indica che funziona su una connessione AX.25, e che il nominativo utilizzato per rispondere è `VK2KTJ`. (Si veda il paragrafo ’Configurazione del PMS’ per maggiori dettagli).

La configurazione successiva accetta connessioni a `VK2KTJ-1` tramite la porta `radio` e lancia il programma *node* per chiunque si connetta.

Poi c’è una configurazione NET/ROM (si noti l’uso delle parentesi acute "< >"). Questa è più semplice: stabilisce che chiunque si connetta alla porta dal nome ’`NET/ROM`’ farà partire il programma *node*, a meno che non abbia nominativo ’`NOCALL`’, nel qual caso viene disconnesso.

Le ultime due configurazioni sono per le connessioni Rose; la prima per chi chiama ’`vk2ktj-0`’, l’altra per chi chiama ’`VK2KTJ-1` all’indirizzo di nodo Rose della macchina. Queste funzionano esattamente allo stesso modo; si noti l’uso delle parentesi graffe per caratterizzarle come porte Rose.

L’esempio sopra illustrato è piuttosto banale, ma penso che chiarisca sufficientemente le caratteristiche significative della sintassi del file di configurazione. Una descrizione approfondita di quest’ultima si può trovare nelle pagine *man* di `ax25d.conf`. Un esempio più dettagliato è incluso, invece, nelle Li`ax25-utils`.

12.3. Lanciare *ax25d*

Una volta approntati i due file di configurazione, si può lanciare *ax25d* col comando:

```
# /usr/sbin/ax25d
```

Una volta fatto questo, i corrispondenti dovrebbero essere in grado di effettuare connessioni AX.25 alla vostra macchina Linux. Si ricordi di mettere il comando `ax25d` nei propri file *rc*, in modo che venga lanciato ogni volta che la macchina viene fatta ripartire.

13. Configurazione del programma *node*

node è stato sviluppato da Tomi Manninen (mailto:tomi.manninen@hut.fi) e si basa sul programma PMS. Rende disponibili le funzionalità di nodo in modo piuttosto completo e flessibile, permettendo agli utenti, una volta connessi, di fare connessioni in uscita di tipo Telnet, AX.25, NET/ROM e Rose, nonchè di ottenere informazioni con Finger, Nodes, Heard eccetera. Si può inoltre configurare il nodo in modo da far eseguire qualunque comando Linux in modo piuttosto semplice.

Node viene normalmente lanciato dal programma *ax25d*, per quanto possa essere eseguito anche da linea di comando oppure lanciato dal programma TCP/IP *inetd* per permettere agli utenti di fare telnet sulla vostra macchina.

13.1. Creazione del file `/etc/ax25/node.conf`

Il file `node.conf` è dove viene scritta la configurazione principale del nodo. È un semplice file di testo ed è formattato nel seguente modo:

```
# /etc/ax25/node.conf
# file di configurazione del programma node(8).
#
# Le linee che iniziano con '#' sono commenti e vengono ignorate.
# Hostname
# Specifica il nome della macchina che fa da nodo.
hostname radio.gw.vk2ktj.ampr.org

# Rete Locale
# Permette di specificare cosa dev'essere considerato 'locale'
# per il controllo dei permessi usando nodes.perms.
localnet 44.136.8.96/29

# Occultamento di alcune porte
# Se viene specificato, questo parametro permette di rendere le porte
# invisibili agli utenti. Le porte qui elencate non saranno riportate
# dal comando (P)orts.
hiddenports rose NET/ROM

# Identificazione del nodo.
# Questo apparirà al prompt del nodo.
NodeId LINUX:VK2KTJ-9

# Porta NET/ROM
# Questo è il nome della porta NET/ROM che verrà usata per
# le connessioni NET/ROM in uscita dal nodo.
NrPort NET/ROM
```

```
# Node Idle Timeout
# Specifica il tempo di idle in secondi per le connessioni fatte
# a questo nodo (cioè quanto possono rimanere inattive prima che la
# connessione venga interrotta.
idletimout 1800

# Connection Idle Timeout
# Specifica il tempo di idle in secondi per le connessioni fatte
# attraverso questo nodo.
conntimeout 1800

# Riconnessione
# Specifica se gli utenti debbano essere riconnessi al nodo
# se la loro connessione remota si interrompe, o se debbano essere
# definitivamente disconnessi.
reconnect on

# Alias dei comandi
# Permette di rendere semplici dei comandi di nodo più articolati.
alias CONV "telnet vk1xwt.ampr.org 3600"
alias BBS "connect radio vk2xsb"

# Aliases dei comandi esterni
# Permette di eseguire dei comandi esterni all'interno del nodo.
# La sintassi è:
# extcmd <nomecmd> <flag> <userid> <comando>
# Flag == 1 è l'unica funzione implementata.
# <comando> è formattato sullo stile di ax25d.conf
extcmd PMS 1 root /usr/sbin/pms pms -u %U -o VK2KTJ

# Logging
# Stabilisce la quantità di informazioni che vengono scritte nel log.
# 3 per il maggior numero di informazioni, 0 per disabilitare il log.
loglevel 3

# Il carattere di escape
# 20 = (Control-T)
EscapeChar      20
```

13.2. Creazione del file `/etc/ax25/node.perms`

node consente di assegnare permessi agli utenti. Questi permessi permettono di stabilire quali utenti, ad esempio, sono autorizzati a far uso di opzioni come i comandi (T)elnet e (C)onnect. Il file `node.perms` viene usato per stabilire questo genere di permessi e contiene cinque campi chiave; in ognuno di questi un asterisco '*' serve per indicare 'qualunque cosa' e viene usato per creare le regole di default.

user

Il primo campo rappresenta il nominativo o l'utente al quale devono applicarsi i permessi. Ogni ssid viene ignorato, quindi basta mettere il nominativo semplice.

method

Vengono concessi permessi anche ai protocolli o ai metodi di accesso. Per esempio si può permettere l'uso dell'opzione (C)onnect agli utenti connessi via AX.25 o NET/ROM, ma impedirlo agli altri. Il secondo campo perciò, permette di selezionare a quale metodo di accesso deve applicarsi la regola di accesso. I metodi di accesso sono i seguenti:

Metodo	Descrizione
ampr	L'utente è connesso via telnet da un indirizzo amprnet (44.0.0.0)
ax25	L'utente è connesso tramite AX.25
host	L'utente ha lanciato node dalla linea di comando
inet	L'utente è connesso via telnet da un indirizzo non-locale e non-amprnet
local	L'utente è connesso via telnet da un host 'locale'
NET/ROM	L'utente è connesso tramite NET/ROM
rose	L'utente è connesso tramite ROSE
*	L'utente è connesso in un modo qualunque.

port

Volendo è possibile controllare i permessi per gli utenti AX.25 porta per porta; questo permette di determinare cosa gli utenti sono in grado di fare a seconda della porta alla quale sono connessi. Se si sfrutta questa funzionalità (che funziona solo per le connessioni AX.25), il terzo campo contiene il nome della porta.

password

Si può opzionalmente configurare il nodo in modo che chieda una password alla connessione. Questo può essere utile per proteggere utenti con un livello di accesso particolarmente elevato; in questo campo, se presente, è contenuto il valore della password che dev'essere fornita.

permissions

Il campo permessi è l'ultimo per ciascuna voce nel file. Il valore che contiene è a campi di bit (ogni opzione è rappresentata da un bit più o meno settato a seconda che venga più o meno concesso il permesso per essa). La lista delle opzioni che possono essere controllate, e del relativo valore in bit è il seguente:

Valore	Descrizione
1	Login consentito.
2	(C)onnect AX25 consentito.
4	(C)onnect NET/ROM consentito.
8	(T)elnet verso host locali consentiti.
16	(T)elnet verso host amprnet (44.0.0.0) consentiti.

Valore	Descrizione
32	(T)elnet verso host non-locali e non-amprnet consentiti.
64	(C)onnect AX25 consentito anche su porte occultate.
128	(C)onnect Rose consentito.

Per stabilire una regola particolare, occorre sommare i valori dei singoli permessi che si vuole dare e mettere il numero risultante nel quinto campo.

Un esempio di file `nodes.perms` potrebbe essere il seguente:

```
# /etc/ax25/node.perms
#
#L'operatore del nodo è VK2KTJ, ha password 'secret' e ha tutti
#i permessi per tutti i tipi di connessione
vk2ktj * * secret 255

# Ai seguenti nominativi è impedito connettersi
NOCALL * * * 0
PK232 * * * 0
PMS * * * 0

# Agli utenti INET è impedito connettersi.
* inet * * 0

# Gli utenti AX.25, NET/ROM, Local, Host e AMPR possono fare (C)onnect
# e (T)elnet a host locali e ampr, ma non ad altri indirizzi IP.
* ax25 * * 159
* NET/ROM * * 159
* local * * 159
* host * * 159
* ampr * * 159
```

13.3. Configurazione di *node* per funzionare da *ax25d*

Il programma *node* viene lanciato di solito dal programma *ax25d*. Per fare questo occorre aggiungere delle particolari regole al file `/etc/ax25/ax25d.conf`. Nella configurazione che adotto, voglio permettere agli utenti di scegliere se connettersi a *node* o ad altri servizi. *ax25d* consente di fare questo attraverso l'intelligente creazione di alias delle porte. Ad esempio, data la configurazione di *ax25d* presentata sopra, si vuole configurare *node* in modo che venga lanciato per tutti gli utenti che si connettono a `VK2KTJ-1` Per fare questo si aggiunge questo al file `/etc/ax25/ax25d.conf`:

```
[vk2ktj-1 via radio]
default    *     *    *   *   *   0    root /usr/sbin/node node
```

Questo dice che il kernel di Linux risponderà ad ogni richiesta di connessione al nominativo '`VK2KTJ-1`' ascoltato sulla porta chiamata '`radio`' lanciando il programma *node*

13.4. Configurazione di *node* per funzionare da *inetd*

Se si vuole che i propri utenti siano in grado di fare telnet su una porta della macchina e avere accesso a *node* lo si può fare piuttosto facilmente. La prima cosa da decidere e a quale porta gli utenti si devono connettere In questo esempio si è arbitrariamente scelta la porta 4000, sebbene Toi nella sua documentazione fornisca i passi per sostituire il normale demone Telnet con *node*.

Occorre modificare due file.

In `/etc/services` occorre aggiungere:

```
node 3694/tcp #OH2BNS's node software
```

E in `/etc/inetd.conf` va aggiunto:

```
node stream tcp nowait root /usr/sbin/node node
```

Una volta fatto questo, facendo ripartire il programma *inetd*, ogni utente connesso via telnet alla porta 3694 della macchina riceverà la richiesta di login, l'eventuale richiesta di password e sarà connesso a *node*.

14. Configurazione di *axspawn*

Il programma *axspawn* permette alle stazioni connesse via AX.25 di fare il login sulla macchina Linux. Può essere lanciato da *ax25d* in modo simile a quanto visto per *node*. Per permettere ad un utente l'accesso alla propria macchina occorre aggiungere una linea simile alla seguente nel proprio file `/etc/ax25/ax25d.conf`:

```
default * * * * * 1 root /usr/sbin/axspawn axspawn %u
```

Se la linea finisce con un carattere `+`, l'utente che si connette deve battere invio prima che gli venga concesso il login. Di default la scelta è di non attendere input dall'utente. Ogni singola configurazione host che segue queste righe lancia *axspawn* alla connessione del corrispondente. Alla partenza, *axspawn* controlla che la linea di comando che gli viene passata corrisponda ad un nominativo valido, toglie lo ssid e infine controlla il file `/etc/passwd` per vedere se quell'utente ha un account configurato sulla macchina. Se esiste e la password è `""` (null) o `+`, l'utente è subito fatto entrare; se esiste una password da fornire, viene invitato a digitarla. Se non esiste un'account corrispondente all'utente in `/etc/passwd`, si può configurare *axspawn* affinché ne crei automaticamente uno.

14.1. Creazione del file `/etc/ax25/axspawn.conf`

È possibile modificare il comportamento di *axspawn* agendo sul file di configurazione `/etc/ax25/axspawn.conf` che è formattato nel seguente modo:

```
# /etc/ax25/axspawn.conf
#
# permette la creazione automatica di account utente
create    yes
#
# accesso come utente guest (ospite) se sopra si è scelto "no" o se tutto
# fallisce. Disabilita con "no"
guest     no
#
# nome del gruppo o group id degli utenti creati automaticamente
group     ax25
#
# primo user id da usare
first_uid 2001
#
# massimo numero di user id
max_uid   3000
#
# dove creare la home directory dei nuovi utenti
home      /home/ax25
#
# shell dell'utente
shell     /bin/bash
#
# lega lo user id al nominativo per le chiamate in uscita
associate yes
```

Gli otto parametri di configurazione di *axspawn* hanno il seguente significato:

#

indica una riga di commento

create

se questo campo è settato a `yes`, *axspawn* tenterà di creare automaticamente un account per ogni utente che si connetta e non sia già presente nel file `/etc/passwd`

guest

questo campo indica il nome dell'account che sarà usato per gli utenti che non hanno un account se *create* è settato a `no` e che di solito è `ax25` o `guest`.

group

questo campo indica il nome del gruppo per gli account degli utenti che sono creati automaticamente se non sono presenti nel file `/etc/passwd` first_uid

è il numero del primo userid che sarà utilizzato per la creazione automatica degli utenti.

max_uid

è il valore massimo dell'userid che verrà usato nella creazione di nuovi utenti.

home

è la home directory dei nuovi utenti.

shell

è la shell di login usata dai nuovi utenti.

associate

indica se le connessioni AX.25 in uscita fatte dagli utenti collegati devono essere fatte usando il loro nominativo o quello della macchina.

15. Configurazione di *pms*

Il programma *pms* è l'implementazione di un semplice programma di messaggistica. Sviluppato in origine da Alan Cox, è stato successivamente ampliato da Dave Brown (mailto:dcb@vectorbd.com), N2RJT. Allo stadio attuale è ancora piuttosto semplice, visto che supporta solo l'invio di messaggi al gestore del sistema e la possibilità di ricavare alcune informazioni su di esso, ma Dave è al lavoro per espandere le sue funzionalità e renderlo più utilizzabile.

Una volta che il programma è stato compilato ed installato occorre creare un paio di semplici file per far sì che gli utenti abbiano alcune informazioni sul sistema e modificare le voci opportune nel file `ax25d.conf`, in modo che, alla connessione, agli utenti si presenti l'interfaccia del PMS.

15.1. Creazione del file `/etc/ax25/pms.motd`

Il file `/etc/ax25/pms.motd` contiene il 'messaggio del giorno' che verrà inviato a chi si connette dopo lo header usuale del BBS. Si tratta di un file di testo; qualunque cosa vi inseriate sarà presentata all'utente all'atto della connessione.

15.2. Creazione del file `/etc/ax25/pms.info`

Anche `/etc/ax25/pms.info` è un semplice file di testo, nel quale si possono mettere informazioni più dettagliate sulla configurazione della propria stazione, che viene inviato all'utente in risposta al comando `Info` dal prompt `PMS>`.

15.3. Associazione di nominativi AX.25 con gli utenti di sistema

Quando un utente connesso manda posta ad un nominativo AX.25, *pms* si aspetta che questo sia mappato o associato con un reale utente creato sulla vostra macchina. Questo è descritto in una sezione a parte.

15.4. Aggiunta di PMS in `/etc/ax25/ax25d.conf`

L'aggiunta di *pms* al proprio file `ax25d.conf` è un'operazione molto semplice, tuttavia c'è una piccola cosa da tenere in considerazione. Dave ha aggiunto degli argomenti alla linea di comando di *pms* per permettergli di gestire diverse regole di testo di fine linea. AX.25 e NET/ROM, per convenzione, si aspettano che la fine della linea di testo sia indicata dal comando di ritorno a capo e dall'avanzamento di linea (*carriage return, linefeed*), mentre nei sistemi Unix standard si usa solo il comando di nuova linea (*newline*). Quindi per esempio, se si vuole aggiungere una voce per indicare il lancio del programma pms come azione di default per ogni connessione, si aggiunge una linea di questo tipo:

```
default  1  10 5 100 5   0    root  /usr/sbin/pms pms -a -o vk2ktj
```

Questo lancia il programma *pms*, indicandogli che deve gestire una connessione AX.25 e che il gestore del bbs è `vk2ktj`. Si vedano le relative pagine *man* per vedere come indicare al programma altri tipi di connessione.

15.5. Test del PMS

Per testare il PMS, si può dare il seguente comando dal prompt della shell: # /usr/sbin/pms -u vk2ktj -o vk2ktj Sostituire il proprio nominativo con quello dell'autore, in modo che il comando lanci il pms indicandogli che deve usare la convenzione Unix per il comando di fine linea e che l'utente che si connette è `vk2ktj`. In questo modo ci si trova nella stessa situazione di un utente remoto che si connette.

In aggiunta a questo si può provare a far connettere altri nodi alla propria macchina in modo da controllare che le impostazioni in `ax25d.conf` siano corrette.

16. Configurazione dei programmi *user_call*

I programmi '*user_call*' sono in realtà: *ax25_call* e *NET/ROM_call*. Sono molto semplici e concepiti per essere chiamati da *ax25d* per automatizzare le connessioni a host remoti, anche se nulla vieta di usarli in script della shell o con altri demoni come il programma *node*.

Possono essere considerati simili a un semplice programma *call*. A causa della loro semplicità, non trattano in alcun modo i dati che gestiscono, quindi ad esempio il problema del fine-linea deve essere gestito dall'utente.

Iniziamo con un esempio su come si possano impiegare. Immaginiamo di avere una piccola rete a casa composta da una macchina Linux che funziona come gateway radio e da un'altra macchina, diciamo un nodo BPQ connesso via ethernet.

Normalmente, se si vuole che gli utenti che si connettono via radio possano raggiungere il nodo BPQ, occorre che la prima macchina funga da digipeater, oppure che gli utenti possano connettersi al nodo Linux e poi connettersi all'altra macchina da lì. Il programma *ax25_call* può semplificare quest'operazione se viene chiamato dal programma *ax25_call*.

Supponiamo che il nodo BPQ abbia il nominativo `VK2KTJ-9` e che la macchina Linux abbia la porta AX.25/ethernet chiamata '`bpq`'; immaginiamo anche che questa abbia una porta radio chiamata '`radio`'.

Una voce nel file `/etc/ax25/ax25d.conf` del tipo:

```
[VK2KTJ-1 via radio]
default    * * * *   *   *   *
root /usr/sbin/ax25_call ax25_call bpq %u vk2ktj-9
```

abilita gli utenti alla connessione diretta a 'VK2KTJ-1' che sarebbe in realtà il demone *ax25d*, che li connette automaticamente a 'VK2KTJ-9' attraverso l'interfaccia 'bpq' con una connessione AX.25

C'è tutta una serie di altre possibili configurazioni che si possono provare. Le utilità '*NET/ROM_call*' e '*rose_call*' funzionano in modo simile. Un radioamatore ha usato questa utilità per rendere più semplici le connessioni ad una BBS remota. Poiché normalmente gli utenti dovrebbero inserire una lunga stringa di connessione per fare la chiamata, ha creato una voce che fa apparire la BBS come se si fosse in una rete locale e dove *ax25d* fa da proxy per la connessione alla macchina remota.

17. Configurazione dei comandi di uplink e downlink di ROSE

Se l'implementazione ROM di ROSE è familiare, non sarà difficile orientarsi tra i metodi coi quali gli utenti AX.25 effettuano chiamate su una rete ROSE. Se un nodo ROSE ha il nominativo VK2KTJ-5 e l'utente AX.25 vuole connettersi a VK5XXX al nodo ROSE 5050882960, deve dare il comando:

```
c vk5xxx v vk2ktj-5 5050 882960
```

Sul nodo remoto, VK5XXX vedrà una connessione in entrata dal nominativo locale dell'utente AX.25 propagata dal nominativo del nodo ROSE remoto.

L'implementazione ROSE di Linux non supporta questa funzionalità nel kernel, tuttavia vi sono due programmi che svolgono questa funzione, che sono *rsuplnk* e *rsdwnlnk* .

17.1. Configurazione del downlink ROSE

Per configurare la propria macchina Linux perché accetti connessioni ROSE e stabilisca collegamenti AX.25 per nominativi diversi da quelli locali occorre aggiungere una voce nel proprio file. /etc/ax25/ax25d.conf. Di solito si configura questa voce come scelta di default per ogni connessione ROSE entrante. Ad esempio si può decidere di gestire localmente chiamate Rose a destinazioni come NODE-0 o NODE-1, e di passare le altre chiamate al comando *rsdwnlink*, assumendo che siano utenti AX.25.

Una tipica configurazione può dunque essere:

```
#
{* via rose}
NOCALL   * * * * * *  L
default  * * * * * *  - root  /usr/sbin/rsdwnlnk rsdwnlnk 4800 vk2ktj-5
#
```

Con questa configurazione, ogni utente che stabilisce una connessione ROSE con l'indirizzo del vostro nodo Linux e con un nominativo di destinazione che non avete esplicitamente specificato, sarà convertita in una connessione AX.25 dalla porta 4800 usando come digipeater VK2KTJ-5.

17.2. Configurazione di un uplink ROSE

Per configurare la propria macchina Linux affinché accetti le connessioni AX.25 allo stesso modo di un nodo ROM Rose occorre aggiungere le voci opportune nel proprio file `/etc/ax25/ax25d.conf` che assume un aspetto simile a questo:

```
#
[VK2KTJ-5* via 4800]
NOCALL   * * * * * *  L
default  * * * * * *  - root  /usr/sbin/rsuplnk rsuplnk rose
#
```

Si noti la speciale sintassi per il nominativo locale. il carattere '`*`' indica che l'applicazione dovrebbe essere invocata se il nominativo è presente nel percorso digipeater di una connessione.

Questa configurazione permette ad un utente AX.25 di effettuare connessioni Rose usando l'esempio presentato nell'introduzione. Ogni connessione che tenti di usare `VK2KTJ-5` come digipeater sulla porta AX.25 chiamata `4800` sarà gestita dal comando *rsuplnk*

18. Associare nominativi AX.25 con utenti Linux

Ci sono diverse situazioni in cui è particolarmente opportuno associare un nominativo con un account utente di Linux, ad esempio quando diversi radioamatori condividono la stessa macchina Linux e vogliono usare il proprio nominativo per effettuare chiamate, oppure quando utenti del PMS vogliono fare un talk con un particolare utente della macchina.

Il software AX.25 permette di gestire l'associazione tra utenti e nominativi (se n'è già parlato nella sezione dedicata al PMS).

Questa associazione viene fatta tramite il comando *axparms*. Ad esempio:

```
# axparms -assoc vk2ktj terry
```

associa il nominativo AX.25 `vk2ktj` con l'utente `terry` della macchina Linux. In questo modo la posta di `vk2ktj` del PMS sarà inviata all'account Linux `terry`.

Si rammenti di mettere queste associazioni nel proprio file *rc*, in modo che siano disponibili ad ogni reboot.

Nota non bisogna associare un nominativo con l'accont di `root`, in quanto ciò potrebbe interferire con la configurazione di altri programmi.

19. Configurazione dell'APRS

> **Nota:** Questa sezione dev'essere ancora scritta. Dovrebbe trattare aprsd, aprsdigi, aprsmon, xastir, JavAPRS, ecc.

20. Le voci del file sistem `/proc/`

La directory `/proc` presenta diversi file che sono in relazione con il kernel AX.25 e NET/ROM. Questi sono di solito utilizzati dalle utilità AX25, ma essendo scritti in forma leggibile, può essere interessante dar loro un'occhiata. Il formato che viene usato è facilmente comprensibile, quindi non è neccessario andare molto in dettaglio per comprenderne il significato.

/proc/net/arp

contiene l'elenco delle mappature Address Resolution Protocol tra gli indirizzi IP e gli indirizzi dello strato MAC. Possono essere AX.25, ethernet o altri protocolli di questo strato.

/proc/net/ax25

contiene una lista dei socket AX.25 aperti. Questi possono essere in attesa di una connessione, oppure connessioni attive.

/proc/net/ax25_bpqether

contiene le corrispondenze dei nominativi AX.25 con quelli BPQ ethernet.

/proc/net/ax25_calls

contiene l'elenco delle corrispondenze degli utenti linux con i nominativi; elenco che è stato creato col comando *axparms -assoc*

/proc/net/ax25_route

contiene informazioni per i percorsi AX.25 da effettuare tramite digipeater.

/proc/net/nr

contiene una lista di socket NET/ROM aperti. Questi possono essere in attesa di una connessione, oppure collegamenti attivi.

/proc/net/nr_neigh

contiene informazioni riguardo i nodi NET/ROM vicini (NET/ROM neighbours) conosciuti dal programma.

/proc/net/nr_nodes

contiene informazioni riguardo i nodi NET/ROM conosciuti dal programma.

/proc/net/rose

contiene una lista di socket Rose aperti. Questi possono essere in attesa di una connessione, oppure collegamenti attivi.

/proc/net/rose_nodes

contiene informazioni riguardo percorsi ai nodi Rose vicini (Rose Neighbours).

/proc/net/rose_neigh

contiene una lista di nodi Rose vicini (Rose Neighbours).

/proc/net/rose_routes

contiene una lista di tutte le connessioni Rose attive.

21. Programmazione di rete per AX.25, NET/ROM e Rose

Il vantaggio più grande nell'usare un'implementazione dei protocolli packet per radioamatori è probabilmente la facilità con cui si possono sviluppare applicazioni e programmi che li sfruttino.

Sebbene la programmazione di applicativi di rete in Unix vada al di là degli scopi di questo documento, si descriveranno gli elementi essenziali per utilizzare i protocolli AX.25, NET/ROM e Rose all'interno dei vostri programmi.

21.1. Le famiglie degli indirizzi

La programmazione di rete per AX.25, NET/ROM e Rose è, in Linux, piuttosto simile a quella per TCP/IP, visto che la differenza maggiore sta nelle diverse famiglie di indirizzi e nella loro diversa struttura.

I nomi delle famiglie degli indirizzi per AX.25, NET/ROM e Rose sono rispettivamente `AF_AX25`, `AF_NET/ROM` e `AF_ROSE`.

21.2. I file header

Occorre sempre includere i file header '`netax25/ax25.h`', nonchè '`NET/ROM/NET/ROM.h`' o '`netrose/rose.h`' se avete a che fare con questi protocolli. La struttura di base sarà simile alla seguente:

Per AX.25:

```
#include <netax25/ax25.h>
int s, addrlen = sizeof(struct full_sockaddr_ax25);
struct full_sockaddr_ax25 sockaddr;
sockaddr.fsa_ax25.sax25_family = AF_AX25
```

Per NET/ROM:

```
#include <netax25/ax25.h>
#include <NET/ROM/NET/ROM.h>
int s, addrlen = sizeof(struct full_sockaddr_ax25);
struct full_sockaddr_ax25 sockaddr;
sockaddr.fsa_ax25.sax25_family = AF_NET/ROM;
```

Per ROSE:

```
#include <netax25/ax25.h>
#include <netrose/rose.h>
int s, addrlen = sizeof(struct sockaddr_rose);
```

```
struct sockaddr_rose sockaddr;
sockaddr.srose_family = AF_ROSE;
```

21.3. Trattamento dei nominativi ed esempi

Nella libreria `lib/ax25.a` delle AX.25-utilities vi sono routine che effettuano la conversione e il trattamento dei nominativi, anche se naturalmente potete scriverne di vostre.

Le utilità *user_call* sono eccellenti esempi su cui impostare il vostro lavoro; spendendoci su un po' di tempo si comprende come il novanta percento del lavoro consiste nel riuscire ad aprire il socket. Per la verità effettuare la connessione è semplice, è la preparazione che richiede tempo.

Gli esempi sono semplici a sufficienza da non creare confusione. In caso di dubbi è buona cosa rivolgersi alla mailing list `linux-hams`, dove senz'altro ci sarà qualcuno che vi darà una mano.

22. Alcuni esempi di configurazione

Vengono ora presentati esempi delle configurazioni più tipiche. Devono essere prese come spunto, visto che ci sono tanti modi di configurare reti, quanto il numero di reti stesso, ma possono comunque essere d'ispirazione.

22.1. Piccola LAN Ethernet con macchina Linux che fa da router verso una LAN radio

Molti hanno delle piccole reti locali a casa e vogliono connettere le macchine ivi presenti alla rete radio. Questo è il tipo di configurazione che uso a casa. Ho fatto in modo di avere un blocco di indirizzi a me allocato che per comodità posso catturare in una singola route e che uso per la mia LAN domestica. Il vostro coordinatore IP locale vi assisterà se volete farlo anche voi. Gli indirizzi per la rete locale Ethernet formano un sottoinsieme degli indirizzi della LAN radio; quella che segue è la configurazione che adotto:

```
                                          . . . . . .
  ___               _________          .
   | Rete          /         \      .    Rete
   | 44.136.8.96/29|         |     .     44.136.8/24         \ | /
   |               | Router  |    .                           \|/
   |               |         |   .  _____      __________     |
   |          eth0 | Linux   |   . /     \    /          \    |
   |_______________|         |_____| TNC |____|  Radio   |__/
   |   44.136.8.97 |   e     |   . \_____/    \__________/
   |               |         | sl0
   |               | Server  | 44.136.8.5
   |               |         |    .
   |               |         |      .
   |               \_________/         .
  _|_                                     . . . . . .
```

```
#!/bin/sh
# /etc/rc.net
# Questa configurazione rende disponibile una porta KISS AX.25
# e un dispositivo Ethernet.

echo "/etc/rc.net"
echo "  Sto configurando:"

echo -n "    loopback:"
/sbin/ifconfig lo 127.0.0.1
/sbin/route add 127.0.0.1
echo " fatto."

echo -n "    ethernet:"
/sbin/ifconfig eth0 44.136.8.97 netmask 255.255.255.248 \
broadcast 44.136.8.103 up
/sbin/route add 44.136.8.97 eth0
/sbin/route add -net 44.136.8.96 netmask 255.255.255.248 eth0
echo " fatto."

echo -n "    AX.25: "
kissattach -i 44.136.8.5 -m 512 /dev/ttyS1 4800
ifconfig sl0 netmask 255.255.255.0 broadcast 44.136.8.255
route add -host 44.136.8.5 sl0
route add -net 44.136.8.0 window 1024 sl0

echo -n "    NET/ROM: "
nrattach -i 44.136.8.5 NET/ROM

echo "  Routing:"
/sbin/route add default gw 44.136.8.68 window 1024 sl0
echo "    default route."
echo done.

# end
```

```
/etc/ax25/axports

# nome  nominativo     velocità paclen  window  descrizione
4800    VK2KTJ-0        4800    256     2       144.800 MHz
```

```
/etc/ax25/nrports

# nome  nominativo      alias   paclen  descrizione
NET/ROM  VK2KTJ-9        LINUX   235     Linux Switch Port
```

```
/etc/ax25/nrbroadcast

# nome_ax25     min_obs def_qual        worst_qual      verbose
```

```
4800            1       120             10              1
```

- Occorre che IP_FORWARDING sia abilitato nel vostro kernel.
- Quando necessario fare riferimento ai file di configurazione AX.25 presentati nelle sezioni precedenti.
- Ho scelto di usare un indirizzo IP per la mia porta radio che non fa parte del blocco usato per la mia rete. Ciò non è obbligatorio, per quella porta si sarebbe potuto tranquillamente usare anche `44.136.8.97`.
- `44.136.8.68` è il mio gateway locale dove il traffico IPIP viene incapsulato, e per questo vi punto la mia route di default.
- Ogni macchina sulla mia rete Ethernet ha una route:

  ```
  route add -net 44.0.0.0 netmask 255.0.0.0 \
  gw 44.136.8.97 window 512 mss 512 eth0
  ```

  L'uso dei parametri *mss* e *window* permette di ottimizzare le connessioni sia dal lato radio che da quello Ethernet.

- Sulla macchina router vengono fatti girare smail, http, ftp ed altri demoni, in modo che questa sia l'unica macchina che fornisce servizi ad altri.
- La macchina che funge da router è un piccolo 386DX20 con un disco fisso da 20Mb ed una installazione di Linux minimale.

22.2. Configurazione del gateway di incapsulamento IPIP

> **Attenzione**
>
> Alcune informazioni sul routing qui fornite, sono obsolete. Il setup è cambiato dai tempi del kernel 2.0.x, e ora andrebbe utilizzato il comando "ip" presente nel pacchetto iproute2, come descritto nell'Advanced Routing HOWTO.

Linux oggi è utilizzato spessissimo per i gateway di incapsulamento TCP/IP in tutto il mondo. Il driver per il tunneling supporta route di incapsulamento multiple rendendo obsoleto il vecchio demone *ipip*.

Una configurazione tipica è simile alla seguente:

```
                                                    . . . . . .
  ___                  _________          .
   | Rete             /         \      .    Rete
   | 154.27.3/24      |         |     .     44.136.16/24       \ | /
   |                  | Gateway |    .                          \|/
   |                  |         |   .   _____      __________    |
   |          eth0    | Linux   |   .  /     \    /          \   |
___|__________________|         |_____| TNC  |____|  Radio   |___/
```

```
   |    154.27.3.20 | IPIP     |  .  \_____/     \__________/
   |                |          | sl0
   |                |          | 44.136.16.1
   |                |          |    .
   |                |          |     .
   |                \__________/       .
  _|_                                      . . . . . .
```

I file di configurazione che si usano sono:

```
# /etc/rc.net
# Questa è una semplice configurazione che rende disponibile una
# porta radio KISS AX.25, un dispositivo Ethernet e sfrutta il tunnel
# driver del kernel per effettuare l'incapsulamento/deincapsulamento
# IPIP
#
echo "/etc/rc.net"
echo "  Sto configurando:"
#
echo -n "    loopback:"
/sbin/ifconfig lo 127.0.0.1
/sbin/route add 127.0.0.1
echo " fatto."
#
echo -n "    ethernet:"
/sbin/ifconfig eth0 154.27.3.20 netmask 255.255.255.0 \
broadcast 154.27.3.255 up
/sbin/route add 154.27.3.20 eth0
/sbin/route add -net 154.27.3.0 netmask 255.255.255.0 eth0
echo " fatto."
#
echo -n "    AX.25: "
kissattach -i 44.136.16.1 -m 512 /dev/ttyS1 4800
/sbin/ifconfig sl0 netmask 255.255.255.0 broadcast 44.136.16.255
/sbin/route add -host 44.136.16.1 sl0
/sbin/route add -net 44.136.16.0 netmask 255.255.255.0 window 1024 sl0
#
echo -n "    tunnel:"
/sbin/ifconfig tunl0 44.136.16.1 mtu 512 up
#
echo fatto.
#
echo -n "Routing ... "
source /etc/ipip.routes
echo fatto.
#
# end.
```

e:

```
# /etc/ipip.routes
# Questo file è generato dallo script munge
#
/sbin/route add -net 44.134.8.0 netmask 255.255.255.0 tunl0 gw 134.43.26.1
/sbin/route add -net 44.34.9.0 netmask 255.255.255.0 tunl0 gw 174.84.6.17
/sbin/route add -net 44.13.28.0 netmask 255.255.255.0 tunl0 gw 212.37.126.3
   ...
   ...
   ...
```

```
/etc/ax25/axports

# nome  nominativo     velocità paclen  window  descrizione
4800    VK2KTJ-0        4800    256     2       144.800 MHz
```

Alcuni punti su cui soffermarsi:

- Il nuovo driver per il tunnelling usa il campo *gw* nella tabella di routing al posto del parametro *pointopoint* per specificare l'indirizzo del gateway IPIP remoto. Questo è il motivo per cui ora sono supportate route multiple per ciascuna interfaccia.
- Si *possono* configurare due dispositivi di rete con lo stesso indirizzo. In quest'esempio sia `sl0` che `tunl0` sono stati configurati con l'indirizzo IP della porta radio; in questo modo il gateway remoto vede l'indirizzo corretto nei datagrammi incapsulati che il gateway locale gli invia.
- I comandi di routing usati per generare le route incapsulate possono essere generati da una versione modificata dello script *munge* che viene riportato più sotto. I comandi di routing vengono scritti in un file separato e letto usando il comando *bash* `source /etc/ipip.routes` (supponendo di aver chiamato `/etc/ipip.routes` il file in questione). Il file deve essere nel formato delle route di NOS.
- Si noti l'uso dell'argomento *window* nel comando *route*. Impostando opportunamente questo parametro si migliorano le prestazioni del collegamento radio.

Il nuovo script tunnel-munge:

```
#!/bin/sh
#
# Da: Ron Atkinson <n8fow@hamgate.cc.wayne.edu>
#
#  Questo script è basato sullo script 'munge' scritto da Bdale N3EUA
#  per il demone IPIP, modificato da Ron Atkinson N8FOW. Il suo scopo
#  è quello di convertire un file di gateway nel formato NOS di KA9Q
#  (chiamato di solito 'encap.txt') nel formato delle tabelle di
#  routing di Linux per il tunnel driver IP
#
# Uso: File dei gateway su stdin, file nel formato Linux su stdout.
#            esempio:  tunnel-munge < encap.txt > ampr-routes
#
```

```
# NOTA: Prima di usare questo script assicurarsi di controllare ed
#       eventualmente cambiare i seguenti parametri:
#
#     1) Cambiare le sezioni 'Route locali e 'Altre route
#        dell'utente' con le route presenti nella vostra area
#        (rimuovete le mie!)
#     2) Sulla riga di fgrep assicurarsi di cambiare l'indirizzo IP
#        con il VOSTRO indirizzo di gateway internet. Se non si
#        effettua questa operazione si rischiano seri routing loop.
#     3) L'interfaccia ha nome di default 'tunl0'. Assicurarsi che
#        questa assunzione  sia corretta sul vostro sistema.


echo "#"
echo "# Tabella di routing con IP tunnelling generata da $LOGNAME il
`date`"
echo "# dallo script tunnel-munge v960307."
echo "#"
echo "# Route locali"
echo "route add -net 44.xxx.xxx.xxx netmask 255.mmm.mmm.mmm dev sl0"
echo "#"
echo "# Altre route dell'utente"
echo "#"
echo "# route remote"


fgrep encap | grep "^route" | grep -v " XXX.XXX.XXX.XXX" | \
awk '{
split($3, s, "/")
split(s[1], n,".")
if (n[1] == "") n[1]="0"
if (n[2] == "") n[2]="0"
if (n[3] == "") n[3]="0"
if (n[4] == "") n[4]="0"
if (s[2] == "1") mask="128.0.0.0"
else if (s[2] == "2") mask="192.0.0.0"
else if (s[2] == "3") mask="224.0.0.0"
else if (s[2] == "4") mask="240.0.0.0"
else if (s[2] == "5") mask="248.0.0.0"
else if (s[2] == "6") mask="252.0.0.0"
else if (s[2] == "7") mask="254.0.0.0"
else if (s[2] == "8") mask="255.0.0.0"
else if (s[2] == "9") mask="255.128.0.0"
else if (s[2] == "10") mask="255.192.0.0"
else if (s[2] == "11") mask="255.224.0.0"
else if (s[2] == "12") mask="255.240.0.0"
else if (s[2] == "13") mask="255.248.0.0"
else if (s[2] == "14") mask="255.252.0.0"
else if (s[2] == "15") mask="255.254.0.0"
else if (s[2] == "16") mask="255.255.0.0"
else if (s[2] == "17") mask="255.255.128.0"
else if (s[2] == "18") mask="255.255.192.0"
else if (s[2] == "19") mask="255.255.224.0"
```

```
else if (s[2] == "20") mask="255.255.240.0"
else if (s[2] == "21") mask="255.255.248.0"
else if (s[2] == "22") mask="255.255.252.0"
else if (s[2] == "23") mask="255.255.254.0"
else if (s[2] == "24") mask="255.255.255.0"
else if (s[2] == "25") mask="255.255.255.128"
else if (s[2] == "26") mask="255.255.255.192"
else if (s[2] == "27") mask="255.255.255.224"
else if (s[2] == "28") mask="255.255.255.240"
else if (s[2] == "29") mask="255.255.255.248"
else if (s[2] == "30") mask="255.255.255.252"
else if (s[2] == "31") mask="255.255.255.254"
else  mask="255.255.255.255"

if (mask == "255.255.255.255")
printf "route add -host %s.%s.%s.%s gw %s dev tunl0\n"\
,n[1],n[2],n[3],n[4],$5
else
printf "route add -net %s.%s.%s.%s gw %s netmask %s dev tunl0\n"\
,n[1],n[2],n[3],n[4],$5,mask
 }'

echo "#"
echo "# si utilizza mirrorshades.ucsd.edu come gateway di default per la parte rimanente di amprnet"
echo "route add -net 44.0.0.0 gw 128.54.16.18 netmask 255.0.0.0 dev tunl0"
echo "#"
echo "# the end"
```

22.3. Configurazione del gateway di incapsulamento AXIP

Molti gateway radioamatoriali per Internet incapsulano l'AX.25, NET/ROM e Rose oltre che il tcp/ip. L'incapsulamento di frame AX.25 in datagrammi IP viene descritta nell'RFC-1226 da Brian Kantor. Nel 1991 Mike Westerhof ha scritto un'implementazione del demone di incapsulamento dell'AX.25 per Unix, che viene proposta per Linux nelle ax25-utils in una versione leggermente migliorata.

Un gateway di incapsulamento AXIP prende i frame AX.25, ne ricava l'indirizzo AX.25 di destinazione e in base a questo determina a quale indirizzo IP inviarli, dopo averli incapsulati in datagrammi tcp/ip, che vengono mandati all'indirizzo di destinazione. Inoltre permette anche il percorso inverso, accettando datagrammi tcp/ip che contengono frame AX.25. Questi vengono estratti e trattati come se fossero pervenuti direttamente da una porta AX.25. Per distinguere i datagrammi che contengono frame AX.25, li si marca con un protocol id uguale a 4 (o 94 anche se questo è ora sconsigliato), come descritto dalla RFC-1226.

Il programma *ax25ipd* incluso nelle ax-25utils gestisce un'interfaccia KISS sulla quale si possono far transitare pacchetti AX.25 ed un'interfaccia tcp/ip. Viene configurato tramite il file di configurazione `/etc/ax25/ax25ipd.conf`.

22.3.1. Opzioni di configurazione di AXIP

Il programma *ax25ipd* possiede due modi principali di funzionamento: il modo "digipeater" e il modo "tnc". Nel modo "tnc" il demone viene considerato come un tnc KISS, gli si passano frame KISS incapsulati in modo che li trasmetta, mentre nel modo "digipeater" si comporta, appunto, come un digipeater AX.25. Tra queste due modalità vi sono delle sottili differenze.

Nel file di configurazione si stabiliscono le "route" o le corrispondenze tra i nominativi AX.25 e gli indirizzi IP degli host ai quali si vogliono mandare i pacchetti AX.25. Ogni route possiede delle opzioni che verranno spiegate più avanti.

Altre opzioni che vengono configurate sono:

- la tty che il demone *ax25ipd* apre e la sua velocità (di solito è un'estremità di una pipe)
- che nominativo usare in modo "digipeater"
- l'intervallo di emissione e il testo trasmesso dal beacon.
- se si vogliono incapsulare i frame AX.25 in datagrammi IP oppure UDP/IP. Quasi tutti i gateway AXIP usano l'IP encapsulation, ma alcuni sono dietro a firewall che non permettono il passaggio a datagrammi col protocol id dell'AXIP, costringendoli ad usare UDP/IP. Quale che sia la scelta, deve essere uguale a quella dell'host TCP/IP dall'altra parte del collegamento.

22.3.2. Un tipico esempio di `/etc/ax25/ax25ipd.conf`

```
#
# file di configurazione ax25ipd per la stazione floyd.vk5xxx.ampr.org
#
# Selezione del tipo di trasporto. 'ip' permette la compatibilità
# con la maggior parte dei gateway
#
socket ip
#
# Indicazione del tipo di modalità ax25ipd (digi or tnc)
#
mode tnc
#
# Se si è scelta la modalità digi, occorre definire un nominativo.
# Se si è in modo tnc, il nominativo è attualmente opzionale, ma ciò
# può cambiare in futuro (2 nominativi se si usano due porte kiss)
#
#mycall vk5xxx-4
#mycall2 vk5xxx-5
#
# In modalità digi si può indicare un alias. (2 se si usano due porte kiss)
#
#myalias svwdns
#myalias2 svwdn2
#
# Si manda l'identificativo ogni 540 secondi ...
```

```
#
#beacon after 540
#btext ax25ip -- tncmode rob/vk5xxx -- Gateway sperimentale AXIP
#
# Porta seriale, o pipe connessa a kissattach in questo caso.
#
device /dev/ttyq0
#
# Velocità del dispositivo
#
speed 9600
#
# loglevel 0 - nessun output
# loglevel 1 - solo informazioni di configurazione
# loglevel 2 - principali eventi ed errori
# loglevel 3 - principali eventi ed errori, nonchè la traccia dei
#              frame AX.25
# loglevel 4 - tutti gli eventi
# log 0 per il momento, con syslog ancora non funziona ....
#
loglevel 2
#
# Se siamo in modalità digi, ci dev'essere un vero tnc,
# quindi uso param per settare i suoi parametri ....
#
#param 1 20
#
# Definizione degli indirizzi di broadcast. Ognuno degli indirizzi
# indicati sarà inoltrato ad ogni route indicate come in grado di effettuare il
# broadcast.
#
broadcast QST-0 NODES-0
#
# definizione delle route ax.25
# il formato è route (nominativo/carattere jolly) (ip dell'host di
# destinazione) Se il ssid è zero la regola si applica a tutti i ssid.
#
# route <destcall> <destaddr> [flags]
#
# I flag validi sono
#         b  - permette il transito dei broadcast attraverso questa
#              route
#         d  - indica che questa è la route di default
#
route vk2sut-0 44.136.8.68 b
route vk5xxx 44.136.188.221 b
route vk2abc 44.1.1.1
#
#
```

22.3.3. Uso di *ax25ipd*

Occorre creare le voci opportune nel file `/etc/ax25/axports`:

```
# /etc/ax25/axports
#
axip VK2KTJ-13 9600 256 porta AXIP
#
```

Va usato il comando *kissattach* per creare la porta da utilizzare:

```
/usr/sbin/kissattach /dev/ptyq0 axip 44.135.96.242
```

Si lancia il programma *ax25ipd*:

```
/usr/sbin/ax25ipd &
```

Per testare il link AXIP link:

```
call axip vk5xxx
```

22.3.4. Alcune note riguardo le route e i loro flag

Col comando "`route`" si specifica dove si vuole che i propri pacchetti AX.25 siano incapsulati e spediti. Quando il demone *ax25ipd* riceve un pacchetto dalla sua interfaccia, confronta il nominativo di destinazione con tutti quelli presenti nella tabella di routing. Se lo trova, il pacchetto AX.25 viene incapsulato in un datagramma IP e poi trasmesso all'host avente l'indirizzo IP indicato.

Ci sono due flag che si possono aggiungere ad ogni comando di route nel file `ax25ipd.conf`:

b

: il traffico che ha come destinazione gli indirizzi definiti dalla parola chiave "`broadcast`" devono essere trasmessi attraverso questa route.

d

: ogni pacchetto il cui indirizzo non compare in alcuna route deve essere trasmessa attraverso questa route.

Il flag di broadcast è molto utile, poiché permette di inviare informazioni destinate a tutte le stazioni, a molte destinazioni AXIP. Normalmente le route AXIP sono di tipo punto-punto ed incapaci di gestire pacchetti di tipo 'broadcast'.

22.4. Collegare NOS e Linux con un dispositivo pipe

Molti radioamatori utilizzano alcune versioni di NOS sotto Linux, poiché hanno a disposizione tutte le funzionalità a cui sono abituati; a molti di questi piacerebbe che il loro NOS potesse colloquiare col kernel di Linux in modo di poter mettere a disposizione le funzionalità del sistema operativo agli utenti che si collegano via radio con NOS.

Brandon S. Allbery, KF8NH, ha fornito queste informazioni, che consentono di interconnettere il NOS con Linux tramite il dispositivo pipe.

Poiché sia Linux che NOS supportano il protocollo slip, si possono connettere creando un link di questo tipo. È possibile realizzare il collegamento utilizzando due porte seriali della stessa macchina con un cavo loopback tra loro, ma questa realizzazione risulterebbe lenta e costosa. Linux, al contrario, fornisce una funzionalità tipica dei sistemi Unix chiamata 'pipe'. I pipe sono degli 'pseudo-dispositivi' che vengono visti dai programmi come normali dispositivi tty, ma che in realtà fungono da collegamento verso un altro pipe. Per usarli il programma chiamante deve attivare il pipe *master* e, successivamente, il programma chiamato deve fare lo stesso col pipe *slave*. Una volta aperte le due porte, i programmi possono comunicare tra loro semplicemente scrivendo caratteri sul dispositivo pipe, esattamente come se fossero normali terminali seriali.

Per usare questa funzionalità per connettere il kernel di Linux con una copia di NOS od altri programmi, occorre per prima cosa scegliere il pipe da usare. I pipe trovano nella directory `/dev`; le parti master del pipe sono chiamate `ptyq[1-f]`, mentre quelle slave sono `ttyq[1-f]`. Si ricordi che vanno sempre in coppia, per cui se si sceglie `/dev/ptyqf` come parte master, occorre scegliere `/dev/ttyqf` come parte slave.

Una volta scelta una coppia di dispositivi pipe da usare, la parte master va allocata al kernel Linux, mentre la parte slave va assegnata a NOS, poiché il kernel si attiva per primo e il master deve essere aperto prima dello slave; occorre quindi allocare un indirizzo unico per NOS, se non si è già provveduto a farlo.

I pipe si configurano come se fossero dispositivi seriali, per cui per creare il collegamento slip dal kernel Linux, si possono usare i seguenti comandi:

```
# /sbin/slattach -s 38400 -p slip /dev/ptyqf &
# /sbin/ifconfig sl0 broadcast 44.255.255.255 pointopoint 44.70.248.67 /
mtu 1536 44.70.4.88
# /sbin/route add 44.70.248.67 sl0
# /sbin/route add -net 44.0.0.0 netmask 255.0.0.0 gw 44.70.248.67
```

In questo esempio al kernel Linux è stato assegnato l'indirizzo IP `44.70.4.88`, mentre NOS usa l'indirizzo `44.70.248.67`. Il comando *route* nell'ultima riga indica al kernel Linux di instradare, attraverso il collegamento slip creato dal comando *slattach*, tutti i datagrammi indirizzati verso amprnet. Normalmente questi comandi vanno messi nel file `/etc/rc.d/rc.inet2` immediatamente dopo tutti gli altri comandi di configurazione della rete, in modo che il collegamento slip sia creato alla partenza del sistema. Nota: non c'è alcun vantaggio nell'uso del comando *cslip* al posto di *slip*, anzi, con cslip si avverte un calo di prestazioni poiché, essendo un collegamento virtuale, il tempo impiegato per comprimere gli header è superiore di quello che viene impiegato per trasmettere i datagrammi non compressi.

Per configurare la parte NOS dall'altra parte del collegamento si può usare la seguente configurazione:

```
# si può chiamare l'interfaccia come si vuole, in questo caso la si è
# chiamata "linux" per comodità
attach asy ttyqf - slip linux 1024 1024 38400
route addprivate 44.70.4.88 linux
```

Questi comandi creano una porta slip chiamata ’linux’ attraverso la parte slave del pipe che lo collega al kernel di Linux, ed un comando di route per farla funzionare. Una volta fatto partire NOS, si deve essere in grado di eseguire ping e telnet da Linux a NOS e viceversa. Se ciò non si verificasse, controllare con attenzione soprattutto la corretta configurazione degli indirizzi e del pipe.

23. Sommario dei comandi Linux relativi al protocollo AX.25

Questa sezione riassume tutti i comandi specifici al protocollo AX.25

Comando	Pacchetto	Descrizione
mheard	ax25-tools	Mostra i nominativi AX.25 ascoltati di recente
ax25d	ax25-tools	Demone generico per AX.25, NET/ROM e ROSE
axctl	ax25-tools	Configura/termina le connessioni AX.25 attive
axparms	ax25-tools	Configura le interfacce AX.25
axspawn	ax25-tools	Consente il login automatico ad una macchina Linux
beacon	ax25-tools	Trasmette messaggi periodici su una porta AX.25
bpqparms	ax25-tools	Configura i dispositivi ethernet BPQ
mheardd	ax25-tools	Raccoglie informazioni relative all’attività packet
rxecho	ax25-tools	Instrada i pacchetti AX.25 tra le porte in modo trasparente
sethdlc	ax25-tools	Restituisce o imposta i parametri della porta per il packet radio modem Linux HDLC
smmixer	ax25-tools	Restituisce o imposta i parametri del Linux soundcard packet radio modem
smdiag	ax25-tools	utilità di diagnostica per il driver del Linux soundcard packet radio modem
kissattach	ax25-tools	Si connette ad un interfaccia KISS o 6PACK
kissnetd	ax25-tools	Crea una rete virtuale
kissparms	ax25-tools	Configura i TNC KISS
net2kiss	ax25-tools	Converte in driver di rete AX.25 in un flusso KISS su una pseudo-tty
mkiss	ax25-tools	Si connette ad un interfaccia multi KISS

Comando	Pacchetto	Descrizione
nodesave	ax25-tools	Salva le informazioni di routing NET/ROM
nrattach	ax25-tools	Attiva un'interfaccia NET/ROM
nrparms	ax25-tools	Configura l'interfaccia NET/ROM
nrsdrv	ax25-tools	convertitore seriale da KISS a NET/ROM
NET/ROMd	ax25-tools	Invia e riceve informazioni di routing NET/ROM
rsattach	ax25-tools	Attiva un'interfaccia ROSE
rsdwnlnk	ax25-tools	Comando per l'utente di uscita dalla rete ROSE
rsparms	ax25-tools	Configura l'interfaccia ROSE
rsuplnk	ax25-tools	Comando per l'utente di ingresso alla rete ROSE
ttylinkd	ax25-tools	Demone TTYlink per AX.25, NET/ROM, ROSE and IP
rip98d	ax25-tools	Invia e riceve messaggi di routing RIP98
ax25_call	ax25-tools	Effettua una connessione AX.25, NET/ROM, ROSE o TCP
NET/ROM_call	ax25-tools	Effettua una connessione AX.25, NET/ROM, ROSE o TCP
rose_call	ax25-tools	Effettua una connessione AX.25, NET/ROM, ROSE o TCP
tcp_call	ax25-tools	Effettua una connessione AX.25, NET/ROM, ROSE o TCP
yamcfg	ax25-tools	Configura i parametri del driver YAM
dmascc_cfg	ax25-tools	Configura i dispositivi dmascc
ax25ipd	ax25-apps	Incapsulatore di frame AX.25 in IP
ax25rtd	ax25-apps	Demone per il routing AX.25
ax25rtctl	ax25-apps	utilità di controllo del demone per il routing AX.25
call	ax25-apps	Effettua una connessione AX.25, NET/ROM o ROSE
listen	ax25-apps	Monitor del traffico AX.25
ax25mond	ax25-apps	Effettua un dump del traffico di rete AX.25 e mette a disposizione dei socket verso i quali i dati sono ritrasmessi
soundmodem	soundmodem	Driver per il modem soundcard
soundmodemconfig	soundmodem	Utilità di configurazione del modem soundcard

Comando	Pacchetto	Descrizione
aprsd	aprsd	Demone APRS
aprspass	aprsd	Generatore di passcode APRS
aprsdigi	aprsdigi	Digipeater APRS
aprsmon	aprsdigi	Monitor del traffico APRS AX.25 per JavAPRS

24. Dove trovare maggiori informazioni su....?

Poiché questo documento suppone che si abbia già maturato una certa esperienza col packet radio, nel caso fosse necessario ho raccolto un elenco di altre fonti di informazione che possono risultare utili.

24.1. Packet Radio

Informazioni generali sul packet radio si possono trovare su questi siti:

- American Radio Relay League (http://www.arrl.org/)
- Radio Amateur Teleprinter Society (http://www.rats.org/)
- Tucson Amateur Packet Radio Group (http://www.tapr.org/)

24.2. Documentazione sui protocolli

- AX.25, NET/ROM - Jonathon Naylor ha raccolto una serie di documenti riguardo i protocolli usati nel packet radio. Questa documentazione è stata raccolta nel file ax25-doc-1.0.tar.gz (ftp://ftp.hes.iki.fi/pub/ham/unix.linux/ax25/ax25-doc-1.0.tar.gz)

24.3. Documentazione sull'hardware

- Informazioni sulla *Scheda PI2* possono essere reperite presso l' Ottawa Packet Radio Group (http://hydra.carleton.ca/).
- Informazioni sull'hardware del *Baycom* si trovano sulla Web Page Baycom (http://www.baycom.de/).

24.4. Software Linux per radioamatori

John Ackermann mantiene un sito web con informazioni sulla configurazione di AX.25 su Linux presso http://www.febo.com/linux-ax25/index.html.

L'Hamsoft Linux Ham Radio Applications and Utilities Database si propone di mantenere una lista completa di applicazioni per Linux per radioamatori. La si può trovare su http://radio.linux.org.au (http://radio.linux.org.au/).

25. Discussioni riguardo i radioamatori e Linux

Si possono trovare diversi ambiti di discussione su i radioamatori e Linux, come ad esempio i newsgroup `comp.os.linux.*`, o la mailing list `linux-hams` presso `vger.kernel.org`. Altre aree includono la mailing list `tcp-group` su `ucsd.edu` (il sito principale per le discussioni sul TCP/IP radioamatoriale), e il canale irc `#linpeople`

Per iscriversi alla mailing list *linux-hams* occorre mandare un messaggio a: `majordomo@vger.kernel.org` con la riga:`subscribe linux-hams` nel corpo del messaggio. Il titolo del messaggio stesso viene ignorato.

I messaggi della mailing list *linux-hams* sono archiviati presso: http://hes.iki.fi/archive/linux-hams/ and http://web.gnu.walfield.org/mail-archive/linux-hams. Siete invitati a controllare questi archivi prima di mandare quesiti alla mailing list, poiché a molte domande è già stata data un'esauriente risposta.

Per iscriversi alla lista `tcp-group` occorre mandare una mail a: `listserver@ucsd.edu` con la riga `subscribe tcp-group` nel corpo del testo.

> **Nota:** Ricordate che la lista `tcp-group` serve principalmente per la discussione sull'uso di protocolli avanzati, come il tcp/ip in campo radioamatoriale. *Le domande esclusivamente su Linux non devono essere formulate in quest'area.*

26. Riconoscimenti

Terry Dawson è l'autore originario e manutentore di questo HOWTO. Nel 2001 Jeff Tranter gli è succeduto come manutentore per concedere a Terry più tempo per concentrarsi sullo sviluppo del software per l'AX.25.

Le seguenti persone hanno contribuito in un modo o nell'altro alla stesura di questo documento magari anche senza saperlo. Senza un particolare ordine (così come li ho trovati): Jonathon Naylor, Thomas Sailer, Joerg Reuter, Ron Atkinson, Alan Cox, Craig Small, John Tanner, Brandon Allbery, Hans Alblas, Klaus Kudielka, Carl Makin, John Ackermann, Riley Williams.

27. Feedback

Mi affido a voi lettori affinché questo HOWTO sia utile. Se avete suggerimenti, correzioni o commenti, vi prego di mandarmeli a tranter@pobox.com (mailto:tranter@pobox.com), in modo da vedere di includerli nella prossima versione.

Nel caso intendiate pubblicare questo documento su CD-ROM o su carta, una copia di cortesia sarebbe apprezzata; mandatemi un email per sapere il mio indirizzo postale. Prendete anche in considerazione l'idea di una donazione al

Linux Documentation Project per aiutare lo sviluppo della documentazione libera per Linux. Contattate l'LDP a feedback@linuxdoc.org (mailto:feedback@linuxdoc.org) per maggiori informazioni.

28. Regole di distribuzione