Linux AX25-HOWTO, Amateur Radio.

Terry Dawson, VK2KTJ, terry@perf.no.itg.telstra.com.au v1.5, 17 Ottobre 1997

Linux è forse l'unico sistema operativo al mondo che offra 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 di Nico Alberti IZ4APS@IK4MGV.PR.IEMR.ITA.EU albertin@usa.net

Indice

1 Note sulla traduzione italiana. **4**

2 Introduzione. **4**

2.1 Modifiche rispetto alla versione precedente. 4

2.2 Dove reperire nuove versioni di questo documento (in inglese). 5

2.3 Documenti correlati. 5

3 Linux e i protocolli per Packet Radio. **5**

3.1 Come tutto si combina assieme. 6

4 I componenti software per AX.25/NetRom/Rose. **7**

4.1 Dove reperire il kernel, i tools e i programmi di utilità. 7

4.1.1 I sorgenti del kernel: 7

4.1.2 Gli strumenti (tools) di configurazione di rete: 8

4.1.3 Le AX25 utility: 8

5 Installazione del software per AX.25/NetRom/Rose. **8**

5.1 La compilazione del kernel. 8

5.1.1 Una parola sui moduli del Kernel. 10

5.1.2 Le novità dei kernel 2.0.*+ModuleXX o 2.1.* 11

5.2 I programmi di configurazione della rete (nettools). 11

5.2.1 Un'aggiunta per correggere alcuni bug e avere il supporto per Rose. 11

5.2.2 Come compilare la versione standard di net-tools. 11

5.3 I programmi AX.25. 12

6 Due parole prima di partire sui nominativi, indirizzi e simili. **13**

6.1 Il significato di T1, T2, N2 eccetera. 13

6.2 Parametri configurabili durante il funzionamento. 14

7 Configurazione di una porta AX.25. 15

7.1 Creazione del device di rete AX.25. 15

7.1.1 Creazione di un device KISS 15

7.1.2 Creazione di un device Baycom. 17

7.1.3 Configurazione dei parametri dei accesso al canale AX.25. 18

7.1.4 Creazione di un device SoundModem. 19

7.1.5 Creazione di un device per scheda PI. 22

7.1.6 Creazione di un device PacketTwin. 22

7.1.7 Creazione di un generico device SCC. 23

7.1.8 Creazione di un device ethernet BPQ. 29

7.1.9 Configurazione del nodo BPQ per il colloqio con il supporto AX.25 di Linux. 30

7.2 Creazione del file `/etc/ax25/axports`. 30

7.3 Configurazione del routing AX.25. 31

8 Configurazione di un'interfaccia AX.25 per TCP/IP. 31

9 Configurazione di una porta NetRom. 32

9.1 Configurazione di `/etc/ax25/nrports` 32

9.2 Configurazione di `/etc/ax25/nrbroadcast` 33

9.3 Creazione del device di rete NetRom 33

9.4 Lancio del demone NetRom 34

9.5 Configurazione del routing NetRom. 34

10 Configurazione di un interfaccia NetRom per TCP/IP. 34

11 Configurazione di una porta Rose. 35

11.1 Configurazione di `/etc/ax25/rsports` 35

11.2 Creazione di un device di rete Rose. 36

11.3 Configurazione del routing Rose 36

12 Effettuazione di chiamate AX.25/NetRom/Rose. 36

13 Configurare Linux per accettare connessioni Packet. 37

13.1 Creazione del file `/etc/ax25/ax25d.conf`. 37

13.2 Un semplice esempio di file `ax25d.conf`. 40

13.3 Lanciare *ax25d* 41

14 Configurazione del programma *node*. 41

14.1 Creazione del file `/etc/ax25/node.conf`. 41

14.2 Creazione del file `/etc/ax25/node.perms` 43

14.3 Configurazione di *node* per funzionare da *ax25d* 44
14.4 Configurazione di *node* per funzionare da *inetd* 44

15 Configurazione di *axspawn* **45**
15.1 Creazione del file `/etc/ax25/axspawn.conf` 45

16 Configurazione di *pms* **46**
16.1 Creazione del file `/etc/ax25/pms.motd`. 47
16.2 Creazione del file `/etc/ax25/pms.info`. 47
16.3 Associazione di nominativi AX.25 con gli utenti di sistema. 47
16.4 Aggiunta di PMS in `/etc/ax25/ax25d.conf` 47
16.5 Test del PMS. 47

17 Configurazione dei programmi *user_call*. **47**

18 Configurazione dei comandi di uplink e downlink di Rose **48**
18.1 Configurazione del downlink Rose 48

19 Associare nominativi AX.25 con utenti Linux. **49**

20 I file in `/proc`. **49**

21 Programmazione di rete per AX.25, NetRom e Rose. **50**
21.1 Le famiglie degli indirizzi. 51
21.2 I file header. 51
21.3 Trattamento dei nominativi ed esempi. 51

22 Alcune semplici configurazioni. **51**
22.1 Piccola LAN Ethernet con macchina Linux che fa da router verso una LAN radio. 52
22.2 Configurazione del gateway di incapsulamento IPIP. 53
22.3 Configurazione del gateway di incapsulamento AXIP 57
22.3.1 Opzioni di configurazione di AXIP. 57
22.3.2 Un tipico esempio di `/etc/ax25/ax25ipd.conf`. 58
22.3.3 Uso di *ax25ipd* 59
22.3.4 Alcune note riguardo le route e i loro flag 60
22.4 Collegare NOS e Linux con un pipe device. 60

23 Dove trovare maggiori informazioni su.... ? **61**
23.1 Packet Radio 61
23.2 Documentazione sui protocolli 61
23.3 Documentazione sull'hardware 62

24 Discussioni riguardo i radioamatori e Linux. 62

25 Riconoscimenti. 62

26 Copyright. 63

1 Note sulla traduzione italiana.

Ho affrontato il lavoro di traduzione di questo HOWTO principalmente perché è la parte più interessante di documentazione sull'uso di Linux in ambiente radioamatoriale, e quindi credo che sia la prima fonte di informazioni tecniche per chi ne vuol sapere di più sulla comunicazione radio digitale con questo sistema operativo. Tuttavia nel documento vengono affrontati svariati problemi di configurazione di schede e protocolli per alcuni dei quali non ho l'enorme esperienza e capacità tecnica dell'autore (un eufemismo per dire che ne so veramente poco :-)). Pur avendo cercato di produrre il miglior lavoro di traduzione possibile compatibilmente con le mie capacità, è possibile che la traduzione di certi argomenti non sia chiara o che addirittura non sia all'altezza della situazione. In questi casi vi prego di farmi pervenire tutte le correzioni e i suggerimenti che riterrete opportuni, in modo da rendere questo lavoro più completo possibile.

Va inoltre notato che la grande mole dei dati forniti e il fatto che questo documento sia relativamente recente fa sì che la disposizione degli argomenti non sia forse la migliore, rendendo più complessa la comprensione dei concetti più impegnativi; l'autore, tuttavia, lavora continuamente su nuove versioni di questo documento che risultano sempre più complete ed esaurienti.

2 Introduzione.

Questo documento era originariamente un'appendice dell'HAM-HOWTO, ma è diventato troppo grande per essere gestito in quel modo. AX.25 HOWTO descrive come installare e configurare il supporto nativo AX.25, NetRom 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.

2.1 Modifiche rispetto alla versione precedente.

```
Aggiunte:
        la pagina Web di Joerg Reuters
        la sezione "Ulteriori Informazioni"
        la configurazione di ax25ipd.

Correzioni/Aggiornamenti:

        Modificati i pty con valori tali da evitare possibili conflitti.

        Aggiornate le versioni dei moduli e delle ax25-utils
```

```
Da Fare:
        Correggere la parte relativa alle schede SCC, perché
        probabilmente è sbagliata.

        Espandere la sezione dedicata alla programmazione.
```

2.2 Dove reperire nuove versioni di questo documento (in inglese).

La fonte migliore è da un archivio del Linux Documentation Project. In particolare il Linux Documentation Project gestisce un server Web nel quale è presente

l'AX25-HOWTO <http://sunsite.unc.edu/LDP/HOWTO/AX25-HOWTO.html>. Questo documento è presente inoltre in vari formati presso *sunsite.unc.edu* <ftp://sunsite.unc.edu/pub/Linux/docs/howto/>.

Si può sempre contattare l'autore, ma dato che passa le nuove versioni del documento direttamente al coordinatore del LDP, è probabile che non sia in grado di darvi versioni più aggiornate di quella presente nell'archivio.

2.3 Documenti correlati.

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

Sono le seguenti (in inglese):

The HAM-HOWTO <http://sunsite.unc.edu/LDP/HOWTO/HAM-HOWTO.html>,

The NET-3-HOWTO <http://sunsite.unc.edu/LDP/HOWTO/NET-3-HOWTO.html>,

The Ethernet-HOWTO <http://sunsite.unc.edu/LDP/HOWTO/Ethernet-HOWTO.html>,

e:

The Firewall-HOWTO <http://sunsite.unc.edu/LDP/HOWTO/Firewall-HOWTO.html>

Informazioni più generali possono essere reperite in altri *HOWTO Linux* <http://sunsite.unc.edu/LDP/HOWTO/>

3 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 per trasporto di altri protocolli come il TCP/IP e NetRom

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

Il protocollo NetRom rappresenta un tentativo di realizzare un protocollo di rete completo e usa AX.25 al livello più basso come protocollo dati. Presenta un livello 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 di AX.25 che viene usato a sua volta a livello dati, funzionando anche a livello 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 *RATS Web server* <http://www.rats.org/>.

Alan Cox è stato lo sviluppatore del primo supporto AX.25 per il kernel di Linux, che è stato successivamente preso in carico da Jonathon Naylor <g4klx@g4klx.demon.co.uk> che ha aggiunto anche il supporto per NetRom e Rose. Il supporto per il protocollo DAMA è stato sviluppato da Joerg, DL1BKE, jreuter@poboxes.com mentre il supporto per il Baycom e il SoundModem è stato aggiunto da Thomas Sailer, <sailer@ife.ee.ethz.ch>. Il supporto e lo sviluppo delle utility software per AX.25 è ora gestita da Terry Dawson, autore di queste note.

Linux supporta i TNC (Terminal Node Controllers) in modo KISS, l'Ottawa PI card, la Gracilis PacketTwin card e le altre schede SCC basate su Z8530 attraverso l'apposito driver, nonché il modem Baycom seriale e parallelo. Il nuovo driver soundmodem di Thomas Sailer permette l'utilizzo come modem della SoundBlaster e delle schede sonore basate sul chipset crystal.

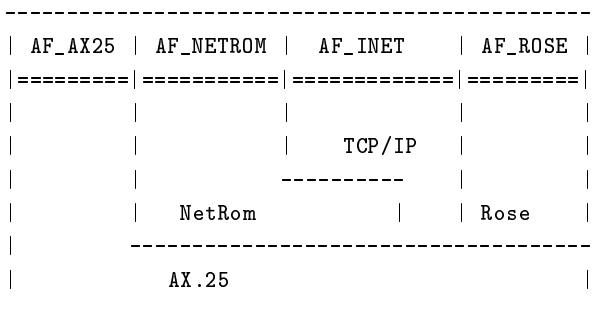
Il pacchetto User contiene un semplice PMS (Personal Message System), un beacon, un programma a linea di comando per effettuare connessioni, 'listen' un esempio su come catturare tutti i pacchetti AX.25 a livello di interfaccia e programmi per configurare il protocollo NetRom. E' incluso anche un programma tipo server AX.25 per gestire ed instradare connessioni AX.25 e un demone NetRom che svolge la maggior parte del lavoro di supporto per questo protocollo.

3.1 Come tutto si combina assieme.

Quella di Linux è un'implementazione di AX.25 piuttosto nuova. Sebbene somigli 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 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.

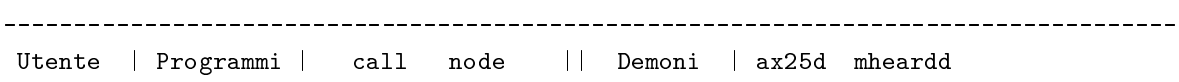
Diagramma semplificato dei livelli dei protocolli di rete

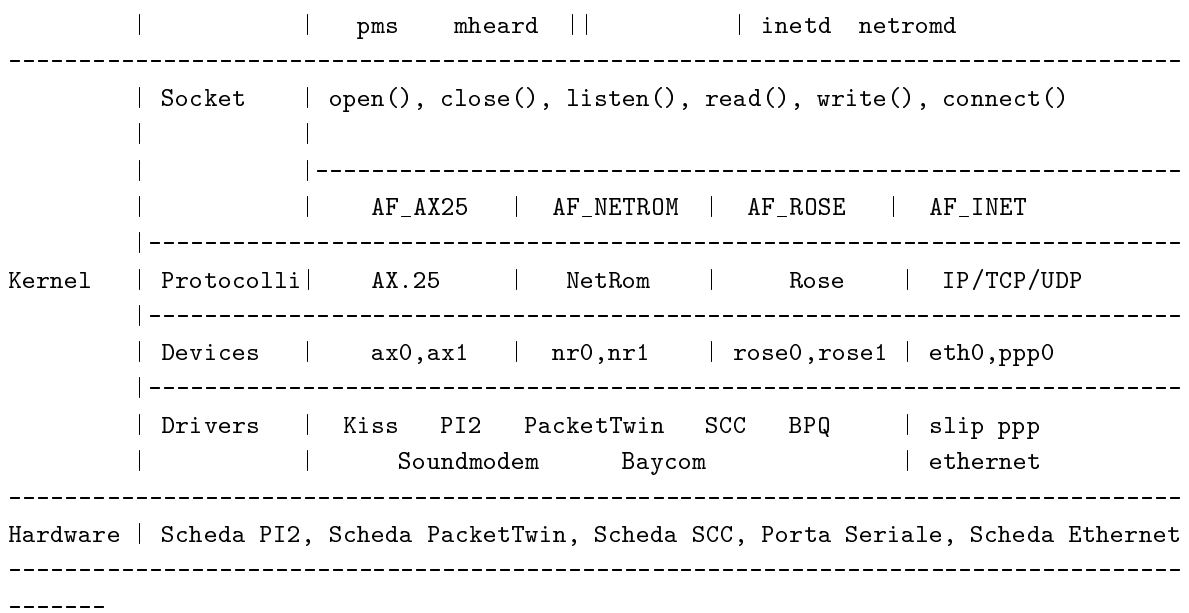


Questo diagramma illustra con chiarezza come NetRom, Rose e TCP/IP lavorino sopra l'AX.25, ma che ognuno di questi sia considerato come un diverso protocollo a livello applicazione.

I nomi 'AF_' sono semplicemente quelli dati alla '*Address Family*' di ognuno di questi, quando si scrivono programmi che li utilizzano. Si noti l'implicita dipendenza dalla configurazione della parte AX.25 presente in quelle di NetRom Rose o del TCP/IP.

Moduli software presenti nell'implementazione di rete di Linux





Questo diagramma è un po' più esteso 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 relativi ai vari protocolli, i device di rete del kernel e l'hardware.

Ogni cosa in questo diagramma dipende da ciò che è indicato sotto di lui, quindi in generale le parti da configurare devono essere fatte dal basso verso l'alto. Se, per esempio, si vuole far funzionare il programma *call* occorre configurare l'hardware, poi assicurarsi che il kernel abbia l'opportuno device driver, che sia stata creata l'opportuno device di rete e che il kernel includa il protocollo desiderato che a sua volta possa essere utilizzato dal programma *call*. La stesura di questo documento ricalca a grandi linee quest'ordine.

4 I componenti software per AX.25/NetRom/Rose.

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

Il kernel di Linux, dalla versione 2.0.0 in poi include i driver per AX.25, NetRom, Z8530 SCC, schede PI e i driver PacketTwin. Questi driver, nelle versioni 2.1.* del kernel sono stati significativamente migliorati; sfortunatamente i kernel con questa versione contengono codice non molto stabile e ciò non li rende adatti per un sistema che non sia di test. Per ovviare a questo problema Jonathon Naylor ha preparato un kit di aggiornamento in grado di garantire già sul kernel 2.0.28 le caratteristiche dei kernel 2.1.*. Questo kit è molto semplice da installare e permette di disporre di diversi miglioramenti non presenti nel kernel standard, come ad esempio il supporto per Rose.

4.1 Dove reperire il kernel, i tools e i programmi di utilità.

4.1.1 I sorgenti del kernel:

I sorgenti del kernel si possono trovare nel loro solito posto presso: **ftp.kernel.org**

```
/pub/linux/kernel/v2.0/linux-2.0.31.tar.gz
```

La versione corrente dell'aggiornamento per AX.25 è disponibile su: **ftp.pspt.fi**

```
/pub/linux/ham/ax25/ax25-module-14e.tar.gz
```

4.1.2 Gli strumenti (tools) di configurazione di rete:

L'ultima versione alfa degli strumenti standard di configurazione di rete AX.25 e NetRom per Linux si trova presso: **ftp.inka.de**

```
/pub/comp/Linux/networking/net-tools/net-tools-1.33.tar.gz
```

L'ultimo pacchetto ipfwadm si trova su: **ftp.xos.nl**

```
/pub/linux/ipfwadm/
```

4.1.3 Le AX25 utility:

Ci sono due diverse famiglie di AX25-utilities. Una è per i kernel `2.0.*` e l'altra funziona sia per per i kernel `2.1.*` che quelli `2.0.*+moduleXX`. Il numero di versione delle ax25-utils è quello del kernel più vecchio col quale queste funzionano, quindi occorre usare quello adatto al vostro kernel. Quelle che seguono sono combinazioni kernel-ax25-utils funzionanti. **bisogna** usare una di queste, perché altre non funzioneranno del tutto o in parte.

```
Kernel Linux             AX25 Utility
----------------------   ------------------------
linux-2.0.29             ax25-utils-2.0.12c.tar.gz **
linux-2.0.28+module12    ax25-utils-2.1.22b.tar.gz **
linux-2.0.30+module14c   ax25-utils-2.1.42a.tar.gz
linux-2.0.31+module14d   ax25-utils-2.1.42a.tar.gz
linux-2.1.22 ++          ax25-utils-2.1.22b.tar.gz
linux-2.1.42 ++          ax25-utils-2.1.42a.tar.gz
```

Nota: la serie delle `ax25-utils-2.0.*` (indicate sopra con '**') è ora obsoleta e non più supportata. Questo documento è relativo alle configurazioni indicate sopra. Dato che ci sono delle differenze tra una versione e l'altra, la maggior parte delle informazioni date in questo documento saranno relative all'ultima versione dei programmi.

Le AX.25 utility si trovano su: *ftp.pspt.fi* `<ftp://ftp.pspt.fi/pub/linux/ham/ax25/>`

o su: *sunsite.unc.edu* `<ftp://sunsite.unc.edu/pub/Linux/apps/ham/>`

5 Installazione del software per AX.25/NetRom/Rose.

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

5.1 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, che verranno trattate diffusamente qua sotto.

Il posto usuale in cui si decomprime il sorgente kernel è la directory `/usr/src`, nella quale viene creata una sottodirectory `linux`. Per fare ciò occorre essere loggati come `root` ed eseguire una serie di comandi come questi:

```
# mv linux linux.old
# cd /usr/src
# tar xvfz linux-2.0.31.tar.gz
# tar xvfz /pub/net/ax25/ax25-module-14e.tar.gz
# patch -p0 </usr/src/ax25-module-14/ax25-2.0.31-2.1.47-2.diff
# 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
```

Oppure potete usare:

```
# make config
```

Descriverò 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).

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  --->
    ...
    [*] TCP/IP networking
    [?] IP: forwarding/gatewaying
    ...
    [?] IP: tunneling
    ...
    [?] IP: Allow large windows (not recommended if <16Mb of memory)
    ...
    [*] Amateur Radio AX.25 Level 2
    [?] Amateur Radio NET/ROM
    [?] Amateur Radio X.25 PLP (Rose)
    ...
Network device support  --->
    [*] Network device support
    ...
    [*] Radio network interfaces
    [?] BAYCOM ser12 and par96 driver for AX.25
```

```
[?] Soundcard modem driver for AX.25
[?] Soundmodem support for Soundblaster and compatible cards
[?] Soundmodem support for WSS and Crystal cards
[?] Soundmodem support for 1200 baud AFSK modulation
[?] Soundmodem support for 4800 baud HAPN-1 modulation
[?] Soundmodem support for 9600 baud FSK G3RUH modulation
[?] Serial port KISS driver for AX.25
[?] BPQ Ethernet driver for AX.25
[?] Gracilis PackeTwin support for AX.25
[?] Ottawa PI and PI/2 support for AX.25
[?] Z8530 SCC KISS emulation 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:

```
# 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.

5.1.1 Una parola sui moduli del Kernel.

Suggerisco di **non** compilare alcun driver come modulo, poiché così facendo non si guadagna in altro che in complessità. Molti hanno incontrato problemi tentando di far funzionare le parti modularizzate, non perché il software abbia dei problemi, ma perché l'uso dei moduli rende più complessa la fase di installazione e configurazione del sistema.

Se comunque avete scelto di compilare qualche componente come moduli, occorre dare questi comandi:

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

per installare opportunamente i vostri moduli.

Occorrerà inoltre aggiungere/modificare le seguenti voci nel vostro file `/etc/conf.modules` in modo che il programma *kerneld* sappia come gestire i moduli in modo corretto.

```
alias net-pf-3     ax25
alias net-pf-6     netrom
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          netrom
```

```
alias pi0a          pi2
alias pt0a          pt
alias scc0          optoscc    (o uno degli altri drivers scc)
alias sm0           soundmodem
alias tunl0         newtunnel
alias char-major-4 serial
alias char-major-5 serial
alias char-major-6 lp
```

5.1.2 Le novità dei kernel 2.0.*+ModuleXX o 2.1.*

I kernel `2.1.*` presentano una versione migliorata di quasi tutti i protocolli e drivers. I miglioramenti più significativi sono:

modularizzazione

i protocolli e i driver sono stati modularizzati in modo che li possiate aggiungere o togliere con *insmod* e *rmmod*. Questo riduce il fabbisogno di memoria del kernel per moduli usati poco frequentemente e rende più semplici lo sviluppo e la ricerca di errori. Come detto, però, rende anche un po' più complessa la configurazione.

Tutti i driver sono driver di rete

tutti i device di rete come Baycom, SCC, PI, Packettwin eccetera, si presentano come normali interfacce di rete, come ad esempio il driver ethernet; non sono più come dei TNC kiss. Un nuovo programma di utilità chiamato *net2kiss* consente all'occorrenza di creare un'interfaccia kiss per questi device

Correzione di bug

Sono stati corretti molti problemi e aggiunte nuove funzionalità, tra cui il protocollo Rose è una delle più significative.

5.2 I programmi di configurazione della rete (nettools).

Una volta compilato il kernel, occorre compilare i nuovi programmi di configurazione della rete. Questi permettono di modificare la configurazione dei device di rete e di aggiungere voci di instradamento (route) alla tabella di instradamento (routing table).

La nuova versione alfa del pacchetto standard `net-tools` include il supporto per AX.25 e NetRom. Io l'ho testato e sembra funzionare correttamente.

5.2.1 Un'aggiunta per correggere alcuni bug e avere il supporto per Rose.

Il pacchetto standard net-tools-1.33.tar.gz ha alcuni piccoli bachi nel supporto dei protocolli AX.25 e NetRom. Ho dunque realizzato una piccola correzione per risolverli e aggiungere nel contempo il supporto per il protocollo Rose.

Potete scaricarla da: *zone.pspt.fi* `<ftp://zone.pspt.fi/pub/linux/ham/ax25/net-tools-1.33.rose.tjd.diff.gz>`.

5.2.2 Come compilare la versione standard di net-tools.

Non dimenticate di leggere il file `Release` e di seguire le istruzioni in esso contenute. I comandi per compilare sono:

```
# cd /usr/src
# tar xvfz net-tools-1.33.tar.gz
# zcat net-tools-1.33.rose.tjd.diff.gz | patch -p0
# cd net-tools-1.33
# make config
```

A questo punto vi verranno fatte una serie di domande per effettuare la configurazione simili a quelle trovate nel kernel. Assicuratevi di includere il supporto per tutti i protocolli che volete usare. Se non sapete cosa rispondere ad una particolare domanda, rispondete 'Y'.

Alla fine della compilazione occorre fare:

```
# make install
```

per installare il programma correttamente

Se siete intenzionati ad usare funzioni di IP firewall, vi occorrono i più recenti strumenti di amministrazione di firewall presenti nel pacchetto `ipfwadm`. Questo tool sostituisce il vecchio `ipfw` che non funziona coi nuovo kernel.

Ho compilato `ipfwadm` coi seguenti comandi:

```
# cd /usr/src
# tar xvfz ipfwadm-2.0beta2.tar.gz
# cd ipfwadm-2.0beta2
# make install
# cp ipfwadm.8 /usr/man/man8
# cp ipfw.4 /usr/man/man4
```

5.3 I programmi AX.25.

Una volta compilato in nuovo kernel e fatto il reboot con questo, occorre compilare i programmi per l'utente. Per compilarli e installarli occorre dare una sequenza di comandi simili a questi:

```
# cd /usr/src
# tar xvfz ax25-utils-2.1.42a.tar.gz
# cd ax25-utils-2.1.42a
# make config
# make
# make install
```

Questi file verranno installati come scelta predefinita sotto la directory `/usr` nei direttori `bin`, `sbin`, `etc` e `man`.

Se sulla vostra macchina non sono mai state installati i programmi per AX.25, dovete dare anche il comando:

```
# make installconf
```

per installare anche alcune configurazioni di esempio nella directory `/etc/ax25/` su cui provare a mettere le mani.

Se durante la compilazione vi escono messaggi 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.)
```

dovete controllare con la massima attenzione di avere il pacchetto *ncurses* installato correttamente sul vostro sistema. Lo script di configurazione cerca le ncurses nelle directory standard, ma alcune installazioni le installano in modo non corretto per cui non è in grado di trovarle.

6 Due parole prima di partire sui nominativi, indirizzi e simili.

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

Alcune implementazioni AX.25 come NOS e BPQ permettono però, di configurare lo stesso nominativo/ssid sulla stessa porta AX.25 e NetRom, ma 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 NetRom 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 NetRom usa il nominativo/ssid specificato nel proprio file di configurazione, ma esso viene usato solo quando si parla con un altra stazione NetRom; questo **non** è il nominativo/ssid che useranno gli utenti AX.25 che intendono usare il vostro 'nodo' NetRom. Ne parleremo più diffusamente tra un po'.
4. Il Rose usa di default il nominativo/ssid delle porte AX.25, eccetto il caso che 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.

6.1 Il significato di T1, T2, N2 eccetera.

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

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 di inattività della connessione
       |          | prima di essere interrotta.
-------------------------------------------------------------------
Window | MAXFRAME | Massimo numero di frame trasmessi senza
       |          | avere conferma di ricezione.
-------------------------------------------------------------------
```

6.2 Parametri configurabili durante il funzionamento.

I kernel `2.1.*` e `2.0.* +moduleXX` hanno la nuova proprietà di poter cambiare durante il funzionamento parametri che precedentemente non era possibile modificare. 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	Valori	Default
ip_default_mode	Modo IP di default	0=DG 1=VC	0
ax25_default_mode	Modo AX.25 di default	0=Normale 1=Esteso	0
backoff_type	Backoff	0=Lineare 1=Esponenziale	1
connect_mode	Modo Connesso	0=No 1=Sì	1
standard_window_size	Finestra Standard	1 <= N <= 7	2
extended_window_size	Finestra Estesa	1 <= N <= 63	32
t1_timeout	Valore di T1	1s <= N <= 30s	10s
t2_timeout	Valore di T2	1s <= N <= 20s	3s
t3_timeout	Valore di T3	0s <= N <= 3600s	300s
idle_timeout	Valore di Idle	0m <= N	20m
maximum_retry_count	N2	1 <= N <= 31	10
maximum_packet_length	Lunghezza frame AX.25	1 <= N <= 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/netrom/` è la seguente:

Nome File	Valori	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                      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
```

7 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/axport` e occorre che vi sia una voce per ognuna delle porte attive.

7.1 Creazione del device di rete AX.25.

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

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

Ogni device driver, quando viene lanciato, crea un device di rete.

7.1.1 Creazione di un device KISS

Opzioni di compilazione del Kernel:

```
General setup  --->
    [*] Networking support
Network device support  --->
```

```
[*] Network device support
...
[*] Radio network interfaces
[*] Serial port KISS driver for AX.25
```

Probabilmente la configurazione più comune è quella con un TNC KISS su 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 device KISS si usa il programma *kissattach*. Nella sua forma più semplice si può usare questo programma come segue:

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

Il comando *kissattach* crea in device di rete di tipo KISS. I device di questo tipo hanno nome '`ax[0-9]`'. La prima volta che si usa *kissattach* viene creato '`ax0`', la seconda '`ax1`' e così via. Ogni device KISS ha associato una porta seriale.

Il comando *kissparms* permette di modificare i parametri di un device KISS

In particolare nell'ultimo esempio viene creato un device KISS usando il device 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, nonchè con un valore di *ppersist* pari a 25

Per maggiori informazioni potete far riferimento alle pagine *man*

Configurazione di un TNC Dual Port L'utility *mkiss* inclusa in ax25-utils permette di usare entrambi i modem di un TNC dual port. La configurazione è piuttosto semplice e consiste nel prendere un singolo device seriale connesso ad un singolo TNC multiporta e facendolo apparire come diversi device connessi ognuno ad un TNC single port. Questa operazione va fatta **prima** di ogni configurazione della parte AX.25. I device sui quali si effettua la configurazione AX.25 sono interfacce pseudo-TTY, (`/dev/ttyq*`), e non gli effettivi device seriali.

I device pseudo-TTY creano una specie di tunnel detto pipe attraverso il quale possono parlarsi i programmi che devono colloquiare con i device 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 device.

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
# /usr/sbin/kissattach /dev/ttyq1 port2
```

creano due device pseudo-tty ognuna 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 device 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 porta seriale vera poiché viene usata dal programma *mkiss*.

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

-c

aggiunge 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 seriale.

-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 *syslog*.

7.1.2 Creazione di un device Baycom.

Opzioni di compilazione del Kernel:

```
Code maturity level options  --->
    [*] Prompt for development and/or incomplete code/drivers
General setup  --->
    [*] Networking support
Network device support  --->
    [*] Network device support
    ...
    [*] Radio network interfaces
    [*] BAYCOM ser12 and par96 driver for AX.25
```

Thomas Sailer, <sailer@ife.ee.ethz.ch>, a dispetto del luogo comune che non funzioni 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 *Baycom Web site* <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, alla configurazione, crea dei device di rete chiamati `bc0`, `bc1`, `bc2` ecc.

L'utility *sethdlc* permette di configurare i parametri del driver, cosa che può essere fatta anche nella linea di comando di *insmod* al caricamento del modulo di controllo del Baycom

Segue una piccola configurazione di esempio che:

Disabilita 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 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 molto consigliato, visto che l'utility *sethdlc* funziona senza problemi anche con un solo dispositivo connesso.

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

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

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

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

7.1.3 Configurazione dei parametri dei 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* .

Segue un esempio; per maggiori informazioni la pagina man relativa a *sethdlc* è il posto più indicato dove reperire le informazioni più dettagliate.

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

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

Si noti che i valori di temporizzazione sono in millisecondi.

Configurazione del Kernel AX.25 per l'uso con un modem Baycom Il driver Baycom crea dei device 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 device con un nominativo AX.25. L'utility *ifconfig* può essere utile allo scopo.

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

assegna al device Baycom `bc0` il nominativo AX.25 `VK2KTJ-15`. In alternativa sarebbe possibile usare il comando *axparms*, ma occorrerebbe sempre l'uso di *ifconfig* per attivare il device.

```
# 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 device. La voce nel file `axports` è associata col device di rete configurato per il nominativo ad esso legato. La voce nel file axports che ha il nominativo con il qual si è configurato il devide BayCom è quella che verrà usata per riferirlo.

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

7.1.4 Creazione di un device SoundModem.

Opzioni di compilazione del Kernel:

```
Code maturity level options  --->
    [*] Prompt for development and/or incomplete code/drivers
General setup  --->
    [*] Networking support
Network device support  --->
    [*] Network device support
    ...
    [*] Radio network interfaces
    [*] Soundcard modem driver for AX.25
    [?] Soundmodem support for Soundblaster and compatible cards
    [?] Soundmodem support for WSS and Crystal cards
    [?] Soundmodem support for 1200 baud AFSK modulation
    [?] Soundmodem support for 4800 baud HAPN-1 modulation
    [?] Soundmodem support for 9600 baud FSK G3RUH modulation
```

Thomas Sailer ha sviluppato un nuovo driver per il kernel che permette l'uso come modem della scheda audio che, 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 è 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. Questo sistema richiede un circuito addizionale, per poter far pilotare dalla scheda audio il PTT della radio; per sapere come realizzarlo si può andare alla *Thomas's SoundModem PTT circuit web page* <http://www.ife.ee.ethz.ch/~sailer/pcf/ptt_circ/ptt.html>. Sono possibili diverse opzioni che spaziano dal recuperare l'uscita audio della scheda, usare l'uscita di una porta parallela o di una porta midi. Esempi di circuito sono sul sito di Thomas Sailer.

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

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

Configurazione della scheda audio. Il driver SoundModem non inizializza la scheda audio, per cui occorre utilizzare l'apposito programma '*setcrystal*' presente nelle ax25-utils che può essere usato con schede basate sul chipset Crystal (con altre schede occorrerà usare altri programmi per inizializarle).

La sua sintassi è piuttosto semplice:

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

Quindi, 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
```

Mantre per una scheda WindowSoundSystem 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]` modifica l'indirizzo del sintetizzatore, mentre `[-c dma2]` serve per settare il secondo canale DMA per permettere operazioni in full duplex

Configurazione del driver SoundModem. Una volta configurata la scheda audio occorre configurare il driver, indicandogli l'indirizzo della scheda e che tipo di modem emulare.

L'utility *sethdlc* permette di configurare il driver con questi parametri o, se si ha solo una scheda audio installata, si può specificarne i parametri dalla linea di comando di *insmod* quando si carica il modulo SoundModem.

L'esempio seguente configura la SoundBlaster nel modo già visto precedentemente ed in modo da emulare un modem a 1200 bps:

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

Non è in effetti il modo migliore per effettuare le modifiche dato che, come detto già precedentemente, l'utility *sethdlc* funziona senza particolari problemi con una o più device.

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 SoundModem coi comandi:

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

o che il kernel sia stato compilato con incluso il driver.

Configurazione della scheda WindowsSoundSystem configurata come negli esempi precedenti e settata in modo da emulare un modem G3RUH 9600 compatible come device `sm0` usando la porta parallela all'indirizzo 0x378 per pilotare il Push-To-Talk della radio.

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

Configurazione del driver per supportare la SoundBlaster configurata come negli esempi precedenti e settata in modo da emulare un modem 4800 bps HAPN come device `sm1` usando la porta seriale all'indirizzo 0x2f8 per pilotare il Push-To-Talk della radio.

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

Configurazione del driver per supportare la SoundBlaster configurata come negli esempi precedenti e settata in modo da emulare un modem 1200 bps AFSK come device `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
```

Configurazione dei parametri dei accesso al canale 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* .

Seguono un paio di esempi; per maggiori informazioni la pagina man relativa a *sethdlc* è il posto più indicato dove reperire le informazioni più dettagliate.

Configurazione del device `sm0` con TxDelay di 100 mS, SlotTime di 50 mS, PPersist di 120 in modalità full duplex:

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

Si noti che i valori di temporizzazione sono in millisecondi.

Messa a punto del driver e del livello audio. Per ogni modem radio è molto importante che i livelli audio siano settati correttamente e il SoundModem non fa eccezione. Thomas Sailer ha per questo scritto alcne utility che facilitano questo compito, chiamate *smdiag* e *smmixer*.

smdiag

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

smmixer

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

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

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

Per lanciare l'utility *smmixer* per il device SoundModem `sm0` si usa il comando:

```
# smmixer -i sm0
```

Configurazione del Kernel AX.25 per l'uso con SoundModem Il driver Baycom crea dei device 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 device con un nominativo AX.25. l'utility *ifconfig* può essere utile allo scopo.

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

assegna al device SoundModem `sm0` il nominativo AX.25 `VK2KTJ-15`. In alternativa sarebbe possibile usare il comando *axparms*, ma occorrerebbe sempre l'uso di *ifconfig* per attivare il device.

```
# 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 device. La voce nel file `axports` è associato col device di rete configurato per il nominativo ad esso legato; verrà usata quella col nominativo assegnato al device SoundModem.

A questo punto si userà il nuovo device AX.25 come ogni altro; lo si potrà configurare per l'uso col TCP/IP, aggiungerlo a ax25d e usarci sopra NetRom o Rose a proprio piacimento.

7.1.5 Creazione di un device per scheda PI.

Opzione 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 device driver per schede PI crea dei device 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 device 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 di inserire una voce nel file /etc/ax25/axports col medesimo nominativo/ssid.

Il driver per la scheda PI è stato scritto da David Perry, <dp@hydra.carleton.edu>

7.1.6 Creazione di un device 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 device driver per la scheda PacketTwin crea i device 'pt[0-9][ab]'; la prima scheda PacketTwin localizzata nel calcolatore sarà indicata come 'pt0', la seconda 'pt1' e così via, mentre'a' and '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 possono usare i seguenti comandi per configurare i device 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 device, tutto ciò che occorre fare è confiurare una voce nel proprio file /etc/ax25/axports con il medesimo nominativo/ssid.

Il driver per schede PacketTwin è stato scritto da Craig Small VK2XLZ, <csmall@triode.apana.org.au>.

7.1.7 Creazione di un generico device 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, DL1BKE, `jreuter@poboxes.com` ha sviluppato il supporto generico per le schede basate sullo Z8530 SCC. 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.

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 dalla *pagina web di Joerg* `<http://www.rat.de/jr/>`

o da:

db0bm.automation.fh-aachen.de

```
/incoming/dl1bke/
```

o:

insl1.etec.uni-karlsruhe.de

```
/pub/hamradio/linux/z8530/
```

o:

ftp.ucsd.edu

```
/hamradio/packet/tcpip/linux
/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   2.0.*
z8530drv-utils-3.0.tar.gz    2.1.6 o superiore
```

I seguenti comandi mi hanno permesso di 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 nel kernel
# make install
```

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

Verrà anche creato nella directory `/dev` un un gruppo di device speciali chiamati `scc0-scc7`; questi saranno trattati in dettaglio più avanti e saranno i device 'KISS' da utilizzare.

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

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

Nel paccheto è fornita un'esauriente documentazione che va letta in caso di difficoltà. In particolare, maggiori informazioni si possono trovare in `doc/scc_eng.doc` or `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 `/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 sarà inizializzato nel modo indicato dal file di configurazione. Questa operazione va fatta prima di provare ad usare il driver, naturalmente.

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 con 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

serve per separare le sottosezioni. Non usa argomenti.

data_a

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 'A' dello z8530. L'argomento è un numero esadecimale, ad esempio 0x305

irq

specifica l'IRQ (l'interupt) usata 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 PA0HZP SCC

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 unici 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 (sic, n.d.t.).

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 PA0HZP SCC

```
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 DRSI SCC

```
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 forma che possa andare bene nel file di configurazione del driver z8530.

Per usare *gencfg* basta invocare il comando con gli stesi 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.

Configurazione del canale La sezione di configurazione del canale è quella nella quale si specificano tutte 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.

device

questa parola chiave deve apparire all'inizio della definizione della porta e specifica il nome del device 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` or `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 ms.

persist

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

slot

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

tail

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

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 ms.

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 ms.

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 ms.

softdcd

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

slip

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

Uso del driver. Per usare il driver basta semplicemente utilizzare i device `/dev/scc*` come se si avesse un device 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 esmpio, si può fare:

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

***sccstat* e *sccparam*.** Per la diagnosi di eventuali problemi, si può usare il programma *sccstat* per mostrare la configurazione corrente di un device 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 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
```

7.1.8 Creazione di un device 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 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 device di rete BPQ hanno nome '`bpq[0-9]`'. Il device '`bpq0`' è associato com il device '`eth0`', '`bpq1`' con '`eth1`' e così via.

La configurazione è abbastanza semplice. Prima di tutto occorre che il proprio kernel sia stato compilato per supportare la scheda Ethernet da utilizzare e che si sia già verificato il corretto funzionamento di quest'ultima con Linux. Per maggiori informazioni si può fare riferimento all'*Ethernet-HOWTO* `<Ethernet-HOWTO.html>`.

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

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

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

7.1.9 Configurazione del nodo BPQ per il colloqio 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
```

7.2 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 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 device AX.25 che si vuole utilizzare (KISS, SCC, PI, PT, Baycom o SoundModem che sia). I device di rete sono associati alle porte attraverso il nominativo/ssid, per questo esso deve essere univoco.

7.3 Configurazione del routing AX.25.

E' possibile configurare i 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 `VK2XLZ` sulla porta AX.25 chiamata `radio`.

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

E' 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 configurarlo; 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 occorre 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 device per una scheda PI, ma funziona altrettanto bene per ogni altro device 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 device 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.

9 Configurazione di una porta NetRom.

Il protocollo NetRom 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 NetRom, e l'altro quali porte AX.25 verranno usate per trasportare questo protocollo. Si possono configurare più porte NetRom, ognuna col proprio nominativo e alias, usando la stessa procedura.

9.1 Configurazione di `/etc/ax25/nrports`

Il primo file è `/etc/ax25/nrports`. Questo file descrive le porte NetRom pressapoco come `/etc/ax25/axports` descrive le porte AX.25. Ogni device NetRom che si vuole creare deve essere descritto all'interno di `/etc/ax25/nrports`. Normalmente una macchina Linux avrà configurato un unico device NetRom che usa un certo numero delle porte AX.25 definite, ma in alcune situazioni, come ad esempio nei BBS, si potrebbero volere diversi alias NetRom.

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 NetRom di questa porta. Nota: questa **non** è quell'indirizzo al quale si connettono gli utenti per avere un accesso 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 NetRom assegnato a questa porta.

paclen

è la grandezza massima dei frame netRom 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 NetRom, conosciuta dal resto della rete NetRom come 'LINUX:VK2KTJ-9'.

Questo programma viene usato da programmi come *call*

9.2 Configurazione di /etc/ax25/nrbroadcast

Questo file può contenere diverse voci; una per ogni porta AX.25 attraverso la quale si vuol far passare traffico NetRom ed è formattato nel seguente modo:

```
axport min_obs def_qual worst_qual verbose
```

Dove:

axport

è il nome dela porta ricavato dal file /etc/ax25/axports. Se per una porta non è presente una voce in /etc/ax25/nrbroadcasts significa che da questa non transiterà traffico NetRom.

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 NetRom completi, o solo di avvertimento per il nodo stesso.

Un esempio può essere il seguente:

```
radio    1       200      100         1
```

9.3 Creazione del device di rete NetRom

Una volta pronti i due file, occorre creare il device NetRom con un metodo molto simile a quello usato per creare i device 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 device di rete NetRom chiamati 'nr[0-9]'. Anche in questo caso, per primo verrà creato il device 'nr0', poi 'nr1' eccetera. Per creare il device di rete per la porta NetRom definita in precedenza si userà:

```
# nrattach netrom
```

Questo comando inizializzerà il device NetRom (nr0) chiamato netrom e configurato nel modo definito dai parametri del file /etc/ax25/nrports.

9.4 Lancio del demone NetRom

Il kernel di Linux gestisce tutto il protocollo NetRom, ad eccezione di alcune funzioni. Il demone NetRom gestisce le tavole di indirizzamento (routing tables) e genera la trasmissione del routing NetRom. Il demone NetRom 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 NetRom 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.

9.5 Configurazione del routing NetRom.

Volendo configurare degli instradamenti NetRom statici per degli host specifici si può usare il comando *nrparms*; ancora una volta si rimanda alle pagine *man* relative, non prima di dare un esempio che puo essere il seguente:

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

Questo comando abilita una route NetRom 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 NetRom adiacente con qualità `120`; questa voce sarà statica e quindi non sarà cancellata automaticamente.

10 Configurazione di un interfaccia NetRom per TCP/IP.

Il processo di configurazione di un interfaccia NetRom per TCP/IP è quasi del tutto identica 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'è nessun meccanismo che permetta alla propria macchina di sapere quale indirizzo NetRom usare per raggiungere un particolare host IP.

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

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

oppure una cosa simile alla seguente sequenza di comandi:

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

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

```
# route add 44.136.80.4 nr0
# arp -t netrom -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 NetRom di *qualità* e *obsolescence count* per la route.

11 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 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.

11.1 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  addresss  description
```

dove:

name
: è il nome con il quale ci si riferisce 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/rsprams -call VK2KTJ-10
```

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

11.2 Creazione di un device di rete Rose.

Una volta creato il file `/etc/ax25/rsports` si possono creare i device Rose allo stesso modo in cui sono stati creati i device AX.25. In questo caso si usa il comando *rsattach*, che crea i device 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 device Rose (`rose0`) configurato nel modo descritto nel file `/etc/ax25/rsports` per la porta chiamata '`rose`'.

11.3 Configurazione del routing Rose

Attualmente il protocollo Rose supporta solo l'instradamento statico. L'utiliity *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 mask per includere diverse destinazioni in un unica voce. 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 il modo più chiaro.

12 Effettuazione di chiamate AX.25/NetRom/Rose.

Una volta configurate le interfacce AX.25, NetRom e Rose, si è in grado di effettuare chiamate di prova.

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

Una semplice chiamata AX.25 sarà del tipo

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

Una semplice chiamata NetRom ad un nodo con alias `SUNBBS` sarà fatta con;

```
/usr/bin/call netrom SUNBBS
```

Una seplice chiamata Rose verso `HEARD` al nodo `5050882960` sarà fatta nel seguente modo:

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

Nota: occorre specificare a *call* su quale porta si vuole effettuare la chiamata, poiché lo stesso nodo di destinazione potrebbe essere raggiungibile su ogni porta configurata.

call è un programma di terminale a linea di comando per effettuare chiamate AX.25. Riconosce linee che iniziano con '~' come comandi. Il comando '~.' terminerà la connessione.

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

13 Configurare Linux per accettare connessioni Packet.

Linux è un sistema operativo molto potente è 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, NetRom 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 AX25 utilities). Alternativamente si potrebbe dare agli utenti un account e un nome di login, oppure potreste aver scritto un vostro programma, come un database customizzato, 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 connessioni TCP/IP su macchine unix. Il compito di *ax25d* è quello di raccogliere e gestire i tentativi di connesisone 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, NetRom e Rose, verrà descritto come configurarlo.

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

Questo è il file di configurazione per *ax25d*

Ad una prima occhiata può apparire un po' criptico, ma presto si nota come in pratica sia molto semplice, evitando 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>
<argomenti>
<peer2>    window T1 T2 T3 idle N2 <mode> <uid> <cmd> <cmd-name>
<argomenti>
parameters window T1 T2 T3 idle N2 <mode>
<peer3>    window T1 T2 T3 idle N2 <mode> <uid> <cmd> <cmd-name>
<argomenti>
   ...
default    window T1 T2 T3 idle N2 <mode> <uid> <cmd> <cmd-name>
<argomenti>
```

Dove:

#

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

<nome_porta>

è il nome della porta AX.25, NetRom o Rose come specificato nei file `/etc/ax25/axports`, `/etc/ax25/nrports` o `/etc/ax25/rsports`. Il nome della porta è circondato da parentesi quadre '`[]`' se è una porta AX.25, acute '<>' se NetRom, o graffe '{}' se Rose. Esiste una forma alternativa per questo campo, che consiste nel far precedere il nome della porta da '`nominativo/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 espressa in secondi, prima che il software AX.25 interrompa la sessione.

idle

è il valore di idle espresso in secondi.

N2

è il numero di ritrasmissioni consecutive che possono essre 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 NetRom 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, non vengono settati i modi.

<uid>

è l'identificativo col quale deve deve essere lanciato il programma che supporta la connessione.

<cmd>

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

<cmd-name>

è il testo che deve apparire in un *ps* come nome del comando lanciato (normalmente è lo stesso valore di <cmd>, però senza 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 del chiamante senza ssid e scritto in maiuscolo.

%u

Nominativo AX.25 del chiamante senza ssid e scritto in minuscolo.

%S

Nominativo AX.25 del chiamante con ssid e scritto in maiuscolo.

%s

Nominativo AX.25 del chiamante 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 nel formato visto sopra per ogni interfaccia AX.25, NetRom o Rose dalla quale si vogliano accettare connessioni.

Nel paragrafo ci sono due tipi speciali di linee, uno inizia con la stringa '`parameters`' è l'altro con la stringa '`default`' (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. 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 linea '`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 software del kernel ha alcuni valori di default che posono 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`' in modo da creare gruppi di configurazioni di default. E' importante notare che '`parameters`' non permette di settare i valori dei campi '`uid`' o '`command`'.

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

Ok, a questo punto si impone un esempio chiarificatore:

```
# ax25d.conf di VK2KTJ - 03/02/97
# 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

<netrom>
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 col nominativo '`VK2KTJ-0`' ascoltato sulla porta AX.25 chiamata '`radio`' sottostarà 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 copia di */usr/sbin/axspawn* lanciata in questo modo apparirà come *axspawn* in una lista *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.

Poi c'è una configurazione NetRom (si noti l'uso delle parentesi acute "< >"). Questa è più semplice: stabilisce che chiunque si connetta alla porta dal nome '`netrom`' 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 parentei 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 `ax25-utils`.

13.3 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.

14 Configurazione del programma *node.*

node è stato sviluppato da Tomi Manninen <`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, NetRom 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.

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

Il file `node.conf` è dove viene scritta la configurazione principale del nodo. E' 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 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 netrom

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

# Porta NetRom
# Questo è il nome della porta NetRom che verrà usata per
# le connessioni NetRom in uscita dal nodo.
NrPort          netrom

# 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
```

14.2 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.

utente

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.

metodo

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 NetRom, 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  descriztione
------  ------------------------------------------------------
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'
netrom  L'utente è connesso tramite NetRom
rose    L'utente è connesso tramite rose
*               L'utente è connesso in un modo qualunque.
```

porta

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.

permessi

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)onnessione AX25 consentita.
 4      (C)onnessione NetRom consentita.
```

```
8       (T)elnet verso host locali consentiti.
16      (T)elnet verso host amprnet (44.0.0.0) consentiti.
32      (T)elnet verso host non-locali e non-amprnet consentiti.
64      (C)onnessione AX25 consentita anche su porte occultate.
128     (C)onnessione Rose consentita.
```

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 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, NetRom, Local, Host e AMPR possono fare (C)onnect
# e (T)elnet a host locali e ampr, ma non ad altri indirizzi IP.
*       ax25    *       *       159
*       netrom  *       *       159
*       local   *       *       159
*       host    *       *       159
*       ampr    *       *       159
```

14.3 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`' sulla porta chiamata '`radio`' lanciando il programma *node*

14.4 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.

Occorre modificare due file.

In `/etc/services` occorre aggiungere:

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

a 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*.

15 Configurazione di *axspawn*

Il programma *axspawn* permette alle stazioni connesse via AX.25 di loggarsi 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 line 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* affinchè ne crei automaticamente uno.

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

E' 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
#
# uso di utente guest (ospite) se sopra si è scelto "no" o se tutto
# fallisce. Disabilita con "no"
guest     no
#
# nome o group id degli utenti creati automaticamente
group     ax25
#
# primo user id da usare
first_uid 2001
#
# user id massimo
```

```
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 linea 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 deli 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 login shell 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.

16 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 <`dcb@vectorbd.com`>. Allo stadio attuale è ancora molto semplice, visto che supporta solo l'invio di messaggi solo al gestore del sistema e di ricavare alcune informazioni sul sistema, ma Dave è al lavoro per espandere le sue funzionalità e renderlo più utile.

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 il PMS.

16.1 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.

16.2 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>`.

16.3 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.

16.4 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 la possibilità di fornire al programma pms attraverso la linea di comando la possibilità di gestire in diversi modi la fine di una linea di testo. AX.25 e NetRom, 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
```

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

16.5 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 correttie.

17 Configurazione dei programmi *user_call*.

I programmi '*user_call*' sono in realtà: *ax25_call*,'*rose_call*' e *netrom_call*. Sono molto semplici e sono 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*.

A causa della loro semplicità, non trattano in alcun modo i dati che gestiscono, quindi ad esempio il problema della 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 *ax25d*.

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 utility '*netrom_call*' e '*rose_call*' funzionano in modo simile. Un amatore ha usato questa utility 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 attraverso di essa di fosse in una rete locale dove *ax25d* fa da proxy per la connessione alla macchina remota.

18 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*.

18.1 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 `HEARD-0`, e di passare le altre chiamate al comando *rsdwnlink*, assumendo che siano utenti AX.25. Una tipica configurazione può dunque essre:

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

con questa configurazione, ogni connessione al nodo Linux con un nominativo di destinazione che non sia specificato esplicitamente, sarà convertita in una connessione AX.25 dalla porta chiamata `4800` usando come digipeater `VK2KTJ-5`.

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*

19 Associare nominativi AX.25 con utenti Linux.

Ci sono diverse situazioni in cui è desiderabile associare un nominativo con un account utente di Linux, ad esempio quando diversi radioamatori condividono la stessa machina 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.

20 I file in `/proc`.

La directory `/proc` presenta diversi file che sono in relazione con il kernel AX.25 e NetRom. Questi sono di solito utilizzati dalle utility 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 la loro descrizione.

/proc/net/arp

contiene le lista delle mappatture ARP (Address Resolution Protocol)di indirizzi IP in indirizzi di protocollo a livello MAC. Questi possono essere AX.25, ethernet o qualche altro protocollo a livello MAC.

/proc/net/ax25

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

/proc/net/ax25_bpqether

contiene le mappature dei nominativi di tipo AX.25 su ethernet BPQ.

/proc/net/ax25_calls

contiene le mappature degli userid Linux nei confronti dei nominativi configurate dal 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 NetRom aperti. Questi possono essere in attesa di una connessione, oppure collegamenti attivi.

/proc/net/nr_neigh

contiene informazioni riguardo i nodi NetRom vicini (NetRom neighbours) conosciuti dal programma.

/proc/net/nr_nodes

contiene informazioni riguardo i nodi NetRom 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 stabilite.

21 Programmazione di rete per AX.25, NetRom 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, NetRom e Rose all'interno dei vostri programmi.

21.1 Le famiglie degli indirizzi.

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

I nomi delle famiglie degli indirizzi per AX.25, NetRom e Rose sono rispettivamente `AF_AX25`, `AF_NETROM` e `AF_ROSE`.

21.2 I file header.

Occorre sempre includere i file header '`ax25.h`', nonchè '`netrom.h`' o '`rose.h`' se avete a che fare con questi protocolli. Le impostazioni iniziali saranno simili alle seguenti:

Per AX.25:

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

Per NetRom:

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

Per Rose:

```
#include <ax25.h>
#include <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 utility *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.

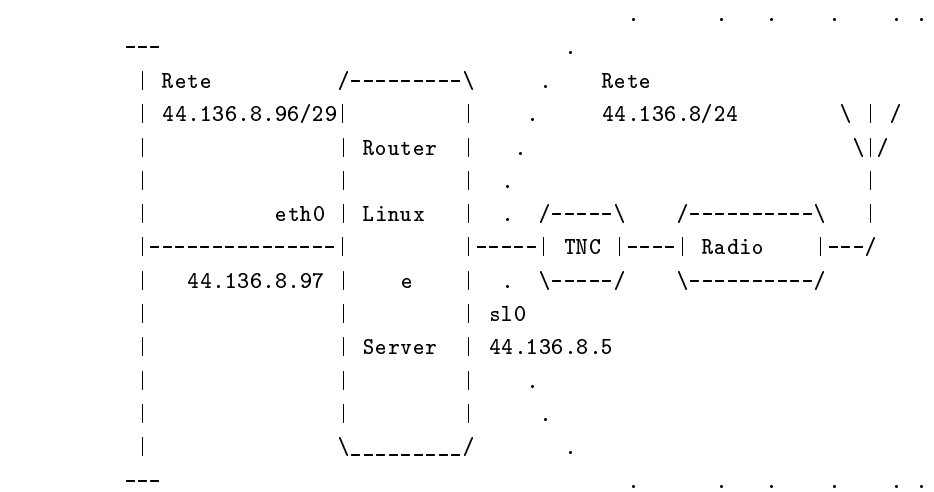
Gli esempi sono semplici a sufficienza da non creare confusione. In caso di dubbi è buona cosa rivolgerli alla mailing list `linux-hams`, dove senz'altro ci sarà qualcuno che vi darà una mano.

22 Alcune semplici configurazioni.

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 di quella rete alla rete radio. Questa 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 numeratore IP per la vostra zona vi assisterà in questo 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:




```
#!/bin/sh
# /etc/rc.net
# Questa configurazione rende disponibile una porta KISS AX.25
# e un device 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 "    Netrom: "
nrattach -i 44.136.8.5 netrom
```

```
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

```
# name  nominativo      alias   paclen  descrizione
netrom  VK2KTJ-9        LINUX   235     Linux Switch Port
```

/etc/ax25/nrbroadcast

```
# ax25_name     min_obs def_qual        worst_qual      verbose
4800            1       120             10              1
```

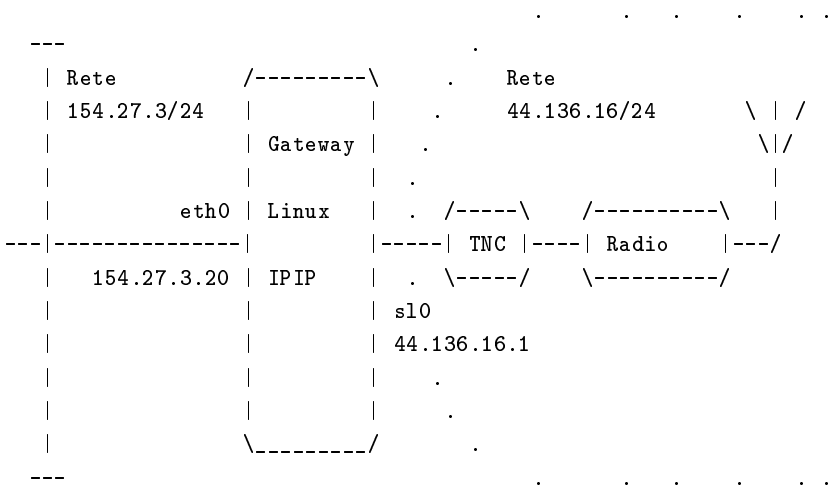
- Occorre che IP_FORWARDING sia abilitato nel vostro kernel.
- Quando necessario fare riferimento ai file di configurazione AX.25 presentati nelle sezioni precedenti.
- Ho scelto di usare un indirizzo IP per la mia porta radio che non fa parte del blocco usato per la mia rete. Ciò non è obbligatorio, per quella porta si sarebbe potuto tranquillamente usare anche `44.136.8.97`
- `44.136.8.68` è il mio gateway locale di incapsulamento IPIP, 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 quallo Ethernet.
- Sulla macchina router vengono fatti girare smail, http, ftp ed altri demoni in modo che questi servizi vengano resi disponibili ad entrambe le reti.
- 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.

Linux oggi è utilizzato spessissimo per i gateway di incapsulamento TCP/IP in tutto il mondo. Il driver per il tunnelling supporta route di incapsulamento multiple rendendo obsoleto il vecchio demone *ipip*. Una configurazione tipica è la seguente:



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 device Ethernet e sfrutta il tunnel
# driver del kernel per effettuare l'incapsulamento/deincapsulamento
# IPIP
#
echo "/etc/rc.net"
echo "  Configurazione:"
#
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 tunnel driver 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 ciascun'interfaccia.
- Si **possono** configurare due device 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 generate 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*. Settando opportunamente questo parametro si migliorano le prestazioni del collegamento radio.

Ecco 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 "# default the rest of amprnet via mirrorshades.ucsd.edu"
echo "route add -net 44.0.0.0 gw 128.54.16.18 netmask 255.0.0.0 dev
tunl0"
echo "#"
echo "# fine"
```

22.3 Configurazione del gateway di incapsulamento AXIP

Molti gateway radioamatoriali per Internet incapsulano l'AX.25, NetRom 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 geteway 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 frames KISS incapsulati in modo che li trasmetta, mentre nel modo "digipeater" si comporta, appunto, come un digipeater AX.25. Tra questi due modi 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 spegate 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 vuole 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 configurazioen 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 (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 future (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 device
#
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 fuziona ....
#
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 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     AXIP port
#
```

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

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

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 inoltrare i propri pacchetti incapsulati. 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'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 pipe device.

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 device pipe.

Poiché sia Linux che NOS supportano il protocollo slip, è possibile connettere i due creando un link di tipo slip. E' possibile realizzare questo collegamento realizzando in collegamento tra due porte seriali della stessa macchina collegate in loopback con un cavo, ma questa realizzazione risulterebbe lenta e costosa. Linux, al contrario, fornisce una funzionalità tipica dei sistemi Unix chiamata 'pipe'. I pipe sono degli 'pseudo-device' che sono visti dai programmi come normali device tty, ma che in realtà fungono da collegamento verso un altro pipe. Per usare i pipe 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 device 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 device pipe da usare, la parte master va allocata al kernel Linux, mentre la parte slave va assegnata a NOS; occorre quindi allocare un indirizzo unico per NOS, se non si è già provveduto a farlo.

I pipe si configurano come se fossero device 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 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.

23.1 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/>

23.2 Documentazione sui protocolli

AX.25, NetRom - 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.pspt.fi/pub/ham/linux/ax25/ax25-doc-1.0.tar.gz>

23.3 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 alla *Baycom Web Page* <http://www.baycom.de/>.

24 Discussioni riguardo i radioamatori e Linux.

Si possono trovare diversi luoghi di discussione riguardanti i radioamatori e Linux, come ad esempio i newsgroup `comp.os.linux.*`, o la mailing list `HAMS` presso `vger.rutgers.edu` (il luogo di discussione principale per l'uso del TCP/IP tra i radioamatori), oppure anche il canale `#linpeople` della rete irc `linuxnet`.

Per iscriversi alla mailing list **linux-hams** occorre mandare un messaggio a:

```
Majordomo@vger.rutgers.edu
```

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:

zone.pspt.fi <http://zone.pspt.fi/archive/linux-hams/> e *zone.oh7rba.ampr.org* <http://zone.oh7rba.ampr.org/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 al `tcp-group` mandare una mail a:

```
listserver@ucsd.edu
```

con la riga:

```
subscribe tcp-group
```

nel testo del messaggio.

Nota: Ricordate che il `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.*

25 Riconoscimenti.

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

26 Copyright.

AX25-HOWTO, informazioni sull'installazione e la configurazione di alcuni dei più importanti pacchetti software per il supporto AX.25 sotto Linux. Copyright (c) 1996 Terry Dawson. Traduzione italiana di Nico Alberti

Questo programma è free software; la distribuzione e l'utilizzo di questo documento sono vincolati dai termini indicati nella versione 2 (o da qualunque versione successiva) della GNU General Public License pubblicata dalla Free Software Foundation.

Questo documento viene distribuito con l'intento di essere utile, ma SENZA ALCUNA GARANZIA, compreso quella implicita di FUNZIONALITA' o di UTILITA' PER UN PARTICOLARE UTILIZZO. Vedere la GNU General Public License per ulteriori dettagli.

Assieme a questo documento dovreste aver ricevuto una copia della GNU General Public License; se ciò non fosse vi preghiamo di scrivere alla

Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

The AX25-HOWTO, information on how to install and configure some of the more important packages providing AX25 support for Linux. Copyright (c) 1996 Terry Dawson.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the:

Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.