Linux NET-3-HOWTO, Linux Networking.

Terry Dawson (autore principale), VK2KTJ, Alessandro Rubini (manutentore). Traduzione: Alessandro Rubini rubini@linux.it, Riccardo Fabris skizzo@mail.seta.it. v1.4, Agosto 1998

Il sistema operativo Linux vanta un supporto per la rete a livello kernel scritto quasi completamente da zero. Le elevate prestazioni dell'implementazione tcp/ip delle versioni del kernel più recenti lo rendono un'alternativa significativa anche ai migliori fra i sistemi operativi similari. Questo documento si propone di descrivere come installare e configurare il software di rete di Linux e gli strumenti associati.

Indice

1 Cambiamenti dalla versione 1.3 (Aprile 1998) 4

2 Introduzione. 4

2.1 Commenti 5

3 Come usare questo documento (HOWTO sul NET-3-HOWTO ?). 5

3.1 Convenzioni usate in questo documento 6

4 Informazioni generali sul networking di Linux. 6

4.1 Una breve storia dello sviluppo del supporto di rete nel kernel Linux. 6

4.2 Dove trovare altre informazioni riguardo al Networking in Linux 8

4.3 Dove trovare informazioni di rete non correlate a Linux 9

5 Informazioni generali sulla configurazione di rete. 9

5.1 Da dove devo partire? 10

5.1.1 Sorgenti aggiornati del kernel. 10

5.1.2 Gli strumenti di rete aggiornati 10

5.1.3 Programmi Applicativi di Rete. 11

5.1.4 Indirizzi. 12

5.2 Dove bisogna mettere i comandi di configurazione? 13

5.3 Creazione delle interfacce di rete. 14

5.4 Configurazione delle interfacce di rete. 15

5.5 Configurazione del risolutore dei nomi (name resolver). 16

5.5.1 Cosa compone un nome? 16

5.5.2 Quali informazioni sono necessarie. 17

5.5.3 /etc/resolv.conf 17

5.5.4 /etc/host.conf 18

5.5.5 /etc/hosts 18

5.5.6 Attivare un name server 19
5.6 Configurazione dell'interfaccia di loopback. 19
5.7 Routing. 19
5.7.1 Allora a cosa serve il programma routed? 21
5.8 Configurazione dei servizi di rete e dei programmi server. 23
5.8.1 /etc/services 24
5.8.2 /etc/inetd.conf 28
5.9 Altri file di configurazione relativi alla rete. 31
5.9.1 /etc/protocols 31
5.9.2 `/etc/networks` 32
5.10 Sicurezza di rete e controllo degli accessi. 32
5.10.1 /etc/ftpusers 33
5.10.2 /etc/securetty 33
5.10.3 Il meccanismo di controllo degli accessi *tcpd*. 33
5.10.4 /etc/hosts.equiv 35
5.10.5 Come configurare correttamente il server *ftp*. 35
5.10.6 I firewall di rete. 35
5.10.7 Altri suggerimenti. 35

6 Informazioni specifiche ad IP ed Ethernet. 35
6.1 Ethernet 36
6.2 EQL - equalizzazione del traffico su linea multipla 36
6.3 IP Accounting (per Linux-2.0) 37
6.4 IP Accounting (per Linux-2.2) 39
6.5 IP Aliasing 39
6.6 IP Firewall (per Linux-2.0) 40
6.7 IP Firewall (per Linux-2.2) 42
6.8 Incapsulazione IPIP 42
6.8.1 Una configurazione di rete a "tunnel" 43
6.8.2 Una configurazione di calcolatore con tunnel 44
6.9 Mascheramento IP (IP Masquerade) per Linux-2.0 46
6.10 IP Transparent Proxy 47
6.11 IPv6 48
6.12 IP mobile 48
6.13 Multicast 48
6.14 NAT - Network Address Translation 48
6.15 Il 'Traffic Shaper' - Come cambiare l'ampiezza di banda assegnata 49

6.16 Instradamento (routing) in Linux-2.2 . . . 49
6.17 ISDN . . . 49
6.18 PLIP per Linux-2.0 . . . 51
6.19 PLIP per Linux-2.2 . . . 52
6.20 PPP . . . 53
6.20.1 Gestire una connessione permanente ad internet con *pppd*. . . . 53
6.21 SLIP client . . . 53
6.21.1 dip . . . 54
6.21.2 slattach . . . 54
6.21.3 Come scegliere se usare l'uno o l'altro? . . . 54
6.21.4 SLIP server statico con linea telefonica e DIP. . . . 55
6.21.5 SLIP server dinamico con linea telefonica e DIP. . . . 55
6.21.6 Uso di DIP. . . . 55
6.21.7 Connessione SLIP permanente con linea dedicata e slattach. . . . 58
6.22 Server SLIP. . . . 59
6.22.1 Server SLIP usando *sliplogin*. . . . 59
6.22.2 Server Slip che usa *dip*. . . . 62
6.22.3 Server SLIP che usa il pacchetto *dSLIP*. . . . 64

7 Altre Tecnologie di Rete 65

7.1 ARCNet . . . 65
7.2 Appletalk (`AF_APPLETALK`) . . . 65
7.2.1 Configurazione del software Appletalk. . . . 66
7.2.2 Esportare un disco tramite Appletalk. . . . 66
7.2.3 Condividere le stampanti Linux usando Appletalk. . . . 67
7.2.4 Far partire il software appletalk. . . . 67
7.2.5 Provare il software appletalk. . . . 67
7.2.6 Alcune attenzioni con il software appletalk. . . . 67
7.2.7 Ulteriori informazioni. . . . 68
7.3 ATM . . . 68
7.4 AX25 (AF_AX25) . . . 68
7.5 DECNet . . . 68
7.6 FDDI . . . 69
7.7 Frame Relay . . . 69
7.8 IPX (`AF_IPX`) . . . 73
7.9 NetRom (`AF_NETROM`) . . . 73
7.10 Il protocollo Rose (`AF_ROSE`) . . . 73

7.11 Supporto SAMBA - 'NetBEUI', 'NetBios'. 74
7.12 Supporto STRIP (Starmode Radio IP) 74
7.13 Token Ring 74
7.14 X.25 75
7.15 Scheda WaveLan 75

8 Cavi e cablaggio 75

8.1 Cavo Seriale NULL Modem 76
8.2 Cavo per la porta parallela (cavo PLIP) 76
8.3 Cablaggio ethernet 10base2 (coassiale sottile) 77
8.4 Cavo Ethernet TP (Twisted Pair) 77

9 Glossario dei termini usati in questo documento. 77

10 Linux per un provider? 79

11 Ringraziamenti 79

12 Copyright. 79

1 Cambiamenti dalla versione 1.3 (Aprile 1998)

```
Aggiunte:
        Traffic Shaper (limitatore di banda)
        Plip per i nuovi kernel

Correzioni/Aggiornamenti:
        Indirizzo del manutentore del netkit
        Riviste le descrizioni dei nomi di dominio (domain name)
        Riordino generale delle sezioni
        Sono indicate le differenze tra 2.0 e 2.2, sebbene alcune
        informazioni sul 2.2 siano mancanti
        Corretti molti riferimenti a documenti esterni

Da aggiungere:
        Descrivere i nuovi algoritmi di routing
        Aggiungere le opzioni di compilazione del kernel per IPV6
        Descrivere le voci in  /proc/sys/net/*
        Parlare della periferica "WanRouter"
        Descrivere i nuovi comandi firewall dei 2.2
```

2 Introduzione.

Il documento NET-FAQ originale fu scritto da Matt Welsh e da me, al fine di rispondere alle domande poste più frequentemente a proposito del supporto di rete per Linux, quando il "Linux Documentation Project" non era ancora formalmente partito. Questo documento copriva le versioni di sviluppo iniziali del Networking per il kernel Linux. Il documento NET-2-HOWTO ha rimpiazzato la NET-FAQ ed è stato uno dei primi documenti HOWTO dell'LDP, coprendo quella che è stata chiamata la versione 2, e successivamente versione 3, del software di rete per il kernel Linux. Questo documento a sua volta rimpiazza il NET-2-HOWTO e si occupa solo della versione 3 del software.

Le versioni precedenti di questo documento sono diventate molto ingombranti a causa dell'enorme mole di materiale esistente sull'argomento. Per circoscrivere questo problema sono stati scritti un certo numero di HOWTO a proposito di specifici argomenti di networking. Questo documento fornisce i puntatori a tali documenti dove necessario e copre le aree non ancora coperte da altri documenti.

Nell'Aprile del 1998 Terry ha lasciato il suo ruolo di manutentore a causa dei suoi impegni di lavoro. Io, Alessandro Rubini, sono nuovo in questo ruolo ma cercherò di fare del mio meglio per riuscirci.

2.1 Commenti

Apprezzo sempre i commenti e in particolare i contributi di valore. Mandate pure tutti i commenti e i contributi al mio indirizzo *rubini@linux.it* `<mailto:rubini@linux.it>`.

3 Come usare questo documento (HOWTO sul NET-3-HOWTO ?).

Questo documento è organizzato con metodo top-down. Le prime sezioni presentano materiale informativo che può essere saltato da chi non sia interessato; successivamente appare materiale generico che occorre essere sicuri di aver capito prima di procedere oltre; il resto del documento è composto da tre sezioni principali che si occupano di tecnologie specifiche: informazioni circa Ethernet e IP, tecnologie di largo impiego nell'hardware dei PC e tecnologie di uso ristretto.

Si suggerisce quindi di percorrere il documento nel seguente modo:

Leggere le sezioni generiche

Queste sezioni si applicano a tutte, o quasi tutte, le tecnologie descritte più avanti, ed è quindi molto importante comprenderle. Del resto ci si attende che molti dei lettori abbiano già confidenza con tale materia.

Osservare la propria rete

Occorre sapere come è progettata la propria rete, o come lo sarà. Occorre anche sapere esattamente quali tipi di tecnologie hardware verranno utilizzate.

Leggere la sezione "Ethernet e IP" se si è connessi direttamente ad una rete locale o a Internet

Questa sezione descrive una configurazione Ethernet di base e le varie funzionalità offerte da Linux per le reti IP, come il firewalling, l'instradamento di rete avanzato e così via.

Leggere la sezione successiva se si è interessati a reti locali a basso costo o a connessioni telefoniche

La sezione descrive PLIP, PPP, SLIP e ISDN, che sono le tecnologie maggiormente diffuse nell'ambito PC.

Leggere le parti specifiche alle proprie esigenze

Se si ha bisogno di qualcosa di diverso da informazioni su IP e/o hardware comunemente diffuso, la sezione finale si occupa dei dettagli specifici dei protocolli non-IP e di hardware di comunicazione particolare.

Configurare la rete

Bisogna provare a configurare la rete in pratica e prendere scrupolosamente nota di ogni problema incontrato.

Cercare ulteriore aiuto se necessario

Se si incontrano problemi che questo documento non aiuta a risolvere, si legga la sezione relativa a dove trovare aiuto e dove far presenti gli errori.

Divertirsi!

Il networking è divertente, bisogna saperlo apprezzare!

3.1 Convenzioni usate in questo documento

Non verranno usate convenzioni speciali, si faccia però una particolare attenzione al modo in cui sono espressi i comandi. Seguendo il classico stile di documentazione Unix, ogni comando da digitare sulla linea di comando di una shell è preceduto da un prompt. Questo HOWTO adopera "`user%`" quale prompt per i comandi che non richiedono privilegi di superutente, e "`root#`" come prompt per i comandi che necessitano di essere lanciati da 'root'. La scelta di usare "`root#`" anzichún semplice "`#`" è volta a prevenire confusioni con esempi tratti da script di shell, dove il segno "#" è usato per definire una linea di commento.

Quando vengono indicate opzioni di compilazione del kernel, esse sono rappresentate nel formato usato da *menuconfig*. Dovrebbero risultare comprensibili anche a chi (come me) non ha confidenza con *menuconfig*. Nel caso ci siano dei dubbi con l'annidamento delle opzioni, lanciare una volta il programma suddetto non può che essere d'aiuto.

Da notare che ogni link ad altri HOWTO è locale per agevolare la navigazione all'interno della propria copia locale dei documenti LDP, ove si stia usando la versione html di questo documento. In caso risulti mancante qualche documento, si sappia che qualunque HOWTO può essere recuperato da `sunsite.unc.edu` (nella directory `/pub/Linux/HOWTO`) e dai suoi numerosi mirror.

4 Informazioni generali sul networking di Linux.

4.1 Una breve storia dello sviluppo del supporto di rete nel kernel Linux.

Sviluppare partendo da zero un'implementazione della pila di protocolli tcp/ip dalle prestazioni buone quanto le altre implementazioni esistenti non è stato un lavoro semplice. La decisione di non basarsi su una delle implementazioni esistenti è stata presa in un momento in cui sembrava che le implementazioni esistenti sarebbero finite sotto una licenza più restrittiva a causa della sentenza di U.S.L., e quando c'era tanto entusiasmo da volerlo fare diversamente dagli altri e magari anche meglio di quello che era stato già fatto.

Il primo volontario che ha guidato lo sviluppo del codice di rete nel kernel è stato Ross Biro <`biro@yggdrasil.com`>. Ross ha implementato un insieme di funzionalità semplice ed incompleto ma usabile per la maggior parte dei bisogni. Queste procedure erano distribuite insieme ad un driver per la scheda di rete WD-8003. Questo fu sufficiente a far sì che molte persone iniziassero a provare e sperimentare il software, e alcuni riuscirono pure a connettere delle macchine ad internet con questa configurazione. Ma nel frattempo la richiesta di software di rete da parte della comunità Linux stava crescendo, così che a un certo punto la domanda pressante che gravava su Ross ed il suo coinvolgimento personale in termini di tempo superarono a suo giudizio i vantaggi che ne derivavano. Ross lasciò quindi il suo posto alla guida del gruppo di programmatori. Gli sforzi di Ross per far partire il progetto e la responsabilità che si era preso di produrre davvero qualcosa di utile nonostante la situazione fosse controversa, sono state le cose che hanno catalizzato tutto il lavoro futuro, e sono state perciò una componente essenziale del successo del prodotto attuale.

Orest Zborowski <obz@Kodak.COM> ha scritto la prima interfaccia di programmazione per i socket BSD all'interno del kernel Linux. Questo è stato un grande passo in avanti, e ha permesso di far funzionare sotto Linux molte delle applicazioni di rete esistenti senza cambiamenti notevoli.

All'incirca nel medesimo momento Laurence Culhane <loz@holmes.demon.co.uk> ha sviluppato i primi driver per supportare il protocollo SLIP all'interno di Linux. Questo ha permesso a molte persone che non avevano accesso a reti di tipo Ethernet di provare il nuovo software di rete. Ancora una volta, qualcuno prese questo driver e lo mise al lavoro per collegarsi ad Internet. Questa possibilità diede a molti un'idea delle possibilità che si sarebbero potute realizzare se Linux avesse avuto un supporto di rete completo, ed accrebbe il numero di utenti che provavano ed usavano il software esistente.

Un'altra persona che ha lavorato attivamente nel compito di costruire il supporto per la rete è stata Fred van Kempen, <waltje@uwalt.nl.mugnet.org>. Dopo un periodo di incertezza che aveva seguito l'abbandono di Ross dalla posizione di coordinatore dello sviluppo, Fred aveva offerto il suo tempo e le sue capacità e aveva accettato il ruolo senza alcuna opposizione. Fred aveva dei piani ambiziosi per la direzione che il software di rete di Linux avrebbe dovuto prendere, e si preparò a progredire in queste direzioni. Fred ha prodotto una serie di programmi di rete chiamati 'NET-2' che molte persone sono state in grado di usare proficuamente (il codice 'NET' era quello di Ross). Fred ha introdotto un certo numero di innovazioni nell'agenda dello sviluppo, come l'interfaccia dinamica per le periferiche, il protocollo AX.25 per le radio amatoriali e un'implementazione del networking progettata più modularmente. Il codice NET-2 di Fred fu usato da un numero molto elevato di entusiasti, ed il loro numero cresceva in continuazione mentre si diffondeva la voce che questa implementazione funzionava bene. Il software di rete a questo punto esisteva ancora sotto forma di un ampio numero di modifiche da applicare alla distribuzione ufficiale del kernel, non essendo ancora stato incluso in tale distribuzione. I documenti NET-FAQ e successivamente NET-2-HOWTO descrivevano la procedura (allora abbastanza complessa) per avere tutto quanto funzionante. L'impegno di Fred era volto allo sviluppo di innovazioni all'implementazione di rete convenzionale e questo gli prendeva tempo. La comunità di utenti stava diventando impaziente aspettando qualcosa che funzionasse in modo affidabile e che soddisfacesse l'80% degli utenti. Come era successo con Ross, la pressione su Fred come coordinatore del progetto crebbe considerevolmente.

Alan Cox <iialan@www.uk.linux.org> propose una soluzione al problema che avrebbe dovuto risolvere le controversie. Egli propose di prendere il codice NET-2 di Fred e di sistemarlo, rendendolo affidabile e stabile così da soddisfare gli utenti impazienti, nel contempo avrebbe alleggerito la pressione su Fred permettendogli di continuare il suo lavoro. Alan si mise a far ciò con un certo successo e la sua versione del codice di rete per Linux prese il nome di 'Net-2D(ebugged)'. Il codice funzionava in maniera affidabile in molte configurazioni comuni e la base degli utenti era contenta. Alan chiaramente aveva idee e capacità proprie da mettere nel progetto e in molte delle discussioni relative alla direzione che il codice NET-2 stava prendendo. Si svilupparono allora due diverse scuole di pensiero nella comunità Linux: una che seguiva la filosofia "farlo funzionare subito, migliorarlo in seguito" e un'altra che diceva "farlo meglio subito". Linus fece la parte dell'arbitro e incluse il codice di Alan nella distribuzione ufficiale del kernel. Questo pose Fred in una posizione difficile. Ogni sviluppo innovativo avrebbe mancato della larga base di utenti che testassero ed usassero attivamente il codice, e questo avrebbe significato un rallentamento e certe difficoltà nello sviluppo. Fred continuò a lavorare per un breve periodo e alla fine abbandonò, così Alan si trovò ad essere il nuovo leader dello sviluppo del supporto di rete nel kernel di Linux.

Donald Becker <becker@cesdis.gsfc.nasa.gov> ha rivelato presto le sue capacità negli aspetti di più basso livello del networking, e ha prodotto un grosso campionario di driver per schede di rete: quasi tutte quelle incluse nei kernel attuali sono state sviluppate da Donald. Ci sono state altre persone che hanno contribuito in maniera significativa, ma il lavoro di Donald è stato notevole, per cui merita una menzione particolare.

Alan ha continuato a migliorare il codice NET-2-Debugged per un certo tempo, mentre lavorava nell'implementazione di alcune delle cose nella lista delle priorità che non erano ancora state affrontate. Prima che il sorgente del kernel `1.3.*` diventasse maturo, il codice di rete nel kernel era passato alla distribuzione NET-3, sulla quale sono basate le versioni attuali. Alan ha lavorato su molti aspetti del codice di rete e ha migliorato il codice in molte direzioni con l'aiuto di un certo numero di altre persone di talento nella comunità Linux. Alan ha prodotto i dispositivi di rete dinamici e le prime implementazioni standard di AX.25 e IPX. Alan ha continuato a giocare col codice, ristrutturandolo e migliorandolo lentamente fino allo stato in cui si trova oggi.

Il supporto per PPP è stato aggiunto da Michael Callahan <`callahan@maths.ox.ac.uk`> e Al Longyear <`longyear@netcom.com`>. Anche questa è stata un'innovazione significativa per aumentare il numero di persone che usano Linux attivamente nel networking.

Jonathon Naylor <`jsn@cs.nott.ac.uk`> ha contribuito migliorando in maniera significativa il codice AX.25 di Alan, aggiungendo il supporto per i protocolli NetRom e Rose. La capacità di usare AX.25, NetRom e Rose è un fatto significativo in se, poiché nessun altro sistema operativo può vantare il supporto nativo per tutti questi protocolli.

Naturalmente, ci sono state altre centinaia di persone che hanno contribuito significativamente allo sviluppo del software di rete per Linux. Alcune di queste le incontrerete più avanti nelle sezioni relative alle singole tecnologie, altre persone hanno fornito moduli, driver, correzioni di errori, suggerimenti, risultati di test e supporto morale. In ognuno di questi casi ciascuno può dire di aver giocato la sua parte e di avere offerto quello che poteva. Il codice di rete del kernel è un esempio eccellente dei risultati che si possono ottenere dallo stile di lavoro anarchico della comunità Linux. Se questo non vi ha ancora meravigliato, lo farà presto: lo sviluppo non si è fermato.

4.2 Dove trovare altre informazioni riguardo al Networking in Linux

Ci sono vari posti in cui si possono trovare delle buone informazioni riguardo all'uso delle reti in Linux.

Alan Cox, il coordinatore attuale del codice di rete nel kernel, ha una pagina WWW che contiene informazioni sullo stato attuale e sul nuovo sviluppo del supporto di rete per Linux. La sua pagina è: *www.uk.linux.org* `<http://www.uk.linux.org/NetNews.html>`.

Un'altra risorsa interessante è un libro scritto da Olaf Kirch, intitolato "`Network Administrators Guide`". È un lavoro del *Linux Documentatation Project* `<http://sunsite.unc.edu/LDP/>` e può essere letto interattivamente presso *Network Administrators Guide, versione HTML* `<http://sunsite.unc.edu/LDP/LDP/nag/nag.html>`. Oppure può essere scaricato tramite ftp in vari formati dall'archivio ftp dell'LDP su *sunsite.unc.edu LDP ftp archive* `<ftp://sunsite.unc.edu/pub/Linux/docs/LDP/network-guide/>`. Il libro di Olaf è molto completo e fornisce una buona panoramica di alto livello della configurazione di rete sotto Linux.

Esiste un newsgroup dedicato alla rete e argomente connessi nella gerarchia di gruppi relativi a Linux. È: *comp.os.linux.networking* `<news:comp.os.linux.networking>`

C'è una mailing list alla quale ci si può iscrivere, nella quale si possono chiedere informazioni a proposito del networking sotto Linux. Per iscriversi occorre mandare un messaggio di posta elettronica:

```
To: majordomo@vger.rutgers.edu
Subject: anything at all
Message:

subscribe linux-net
```

Nelle varie reti IRC si trovano spesso dei canali `#linux` sui quali si possono trovare persone in grado di rispondere a domande sull'argomento.

Ogni volta che si presenta un problema bisogna ricordare di includere il maggior numero possibile di dettagli riguardo ai problemi. In particolare, occorre includere la versione del software che si sta usando, in particolare la versione del kernel, la versione dei programmi come *pppd* e *dip*, e la natura esatta del problema che si è verificato. Questo vuol dire prendere nota della sintassi esatta di ogni messaggio di errore ricevuto, e di tutti i comandi che si stanno invocando.

4.3 Dove trovare informazioni di rete non correlate a Linux

Per chi sta cercando informazioni introduttive e pratiche sul networking tcp/ip in genere, io raccomando di dare un'occhiata ai seguenti documenti:

tcp/ip introduction

questo documento esiste sia in *versione testo* <ftp://athos.rutgers.edu/runet/tcp-ip-intro.doc> sia in *versione postscript* <ftp://athos.rutgers.edu/runet/tcp-ip-intro.ps>.

tcp/ip administration

questo documento esiste sia in *versione testo* <ftp://athos.rutgers.edu/runet/tcp-ip-admin.doc> sia in *versione postscript* <ftp://athos.rutgers.edu/runet/tcp-ip-admin.ps>.

Per chi sta cercando informazioni più dettagliate su tcp/ip raccomando caldamente:

> *Internetworking with TCP/IP, Volume 1: principles, protocols and architecture*, by Douglas E. Comer, ISBN 0-13-227836-7, Prentice Hall publications, Third Edition, 1995.

[tradotto in italiano per Jackson Libri N.d.T.]

Per chi vuole imparare come scrivere applicazioni di rete in un ambiente compatibile a Unix, raccomando altrettanto caldamente:

> *Unix Network Programming*, by W. Richard Stevens, ISBN 0-13-949876-1, Prentice Hall publications, 1990.

Una seconda edizione di questo libro sta per apparire sugli scaffali delle librerie; la nuova edizione è composta da tre volumi: *sito web di Prentice-Hall* <http://www.phptr.com/> per ulteriori informazioni.

Si può anche provare a guardare nel newsgroup *comp.protocols.tcp-ip* <news:comp.protocols.tcp-ip>.

Una fonte importante di informazione prettamente tecnica riguardo a Internet e alla suite di protocolli tcp/ip sono gli RFC. RFC è un acronimo che significa 'Request For Comment' ed è il modo convenzionale per pubblicare e documentare gli standard sui protocolli Internet. Ci sono svariati archivi di RFC, molti di questi sono siti ftp mentre altri forniscono accesso via WWW con un motore di ricerca che permette di cercare le parole chiave in un database associato agli RFC.

Un possibile luogo ove trovare gli RFC è il *database RFC di Nexor* <http://pubweb.nexor.co.uk/public/rfc/index/rfc.html>.

5 Informazioni generali sulla configurazione di rete.

È piuttosto importante conoscere e capire le seguenti sottosezioni prima di provare a configurare in pratica la propria rete. Questi sono principi fondamentali che si applicano indipendentemente dall'esatta natura della rete che si intende costruire.

5.1 Da dove devo partire?

Prima di iniziare a costruire o configurare la propria rete occorrono alcune cose. Le più importanti di queste sono:

5.1.1 Sorgenti aggiornati del kernel.

Siccome il kernel che si sta usando al momento potrebbe non avere ancora il supporto per i tipi di rete o di schede di interfaccia che si desidera usare, probabilmente occorrono i sorgenti del kernel, in modo da essere in grado di ricompilare il kernel con le opzioni appropriate.

I sorgenti più recenti si possono sempre ottenere da *ftp.kernel.org* `<ftp://ftp.kernel.org/pub/linux/kernel>`.

Meglio ricordarsi che ftp.kernel.org viene seriamente sovraccaricato: un modo preferibile di ottenere i sorgenti aggiornati è scaricare le patch invece degli interi file sorgente in tar; meglio ancora cercare di utilizzare mirror del sito ftp principale, come *ftp.funet.fi* `<ftp://ftp.funet.fi//mirrors/ftp.kernel.org/pub/linux/kernel>`; ricordarsi inoltre che ogni sito Linux di solito fornisce sorgenti del kernel aggiornati.

Normalmente i sorgenti del kernel saranno scompattati da tar nella directory `/usr/src/linux`. Per informazioni riguardo a come si applicano le modifiche (patch) e come si ricompila il kernel si legga *Kernel-HOWTO* `<Kernel-HOWTO.html>`. Per informazioni su come si configurano i moduli del kernel si legga "Modules mini-HOWTO". Inoltre il file `README` che si trova nei sorgenti del kernel e la directory `Documentation` sono molto istruttivi per il lettore motivato.

A meno di trovare indicazioni differenti, io consiglio di utilizzare la distribuzione stabile del kernel (quella con un numero pari come seconda cifra nel numero di versione). Le distribuzioni dei kernel di sviluppo (quelle con la seconda cifra dispari) possono presentare differenze strutturali o altri cambiamenti che possono causare problemi quando usati con altro software di sistema. Se non si è sicuri di essere in grado di risolvere questo tipo di problemi, anche considerando il rischio che ci siano altri errori nel software, allora è bene non usare le versioni dispari.

D'altra parte, alcune delle funzionalità ivi descritte sono state introdotte nel corso dello sviluppo dei kernel 2.1, per cui è necessario fare una scelta: si può rimanere al 2.0, aspettando che esca il 2.2 e qualche distribuzione aggiornata e completa di ogni programma di supporto, o passare ai 2.1 ed aggiungere in proprio i vari programmi di supporto necessari per utilizzare al meglio le nuove funzionalità. Al momento della stesura di questo paragrafo, nell'Agosto 1998, è già stato rilasciato il 2.1.115 e si pensa che non manchi molto al rilascio del 2.2 [al momento della traduzione sono già uscite alcune distribuzioni con i nuovi kernel 2.2, tra le quali Debian 2.1 N.d.T.].

5.1.2 Gli strumenti di rete aggiornati

Gli strumenti di rete (i "network tools") sono i programmi che si usano per configurare le periferiche di rete di Linux. Questi programmi permettono per esempio di assegnare gli indirizzi alle periferiche e configurare l'instradamento.

La maggior parte delle moderne distribuzioni di Linux contengono già gli strumenti di rete, se perciò si è installato il proprio sistema da una distribuzione senza aver ancora installato gli strumenti di rete, occorre farlo.

Se il proprio sistema non è stato installato da una distribuzione, occorre recuperare i sorgenti e compilare i programmi da se. Questo non è difficile.

Gli strumenti di rete sono attualmente distribuiti da Bernd Eckenfels e sono disponibili presso: *ftp.inka.de*

<ftp://ftp.inka.de/pub/comp/Linux/networking/NetTools/>, sito che è presente in mirror anche su *ftp.uk.linux.org* <ftp://ftp.uk.linux.org/pub/linux/Networking/base/>.

Occorre essere sicuri di scegliere la versione più appropriata per il kernel che si intende usare e di seguire le istruzioni di installazione che si trovano nel pacchetto.

Per installare e configurare la versione corrente al momento della stesura di questo documento occorre invocare i seguenti comandi:

```
user% tar xvfz net-tools-1.33.tar.gz
user% cd net-tools-1.33
user% make config
user% make
root# make install
```

Inoltre, se si intende configurare un firewall o utilizzare la possibilità di mascheramento dei pacchetti ("IP masquerading") occorre il comando *ipfwadm*, la cui versione aggiornata si può recuperare da: *ftp.xos.nl* <ftp:/ftp.xos.nl/pub/linux/ipfwadm>. Ancora una volta, esistono diverse versioni del programma, e bisogna prendere la versione che meglio si adatta al proprio kernel. Da notare che le funzionalità di firewall di Linux sono cambiate durante lo sviluppo dei 2.1. Quanto si dice si applica solo alle versioni 2.0 del kernel.

Per installare e configurare la versione cui ho accesso in questo momento occorre invocare:

```
user% tar xvfz ipfwadm-2.3.0.tar.gz
user% cd ipfwadm-2.3.0
user% make
root# make install
```

Notare che se si utilizza una versione 2.2 (o uno degli ultimi 2.1) del kernel, *ipfwadm* non è lo strumento giusto per configurare il Firewall IP. Questa versione del NET-3-HOWTO attualmente non si occupa della nuova configurazione del firewall.

5.1.3 Programmi Applicativi di Rete.

Gli applicativi di rete sono i programmi come *telnet* e *ftp*, unitamente ai programmi server associati. David Holland <dholland@cs.harvard.edu> coordina la distribuzione dei più comuni di questi. Tale distribuzione si può ottenere *ftp.uk.linux.org* <ftp://ftp.uk.linux.org/pub/linux/Networking/base>.

Nel Marzo 1997 il pacchetto è stato suddiviso in pacchetti distinti più piccoli, ma nel Maggio 1997 i programmi fondamentali sono stati incorporati in un pacchetto chiamato `netkit-base-0.10`. Potrebbe essere necessario procurarsi il pacchetto di base e/o pacchetti addizionali.

Per installare e configurare la versione corrente al momento della stesura di questo documento occorre invocare i seguenti comandi:

```
user% tar xvfz netkit-base-0.10
user% cd netkit-base-0.10
user% more README
user% vi MCONFIG
user% make
root# make install
```

5.1.4 Indirizzi.

Gli indirizzi IP sono composti da 4 byte. La convenzione usata per scrivere gli indirizzi è chiamata 'dotted decimal notation', che significa "notazione decimale con i punti". In questa forma ogni byte è convertito in un numero decimale (tra 0 e 255) scartando gli zeri prefissi a meno che il numero sia zero, ogni byte è poi separato da un carattere '.'. Per convenzione ogni interfaccia di un calcolatore o router ha associato un indirizzo IP. In certe circostanze è permesso assegnare lo stesso indirizzo a tutte le interfacce di un singolo calcolatore, ma solitamente ogni interfaccia avrà un indirizzo diverso.

Le reti IP sono sequenze di indirizzi IP contigui. Tutti gli indirizzi all'interno di una rete hanno un certo numero di cifre del loro indirizzo in comune. La porzione di indirizzo comune all'interno della rete si chiama "network portion" (porzione di rete) dell'indirizzo. Le cifre rimanenti si chiamano "host portion". Il numero di bit che vengono condivisi tra tutti gli indirizzi all'interno della rete è chiamato "netmask" ed il suo ruolo è determinare quali indirizzi appartengono alla rete cui la maschera è applicata e quali no. Consideriamo il seguente esempio:

```
-----------------  ---------------
Host Address       192.168.110.23
Network Mask       255.255.255.0
Network Portion    192.168.110.
Host portion                  .23
-----------------  ---------------
Network Address    192.168.110.0
Broadcast Address  192.168.110.255
-----------------  ---------------
```

Ogni indirizzo sottoposto ad un'operazione di "AND bit-a-bit" con la sua maschera di rete darà l'indirizzo della rete cui appartiene. La maschera di rete è perciò sempre il numero più basso all'interno dell'intervallo di indirizzi che formano quella rete, e ha sempre la parte di host dell'indirizzo pari a zero.

L'indirizzo di broadcast è un indirizzo speciale cui tutti gli host della rete ascoltano, in aggiunta al loro proprio indirizzo. Questo indirizzo è quello cui vengono mandati i pacchetti che devono essere ricevuti da tutti i calcolatori della rete. Certi tipi di dati, come le informazioni di instradamento e i messaggi di errore vengono trasmessi all'indirizzo di broadcast in modo che tutti i calcolatori sulla rete li possano ricevere contemporaneamente. Ci sono due standard comunemente usati su come debba essere un indirizzo di broadcast. Il più comunemente accettato stabilisce che debba essere usato come indirizzo di broadcast l'indirizzo più alto possibile della rete. Nell'esempio precedente questo sarebbe `192.168.110.255`. Per qualche ragione alcuni siti hanno adottato la convenzione di usare l'indirizzo zero come indirizzo di broadcast. In pratica non fa molta differenza quale viene usato, ma bisogna assicurarsi che ogni host sulla rete sia configurato con lo stesso indirizzo di broadcast.

Per ragioni amministrative, ad un certo punto durante lo sviluppo iniziale del protocollo IP sono stati formati alcuni gruppi di arbitrari di indirizzi e le reti sono state raggruppate in quelle che sono chiamate classi. Le classi caratterizzano le dimensioni delle reti che si possono allocare. Queste classi offrono un numero di dimensioni fisse di rete che possono essere allocate. Gli intervalli scelti per le varie classi sono:

```
----------------------------------------------------------
| Network | Netmask       | Network Addresses            |
| Class   |               |                              |
----------------------------------------------------------
|    A    | 255.0.0.0     | 0.0.0.0     - 127.255.255.255 |
|    B    | 255.255.0.0   | 128.0.0.0   - 191.255.255.255 |
|    C    | 255.255.255.0 | 192.0.0.0   - 223.255.255.255 |
```

```
|Multicast| 240.0.0.0     | 224.0.0.0  - 239.255.255.255 |
----------------------------------------------------------
```

La scelta di quali indirizzi usare per la propria rete dipende da cosa esattamente si sta facendo. Per ottenere gli indirizzi di cui si ha bisogno si può procedere in uno dei seguenti modi:

Installare una macchina Linux su una rete IP esistente

Se si desidera installare un calcolatore su una rete esistente si deve contattare chi amministra tale rete e chiedere loro le seguenti informazioni:

- L'indirizzo IP da assegnare al calcolatore
- L'indirizzo IP di rete
- L'indirizzo IP di broadcast
- La netmask
- L'indirizzo del router
- L'indirizzo del "name server".

Dopo di che bisogna configurare l'interfaccia di rete Linux in base a quei numeri. Non è possibile inventare dei numeri e sperare che la configurazione funzioni.

Costruire una rete nuova che non si connetterà mai a Internet

Se si sta preparando una rete privata e non si ha intenzione nemmeno in futuro di connettere tale rete ad Internet, allora si può scegliere qualunque indirizzo. Ciononostante, per ragioni di sicurezza e di coerenza ci sono alcuni indirizzi di reti IP che sono stati riservati specificamente a questo fine. Essi sono specificati nell'RFC1597 come segue:

```
-----------------------------------------------------------
|         RESERVED PRIVATE NETWORK ALLOCATIONS            |
-----------------------------------------------------------
| Network | Netmask       | Network Addresses             |
| Class   |               |                               |
-----------------------------------------------------------
|    A    | 255.0.0.0     | 10.0.0.0    - 10.255.255.255  |
|    B    | 255.255.0.0   | 172.16.0.0  - 172.31.255.255  |
|    C    | 255.255.255.0 | 192.168.0.0 - 192.168.255.255 |
-----------------------------------------------------------
```

Occorre innanzitutto decidere quanto grande deve essere la nuova rete, e poi scegliere gli indirizzi di cui si ha bisogno.

5.2 Dove bisogna mettere i comandi di configurazione?

Ci sono differenti approcci sotto Linux per le procedure di inizializzazione del sistema. Dopo che il kernel è partito, viene sempre eseguito un programma chiamato '*init*'. Il programma *init* poi legge il suo file di configurazione chiamato `/etc/inittab` ed inizia il processo di boot. Esistono versioni di *init* leggermente differenti, quantunque si stia convergendo verso lo stile System V (Five), sviluppato da Miguel van Smoorenburg.

Malgrado il fatto che il programma *init* sia sempre lo stesso, la configurazione di boot del sistema è organizzata in modi diversi nelle varie distribuzioni.

Solitamente il file `/etc/inittab` contiene una voce che assomiglia a:

```
si::sysinit:/etc/init.d/boot
```

Questa riga specifica il nome dello script di shell che gestisce praticamente la sequenza di boot. Questo file è in qualche modo simile al file `AUTOEXEC.BAT` in DOS.

Di solito ci sono altri script che vengono chiamati dallo script di inizializzazione, e spesso la rete è configurata all'interno di uno di questi.

La seguente tabella può essere usata come guida per il proprio sistema:

```
-------------------------------------------------------------------
Distrib. | Interface Config/Routing          | Server Initialization
-------------------------------------------------------------------
Debian   | /etc/init.d/network               | /etc/rc2.d/*
-------------------------------------------------------------------
Slackware| /etc/rc.d/rc.inet1                | /etc/rc.d/rc.inet2
-------------------------------------------------------------------
RedHat   | /etc/rc.d/init.d/network          | /etc/rc.d/rc3.d/*
-------------------------------------------------------------------
```

Si noti che Debian e Red Hat usano un'intera directory per ospitare gli script che attivano i servizi di sistema. Gli script di solito non contengono al loro interno le informazioni di configurazione necessarie. Ad esempio nei sistemi RedHat gli script di inizializzazione (boot script) si aspettano di trovarle nei file presenti in `/etc/sysconfig`. Se si vogliono comprendere i dettagli del processo di boot, suggerisco di dare un occhiata a */etc/inittab* e alla documentazione associata a *init*. Linux Journal sta per pubblicare un articolo sulla inizializzazione del sistema, e a questo documento verrà aggiunto un puntatore ad esso non appena sarà disponibile sul web.

La maggior parte delle distribuzioni moderne includono un programma che permette di configurare la maggior parte dei tipi diffusi di interfacce di rete. Se si ha una di queste, allora bisognerebbe controllare se tale programma fa quello di cui si ha bisogno prima di tentare una configurazione manuale.

```
-----------------------------------------
Distrib   | Network configuration program
-----------------------------------------
RedHat    | /usr/bin/netcfg
Slackware | /sbin/netconfig
-----------------------------------------
```

5.3 Creazione delle interfacce di rete.

In molti sistemi Unix le periferiche di rete hanno il loro posto nella directory */dev*. Questo non succede in Linux, dove i dispositivi di rete sono creati dinamicamente via software e quindi non hanno bisogno della presenza di file speciali.

Nella maggior parte dei casi i dispositivi di rete sono creati automaticamente dal driver durante la sua inizializzazione dopo che l'hardware è stato riconosciuto. Per esempio, il driver per la rete ethernet crea le interfacce `eth[0..n]` sequenzialmente, nell'ordine in cui le schede ethernet vengono riconosciute. La prima scheda che viene trovata prende il nome `eth0`, la seconda `eth1`, eccetera.

Ciononostante in alcuni casi, come succede per *slip* e *ppp*, i dispositivi di rete vengono creati a seguito di operazioni svolte da un programma. Si applica la stessa regola di numerazione sequenziale, ma le periferiche non sono create automaticamente all'accensione del sistema. La ragione di questo è che, a differenza di quello che accade per le schede ethernet, il numero di periferiche *slip* o *ppp* può cambiare durante la vita della macchina. Questi casi vengono analizzati in maggior dettaglio nelle sezioni successive.

5.4 Configurazione delle interfacce di rete.

Dopo aver recuperato tutti i programmi di cui si ha bisogno e tutti gli indirizzi e le informazioni sulla rete, si può procedere alla configurazione delle proprie interfacce di rete. Quando si parla di configurazione dell'interfaccia si intende il processo di assegnazione degli indirizzi appropriati alla periferica e all'assegnazione di valori appropriati per gli altri valori configurabili di un dispositivo di rete. Il programma più comunemente usato per questo compito è il comando *ifconfig* (interface configure).

Normalmente viene invocato un comando simile al seguente:

```
root# ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
```

In questo esempio si è configurata una scheda ethernet '`eth0`' con l'indirizzo IP '`192.168.0.1`' e una maschera pari a '`255.255.255.0`'. La parola '*up*' finale significa che l'interfaccia deve essere attivata, ma può di solito essere omessa, dato che è l'opzione di default.

Il kernel assume alcuni valori di default quando un'interfaccia viene configurata. Per esempio, si può specificare l'indirizzo di rete e quello di broadcast per un'interfaccia, ma se questo non viene fatto, come nell'esempio precedente, il kernel farà delle scelte ragionevoli per i valori da assegnare in base alla maschera di rete fornita; se non viene fornita una maschera di rete si baserà sulla classe della rete associata all'indirizzo IP. Nell'esempio appena visto il kernel assegnerà un indirizzo di rete '`192.168.0.0`' e un indirizzo di broadcast '`192.168.0.255`' in base alla netmask specificata nel comando.

Si possono passare al comando *ifconfig* molte altre opzioni. Le più importanti di queste sono:

up

questa opzione attiva l'interfaccia (ed è l'impostazione di default).

down

questa opzione la disattiva.

[- arp]

questa opzione abitita o disabilita l'uso del protollo ARP ("Address Resulution Protocol") per questa interfaccia.

[- allmulti]

questa opzione abilita o disabilita la ricezione di tutti i pacchetti multicast. Il multicast permette di associare speciali indirizzi hardware di destinazione a gruppi di calcolatori, in modo che tutti gli appartenenti al gruppo ricevano certi pacchetti. Questa opzione può essere importante se si usano applicazioni come la videoconferenza ma normalmente non viene usata.

mtu N

questo parametro permette di assegnare il valore MTU ("Maximum Transfer Unit") per questa periferica.

netmask addr

questo parametro permette di specificare la maschera relativa alla rete cui questa interfaccia appartiene.

irq addr

questo parametro funziona solo con certi tipi di hardware e permette di scegliere la linea interruzione usata dall'hardware di questa interfaccia.

[- broadcast [addr]]

questo parametro permette di abilitare la ricezione di pacchetti di broadcast e assegna l'indirizzo di broadcast per questa interfaccia, oppure disabilita la ricezione di tali pacchetti.

[- pointopoint [addr]]

questo parametro permette di assegnare l'indirizzo della macchina che sta all'altro capo di una connessione punto-a-punto, come accade per *slip* e *ppp*.

hw <type> <addr>

questo parametro permette di assegnare l'indirizzo hardware di certi tipi di schede di rete. Questo è spesso inutile per per le interfaccie ethernet, ma è comodo per altri tipi di rete, come AX.25.

Il comando *ifconfig* può essere usato per agire su tutte le interfacce di rete. Alcuni programmi utente come *pppd* e *dip* configurano automaticamente le periferiche di rete quando le creano, e in questi casi l'uso manuale di *ifconfig* diventa inutile.

5.5 Configurazione del risolutore dei nomi (name resolver).

Il '*Name Resolver*' fa parte della libreria standard di Linux. La sua funzione principale è quella di fornire un servizio che converta i nomi dei calcolatori, quelli comprensibili all'uomo come '`ftp.funet.fi`', in indirizzi comprensibili alle macchine, come `128.214.248.6`.

5.5.1 Cosa compone un nome?

Anche chi ha familiarità con i nomi degli host che costituiscono Internet potrebbe non comprendere come sono costituiti. I nomi di dominio (domain name) di Internet sono inerentemente gerarchici, hanno cioè una struttura ad albero. Un '*dominio*' (domain) è una famiglia, un gruppo di nomi. Un dominio può essere suddiviso in vari '*sottodomini*'. Un '*dominio pricipale*' (toplevel domain) è un dominio che non è un sottodominio. I domini principali sono specificati nell'RFC 920. I domini principali più comuni sono:

COM

Organizzazioni commerciali

EDU

Organizzazioni educative

GOV

Organizzazioni governative

MIL

Organizzazioni militari

ORG

Altre organizzazioni

NET

Organizzazioni connesse a Internet

Identificativo nazionale

questi sono codici di due lettere che rappresentano la nazione.

Per ragioni storiche la maggior parte dei domini che appartengono ai domini principali non nazionali sono stati usati da organizzazioni con sede negli Stati Uniti, malgrado gli Stati Uniti abbiano anche un proprio dominio nazionale '`.us`'. Questo non è più vero per i domini `.com` and `.org`, che vengono comunemente usati da società e organizzazioni non statunitensi.

Ognuno di questi domini pricipali ha i suoi sottodomini. I domini principali basati sui nomi di nazione sono spesso suddivisi in sottodomini come `com`, `edu`, `gov`, `mil` e `org`. Questo non succede in Italia, ma per esempio esistono i sottodomini `com.au` e `gov.au` per le organizzazioni commerciali e governative in Australia; da notare che questa non è una regola generale, dato che le linee effettive di condotta dipendono dalle autorità che gestiscono a livello locale l'assegnazione dei nomi.

Il livello successivo di divisione spesso rappresenta il nome dell'organizzazione. I sottodomini ulteriori possono essere di diversa natura; spesso il livello successivo è basato sulla suddivisione interna della organizzazione, ma può essere basato su qualsiasi criterio che sia considerato ragionevole e significativo all'interno dell'organizzazione.

La parte più a sinistra del nome è sempre il nome univocamente assegnato ad un calcolatore, e si chiama '*hostname*'. La parte di nome che sta alla destra del "hostname" si chiama '*domainname*', e il nome completo si chiama '*Fully Qualified Domain Name*', abbreviato FQDN.

Usando l'host di Terry come esempio, il FQDN è '`perf.no.itg.telstra.com.au`'. Questo significa che il suo "hostname" è '`perf`', e il "domain name" è '`no.itg.telstra.com.au`'. Il suo "domain name" è composto da un dominio principale basato sul nome della sua nazione, l'Australia, e poiché il suo indirizzo di posta elettronica appartiene ad un organizzazione commerciale c'è '`.com`' come sottodominio. Il nome della compagnia è (era) '`telstra`', e la sua struttura di nomi interna è basata sulla struttura gestionale, in questo caso il calcolatore appartiene al gruppo di tecnologia delle informazioni (Information Technology Group), sezione operazioni di rete (Network Operations).

Di solito, i nomi sono un po' più brevi; ad esempio il mio ISP (Internet Service Provider = fornitore di accesso a internet) si chiama '`systemy.it`' e la mia organizzazione senza scopo di lucro si chiama '`linux.it`', senza nessun sottodominio `com` e `org` , così che il mio host personale si chiama '`morgana.systemy.it`' e `rubini@linux.it` è un indirizzo di posta elettronica valido. Da notare che chi amministra un dominio può registrare nomi di singoli host come anche sottodomini; ad esempio il LUG a cui appartengo usa il dominio `pluto.linux.it`, poiché i titolari di `linux.it` hanno acconsentito a creare un sottodominio per il LUG.

5.5.2 Quali informazioni sono necessarie.

Per accedere al servizio di risoluzione dei nomi occorre sapere a quale dominio appartiene il proprio calcolatore. Il software di risoluzione dei nomi offre il servizio di traduzione dei nomi facendo delle richieste ad un '*Domain Name Server*' (di solito chiamato semplicemente 'name server'), per cui occorre anche conoscere l'indirizzo di un name server locale che offra questo servizio.

Per configurare il proprio calcolatore occorre sistemare tre file, che descriverò uno alla volta.

5.5.3 /etc/resolv.conf

Il file `/etc/resolv.conf` è il file di configurazione principale per accedere al servizio di risoluzione dei nomi. Il suo formato è abbastanza semplice: si tratta di un file di testo con una parola chiave (direttiva) per linea. Le direttive comunemente usate sono tre:

domain

questa direttiva specifica il nome del dominio locale.

search

questa direttiva specifica una lista di domini da consultare in alternativa.

nameserver

questa direttiva, che può essere usata più di una volta, specifica l'indirizzo IP di un name server a cui rivolgersi per la risoluzione dei nomi.

Per esempio, `/etc/resolv.conf` potrebbe essere qualcosa di simile a:

```
domain maths.wu.edu.au
search maths.wu.edu.au wu.edu.au
nameserver 192.168.10.1
nameserver 192.168.12.1
```

Questo esempio specifica che il dominio di default da usare per i nomi non qualificati (cioè gli hostname senza un dominio associato) è `maths.wu.edu.au`, e che se l'host non viene trovato in quel dominio bisogna provare direttamente anche il dominio `wu.edu.au`. Il file specifica anche due name server, ciascuno dei quali può essere consultato dal software di risoluzione per risolvere un nome.

5.5.4 /etc/host.conf

Il file `/etc/host.conf` è quello dove si dichiarano alcuni elementi che governano il comportamento del codice di risoluzione dei nomi. Il formato di questo file è descritto in dettaglio nella pagina del manuale di '`resolv+`'. In quasi tutti i casi queste due linee sono tutto quello che serve:

```
order hosts,bind
multi on
```

Questa configurazione dice al codice di risoluzione dei nomi di controllare il file `/etc/hosts` prima di tentare una ricerca attraverso un name server, e dice di ritornare tutti gli indirizzi definiti per un host descritto in `/etc/hosts`, invece che ritornare solo il primo.

5.5.5 /etc/hosts

Il file `/etc/hosts` è quello che contiene i nomi e gli indirizzi IP dei calcolatori locali. Se un host appare in questo file non occorre interrogare un name server per conoscere il suo indirizzo IP. Lo svantaggio di questo tipo di approccio è che occorre aggiornare questo file ogniqualvolta l'indirizzo IP relativo ad un calcolatore cambia. Solitamente, in un sistema ben gestito le uniche voci che appaiono in questo file sono una per l'interfaccia di loopback e una per il nome del calcolatore stesso.

```
# /etc/hosts
127.0.0.1       localhost loopback
192.168.0.1     this.host.name
```

È possibile specificare più di un nome di calcolatore per linea, come dimostrato dalla prima voce qui sopra, che è il modo convenzionale per dare un nome all'interfaccia di loopback.

5.5.6 Attivare un name server

Se si desidera far girare un name server locale, lo si può fare facilmente. Si prega di fare riferimento al *DNS-HOWTO* `<DNS-HOWTO.html>` e a ogni documento incluso nella propria versione di *BIND* (Berkeley Internet Name Domain).

5.6 Configurazione dell'interfaccia di loopback.

L'interfaccia '`loopback`' è un tipo speciale di interfaccia che permette ad un calcolatore di effettuare connessioni con se stesso. Ci sono diversi motivi per cui capita di aver bisogno di farlo; per esempio quando occorre verificare il funzionamento di programmi di rete senza interferire con nessun altro sulla rete. Per convenzione è stato assegnato all'interfaccia di loopback l'indirizzo IP '`127.0.0.1`'. Perciò, indipendentemente dal calcolatore sul quale si lavora, se si apre una connessione telnet a `127.0.0.1` si raggiungerà sempre la macchina da cui si è partiti.

La configurazione dell'interfaccia di loopback è semplice, ed è una cosa che bisogna assicurarsi di fare (ma da notare che di solito tale compito è svolto dagli script standard di inizializzazione).

```
root# ifconfig lo 127.0.0.1
root# route add -host 127.0.0.1 lo
```

Il comando *route* verrà trattato più estesamente nella prossima sezione.

5.7 Routing.

Il "routing", ovvero le questioni relative all'instradamento dei pacchetti, costituisce un argomento ampio. Si possono facilmente scrivere grossi libri su queste tematiche. La maggior parte delle persone, comunque, hanno necessità di instradamento abbastanza semplici, mentre poche altre hanno esigenze più complicate. Tratterò qui solo i concetti fondamentali dell'instradamento dei pacchetti. Suggerisco a chi è interessato ad avere informazioni più dettagliate di consultare i manuali elencati all'inizio di questo documento.

Inizierei con una definizione. Cos'è l'instradamento IP? Questa è la definizione che uso io:

> L'instradamento IP (routing) è il procedimento attraverso il quale un calcolatore con connessioni di rete multiple decide dove trasmettere i pacchetti IP che ha ricevuto.

Potrebbe essere utile illustrare questa definizione con un esempio. Immaginiamo un tipico router in un ufficio, che abbia una connessione PPP verso Internet, un certo numero di segmenti ethernet che collegano le postazioni locali e un secondo collegamento PPP verso un altro ufficio. Quando il router riceve un pacchetto da una qualsiasi delle sue connessioni di rete, il meccanismo usato per determinare su quale interfaccia deve essere spedito il pacchetto è proprio il routing. Anche gli host più semplici hanno bisogno di instradare i pacchetti, tutti gli host di Internet hanno infatti due interfacce di rete: una è quella di loopback descritta prima e l'altra è queslla usata per comunicare col resto della rete; questa può essere una ethernet oppure un collegamento su porta seriale PPP o SLIP.

Bene, ma come funziona l'instradamento? Ogni host mantiene una lista di regole di instradamento, chiamata "routing table". Questa tabella contiene delle voci che sono in genere formate da almeno tre campi: il primo è l'indirizzo di destinazione, il secondo è il nome dell'interfaccia attraverso la quale instradare il pacchetto, e il terzo è l'indirizzo IP opzionale di un'altra macchina che si incarichi di decidere il prossimo passo che il pacchetto deve fare attraverso la rete. In Linux la tabella di instradamento può essere visualizzata usando il seguente comando:

```
user% cat /proc/net/route
```

oppure uno dei seguenti:

```
user% /sbin/route -n
user% /bin/netstat -r
```

Il procedimento di instradamento è abbastanza semplice: viene ricevuto un pacchetto, viene esaminato il suo indirizzo di destinazione (a chi è destinato quel pacchetto) e tale indirizzo viene confrontato con tutte le voci della tabella. La voce che meglio rispecchia l'indirizzo di destinazione viene poi usata per la ritrasmissione del pacchetto, attraverso l'interfaccia specificata dalla voce. Se poi il campo 'gateway' di questa voce è valido il pacchetto viene passato a tale host attraverso l'interfaccia specificata; in caso contrario si assume che l'indirizzo di destinazione sia sulla rete connessa all'interfaccia scelta.

Per manipolare la tabella esiste un comando specifico. Questo comando riceve degli argomenti sulla linea di comando e li converte in chiamate di sistema che richiedono al kernel di aggiungere, rimuovere o modificare le voci della tabella di routing. Tale comando si chiama '*route*'.

Passiamo ora ad un semplice esempio, immaginando che voi siate connessi ad una ethernet e che vi sia stato detto che la rete è una classe C con indirizzo `192.168.1.0`; immaginiamo inoltre che l'indirizzo `192.168.1.10` sia stato assegnato alla vostra macchina, e che `192.168.1.1` sia il router connesso al resto di Internet.

Il primo passo da fare è configurare l'interfaccia come descritto in precedenza. A questo fine si userà un comando tipo:

```
root# ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
```

Adesso occorre aggiungere nella tabella di routing una voce che dica al kernel che tutti i pacchetti destinati a calcolatori con indirizzi del tipo `192.168.1.*` devono essere spediti sull'interfaccia ethernet. Il comando per dire ciò sarà:

```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
```

Si noti l'uso dell'argomento '`-net`', che dice al programma "route" che questa voce si riferisce ad un'intera rete. L'altra possibilità è quella di specificare una regola di tipo '`-host`', cioè una regola di instradamento specifica ad un singolo indirizzo IP.

La regola di instradamento appena mostrata è quella che permette di stabilire connessioni con tutti gli host del proprio segmento ethernet. Ma come si fa a connettersi a tutte le macchine che non sono sul proprio ramo ethernet?

Dover aggiungere regole di instradamento per tutte le possibili reti sarebbe un lavoro molto difficile; perciò esiste un trucco per semplificare questo compito. Il trucco si chiama "regola di instradamento di '`default`'". La regola "di default" si riferisce a tutti gli indirizzi di destinazione, ma in modo "blando", cosicchśe esiste un'altra voce nella tabella che si riferisce all'indirizzo di destinazione del pacchetto, questa voce verrà usata al posto di quella di default. L'idea della regola di `default` è semplicemente quella di dire "e tutto il resto deve andare qui". Nell'esempio di cui ci stiamo occupando si userà un comando come:

```
# route add default gw 192.168.1.1 eth0
```

L'argomento 'gw' dice al comando 'route' che l'argomento seguente è l'indirizzo numerico, o il nome, di una macchina che fa da gateway, o router, cui devono essere spediti tutti i pacchetti ai quali questa regola si applica, ai fini di un ulteriore instradamento.

Perciò, la vostra configurazione completa sarà:

```
root# ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# route add default gw 192.168.1.1 eth0
```

Guardando attentamente i file di configurazione di rete che si trovano sulle macchine Linux troverete che almeno uno di essi sarà molto simile a quello appena mostrato. Questo tipo di configurazione è molto diffuso.

Vediamo ora una configurazione dell'instradamento leggermente più complicata. Immaginiamo di configurare il router che abbiamo visto prima, quello con la connessione PPP verso Internet e i segmenti ethernet verso i calcolatori nell'ufficio. Immaginiamo che il router abbia tre segmenti ethernet e un collegamento PPP. La nostra configurazione di routing sarà qualcosa come:

```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# route add -net 192.168.2.0 netmask 255.255.255.0 eth1
root# route add -net 192.168.3.0 netmask 255.255.255.0 eth2
root# route add default ppp0
```

Ognuna delle workstation sulla rete locale userà la forma più semplice presentata prima, solo il router deve specificare separatamente le informazioni relativa a ciascun tratto della rete locale, perché per le workstation il meccanismo della regola di `default` si occuperà di tutte le sottoreti, lasciando al router il compito di suddividere correttamente i pacchetti. Ci si può chiedere perché la regola di default appena vista per il router non specifichi un 'gw'. La ragione è semplice: i protocolli su linea seriale come PPP e SLIP hanno sempre e solo due calcolatori sulla loro "rete": uno ad ogni estremo. Specificare il calcolatore che si trova all'altro estremo del cavo come 'gateway' è inutile e ridondante, poiché non esistono altre possibilità, e per questo motivo non occorre esplicitare il gateway per questo tipo di connessioni. Altri tipi di rete, come ethernet, arcnet o token ring, richiedono invece che si specifichi il numero del gateway, poiché queste reti permettono a molti calcolatori di essere collegati insieme.

5.7.1 Allora a cosa serve il programma routed?

La configurazione di routing descritta fino adessso si applica bene a situazioni di rete semplici, dove c'è sempre solo un singolo percorso possibile per una data destinazione. Se la propria rete è più complessa, le cose diventano più complicate. Fortunatamente per la maggior parte delle persone questo non è un problema.

Il problema principale con il cosiddetto 'instradamento manuale' o 'instradamento statico' come quello appena descritto è che se un calcolatore o un collegamento all'interno della propria rete smette di funzionare, l'unico modo (se possibile) per dirigere i pacchetti su di un'altra strada consiste nell'intervenire a mano ed eseguire i comandi appropriati. Naturalmente questo è impegnativo, lento, poco pratico e rischia di fallire. Sono state sviluppate varie tecniche per correggere automaticamente le tabelle di routing in caso di problemi sulla rete quando ci siano percorsi alternativi. Tutte queste tecniche sono raggruppate sotto il nome di "protocolli dinamici di instradamento".

Può essere capitato a tutti di sentir nominare i protocolli dinamici più usati. I più comuni probabilmente sono RIP (Routing Information Protocol) e OSPF (Open Shortest Path First). Il protocollo RIP è molto comune nelle reti piccole, come le reti di organizzazioni di dimensione medio-piccola, o reti situate in un unico edificio. OSPF è più moderno e più in grado di gestire grosse configurazioni di rete, e pure più adatto ad ambienti dove esistono molti percorsi possibili attraverso la rete. Le implementazioni più diffuse di questi protocolli sono '*routed*' (per RIP), e '*gated*' (per RIP, OSPF e altri protocolli). Il programma '*routed*' di solito fa parte delle distribuzioni di Linux, oppure si può trovare incluso nel pacchetto 'NetKit' descritto all'inizio.

Un esempio di dove e quando serva usare un protocollo di instradamento dinamico potrebbe somigliare al seguente:

```
    192.168.1.0 /                         192.168.2.0 /
       255.255.255.0                         255.255.255.0
     -                                     -
     |                                     |
     |   /-----\                 /-----\   |
     |   |     |ppp0   //    ppp0|     |   |
eth0 |---|  A  |------//---------|  B  |---| eth0
     |   |     |     //          |     |   |
     |   \-----/                 \-----/   |
     |      \ ppp1            ppp1 /       |
     -       \                    /        -
              \                  /
               \                /
                \              /
                 \            /
                  \          /
                   \        /
                    \      /
                     \    /
                  ppp0\   /ppp1
                     /-----\
                     |     |
                     |  C  |
                     |     |
                     \-----/
                        |eth0
                        |
                   |---------|
                   192.168.3.0 /
                      255.255.255.0
```

Ci sono qui tre router: A, B e C. Ognuno di essi è connesso ad una ethernet che porta una rete di classe C (netmask 255.255.255.0). Ogni router ha anche una connessione PPP verso ognuno degli altri due router. La rete forma un triangolo.

Dovrebbe essere chiaro che la tabella di routing per il router A dovrebbe essere presente qualcosa come:

```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# route add -net 192.168.2.0 netmask 255.255.255.0 ppp0
root# route add -net 192.168.3.0 netmask 255.255.255.0 ppp1
```

Questa situazione funzionerebbe bene, finché non si interrompesse il collegamento tra il router A e il router B. Se la connessione si interrompe, la tabella di instradamento mostrata non permetterebbe ai calcolatori sul segmento ethernet A di raggiungere calcolatori sul segmento B, perché i loro pacchetti sarebbero rediretti da A sulla connessione ppp0, che è momentaneamente caduta. I calcolatori sulla rete A possono comunque continuare a parlare ai calcolatori sulla rete C, e questi ultimo possono continuare a parlare con i calcolatori sulla rete B, poiché il collegamento tra B e C è ancora funzionante.

Ma allora, se A può parlare a C, e C può ancora parlare a B, perché non potrebbe A mandare i suoi pacchetti a B attraverso C? Questo è esattamente il tipo di problemi che viene affrontato dai protocolli dinamici come RIP. Se ognuno dei router A, B e C facesse girare un programma demone di instradamento, allora le loro tabelle di routing sarebbero corrette automaticamente per riflettere in nuovo stato della rete ogniqualvolta uno dei collegamenti della rete si interrompesse. Configurare questa rete è semplice: occorre fare solo due cose su ognuno dei router. In questo caso, per A bisogna invocare:

```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# /usr/sbin/routed
```

Il demone '*routed*' trova automaticamente le interfacce di rete attive nel momento in cui viene invocato, e successivamente manda dei messaggi e ascolta le risposte su ogni interfaccia di rete, in modo da poter determinare la routing table ed aggiornarla.

Questo termina la breve spiegazione dell'instradamento dinamico e dove è il caso di usarlo. Se si vogliono più informazioni al proposito occorre riferirsi alle fonti di informazione elencate all'inizio di questo documento.

I punti importanti relativi all'instradamento dinamico sono:

1. È necessario eseguire un demone per gestire un protocollo di instradamento dinamico solo quando la propria macchina Linux ha la possibilità di selezionare tra piu d'un percorso possibile per una stessa destinazione.

2. Il programma di instradamento dinamico modifica automaticamente la tabella di intradamento perché rispecchi i cambiamenti della rete.

3. RIP ben si presta alla gestione di reti piccole e medie.

5.8 Configurazione dei servizi di rete e dei programmi server.

I programmi "server" ed i servizi di rete sono quei programmi che permettono ad un utente remoto di utilizzare la macchina Linux locale. I programmi server attendono le connessioni entranti su di una porta di rete. Le porte sono un mezzo per indirizzare un particolare servizio su di un particolare calcolatore e sono il modo in cui un calcolatore server può distinguere tra una connessione entrante di tipo telnet ed una di tipo ftp. L'utente remoto stabilisce una connessione di rete con la macchina server e il programma server (anche detto "daemon" di rete) che sta attendendo una connessione su quella porta accetta la connessione ed inizia a funzionare. Ci sono due modi di utilizzare i servizi di rete, entrambi usati comunemente nella pratica. Questi modi sono:

standalone (da solo)

. Il programma di rete ascolta una porta specifica di rete e quando avverte una connessione in ingresso la gestisce da solo per fornire il servizio richiesto.

dipendente dal server *inetd*

Il server *inetd* è un programma-demone speciale di rete che si occupa di gestire le connessioni in ingresso. *inetd* fa uso di un file di configurazione che gli dice quale programma deve essere invocato quando viene ricevuta una connessione entrante su una particolare porta. Una porta può essere configurata per entrambi i protocolli, tcp o udp. Le porte stesse sono descritte in un altro file, di cui si parlerà presto.

Ci sono due file importanti che occorre configurare per usare inetd: `/etc/services` assegna dei nomi simbolici ai numeri delle porte, mentre `/etc/inetd.conf` è il file di configurazione per il programma *inetd*.

5.8.1 /etc/services

Il file `/etc/services` è un semplice database che associa ad ogni numero di porta comprensibile alla macchina un nome comprensibile all'uomo. Il suo formato è molto semplice: il file è un testo ciascuna riga del quale rappresenta una voce del database. Ogni voce è composta da tre campi separati da un numero qualunque di spazi bianchi (spazi o caratteri 'tab'). I campi sono:

```
nome      porta/protocollo     alias       # commento
```

nome

è una singola parola che rappresenta il servizio che si sta descrivendo.

porta/protocollo

questo campo è diviso in due sottocampi.

porta

un numero che specifica il numero di porta sulla quale il servizio è reso disponibile. La maggior parte dei servizi comunemente usati hanno un numero assegnato loro. Questi numeri sono descritti nel `RFC-1340`.

protocollo

questo sottocampo è o "`tcp`" o "`udp`".

È importante notare che un valore pari a `18/tcp` è molto differente da `18/udp` e che non ci sono ragioni tecniche per cui un servizio debba esistere su entrambe le porte. Normalmente si usa il buon senso, e il database contiene entrambe le voci solo se un servizio è disponibile sia attraverso `tcp` che attraverso `udp`.

alias

altri nomi che possono essere usati per indicare questo servizio.

Tutto il testo che appare in una riga dopo un carattere '`#`' è trattato come commento.

Un file /etc/services di esempio. Tutte le distribuzioni recenti di Linux contengono un buon file `/etc/services`. Nel caso occorra configurare un calcolatore da zero, questa è una copia del file `/etc/services` fornito con una vecchia distribuzione *Debian* `<http://www.debian.org/>`:

```
# /etc/services:
# $Id: services,v 1.3 1996/05/06 21:42:37 tobias Exp $
#
# Network services, Internet style
#
# Si noti che attualmente la politica dello IANA è di assegnare
# un singolo numero di porta per entrambi TCP e UDP; perciò la maggior
# parte delle voci sono duplicate, anche se il protocollo non funziona
# con UDP.
# Aggiornato dall'RFC 1340, ``Assigned Numbers'' (Luglio 1992).
# Non sono incluse tutte le porte, solo le più comuni.
```

```
tcpmux          1/tcp                               # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
msp             18/tcp                              # message send protocol
msp             18/udp                              # message send protocol
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
ssh             22/tcp                              # SSH Remote Login Protocol
ssh             22/udp                              # SSH Remote Login Protocol
telnet          23/tcp
# 24 - private
smtp            25/tcp          mail
# 26 - unassigned
time            37/tcp          timserver
time            37/udp          timserver
rlp             39/udp          resource            # resource location
nameserver      42/tcp          name                # IEN 116
whois           43/tcp          nicname
re-mail-ck      50/tcp                              # Remote Mail Checking Protocol
re-mail-ck      50/udp                              # Remote Mail Checking Protocol
domain          53/tcp          nameserver          # name-domain server
domain          53/udp          nameserver
mtp             57/tcp                              # deprecated
bootps          67/tcp                              # BOOTP server
bootps          67/udp
bootpc          68/tcp                              # BOOTP client
bootpc          68/udp
tftp            69/udp
gopher          70/tcp                              # Internet Gopher
gopher          70/udp
rje             77/tcp          netrjs
finger          79/tcp
www             80/tcp          http                # WorldWideWeb HTTP
www             80/udp                              # HyperText Transfer Protocol
link            87/tcp          ttylink
kerberos        88/tcp          kerberos5 krb5      # Kerberos v5
kerberos        88/udp          kerberos5 krb5      # Kerberos v5
supdup          95/tcp
# 100 - reserved
hostnames       101/tcp         hostname            # usually from sri-nic
iso-tsap        102/tcp         tsap                # part of ISODE.
csnet-ns        105/tcp         cso-ns              # also used by CSO name server
csnet-ns        105/udp         cso-ns
rtelnet         107/tcp                             # Remote Telnet
rtelnet         107/udp
```

```
pop-2           109/tcp         postoffice      # POP version 2
pop-2           109/udp
pop-3           110/tcp                         # POP version 3
pop-3           110/udp
sunrpc          111/tcp         portmapper      # RPC 4.0 portmapper TCP
sunrpc          111/udp         portmapper      # RPC 4.0 portmapper UDP
auth            113/tcp         authentication tap ident
sftp            115/tcp
uucp-path       117/tcp
nntp            119/tcp         readnews untp   # USENET News Transfer Protocol
ntp             123/tcp
ntp             123/udp                         # Network Time Protocol
netbios-ns      137/tcp                         # NETBIOS Name Service
netbios-ns      137/udp
netbios-dgm     138/tcp                         # NETBIOS Datagram Service
netbios-dgm     138/udp
netbios-ssn     139/tcp                         # NETBIOS session service
netbios-ssn     139/udp
imap2           143/tcp                         # Interim Mail Access Proto v2
imap2           143/udp
snmp            161/udp                         # Simple Net Mgmt Proto
snmp-trap       162/udp         snmptrap        # Traps for SNMP
cmip-man        163/tcp                         # ISO mgmt over IP (CMOT)
cmip-man        163/udp
cmip-agent      164/tcp
cmip-agent      164/udp
xdmcp           177/tcp                         # X Display Mgr. Control Proto
xdmcp           177/udp
nextstep        178/tcp         NeXTStep NextStep       # NeXTStep window
nextstep        178/udp         NeXTStep NextStep       # server
bgp             179/tcp                         # Border Gateway Proto.
bgp             179/udp
prospero        191/tcp                         # Cliff Neuman's Prospero
prospero        191/udp
irc             194/tcp                         # Internet Relay Chat
irc             194/udp
smux            199/tcp                         # SNMP Unix Multiplexer
smux            199/udp
at-rtmp         201/tcp                         # AppleTalk routing
at-rtmp         201/udp
at-nbp          202/tcp                         # AppleTalk name binding
at-nbp          202/udp
at-echo         204/tcp                         # AppleTalk echo
at-echo         204/udp
at-zis          206/tcp                         # AppleTalk zone information
at-zis          206/udp
z3950           210/tcp         wais            # NISO Z39.50 database
z3950           210/udp         wais
ipx             213/tcp                         # IPX
ipx             213/udp
imap3           220/tcp                         # Interactive Mail Access
imap3           220/udp                         # Protocol v3
ulistserv       372/tcp                         # UNIX Listserv
ulistserv       372/udp
#
```

```
# UNIX specific services
#
exec            512/tcp
biff            512/udp         comsat
login           513/tcp
who             513/udp         whod
shell           514/tcp         cmd             # no passwords used
syslog          514/udp
printer         515/tcp         spooler         # line printer spooler
talk            517/udp
ntalk           518/udp
route           520/udp         router routed   # RIP
timed           525/udp         timeserver
tempo           526/tcp         newdate
courier         530/tcp         rpc
conference      531/tcp         chat
netnews         532/tcp         readnews
netwall         533/udp                         # -for emergency broadcasts
uucp            540/tcp         uucpd           # uucp daemon
remotefs        556/tcp         rfs_server rfs  # Brunhoff remote filesystem
klogin          543/tcp                         # Kerberized ‘rlogin’ (v5)
kshell          544/tcp         krcmd           # Kerberized ‘rsh’ (v5)
kerberos-adm    749/tcp                         # Kerberos ‘kadmin’ (v5)
#
webster         765/tcp                         # Network dictionary
webster         765/udp
#
# From ‘‘Assigned Numbers’’:
#
#> Le "porte registrate" non sono controllate dallo IANA e su molti
#> sistemi possono essere usate da ordinari processi dell’utente
#> o programmi eseguiti da utenti non privilegiati.
#
#> Le porte sono usate in TCP [45,106] per dare un nome agli estremi
#> di connessioni logiche che trasportano conversazioni a lungo termine.
#> Al fine di fornire servizi ad anonimi, viene definita
#> una porta di contatto per il servizio. Questa lista specifica
#> la porta usata dal processo server come porta di contatto. Anche se
#> lo IANA non può controllare l’uso di queste porte, registra
#> comunque queste porte e riconosce il loro uso per la convenienza
#> della comunità.
#
ingreslock      1524/tcp
ingreslock      1524/udp
prospero-np     1525/tcp                # Prospero non-privileged
prospero-np     1525/udp
rfe             5002/tcp                # Radio Free Ethernet
rfe             5002/udp                # Actually uses UDP only
bbs             7000/tcp                # BBS service
#
#
# Servizi Kerberos (Progetto Athena/MIT)
# Si noti che questi servizi sono usati da Kerberos versione 4,
# e non sono ufficiali. Chi usa la versione 4 dovrebbe scommentare
# queste voci e commentare quelle per la versione 5 definite più sopra.
```

```
#
kerberos4       750/udp         kdc     # Kerberos (server) udp
kerberos4       750/tcp         kdc     # Kerberos (server) tcp
kerberos_master 751/udp                 # Kerberos authentication
kerberos_master 751/tcp                 # Kerberos authentication
passwd_server   752/udp                 # Kerberos passwd server
krb_prop        754/tcp                 # Kerberos slave propagation
krbupdate       760/tcp         kreg    # Kerberos registration
kpasswd         761/tcp         kpwd    # Kerberos "passwd"
kpop            1109/tcp                # Pop with Kerberos
knetd           2053/tcp                # Kerberos de-multiplexor
zephyr-srv      2102/udp                # Zephyr server
zephyr-clt      2103/udp                # Zephyr serv-hm connection
zephyr-hm       2104/udp                # Zephyr hostmanager
eklogin         2105/tcp                # Kerberos encrypted rlogin
#
# Servizi non ufficiali ma necessari per NetBSD
#
supfilesrv      871/tcp                 # SUP server
supfiledbg      1127/tcp                # SUP debugging
#
# Servizi "Datagram Delivery Protocol"
#
rtmp            1/ddp                   # Routing Table Maintenance Protocol
nbp             2/ddp                   # Name Binding Protocol
echo            4/ddp                   # AppleTalk Echo Protocol
zip             6/ddp                   # Zone Information Protocol
#
# Servizi Debian GNU/Linux
rmtcfg          1236/tcp                # Gracilis Packeten remote config server
xtel            1313/tcp                # french minitel
cfinger         2003/tcp                # GNU Finger
postgres        4321/tcp                # POSTGRES
mandelspawn     9359/udp        mandelbrot      # network mandelbrot

# Local services
```

In realtà il file effettivo è in continua crescita dato che vengono continuamente creati nuovi servizi. Se si teme che la propria copia sia incompleta, si suggerisce di copiare un nuovo `/etc/services` da una distribuzione recente.

5.8.2 /etc/inetd.conf

Il file `/etc/inetd.conf` è il file di configurazione per il server di rete *inetd*. La sua funzione è quella di dire a *inetd* cosa fare quando riceve una richiesta di connessione per un particolare servizio. Bisogna dire ad *inetd* quale server demone di rete far partire per ciascun servizio che si vuole fornire, bisogna anche dire come farlo partire.

Il formato del file è abbastanza semplice: si tratta di un file di testo in cui ogni riga descrive un servizio che si intende offrire. Tutto quello che in una linea segue un segno `#` è ignorato e considerato un commento. Ogni linea contiene sette campi separati da un numero qualsiasi di spazi bianchi (tab o carattere di spazio). Il formato generale è:

```
service  socket_type  proto  flags  user  server_path  server_args
```

service

è il servizio al quale si riferisce questa riga, è cioè uno dei nomi che stanno nel file `/etc/services`.

socket_type

questo campo descrive il tipo di socket cui questa voce si riferisce, i cui valori validi sono: `stream`, `dgram`, `raw`, `rdm`, o `seqpacket`. Questa questione è abbastanza tecnica, ma come regola pratica basti ricordare che quasi tutti i servizi basati su `tcp` usano `stream` e quasi tutti i servizi basati su `udp` usano `dgram`. Solo servizi molto particolari useranno uno degli altri valori.

proto

il protocollo usato da questa voce. Questo deve corrispondere alla voce appropriata di `/etc/services` e sarà di solito `tcp` o `udp`. I servizi basati su "Sun RPC" (Remote Procedure Call) useranno `rpc/tcp` o `rpc/udp`.

flags

ci sono solo due valori possibili per questo campo, che dice a *inetd* se il programma server libera il socket dopo avere iniziato a lavorare. Il campo dice quindi se *inetd* deve far partire un altro server alla prossima richiesta di connessione oppure se deve attendere, assumendo che il processo già in funzione gestisca anche le nuove richieste di connessione. Ancora una volta, questa informazione può essere difficile da ottenere, ma in genere tutti i server `tcp` dovranno avere `nowait` in questo campo, mentre la maggior parte dei server `udp` dovrebbero avere il valore `wait`. Bisogna però fare attenzione alle eccezioni a questa regola, perciò quando non si è sicuri conviene farsi guidare dall'esempio che verrà introdotto a breve.

user

questo campo descrive a quale degli account presenti in `/etc/passwd` deve essere assegnata la proprietà del server di rete che viene fatto partire. Questo campo è spesso utile per proteggersi da possibili problemi di sicurezza. Si può assegnare l'utente `nobody` come proprietario di un server, in modo da minimizzare il danno possibile in caso di compromissione della sicurezza del server di rete. Di solito, comunque, questo campo viene posto a `root`, poiché molti server hanno bisogno dei privilegi del superutente per funzionare correttamente.

server_path

questo campo è il percorso completo (pathname) del programma server che deve essere eseguito in relazione a questo servizio.

server_args

questo campo comprende il resto della riga ed è opzionale. In questo campo si mettono gli argomenti di linea di comando che si intendono passare al programma server quando questo viene lanciato.

Un esempio di /etc/inetd.conf Come per `/etc/services`, tutte le distribuzioni aggiornate di Linux includono un buon file `/etc/inetd.conf` con cui poter lavorare. Qui per completezza riporto il file `/etc/inetd.conf` che appare nella distribuzione *Debian* `<http://www.debian.org/>`.

```
# /etc/inetd.conf:  see inetd(8) for further informations.
#
# Internet server configuration database
#
#
# Modified for Debian by Peter Tobias <tobias@et-inf.fho-emden.de>
#
```

```
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
# Internal services
#
#echo           stream          tcp     nowait          root    internal
#echo           dgram           udp     wait            root    internal
discard stream  tcp     nowait          root    internal
discard dgram           udp     wait            root    internal
daytime stream          tcp     nowait          root    internal
daytime dgram           udp     wait            root    internal
#chargen        stream          tcp     nowait          root    internal
#chargen        dgram           udp     wait            root    internal
time            stream          tcp     nowait          root    internal
time            dgram           udp     wait            root    internal
#
# These are standard services.
#
telnet          stream          tcp     nowait          root    /usr/sbin/tcpd  /usr/sbin/in.telnetd
ftp             stream          tcp     nowait          root    /usr/sbin/tcpd  /usr/sbin/in.ftpd
#fsp            dgram           udp     wait            root    /usr/sbin/tcpd  /usr/sbin/in.fspd
#
# Shell, login, exec and talk are BSD protocols.
#
shell   stream          tcp     nowait          root    /usr/sbin/tcpd  /usr/sbin/in.rshd
login   stream          tcp     nowait          root    /usr/sbin/tcpd  /usr/sbin/in.rlogind
#exec   stream          tcp     nowait          root    /usr/sbin/tcpd  /usr/sbin/in.rexecd
talk    dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.talkd
ntalk   dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.ntalkd
#
# Mail, news and uucp services.
#
smtp    stream          tcp     nowait          root    /usr/sbin/tcpd  /usr/sbin/in.smtpd
#nntp   stream          tcp     nowait          news    /usr/sbin/tcpd  /usr/sbin/in.nntpd
#uucp   stream          tcp     nowait          uucp    /usr/sbin/tcpd  /usr/lib/uucp/uucico
#comsat dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.comsat
#
# Pop et al
#
#pop-2  stream          tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.pop2d
#pop-3  stream          tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.pop3d
#
# `cfinger' is for the GNU finger server available for Debian.  (NOTE: The
# current implementation of the `finger' daemon allows it to be run as `root'.)
#
#cfinger        stream          tcp     nowait          root    /usr/sbin/tcpd  /usr/sbin/in.cfingerd
#finger stream          tcp     nowait          root    /usr/sbin/tcpd  /usr/sbin/in.fingerd
#netstat        stream          tcp     nowait          nobody          /usr/sbin/tcpd  /bin/netstat
#systat stream          tcp     nowait          nobody          /usr/sbin/tcpd  /bin/ps -auwwx
#
# Tftp service is provided primarily for booting.  Most sites
# run this only on machines acting as "boot servers."
#
#tftp   dgram   udp     wait    nobody  /usr/sbin/tcpd  /usr/sbin/in.tftpd
#tftp   dgram   udp     wait    nobody  /usr/sbin/tcpd  /usr/sbin/in.tftpd /boot
#bootps dgram   udp     wait    root    /usr/sbin/bootpd        bootpd -i -t 120
```

```
#
# Kerberos authenticated services (these probably need to be corrected)
#
#klogin        stream         tcp    nowait         root    /usr/sbin/tcpd  /usr/sbin/in.rlogind -k
#eklogin       stream         tcp    nowait         root    /usr/sbin/tcpd  /usr/sbin/in.rlogind -k -x
#kshell        stream         tcp    nowait         root    /usr/sbin/tcpd  /usr/sbin/in.rshd -k
#
# Services run ONLY on the Kerberos server (these probably need to be corrected)
#
#krbupdate     stream tcp     nowait         root    /usr/sbin/tcpd  /usr/sbin/registerd
#kpasswd       stream         tcp    nowait         root    /usr/sbin/tcpd  /usr/sbin/kpasswdd
#
# RPC based services
#
#mountd/1      dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.mountd
#rstatd/1-3    dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rstatd
#rusersd/2-3   dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rusersd
#walld/1       dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rwalld
#
# End of inetd.conf.
ident   stream         tcp    nowait         nobody         /usr/sbin/identd       identd -i
```

5.9 Altri file di configurazione relativi alla rete.

C'è un certo numero di altri file, relativi alla configurazione di rete sotto Linux, ai quali merita dare un'occhiata. Non ci sarà bisogno di modificare questi file, ma vale la pena di descriverli, in modo da sapere cosa contengono e a cosa servono.

5.9.1 /etc/protocols

Il file `/etc/protocols` è un database che associa i numeri identificativi dei protocolli al nomi di ciascun protocollo. Questa informazione viene usata dai programmatori per poter specificare per nome i protocolli all'interno dei programmi, e viene usata da alcuni programmi come *tcpdump* al fine di mostrare nei loro messaggi i nomi di protocollo invece dei numeri. La sintassi del file è:

```
nome-protocollo  numero alias
```

Il file `/etc/protocols` che fa parte della distribuzione *Debian* `<http://www.debian.org/>` è fatto così:

```
# /etc/protocols:
# $Id: protocols,v 1.1 1995/02/24 01:09:41 imurdock Exp $
#
# Internet (IP) protocols
#
#      from: @(#)protocols     5.1 (Berkeley) 4/17/89
#
# Updated for NetBSD based on RFC 1340, Assigned Numbers (July 1992).

ip      0       IP              # internet protocol, pseudo protocol number
icmp    1       ICMP            # internet control message protocol
igmp    2       IGMP            # Internet Group Management
ggp     3       GGP             # gateway-gateway protocol
```

```
ipencap 4       IP-ENCAP        # IP encapsulated in IP (officially ``IP'')
st      5       ST              # ST datagram mode
tcp     6       TCP             # transmission control protocol
egp     8       EGP             # exterior gateway protocol
pup     12      PUP             # PARC universal packet protocol
udp     17      UDP             # user datagram protocol
hmp     20      HMP             # host monitoring protocol
xns-idp 22      XNS-IDP         # Xerox NS IDP
rdp     27      RDP             # "reliable datagram" protocol
iso-tp4 29      ISO-TP4         # ISO Transport Protocol class 4
xtp     36      XTP             # Xpress Tranfer Protocol
ddp     37      DDP             # Datagram Delivery Protocol
idpr-cmtp       39      IDPR-CMTP       # IDPR Control Message Transport
rspf    73      RSPF            # Radio Shortest Path First.
vmtp    81      VMTP            # Versatile Message Transport
ospf    89      OSPFIGP         # Open Shortest Path First IGP
ipip    94      IPIP            # Yet Another IP encapsulation
encap   98      ENCAP           # Yet Another IP encapsulation
```

5.9.2 /etc/networks

Il file `/etc/networks` ha una funzione simile a `/etc/hosts`. Il file è un semplice database che associa i nomi delle reti ai loro indirizzi. Il suo formato è differente da `/etc/hosts` in quanto ci sono solo due campi per riga, che sono codificati come:

```
nome-rete indirizzo-di-rete
```

Per esempio, il file potrebbe assomigliare al seguente:

```
loopnet    127.0.0.0
localnet   192.168.0.0
amprnet    44.0.0.0
```

Quando si usano comandi come *route*, se un indirizzo di destinazione è una rete e quella rete appare in`/etc/networks`, allora il comando mostrerà il nome della rete invece del suo indirizzo.

5.10 Sicurezza di rete e controllo degli accessi.

Vorrei iniziare questa sezione avvisando che la protezione di un calcolatore e di una rete da attacchi malevoli è un'arte complessa. Non mi considero assolutamente un esperto in questo campo: mentre i meccanismi che descrivo in seguito possono essere di aiuto, raccomando a chi prende seriamente il problema della sicurezza di fare qualche ricerca personale sull'argomento. Su Internet ci sono molti buoni documenti al proposito, compreso *Security-HOWTO* `<Security-HOWTO.html>`.

Una importante regola base è: '**Non far girare i servizi che non si intendono usare**'. Molte distribuzioni sono configurate per attivare ogni sorta di servizi, che vengono fatti partire automaticamente all'accensione della macchina. Per assicurare un livello di sicurezza minimale occorre passare in rassegna il proprio `/etc/inetd.conf` e commentare (mettendo un '#' all'inizio della riga) ogni voce relativa a servizi che non si intendono usare. Buoni candidati per questa operazione sono servizi come `shell`, `login`, `exec`, `uucp`, `ftp`, e servizi informativi come `finger`, `netstat` e `systat`.

Ci sono molti tipi di meccanismi di sicurezza e controllo degli accessi; qui descriverò solo i più elementari.

5.10.1 /etc/ftpusers

Il file `/etc/ftpusers` è un semplice meccanismo che permette di negare a certi utenti l'accesso via ftp alla macchina. Il file `/etc/ftpusers` viene letto dal demone ftp server (*ftpd*) quando vengono ricevute delle connessioni ftp in ingresso. Il file è semplicemente una lista di utenti a cui è impedito di collegarsi. Il file assomiglia al seguente:

```
# /etc/ftpusers - users not allowed to login via ftp
root
uucp
bin
mail
```

5.10.2 /etc/securetty

Il file `/etc/securetty` permette di specificare a quali periferiche di tipo `tty` l'utente root può collegarsi. Il file `/etc/securetty` viene letto dal programma di login (di solito */bin/login*). Il file è una lista di nomi di terminali ai quali root può collegarsi, mentre su tutti gli altri non è permesso di collegarsi come superutente:

```
# /etc/securetty - tty's on which root is allowed to login
tty1
tty2
tty3
tty4
```

5.10.3 Il meccanismo di controllo degli accessi *tcpd*.

Il programma *tcpd* che avrete notato in `/etc/inetd.conf` fornisce i meccanismi di controllo degli accessi e di registrazione d'utilizzo (logging) per i servizi che protegge.

Quando viene invocato dal programma *inetd*, *tcpd* legge due file contenenti regole di accesso e di conseguenza permette l'accesso al servizio o lo rifiuta.

tcpd scandisce i file di regole finché non trova una corrispondenza. Se non ci sono corrispondenze valide, si assume che l'accesso sia permesso a tutti. I file che vengono scanditi in sequenza sono `/etc/hosts.allow` e `/etc/hosts.deny`. Li descriverò uno alla volta. Per una descrizione completa di questa funzionalità conviene riferirsi alle pagine del manuale (`hosts_access(5)` è un buon punto di partenza).

/etc/hosts.allow Il file `/etc/hosts.allow` è uno dei file di configurazione del programma */usr/sbin/tcpd*. `hosts.allow` contiene le regole che descrivono a quali calcolatori è permesso accedere ai servizi di questa macchina.

Il formato del file è molto semplice:

```
# /etc/hosts.allow
#
# <lista servizi>: <lista calcolatori> [: comando]
```

`lista servizi`

è una lista delimitata da virgole di nomi di programmi server cui questa regola si applica. Esempi di nomi di server sono `ftpd`, `telnetd` e `fingerd`.

`lista calcolatori`

è una lista delimitata da virgole di nomi di host. Si possono, alternativamente, usare gli indirizzi IP. Si possono anche specificare nomi o indirizzi usando caratteri speciali per indicare gruppi di host. Per esempio, `gw.vk2ktj.ampr.org` corrisponde ad un host, `.uts.edu.au` indica tutti i nomi che terminano con questa stringa, `44.` indica ogni indirizzo IP che inizia con 44. Ci sono alcune parole speciali per semplificare la configurazione, alcune delle quali sono: `ALL` per indicare tutti gli host, `LOCAL` per indicare gli host il cui nome non contiene un '`.`', cioè che sono nello stesso dominio di questa macchina, `PARANOID` indica tutti gli host il cui nome non corrisponde all'indirizzo (cioè nel caso sia in atto un 'name spoofing'). Un'altra parola speciale che risulta utile è `EXCEPT`: permette di specificare una lista con delle eccezioni. Questo caso verrà coperto più avanti da un esempio.

`comando`

è un argomento opzionale. Questo parametro corrisponde al pathname completo di un comando che deve essere eseguito ogni volta che questa regola si applica. Per esempio potrebbe essere un comando che cerchi di identificare chi è collegato sul calcolatore che cerca di connettersi, o un comando che spedisce un messaggio di posta o altri avvertimenti all'amministratore di sistema riguardo al tentativo di connessione. Ci sono un certo numero di estensioni che possono essere incluse nel comando; alcuni esempi tipici sono: `%h` è il nome dell'host che cerca di collegarsi, o il suo indirizzo se il nome non può essere risolto, `%d` è il demone server che viene invocato.

Un esempio:

```
# /etc/hosts.allow
#
# La posta è permessa e chiunque
in.smtpd: ALL
# telnet e ftp sono permessi solo a questo dominio e al mio
# calcolatore di casa
telnetd, ftpd: LOCAL, myhost.athome.org.au
# finger è permesso a tutti, ma tenendo traccia di chi lo usa.
fingerd: ALL: (finger @%h | mail -s "finger from %h" root)
```

/etc/hosts.deny Il file `/etc/hosts.deny` è un file di configurazione del programma */usr/sbin/tcpd*. Il file `hosts.deny` contiene le regole che descrivono a quali calcolatori *non è permesso* di accedere un servizio sul calcolatore locale.

Un semplice esempio potrebbe somigliare a questo:

```
# /etc/hosts.deny
#
# Impedisci l'accesso a tutti i calcolatori con nomi sospetti
ALL: PARANOID
#
# Impedisci l'accesso a tutti i calcolatori
ALL: ALL
```

In realtà la voce `PARANOID` è ridondante perché l'altra voce si riferisce in ogni caso a tutti i calcolatori. L'uso di una di queste due voci potrebbe essere una scelta ragionevole, in base alle specifiche esigenze di controllo degli accessi.

La configurazione più sicura consiste nell'avere un default di `ALL: ALL` esplicito in `/etc/hosts.deny` ed abilitare esplicitamente in `/etc/hosts.allow` i servizi e gli host che si vogliono autorizzare.

5.10.4 /etc/hosts.equiv

Il file `hosts.equiv` viene usato per autorizzare alcuni calcolatori e alcuni utenti ad utilizzare gli account sulla macchina locale senza aver bisogno di fornire una password. Questo è utile in un ambiente sicuro, in cui si possano controllare tutte le macchine, ma è un grosso rischio in altre circostanze. Un calcolatore è sicuro solo tanto quanto lo è il meno sicuro dei calcolatori di cui ci si fida. Per massimizzare la sicurezza è bene non usare il meccanismo di `hosts.equiv` ed incoraggiare gli utenti a non usare nemmeno il file `.rhosts`.

5.10.5 Come configurare correttamente il server *ftp*.

Molti siti sono interessati ad offrire il servizio di ftp anonimo, per permettere ad altre persone di scaricare e depositare dei dati senza bisogno di un account specifico sulla macchina. Se si decide di offrire questo servizio bisogna assicurarsi di configurare correttamente il server *ftp* per l'accesso anonimo. La maggior parte delle pagine del manuale disponibili per `ftpd(8)` descrivono accuratamente come adempiere questo compito, e bisogna assicurarsi di seguire le istruzioni. Un consiglio importante è di non usare una copia del proprio file `/etc/passwd` nella directory `/etc` dell'account anonimo: bisogna assicurarsi di rimuovere tutti i dettagli tranne quelli necessari, altrimenti si diventa vulnerabili alle tecniche di rottura delle password "per forza bruta".

5.10.6 I firewall di rete.

Un eccellente modo per avere una certa sicurezza è impedire ai pacchetti di raggiungere il calcolatore che si intende proteggere. Questa tecnica è discussa in dettaglio nel *Firewall-HOWTO* `<Firewall-HOWTO.html>`, e (in forma concisa) in una sezione successiva di questo documento.

5.10.7 Altri suggerimenti.

Questi sono altri suggerimenti che val la pena di prendere in considerazione, anche se potenzialmente si prestano a guerre di religione.

sendmail

nonostante la sua popolarità, appare sugli annunci di attenzione alla sicurezza con impressionante regolarità. Dipende da voi, ma io preferisco non usarlo [un'ottima alternativa è *postfix*, si trova su *mirror italiano di www.postfix.org* `<http://postfix.linux.it/>` N.d.T.].

NFS e altri servizi di tipo "Sun RPC"

bisogna fare attenzione: esistono un sacco di modi per sfruttare a fini malevoli questi servizi. È difficile trovare un'alternativa a servizi come NFS, ma se questi vengono abilitati bisogna fare estrema attenzione riguardo chi ha il permesso di montare i dischi della propria macchina.

6 Informazioni specifiche ad IP ed Ethernet.

Questa sezione presenta informazioni specifiche a Ethernet e IP. Queste sottosezioni sono state raggruppate assieme perché credo siano le più interessanti della sezione precedentemente intitolata "Informazioni specifiche alle singole tecnologie di rete". Chiunque abbia una rete locale (LAN) dovrebbe poterne beneficiare.

6.1 Ethernet

I nomi dei dispositivi Ethernet sono '`eth0`', '`eth1`', '`eth2`' eccetera. Alla prima scheda di rete riconosciuta del kernel viene assegnato '`eth0`', alle altre eventuali schede viene assegnato un nome in sequenza, nell'ordine in cui vengono riconosciute.

Per default, il kernel Linux effettua il probing di un solo dispositivo Ethernet, è necessario passare degli argomenti dalla linea di comando al kernel per poter forzare il riconoscimento di ulteriori schede.

Per imparare come far lavorare correttamente sotto Linux le proprie schede ethernet, si può far riferimento al *Ethernet-HOWTO* `<Ethernet-HOWTO.html>`.

Quando il kernel è compilato con le opzioni corrette per supportare la propria scheda ethernet, la sua configurazione è facile.

Tipicamente sarà necessario qualcosa di simile (di solito la maggior parte delle distribuzioni lo fa automaticamente, se configurata per supportare la scheda ethernet presente):

```
root# ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
root# route add -net 192.168.0.0 netmask 255.255.255.0 eth0
```

La maggior parte dei driver ethernet sono stati sviluppati da Donald Becker, `becker@CESDIS.gsfc.nasa.gov`.

6.2 EQL - equalizzazione del traffico su linea multipla

Il nome della periferica EQL è '`eql`'. Con la distribuzione standard del kernel si può avere al massimo una periferica EQL per macchina. Il protocollo EQL offre un mezzo per utilizzare connessioni multiple punto-a-punto (come PPP, slip o plip) come se fossero un singolo collegamento TCP/IP. Spesso è più economico usare vari collegamenti a bassa velocità che installare un singolo collegamento ad alta velocità.

Opzioni di compilazione del kernel:

```
Network device support  --->
    [*] Network device support
    <*> EQL (serial line load balancing) support
```

Per utilizzare questo meccanismo occorre che anche la macchina che sta all'altro capo del collegamento supporti EQL. Linux, i "Livingstone Portmaster" e altri recenti server a chiamata telefonica (dial-in server) supportano funzionalità compatibili.

Per configurare EQL occorrono gli strumenti EQL, che si posso prendere da: *sunsite.unc.edu* `<ftp://sunsite.unc.edu/pub/linux/system/Serial/eql-1.2.tar.gz>`.

La configurazione è abbastanza intuitiva. Si inizia configurando l'interfaccia eql. A questa interfaccia, come a qualunque altra interfaccia di rete, viene assegnato un indirizzo IP ed una MTU usando il comando *ifconfig*. Per esempio:

```
ifconfig eql 192.168.10.1 mtu 1006
```

Poi occorre attivare manualmente ognuna delle linee che si intendono usare. Queste possono essere una qualsiasi combinazione di dispositivi di rete punto-a-punto. Come effettuare tali connessioni dipende dal tipo di collegamenti in gioco. Per avere ulteriori informazioni occorre riferirsi alle sezioni appropriate.

Infine occorre associare questi collegamenti seriali alla periferica EQL. Questa operazione si chiama "asservimento" (enslave) e viene effettuata con il comando *eql_enslave* come mostrato qui:

```
eql_enslave eql sl0 28800
eql_enslave eql ppp0 14400
```

Il parametro che viene passato è la velocità stimata, e non ha alcun effetto diretto: viene solo usato dal driver EQL per determinare la suddivisione dei pacchetti, in modo da poter aggiustare con precisione il bilanciamento delle linee mediante la scelta oculata di questo valore.

Per staccare una linea dal dispositivo EQL occorre usare il comando *eql_emancipate*, come qui mostrato:

```
eql_emancipate eql sl0
```

Per aggiungere informazioni di instradamento si fa come per qualunque altro collegamento punto-a-punto, tranne che i percorsi devono riferirsi al dispositivo `eql` piuttosto che ai singoli collegamenti. Di solito si usano comandi come il seguente:

```
root# route add default eql
```

Il driver EQL è stato scritto da Simon Janes, `simon@ncm.com`.

6.3 IP Accounting (per Linux-2.0)

Le caratteristiche di accounting del kernel Linux permettono di raccogliere ed analizzare alcuni dati di utilizzo della rete. I dati raccolti comprendono il numero di pacchetti e il numero di byte accumulati dal momento in cui le cifre sono state azzerate l'ultima volta. Si possono specificare una varietà di regole per raccogliere le cifre in categorie secondo le proprie finalità. Questa possibilità è stata rimossa a partire dal kernel 2.1.102, poiché il firewalling basato su 'ipfwadm' è stato rimpiazzato da 'ipfwchains'.

Opzioni di compilazione del kernel:

```
Networking options  --->
    [*] IP: accounting
```

Dopo aver ricompilato ed installato il kernel occorre usare il comando *ipfwadm* per configurare l'accounting IP. Si può scegliere tra diversi modi di dividere le informazioni di accounting. Ho scelto qui un semplice esempio di cosa può essere utile usare, bisognerebbe leggere la pagina del manuale di *ipfwadm* per avere ulteriori informazioni.

Scenario: una rete ethernet è collegata a Internet tramite una connessione PPP. Sulla ethernet c'è una macchina che offre svariati servizi, e interessa sapere quanto traffico viene generato da ftp e WWW, come pure il traffico totale TCP e UDP.

Si può usare a questo fine un insieme di comandi simile al seguente, riportato come script di shell:

```
#!/bin/sh
#
# Dimentica tutte le regole di accounting
```

```
ipfwadm -A -f
#
# assegna dei riferimenti simbolici
localnet=44.136.8.96/29
any=0/0
# Aggiungi le regole per il segmento ethernet locale
ipfwadm -A in  -a -P tcp -D $localnet ftp-data
ipfwadm -A out -a -P tcp -S $localnet ftp-data
ipfwadm -A in  -a -P tcp -D $localnet www
ipfwadm -A out -a -P tcp -S $localnet www
ipfwadm -A in  -a -P tcp -D $localnet
ipfwadm -A out -a -P tcp -S $localnet
ipfwadm -A in  -a -P udp -D $localnet
ipfwadm -A out -a -P udp -S $localnet
#
# Regole di default
ipfwadm -A in  -a -P tcp -D $any ftp-data
ipfwadm -A out -a -P tcp -S $any ftp-data
ipfwadm -A in  -a -P tcp -D $any www
ipfwadm -A out -a -P tcp -S $any www
ipfwadm -A in  -a -P tcp -D $any
ipfwadm -A out -a -P tcp -S $any
ipfwadm -A in  -a -P udp -D $any
ipfwadm -A out -a -P udp -S $any
#
# Stampa le regole
ipfwadm -A -l -n
#
```

I nomi ‘ftp-data’ e ‘www’ si riferiscono a voci di `/etc/services`. L’ultimo comando stampa tutte le regole di accounting e mostra i totali raccolti.

Un punto importante da sottolineare quando si parla di accounting IP è che **vengono incrementati i totali per tutte le regole che si applicano**, cosicché per ottenere cifre relative alle differenza ci vuole un po’ di matematica. Per esempio, se si vuol sapere quanto traffico non era ftp o www bisogna sottrarre i totali individuali dalla regola che si applica a tutte le porte.

```
root# ipfwadm -A -l -n
IP accounting rules
 pkts bytes dir prot source               destination          ports
    0     0 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> 20
    0     0 out tcp  44.136.8.96/29       0.0.0.0/0            20 -> *
   10  1166 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> 80
   10   572 out tcp  44.136.8.96/29       0.0.0.0/0            80 -> *
  252 10943 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> *
  231 18831 out tcp  44.136.8.96/29       0.0.0.0/0             * -> *
    0     0 in  udp  0.0.0.0/0            44.136.8.96/29       * -> *
    0     0 out udp  44.136.8.96/29       0.0.0.0/0            * -> *
    0     0 in  tcp  0.0.0.0/0            0.0.0.0/0            * -> 20
    0     0 out tcp  0.0.0.0/0            0.0.0.0/0            20 -> *
   10  1166 in  tcp  0.0.0.0/0            0.0.0.0/0            * -> 80
   10   572 out tcp  0.0.0.0/0            0.0.0.0/0            80 -> *
  253 10983 in  tcp  0.0.0.0/0            0.0.0.0/0            * -> *
  231 18831 out tcp  0.0.0.0/0            0.0.0.0/0            * -> *
```

```
0      0 in  udp  0.0.0.0/0            0.0.0.0/0            * -> *
0      0 out udp  0.0.0.0/0            0.0.0.0/0            * -> *
```

6.4 IP Accounting (per Linux-2.2)

Il nuovo accounting IP è gestito mediante 'IP Firewall Chains'. Si può fare riferimento alla *home page di IP chains* <http://www.adelaide.net.au/~rustcorp/ipfwchains/ipfwchains.html> per ulteriori informazioni. Tra le altre cose adesso è necessario usare *ipchains* al posto di `ipfwadm` per configurare le proprie regole. (Da `Documentation/Changes` nei sorgenti più recenti del kernel). [È disponibile IP-Chains mini-HOWTO N.d.T.]

6.5 IP Aliasing

Ci sono alcune applicazioni dove è utile poter configurare diversi indirizzi IP associati ad un unica scheda di rete. I fornitori di accesso ad internet spesso usano questa funzionalità per la personalizzazione delle loro offerte di servizi ftp e WWW per i loro clienti. Si può fare riferimento a 'IP-Alias mini-HOWTO' per maggiori informazioni.

Opzioni di compilazione del kernel:

```
Networking options  --->
    ....
    [*] Network aliasing
    ....
    <*> IP: aliasing support
```

Dopo aver compilato ed installato il kernel con il supporto per l'IP Aliasing, la configurazione è molto semplice. I nomi alternativi (alias) vengono aggiunti a periferiche di rete virtuali associate all'interfaccia di rete fisica. Esiste una semplice convenzione per l'assegnamento dei nomi a queste periferiche, del tipo <`nomePeriferica`>`:`<`numeroPerifericaVirtuale`>, per esempio `eth0:0`, `ppp0:10`. Si noti che l'interfaccia nome:numero può essere configurata solo *dopo* aver configurato l'interfaccia principale.

Assumiamo per esempio di avere una rete ethernet che porta due sottoreti IP contemporaneamente, e che si voglia che una macchina abbia accesso diretto ad entrambe le sottoreti; in questo caso si può usare qualcosa come:

```
root# ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0

root# ifconfig eth0:0 192.168.10.1 netmask 255.255.255.0 up
root# route add -net 192.168.10.0 netmask 255.255.255.0 eth0:0
```

Per rimuovere un alias basta aggiungere un carattere '-' alla fine del suo nome e riferirsi ad esso, così:

```
# ifconfig eth0:0- 0
```

Tutte le regole di instradamento associate a quell'alias verranno rimosse automaticamente.

6.6 IP Firewall (per Linux-2.0)

Il Firewall IP e gli argomenti correlati sono trattati con maggior dettaglio nel *Firewall-HOWTO* `<Firewall-HOWTO.html>`. Le tecniche di firewall permettono di rendere sicura la propria macchina verso gli accessi di rete non autorizzati filtrando i pacchetti: i pacchetti di rete sono accettati oppure no in base agli indirizzi IP di partenza/destinazione. Ci sono tre classi di regole: filtro di ingresso (incoming), di uscita (outgoing) e passante (forwarding). Le regole di ingresso vengono applicate ai pacchetti che vengono ricevuti dalle interfacce di rete, le regole di uscita vengono applicate ai pacchetti che devono essere trasmessi da un'interfaccia. Le regole di filtro passante vengono applicate ai pacchetti che sono stati ricevuti ma non sono destinati a questa macchina, cioè i pacchetti che devono essere instradati.

Opzioni di compilazione del kernel:

```
Networking options  --->
    [*] Network firewalls
    ....
    [*] IP: forwarding/gatewaying
    ....
    [*] IP: firewalling
    [ ] IP: firewall packet logging
```

La configurazione delle regole del firewall IP viene effettuata tramite il comando *ipfwadm*. Come ho accennato prima, non sono un esperto nel campo della sicurezza informatica; perciò, mentre l'esempio che sto per presentare è utilizzabile, consiglio di fare le proprie ricerche e sviluppare le proprie regole se la sicurezza è un problema importante.

Probabilmente l'uso più comune del firewall IP si ha quando la propria macchina Linux è usata come router e come filtro per proteggere la propria rete locale dall'accesso non autorizzato dall'esterno della rete.

La configurazione seguente è basata su di un contributo di Arnt Gulbrandsen, <`agulbra@troll.no`>.

L'esempio descrive la configurazione delle regole di firewall nella macchina Linux rappresentata in figura:

```
-                                  -
 \                                 | 172.16.37.0
  \                                |   /255.255.255.0
   \                 ---------     |
    |  172.16.174.30 | Linux |     |
NET =================|  f/w  |------|    ..37.19
    |    PPP         | router|     |  --------
   /                 ---------     |--| Mail |
  /                                |  | /DNS |
 /                                 |  --------
-                                  -
```

I comandi seguenti risiederanno probabilmente in un file nella directory `rc` in modo da essere eseguiti automaticamente tutte le volte che il sistema viene avviato. Ai fini della massima sicurezza, i comandi dovrebbero essere eseguiti dopo aver configurato le interfacce di rete, ma prima di attivarle, in modo da impedire a chiunque di accedere al calcolatore mentre sta riavviandosi.

```
#!/bin/sh

# Azzera la tabella di 'Forward'
# Cambia il comportamento di default perché accetti i pacchetti.
```

```
#
/sbin/ipfwadm -F -f
/sbin/ipfwadm -F -p accept
#
# .. e lo stesso per le regole di entrata.
#
/sbin/ipfwadm -I -f
/sbin/ipfwadm -I -p accept

# Prima di tutto, chiudere l'interfaccia PPP
# Vorrei usare '-a deny' invece di '-a reject -y', ma non sarebbe
# possibile creare delle connessioni da questa interfaccia.
# Il -o fa sì che tutti pacchetti rifiutati siano registrati sul log.
# Questo spreca spazio su disco ma lascia informazione in casi di
# attacco o errore di configurazione.
#
/sbin/ipfwadm -I -a reject -y -o -P tcp -S 0/0 -D 172.16.174.30

# Rigetta subito i pacchetti chiaramente costruiti a scopo malevolo:
# niente dovrebbe provenire da indirizzi multicast/anycast/broadcast
#
/sbin/ipfwadm -F -a deny -o -S 224.0/3 -D 172.16.37.0/24
#
# e nulla proveniente dalla rete loopback deve apparire su un cavo
# di rete
/sbin/ipfwadm -F -a deny -o -S 127.0/8 -D 172.16.37.0/24

# Accetta connessioni SMTP e DNS entranti, ma solo verso il server
# di posta e di risoluzione dei nomi (name server)
#
/sbin/ipfwadm -F -a accept -P tcp -S 0/0 -D 172.16.37.19 25 53
#
# Il DNS usa anche UDP oltre a TCP, quindi bisogna permetterlo
# per le interrogazioni al nostro name server
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 -D 172.16.37.19 53
#
# ma non autorizzare risposte provenienti da porte
# pericolose, come NFS e le sue estensioni a cura di Larry MCVoy.
# Se si usa squid, aggiungere qui la sua porta.
#
/sbin/ipfwadm -F -a deny -o -P udp -S 0/0 53 \
          -D 172.16.37.0/24 2049 2050

# risposte per le altre porte non privilegiate vanno bene
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 53 \
          -D 172.16.37.0/24 53 1024:65535

# Rifiuta le connessioni entranti per identd
# Usare 'reject' così alla connessione viene notificato subito
# di non continuare. Altrimenti avremmo dei ritardi mentre identd
# aspetta il time out.
#
/sbin/ipfwadm -F -a reject -o -P tcp -S 0/0 -D 172.16.37.0/24 113
```

```
    # Accetta le connessioni per alcuni servizi comuni dalle reti
# 192.168.64 e 192.168.65: sono amici e ci fidiamo.
    #
    /sbin/ipfwadm -F -a accept -P tcp -S 192.168.64.0/23 \
              -D 172.16.37.0/24 20:23

    # accetta e ritrasmetti tutto quello che viene dall'interno
    #
    /sbin/ipfwadm -F -a accept -P tcp -S 172.16.37.0/24 -D 0/0

    # impedisci la maggior parte delle altre connessioni TCP entranti, e
    # registrale sul log di sistema.
    # (occorre aggiungere 1:1023 se ftp non funziona)
    #
    /sbin/ipfwadm -F -a deny -o -y -P tcp -S 0/0 -D 172.16.37.0/24

    # ... e lo stesso per UDP
    #
    /sbin/ipfwadm -F -a deny -o -P udp -S 0/0 -D 172.16.37.0/24
```

Una buona configurazione del firewall è abbastanza difficile da raggiungere. Questo esempio dovrebbe essere un punto di partenza ragionevole. La pagina di manuale di *ipfwadm* offre un po' di assistenza nell'uso del programma. Se si intende configurare un firewall occorre essere sicuri di chiedere e di recuperare il maggior numero possibile di informazioni da fonti che si considerino affidabili. Occorre anche che qualcuno verifichi la configurazione dall'esterno.

6.7 IP Firewall (per Linux-2.2)

Il nuovo firewall è gestito mediante 'IP Firewall Chains'. Si può fare riferimento alla *home page di IP chains* `<http://www.adelaide.net.au/~rustcorp/ipfwchains/ipfwchains.html>` per ulteriori informazioni. Tra le altre cose adesso è necessario usare *ipchains* al posto di `ipfwadm` per configurare le proprie regole. (Da `Documentation/Changes` nei sorgenti più recenti del kernel). [È disponibile IP-Chains mini-HOWTO N.d.T.]

6.8 Incapsulazione IPIP

Perché si dovrebbe aver bisogno di incapsulare i pacchetti IP in altri pacchetti IP? Deve sembrare una cosa molto strana se non si è mai vista prima una sua applicazione. Ok, ecco un paio di esempi di uso abbastanza comune dell'incapsulazione: gli indirizzi mobili (mobile-IP) e il multicast. Quello che probabilmente è l'uso più comune di questa tecnica, anche se probabilmente il meno noto, è la radio amatoriale.

Opzioni di compilazione del kernel:

```
Networking options  --->
    [*] TCP/IP networking
    [*] IP: forwarding/gatewaying
    ....
    <*> IP: tunneling
```

Le periferiche tunnel sono chiamate '`tunl0`', '`tunl1`' eccetera.

"Ma perché?". Ecco: l'instradamento convenzionale dei pacchetti IP richiede che una rete IP comprenda un indirizzo di rete ed una maschera di rete. Questo produce una serie di indirizzi contigui che possono essere instradati collettivamente da una singola voce di instradamento. Questo è molto comodo, ma significa che un particolare indirizzo può essere usato solo mentre si è connessi alla rete cui quell'indirizzo appartiene. Nella maggior parte di casi questo va bene, ma se si è un utente mobile della rete può essere difficile collegarsi sempre nello stesso posto. L'incapsulazione IP/IP (o tunneling IP) permette di scavalcare questa restrizione permettendo ai pacchetti destinati all'indirizzo IP dell'utente mobile di essere reimpacchettati e rediretti ad un altro indirizzo IP. Se si sa di doversi collegare ad un altro indirizzo IP per un certo tempo, si può predisporre un calcolatore sulla propria rete di appartenenza perché accetti i pacchetti per il vecchio indirizzo IP e li ridiriga all'indirizzo che si utilizzerà temporaneamente.

6.8.1 Una configurazione di rete a "tunnel"

Come sempre, credo che un diagramma possa evitare un sacco di testo poco chiaro, perciò eccone qui uno:

```
 192.168.1/24                        192.168.2/24

     -                                  -
     |      ppp0 =           ppp0 =      |
     |  aaa.bbb.ccc.ddd  fff.ggg.hhh.iii |
     |                                   |
     |   /-----\                /-----\  |
     |   |     |       //       |     |  |
     |---|  A  |------//--------|  B  |---|
     |   |     |     //         |     |  |
     |   \-----/                \-----/  |
     |                                   |
     -                                  -
```

Il diagramma mostra un'altra ragione possibile per usare l'incapsulazione IP/IP: le reti private virtuali. Questo esempio presuppone che si abbiano due macchine, ciascuna con una semplice connessione telefonica alla rete, e ogni calcolatore ha un solo indirizzo IP. Dietro a queste due macchine ci sono delle reti locali private, configurate con gli indirizzi di rete riservati a tal fine. Supponiamo di voler permettere a tutti i calcolatori della rete A di collegarsi a qualsiasi calcolatore della rete B, come se fossero connessi normalmente a Internet tramite una regola di instradamento. L'incapsulazione IPIP permette di fare questo. Si noti che l'incapsulazione non risolve il problema di come far parlare le reti A e B con gli altri calcolatori di internet: per fare questo occorrono altri trucchi, come il mascheramento. L'incapsulazione di solito viene effettuata dalle macchine che funzionano da router.

Il router Linux 'A' dovrà essere configurato con uno script come il seguente:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
mask=255.255.255.0
remotegw=fff.ggg.hhh.iii
#
# configurazione Ethernet
ifconfig eth0 192.168.1.1 netmask $mask up
route add -net 192.168.1.0 netmask $mask eth0
#
# configurazione ppp0  (avvio del collegamento punto-a-punto,
# configura il percorso di default)
pppd
```

```
route add default ppp0
#
# configurazione della periferica Tunnel
ifconfig tunl0 192.168.1.1 up
route add -net 192.168.2.0 netmask $mask gw $remotegw tunl0
```

Il router ‘B’ dovrà essere configurato con uno script simile:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
mask=255.255.255.0
remotegw=aaa.bbb.ccc.ddd
#
# configurazione Ethernet
ifconfig eth0 192.168.2.1 netmask $mask up
route add -net 192.168.2.0 netmask $mask eth0
#
# configurazione ppp0  (avvio del collegamento punto-a-punto,
# configura il percorso di default)
pppd
route add default ppp0
#
# configurazione della periferica Tunnel
ifconfig tunl0 192.168.2.1 up
route add -net 192.168.1.0 netmask $mask gw $remotegw tunl0
```

Il comando:

```
route add -net 192.168.1.0 netmask $mask gw $remotegw tunl0
```

significa: ‘Manda tutti i pacchetti destinati a `192.168.1.0/24` incapsulati entro un datagramma IPIP con indirizzo di destinazione `aaa.bbb.ccc.ddd`’.

Si noti che le configurazioni delle due macchine sono simmetriche. Il dispositivo tunnel usa l’opzione ‘`gw`’ nell’informazione di instradamento come *destinazione* del pacchetto IP incapsulante. Tale macchina deve sapere come de-capsulare i pacchetti IPIP, in altre parole deve anch’essa essere configurata con un dispositivo tunnel.

6.8.2 Una configurazione di calcolatore con tunnel

Non occorre aver bisogno di instradare un’intera rete. Si può per esempio instradare un singolo indirizzo IP. In questo caso si configurerà l’interfaccia `tunl` sulla macchina remota con il suo proprio indirizzo IP, mentre all’estremo A si userà un normale instradamento di host (usando Proxy Arp), piuttosto che un’instradamento di rete attraverso l’interfaccia tunnel. Si deve ridisegnare e modificare a questo fine la configurazione precedente. Adesso c’è un solo calcolatore ‘B’ che vuole comportarsi come se fosse completamente connesso a Internet e anche parte della rete supportata dall’host ‘A’:

```
192.168.1/24

    -
```

```
|      ppp0 =               ppp0 =
|  aaa.bbb.ccc.ddd      fff.ggg.hhh.iii
|
|   /-----\                /-----\
|   |     |       //       |     |
|---|  A  |------//---------|  B  |
|   |     |     //         |     |
|   \-----/                \-----/
|                      anche: 192.168.1.12
-
```

Il router 'A' sarà così configurato tramite lo script:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
mask=255.255.255.0
remotegw=fff.ggg.hhh.iii
#
# configurazione Ethernet
ifconfig eth0 192.168.1.1 netmask $mask up
route add -net 192.168.1.0 netmask $mask eth0
#
# configurazione ppp0  (avvio del collegamento punto-a-punto,
# configura il percorso di default)
pppd
route add default ppp0
#
# configurazione della periferica Tunnel
ifconfig tunl0 192.168.1.1 up
route add -host 192.168.1.12 gw $remotegw tunl0
#
# Proxy ARP per l'host remoto
arp -s 192.168.1.12 xx:xx:xx:xx:xx:xx pub
```

Il calcolatore Linux 'B' sarà configurato con:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
#
# configurazione ppp0  (avvio del collegamento punto-a-punto,
# configura il percorso di default)
pppd
route add default ppp0
#
# configurazione della periferica Tunnel
ifconfig tunl0 192.168.1.12 up
route add -net 192.168.1.0 netmask $mask gw $remotegw tunl0
```

Questo tipo di configurazione è tipico per le applicazioni di Mobile-IP. In questo caso un singolo calcolatore vuole spostarsi in Internet e mantenere un singolo indirizzo IP per tutto il tempo. Nella sezione su Mobile-IP ci sono ulteriori informazioni su come affrontare in pratica questo problema.

6.9 Mascheramento IP (IP Masquerade) per Linux-2.0

Molte persone si connettono a Internet attraverso un semplice accesso telefonico. Quasi tutti quelli che hanno questo tipo di configurazione hanno solo un indirizzo IP destinato a loro dall'Internet Provider. Questo è di solito sufficiente per collegare una sola macchina alla rete. Il mascheramento IP (IP masquerading) è un trucco intelligente che permette di avere molte macchine che usino un singolo indirizzo IP, facendo in modo che i calcolatori aggiuntivi sembrino (da cui il termine "mascheramento") quello che ha il collegamento telefonico. C'è però un piccolo problema, ed è che la funzione di mascheramento funziona quasi sempre in una sola direzione, per cui i calcolatori mascherati possono chiamare all'esterno ma non possono accettare connessioni dall'esterno. Questo vuol dire che alcuni servizi di rete (ad esempio *talk*) non funzionano mentre altri, tra cui *ftp*, devono essere configurati per operare in modo passivo (PASV) per poter essere usati. Per fortuna i servizi più comuni, come *telnet*, WWW e *irc* funzionano perfettamente.

Opzioni di compilazione del kernel:

```
Code maturity level options  --->
    [*] Prompt for development and/or incomplete code/drivers
Networking options  --->
    [*] Network firewalls
    ....
    [*] TCP/IP networking
    [*] IP: forwarding/gatewaying
    ....
    [*] IP: masquerading (EXPERIMENTAL)
```

Normalmente si avrà la propria macchina Linux collegata tramite una linea telefonica con slip o PPP, come se fosse un normale calcolatore isolato. Inoltre avrà un'altra periferica di rete attiva, probabilmente un'interfaccia ethernet, configurata usando uno degli indirizzi di rete riservati. I calcolatori da mascherare saranno su questa seconda rete. Ognuno di questi calcolatori userà l'indirizzo ethernet della macchina Linux configurato come proprio router.

Una tipica configurazione potrebbe apparire come la seguente:

```
-                                   -
 \                                  | 192.168.1.0
  \                                 |   /255.255.255.0
   \                ---------       |
    |               | Linux | .1.1 |
NET =================| masq  |------|
    |    PPP/slip    | router|      |  --------
   /                ---------       |--| host |
  /                                 |  |      |
 /                                  |  --------
-                                   -
```

I comandi più importanti per questa configurazione sono i seguenti:

```
# Instradamento per la ethernet
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# Instradamento di default per il resto della rete.
root# route add default ppp0
#
# Facciamo in modo che tutte le macchine sulla rete 192.168.1/24
```

```
# siano mascherate
root# ipfwadm -F -a m -S 192.168.1.0/24 -D 0.0.0.0/0
```

Se siete minimalisti e non avete voglia di pigiare troppi tasti, come me, e la macchina mascherante ha due sole interfacce (per cui ogni pacchetto che vi transita dev'essere mascherato), il seguente comando sarà sufficiente:

```
root# /sbin/ipfwadm -F -a accept -m
```

Si possono trovare ulteriori informazioni sulla funzionalità di mascheramento IP di Linux su *pagina dell'IP Masquerade* `<http://www.hwy401.com/achau/ipmasq/>`. Inoltre un *ben* dettagliato documento circa il mascheramento è 'IP-Masquerade mini-HOWTO'. Fornisce anche istruzioni su come configurare altri sistemi operativi per utilizzare un server di mascheramento linux).

6.10 IP Transparent Proxy

Il "proxy trasparente" è una funzionalità che permette di ridirigere dei server o dei servizi destinati ad un'altra macchina verso server e servizi su questa stessa macchina. Di solito questo è utile quando si usa un calcolatore Linux come router che fa anche da server proxy. Si vuole in questo caso ridirigere tutte le connessioni dirette ad un dato servizio su macchine remote verso il server locale.

Opzioni di compilazione del kernel:

```
Code maturity level options  --->
        [*] Prompt for development and/or incomplete code/drivers
Networking options  --->
        [*] Network firewalls
        ....
        [*] TCP/IP networking
        ....
        [*] IP: firewalling
        ....
        [*] IP: transparent proxy support (EXPERIMENTAL)
```

La configurazione delle capacità di proxy trasparente si effettua usando il comando *ipfwadm*.

Un esempio che potrebbe rivelarsi utile è il seguente:

```
root# ipfwadm -I -a accept -D 0/0 telnet -r 2323
```

Questo esempio fa sì che ogni tentativo di connettersi alla porta `telnet` (23) di qualunque host venga rediretto verso la porta 2323 di questo calcolatore. Se viene fatto girare un servizio su tale porta, risulta possibile fare il forward delle connessioni telnet, registrarle o fare qualsivoglia altra operazione che soddisfi le proprie necessità.

Un esempio più interessante consiste nel redirigere tutto il traffico `http` verso una cache locale. D'altra parte, il protocollo usato dai server proxy è diverso dall'http nativo: mentre un client che si connette a `www.server.com:80` chiederà la pagine `/path/page`, un client che si connette al proxy locale contatta `proxy.local.domain:8080` e richiede `www.server.com/path/page`.

Per filtrare una richiesta `http` attraverso il proxy locale, occorre adattare il protocollo, tramite l'inserimento di un piccolo server, chiamato `transproxy` (è possibile trovarlo sulla rete). Si può decidere di far andare `transproxy` sulla porta 8081 e usare il seguente comando:

```
root# ipfwadm -I -a accept -D 0/0 80 -r 8081
```

Il programma `transproxy` quindi riceverà tutti i pacchetti che dovrebbero raggiungere server esterni alla rete locale e li passerà al proxy locale sistemando le differenze di protocollo.

6.11 IPv6

Proprio nel momento in cui si credeva di aver iniziato a capire come funzionano le reti IP, le regole sono cambiate! IPv6 è l'abbreviazione usata per riferirsi alla versione 6 del protocollo internet. IPv6 è stato sviluppato principalmente per risolvere i timori della comunità Internet riguardo alla prossima saturazione dello spazio di indirizzi IP. Gli indirizzi IPv6 sono lunghi 16 byte (128 bit). IPv6 incorpora un certo numero di altri cambiamenti, principalmente semplificazioni, che renderanno le reti IPv6 più gestibili di quelle IPv4.

Linux contiene un'implementazione funzionante, ma non completa, del protocollo IPv6 nella serie `2.1.*` dei kernel.

Chi vuole sperimentare questa nuova generazione di tecnologia Internet, o ha bisogno di essa, dovrebbe leggere il documento IPv6-FAQ, diponibile presso *www.terra.net* `<http://www.terra.net/ipv6/>`.

6.12 IP mobile

L'espressione "Mobile-IP" descrive l'abilità di un calcolatore di muovere la propria connessione di rete da un punto di Internet ad un altro senza cambiare il proprio indirizzo IP e senza perdere la connettività. Di solito, quando un calcolatore IP cambia il suo punto di connessione deve anche cambiare indirizzo IP. La mobilità IP risolve questo problema allocando un indirizzo IP fisso per il calcolatore mobile e usando l'incapsulazione IP (il tunneling) con instradamento automatico, per assicurarsi che i pacchetti destinati a tale calcolatore siano instradati all'indirizzo IP che sta usando al momento.

C'è un progetto attivo al fine di fornire un insieme completo di strumenti per la mobilità IP sotto Linux. Lo stato attuale del progetto e gli strumenti sviluppati si possono trovare alla *home page del Mobile-IP per Linux* `<http://anchor.cs.binghamton.edu/~mobileip/>`.

6.13 Multicast

Il multicast IP permette ad un numero arbitrario di host IP su reti IP diverse di avere instradati verso di loro simultaneamente i pacchetti IP. Questo meccanismo viene usato per distribuire su Internet materiale "broadcast", come le trasmissioni audio e video, o altre applicazioni innovative.

Opzioni di compilazione del kernel:

```
Networking options  --->
        [*] TCP/IP networking
        ....
        [*] IP: multicasting
```

È richiesto a tal fine un pacchetto di programmi e un minimo sforzo di configurazione. Si può fare riferimento a *Multicast-HOWTO* `<Multicast-HOWTO.html>` per maggiori informazioni sul supporto al multicast in Linux.

6.14 NAT - Network Address Translation

La funzionalità di traduzione degli indirizzi di rete è in qualche modo il fratello maggiore standardizzato del mascheramento IP di Linux. Si possono trovare i dettagli nel RFC-1631 in un qualunque archivio di RFC.

Il NAT offre delle funzionalità non fornite dal mascheramento dei pacchetti; queste capacità aggiuntive lo rendono molto più adatto all'uso nei progetti di router che facciano da firewall per gruppi di aziende e per installazioni su larga scala.

Una implementazione di prova del NAT per Linux 2.0.29 è stata sviluppata da Michael Hasenstein, `Michael.Hasenstein@informatik.tu-chemnitz.de`. La documentazione e l'implementazione di Michael si possono recuperare dalla *pagina web del 'Linux IP Network Address'* `<http://www.csn.tu-chemnitz.de/HyperNews/get/linux-ip-nat.html>`

I più recenti kernel 2.1.* includono parte della funzionalità NAT negli algoritmi di instradamento.

6.15 Il 'Traffic Shaper' - Come cambiare l'ampiezza di banda assegnata

Il 'traffic shaper' (limitatore di banda) è un driver per creare nuovi dispositivi di rete. Tali dispositivi sono caratterizzati da una limitazione del traffico consentito in base all'utente. Si basano sulle interfacce fisiche di rete per l'effettiva trasmissione e possono essere usati come percorsi di instradamento in uscita per il traffico di rete.

Il 'traffic shaper' è stato introdotto a partire da Linux 2.1.15 e portato sul Linux-2.0.36 a partire dalla patch `2.0.36-pre-patch-2` distribuita da Alan Cox autore del driver e manutentore di Linux-2.0.

Il 'traffic shaper' può essere compilato solo come modulo e viene configurato tramite il programma *shapecfg* con comandi simili ai seguenti:

```
shapecfg attach shaper0 eth1
shapecfg speed shaper0 64000
```

Tale dispositivo può controllare solo l'ampiezza di banda del traffico in uscita, dato che i pacchetti sono trasmessi attraverso lo 'shaper' secondo le tabelle di instradamento. Dunque una funzionalità di "instradamento secondo l'indirizzo sorgente" potrebbe aiutare a limitare lo spreco di banda da parte di specifici host usando un router Linux.

Linux-2.1 supporta già tale tipo di instradamento, se è necessario su un Linux-2.0 si può utilizzare la patch di Mike McLagan, presso `ftp.invlogic.com`. Si può fare riferimento a `Documentation/networking/shaper.txt` per ulteriori informazioni sul 'traffic shaper'.

Se si desidera provare un limitatore (sperimentale) per i pacchetti in entrata, c'è `rshaper-1.01` (o una versione più recente), da *ftp.systemy.it* `<ftp://ftp.systemy.it/pub/develop>`.

6.16 Instradamento (routing) in Linux-2.2

Le versioni più recenti di Linux-2.1 offrono un sacco di flessibilità nella gestione dell'instradamento. Sfortunatamente dovete aspettare la prossima versione di questo howto o andare a leggere i sorgenti del kernel.

6.17 ISDN

La "rete digitale di servizi integrati" (Integrated Services Digital Network - ISDN) consiste in una serie di standard che definiscono una rete di trasmissione dati a commutazione di circuito per uso generale. Una "chiamata" ISDN crea un servizio di trasmissione dati punto-a-punto sincrono verso la destinazione. La rete ISDN in genere passa su una connessione ad alta velocità che viene poi suddivisa in un numero di canali discreti (i "canali B"), che portano effettivamente i dati dell'utente, e un "canale D" che viene usato per mandare le informazioni di controllo ai commutatori ISDN, per effettuare le chiamate e altre funzioni. In Australia, per esempio, ISDN può viaggiare su una connessione a 2 megabit per secondo che viene suddiviza in 30 canali B discreti da 64 kilobit per secondo e un canale D sempre da 64 kilobit. Qualunque numero di canali può essere usato in ogni momento e in ogni combinazione. Si possono per esempio effettuare 30 chiamate a 30 destinazioni diverse, ciascuna di queste a 64 Kbit, oppure 15 chiamate a 15 destinazioni, ciascuna da 128 Kbit (due canali per chiamata), oppure si può fare un numero minore di chiamate e lasciare il resto delle linee inattive. Un canale può essere usato per chiamate entranti o uscenti. L'intenzione originale di ISDN era quella di permettere alle aziende di telecomunicazione di fornire un singolo servizio dati che potesse portare sia la comunicazione telefonica (tramite la digitalizzazione della voce) sia i servizi dati verso le case e gli uffici degli utenti senza richiedere speciali modifiche alla configurazione.

Ci sono diversi modi per connettere il proprio computer ad un servizio ISDN. Un modo è quello di usare una periferica chiamata "Adattatore di Terminale", che si inserisce nell'"Unità di Terminazione di Rete" che viene installata dalla propria compagnia di telecomucazioni quando consegna il servizio ISDN. Tale Adattatore di Terminale offre in uscita svariate interfacce seriali. Una di queste interfacce viene usata per dare comandi al fine di effettuare le chiamate e la configurazione, mentre le altre sono connesse alle interfacce di rete che, una volta connesse, useranno i circuiti dati. In questa situazione Linux lavora senza alcun bisogno di modifiche, basta trattare la porta dell'adattatore di terminale come se fosse una qualsiasi interfaccia seriale. Un'alternativa, che è quella per cui il supporto ISDN di Linux è stato progettato, è installare una scheda ISDN nella propria macchina Linux e permettere al software di sistema di gestire i protocolli ed effettuare le chiamate.

Opzioni di compilazione del kernel:

```
ISDN subsystem  --->
        <*> ISDN support
        [ ] Support synchronous PPP
        [ ] Support audio via ISDN
        < > ICN 2B and 4B support
        < > PCBIT-D support
        < > Teles/NICCY1016PC/Creatix support
```

L'implementazione Linux di ISDN supporta un certo numero di schede ISDN diverse. Queste sono quelle elencate nelle opzioni di configurazione del kernel:

- ICN 2B and 4B
- Octal PCBIT-D
- Schede ISDN Teles e compatibili

Alcune di queste schede richiedono, per funzionare, che venga loro scaricato del software aggiuntivo. C'è un programma separato che svolge questa funzione.

Maggiori dettagli su come configurare il supporto ISDN per Linux sono disponibili nella directory `/usr/src/linux/Documentation/isdn/`. Una FAQ (Frequently Asked Questions) dedicata a *isdn4linux* (ISDN per Linux) è disponibile presso *www.lrz-muenchen.de* `<http://www.lrz-muenchen.de/~ui161ab/www/isdn/>`. (Cliccando sulla bandiera inglese si ottiene la versione inglese).

Una nota su PPP. La suite di protocolli PPP funziona sia su linee seriali asincrone che sincrone. Il server PPP distribuito normalmente per Linux '*pppd*' funziona solo in modo asincrono. Se si vuole far girare i protocolli PPP sul proprio sevizio ISDN occorre una versione modificata a tal fine. I dettagli su dove trovare tale programma fanno parte della documentazione presentata qui sopra.

6.18 PLIP per Linux-2.0

I nomi delle periferiche PLIP sono ‘`plip0`’, ‘`plip1` e `plip2`.

Opzioni di compilazione del kernel:

```
Networking options  --->
    <*> PLIP (parallel port) support
```

plip (Parallel Line IP), è come SLIP per il fatto che viene usato per avere una connessione di rete punto-a-punto tra due macchine, solo che è progettato per usare la porta parallela dei calcolatori invece della porta seriale (lo schema del cavo è incluso più avanti in questo stesso documento). Siccome è possibile trasmettere più di un bit per volta con la porta parallela, con *plip* si può ottenere una velocità di trasferimento dati più alta di quella che si ottiene con la porta seriale. Inoltre, anche le porte parallele più semplici possono essere usate, mentre per usare le porte seriali a velocità accettabili occorre comperare delle UART 16550AFN che sono relativamente più care. PLIP usa molto tempo macchina in confronto ad una connessione seriale e molto probabilmente non è una buona soluzione per chi può trovare delle schede ethernet economiche. PLIP ha però il grosso vantaggio di funzionare quando non si ha altro a disposizione, e di farlo abbastanza bene. Ci si può aspettare una velocità di trasferimento dati di 20 kB per secondo quando la connessione funziona a regime.

Il driver PLIP è in competizione con il driver per la stampante parallela per quanto riguarda l’accesso alla periferica fisica. Se si desidera utilizzare entrambi i driver bisogna compilarli sotto forma di modulo per poter scegliere quali interfacce parallele dedicare a PLIP e quali alla stampante. Si veda il *Modules-HOWTO* `<Modules-HOWTO.html>` per avere ulteriori informazioni riguardo alla configurazione dei moduli.

Si noti che alcuni portatili usano delle porte che non funzionano con PLIP perché non permettono alcune combinazioni di segnali necessarie per il funzionamento di PLIP, ma che non vengono usate dalle stampanti.

L’interfaccia *plip* di Linux è compatibile con il packet-driver PLIP della *Crynwyr*, questo significa che si può connettere una macchina DOS ad una macchina Linux tramite *plip*, a patto che la macchina DOS abbia qualche tipo di software tcp/ip.

Nella serie 2.0 di kernel Linux le periferiche PLIP sono mappate sulle porte di I/O e sulle linee di interruzione in questo modo:

```
dispositivo  indirizzo I/O    IRQ
-----------  -------------    -----
   plip0         0x3BC          5
   plip1         0x378          7
   plip2         0x278          2
```

Se le porte parallele usate non corrispondono alle combinazioni qui sopra, si può cambiare il numero di interrupt associato ad una porta tramite il comando *ifconfig* usando il parametro ‘`irq`’. Bisogna assicurarsi di abilitare la generazione di interrupt sulla porta stampante dalla configurazione del BIOS se si vuole usare PLIP. Come alternativa, si possono specificare le opzioni ‘`io=`’ e ‘`irq=`’ sulla linea di comando di *insmod*, se si usano i moduli. Ad esempio:

```
root# insmod plip.o io=0x288 irq=5
```

Il funzionamento di PLIP è controllato tramite due timeout, i cui valori di default probabilmente vanno bene nella maggior parte dei casi. Potrebbe essere necessario incrementare i valori nel caso si usi un elaboratore particolarmente lento, nel qual caso bisogna aumentare il valore di timeout dell'**altro** elaboratore. Esiste un programma, *plipconfig*, che permette di cambiare tali valori dei timer senza dover ricompilare il kernel; tale programma è presente in tutte le maggiori distribuzioni Linux.

Per configurare un interfaccia *plip* è necessario lanciare i seguenti comandi (o aggiungerli ai propri script di inizializzazione):

```
root# /sbin/ifconfig plip1 plip_locale pointopoint plip_remoto
root# /sbin/route add plip_remoto plip1
```

Nell'esempio viene usata la porta all'indirizzo di I/O 0x378; *plip_locale* e *plip_remoto* sono i nomi o gli indirizzi IP usati nella connessione PLIP. Personalmente uso metterli tra le voci del mio `/etc/hosts`:

```
# nomi per plip
192.168.3.1   plip_locale
192.168.3.2   plip_remoto
```

Il parametro *pointopoint* ha lo stesso significato che in SLIP, specifica cioè l'indirizzo della macchina all'altro capo della connessione.

In quasi tutti i casi si può trattare un'interfaccia *plip* come fosse un'interfaccia *SLIP*, eccetto che né *dip* né *slattach* possono o debbono venir usati.

Ulteriori informazioni su PLIP si possono trovare in 'PLIP mini-HOWTO'.

6.19 PLIP per Linux-2.2

Durante lo sviluppo delle versioni 2.1 del kernel, il supporto per la porta parallela è stato migliorato.

Opzioni di compilazione del kernel:

```
General setup  --->
    [*] Parallel port support
Network device support  --->
    <*> PLIP (parallel port) support
```

Il nuovo codice per PLIP si comporta come quello vecchio (usa gli stessi comandi *ifconfig* e *route* visti nella sezioni precedente) ma l'inizializzazione della periferica è diversa a causa del supporto per le porte parallele avanzate.

In modo simile a quanto accade per le schede Ethernet, il primo dispositivo PLIP riconosciuto dal kernel viene chiamato 'plip0'. La porta parallela effettivamente utilizzata è una di quelle disponibili, come mostrato in `/proc/parport`. Ad esempio se sul proprio sistema è presente una sola porta parallela, ci sarà una sola sottodirectory: `/proc/parport/0`.

Nel caso il proprio kernel non riconosca da sé la linea di interrupt usata dalla porta, il comando '`insmod plip`' non andrà a buon fine. In questo caso sarà sufficiente scrivere il valore corretto in `/proc/parport/0/irq` e lanciare nuovamente *insmod*.

Informazioni esaurienti sulla gestione della porta parallela sono disponibili nel file `Documentation/parport.txt` nei sorgenti del kernel.

6.20 PPP

I nomi delle periferiche PPP sono ‘`ppp0`’, ‘`ppp1` eccetera. I dispositivi sono numerati sequenzialmente, ove il primo di essi riceve il numero ‘`0`’.

Opzioni di compilazione del kernel:

```
Networking options  --->
    <*> PPP (point-to-point) support
```

La configurazione di PPP è coperta ad un buon livello di dettaglio nel documento *PPP-HOWTO* `<PPP-HOWTO.html>`.

6.20.1 Gestire una connessione permanente ad internet con *pppd*.

Se si è abbastanza fortunati da avere una connessione semi-permanente con internet e si desidera che la propria macchina ristabilisca automaticamente la connessione PPP quando questa viene interrotta, ecco un semplice trucco per riuscirci.

Configurare PPP in modo che possa essere avviato dall'utente `root` tramite il seguente comando:

```
# pppd
```

Ci si assicuri di avere attivato l'opzione ‘`-detach`’ nel proprio file `/etc/ppp/options`. Poi si inserisca la linea seguente nel proprio file `/etc/inittab`, in fondo, dove si trovano le definizioni per *getty*:

```
pd:23:respawn:/usr/sbin/pppd
```

Questa linea fa sì che il programma *init* faccia partire il programma *pppd* e lo controlli, facendolo ripartire automaticamente nel caso in cui termini.

6.21 SLIP client

Le periferiche SLIP sono chiamate ‘`sl0`’, ‘`sl1`’ eccetera, dove alla prima periferica viene assegnato lo ‘`0`’, e alle altre i numeri successivi, nell'ordine in cui vengono configurate.

Opzioni di compilazione del kernel:

```
Network device support  --->
    [*] Network device support
    <*> SLIP (serial line) support
    [ ]  CSLIP compressed headers
    [ ]  Keepalive and linefill
    [ ]  Six bit SLIP encapsulation
```

SLIP (Serial Line Internet Protocol) permette di usare tcp/ip su una linea seriale, sia si tratti di una linea telefonica con collegato un modem, o una linea dedicata di qualche altro tipo. Naturalmente, per usare SLIP occorre accedere ad uno *SLIP-server* nella propria area. Molte università e molte aziende in tutto il mondo offrono accesso tramite SLIP.

SLIP usa la porta seriale del calcolatore per trasportare pacchetti IP. Per fare questo deve prendere controllo dell'interfaccia seriale. Già sappiamo che le periferiche SLIP si chiamano *sl0*, *sl1* eccetera, ma come corrispondono questi nomi a quelli delle porte seriali? Il codice di rete usa una chiamata *ioctl* (I/O control) per trasformare una porta seriale in una periferica SLIP. Ci sono due programmi che svolgono questo compito, chiamati *dip* e *slattach*.

6.21.1 dip

dip (Dialup IP) è un simpatico programma che può assegnare la velocità della porta seriale, dire al modem di chiamare l'altro estremo della connessione, autenticarsi nel server remoto e ascoltare i messaggi mandati dal server al fine di estrarre informazioni quale l'indirizzo IP. Il programma invoca poi le chiamate *ioctl* necessarie per trasformare l'interfaccia seriale in una porta SLIP. *dip* ha una potente funzionalità di "scripting", ed è questo che viene sfruttato per automatizzare la procedura di collegamento.

Si può trovare il programma su: *sunsite.unc.edu* `<ftp://sunsite.unc.edu/pub/Linux/system/Network/serial/dip/dip337o-uri.tgz>`.

Per installarlo, usare questi comandi:

```
user% tar xvzf dip337o-uri.tgz
user% cd dip-3.3.7o
user% vi Makefile
root# make install
```

Il `Makefile` assume l'esistenza di un gruppo chiamato *uucp*, ma questo può essere impostato a *dip* oppure a *SLIP*, in base alla propria configurazione.

6.21.2 slattach

slattach, a differenza di *dip*, è un programma molto semplice, facile da usare, ma non così sofisticato come *dip*. Non ha la capacità di "scripting", e tutto quello che fa è configurare la porta seriale come interfaccia SLIP. Il programma assume che si abbia tutta l'informazione necessaria, e che la comunicazione seriale sia già stata stabilita prima di invocare *slattach*. Questo programma è ideale da usare quando si ha una connessione permanente con il proprio server, come un cavo fisico o una linea dedicata.

6.21.3 Come scegliere se usare l'uno o l'altro?

Conviene usare *dip* quando il collegamento verso il server SLIP è un modem telefonico, o qualche altro tipo di collegamanto temporaneo. Conviene usare *slattach* quando si ha una linea dedicata, o un cavo fisico, tra la propria macchina e il server, perché in questi casi non occorre nessuna azione speciale per far funzionare il collegamento. Vedere la sezione "connessione SLIP permanente" per avere ulteriori informazioni.

La configurazione di SLIP è molto simile alla configurazione di una interfaccia ethernet (vedere la sezione 'Configurazione di un interfaccia ethernet', più sopra). Nonostante ciò, ci sono alcune differenze chiave.

Prima di tutto, le connessioni SLIP sono diverse dalle reti ethernet, in quanto ci sono sempre solo due calcolatori sulla rete, uno ad ogni estremo della connessione. A differenza della ethernet che è disponibile all'uso non appena passati i cavi, con SLIP, a seconda del tipo di collegamento che si usa, può essere necessario inizializzare la propria connessione di rete in qualche modo speciale.

Se si usa *dip*, questo non verrà solitamente effettuato all'avvio della macchina, ma qualche tempo dopo, quando si è pronti ad usare la connessione ed è possibile automatizzare questa procedura. Se si usa *slattach*, probabilmente si vorrà aggiungere una sezione al proprio file *rc.inet1*. Questo verrà descritto tra poco.

Ci sono due tipi principali di server SLIP: quelli che forniscono un indirizzo IP dinamico, e quello che lo forniscono statico. Quasi tutti i server SLIP chiederanno il nome utente e la password quando viene stabilita la connessione. *dip* può gestire automaticamente l'autenticazione.

6.21.4 SLIP server statico con linea telefonica e DIP.

Un server SLIP statico è quello con il quale si riceve un indirizzo IP che è solo proprio. Ogni volta che ci si connette con il server, la porta viene configurata con lo stesso indirizzo. Il server statico risponderà alla chiamata del modem, probabilmente chiederà nome utente e password, e poi instraderà tutti i pacchetti destinati all'indirizzo corrispondente attraverso quella connessione. Se si ha un server statico, probabilmente si desidererà mettere in `/etc/hosts` le associazioni tra nome della macchina e indirizzo IP (che sarà noto). Si dovrebbero anche sistemare altri file, cioè: `rc.inet2`, `host.conf`, `resolv.conf`, `/etc/HOSTNAME` ed `rc.local`. Si ricordi che quando si configura `rc.inet1` non occorre aggiungere alcun comando speciale per la connessione SLIP, poiché *dip* fa già tutto il lavoro impegnativo di configurare l'interfaccia. Occorrerà dare a *dip* le informazioni appropriate e lui configurerà l'interfaccia da solo dopo aver detto al modem di stabilire la chiamata e dopo aver autenticato il chiamante presso il server SLIP.

Se il server SLIP che si usa funziona così, allora si può passare direttamente alla sezione 'Uso di Dip' per imparare come configurare il programma correttamente.

6.21.5 SLIP server dinamico con linea telefonica e DIP.

Un server SLIP *dinamico* è quello che assegna un indirizzo IP casualmente, da un insieme di indirizzi possibili, tutte le volte che ci si collega. Questo significa che non c'è alcuna garanzia che si avrà un particolare indirizzo ogni volta. Significa anche che lo stesso indirizzo può essere usato da qualcun altro dopo che si è terminata la connessione. L'amministratore di rete che ha configurato il server SLIP avrà ricevuto un gruppo di indirizzi da usare per il server: quando il server riceve una nuova chiamata sceglie il primo numero non utilizzato, guida l'utente nella procedura di autenticazione e poi stampa un messaggio di benvenuto che contiene l'indirizzo IP da usarsi per la durata della chiamata.

La configurazione per usare questo tipo di server è simile alla configurazione nel caso di un server statico, tranne per il fatto che occorre aggiungere un passo, durante il quale si ottiene l'indirizzo IP che il server ha scelto per questa sessione e si configura la periferica SLIP con quell'indirizzo.

Ancora una volta, *dip* si occupa dei dettagli laboriosi. Le nuove versioni sono abbastanza furbe da recuperare automaticamente l'indirizzo IP dal messaggio di benvenuto e salvarlo per configurare l'interfaccia SLIP, in aggiunta a gestire il processo di autenticazione.

Se il server SLIP che si usa funziona così, allora si può passare direttamente alla sezione 'Uso di Dip' per imparare come configurare il programma correttamente.

6.21.6 Uso di DIP.

Come spiegato in precedenza, *dip* è un programma potente che può semplificare ed automatizzare il processo di chiamare il server SLIP, autenticarsi, iniziare la connessione e invocare i comandi *ifconfig* e *route* appropriati per la propria interfaccia.

Fondamentalmente, per usare *dip* bisogna scrivere uno "script", che è in pratica una lista di comandi comprensibili a *dip* che dicono al programma come eseguire ogni azione che si vuole esegua. Si veda il file `sample.dip` che viene distribuito con *dip* per avere un'idea di come funziona. *dip* è un programma abbastanza potente, con molte opzioni; piuttosto che descriverle tutte qui, si consiglia di guardare la pagina del manuale, il file README e gli esempi che fanno parte del pacchetto *dip*.

Si noterà che lo script `sample.dip` assume che si sta usando un server SLIP statico, cioè che si conosca il proprio indirizzo IP in anticipo. Per quando si usa un server dinamico, le versioni più recenti di *dip* includono un comando che si può usare per leggere automaticamente l'indirizzo che il server dinamico ha assegnato e configurare di conseguenza la periferica SLIP. L'esempio seguente è una versione modificata del file `sample.dip` che viene distribuito con *dip337j-uri.tgz*, ed è probabilmente un buon punto di partenza. Si può salvare questo script come `/etc/dipscript` e modificarlo per rispecchiare la propria configurazione.

```
#
# sample.dip    Programma di supporto alla connessione telefonica.
#
#       Questo file mostra (dovrebbe mostrare) come usare DIP
#       Questo file dovrebbe funzionare con server dinamici tipo "Annex".
#       Se si usa un server statico, usare il file "sample.dip" che
#       è distribuito con il pacchetto dip337-uri.tgz.
#
#
# Version:      @(#)sample.dip  1.40    07/20/93
#
# Author:       Fred N. van Kempen, <waltje@uWalt.NL.Mugnet.ORG>
#

main:
# Predisporre il nome e indirizzo dell'altro estremo.
# La mia macchina remota si chiama 'xs4all.hacktic.nl' (== 193.78.33.42)
get $remote xs4all.hacktic.nl
# Assegnare la netmask su sl0 a 255.255.255.0
netmask 255.255.255.0
# Assegnare la porta specificata e la velocità.
port cua02
speed 38400

# Reinizializza il modem e la linea del terminale
# Questo causa problemi ad alcune persone!
reset

# Nota: i valori di errore predefiniti sono:
#  0 - OK
#  1 - CONNECT
#  2 - ERROR
#
# Si possono cambiare cercando "addchat()" usando "grep" su *.c...

# Prepara la chiamata
send ATQ0V1E1X4\r
wait OK 2
if $errlvl != 0 goto modem_trouble
dial 555-1234567
if $errlvl != 1 goto modem_trouble

# Siamo connessi. Autentichiamoci.
login:
sleep 2
wait ogin: 20
if $errlvl != 0 goto login_trouble
send MYLOGIN\n
```

```
wait ord: 20
if $errlvl != 0 goto password_error
send MYPASSWD\n
loggedin:

# Adesso siamo autenticati.
wait SOMEPROMPT 30
if $errlvl != 0 goto prompt_error

# Ordiniamo al server di andare in modo SLIP
send SLIP\n
wait SLIP 30
if $errlvl != 0 goto prompt_error

# Recuperiamo l'indirizzo IP dal server
#   Si assume che dopo aver detto al server di passare in SLIP, questo
#   stampi il nostro indirizzo.
get $locip remote 30
if $errlvl != 0 goto prompt_error

# Assegnamo i parametri operativi SLIP
get $mtu 296
# Assicuriamoci di dare "route add -net default xs4all.hacktic.nl"
default

# Salutiamo e via!
done:
print CONNECTED $locip ---> $rmtip
mode CSLIP
goto exit

prompt_error:
print TIME-OUT waiting for sliplogin to fire up...
goto error

login_trouble:
print Trouble waiting for the Login: prompt...
goto error

password:error:
print Trouble waiting for the Password: prompt...
goto error

modem_trouble:
print Trouble occurred with the modem...
error:
print CONNECT FAILED to $remote
quit

exit:
exit
```

L'esempio precedente assume che si stia chiamando un server SLIP dinamico. Se si chiama un server statico, allora il file `sample.dip` del pacchetto *dip337j-uri.tgz* dovrebbe funzionare senza alcuna modifica.

Quando a *dip* viene passato il comando *get $local*, il programma cerca una stringa che assomigli ad un indirizzo IP all'interno del testo che arriva dall'altra estremità della connessione; cerca cioè delle stringhe di numeri separate dal carattere '.'. Questa modifica è stata inserita specificamente per i server SLIP dinamici, in modo da automatizzare il processo di lettura dell'indirizzo IP fornito dal server.

L'esempio precedente creerà automaticamente una regola di instradamento di default attraverso la connessione SLIP. Se questo non è quello che si desidera, per esempio perché si vuole avere un instradamento di default sulla propria ethernet, allora occorre rimuovere il comando *default* dallo script. Dopo che questo script ha terminato l'esecuzione, chi provasse ad invocare *ifconfig* vedrà che una periferica *sl0* esiste nel sistema: si tratta della porta SLIP. Se occorre, si può modificare la configurazione di tale interfaccia manualmente, dopo che il comando *dip* ha terminato di girare, usando i comandi *ifconfig* e *route*.

Si noti che *dip* permette di scegliere un certo numero di protocolli usando il comando `mode`, il più comune esempio al proposito è *cSLIP*, per abilitare la compressione di SLIP. Si noti che entrambi gli estremi della connessione devono essere d'accordo, e bisogna assicurarsi che qualunque protocollo si scelga questo corrisponda con quello che viene attivato dal server.

L'esempio precedente è abbastanza robusto, e dovrebbe gestire correttamente la maggior parte degli errori. Si faccia riferimento alla pagina del manuale di *dip* per avere ulteriori informazioni. Si può, ovviamente, fare di più, come scrivere uno script che faccia cose come richiamare il server se non riesce a collegarsi entro un certo tempo limite, o provare una lista di server, se si ha accesso a più di uno di essi.

6.21.7 Connessione SLIP permanente con linea dedicata e slattach.

Se si possiede un cavo che collega due macchine, o se si è abbastanza fortunati da avere una linea dedicata o qualche altro tipo di connessione seriale permanente tra la propria macchina e un'altra, allora non occorre incontrare tutte le difficoltà relative all'uso di *dip* al fine di preparare la propria connessione. *slattach* è un programma molto semplice da usare che offre una funzionalità sufficiente a configurare la propria connessione.

Siccome la connessione sarà permanente, si vorranno aggiungere alcuni comandi al proprio file `rc.inet1`. In sintesi, tutto quello che bisogna fare per avere una connessione permanente è assicurarsi di configurare la periferica seriale alla velocità corretta, e far passare la seriale alla modalità SLIP. *slattach* permette di fare questo lavoro con un comando solo. Le seguenti linee vanno **aggiunte** al proprio file `rc.inet1`:

```
#
# Attiva una connessione SLIP statica su linea dedicata
#
#  configura /dev/cua0 per 19.2kbps e cslip
/sbin/slattach -p cslip -s 19200 /dev/cua0 &
/sbin/ifconfig sl0 IPA.IPA.IPA.IPA pointopoint IPR.IPR.IPR.IPR up
#
# Fine configurazione SLIP statico.
```

Dove:

IPA.IPA.IPA.IPA

rappresenta il proprio indirizzo IP.

IPR.IPR.IPR.IPR

rappresenta l'indirizzo IP del calcolatore remoto.

slattach alloca la prima periferica SLIP disponibile all'interfaccia seriale specificata. *slattach* parte da *sl0*, quindi il primo comando *slattach* connette la periferica SLIP *sl0* alla porta seriale specificata. La seguente invocazione collegherà *sl1*, eccetera.

slattach permette di configurare uno tra diversi protocolli con l'argomento `-p`. In questo caso viene usato *SLIP* oppure *cSLIP*, a seconda se si voglia usare la compressione o no. Nota: i due capi della connessione devono essere d'accordo se usare o meno la compressione.

6.22 Server SLIP.

Se si possiede una macchina, magari connessa in rete, alla quale si vuole che altri si connettano telefonicamente per accedere ai propri servizi di rete, allora occorre configurare la propria macchina come un server. Se si vuole usare SLIP come protocollo seriale, ci sono attualmente tre possibilità per configurare la propria macchina come server SLIP. Personalmente preferirei la prima possibilità che sto per introdurre (*sliplogin*), in quanto sembra essere la più semplice da configurare e capire. Presenterò in ogni caso un'introduzione a ciascun metodo, in modo da permettere a tutti di fare la propria scelta.

6.22.1 Server SLIP usando *sliplogin*.

sliplogin è un programma che può essere usato al posto della normale shell di login per gli utenti SLIP che devono convertire il terminale seriale in una linea di comunicazione SLIP. Il programma permette di configurare la propria macchina Linux come un *server di indirizzi statici* (dove gli utenti ricevono lo stesso indirizzo IP tutte le volte che si connettono) oppure come un *server di indirizzi dinamici* (dove gli utenti ricevono un indirizzo che puoò non essere lo stesso della volta precedente).

Il chiamante si collegherà come si fa per il processo standard di login: fornendo il proprio nome e utente e la password; ma dopo aver autenticato l'utente, invece di una shell il sistema eseguirà *sliplogin*. Il programma cercherà poi nel suo file di configurazione (`/etc/slip.hosts`) una voce che corrisponda al nome di login dell'utente. Se tale voce viene trovata, la linea viene configurata ad 8 bit e la "disciplina di linea" viene convertita a quella di SLIP. Quando questo processo è completo, viene svolta l'ultima parte della configurazione, nella quale *sliplogin* invoca uno script di shell che configura l'interfaccia SLIP con i valori IP appropriati: indirizzo, maschera di rete e informazioni di instradamento. Lo script viene di solito chiamato `/etc/slip.login`, ma si possono creare script personalizzati per gli utenti che hanno bisogno di un'inizializzazione particolare, come si fa con *getty*. Questi script si chiameranno `/etc/slip.login.loginname` e verranno eseguiti al posto di quello di default per gli utenti con esigenze particolari.

Ci sono tre o quattro file che occorre configurare perché *sliplogin* funzioni; mostrerò ora come ottenere il software e come ciascuno di questi file viene configurato. I file coinvolti sono:

- `/etc/passwd`, per gli account degli utenti telefonici.
- `/etc/slip.hosts`, per fornire le informazioni specifiche a ciascun utente.
- `/etc/slip.login`, che gestisce la configurazione dell'instradamento per l'utente.
- `/etc/slip.tty`, necessario solo se il server viene configurato per l'allocazione dinamica degli indirizzi. Tale file contiene una tabella di indirizzi per l'allocazione.
- `/etc/slip.logout`, contiene i comandi per fare pulizia dopo che un utente ha messo giù o si è scollegato.

Dove ottenere *sliplogin* È probabile che il pacchetto *sliplogin* sia già installato come parte della propria distribuzione, se questo non accade, *sliplogin* si può trovare su: *sunsite.unc.edu* `<ftp://sunsite.unc.edu/pub/linux/system/Network/serial/sliplogin-2.1.1.tar.gz>`. Il pacchetto contiene il sorgente, dei binari precompilati e la pagina del manuale.

Per assicurarsi che solo gli utenti autorizzati possano eseguire il programma *sliplogin*, bisognerebbe aggiungere una voce simile alla seguente nel proprio file `/etc/group`:

```
 ..
slip::13:radio,fred
 ..
```

Quando si installa il pacchetto *sliplogin*, il `Makefile` cambierà il gruppo del programma *sliplogin* in modo che sia `slip`. Questo significa che solo gli utenti che appartengono a tale gruppo saranno in grado di eseguire il programma. L'esempio precedente permetterebbe solo agli utenti `radio` e `fred` di eseguire *sliplogin*.

Per installare gli eseguibili nella directory `/sbin` e la pagina del manuale nella sezione 8, si fa così:

```
# cd /usr/src
# gzip -dc .../sliplogin-2.1.1.tar.gz | tar xvf -
# cd sliplogin-2.1.1
# <..si modifichi il Makefile se non si usano le shadow password..>
# make install
```

Se si vogliono ricompilare gli eseguibili prima di installare, si aggiunga un `make clean` prima del `make install`. Se si vogliono installare gli eseguibili in qualche altra directory, occorre modificare la regola *install* del `Makefile`.

Si legga il file `README` del pacchetto per ulteriori informazioni.

Configurazione di `/etc/passwd` per gli host Slip Di solito si creano dei nomi utente speciali in `/etc/passwd` per gli utenti Slip. Una convenzione seguita comunemente consiste nell'usare il nome del calcolatore chiamante preceduta da una 's' maiuscola. Quindi, per esempio, se il calcolatore chiamante si chiama `radio` la voce in `/etc/passwd` sarà simile a questa:

```
Sradio:FvKurok73:1427:1:radio SLIP login:/tmp:/sbin/sliplogin
```

In effetti, non ha nessuna importanza come viene chiamato l'utente; basta che sia significativo per chi amministra il sistema.

Nota: il chiamante non ha bisogno di una directory home, in quanto non usufruirà di un interprete di comandi sulla macchina server, perciò `/tmp` è una buona scelta. Si noti anche che il programma *sliplogin* viene usato al posto della shell di login.

Configurazione di `/etc/slip.hosts` Il file `/etc/slip.hosts` è quello che viene letto da *sliplogin* alla ricerca di voci corrispondenti al nome di login, al fine di ottenere i dettagli di configurazione per questo calcolatore chiamante. Questo è il file in cui si specificano l'indirizzo IP e la maschera di rete da assegnare al chiamante e che saranno configurati per questo uso. Due voci esemplificative per due calcolatori, uno con configurazione statica (`radio`) e uno con configurazione dinamica (`albert`) sono le seguenti:

```
#
Sradio   44.136.8.99   44.136.8.100  255.255.255.0  normal      -1
Salbert  44.136.8.99   DYNAMIC       255.255.255.0  compressed  60
#
```

Le voci in `/etc/slip.hosts` consistono dei seguenti campi:

1. il nome di login del chiamante.

2. l'indirizzo ip del server, cioè di questo calcolatore.

3. l'indirizzo IP da assegnare al chiamante. Se questo campo vale `DYNAMIC`, allora l'indirizzo sarà allocato in base alle informazioni contenute nel file `/etc/slip.tty`, presentato più avanti. **Nota:** occorre usare almeno la versione 1.3 di sliplogin perché l'assegnazione dinamica funzioni.
4. la maschera di rete per la macchina chiamante, in notazione decimale con punti. Per esempio, 255.255.255.0 per una classe C.
5. il modo SLIP, che permette di abilitare o disabilitare la compressione e altre caratteristiche. I valori possibili sono "`normal`" e "`compressed`".
6. Un parametro di timeout che specifica per quanto tempo la linea può rimanere inattiva (senza trasmissione di pacchetti) prima che venga automaticamente scollegata. Un valore negativo disabilita questa funzionalità.
7. argomenti opzionali.

Nota: per i campi 2 e 3 si possono usare sia i nomi degli host che gli indirizzi IP in notazione decimale. Se si usano i nomi, questi nomi devono essere risolubili, altrimenti lo script fallirà quando verrà invocato. Si può verificare se il nome viene risolto provando a fare telnet verso il nome: se si riceve il messaggio '*Trying nnn.nnn.nnn...*', allora il nome viene correttamente risolto; se si riceve il messaggio '*Unknown host*', il nome non viene risolto. Se il nome non viene risolto bisogna usare l'indirizzo in notazione decimale con punti oppure bisogna sistemare la configurazione del risolutore (si veda la sezione sulla risoluzione dei nomi).

I modi SLIP più comuni sono:

normal

per abilitare il normale modo SLIP non compresso.

compressed

per abilitare la compressione degli header con l'algoritmo di van Jacobsen (cSLIP).

Ovviamente, questi due modi sono mutuamente esclusivi: si può usare uno o l'altro. Per ulteriori informazioni sulle opzioni disponibili si vedano le pagine del manuale.

Configurazione del file `/etc/slip.login`. Quando *sliplogin* ha trovato una voce corretta in `/etc/slip.hosts`, proverà ad eseguire il file `/etc/slip.login` per assegnare indirizzo e maschera di rete all'interfaccia SLIP.

Il file `/etc/slip.login` di esempio distribuito con il pacchetto *sliplogin* assomiglia al seguente:

```
#!/bin/sh -
#
#       @(#)slip.login  5.1 (Berkeley) 7/1/90
#
# file di login generico per una linea SLIP
# sliplogin lo invoca con i seguenti parametri:
#     $1       $2       $3     $4, $5, $6 ...
#   SLIPunit velocità  pid   arcomenti da slip.hosts
#
/sbin/ifconfig $1 $5 pointopoint $6 mtu 1500 -trailers up
/sbin/route add $6
arp -s $6 <hw_addr> pub
exit 0
#
```

Si noterà che questo script usa i normali comandi *ifconfig* e *route* per configurare l'indirizzo dell'interfaccia, l'indirizzo remoto e la sua maschera di rete, e per creare un instradamento per il calcolatore remoto attraverso l'interfaccia SLIP. Questi compiti sono gli stessi che vanno svolti se si usa il comando *slattach*.

Si noti anche l'uso del *Proxy ARP* per assicurarsi che altri host sulla stessa ethernet del server siano in grado di raggiungere il calcolatore chiamante. Il campo <`hw_addr`> dovrebbe essere l'indirizzo hardware della scheda ethernet del server. Se il server non è su una rete ethernet questa linea può essere rimossa.

Configurazione del file `/etc/slip.logout`. Quando cade la linea ci si vuole assicurare che la periferica seriale ritorni al suo stato normale, in modo che altri possano collegarsi correttamente. Questo compito viene svolto tramite il file `/etc/slip.logout`. Questo file ha un formato abbastanza semplice e viene chiamato con gli stessi argomenti di `/etc/slip.login`.

```
#!/bin/sh -
#
#               slip.logout
#
/sbin/ifconfig $1 down
arp -d $6
exit 0
#
```

Tutto quello che fa è disattivare l'interfaccia, il che causerà la rimozione delle informazioni di instradamento associate. Lo script usa anche il comando *arp* per rimuovere le informazioni di proxy arp. Ancora una volta, non occorre il comando *arp* nello script se il server non ha una porta ethernet.

Configurazione di `/etc/slip.tty`. Se si usa l'allocazione dinamica degli indirizzi IP (se qualche host è configurato con la parola chiave `DYNAMIC` in `/etc/slip.hosts`), allora bisogna configurare il file `/etc/slip.tty` perché elenchi quali indirizzi sono assegnati alle porte. Questo file occorre solo se si vuole che il proprio server allochi dinamicamente gli indirizzi agli utenti.

Il file è una tabella che elenca le periferiche di tipo *tty* che supportano le connessioni SLIP entranti e gli indirizzi IP che devono essere assegnati agli utenti che si collegano su quelle porte.

Il suo formato è il seguente:

```
# slip.tty    mappatura terminale -> indirizzo IP per lo SLIP dinamico
# formato: /dev/tty?? xxx.xxx.xxx.xxx
#
/dev/ttyS0      192.168.0.100
/dev/ttyS1      192.168.0.101
#
```

Quello che questa tabella dice è che gli utenti che chiamano su `/dev/ttyS0` e che hanno l'indirizzo assegnato a `DYNAMIC` in `/etc/slip.hosts`, devono ricevere l'indirizzo `192.168.0.100`.

In questo modo occorre allocare solo un indirizzo per porta per tutti gli utenti che non hanno bisogno di un indirizzo dedicato. Questo aiuta a tenere al minimo il numero di indirizzi necessari, per evitare sprechi.

6.22.2 Server Slip che usa *dip*.

Lasciatemi dire fin dall'inizio che alcune delle informazioni seguenti vengono dalle pagine del manuale di *dip*, dove è spiegato brevemente come far girare Linux come server SLIP. Bisogna anche fare attenzione al fatto che le informazioni seguenti si basano sul pacchetto *dip337o-uri.tgz*, e probabilmente non si applicano ad altre versioni di *dip*.

dip offre un modo operativo "di input", nel quale trova automaticamente in `/etc/diphosts` la voce corrispondente all'utente che lo ha chiamato, e configura la porta seriale come connessione SLIP in base alle informazioni che trova nel file. Questo modo di inpupt viene attivato chiamando il programma *dip* come *diplogin*. Per usare *dip* come server SLIP, perciò, basta creare degli account che usino *diplogin* come shell.

La prima cosa da fare è creare un link simbolico come segue:

```
# ln -sf /usr/sbin/dip /usr/sbin/diplogin
```

Poi occorre aggiungere le voci ai file `/etc/passwd` ed `/etc/diphosts`. Le voci da creare sono fatte come segue.

Per configurare Linux come server SLIP con *dip*, occorre creare gli account SLIP per gli utenti, dove il comando *dip* è usato in modalità input come shell. Una convenzione suggerita è quella di usare una 'S' maiuscola all'inizio dei nomi di account SLIP.

Una voce esemplificativa in `/etc/passwd` per un utente SLIP è la seguente:

```
Sfredm:ij/SMxiTlGVCo:1004:10:Fred:/tmp:/usr/sbin/diplogin
^^         ^^         ^^  ^^   ^^   ^^   ^^
|          |          |   |    |    |    \__ diplogin come shell
|          |          |   |    |    \_______ directory home
|          |          |   |    \____________ nome dell'utente
|          |          |   \_________________ group ID
|          |          \_____________________ user ID
|          \________________________________ password crittata
\___________________________________________ nome utente slip
```

Dopo che l'utente si è collegato, il programma *login*, dopo aver autenticato l'utente, eseguirà il comando *diplogin*. *dip*, quando viene invocato con il nome di *diplogin* sa che deve assumere di essere usato come shell dell'utente. La prima cosa che fa quando viene invocato come *diplogin* è usare la funzione *getuid()* per sapere la UID dell'utente che l'ha invocato. Poi cerca in `/etc/diphosts` la prima voce che corrisponde o al nome della periferica che ha ricevuto la chiamata oppure al nome utente, e si configura di conseguenza. Si può creare un server che si comporti come server statico per alcuni utenti e come server dinamico per gli altri, creando una voce nel file `diphosts` per i primi e lasciando la configurazione di default per i secondi.

dip, quando chiamato in "modo input" aggiunge automaticamente una voce 'Proxy-ARP', per cui non occorre preocuparsi di svolgere questo compito manualmente.

Configurazione di `/etc/diphosts` `/etc/diphosts` viene usato da *dip* per recuperare le informazioni di configurazione per i calcolatori remoti. Tali calcolatori possono essere utenti che si collegano telefonicamente a questa macchina linux, oppure macchine che vengono chiamate da questa.

Il formato generale per `/etc/diphosts` è il seguente:

```
 ..
Suwalt::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:CSLIP,1006
ttyS1::145.71.34.3:145.71.34.2:255.255.255.0:Dynamic ttyS1:CSLIP,296
 ..
```

I campi sono:

1. `nome di login`: come ritornato da getpwuid(getuid()), o il nome del terminale.
2. `inutilizzato`: compatibile con /etc/passwd
3. `indirizzo remoto`: indirizzo IP dell'host chiamante, numerico o nome
4. `indirizzo locale`: indirizzo IP di questa macchina, numerico o nome
5. `Netmask`: in notazione decimale con punti
6. `Commento`: si scriva quello che si vuole
7. `protocollo`: Slip, CSlip eccetera
8. `MTU`: numero decimale

Un esempio di voce di `/etc/net/diphosts` relativa ad un utente SLIP remoto potrebbe essere:

```
Sfredm::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:SLIP,296
```

che specifica una connessione SLIP con un indirizzo remoto di 145.71.34.1 e una MTU di 296. Oppure:

```
Sfredm::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:CSLIP,1006
```

specifica una connessione SLIP con un indirizzo remoto di 145.71.34.1 e una MTU di 1006.

Perciò, tutti gli utenti ai quali si vuole dare un indirizzo IP statico avranno una voce corrispondente in `/etc/diphosts`, mentre chi deve ricevere un indirizzo dinamico associato alla porta che viene chiamata deve avere una voce associata alla periferica invece che al suo nome. Bisogna ricordarsi di creare almeno una voce per ciascuna periferica `tty` alla quale gli utenti possono connettersi telefonicamente, per assicurarsi che sia sempre disponibile una configurazione appropriata indipendentemente da quale modem riceva la chiamata.

Quando un utente si collegherà, riceverà una normale richiesta di login e password, alla quale risponderà con il proprio nome utente e la propria password. Se questi saranno corretti non si vedranno altri messaggi e sarà sufficiente che ai due capi della connessione si passi in modo SLIP per avere una connessione funzionante configurata con i parametri del file `diphosts`.

6.22.3 Server SLIP che usa il pacchetto *dSLIP*.

Matt Dillon <`dillon@apollo.west.oic.com`> ha scritto un pacchetto che non solo permette le connessioni SLIP entranti, ma anche quelle uscenti. Il pacchetto di Matt è una combinazione di piccoli programmi e di script che gestiscono le connessioni. Occorre avere *tcsh* installata, perché almeno uno degli script usa questa shell. Matt fornisce nel pacchetto una copia binaria dell'utility *expect*, perché anch'essa è richiesta da uno degli script. Probabilmente occorrerà un po' di esperienza con *expect* per far funzionare il pacchetto a proprio piacimento, ma non conviene lasciarsi intimidire da ciò.

Matt ha scritto un buon insieme di istruzioni di installazione nel file README, perciò non le ripeterò qui.

Il pacchetto *dSLIP* si può ottenere dal suo sito ftp:

apollo.west.oic.com

```
/pub/linux/dillon_src/dSLIP203.tgz
```

oppure da:

sunsite.unc.edu

```
/pub/Linux/system/Network/serial/dSLIP203.tgz
```

L'unica attenzione da porre è che occorre leggere il file `README` e creare le voci in `/etc/passwd` ed `/etc/group` **prima** di fare `make install`.

7 Altre Tecnologie di Rete

Le sottosezioni successive contengono informazioni specifiche a particolari tecnologie di rete. Le informazioni contenute in queste sezioni non si applicano necessariamente ad altri tipi di tecnologie di rete. Gli argomenti sono presentati in ordine alfabetico.

7.1 ARCNet

I nomi delle periferiche ARCNet sono '`arc0e`', '`arc1e`', '`arc2e`' eccetera, oppure '`arc0s`', '`arc1s`', '`arc2s`' eccetera. La prima scheda che viene vista dal kernel prende il nome di '`arc0e`' o '`arc0s`' e le altre prendono sequenzialmente gi altri nomi, nell'ordine in cui vengono viste. La lettera finale indica se l'interfaccia usa un incapsulamento dei pacchetti di tipo ethernet o il formato standard secondo l'RFC1051.

Opzioni di compilazione del kernel:

```
Network device support  --->
    [*] Network device support
    <*> ARCnet support
    [ ]   Enable arc0e (ARCnet "Ether-Encap" packet format)
    [ ]   Enable arc0s (ARCnet RFC1051 packet format)
```

Una volta che il kernel è stato propriamente ricompilato per supportare le schede di rete ARCNet, la configurazione delle schede stesse risulta abbastanza semplice.

Normalmente verranno usati comandi come:

```
root# ifconfig arc0e 192.168.0.1 netmask 255.255.255.0 up
root# route add -net 192.168.0.0 netmask 255.255.255.0 arc0e
```

Consiglio di riferirsi ai file `/usr/src/linux/Documentation/networking/arcnet.txt` e `/usr/src/linux/Documentation/networking/arcnet-hardware.txt` per avere ulteriori informazioni sull'argomento.

Il supporto per ARCNet è stato sviluppato da Avery Pennarun, `apenwarr@foxnet.net`.

7.2 Appletalk (AF_APPLETALK)

Il protocollo Appletalk non ha bisogno di nomi particolari per i dispositivi, in quanto usa i dispositivi di rete esistenti.

Opzioni di compilazione del kernel:

```
Networking options  --->
    <*> Appletalk DDP
```

Il supporto per Appletalk permette alle macchine Linux di lavorare insieme alle reti Apple. Un uso importante di questa abilità è la condivisione di risorse, come stampanti e dischi, tra computer Linux e Apple. Per utilizzare Appletalk occorre del software aggiuntivo, che prende il nome di *netatalk*. Wesley Craig `netatalk@umich.edu` è il rappresentante di un gruppo chiamato 'Research Systems Unix Group' che lavora all'Università del Michigan e ha prodotto il pacchetto *netatalk*, contenente software che implementa lo stack di protocolli Appletalk e alcuni utili programmi di servizio. Il pacchetto *netatalk*. se non è supportato dalla vostra distribuzione, deve essere recuperato dal suo sito tramite ftp: *University of Michigan* `<ftp://terminator.rs.itd.umich.edu/unix/netatalk/>`

Per compilare ed installare il pacchetto occorre fare qualcosa come:

```
user% tar xvfz .../netatalk-1.4b2.tar.Z
user% make
root# make install
```

Potrebbe essere utile editare il 'Makefile' prima di chiamare *make* per compilare effettivamente il programma. In particolare potrebbe essere utile cambiare la variabile DESTDIR, che definisce dove verranno successivamente installati i file. Una buona scelta è il valore predefinito /usr/local/atalk.

7.2.1 Configurazione del software Appletalk.

La prima cosa che occorre fare perché tutto funzioni correttamente è assicurarsi che il proprio file `/etc/services` contenga le voci appropriate. Le voci di cui si ha bisogno sono:

```
rtmp    1/ddp   # Routing Table Maintenance Protocol
nbp     2/ddp   # Name Binding Protocol
echo    4/ddp   # AppleTalk Echo Protocol
zip     6/ddp   # Zone Information Protocol
```

Il passo successivo consiste nella creazione dei file di configurazione Appletalk nella directory `/usr/local/atalk/etc` (o in quella in cui il pacchetto è stato installato).

Il primo file da creare è `/usr/local/atalk/etc/atalkd.conf`. Questo file inizialmente deve contenere solo una linea, che specifica il nome dell'interfaccia di rete sulla quale sono raggiungibili le macchine Apple:

```
eth0
```

Il programma-demone Appletalk aggiungerà altri dettagli al file nel momento in cui verrà eseguito.

7.2.2 Esportare un disco tramite Appletalk.

Si possono esportare dei filesystem Linux verso la rete, in modo che le macchine Apple sulla rete possano accedervi.

A questo fine occorre configurare il file `/usr/local/atalk/etc/AppleVolumes.system`. Un altro file di configurazione, chiamato `/usr/local/atalk/etc/AppleVolumes.default`, ha esattamente lo stesso formato e descrive quali filesystem verranno visti dagli utenti che si collegano come "guest".

I dettagli riguardo la configurazione di questi file e le varie opzioni si trovano nella pagina del manuale relativa al comando *afpd*.

Un semplice esempio potrebbe somigliare a:

```
/tmp Scratch
/home/ftp/pub "Area Pubblica"
```

L'esempio mostrato esporta il filesystem `/tmp` come volume AppleShare con il nome di 'Scratch', e la directory pubblica di ftp come volume AppleShare con il nome di 'Area Pubblica'. Specificare i nomi dei volumi non è obbligatorio, in quanto il programma-demone li può assegnare autonomamente, ma specificarli esplicitamente non ha alcuna controindicazione.

7.2.3 Condividere le stampanti Linux usando Appletalk.

E abbastanza semplice condividere una stampante Linux con le macchine Apple. A tal fine occorre far girare il programma *papd*, il demone per il protocollo di accesso alle stampanti Appletalk (Printer Access Protocol). Quando il programma viene fatto girare accetta le richieste da parte delle macchine Apple e passa il lavoro di stampa al server *lpd* locale per la stampa.

Occorre editare il file `/usr/local/atalk/etc/papd.conf` per configurare il servizio. La sintassi del file è la stessa del più noto `/etc/printcap`. Il nome che viene dato ad ogni definizione viene registrato tramite il protocollo di gestione dei nomi Appletalk (NBP).

Una semplice configurazione potrebbe essere:

```
TricWriter:\
   :pr=lp:op=cg:
```

Questo esempio rende disponibile sulla rete Appletalk una stampante chiamata 'TricWriter', e tutti i lavori accettati via rete verranno stampati sulla stampante '`lp`' come definita nel file `/etc/printcap`, usando *lpd*. La voce '`op=cg`' dice che l'utente Linux '`cg`' è l'operatore della stampante.

7.2.4 Far partire il software appletalk.

A questo punto si dovrebbe essere pronti a provare la configurazione di base. Il file *rc.atalk* distribuito con il pacchetto *netatalk* dovrebbe funzionare bene per questa prova, perciò l'unica cosa che resta da fare è:

```
root# /usr/local/atalk/etc/rc.atalk
```

e tutto dovrebbe essere attivo e girare correttamente. Non si dovrebbero vedere messaggi di errore a questo punto, ma il software manderà dei messaggi sulla console per indicare ogni passo che viene fatto partire.

7.2.5 Provare il software appletalk.

Per verificare che il software stia funzionando correttamente occorre andare ad una delle macchine Apple, attivare il menu principale, scegliere "Chooser", poi "AppleShare". A questo punto il calcolatore Linux dovrebbe apparire.

7.2.6 Alcune attenzioni con il software appletalk.

- Potrebbe esserci bisogno di far partire il supporto Appletalk prima di configurare la rete IP. Se ci sono problemi a far partire i programmi Appletalk, o se dopo che essi sono partiti si riscontrano dei problemi sulla rete IP, occorre provare ad attivare il software Appletalk prima di invocare `/etc/rc.d/rc.inet1` file.

- Il server *afpd* (Apple Filing Protocol Daemon) scombina seriamente il disco rigido. Sotto il mount-point, il server crea un paio di directory: `.AppleDesktop` e `Network Trash Folder`. Poi, per ogni directory che viene usata il server crea un `.AppleDouble` al suo interno, per salvare le risorse eccetera. Perciò bisogna pensarci due volte prima di esportare `/`: se alla fine si vuole ripulire il disco ci vuole molto tempo.
- Il programma *afpd* si aspetta delle password in chiaro dai Mac. La sicurezza potrebbe essere un problema, perciò bisogna fare molta attenzione quando si fa girare questo demone su una macchina collegata ad Internet: bisogna solo rimproverare se stessi se qualcuno arreca dei danni alla macchina.
- gli strumenti di diagnostica esistenti, come *netstat* e *ifconfig* non supportano Appletalk. L'informazione non trattata da questi programmi è comunque disponibile nella directory `/proc/net/`, se se ne ha bisogno.

7.2.7 Ulteriori informazioni.

Per una descrizione molto più dettagliata su come configurare Appletalk per Linux occorre riferirsi alla pagina *Linux Netatalk-HOWTO* di Anders Brownworth, presso *thehamptons.com* `<http://thehamptons.com/anders/netatalk/>`.

7.3 ATM

Werner Almesberger <`werner.almesberger@lrc.di.epfl.ch`> sta gestendo un progetto per fornire a Linux il supporto per ATM (Asynchronous Transfer Mode). Informazioni aggiornate sullo stato del progetto possono essere ottenute da: *lrcwww.epfl.ch* `<http://lrcwww.epfl.ch/linux-atm/>`.

7.4 AX25 (AF_AX25)

I nomi di periferica AX.25 nei kernel versione `2.0.*` sono '`sl0`', '`sl1`' eccetera, mentre nei kernel `2.1.*` sono '`ax0`', '`ax1`' eccetera.

Opzioni di compilazione del kernel:

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
```

I protocolli AX25, Netrom e Rose sono ampiamente trattati nel documento *AX25-HOWTO* `<AX25-HOWTO.html>`. Questi protocolli sono usati dagli utenti di radio amatoriali per la sperimentazione di trasmissione radio di pacchetti in tutto il mondo.

La maggior parte del lavoro di implementazione di questi protocolli è stata fatta da Jonathan Naylor, `jsn@cs.nott.ac.uk`.

7.5 DECNet

Al supporto per DECnet si sta lavorando adesso. È plausibile che appaia in uno degli ultimi kernel versione 2.1.

7.6 FDDI

I nomi delle periferiche FDDI sono '`fddi0`', '`fddi1`', '`fddi2`' eccetera. La prima scheda rilevata dal kernel prende il nome '`fddi0`' e alle altre vengono sequenzialmente assegnati gli altri nomi nell'ordine in cui vengono riconosciute.

Larry Stefani, `lstefani@ultranet.com`, ha scritto un driver per le schede FDDI per EISA e PCI della Digital.

Opzioni di compilazione del kernel:

```
Network device support  --->
    [*] FDDI driver support
    [*] Digital DEFEA and DEFPA adapter support
```

Dopo aver ricompilato il kernel per supportare il driver FDDI, la configurazione dell'interfaccia è praticamente identica alla configurazione di una scheda ethernet. Occorre solo specificare il nome dell'interfaccia FDDI nei comandi *ifconfig* e *route*.

7.7 Frame Relay

I nomi delle periferiche Frame Relay sono '`dlci00`', '`dlci01`' eccetera per i dispositivi ad incapsulazione DLCI, mentre '`sdla0`', '`sdla1`' eccetera sono i nomi per le periferiche FRAD.

Frame Relay è una nuova tecnologia di rete, progettata per adattarsi a traffico di comunicazione dati che sia di natura intermittente. Ci si connette ad una rete Frame Relay usando un FRAD (Frame Relay Access Device). Il supporto Frame Relay per Linux usa IP-su-Frame-Relay, come specificato nell'RFC 1490.

Opzioni di compilazione del kernel:

```
Network device support  --->
    <*> Frame relay DLCI support (EXPERIMENTAL)
    (24)   Max open DLCI
    (8)   Max DLCI per device
    <*>   SDLA (Sangoma S502/S508) support
```

Il supporto per Frame Relay e i relativi strumenti di configurazione sono stati sviluppati da Mike McLagan, `mike.mclagan@linux.org`.

Al momento i soli FRAD supportati sono i *Sangoma Technologies* `<http://www.sangoma.com/>` S502A, S502E e S508.

Dopo aver ricompilato il kernel, per configurare le periferiche FRAD e DLCI occorrono gli strumenti di configurazione per Frame Relay. Questi sono disponibili da *ftp.invlogic.com* `<ftp://ftp.invlogic.com/pub/linux/fr/frad-0.15.tgz>`. La compilazione e l'installazione di questi strumenti non presenta alcun problema, ma l'assenza di un Makefile nella directory principale fa sì che il processo sia abbastanza "manuale":

```
user% tar xvfz .../frad-0.15.tgz
user% cd frad-0.15
user% for i in common dlci frad; make -C $i clean; make -C $i; done
root# mkdir /etc/frad
root# install -m 644 -o root -g root bin/*.sfm /etc/frad
root# install -m 700 -o root -g root frad/fradcfg /sbin
root# install -m 700 -o root -g root dlci/dlcicfg /sbin
```

Si noti che i comandi sopracitati usano la sintassi *sh*, se si usa una shell di tipo *csh*, come *tcsh*, il ciclo *for* apparirà diversamente.

Dopo l'installazione di questi programmi occorre creare un file `/etc/frad/router.conf`. Si può usare come esempio il seguente, che è una versione modificata di uno dei file di esempio del pacchetto:

```
# /etc/frad/router.conf
# Questo è uno schema di configurazione per frame relay.
# Tutte le parole chiave sono presenti. I valori di default sono basati
# sul codice fornito con il driver DOS per il Sangoma S502A.
#
# Un '#' all'interno di una linea segna un commento
# Spazi e caratteri TAB sono ignorati
# Voci [] sconosciute e chiavi ignote sono ignorate
#

[Devices]
Count=1                 # numero di periferiche da configurare
Dev_1=sdla0             # nome di periferica
#Dev_2=sdla1            # nome di periferica

# Se specificate qui, queste voci si applicano a tutte le periferiche,
# ma possono essere specificati valori alternativi per ciascuna scheda.
#
Access=CPE
Clock=Internal
KBaud=64
Flags=TX
#
# MTU=1500              # Massima lunghezza del IFrame trasmesso,
                        # il default è 4096
# T391=10               # valore di T391    5 - 30, il default è 10
# T392=15               # valore di T392    5 - 30, il default è 15
# N391=6                # valore di N391    1 - 255, il default è 6
# N392=3                # valore di N392    1 - 10, il default è 3
# N393=4                # valore di N393    1 - 10, il default è 4

# Se specificate qui, queste voci si applicano a tutte le schede
# CIRfwd=16             # CIR forward   1 - 64
# Bc_fwd=16             # Bc forward    1 - 512
# Be_fwd=0              # Be forward    0 - 511
# CIRbak=16             # CIR backward  1 - 64
# Bc_bak=16             # Bc backward   1 - 512
# Be_bak=0              # Be backward   0 - 511


#
#
# Configurazione specifica per ciascuna interfaccia
#
#

#
# La prima periferica è una Sangoma S502E
#
```

```
[sdla0]
Type=Sangoma            # Tipo di periferica. Attualmente viene riconosciuto
                        # solo il nome SANGOMA
#
# Queste chiavi sono specifiche al tipo "Sangoma"
#
# Tipo di scheda - S502A, S502E, S508
Board=S502E
#
# Nome del firmware (ROM) di prova della scheda
# Testware=/usr/src/frad-0.10/bin/sdla_tst.502
#
# Nome del firmware FR
# Firmware=/usr/src/frad-0.10/bin/frm_rel.502
#
Port=360                # Porta di I/O per questa scheda
Mem=C8                  # Indirizzo della finestra di memoria (A0-EE)
IRQ=5                   # Numero di interrupt, non specificare per la S502A
DLCIs=1                 # Numero di DLCI attaccati alla periferica
DLCI_1=16               # Numero del primo DLCI, 16 - 991
# DLCI_2=17
# DLCI_3=18
# DLCI_4=19
# DLCI_5=20
#
# Se specificate qui, queste voci si applicano solo a questa periferica,
# e rimpiazzano i valori specificati prima
#
# Access=CPE            # CPE o NODE, default: CPE
# Flags=TXIgnore,RXIgnore,BufferFrames,DropAborted,Stats,MCI,AutoDLCI
# Clock=Internal        # External o Internal, default: Internal
# Baud=128              # Valore nominale di baud rate per il CSU/DSU
# MTU=2048              # Massima lunghezza del IFrame trasmesso,
                        # il default è 4096
# T391=10               # valore di T391    5 - 30, il default è 10
# T392=15               # valore di T392    5 - 30, il default è 15
# N391=6                # valore di N391    1 - 255, il default è 6
# N392=3                # valore di N392    1 - 10, il default è 3
# N393=4                # valore di N393    1 - 10, il default è 4

#
# La seconda periferica è qualche altro tipo
#
# [sdla1]
# Type=FancyCard        # Tipo di periferica
# Board=                # Tipo di scheda Sangoma
# Key=Value             # Valori specifici a questo tipo di scheda


#
# Valori di default per la configurazione DLCI
# Questo posso essere rimpiazzati nella configurazione specifica
# di ciascun DLCI
#
CIRfwd=64               # CIR forward   1 - 64
```

```
# Bc_fwd=16            # Bc forward    1 - 512
# Be_fwd=0             # Be forward    0 - 511
# CIRbak=16            # CIR backward  1 - 64
# Bc_bak=16            # Bc backward   1 - 512
# Be_bak=0             # Be backward   0 - 511

#
# Configurazione di DLCI
# Queste voci sono tutte opzionali. Il nome usato è
# [DLCI_D<devicenum>_<DLCI_Num>]
#

[DLCI_D1_16]
# IP=
# Net=
# Mask=
# Flags defined by Sangoma: TXIgnore,RXIgnore,BufferFrames
# DLCIFlags=TXIgnore,RXIgnore,BufferFrames
# CIRfwd=64
# Bc_fwd=512
# Be_fwd=0
# CIRbak=64
# Bc_bak=512
# Be_bak=0

[DLCI_D2_16]
# IP=
# Net=
# Mask=
# Flags defined by Sangoma: TXIgnore,RXIgnore,BufferFrames
# DLCIFlags=TXIgnore,RXIgnore,BufferFrames
# CIRfwd=16
# Bc_fwd=16
# Be_fwd=0
# CIRbak=16
# Bc_bak=16
# Be_bak=0
```

Dopo aver costruito il file `/etc/frad/router.conf`, l'unico passo che resta da fare è configurare le periferiche stesse. Questo è solo leggermente più difficile che la configurazione di una normale periferica di rete. Occorre ricordare di attivare il FRAD prima dei dispositivi di incapsulamento DLCI. I comandi necessari sono numerosi, meglio quindi metterli in uno script:

```
#!/bin/sh
# Configurazione dell'hardware FRAD e parametri DLCI
/sbin/fradcfg /etc/frad/router.conf || exit 1
/sbin/dlcicfg file /etc/frad/router.conf
#
# Attivazione del dispositivo FRAD
ifconfig sdla0 up
#
# Configurazione delle interfacce di incapsulamento DLCI
# e instradamento
ifconfig dlci00 192.168.10.1 pointopoint 192.168.10.2 up
route add -net 192.168.10.0 netmask 255.255.255.0 dlci00
```

```
#
ifconfig dlci01 192.168.11.1 pointopoint 192.168.11.2 up
route add -net 192.168.11.0 netmask 255.255.255.0 dlci00
#
route add default dev dlci00
#
```

7.8 IPX (AF_IPX)

Il protocollo IPX è il più comunemente utilizzato nelle reti locali Novell NetWare(tm). Linux include il supporto per questo protocollo, e può essere configurato come punto di rete o come router per IPX.

Opzioni di compilazione del kernel:

```
Networking options  --->
    [*] The IPX protocol
    [ ] Full internal IPX network
```

Il protocollo IPX ed NCPFS sono trattati in maggior dettaglio nel documento *IPX-HOWTO* <IPX-HOWTO.html>.

7.9 NetRom (AF_NETROM)

I nomi delle periferiche NetRom sono 'nr0', 'nr1' eccetera.

Opzioni di compilazione del kernel:

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
    [*] Amateur Radio NET/ROM
```

I protocolli AX25, Netrom e Rose sono trattati nell'*AX25-HOWTO* <AX25-HOWTO.html>. Questi protocolli vengono usati dagli operatori di radio amatoriali in tutto il mondo per sperimentare con la trasmissione dati via radio.

La maggior parte del lavoro di implementazione di questi protocolli è stato fatto da Jonathon Naylor, jsn@cs.nott.ac.uk.

7.10 Il protocollo Rose (AF_ROSE)

I nomi delle periferiche Rose nei kernel 2.1.* sono 'rs0', 'rs1' eccetera. Rose è solo disponibile nelle versioni 2.1 del kernel.

Opzioni di compilazione del kernel:

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
    <*> Amateur Radio X.25 PLP (Rose)
```

I protocolli AX25, Netrom e Rose sono trattati nel documento *AX25-HOWTO* `<AX25-HOWTO.html>`. Questi protocolli vengono usati dagli operatori di radio amatoriali in tutto il mondo per sperimentare la trasmissione dati via radio.

La maggior parte del lavoro di implementazione di questi protocolli è stata fatta da Jonathon Naylor, `jsn@cs.nott.ac.uk`.

7.11 Supporto SAMBA - 'NetBEUI', 'NetBios'.

SAMBA è un'implementazione del protocollo "Session Management Block". Samba permette ai sistemi Microsoft e ad altri di montare i dischi Linux e di accedere alle stampanti Linux.

SAMBA e la sua configurazione sono trattati in dettaglio nel documento *SMB-HOWTO* `<SMB-HOWTO.html>`.

7.12 Supporto STRIP (Starmode Radio IP)

I nomi delle periferiche STRIP sono '`st0`', '`st1`', eccetera.

Opzioni di compilazione del kernel:

```
Network device support  --->
        [*] Network device support
        ....
        [*] Radio network interfaces
        < > STRIP (Metricom starmode radio IP)
```

STRIP è un protocollo designato specificatamente per una serie di modem radio Metricom per un progetto di ricerca della Stanford University, chiamato *Progetto MosquitoNet* `<http://mosquitonet.Stanford.EDU/mosquitonet.html>`. Vi si trova un sacco di materiale interessante da leggere, anche per chi non è direttamente interessato al progetto.

Le radio Metricom si connettono alla porta seriale, impiegano tecnologia "spread spectrum" e sono tipicamente in grado di trasmettere circa 100kbps. Informazioni riguardo alle radio Metricom sono disponibili presso *Server web della Metricom* `<http://www.metricom.com/>`.

Al momento gli strumenti di rete e i programmi di utilità standard non supportano il driver STRIP, per cui occorre scaricare dei programmi modificati a tal fine dal server www di MosquitoNet. Ulteriori dettagli riguardo al software necessario si trovano alla *pagina STRIP di MosquitoNet* `<http://mosquitonet.Stanford.EDU/strip.html>`.

In sintesi, occorre usare un programma *slattach* modificato per assegnare la disciplina di linea dell'interfaccia seriale a "STRIP"; occorre poi configurare i dispositivi '`st[0-9]`' come si farebbe con l'ethernet, tranne per una importante differenza: per ragioni tecniche, STRIP non supporta il protocollo ARP, per cui bisogna fornire manualmente al sistema le voci ARP per gli host della propria sottorete. Questo non è comunque un compito oneroso.

7.13 Token Ring

I nomi delle periferiche token ring sono '`tr0`', '`tr1`' eccetera. Token Ring è un protocollo di LAN standardizzato da IBM che evita le collisioni fornendo un meccanismo che ad ogni istante autorizza a trasmettere solo una stazione sulla rete locale. Un 'gettone' (token) è in mano ad una stazione per volta, e solo la stazione che detiene il gettone ha il diritto di trasmettere. Dopo aver trasmesso i dati il gettone verrà passato alla stazione successiva. Il gettone viene passato tra tutte le stazioni attive, da cui il nome 'Token Ring'.

Opzioni di compilazione del kernel:

```
Network device support  --->
        [*] Network device support
        ....
        [*] Token Ring driver support
        < > IBM Tropic chipset based adaptor support
```

La configurazione di token ring è identica a quella di ethernet, e l'unica differenza sta nel nome della periferica di rete da configurare.

7.14 X.25

X.25 è un protocollo a commutazione di pacchetto basato su un circuito, ed è definito dal `C.C.I.T.T.` (un gruppo di standard riconosciuto dalle compagnie di telecomunicazione in quasi tutto il mondo). Un'implementazione di X.25 e LPB è in lavorazione, e i kernel `2.1.*` più recenti includono lo stato attuale del lavoro.

Jonathon Naylor `jsn@cs.nott.ac.uk` sta coordinando lo sviluppo ed esiste una mailing list per discutere di X.25 sotto Linux e di argomenti connessi. Per iscriversi occorre mandare un messaggio a `majordomo@vger.rutgers.edu` con il testo "`subscribe linux-x25`" nel corpo del messaggio.

Le versioni attuali degli strumenti di configurazione si possono ottenere dal sito ftp di Jonathon: *ftp.cs.nott.ac.uk* `<ftp://ftp.cs.nott.ac.uk/jsn/>`.

7.15 Scheda WaveLan

I nomi delle periferiche Wavelan sono '`eth0`', '`eth1`', eccetera.

Opzioni di compilazione del kernel:

```
Network device support  --->
        [*] Network device support
        ....
        [*] Radio network interfaces
        ....
        <*> WaveLAN support
```

La scheda WaveLAN card è una scheda di rete radio ad ampio spettro. In pratica la scheda assomiglia molto ad una scheda ethernet, e viene configurata praticamente allo stesso modo.

Si possono ottenere informazioni sulla scheda Wavelan da *Wavelan.com* `<http://www.wavelan.com/>`.

8 Cavi e cablaggio

Chi ha pratica di lavori con il saldatore può desiderare di costruire da sé i propri cavi per connettere due macchine Linux. I seguenti diagrammi di cablatura dovrebbero essere di aiuto.

8.1 Cavo Seriale NULL Modem

Non tutti i cavi NULL modem sono uguali. Molti cavi null-modem fanno qualcosa in più che far credere al proprio calcolatore che tutti i segnali sono presenti e scambiare i segnali di trasmissione e ricezione. Questo è corretto, ma significa che occorre usare il controllo di trasmissione software (XON/XOFF), che è meno efficiente del controllo di flusso hardware. Il caso seguente offre la connettività migliore possibile e permette di usare il controllo di flusso hardware (RTS/CTS).

```
Pin Name  Pin                              Pin
Tx Data    2  -----------------------------  3
Rx Data    3  -----------------------------  2
RTS        4  -----------------------------  5
CTS        5  -----------------------------  4
Ground     7  -----------------------------  7
DTR        20 -\---------------------------  8
DSR        6  -/
RLSD/DCD   8  ---------------------------/-  20
                                         \-  6
```

8.2 Cavo per la porta parallela (cavo PLIP)

Se si intende usare il protocollo PLIP tra due macchine, allora questo cavo funzionerà qualunque sia il tipo di porta parallela che si possiede.

```
Pin Name    pin            pin
STROBE      1*
D0->ERROR   2  ----------- 15
D1->SLCT    3  ----------- 13
D2->PAPOUT  4  ----------- 12
D3->ACK     5  ----------- 10
D4->BUSY    6  ----------- 11
D5          7*
D6          8*
D7          9*
ACK->D3     10 ----------- 5
BUSY->D4    11 ----------- 6
PAPOUT->D2  12 ----------- 4
SLCT->D1    13 ----------- 3
FEED        14*
ERROR->D0   15 ----------- 2
INIT        16*
SLCTIN      17*
GROUND      25 ----------- 25
```

Note:

- Non collegate i piedini marcati con un asterisco '*'.
- Altri segnali di terra sono ai piedini 18,19,20,21,22,23 e 24.
- Se il cavo ha una schermatura metallica, questa deve essere connessa alla conchiglia metallica del connettore DB-25 **solo ad un estremo del cavo**.

Attenzione: Un cavo PLIP collegato male può distruggere il controllore della parallela. Bisogna fare molta attenzione e controllare più volte ogni connessione per essere sicuri di non dover fare lavoro inutile e non avere problemi.

Anche se si può riuscire a far andare cavi PLIP per lunghe distanze, questo, se possibile, va evitato. Le specifiche del cavo permettono una lunghezza di circa un metro. Bisogna fare molta attenzione quando si usano cavi PLIP molto lunghi: sorgenti di forti campi elettromagnetici come fulmini, linee di potenza e trasmettitori radio possono interferire e talvolta anche danneggiare il controllore della parallela. Se veramente si vogliono collegare due calcolatori molto distanti bisognerebbe procurarsi un paio di schede ethernet thin-net e usare un cavo coassiale.

8.3 Cablaggio ethernet 10base2 (coassiale sottile)

10base2 è uno standard di cablaggio ethernet che richiede l'uso di un cavo coassiale da 52 ohm, del diametro di circa 5 millimetri. Ci sono un paio di regole importanti da ricordare quando si connettono macchine tramite cavi 10base2. La prima è che occorre usare dei terminatori ad **entrambi i capi** del cavo. Un terminatore è una resistenza da 52 ohm che assicura che il segnale sia assorbito (non riflesso) quando raggiunge la fine del cavo. Senza i due terminatori si può verificare che la ethernet risulti non affidabile, o che non funzioni del tutto. Normalmente di usano dei 'connettori a T' per connettere i calcolatori, per cui il diagramma di rete assomiglierà al seguente:

```
|=========T============T============T=========T=========|
          |            |            |         |
          |            |            |         |
        -----        -----        -----     -----
        |   |        |   |        |   |     |   |
        -----        -----        -----     -----
```

dove i '|' ai due capi rappresentano i terminatori, gli '======' rappresentano un pezzo di cavo coassiale con spinotti BNC ad entrambi i capi e le 'T' rappresentano un connettore a T. Bisogna far sì che il pezzo di cavo tra la T e la scheda di rete sia il più corto possibile: idealmente la T deve essere connessa direttamente alla scheda ethernet.

8.4 Cavo Ethernet TP (Twisted Pair)

Se si hanno solo due schede di rete TP e le si vuole connettere insieme non occorre un hub. Si può fare il cavo in modo da connettere le due schede direttamente. Un diagramma che realizza questo è incluso nel documento *Ethernet-HOWTO* `<Ethernet-HOWTO.html>`

9 Glossario dei termini usati in questo documento.

Questa è una lista dei termini più importanti usati in questo howto.

ARP

Acronimo per *Address Resolution Protocol*, il protocollo usato per associare gli indirizzi IP agli indirizzi hardware.

ATM

Acronimo per *Asynchronous Transfer Mode*. Una rete ATM suddivide i dati in blocchi di dimensione standardizzata che poi trasmette efficientemente da un punto all'altro. ATM è una tecnologia di rete di pacchetto a commutazione di circuito.

client

Questo è di solito un pacchetto software che gira dove sta l'utente. Ci sono delle eccezioni a questa definzione sommaria: per esempio nel sistema a finestre X11 il server gira dove sta l'utente, e il client gira sulla macchina remota. Il client è un programma o un'estremo di un sistema che riceve un servizio offerto dal server. Nel caso di sistemi *peer to peer* come *slip* o *ppp* il client è l'estremo della connessione che inizia il collegamento mentre l'estremo remoto, quello chiamato, è detto server.

datagramma

Un datagramma è un pacchetto discreto di dati e intestazioni (header) contenenti informazioni per la consegna del pacchetto. Un datagramma è l'unità base di trasmissione attraverso una rete IP. Viene anche chiamato 'pacchetto'.

DLCI

Il DLCI è il Data Link Connection Identifier, e viene usato per identificare un'unica connessione punto-a-punto in una rete Frame Relay. I DLCI sono normalmente assegnati dal fornitore della rete Frame Relay.

Frame Relay

Frame Relay è una tecnologia di rete designata per distribuire il traffico sia questo ad impulsi (burst) o di natura sporadica. I costi di rete vengono ridotti avendo molti utenti Frame Relay che condividano la stessa capacità di rete, e basandosi sull'assunzione che ognuno di essi desideri usare la rete in tempi leggermente differenti.

Indirizzo Hardware

È un numero che identifica in maniera univoca un calcolatore al livello di accesso al mezzo di trasmizzione in una rete fisica. Esempi di questo sono gli *Indirizzi Ethernet* e gli *Indirizzi AX.25*.

ISDN

Acronimo per *Integrated-Services Digital Network*. ISDN offre un mezzo standardizzato attraverso il quale le compagnie di telecomunicazione possono distribuire sia voce che dati ai clienti. Tecnicamente ISDN è una rete di dati a commutazione di circuito.

ISP

Acronimo per Internet Service Provider. Queste sono organizzazioni o compagnie che offrono alla gente la connettività di rete verso Internet.

Indirizzo IP

Un numero che identifica univocamente un calcolatore TCP/IP sulla rete. L'indirizzo è lungo 4 byte e viene rappresentato in quella che è chiamata "notazione decimale con punti ("dotted decimal notation"), dove ogni byte viene rappresentato in decimale, e i byte sono separati da punti '.'.

MSS

Maximum Segment Size: è la quantita di dati più grande possibile che si può trasmettere in un solo istante. Se si vuole prevenire la frammentazione locale dei dati MSS dovrebbe essere uguale al "MTU - header IP".

MTU

Maximum Transmission Unit è un parametro che specifica la dimensione massima del pacchetto che può essere trasmesso da un'interfaccia IP senza che venga spezzettato in unità più piccole. La MTU dovrebbe essere più grande del pacchetto più grande che si vuole trasmettere non frammentato. Si noti che questo però previene solo la frammentazione locale, in quanto qualche altro collegamento sul percorso del pacchetto potrebbe avere una MTU minore, e il pacchetto a quel punto sarà frammentato. Valori tipici per MTU sono 1500 per le interfacce ethernet e 576 per interfacce SLIP.

route

La *route* è il percorso che i pacchetti prendono attraverso la rete per raggiungere la loro destinazione.

server

Questo è di solito il pacchetto software o l'estremo di connessione sul sistema remoto rispetto all'utente. Il server offre dei servizi ad uno o a molti client. Esempi di server includono *ftp*, *Network File System*, o *Domain Name Server*. Nel caso di connessioni *peer to peer* (da pari a pari) come *slip* o *ppp* il server è considerato quello all'estremo della connessione che viene chiamata, mentre l'estremo chiamante è il client.

window

La *window* (finestra) è la massima quantità di dati che l'estremo ricevente della connessione può accettare in un dato momento.

10 Linux per un provider?

Se siete interessati ad usare Linux come Internet Provider, raccomando di guardare la *homepage dei provider Linux* `<http://www.anime.net/linuxisp/>` per avere una buona lista di puntatori a informazioni di cui si potrebbe aver bisogno.

11 Ringraziamenti

Vorrei ringraziare le seguenti persone per i loro contributi a questo documento (senza un ordine particolare): Terry Dawson, Axel Boldt, Arnt Gulbrandsen, Gary Allpike, Cees de Groot, Alan Cox, Jonathon Naylor, Claes Ensson, Ron Nessim, John Minack, Jean-Pierre Cocatrix, Erez Strauss.

Per la traduzione, ringrazio Andrea Gelmini per le correzioni fornite.

12 Copyright.