Linux har ett kärnbaserat nätverksstöd som är skrivet nästan helt från noll. TCP/IP-prestandan i de senaste kärnorna gör Linux till ett värdefullt alternativ även till de bästa av dess motståndare. Detta dokument beskriver hur man installerar och konfigurerar nätverksmjukvara och tillhörande verktyg i Linux.

Innehåll

1 Ändringar från version 1.2 1
2 Introduktion. 1
  2.1 Feedback 2
  2.2 Översättarens kommentarer 2
3 Så här använder man detta HOWTO dokument (NET-3-HOWTO-HOWTO ?). 2
4 Generell Information om Linux Nätverk. 3
  4.1 En kort historia om utvecklingen av Linux Nätverkskärna. 3
  4.2 Var man hittar ytterligare information om Linux nätverk. 4
  4.3 Var man hittar generell nätverksinformation (icke Linux-specifik) 5
5 Generell Information om Nätverkskonfiguration 5
  5.1 Vad behöver man för att börja? 5
    5.1.1 Källkoden för kärnan. 6
    5.1.2 Nätverksverktyg (Network Tools). 6
    5.1.3 Applikationsprogram för nätverket 7
    5.1.4 Adresser. 7
  5.2 Var skall man skriva konfigurationskommandona? 9
  5.3 Att skapa sina nätverksgränssnitt. 10
  5.4 Att konfigurera ett nätverksgränssnitt. 10
  5.5 Att konfigurera din Name Resolver. 11
    5.5.1 Vad består ett namn av? 11
    5.5.2 Vilken information behöver man. 12
    5.5.3 /etc/resolv.conf. 12
    5.5.4 /etc/host.conf. 13
    5.5.5 /etc/hosts. 13
5.6 Att konfigurera loopbackenheten. .................................................. 13
5.7 Routing. ......................................................................................... 13
5.7.1 Så vad gör programmet routed? .................................................. 15
5.8 Att konfigurera nätverkservrar och tjänster. ...................................... 17
5.8.1 /etc/services .............................................................................. 17
5.8.2 /etc/inetd.conf ........................................................................... 22
5.9 Andra nätverksrelaterade konfigurationsfiler. .................................... 24
5.9.1 /etc/protocols ........................................................................... 25
5.9.2 /etc/networks ............................................................................ 25
5.10 Nätverkssäkerhet och åtkomstkontroll. ............................................. 26
5.10.1 /etc/ftpusers ............................................................................ 26
5.10.2 /etc/securetty .......................................................................... 26
5.10.3 tcpd åtkomstkontrollmekanism. ................................................ 26
5.10.4 /etc/hosts.equiv ................................................................. 28
5.10.5 Konfigurera ftp-daemonen ordentligt. ....................................... 28
5.10.6 Brandväggar. ........................................................................ 28
5.10.7 Andra förslag. ......................................................................... 28

6 Nätverksspecifik Information. .......................................................... 29
6.1 ARCNet .......................................................... 29
6.2 Appletalk (AF_APPLETALK) ............................................................. 29
6.2.1 Att konfigurera mjukvaran för Appletalk. .................................... 30
6.2.2 Att exportera ett Linuxfilsystem via Appletalk. ............................. 30
6.2.3 Att dela sin skrivare via Appletalk. ........................................... 31
6.2.4 Att starta programvaran för Appletalk. .................................... 31
6.2.5 Att testa programvaran för Appletalk. ..................................... 31
6.2.6 Brister i programvaran för Appletalk. ...................................... 31
6.2.7 Mer information. .................................................................. 32
6.3 ATM .......................................................... 32
6.4 AX25 (AF_AX25) ................................................................. 32
6.5 DECNet .......................................................... 32
6.6 EQL - trafikutjämnare för multipla linor. ....................................... 32
6.7 Ethernet .......................................................... 33
6.8 FDDI .......................................................... 33
6.9 Frame Relay ......................................................... 34
6.10 IP-redovisning (IP Accounting) .................................................. 37
6.11 IP Aliasing .............................................................................. 39
INNEHÅLL

6.12 IP Brandväggar (IP Firewalls) .................................................. 40
6.13 IPIP Inkapsling (IPIP Encapsulation) ........................................ 42
  6.13.1 En tunnlad nätverkskonfiguration. ........................................ 43
  6.13.2 En tunnlad datorkonfiguration. ............................................ 44
6.14 IPX (AF IPX) ............................................................................ 45
6.15 IPv6 .......................................................................................... 45
6.16 ISDN ....................................................................................... 46
6.17 IP-maskering (IP Masquerade) .................................................. 47
6.18 IP Transparent Proxy ................................................................. 48
6.19 Mobil IP ................................................................................... 48
6.20 Multicast .................................................................................. 49
6.21 NAT - Översättning av nätverksadresser (Network Address Translation) ... 49
6.22 NetRom (AF_NETROM) ............................................................... 49
6.23 PLIP (Parallel Line Internet Protocol) .......................................... 50
6.24 PPP (Point to Point Protocol) .................................................... 51
  6.24.1 Att vidhålla en permanent anslutning till nätet med pppd. ............ 51
6.25 Rose protokollet (AF_ROSE) ...................................................... 52
6.26 SAMBA (stöd för ‘NetBEUI’, ‘NetBios’). ....................................... 52
6.27 SLIP (Serial Line Internet Protocol) klient. .................................. 52
  6.27.1 dip (Dialup IP) ...................................................................... 52
  6.27.2 slattach ............................................................................... 53
  6.27.3 När använder man vilket? ...................................................... 53
  6.27.4 Statisk SLIP-server med uppringd förbindelse och dip ................. 53
  6.27.5 Dynamisk SLIP-server med uppringd förbindelse och dip ............ 54
  6.27.6 Att använda dip. ................................................................... 54
  6.27.7 Permanent SLIP-anslutning med hyrd ledning och slattach. ........ 57
6.28 SLIP (Serial Line Internet Protocol) server. .................................. 57
  6.28.1 SLIP-server med slpllogin. .................................................... 58
  6.28.2 Slip Server med dip. ............................................................. 61
  6.28.3 SLIP server med paketet dSLIP .............................................. 63
6.29 STRIP (Starmode Radio IP) ........................................................ 63
6.30 Token Ring .............................................................................. 64
6.31 X.25 ....................................................................................... 64
6.32 WaveLan .................................................................................. 64

7 Kablage ....................................................................................... 65

7.1 Seriell NULL-modem kabel. ....................................................... 65
1. Ändringar från version 1.2

Tillägg:
Lade till information om transproxy
Mindre fixar här och där

Rättningar/Uppdateringar:
Ny maintainer

ToDo (ej ändrad, tyvärr):
Lägg till traffic shaper
Beskriv ny routing algoritm
Lägg till kompileringsalternativ i kärnan för IPv6
Beskriv /proc/sys/net/* .
WanRouter enhet

2. Introduktion.

Den första NET-FAQen skrevs av Matt Welsh och Terry Dawson för att besvara ofta ställda frågor om nätverk
för Linux, detta i en tid före Linux Documentation Project hade startat. Den täckte de tidiga utvecklingsver-
sionerna av Linux nätverkskärna. NET-2-HOWTO tog över från NET-FAQ och var ett av de första LDP
HOWTO dokumenten, det täckte version 2, och senare version 3, av Linuxkärnans nätverksmjukvara. Detta
dokument tar i sin tur över från NET-2-HOWTO och behandlar endast version 3 av Linux nätverkskärna.

Tidigare versioner av detta dokument blev ganska stora beroende på den enorma mängd material som
det skulle täcka. För att minska storleken har ett antal HOWTO:s producerats som behandlar specifika
nätverksämnen. Detta dokument kommer, där det är relevant, att tillstå med referenser till dessa och beh-
handla de ämnen som inte täcks i andra dokument.

I April 1998 slutade Terry Dawson att upprätthålla NET-3, pga sin höga belastning. Jag är ny i jobbet, men
skall göra mitt bästa för att lyckas.

2.1 Feedback

Jag uppskattar alltid feedback och särskilt värdefulla bidrag. Var vänlig att skicka alla bidrag till email
<mailto:rubini@linux.it>.
2.2 Översättarens kommentarer


På ett par ställen har jag lagt till ett par meningar, jag har då markerat detta med ‘(SvÖ)’.

I andra sektioner av detta dokument (förutom denna) så betyder ’jag’ originalförfattaren.

–Tomas Carlsson, md5tc@mdstud.chalmers.se <mailto:md5c@mdstud.chalmers.se>

3 Så här använder man detta HOWTO dokument (NET-3-HOWTO-HOWTO ?).

Formatet på detta dokument skiljer sig från tidigare versioner. Vi har grupperat om sektionerna så att det finns upplysande material i början som man kan hoppa över om man inte är intresserad, sedan kommer generellt material som man måste förstå innan man fortsätter till de teknikspecifika delarna i resten av dokumentet.

Läs de generella sektionerna

Dessa sektioner används i alla, eller nästan alla, tekniker som beskrivs senare och det är viktigt att man förstår dessa.

Se över sitt nätverk

Man bör veta hur ens nätverk är, eller kommer att vara, designat och exakt vilken hårdvara och teknik som man kommer att implementera.

Läs det teknikspecifika sektionerna som är relaterade till de krav man har

När man vet vad man vill ha så kan man ta sig an varje komponent i tur och ordning. De här sektionerna behandlar endast detaljer för en specifik teknik.

Gör konfigurationsarbetet

Man bör försöka att konfigurera sitt nätverk och noggrant notera alla problem man stöter på.

Leta efter ytterligare hjälp om det behövs

Om man stöter på problem som detta dokument inte kan hjälpa till med så läs sektionen om var man kan hitta hjälp eller var man kan rapportera buggar.

Ha kul!

Nätverk är roligt, njut av det.

4 Generell Information om Linux Nätverk.

4.1 En kort historia om utvecklingen av Linux Nätverkskärrna.

Att utveckla en helt ny implementering av TCP/IP protokollstacken i kärnan som skulle presteras lika bra som existerande implementeringar var inte en lätt uppgift. Valet att inte porta en av de existerande implementeringarna gjordes vid en tid då det rådde osäkerhet beträffande om de existerande implementeringarna

kunde bli besvärade av restriktiva copyrights på grund av ett rättsfall av U.S.L och då det fanns mycket entusiasm för att göra det annorlunda och kanske till och med bättre än det redan hade gjorts.


Orest Zborowski <obz@Kodak.COM> producerade det första programmeringsgränssnittet för BSD-sockets till Linuxkärnan. Detta var ett stort steg framåt eftersom det tillåt många av de existerande nätverksapplikationerna att bli portade till Linux utan omfattande modifiering.

Någon gång vid denna tid utvecklade Laurence Culhane <loz@holmes.demon.co.uk> de första drivrutinerna för Linux som stödde SLIP-protokollet. Dessa gjorde det möjligt för många människor som inte hade tillgång till Ethernet-nätverk att experimera med den nya nätverksmjukvaran. Återigen så tog några personer denna drivrutin och lyckades ansluta sig till Internet. Detta gav många en smak av de möjligheter som kunde bli realiserade om Linux hade fullt nätverksstöd och ökade antalet användare av den existerande nätverksmjukvaran.

En av de som aktivt hade arbetat med uppgiften att implementera nätverksstöd var Fred van Kempen <waltje@walt.nl.mugnet.org>. Efter en tid av osäkerhet efter Ross avhopp från positionen som huvudutvecklare så erhöjde Fred sin tid och insats. Fred hade några ambitiösa planer för vad han ville leda Linux nätverksmjukvara och han började framskrida i den riktningen. Fred producerade en serie nätverkskod som kallades 'NET-2' kärnkod ('NET'-koden var Ross) som många kunde använda ganska användbart. Fred's NET-2 kod användes av ett hyfsat stort antal entusiaster, som ökade allt eftersom ryktet spred sig att koden fungerade. Vid denna tid var nätverksmjukvaran fortfarande ett stort antal patchar till standardversionen av kärnan och var inte inkluderad i den normala versionen. NET-FAQ och NET-2-HOWTO beskrev då den ganska komplicerade rutinen för att få allt att fungera. Freds mål var att utveckla nyheter till standard implementeringen, och detta tog tid. Linuxvärlden väntade otåligt på något som fungerade pålitligt och, precis som med Ross, så ökade trycket på Fred som huvudutvecklare.


Snart avslöjade Donald Becker <becker@cesdis.gsfc.nasa.gov> sina talanger i lägnivåaspekterna för nätverk och producerade en enorm mängd av Ethernet-drivrutiner, nästan alla de som finns i nuvarande kärnor är utvecklade av Donald. Det finns andra som har givit betydelsefulla bidrag, men Donalds arbete är produktivt och kräver speciell uppmärksamhet.

Alan fortsatte att förhinsa NET-2-Debugged koden en tid samtidigt som han arbetade på det som fanns

på 'TODO'-listan. Då Linux 1.3.* kärnkoden hade växt till sig hade nätverkskoden gått över till NET-3 versionen som nuvarande versioner grundar sig på. Alan arbetade på många olika aspekter på nätverkskoden och med hjälp av en mängd talangfulla personer från Linuxvärlden växte koden i alla möjliga riktningar. Alan producerade dynamiska nätverksenhetter och de första standard AX.25 och IPX implementeringarna. Alan har fortsatt att meka med koden, långsamt strukturerat om och förbättrat den till det tillstånd den är i idag.

PPP-stöd lades till av Michael Callahan <callahan@maths.ox.ac.uk> och Al Longyear <longyear@netcom.com>, detta var också kritiskt för att öka antalet personer som använde Linux för nätverk.

Jonathon Naylor <jsn@cs.nott.ac.uk> har bidragit med att göra betydande förbättringar av Alans AX.25 kod, lagt till stöd för NetRom och Rose protokoll. AX.25/NetRom/Rose stödet i sig själv är ganska betydelsefullt, för att inget annat operativsystem har inbyggt stöd för dessa protokoll förutom Linux.

Det finns naturligtvis hundratals andra personer som har givit betydelsefulla bidrag till utvecklingen av Linux nätverksmjukvara. Några av dessa kommer att nämns senare i de teknikspecifika sektionerna, andra har bidragit med moduler, drivrutiner, bugfixar, förslag, testrapporter och moraliskt stöd. I alla dessa fall kan var och en säga att den varit del av utvecklingen. Linux nätverkskod är ett utmärkt exempel på de resultat som kan uppnås med Linux anarkistiska utvecklingsstil, om man inte är förvånad över detta än, så blir man det snart för utvecklingen har inte slutat än.

4.2 Var man hittar ytterligare information om Linux nätverk.

Det finns ett antal ställen där man kan hitta bra information om Linux nätverk.


Det finns en nyhetsgrupp i Linux nyhetshierarkin som är avsedd för nätverk och dess relaterade ämnen, den är: comp.os.linux.networking <news:comp.os.linux.networking>

Det finns en mailing-lista som man kan prenumerera på där man kan fråga frågor relaterade till Linux nätverk. För att prenumerera skall man skicka ett e-post meddelande:

    To: majordomo@vger.rutgers.edu
    Subject: anything at all
    Message:
    subscribe linux-net

På de olika IRC nätverken finns det ofta #linux kanaler på vilka personer kan svara på frågor om Linux nätverk.

Kom ihåg att när man rapporterar ett problem så skall man inkludera så mycket relevanta detaljer om problemet som man kan. Speciellt så bör man specificera vilka versioner av mjukvara man använder, särskilt kärnversionen, versionen på verktyg såsom pppd eller dip och den exakta karaktären av problemet som man
upplever. Detta betyder bland annat att notera den exakta syntaxen på felmeddelanden som man får och alla kommandon som man använder.

4.3 Var man hittar generell nätverksinformation (icke Linux-specifik)

Om man vill ha grundläggande information om TCP/IP nätverk, så kan jag rekommendera följande dokument:

tcp/ip introduction


tcp/ip administration


Om man vill ha mer detaljerad information om TCP/IP nätverk så kan jag rekommendera:

"Internetworking with TCP/IP"
av Douglas E. Comer

ISBN 0-13-474321-0
Prentice Hall publications.


En viktig informationskälla när man är ute efter teknisk information relaterad till Internet och TCP/IP protokollen är RFCs. RFC är en förkortning av 'Request For Comment' och är det standardiserade sättet att lämna ut och dokumentera protokollstandarder för Internet. Det finns många ställen att hämta RFCs från. Många av dessa är ftp-sajter och andra tillhandahåller World Wide Web sidor med sökmotorer där man kan söka i RFC databasen med nyckelord.


5 Generell Information om Nätverkskonfiguration

Följande delsektioner innehåller information som man måste förstå för att kunna konfigurera sitt nätverk. Det är fundamentala principer som gäller oavsett karaktären på nätverket som man skall använda.

5.1 Vad behöver man för att börja?

Innan man börjar bygga eller konfigurera sitt nätverk behöver man några saker. De viktigaste är:

5.1.1 Källkoden för kärnan.

Eftersom kärnan som man kör nu inte nödvändigtvis ännu har stöd för de nätverk och nätverkskort som man vill använda så behöver man antagligen källkoden för kärnan så att man kan kompilera den med lämpliga tillägg.


Om inte annat sägs specifikt så rekommenderar jag att man håller dig till standardversionerna av kärnan (de med jämna nummer som andra siffra i versionsnummret). Versioner under utveckling (de med udda andra siffra) kan ha strukturella eller andra ändringar som kan skapa problem med annan mjukvara i sitt system. Om man är osäker på om man klarar av att lösa den typen av problem, så skall man inte använda de versionerna.

5.1.2 Nätverksverktyg (Network Tools).

Nätverksverktygen är de program som man använder för att konfigurera nätverksenheter i Linux. Med dessa verktyg kan man tilldela adresser till enheter och konfigurera routes till exempel.

De flesta moderna Linuxdistributioner är utrustade med nätverksverktyg, så om man har installerat från en distribution och ännu inte installerat nätverksverktygen så bör man göra detta nu.

Om man inte har installerat från en distribution så behöver man källkoden för att kompilera verktygen själv. Detta är inte svårt.


Man skall vara noga med att välja en version som passar den kärna som man har tänkt att använda och med att följa instruktionerna i paketet för att installera.

För att installera den senaste versionen då detta skrivs så behöver man göra följande:

```
# cd /usr/src
# tar xvzf net-tools-1.33.tar.gz
# cd net-tools-1.33
# make config
# make
# make install
```


För att installera den senaste versionen då detta skrivs så behöver man göra följande:

```
# cd /usr/src
# tar xvzf ipfwadm-2.3.0.tar.gz
# cd ipfwadm-2.3.0
# make
# make install
```
5. Generell Information om Nätverkskonfiguration

5.1.3 Applikationsprogram för nätverket


För att installera den senaste versionen då detta skrivs så behöver man göra följande:

```
# cd /usr/src
# tar xvfz /pub/net/NetKit-B-0.08.tar.gz
# cd NetKit-B-0.08
# more README
# vi MCONFIG
# make
# make install
#```

5.1.4 Adresser.

Internet Protokoll Adresser är uppbyggda av fyra bytes. Konventionen är att skriva adresser i vad som kallas 'punkterad decimal notation'. I denna formen är varje byte konverterad till ett decimaltal (0-255), utan nollor i början, och skrivs med varje byte separerad med ett '.' tecken. Varje (nätverks-)gränsnitt hos en värdator eller router har en IP-adress. I vissa fall är det tillåtet att ha samma IP-adress på alla gränsnitt på en och samma maskin, men vanligtvis har varje gränsnitt sin egen adress.

Internet Protokoll nätverk är kontinuerliga sekvenser av IP-adresser. Alla adresser inom ett nätverk har ett antal siffror i adressen gemensamma. Den del av adressen som är gemensam för alla adresser inom ett nätverk kallas 'nätverksdelen' ('network portion') av adressen. Antalet bitar som är delade av alla adresser inom ett nätverk kallas 'nätmasken' ('netmask') och det är nätmaskens uppgift att avgöra vilka adresser som hör till nätverket som nätmasken tillhör och vilka som inte gör det. Beakta följande exempel:

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

Alla adresser som blir 'bitvis andade' med sin nätmask kommer att avslöja adressen till nätverket som den tillhör. Nätverksadressen är därför den lägst numrerade adressen i den rad av adresser som finns i nätverket och har värdatordelen av adressen satt till nollor.

sajter nätverksadressen som broadcastadress. I praktiken spelar det ingen större roll vilket man använder men man måste vara säker på att alla maskiner på nätverket använder samma broadcastadress.

Av administrativa skäl någon gång tidigt i utvecklingen av IP-protokollet delades adresserna in i några slumpvisa grupper av nätverk och dessa nätverk grupperades i vad som kallas för klasser. Dessa klasser tillhandahåller ett antal nätverk av standardstorlekar som kan bli allokerade. De allokerade är:

<table>
<thead>
<tr>
<th>Network</th>
<th>Netmask</th>
<th>Network Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>255.0.0.0</td>
<td>0.0.0.0 - 127.255.255.255</td>
</tr>
<tr>
<td>B</td>
<td>255.255.0.0</td>
<td>128.0.0.0 - 191.255.255.255</td>
</tr>
<tr>
<td>C</td>
<td>255.255.255.0</td>
<td>192.0.0.0 - 223.255.255.255</td>
</tr>
<tr>
<td>Multicast</td>
<td>240.0.0.0</td>
<td>224.0.0.0 - 239.255.255.255</td>
</tr>
</tbody>
</table>

Vilka adresser som man skall använda beror på vad det är exakt man skall göra. Man kan behöva använda en kombination av följande för att få de adresser man behöver:

**Installera en Linuxmaskin på ett existerande IP-nätverk**

Om man vill installera en Linuxmaskin på ett existerande IP-nätverk så bör man kontakta den som administrerar nätverket och fråga efter följande:

- IP-adress för värddatorn (Host IP Address)
- IP nätverksadress (IP Network Address)
- IP broadcastadress
- IP nätmask (IP Netmask)
- Routeradress
- DNS adress (Domain Name Server Address)

Man skall sedan konfigurera sin Linux nätverksenhet med de uppgifterna. man kan inte hitta på dem och förvänta sig att det skall fungera.

**Bygga ett helt nytt nätverk som aldrig skall anslutas till Internet**

Om man bygger ett privat nätverk som man aldrig tänker ansluta till Internet så kan man välja vilka adresser man vill. Men, för säkerhets och konsistens skull så har vissa IP-adresser blivit reserverade speciellt för detta ändamål. De är speciferade i RFC1597 och är följande:

<table>
<thead>
<tr>
<th>RESERVED PRIVATE NETWORK ALLOCATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
</tr>
<tr>
<td>Class</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>C</td>
</tr>
</tbody>
</table>

Man bör först bestämma hur stort nätverket skall vara och sedan välja så många adresser som man behöver.
5. Generell Information om Nätverkskonfiguration

5.2 Var skall man skriva konfigurationskommandona?


Vanligtvis så innehåller /etc/inittab filen en rad som ser ut något som liknar:

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

Denna raden specificerar namnet på det shell-script som verkliga utför uppstarten. Denna fil skulle kunna jämföras med filen AUTOEXEC.BAT i MS-DOS.

Vanligtvis finns det andra script som anropas av boot-scriptet och ofta så konfigureras nätverket i något av dessa.

Följande tabell kan användas som guide för ett system:

<table>
<thead>
<tr>
<th>Distrib.</th>
<th>Interface Config/Routing</th>
<th>Server Initialisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debian</td>
<td>/etc/init.d/network</td>
<td>/etc/init.d/netbase</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/etc/init.d/netstd_init</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/etc/init.d/netstd_nfs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/etc/init.d/netstd_misc</td>
</tr>
<tr>
<td>Slackware</td>
<td>/etc/rc.d/rc.inet1</td>
<td>/etc/rc.d/rc.inet2</td>
</tr>
<tr>
<td>RedHat</td>
<td>/etc/sysconfig/network-scripts/ifup-&lt;ifname&gt;</td>
<td>/etc/rc.d/init.d/network</td>
</tr>
</tbody>
</table>

De flesta moderna distributioner har ett program med vilket man kan konfigurera många av de vanligaste sorterna av nätverksgränssnitt. Om man har en av dessa så bör man se om det kan göra det man vill innan man ger sig på manuell konfiguration.

<table>
<thead>
<tr>
<th>Distrib.</th>
<th>Network configuration program</th>
</tr>
</thead>
<tbody>
<tr>
<td>RedHat</td>
<td>/sbin/netcfg</td>
</tr>
<tr>
<td>Slackware</td>
<td>/sbin/netconfig</td>
</tr>
</tbody>
</table>

5.3 Att skapa sina nätverksgränssnitt.

I många Unix operativsystem så framträder nätverksenheterna i /dev katalogen. Så är inte fallet i Linux. I Linux skapas nätverksenheterna dynamiskt i mjukvara och därmed måste inte enhetsfiler finnas.

I de flesta fall skapas nätverksenheterna automatiskt av drivrutinen när den initialiseras och hittar din hårdvara. Till exempel så skapar Ethernetdrivrutinen eth[0..n] gränssnitten sekvensiellt medan den hittar din Ethernethårdvara. Det första Ethernetkortet som hittas blir eth0, det andra eth1 osv.

Men i några fall, tex slip och ppp, så skapas nätverksenheterna genom att något användarprogram körs. Samma sekvensiella nummering gäller, men enheterna skapas inte automatiskt vid systemstarten. Anledningen
till detta är att, till skillnad från Ethernetenheterna, så kan antalet aktiva slip eller ppp enheter variera under tiden systemet är igång. De här fallen gä ses igenom mer noggrant senare.

5.4 Att konfigurera ett nätverksgränssnitt.

När man har alla program man behöver och sina adress- och nätverksuppgifter så kan man konfigurera sina nätverksgränssnitt. När vi pratar om att konfigurera ett nätverksgränssnitt så pratar vi om processen att tilldela en nätverksenhet lämpliga adresser och att ge lämpliga värden till andra konfigurerbara parametrar hos en nätverksenhet. Det mest använda programmet för detta är kommandot `ifconfig` (interface configure). Vanligtvis bör man använda ett kommando som liknar följande:

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

I detta fallet konfigurerar jag ett Ethernetkort `eth0` med IP-adressen `192.168.0.1` och en nätmask `255.255.255.0`. Parameteren `up` som följer kommandot talar om för enheten att den skall bli aktiv.

Kärnan förutsätter vissa förvalda värden när man konfigurerar gränssnitt. Till exempel så kan man specificera nätverksadress och broadcastadress för ett gränssnitt. Men om man inte gör det, som i exemplet ovan, så gör kärnan rimliga gissningar baserade på nätmasken, och om man inte specificerat någon nätmask så tittar kärnan på klassen av IP-adress. I exemplet ovan skulle kärnan anta att det är ett klass C nätverk som skall konfigureras och därmed sätta nätverksadressen till `192.168.0.0` och broadcastadressen till `192.168.0.255`. Det finns många andra parametrar till `ifconfig`. De viktigaste är:

- **up**
  aktiverar ett gränssnitt.

- **down**
  deaktiverar ett gränssnitt.

- **[- arp]**
  slår av eller slår på ’address resolution protocol’ för gränssnittet.

- **[- allmulti]**
  slår av eller slår på mottagning av alla hårdvaru-multicastpaket. Hårdvaru-multicast låter grupper av maskiner att ta emot paket som är adresserade till speciella destinationer. Detta kan vara viktigt om man använder applikationer såsom videokonferenssystem men används normalt inte.

- **mtu N**
  specificerar MTU för enheten.

- **netmask addr**
  specificerar nätmasken för nätverket som maskinen tillhör.

- **irq addr**
  denna parameter fungerar endast för viss hårdvara och sätter IRQ för enheten.

- **[- broadcast [addr]]**
  slår på mottagning av datagram som skickas till broadcastadressen, eller slår av mottagning av dessa datagram.

- **[- pointopoint [addr]]**
  sätter adressen på maskinen på andra änden av en punkt till punkt förbindelse som tex slip eller ppp.
5. Generell Information om Nätverkskonfiguration

hw <type> <addr>

sätter hårdvaruadressen för nätverksenheten. Oftast är detta inte användbart för Ethernet, men är användbart för andra typer av nätverk som tex AX.25.

Man kan använda kommandot `ifconfig` på alla nätverksgränssnitt. Vissa användarprogram, tex `pppd` och `dip`, konfigurerar enheterna automatisk när de skapar dem, så manuell användning av `ifconfig` är inte nödvändig.

5.5 Att konfigurera din Name Resolver.


5.5.1 Vad består ett namn av?

De flesta antagligen bekanta med hur datornamn uppträder i Internet, men man kanske inte vet hur de är konstruerade, eller rekonstruerade. Internet domännamn är av hierarkisk karaktär, dvs de har trädstruktur. En domän är en familj, eller grupp av namn. En domän kan brytas ned i subdomänar. En toppdomän är en domän som inte är en subdomän. Toppdömen är speciferade i RFC-920. Några exempel på de vanligaste toppdomänerna är:

**COM**

Kommersiella Organisationer

**EDU**

Utbildningsorganisationer

**GOV**

Regeringsorganisationer

**MIL**

Militärganisationer

**ORG**

Andra Organisationer

**NET**

Internet-Relaterade Organisationer

**Landskod**

Detta är koder som består av två bokstäver och som representerar ett visst land.

 Alla dessa toppdomän har subdomänar. Toppdömen baserade på landsnamn bryts sedan ned i subdomän baserade på `com, edu, gov, mil och org` domän (detta är ju dock inte fallet i `.se-domänen` ännu. Översättarens kommentar). Så till exempel blir det `com.au` och `gov.au` för kommersiella och regering-organisationer i Australien. Av historiska skäl så används de domän som tillhör de icke landbaserade toppdomänerna mestadels av organisationer i USA, trots att USA har sin egen landskod `.us`.

Nästa nedbrytningsnivå representerar ofta namnet på organisationen. Ytterligare subdomänar varierar i karaktär, ofta så baseras nästa nivå på avdelningar inom organisationen, men den kan baseras på vilket kriterium som helst som har någon betydelse för nätverksadministratörerna i organisationen.
Delen längst till vänster av namnet är alltid det unika namnet som värd datorn har och det kallas för *hostname*, den delen av namnet som finns till höger om *hostname* kallas för *domainname* och hela namnet heter *Fully Qualified Domain Name*.


5.5.2 Vilken information behöver man.

Man behöver veta vilken domän datorns namn skall tillhöra. Name Resolver mjukvaran tillhandahåller namnöversättning genom att göra förfrågningar till en *Domain Name Server*, så man behöver veta IP-adressen till en lokal namnserver som man kan använda.

Det finns tre filer som man behöver editera, jag berättar om dem i tur och ordning.

5.5.3 /etc/resolv.conf

Filens /etc/resolv.conf är huvudfilen för att konfigurera namnöversättaren. Dess format är ganska enkelt. Det är en textfil med ett nyckelord per rad. Det finns tre nyckelord som vanligtvis används, de är:

domain

detta nyckelord specificerar det lokala domännamnet.

search

detta nyckelord specificerar en lista med alternativa domännamn att leta i efter ett datornamn.

nameserver

detta nyckelord, som kan användas flera gånger, specificerar en IP-adress till en namnserver (*Domain Name Server*) att använda när man översätter namn.

Ett exempel på /etc/resolv.conf kan se ut någonting som följande:

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

Detta exempel anger att man lägger till domännamnet maths.wu.edu.au till datornamn som ges utan domännamn, och om datorn inte hittas där så skall man leta även i wu.edu.au domänen. Två namnservrar är specificerade, som båda kan kontaktas av namnöversättaren när ett namn skall översättas.

5.5.4 /etc/host.conf

I filen /etc/host.conf kan man konfigurera hur namnöversättaren beter sig. Formatet för filen är beskrivet i detalj i manualsidan resolv+. I nästan alla fall fungerar följande exempel:

```plaintext
order hosts,bind
multi on
```
Denna konfiguration talar om för översättaren att titta i filen `/etc/hosts` innan den försöker göra en förfrågning till namnservern och att returnera alla giltiga adresser för en dator som den hittar i `/etc/hosts` och inte bara den första.

### 5.5.5 `/etc/hosts`


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

Man kan ange mer än ett datornamn per rad som på den första raden i exemplet, vilket är standardutformningen för loopbackenheten.

### 5.6 Att konfigurera loopbackenheten.

Loopback-enheten är ett särskilt gränssnitt med vilket man kan göra anslutningar till sig självt. Det finns olika anledningar varför man skulle vilja göra detta, till exempel så kanske man vill testa någon nätverksmjukvara utan att störa någon annan på sitt nätverk. Av tradition sår har IP-adressen `127.0.0.1` reserverats speciellt för loopback. Så, oavsett vilken maskin man går till, om man öppnar en telnet-anslutning till `127.0.0.1` så blir man alltid ansluten till den lokala datorn.

Att konfigurera loopbackenheten är enkelt och man bör göra följande:

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

Vi pratar mer om `route` kommandot i nästa sektion.

### 5.7 Routing.


Låt oss börja med en definition. Vad är IP routing? Här följer den definition som jag använder:

> IP routing är processen där en dator med flera nätverksanslutningar avgör vart den skall skicka IP datagram som den har tagit emot.


Hur fungerar routing? Varje dator har en speciell lista med routingregler som kallas för routingtabellen (routing table). Denna tabell innehåller rader som vanligtvis innehåller åtminstone tre fält, det första är en destinationsadress, det andra är namnet på gränssnittet som datagrammet skall routas till och det tredje är IP-adressen på en annan maskin som skall ta datagrammet på sitt nästa steg genom nätverket. I Linux kan man se denna tabell genom att använda följande kommando:

```
# cat /proc/net/route
```

eller genom att använda något av följande kommandon:

```
# /sbin/route -n
# /bin/netstat -r
```


Det första steget är att konfigurera gränssnittet som det beskrivits ovan. Man skulle använda ett kommando som:

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

Man behöver nu lägga till en rad i routingtabellen för att tala om för kärnan att datagram till alla adresser som passar 192.168.1.* skall sändas till en Ethernet-enhet. Detta med ett kommando som:

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


Med denna route kan man upprätta IP-anslutningar till alla datorer på sitt Ethernet-segment. Men vad händer med alla IP-datorer som inte finns på ens Ethernet-segment?

Det skulle vara ett väldigt svårt jobb att behöva lägga till router till varje tänkbara nätverk, så det finns ett specialtrix som används för att förenkla denna uppgift. Trixet kallas för `default` route. Default routen passar varje möjlig destination, men dåligt, så om det finns någon annan rad som passar den önskade adressen bättre så kommer den att användas istället för default routen. Tanken med default routen är helt enkelt att man skall kunna säga "och allt annat skall skickas dit". Så i exemplet som vi följt så bör man använda en rad som:

```
# route add default gw 192.168.1.1 eth0
```
5. Generell Information om Nätverkskonfiguration

Parametern `gw` talar om för route kommandot att nästa parameter är IP-adressen, eller namnet, på en gateway eller router dit alla datagram som passar denna rad skall skickas för vidare routing.

Så exemplet kompletta konfiguration skulle se ut så här:

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

Om man tittar noga i sina `rc`-filer för nätverket så kommer man att hitta åtminstone en som liknar detta. Detta är en vildigt vanlig konfiguration.

Låt oss nu titta på en något mer komplicerad routingkonfiguration. Antag att vi konfigurerar routern som vi tittade på tidigare, med en PPP-länk till Internet och några LAN-segment med arbetsstationer på kontoret. Antag att routern har tre Ethernet-segment och en PPP-länk. Vår routingkonfiguration skulle likna följande:

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


5.7.1 Så vad gör programmet `routed`?

Routingkonfigurationen som beskrivs ovan passar bäst för enklare nätverksarrangemang där det endast finns en möjlig väg till varje destination. När man har ett mer komplext arrangemang så blir det lite besvärligare. Som tur är behöver de flesta inte bry sig om detta.

Det stora problemet med 'manuell routing' eller 'statisk routing' är att om en maskin eller länk fallerar i nätverket så kan man endast dirigera om sina datagram, om det finns någon annan väg, genom att manuellt ge lämpliga kommandon. Naturligtvis är detta klumpigt, långsamt, opraktiskt och felbevänt. Olika tekniker har utvecklats för att automatiskt ändra routingtabeller om det uppstår nätverksfel där det finns alternativa vägar att leda trafiken, alla dessa tekniker är löst samlade under termen 'dynamiska routing algoritmer'.

Man kanske har hört talas om några av de vanligare dynamiska routingalgorimerna. De allra vanligaste är RIP (Routing Information Protocol) och OSPF (Open Shortest Path First). RIP är vildigt vanlig i små nätverk som till exempel små/mellanstora företagsnätverk. OSPF är modernare och mer kapabel att hantera stora nätverk och bättre anpassad till omgivningar där det finns ett stort antal möjliga vägar genom ett nätverk. Vanliga implementeringar av dessa är: `routed` (RIP) och `gated` (RIP, OSPF och andra). Programmet `routed` finns normalt med i Linuxdistributioner eller finns inkluderat i 'NetKit' paketet ovan.

Ett exempel på var och hur man skulle kunna använda en dynamisk routingalgoritm skulle kunna se ut som följer:

```
192.168.1.0 / 255.255.255.0 192.168.2.0 / 255.255.255.0
```

Det bör vara klart att routingtabellen för router A skulle kunna se ut som:

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

Detta skulle fungera utmärkt ända tills länken mellan router A och B fallerar. Om så skulle ske så skulle datorer på A’s Ethernet-segment inte kunna nå datorer på B’s Ethernet-segment eftersom dess datagram skulle routas till A’s ppp0-länk, vilken är trasig. De skulle fortfarande kunna 'prata' med datorer på C’s Ethernet-segment och datorer på C’s Ethernet-segment skulle kunna kontakta datorer på B’s Ethernet-segment eftersom länkarna mellan A och C respektive mellan C och B fortfarande fungerar.

Men vänta lite nu, om A kan prata med C och C fortfarande kan prata med B, varför skulle inte A kunna routa sina datagram till B via C och låta C skicka dem till B? Detta är precis den typen av problem som dynamiska routingalgoritmer som RIP designades för att lösa. Om var och en av routrarna A, B och C körde en routingdaemon så skulle deras routingtabeller automatiskt uppdateras till att spegla det nya tillståndet i nätverket om någon av länkarna skulle fallera. För att konfigurera ett sådant nätverk är enkelt, på varje router behöver man endast göra två saker. I detta fall för router A:

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

Routingdaemonen routed hittar automatiskt alla aktiva nätverksportar när den startar och skickar och
lyssnar efter meddelanden på var och en av nätverksenheterna så att den kan fastställa och uppdatera routingtabellen på datorn.

Detta har varit en väldigt översiktlig förklaring av dynamisk routing och var man bör använda det. Om man vill ha mer information så sök ibland referenserna i början av dokumentet.

De viktiga punkterna relaterade till dynamisk routing är:

1. Man behöver bara använda dynamisk routing när ens Linuxburk har möjligheten att välja flera olika routes till en destination.
2. Den dynamiska routingdaemonen kommer automatiskt att modifiera routingtabellen för att ställa in sig till förändringar i nätverket.
3. RIP är bra anpassat för små till medelstora nätverk.

### 5.8 Att konfigurera nätverksservrar och tjänster.


**fristående**

nätverksdaemonen lyssnar på den angivna porten och när den upptäcker en inkommande anlåtning så sköter den om anlåtningen själv för att vidare tillhandahålla tjänster.

**slav till inetd servern**

*inetd* servern är en särskild nätverks daemon som är specialiserad på att ta hand om inkommande nätverksanslutningar. Den har en konfigurationsfil som talar om vilket program som skall köras då en inkommande anlåtning tas emot. Varje port kan konfigureras för något av protokollen TCP eller UDP. Portarna är beskrivna i en annan fil som vi skall prata om snart.

Det finns två viktiga filer som behöver konfigureras. De är */etc/services* som tilldelar namn till portnummer och */etc/inetd.conf* som är konfigurationsfilen till nätverks daemonen *inetd*.

#### 5.8.1 */etc/services*

Filen */etc/services* är en enkel databas som associerar ett människoväntligt namn med en maskinväntlig port. Formatet är ganska enkelt. Filen är en textfil där varje rad representerar en rad i databasen. Varje rad består av tre fält som är separerade av valfritt antal vita tecken (tab eller space). Fälten är:

- **name**
- **port/protocol**
- **aliases**
- **# comment**

name

ett namn, bestående av ett ord, som representerar tjänsten som skall beskrivas.

port/protocol

detta fält är indelat i två fält.
5. Generell Information om Nätverkskonfiguration

port

protocol
detta delfält kan sättas till antingen tcp eller udp.

Det är viktigt att notera att en rad med 18/tcp är väldigt olik en rad med 18/udp och att det inte finns någon teknisk anledning till att samma tjänst finns tillgänglig på båda. Normalt råder sunt förnuft och det är bara om en viss tjänst finns tillgänglig via både tcp och udp som man kommer att se rader med båda.

aliases
andra namn som kan användas för att referera till denna tjänsten.

All text som finns efter ett # tecken på en rad ignoreras och behandlas som en kommentar.

Ett exempel på en /etc/services fil. Alla moderna Linuxdistributioner kommer med en bra /etc/services fil. Men i fall man råkar bygga ett system från grunden så är här en kopia av /etc/services filen som kommer med Debian <http://www.debian.org/> distributionen.

```
# /etc/services:
# $Id: services,v 1.3 1996/05/06 21:42:37 tobias Exp $
#
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, most entries here have two entries
# even if the protocol doesn't support UDP operations.
# Updated from RFC 1340, "Assigned Numbers" (July 1992). Not all ports
# are included, only the more common ones.

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
```
5. Generell Information om Nätverkskonfiguration

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 ISO/DE.
csnetsvc 105/tcp cso-ns # also used by CSO name server
csnetsvc 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
ntpd 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
cmp-man 163/tcp # ISO mgmt over IP (CMOT)
5. Generell Information om Nätverkskonfiguration

<table>
<thead>
<tr>
<th>Service</th>
<th>Port</th>
<th>Mode</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmip-man</td>
<td>163</td>
<td>udp</td>
<td></td>
</tr>
<tr>
<td>cmip-agent</td>
<td>164</td>
<td>tcp</td>
<td></td>
</tr>
<tr>
<td>cmip-agent</td>
<td>164</td>
<td>udp</td>
<td></td>
</tr>
<tr>
<td>xdmcp</td>
<td>177</td>
<td>tcp</td>
<td># X Display Mgr. Control Proto</td>
</tr>
<tr>
<td>xdmcp</td>
<td>177</td>
<td>udp</td>
<td></td>
</tr>
<tr>
<td>nextstep</td>
<td>178</td>
<td>tcp</td>
<td>NeXTStep NextStep # NeXTStep window</td>
</tr>
<tr>
<td>nextstep</td>
<td>178</td>
<td>udp</td>
<td>NeXTStep NextStep # server</td>
</tr>
<tr>
<td>bgp</td>
<td>179</td>
<td>tcp</td>
<td># Border Gateway Proto.</td>
</tr>
<tr>
<td>bgp</td>
<td>179</td>
<td>udp</td>
<td></td>
</tr>
<tr>
<td>prospero</td>
<td>191</td>
<td>tcp</td>
<td># Cliff Neuman’s Prospero</td>
</tr>
<tr>
<td>prospero</td>
<td>191</td>
<td>udp</td>
<td></td>
</tr>
<tr>
<td>irc</td>
<td>194</td>
<td>tcp</td>
<td># Internet Relay Chat</td>
</tr>
<tr>
<td>irc</td>
<td>194</td>
<td>udp</td>
<td></td>
</tr>
<tr>
<td>smux</td>
<td>199</td>
<td>tcp</td>
<td># SNMP Unix Multiplexer</td>
</tr>
<tr>
<td>smux</td>
<td>199</td>
<td>udp</td>
<td></td>
</tr>
<tr>
<td>at-rtmp</td>
<td>201</td>
<td>tcp</td>
<td># AppleTalk routing</td>
</tr>
<tr>
<td>at-rtmp</td>
<td>201</td>
<td>udp</td>
<td></td>
</tr>
<tr>
<td>at-nbp</td>
<td>202</td>
<td>tcp</td>
<td># AppleTalk name binding</td>
</tr>
<tr>
<td>at-nbp</td>
<td>202</td>
<td>udp</td>
<td></td>
</tr>
<tr>
<td>at-echo</td>
<td>204</td>
<td>tcp</td>
<td># AppleTalk echo</td>
</tr>
<tr>
<td>at-echo</td>
<td>204</td>
<td>udp</td>
<td></td>
</tr>
<tr>
<td>at-zis</td>
<td>206</td>
<td>tcp</td>
<td># AppleTalk zone information</td>
</tr>
<tr>
<td>at-zis</td>
<td>206</td>
<td>udp</td>
<td></td>
</tr>
<tr>
<td>z3950</td>
<td>210</td>
<td>tcp</td>
<td>wais # NISO Z39.50 database</td>
</tr>
<tr>
<td>z3950</td>
<td>210</td>
<td>udp</td>
<td>wais</td>
</tr>
<tr>
<td>ipx</td>
<td>213</td>
<td>tcp</td>
<td># IPX</td>
</tr>
<tr>
<td>ipx</td>
<td>213</td>
<td>udp</td>
<td></td>
</tr>
<tr>
<td>imap3</td>
<td>220</td>
<td>tcp</td>
<td># Interactive Mail Access</td>
</tr>
<tr>
<td>imap3</td>
<td>220</td>
<td>udp</td>
<td># Protocol v3</td>
</tr>
<tr>
<td>ulistserv</td>
<td>372</td>
<td>tcp</td>
<td># UNIX Listserv</td>
</tr>
<tr>
<td>ulistserv</td>
<td>372</td>
<td>udp</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td></td>
<td></td>
<td># UNIX specific services</td>
</tr>
<tr>
<td>exec</td>
<td>512</td>
<td>tcp</td>
<td></td>
</tr>
<tr>
<td>biff</td>
<td>512</td>
<td>udp</td>
<td>comsat</td>
</tr>
<tr>
<td>login</td>
<td>513</td>
<td>tcp</td>
<td></td>
</tr>
<tr>
<td>who</td>
<td>513</td>
<td>udp</td>
<td>whod</td>
</tr>
<tr>
<td>shell</td>
<td>514</td>
<td>tcp</td>
<td>cmd # no passwords used</td>
</tr>
<tr>
<td>syslog</td>
<td>514</td>
<td>udp</td>
<td></td>
</tr>
<tr>
<td>printer</td>
<td>515</td>
<td>tcp</td>
<td>spooler # line printer spooler</td>
</tr>
<tr>
<td>talk</td>
<td>517</td>
<td>udp</td>
<td></td>
</tr>
<tr>
<td>ntalk</td>
<td>518</td>
<td>udp</td>
<td></td>
</tr>
<tr>
<td>route</td>
<td>520</td>
<td>udp</td>
<td>router routed # RIP</td>
</tr>
<tr>
<td>timed</td>
<td>525</td>
<td>udp</td>
<td>timeserver</td>
</tr>
<tr>
<td>tempo</td>
<td>526</td>
<td>tcp</td>
<td>newdate</td>
</tr>
<tr>
<td>courier</td>
<td>530</td>
<td>tcp</td>
<td>rpc</td>
</tr>
<tr>
<td>conference</td>
<td>531</td>
<td>tcp</td>
<td>chat</td>
</tr>
<tr>
<td>netnews</td>
<td>532</td>
<td>tcp</td>
<td>readnews</td>
</tr>
<tr>
<td>netwall</td>
<td>533</td>
<td>udp</td>
<td># -for emergency broadcasts</td>
</tr>
<tr>
<td>uucp</td>
<td>540</td>
<td>tcp</td>
<td>uucpd # uucp daemon</td>
</tr>
<tr>
<td>remotefs</td>
<td>556</td>
<td>tcp</td>
<td>rfs_server rfs # Brunhoff remote filesystem</td>
</tr>
<tr>
<td>klogon</td>
<td>543</td>
<td>tcp</td>
<td># Kerberized ‘rlogin’ (v5)</td>
</tr>
<tr>
<td>kshell</td>
<td>544</td>
<td>tcp</td>
<td>krcmd # Kerberized ‘rsh’ (v5)</td>
</tr>
<tr>
<td>kerberos-adm</td>
<td>749</td>
<td>tcp</td>
<td># Kerberos ‘kadmin’ (v5)</td>
</tr>
</tbody>
</table>
# webster 765/tcp # Network dictionary
webster 765/udp
#
# From 'Assigned Numbers':
#
# The Registered Ports are not controlled by the IANA and on most systems
# can be used by ordinary user processes or programs executed by ordinary
# users.
#
# Ports are used in the TCP [45,106] to name the ends of logical
# connections which carry long term conversations. For the purpose of
# providing services to unknown callers, a service contact port is
# defined. This list specifies the port used by the server process as its
# contact port. While the IANA can not control uses of these ports it
# does register or list uses of these ports as a convienience to the
# community.
#
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
#
# Kerberos (Project Athena/MIT) services
# Note that these are for Kerberos v4 and are unofficial. Sites running
# v4 should uncomment these and comment out the v5 entries above.
#
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
#
# Unofficial but necessary (for NetBSD) services
#
supfilesrv 871/tcp # SUP server
supfiledbg 1127/tcp # SUP debugging
#
# Datagram Delivery Protocol services
#
rtmp 1/ddp # Routing Table Maintenance Protocol
nbp 2/ddp # Name Binding Protocol
5. Generell Information om Nätverkskonfiguration

5.8.2 /etc/inetd.conf


Dess format är ganska enkelt. Det är en textfil där varje rad beskriver en tjänst som man vill tillhandahålla. All text efter ett # tecken på en rad ignoreras och ses som en kommentar. Varje rad innehåller sju fält som är separerade av valfritt antal vita tecken (tab eller space). Det generella formatet är som följer:

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

`service` är tjänsten som är relevant för denna konfiguration som den benämnas i /etc/services.


`proto` protokollet som skall anses som giltigt för denna rad. Detta skall passa ihop med lämplig rad i /etc/services och är vanligtvis antingen tcp eller udp. Sun RPC (Remote Procedure Call) baserade tjänster använder rpc/tcp eller rcp/udp.

`flags` består egentligen bara två möjliga värden för detta fält. Detta fält talar om för inetd om serverprogramvaran stänger socketen efter det att den startats och därför om inetd kan starta en annan server på nästa anslutningsförfrågan. Återigen så är detta lite besvärlikt att lista ut, men som tumregel så skall alla tcp servrar ha värdet `nowait` och de flesta udp servrar `wait`. Observera att det finns undantag för detta, så man bör endast följa exemplet om man inte vet.


```
echo 4/ddp # AppleTalk Echo Protocol
zip 6/ddp # Zone Information Protocol
#
# Debian GNU/Linux services
rmtcfg 1236/tcp # Gracilis Packeten remote config server
xel 1313/tcp # french minitel
cfinger 2003/tcp # GNU Finger
postgres 4321/tcp # POSTGRES
mandelspawn 9359/udp mandelbrot # network mandelbrot
```
5. Generell Information om Nätverkskonfiguration

server_path

detta fält skall innehålla det absoluta filnamnet till serverprogrammet som skall exekveras för denna raden.

server_args

detta fältet utgör resten av raden och är valfri. Det är här som man placerar kommandoradsargument som man vill skicka med till serverdaemonen när den startas.


```bash
# /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/uucp/uucico
#comsat dgram udp wait root /usr/sbin/tcpd /usr/sbin/in.comsat
```
5. Generell Information om Nätverkskonfiguration

# 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
#sysstat stream tcp nowait nobody /usr/sbin/tcpd /bin/ps -auwx
#
# 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/kpasswd
#
# 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.walld
#
# End of inetd.conf.
ident stream tcp nowait nobody /usr/sbin/identd identd -i

5.9 Andra nätverksrelaterade konfigurationsfiler.

Det finns ett antal andra filer som är relaterade till nätverkskonfiguration i Linux som man skulle kunna vara intresserad av. Man behöver kanske aldrig modifiera dessa filer, men det är värts att beskriva dem ändå så att man vet vad de innehåller och vad de används till.

5.9.1 /etc/protocols

Filens /etc/protocols är en databas som mappar protokollens id-nummer mot protokollens namn. Detta används av programmerare för att låta dem ange protokoll med dess namn i program, och även av vissa program, som till exempel tcpdump, för att kunna skriva ut namn istället för nummer. Syntaxen för filen är:
5. Generell Information om Nätverkskonfiguration

protocollnumber number aliases

I Debian <http://www.debian.org/> distributionen ser /etc/protocols ut som följande:

# /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 Transfer Protocol
ddp 37 DDP # Datagram Delivery Protocol
idpr-cmtp 39 IDPR-CMTP # IDPR Control Message Transport
rspf 73 RSPF # Radio Shortest Path First.
vmp 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

Filen /etc/networks har en liknande funktion som filen /etc/hosts. Den tillhandahåller en enkel databas av nätverksnamn som mappas mot nätverksadresser. Dess format skiljer sig genom att det får endast finnas två fält per rad och att fältens skrivs som:

networkname networkaddress

Ett exempel kan se ut så här:

loopnet 127.0.0.0
localnet 192.168.0.0
amprnet 44.0.0.0

När man använder kommandon som route, om en destination är ett nätverk och det nätverket finns i /etc/networks, så kommer route att visa nätverksnamnet istället för dess adress.
5.10 Nätverkssäkerhet och åtkomstkontroll.

Låt mig börja denna sektion med att varna för att säkra en maskin och ett nätverk mot illvilliga attacker är en komplex konst. Jag anser mig inte själv vara en expert på detta område och även om de följande mekanismerna som jag beskriver kommer att hjälpa, så om man är riktigt allvarlig när det gäller säkerhet så rekommenderar jag att man gör egna undersökningar i ämnet. Det finns många bra referenser på Internet relaterade till detta ämne.


Det finns alla möjliga sortsäkerhets- och åtkomstkontrollmekanismer, jag kommer att beskriva de mest elementära av dem.

5.10.1 `/etc/ftpusers`

Filens `/etc/ftpusers` är en enkel mekanism med vilken man kan vägra vissa användare att logga in till maskinen via ftp. Filen(`/etc/ftpusers` läses av ftp-daemonen (`ftpd`) när en inkommande ftp-anslutning tas emot. Filen är en enkelt lista av de användare som inte är tillåtna att logga in. Den skulle kunna se ut som:

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

5.10.2 `/etc/securetty`

I filen `/etc/securetty` kan man specificera vilka tty enheter som `root` är tillåten att logga in på. Filen `/etc/securetty` läses av loginprogrammet (vanligtvis `/bin/login`). Dess format är en lista av de tty enhetssnamm som är tillåtna, på alla andra är `root` login otillåten:

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

5.10.3 `tcpd` åtkomstkontrollmekanism.

Programmet `tcpd` som man sett listat i `/etc/inetd.conf` tillhandahåller loggnings- och åtkomstkontrollmekanismer för tjänster som är konfigurerat att skydda.

När det aktiveras av programmet `inetd` så läser det två filer som innehåller åtkomstregler och antingen tillåter det eller nekar åtkomst till servern som det skyddar.

5. Generell Information om Nätverkskonfiguration

/etc/hosts.allow  Filen /etc/hosts.allow är en konfigurationsfil för programmet /usr/sbin/tcpd. Filen innehåller regler som beskriver vilka datorer som är tillåtna åtkomst till en tjänst på maskinen. Filformatet är ganska enkelt:

`# /etc/hosts.allow`
`#`
`# <service list>: <host list> [: command]`

`service list`
är en kommaseparerad lista av servernamn som denna regel gäller för. Exempel: `ftpd, telnetd` och `fingerd`.

`host list`

`command`

Ett exempel:

`# /etc/hosts.allow`
`#`
`# Allow mail to anyone`
in.smtpd: `ALL`
`# All telnet and ftp to only hosts within my domain and my host at home.`
telnetd, ftpd: `LOCAL, myhost.athome.org.au`
`# Allow finger to anyone but keep a record of who they are.`
fingerd: `ALL: (finger @%h | mail -s "finger from %h" root)`

/etc/hosts.deny  Filen /etc/hosts.deny är en konfigurationsfil för programmet /usr/sbin/tcpd. Filen innehåller regler som beskriver vilka datorer som är nekade åtkomst till en tjänst på maskinen.

Ett enkelt exempel:

`# /etc/hosts.deny`
`#`
`# Disallow all hosts with suspect hostnames`
`ALL: PARANOID`
`#`


# Disallow all hosts.

ALL: ALL

Raden med PARANOID är egentligen redundant eftersom den andra raden fäller allt i vilket fall. Någon av dessa rader skulle vara tänkbare beroende på vilka krav man har.

Att ha ALL: ALL i /etc/hosts.deny och sedan specifikt ange de tjänster och datorer som man vill ha i filen /etc/hosts.allow är den säkraste konfigurationen.

5.10.4 /etc/hosts.equiv


5.10.5 Konfigurera ftp-daemonen ordentligt.


5.10.6 Brandväggar.

Att inte tillåta datagram att ens nå fram till din maskin eller servrar är ett utmärkt sätt att säkra systemet. Detta beskrivs i Firewall-HOWTO <Firewall-HOWTO.html>.

5.10.7 Andra förslag.

Här är några andra, potentiellt religiösa förslag som man kan tänka på.

sendmail

oavsett dess popularitet så framträder sendmail daemonen med skrämmande jämna mellanrum på säkerhetsmeddelanden. Det är upp till en själv om man väljer att köra den.

NFS och andra Sun RPC tjänster


6 Nätverksspecifik Information.

Följande delsekctioner är specifika för vissa nätverkstekniker. Informationen i dessa sektioner gäller inte nödvändigtvis för någon annan nätverksteknik.
6. Nätverksspecifik Information.

6.1 ARCNet

Enhetsnamn för ARCNet är ’arc0e’, ’arc1e’, ’arc2e’ osv eller ’arc0s’, ’arc1s’, ’arc2s’ osv. Det första kortet som hittas av kärnan tilldelas ’arc0e’ eller ’arc0s’ och resten tilldelas namn sekvensiellt i den ordning som de hittas. Bokstaven på slutet av namnet betecknar om man har valt Ethenet-inkapsling som paketformat eller RFC1051 som paketformat.

Kompileringsalternativ för Kärnan:

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

När man väl har en kärna som stöder sitt Ethernetkort så är konfiguration av kortet lätt.

Man skulle kunna använda något som liknar:

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


ARCNet-stöd utvecklades av Avery Pennarun, apenwarr@foxnet.net.

6.2 Appletalk (AF_APPLETALK)

Appletalk har inga särskilda enhetsnamn eftersom det använder existerande nätverksenheter.

Kompileringsalternativ för Kärnan:

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


För att kompilera och installera paketet gör man ungefär så här:

```
# cd /usr/src
# tar xvfz .../netatalk-1.4b2.tar.Z
- You may want to edit the ‘Makefile’ at this point, specifically to change
  the DESTDIR variable which defines where the files will be installed later.
  The default of /usr/local/atalk is fairly safe.
# make
- as root:
# make install
```
6.2.1 Att konfigurera mjukvaran för Appletalk.

Det första man måste göra för att få det att fungera är att se till att de rätta raderna finns med i filen
/etc/services. Raderna man behöver är:

<table>
<thead>
<tr>
<th>Service</th>
<th>Port</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>rtmp</td>
<td>1/ddp</td>
<td>Routing Table Maintenance Protocol</td>
</tr>
<tr>
<td>nbp</td>
<td>2/ddp</td>
<td>Name Binding Protocol</td>
</tr>
<tr>
<td>echo</td>
<td>4/ddp</td>
<td>AppleTalk Echo Protocol</td>
</tr>
<tr>
<td>zip</td>
<td>6/ddp</td>
<td>Zone Information Protocol</td>
</tr>
</tbody>
</table>

Nästa steg är att skapa konfigurationsfiler för Appletalk i katalogen /usr/local/atalk/etc (eller var man
nu installerade paketet).

Den första filen som behövs är /usr/local/atalk/etc/atalkd.conf. Till en början behöver denna fil endast
en rad som anger namnet på den nätverksenhet som är kopplad nätverket där Applemaskinerna finns:

```bash
eth0
```

Daemonen för Appletalk kommer att lägga dit fler detaljer när den körs.

6.2.2 Att exportera ett Linuxfilsystem via Appletalk.

Man kan exportera filsystem från sin Linuxburk till nätverket så att Applemaskinerna på nätverket kan
använda dem.

För att göra detta så behöver man konfigurera filen /usr/local/atalk/etc/AppleVolumes.system. Det
finns ytterligare en konfigurationsfil som heter /usr/local/atalk/etc/AppleVolumes.default som har
precis samma format och beskriver vilka filsystem som kan användas av användare som ansluter med gästrättigheter.

Alla detaljer om hur man konfigurerar dessa finns i manualbladet för afpd.

Ett enkelt exempel:

```bash
/tmp Scratch
/home/ftp/pub "Public Area"
```

Detta skulle exportera /tmp-filsystemet som en AppleShare-volym ‘Scratch’ och /home/ftp/pub-katalogen
som en AppleShare-volym ‘Public Area’. Volymnamnen är inte obligatoriska, daemonen väljer namn om man
inte anger dem, men det skadar inte att ange dem ändå.

6.2.3 Att dela sin skrivare via Appletalk.

Man kan dela sin Linuxskrivare med sina Applemaskiner ganska enkelt. Man behöver köra programmet papd
(Appletalk Printer Access Protocol Daemon). När man kör detta program så tar det emot förfrågningar från
Applemaskinerna och spoolar utskriftsjobben till den lokala line-printer daemonen.

Man behöver ändra filen /usr/local/atalk/etc/papd.conf för att konfigurera daemonen. Syntaxen för
denna fil är densamma som för den vanliga /etc/printcap filen. Namnet som man ger till definitionen
registreras med Appletalks namnättningprotokoll, NBP.

En exempelkonfiguration kan se ut så här:

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

6.2.4 Att starta programvaran för Appletalk.

Nu bör man vara redo att testa denna enkla konfiguration. Det finns en fil `rc.atalk` som följer med paketet `netatalk` som borde fungera för de flesta, så allt man behöver göra är följande:

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

och allt bör startas och fungera. Man skall inte se några felmeddelanden och programvaran kommer att skicka meddelanden till konsolen som indikerar varje steg som startas.

6.2.5 Att testa programvaran för Appletalk.

För att testa att mjukvaran fungerar som den skall, så går man till en av sina Applemaskiner, tar ner äpplemenyn, väljer Väljaren (Chooser), klickar på AppleShare, och Linuxburken bör synas.

6.2.6 Brister i programvaran för Appletalk.

- Daemonen `afpd` förväntar sig lösenord i klarxtext från Macarna. Säkerhet kan vara ett problem, så man skall vara försiktig när man kör denna daemonen på en dator som är ansluten till Internet, man har sig själv att skylla om någon elak person gör någon skada.
- Existerande diagnostiseringsverktyg som `netstat` och `ifconfig` stöder inte Appletalk. Rå information är tillgänglig i katalogen `/proc/net/` om man behöver den.

6.2.7 Mer information.


6.3 ATM

6. Nätverksspecifik Information.

6.4 AX25 (AF_AX25)

Enhetsnamn för AX.25 är 'sl0', 'sl1', osv i 2.0.* kärnor eller 'ax0', 'ax1', osv i 2.1.* kärnor.

Kompileringsalternativ för Kärnan:

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


Det mesta arbetet för att implementera protokollen har utförts av Jonathon Naylor, jsn@cs.nott.ac.uk.

6.5 DECNet

Stöd för DECNet håller för tillfället på att utvecklas. Man kan räkna med att det dyker upp i sena 2.1.* kärnor.

6.6 EQL - trafikutjämnare för multipla linor.

Enhetsnamnet för EQL är 'eql'. Med standardkärnan kan man endast ha en EQL-enhet per maskin. EQL tillhandahåller hjälpmedel för att använda multipla punkt till punkt förbindelser (tex PPP, SLIP eller PLIP) som en ensam logisk länk för att bära TCP/IP. Ofta är det billigare att använda flera linor med lägre hastighet än att ha en höghastighetslina installerad.

Kompileringsalternativ för Kärnan:

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

För att stödja denna mekanism så måste maskinen på andra sidan av linorna också stödja EQL. Linux, Livingstone Portmasters och nyare dial-in servrar stöder kompatibla tjänster.


Konfigurationen är hyfsat okomplicerad. Man börjar med att konfigurera eql-gränssnittet. Eql-gränssnittet är precis som alla andra nätverksgränssnitt. Man kan konfigurera IP-adressen och MTU genom att använda ifconfig, så ungefär som:

```
ifconfig eql 192.168.10.1 mtu 1006
```

Sedan behöver man manuellt initiera var och en av linorna som man skall använda. Dessa kan vara en valfri kombination av punkt till punkt förbindelser. Hur man initierar de anslutningarna beror på vilken typ av länkar de är, se passande sektioner för mer information.

Till sist skall man associera den seriella länken med EQL-enheten, detta kallas för 'enslaving' och görs med kommandot eql_enslave:

```
eql_enslave eql sl0 28800
eql_enslave eql ppp0 14400
```
6. Nätverksspecifik Information.

Parametern ‘estimated speed’ som man ger till eql enslave gör ingen direkt nytta. Den används av EQL-drivrutinen för att avgöra hur stor del av datagrammen som den enheten skall få ta emot, så man kan finjustera balansen hos linorna genom att ändra på detta värde.

För att disassociera en lina från en EQL-enhet så använder man kommandot eql emancipate:

```
eql_emancipate eql sl0
```

Man lägger till routing som om det vore en normal punkt till punkt förbindelse, förutom att router skall referera till eql enheten istället för de verkliga seriella enheterna:

```
route add default eql
```

EQL-drivrutinen utvecklades av Simon Janes, simon@ncm.com.

6.7 Ethernet

Enhetsnamn för Ethernet är ‘eth0’, ‘eth1’, ‘eth2’ osv. Det första kortet som hittas får namnet ‘eth0’ och resten tilldelas namn sekvensiellt i den ordning som de hittas.

För att ta reda på hur man får sitt Ethernet-kort att fungera i Linux så bör man titta i Ethernet-HOWTO <Ethernet-HOWTO.html>.

När man väl har en kärna som stöder sitt Ethernet-kort så är det enkelt att konfigurera kortet.

Vanligtvis så används ungefär följande:

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

De flesta drivrutinerna för Ethernet utvecklades av Donald Becker, becker@CESDIS.gsfc.nasa.gov.

6.8 FDDI

Enhetsnamnen för FDDI är ‘fddi0’, ‘fddi1’, ‘fddi2’ osv. Det första kortet som hittas får namnet ‘fddi0’ och resten tilldelas namn sekvensiellt i den ordning som de hittas.

Larry Stefani, lstefani@ultranet.com, har utvecklat en drivrutin för EISA och PCI FDDI-kort från Digital Equipment Corporation.

Kompileringsalternativ för Kärnan:

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

När man väl har en kärna som stöder sitt FDDI-kort, så konfigureras FDDI-kortet nästan likadant som Ethernet-kortet. Man behöver bara ange lämpliga FDDI-enhetsnamn till kommandona ifconfig och route.

6.9 Frame Relay

Enhetsnamnen för Frame Relay är ‘dlci00’, ‘dlci01’ osv för DLCI inkapslingsenheter och ‘sdlia0’, ‘sdlia1’ osv för FRAD(s).
6. Nätverksspecifik Information.


Kompileringsalternativ för Kärnan:

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

Mike McLagan, mike.mclagan@linux.org, utvecklade stödet och konfigurationsverktygen för Frame Relay. För närvarande är de FRADs som stöds följande: Sangoma Technologies <http://www.sangoma.com/> SS02A, SS02E och SS08.


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

Efter att man har installerat verktygen skall man skapa en fil som heter /etc/frad/router.conf. Man kan använda följande mall, som är en modifierad version av en exempelfil:

```
# /etc/frad/router.conf
# This is a template configuration for frame relay.
# All tags are included. The default values are based on the code
# supplied with the DOS drivers for the Sangoma S502A card.
#
# A ' # ' anywhere in a line constitutes a comment
# Blanks are ignored (you can indent with tabs too)
# Unknown [] entries and unknown keys are ignored
#
[
[Devices]
Count=1 # number of devices to configure
Dev_1=sdlao # the name of a device
#Dev_2=sdlal # the name of a device

# Specified here, these are applied to all devices and can be overridden for
# each individual board.
#
Access=CPE
Clock=Internal
KBaud=64
Flags=TX
```
# MTU=1500  # Maximum transmit IFrame length, default is 4096
# T391=10   # T391 value 5 - 30, default is 10
# T392=15   # T392 value 5 - 30, default is 15
# N391=6    # N391 value 1 - 255, default is 6
# N392=3    # N392 value 1 - 10, default is 3
# N393=4    # N393 value 1 - 10, default is 4

# Specified here, these set the defaults for all boards
# 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

# Device specific configuration
#
# The first device is a Sangoma S502E
#
[sdla0]
Type=Sangoma  # Type of the device to configure, currently only
           # SANGOMA is recognised
#
# These keys are specific to the 'Sangoma' type
#
# The type of Sangoma board - S502A, S502E, S508
Board=S502E
#
# The name of the test firmware for the Sangoma board
# Testware=/usr/src/frad-0.10/bin/sdla_tst.502
#
# The name of the FR firmware
# Firmware=/usr/src/frad-0.10/bin/frm_rel.502
#
Port=360  # Port for this particular card
Mem=C8    # Address of memory window, A0-EE, depending on card
IRQ=5     # IRQ number, do not supply for S502A
DLCIs=1   # Number of DLCI's attached to this device
DLCI_1=16 # DLCI #1's number, 16 - 991
# DLCI_2=17
# DLCI_3=18
# DLCI_4=19
# DLCI_5=20
#
# Specified here, these apply to this device only,
# and override defaults from above
#
# Access=CPE  # CPE or NODE, default is CPE
# Flags=TXIgnore,RXIgnore,BufferFrames,DropAborted,Stats,MCI,AutoDLCI
# Clock=Internal     # External or Internal, default is Internal
# Baud=128           # Specified baud rate of attached CSU/DSU
# MTU=2048           # Maximum transmit IFrame length, default is 4096
# T391=10            # T391 value 5 - 30, default is 10
# T392=15            # T392 value 5 - 30, default is 15
# N391=6             # N391 value 1 - 255, default is 6
# N392=3             # N392 value 1 - 10, default is 3
# N393=4             # N393 value 1 - 10, default is 4

# The second device is some other card
#
# [sdla1]
# Type=FancyCard     # Type of the device to configure.
# Board=             # Type of Sangoma board
# Key=Value          # values specific to this type of device

# DLCI Default configuration parameters
# These may be overridden in the DLCI specific configurations
#
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

# DLCI Configuration
# These are all optional. The naming convention is
# [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
6. Nätverksspecifik Information.

När man har skapat sin /etc/frad/router.conf fil så är det enda som återstår att konfigurera enheterna. Detta är bara lite svårare än att konfigurera en normal nätverksenhet. Man måste komma ihåg att starta upp FRAD-enheten innan DLCI inkapslingsenheterna.

# Configure the frad hardware and the DLCI parameters
/sbin/fradcfg /etc/frad/router.conf || exit 1
/sbin/dlcicfg file /etc/frad/router.conf
#
# Bring up the FRAD device
ifconfig sdla0 up
#
# Configure the DLCI encapsulation interfaces and routing
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
#

6.10 IP-redovisning (IP Accounting)


Kompileringsalternativ för Kärnan:

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


Man skulle då kunna använda följande:

# Flush the accounting rules
ipfwadm -A -f
#
# Add rules for local ethernet segment
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 20
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 20
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 23
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 23
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 80
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 80
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 513
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 513
ipfwadm -A in -a -P tcp -D 44.136.8.96/29
ipfwadm -A out -a -P tcp -D 44.136.8.96/29
ipfwadm -A in -a -P udp -D 44.136.8.96/29
ipfwadm -A out -a -P udp -D 44.136.8.96/29
ipfwadm -A in -a -P icmp -D 44.136.8.96/29
ipfwadm -A out -a -P icmp -D 44.136.8.96/29

# Rules for default
ipfwadm -A in -a -P tcp -D 0/0 20
ipfwadm -A out -a -P tcp -S 0/0 20
ipfwadm -A in -a -P tcp -D 0/0 23
ipfwadm -A out -a -P tcp -S 0/0 23
ipfwadm -A in -a -P tcp -D 0/0 80
ipfwadm -A out -a -P tcp -S 0/0 80
ipfwadm -A in -a -P tcp -D 0/0 513
ipfwadm -A out -a -P tcp -S 0/0 513
ipfwadm -A in -a -P tcp -D 0/0
ipfwadm -A out -a -P tcp -D 0/0
ipfwadm -A in -a -P udp -D 0/0
ipfwadm -A out -a -P udp -D 0/0
ipfwadm -A in -a -P icmp -D 0/0
ipfwadm -A out -a -P icmp -D 0/0

# List the rules
ipfwadm -A -l -n

Det sista kommandot listar var och en av redovisningsreglerna och visar de ihopsamlade summorna.

En viktig notering är att när man analyserar datan är att **summan för alla regler som passar in kommer att ökas** så för att erhålla summor för enskilda protokoll så måste man räkna lite grann. Om jag tex ville veta hur mycket data som inte var ftp, telnet, rlogin eller www så skulle jag subtrahera de individuella summorna från den regel som passar in på alla portarna.

# ipfwadm -A -l -n
IP accounting rules

<table>
<thead>
<tr>
<th>pkts</th>
<th>bytes</th>
<th>dir</th>
<th>prot</th>
<th>source</th>
<th>destination</th>
<th>ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>in</td>
<td>tcp</td>
<td>0.0.0.0/0</td>
<td>44.136.8.96/29</td>
<td>* -&gt; 20</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>out</td>
<td>tcp</td>
<td>44.136.8.96/29</td>
<td>0.0.0.0/0</td>
<td>20 -&gt; *</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>in</td>
<td>tcp</td>
<td>0.0.0.0/0</td>
<td>44.136.8.96/29</td>
<td>* -&gt; 23</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>out</td>
<td>tcp</td>
<td>44.136.8.96/29</td>
<td>0.0.0.0/0</td>
<td>23 -&gt; *</td>
</tr>
<tr>
<td>10</td>
<td>1166</td>
<td>in</td>
<td>tcp</td>
<td>0.0.0.0/0</td>
<td>44.136.8.96/29</td>
<td>* -&gt; 80</td>
</tr>
<tr>
<td>10</td>
<td>572</td>
<td>out</td>
<td>tcp</td>
<td>44.136.8.96/29</td>
<td>0.0.0.0/0</td>
<td>80 -&gt; *</td>
</tr>
<tr>
<td>242</td>
<td>9777</td>
<td>in</td>
<td>tcp</td>
<td>0.0.0.0/0</td>
<td>44.136.8.96/29</td>
<td>* -&gt; 513</td>
</tr>
<tr>
<td>220</td>
<td>18198</td>
<td>out</td>
<td>tcp</td>
<td>44.136.8.96/29</td>
<td>0.0.0.0/0</td>
<td>513 -&gt; *</td>
</tr>
<tr>
<td>252</td>
<td>10943</td>
<td>in</td>
<td>tcp</td>
<td>0.0.0.0/0</td>
<td>44.136.8.96/29</td>
<td>* -&gt; *</td>
</tr>
</tbody>
</table>
6. Nätverksspecifik Information.

6.11 IP Aliasing

Det finns vissa applikationer där det är användbart att kunna tilldela flera IP-adresser till en och samma nätverksenhet. Titta i IP-Aliasing mini-HOWTO för mer information än vad man hittar här.

Kompileringssalternativ för Kärnan:

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

När man har kompilerat och installerat sin kärna med stöd för IP Aliasing så är det väldigt enkelt att konfigurera. Aliasen läggs till virtuella nätverksenheter som är associerade med den verkliga enheten. En enkel namnkonvention används, nämligen `<enhetsnamn>:<virtuellt enhetsnummer>`, tex `eth0:0`, `ppp0:1` osv. Notera att en virtuell enhet endast kan konfigureras efter det att den riktiga enheten har konfigurerats.

Till exempel, antag att man har ett Ethernet-nätverk som innehåller två olika IP-subnät på en gång. Man vill nu att maskinen skall ha direkt access till båda. Man skulle använda något som:

```
# ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# ifconfig eth0:0 192.168.10.1 netmask 255.255.255.0 up
# route add -net 192.168.10.0 netmask 255.255.255.0 eth0:0
#
```

För att ta bort ett alias så lägger man till ett `-` i slutet på dess namn:

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

Alla router som är relaterade till det aliaset kommer också att tas bort automatiskt.
6.12 IP Brandväggar (IP Firewalls)


Kompileringsalternativ för Kärnan:

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

Konfigurationen av reglerna för brandväggen görs med hjälp av kommandot `ipfwadm`. Som jag nämnde tidigare så är jag ingen expert på säkerhet, så även om jag visar ett exempel som går att använda så rekommenderas egna undersökningar i ämnet om säkerhet är särskilt viktigt.

Det kanske vanligaste användningsområdet för en brandvägg är när man använder sin Linuxburk som en router och brandvägg för att skydda sitt lokala nätverk mot otillåten åtkomst utifrån.

Följande konfiguration är baserad på ett bidrag från Arnt Gulbrandsen, <agulbra@troll.no>.

Exemplet beskriver en konfiguration av brandväggsreglerna i Linux-brandväggen/routern som illustreras i denna figur:

```
---
| 172.16.37.0
| /255.255.255.0
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>172.16.174.30</td>
</tr>
<tr>
<td>NET ============= f/w ========= ..37.19</td>
</tr>
</tbody>
</table>
| PPP | router | ------
| / | Mail |
| / | /DNS |
| ---
---
```

Kommandona som följer placeras normalt i en `rc` fil så att de startas automatiskt varje gång systemet startas. För maximal säkerhet så borde de utföras efter det att nätverksgränssnitten konfigurerats, men innan enheterna aktiveras så att man på detta sätt hindrar att någon får tillgång till maskinen medan den startar upp.

```
#!/bin/sh

# Flush the 'Forwarding' rules table
# Change the default policy to 'accept'
# /sbin/ipfwadm -F -f
```
/sbin/ipfwadm -F -p accept
#
# .. and for 'Incoming'
#
/sbin/ipfwadm -I -f
/sbin/ipfwadm -I -p accept

# First off, seal off the PPP interface
# I'd love to use '-a deny' instead of '-a reject -y' but then it
# would be impossible to originate connections on that interface too.
# The -o causes all rejected datagrams to be logged. This trades
# disk space against knowledge of an attack of configuration error.
#
/sbin/ipfwadm -I -a reject -y -o -P tcp -S 0/0 -D 172.16.174.30

# Throw away certain kinds of obviously forged packets right away:
# Nothing should come from multicast/anycast/broadcast addresses
#
/sbin/ipfwadm -F -a deny -o -S 224.0/3 -D 172.16.37.0/24
#
# and nothing coming from the loopback network should ever be
# seen on a wire
#
/sbin/ipfwadm -F -a deny -o -S 127.0/8 -D 172.16.37.0/24

# accept incoming SMTP and DNS connections, but only
# to the Mail/Name Server
#
/sbin/ipfwadm -F -a accept -P tcp -S 0/0 -D 172.16.37.19 25 53
#
# DNS uses UDP as well as TCP, so allow that too
# for questions to our name server
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 -D 172.16.37.19 53
#
# but not "answers" coming to dangerous ports like NFS and
# Larry McVoy's NFS extension. If you run squid, add its port here.
#
/sbin/ipfwadm -F -a deny -o -P udp -S 0/0 53 \
-D 172.16.37.0/24 2049 2050

# answers to other user ports are okay
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 53 \
-D 172.16.37.0/24 53 1024:65535

# Reject incoming connections to identd
# We use 'reject' here so that the connecting host is told
# straight away not to bother continuing, otherwise we'd experience
# delays while ident timed out.
#
/sbin/ipfwadm -F -a reject -o -P tcp -S 0/0 -D 172.16.37.0/24 113

# Accept some common service connections from the 192.168.64 and
# 192.168.65 networks, they are friends that we trust.
6. Nätverksspecifik Information.

```bash
# /sbin/ipfwadm -F -a accept -P tcp -S 192.168.64.0/23
  -D 172.16.37.0/24 20:23

# accept and pass through anything originating inside
#
/sbin/ipfwadm -F -a accept -P tcp -S 172.16.37.0/24 -D 0/0

# deny most other incoming TCP connections and log them
# (append 1:1023 if you have problems with ftp not working)
#
/sbin/ipfwadm -F -a deny -o -y -P tcp -S 0/0 -D 172.16.37.0/24

# ... for UDP too
#
/sbin/ipfwadm -F -a deny -o -P udp -S 0/0 -D 172.16.37.0/24
```

Att göra en bra konfiguration av brandväggen är lite trixigt. Exemplet ovan borde dock vara en hyfsad startpunkt. Manualbladet för kommandot `ipfwadm` innehåller mer hjälp om hur man använder det. Om man tänker konfigurera en brandvägg, så skall man se till att fråga runt och få så mycket råd man kan ifrån källor som man anser vara pärlitliga. Låt sedan någon testa konfigurationen från utsidan.

6.13 IPIP Inkapsling (IPIP Encapsulation)


**Kompileringsalternativ för Kärnan:**

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

Tunnelingsenheterna heter ‘tunl0’, ‘tunl1’ osv.


6.13.1 En tunnlad nätverkskonfiguration.

Som alltid, så tycker jag att en figur fungerar bättre än en massa förvirrande text, så här kommer en:
6. Nätverksspecifik Information.

192.168.1/24 192.168.2/24

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


Linux routern ‘A’ skulle konfigureras med:

```
# PATH=/sbin:/usr/sbin
#
# Ethernet configuration
ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# ppp0 configuration (start ppp link, set default route)
pppd
route add default ppp0
#
# Tunnel device configuration
ifconfig tunl0 192.168.1.1 up
route add -net 192.168.2.0 netmask 255.255.255.0 gw fff.ggg.hhh.iii tunl0
```

Linux routern ‘B’ skulle konfigureras med:

```
# PATH=/sbin:/usr/sbin
#
# Ethernet configuration
ifconfig eth0 192.168.2.1 netmask 255.255.255.0 up
route add -net 192.168.2.0 netmask 255.255.255.0 eth0
#
# ppp0 configuration (start ppp link, set default route)
pppd
route add default ppp0
#
# Tunnel device configuration
ifconfig tunl0 192.168.2.1 up
```
Nätkomander:

```
route add -net 192.168.1.0 netmask 255.255.255.0 gw aaa.bbb.ccc.ddd tunl0
```

Kommandot:

```
rout add -net 192.168.1.0 netmask 255.255.255.0 gw aaa.bbb.ccc.ddd tunl0
```

betyder: Skicka alla datagrammäntade för 192.168.1.0/24 inuti ett IPIP-inkapslat datagram med destinationadressen aaa.bbb.ccc.ddd.


6.13.2 En tunnlad datorkonfiguration.


```
192.168.1/24
```

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

Linux routern ‘A’ skulle konfigureras med:

```
# PATH=/sbin:/usr/sbin
#
# Ethernet configuration
ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# ppp0 configuration (start ppp link, set default route)
pppd
route add default ppp0
#
# Tunnel device configuration
ifconfig tunl0 192.168.1.1 up
route add -host 192.168.1.12 gw fff.ggg.hhh.iii tunl0
#
# Proxy ARP for the remote host
```
6. Nätverksspecifik Information.

Linuxdatorn ’B’ skulle konfigureras med:

```bash
# PATH=/sbin:/usr/sbin
#
# ppp0 configuration (start ppp link, set default route)
pppd
  route add default ppp0
#
# Tunnel device configuration
ifconfig tunl0 192.168.1.12 up
  route add -net 192.168.1.0 netmask 255.255.255.0 gw aaa.bbb.ccc.ddd tunl0
```

Denna typ av konfiguration är mer typisk för Mobil IP. Där en enskild dator vill flytta omkring på Internet och hela tiden använda samma IP-adress. Mer information om detta finns i sektionen om Mobil IP.

6.14 IPX (AF_IPX)

IPX-protokollet används mest i LAN-omgivningar med Novell NetWare (tm). Linux har stöd för att kunna agera som en nätverksändpunkt, eller som en router för IPX.

**Kompletteringsoptioner för Kärnan:**

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

IPX-protokollet och NCPFS täcks mer detaljerat i *IPX-HOWTO* <IPX-HOWTO.html>.

6.15 IPv6


Linux har redan en fungerande, men inte komplett, IPv6-implementation i version 2.1.* av kärnan.


6.16 ISDN

inkommande och utgående anslutningar. Den ursprungliga avsikten med ISDN var att telefonbolagen ville kunna erbjuda en enda datatjänst som kunde ge anstifterna telefon- (digitalt) eller datatjänster till hem och kontor utan att kunden skulle behöva ändra någon konfiguration.


Kompileringsalternativ för Kärnan:

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

Implementationen av ISDN i Linux stöder ett antal olika interna ISDN-kort. Dessa finns listade i konfigurationen för kärnan:

- ICN 2B and 4B
- Octal PCBIT-D
- Teles ISDN-cards and compatibles

Några av dessa kort kräver att man laddar hem särskild programvara för att de skall fungera. Det finns ett separat verktyg att göra detta med.


### 6.17 IP-maskering (IP Masquerade).

Kompileringsalternativ för Kärnan:

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)

Vanligtvis har man sin Linuxburk konfigurerad för uppringd SLIP eller PPP, precis som om det var en fristående dator. Dessutom så skulle den ha en ytterligare nätverksenhet konfigurerad, kanske Ethernet som är konfigurerat med en reserverad nätverksadress. Datorerna som skall förklädas skulle då finnas på det nätverket. Var och en av de datorerna skulle ställa in IP-adressen på Linuxboxens Ethernet-kort som ‘default gateway’ eller router. Observera att de andra datorerna (de som inte har den uppringda anslutningen) inte behöver konfigureras med IP-maskering, de behöver endast veta vilken dator som är ‘default gateway’ (SvÖ).

En typisk konfiguration:

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

De viktigaste kommandona för denna konfiguration är:

```
# Network route for ethernet
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# Default route to the rest of the internet.
route add default ppp0
#
# Cause all hosts on the 192.168.1/24 network to be masqueraded.
ipfwadm -F -a m -S 192.168.1.0/24 -D 0.0.0.0/0
```


6.18 IP Transparent Proxy


Kompileringsalternativ för Kärnan:
6.18 Mobil IP


6.20 Multicast


Kompileringsalternativ för Kärnan:
6. Nätverksspecifik Information.

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


6.21 NAT - Översättning av nätverksadresser (Network Address Translation)

NAT är en standardiserad storebro till Linux IP-maskering. Det finns en detaljerad specifikation i RFC1631. NAT tillhandahåller funktioner som IP-maskering inte gör vilket gör det mer passande för användning i brandväggsroutrar hos företag och i större installationer.

En alpha-implementation av NAT för Linux 2.0.29 kärnan har utvecklats av Michael Hasenstein, Michael.Hasenstein@informatik.tu-chemnitz.de. Michaels dokumentation och implementation finns på Linux IP Network Address Web Page <http://www.csn.tu-chemnitz.de/HyperNews/get/linux-ip-nat.html>

Nyare Linux 2.1.* kärnor har också viss NAT-funktionalitet i routingalgoritmen.

6.22 NetRom (AF_NETROM)

Enhetsnamn för NetRom är ‘nr0’, ‘nr1’, osv.

Kompileringsalternativ för Kärnan:

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

Protokollen AX25, Netrom och Rose finns beskrivna i AX25-HOWTO <AX25-HOWTO.html>. Dessa protokoll används av radioamatörer i hela världen i experiment med paketradio.

Det mesta arbetet med implementationen av dessa protokoll har gjorts av Jonathon Naylor, jsn@cs.nott.ac.uk.

6.23 PLIP (Parallel Line Internet Protocol)

Enhetsnamn för PLIP är ‘plip0’, ‘plip1 and plip2.

Kompileringsalternativ för Kärnan:

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

PLIP, är som SLIP i den mening att det används för att skapa en punkt till punkt nätverksförbindelse mellan två maskiner. Men det skiljer sig genom att det är designat för att använda de parallella skrivarportarna på datorn istället för de seriella (ett kabelschema finns längre fram i dokumentet). Eftersom det är möjligt att överföra mer än en bit åt gången med en parallellport, så är det möjligt att uppnå högre hastigheter med PLIP-gränsnitet än vad man gör med seriell enhet. Dessutom kan även den enklaste av alla parallellportar, skrivarporten, användas i stället för att man skall behöva köpa jämförelsevis dyra 16550AFN UARTs till de
6. Nätverksspecifikt Information.

seriella portarna. PLIP använder dock mycket CPU-tid jämfört med en seriell länk och är naturligtvis inget bra val om man kan få tag på några billiga Ethernet-kort, men det fungerar om inget annat finns tillgängligt och det fungerar dessutom ganska bra. Man kan förvänta sig en överföringshastighet på ungefär 20 kB/s när en länk fungerar bra.


Notera att vissa bärbara datorer använder chipsets som inte fungerar med PLIP därför att de inte tillåter vissa kombinationer av signaler som PLIP behöver, som skrivare inte använder.

Linux PLIP-gränssnitt är kompatibelt med Crynwyr Packet Driver PLIP vilket betyder att man kan ansluta sin Linuxburk till en DOS-maskin som kör en annan typ av TCP/IP via PLIP.

I 2.0.* kärnor är PLIP-enheterna mappade mot I/O-port och IRQ som följer:

<table>
<thead>
<tr>
<th>device</th>
<th>i/o</th>
<th>IRQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>plip0</td>
<td>0x3bc</td>
<td>5</td>
</tr>
<tr>
<td>plip1</td>
<td>0x378</td>
<td>7</td>
</tr>
<tr>
<td>plip2</td>
<td>0x278</td>
<td>2</td>
</tr>
</tbody>
</table>

Om man inte har parallellportar som stämmer överens med någon av ovanstående kombinationer så kan man ändra en ports IRQ med kommandot `ifconfig` och parameter `‘irq’`. Man måste då slå på IRQ på skrivarportarna i sitt ROM BIOS (om det stöder det).

I senare 2.1.* kärnor med Plug’n’Play stöd så allokeras PLIP-enheterna sektionsvis när de hittas precis som Ethernet-enheterna.


För att konfigurera ett PLIP-gränssnitt så måste man lägga till följande rader i sin `rc`-fil för nätverket:

```
#
# Attach a PLIP interface
#
# configure first parallel port as a plip device
/sbin/ifconfig plip0 IPA.IPA.IPA.IPA pointopoint IPR.IPR.IPR.IPR up
#
# End plip
```

Där:

**IPA.IPA.IPA.IPA**

representerar ens egen IP-adress.

**IPR.IPR.IPR.IPR**

representerar IP-adress på den andra maskinen.
6. Nätverksspecifik Information.

Parametern pointopoint har samma betydelse som för SLIP, den specificerar adressen på maskinen i andra änden av länken.

I nästan alla fall kan man behandla PLIP-gränssnittet som om det var ett SLIP-gränssnitt, förutom att varken dip eller slattach behöver, eller kan användas.

Mer information om PLIP kan hittas i: PLIP-mini-HOWTO <mini/PLIP>

6.24 PPP (Point to Point Protocol)

Enhetsnamn för PPP är 'ppp0', 'ppp1', osv. Enheter numrerades sekvensiellt och den första enheten som konfigurerades får 'ppp0'.

Kompileringsalternativ för Kärnan:

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

Detaljer om PPP-konfiguration finns i PPP-HOWTO <PPP-HOWTO.html>.

6.24.1 Att vidhålla en permanent anslutning till nätet med pppd.

Om man har en semi-permanent anslutning till nätet och vill att ens maskin automatiskt skall återuppta PPP-anslutningen om den bryts så finns det ett enkelt trix som gör detta:

Konfigurera PPP så att det kan startas genom att root-användaren ger kommandot:

```
# pppd
```

Se till att '-detach' parametern finns konfigurerad i filen /etc/ppp/options. Sedan skall följande rader läggas in i filen /etc/inittab, tillsammans med getty-definitionerna:

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

Detta gör så att programmet init håller koll på pppd och automatiskt startar om det ifall det dör.

6.25 Rose protokollet (AF_ROSE)

Enhetsnamn för Rose är 'rs0', 'rs1', osv. i 2.1.* kärnor. Rose finns endast i 2.1.* kärnor.

Kompileringsalternativ för Kärnan:

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

Protokollen AX25, Netrom och Rose finns beskrivna i AX25-HOWTO <AX25-HOWTO.html>. Dessa protokoll används av radioamatörer i hela världen i experiment med paketradio.

Det mest arbetet med implementationen av dessa protokoll har gjorts av Jonathon Naylor, jsn@cs.nott.ac.uk.
6. Nätverksspecifik Information.

6.26 SAMBA (stöd för ‘NetBEUI’, ‘NetBios’).

SAMBA är en implementation av protokollet ‘Session Management Block’. Med SAMBA kan system från bland annat Microsoft använda diskar och skrivare i en Linuxbox.

Detaljer om SAMBA och dess konfiguration finns i SMB-HOWTO <SMB-HOWTO.html>.

6.27 SLIP (Serial Line Internet Protocol) klient.

Enhetsnamn för SLIP är ‘sl0’, ‘sl1’ osv. Enheter numreras sekvensiellt och den första enheten som konfigureras får ‘sl0’.

Kompileringsalternativ för Kärnan:

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

Med SLIP kan man använda TCP/IP över en seriell lina, som kan vara en telefonledning och modem, eller en hyrd ledning av något slag. För att kunna använda SLIP behöver man ha tillgång till en SLIP-server i sitt närområde. Många universitet och företag i hela världen tillhandahåller SLIP-åtkomst.

SLIP använder de seriella portarna på datorn för att bära IP-datagram. För att göra detta måste det ha kontroll över de seriella enheterna. SLIP-enheter benämns med sl0, sl1 osv. Hur motsvarar detta de seriella enheterna? Nätverkskoden använder vad som heter ett ioctl-anrop (i/o control) för att ändra de seriella enheterna till SLIP-enheter. Det finns två program som kan göra detta, de heter dip och slattach.

6.27.1 dip (Dialup IP)

dip är ett smart program som kan ställa in hastigheten på den seriella enheten, kommendera modemet att ringa upp den andra ändan av länken, automatiskt logga in på servern, söka efter meddelanden som man får från servern och ta fram information från dem som tex IP-adress och dessutom kan programmet utföra ioctl-kommandot som behövs för att sätta den seriella porten i SLIP-läge. dip har ett kraftfullt scriptspråk i vilket man kan automatisera sitt inloggningsofvarande.


För att installera det, försök med följande:

```
# cd /usr/src
# gzip -dc dip3370-uri.tgz | tar xvf -
# cd dip-3.3.70

<edit Makefile>

# make install
```

Makefile antar att det finns en grupp som heter uucp, men om man vill kan man ändra det till dip eller SLIP beroende på sin konfiguration.
6. Nätverksspecifik Information.

6.27.2 slattach


6.27.3 När använder man vilket?


Först och främst så är SLIP-länkar olika Ethernet-nätverk i den meningen att det alltid endast finns två datorer på nätverket, en i varje ända av länken. Till skillnad från Ethernet, som är tillgängligt så fort man är inkopplad, så måste man kanske, beroende på typen av länk, initialisera anslutningen på något speciellt sätt.

Om man använder *dip* så gör man det normalt inte vid systemstarten, utan senare när man är redo att använda länken. Det är möjligt att automatisera den processen. Om man använder *slattach* så vill man antagligen lägga till några rader i sin *rc.inet1*-fil. Detta beskrivs snart.


6.27.4 Statisk SLIP-server med uppringd förbindelse och *dip*


Om det är såhär ens SLIP-server fungerar så kan man gå vidare till avsnittet ‘Att använda *dip*’ för att få reda på hur man konfigurerar *dip*.

6.27.5 Dynamisk SLIP-server med uppringd förbindelse och *dip*.

En *dynamisk* SLIP-server är en server där man slumpvis allokeras en IP-adress, från en pool med adresser, varje gång man loggar på. Detta betyder att det inte finns någon garanti för att man har en viss adress varje gång och att adressen kan användas av någon annan efter det att man har loggat av. Administratören för SLIP-servrern har angett en mängd IP-adresser där servern väljer den första lediga när den får en ny
anslutning varefter den guidar användaren igenom loginprocessen och sedan skriver ett välkomstmeddelande som innehåller IP-adressen som sedan används under resten av anslutningen.

Konfigurationen för denna typ av server liknar den för en statisk server förutom att man måste lägga till ett steg där man tar emot IP-adressen som servern har allokerat och konfigurera SLIP-enheten med den.

Återigen så gör dip det hårda jobbet och nyare versioner är tillräckligt smarta för att inte bara logga in, utan även ta reda på IP-adressen och spara den så att man kan konfigurera SLIP-enheten med den.

Om det är såhär ens SLIP-server fungerar så kan man gå vidare till avsnittet ‘Att använda dip’ för att få reda på hur man konfigurerar dip.

### 6.27.6 Att använda dip.

Som nämnts tidigare så är dip ett kraftfullt program som kan förenkla och automatisera processen där man ringer upp SLIP-servern, loggar in, aktivera anslutningen och konfigurera SLIP-enheter med lämpliga `ifconfig` och `route` kommandon.


```bash
# sample.dip Dialup IP connection support program.
#
# This file (should show) shows how to use the DIP
# This should work for Annex type dynamic servers, if you
# use a static address server then use the sample.dip file that
# comes as part of the dip337-uri.tgz package.
#
# Version: 0(#)sample.dip 1.40 07/20/93
#
# Author: Fred N. van Kempen, <waltje@walt.NL.Mugnet.ORG>
#
main:
# Next, set up the other side's name and address.
# My dialin machine is called 'xs4all.hacktic.nl' (== 193.78.33.42)
get $remote xs4all.hacktic.nl
# Set netmask on sl0 to 255.255.255.0
netmask 255.255.255.0
# Set the desired serial port and speed.
port cua02
speed 38400

# Reset the modem and terminal line.
# This seems to cause trouble for some people!
reset
```
# Note! "Standard" pre-defined "errlevel" values:
# 0 - OK
# 1 - CONNECT
# 2 - ERROR
#
# You can change those grep'ping for "addchat()" in *.c...

# Prepare for dialing.
send ATQ0V1E1X4\r
wait OK 2
if $errlvl != 0 goto modem_trouble
dial 555-1234567
if $errlvl != 1 goto modem_trouble

# We are connected. Login to the system.
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:

# We are now logged in.
wait SOMEPROMPT 30
if $errlvl != 0 goto prompt_error

# Command the server into SLIP mode
send SLIP\n
wait SLIP 30
if $errlvl != 0 goto prompt_error

# Get and Set your IP address from the server.
# Here we assume that after commanding the SLIP server into SLIP
# mode that it prints your IP address
get $locip remote 30
if $errlvl != 0 goto prompt_error

# Set up the SLIP operating parameters.
get $mtu 296
# Ensure "route add -net default xs4all.hacktic.nl" will be done
default

# Say hello and fire up!
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

Ovanstående exempel förutsätter att man kontaktar en dynamisk SLIP-server, om man kontaktar en statisk server så bör sample.dip som följer med dip337-uri.tgz fungera bra.

Där dip får kommandot get $local så söker det igenom den inkommande texten efter en textsträng som ser ut som en IP-adress, dvs tal som är separerade med '.'-tecken. Denna modifiering gjordes speciellt för att fungera med dynamiska SLIP-servrar, så attprocessen att läsa in IP-adressen från servern skulle bli automatiserad.

Ovanstående exempel skapar automatiskt en ‘default route’ via SLIP-länken, om det inte är detta man vill, om man tex har en Ethernet-anslutning som skall vara ‘default route, så tar man bort kommandot default från scriptet.

Om man, då scriptet kört färdigt, ger kommandot ifconfig så ser man att det finns en enhet sl0. Detta är SLIP-enheten. Om så behövs så kan man modifiera dess konfiguration manuellt efter det att dip har kört färdigt, genom att använda kommandona ifconfig och route.

Notera att med dip kan man välja ett antal olika protokoll att använda med mode-kommandot, det vanligaste är cSLIP för SLIP med kompression. Notera att båda ändar av länken måste komma överens, så man måste välja det som ens server är inställt på.

Ovanstående exempel är hyfsat stabilt och skall klara av de flesta felen. Titta annars i manualbladet för dip för mer information. Naturligtvis så kan man skriva scriptet så att det till exempel försöker ringa upp servern igen om det inte lyckas få en anslutning inom en given tid, eller till och med försöka med en serie olika servrar om man har tillgång till mer än en.

6.27.7 Permanent SLIP-anslutning med hyrd ledning och slattach.

Om man har en kabel mellan två maskiner, eller är lyckligt lottad och har en hyrd ledning, eller någon annan permanent seriell anslutning mellan sin maskin och en annan, så behöver man inte besvär sig med att använda dip för att sätta upp en seriell länk. slattach är ett väldigt enkelt verktyg att använda som har precis tillräckligt med funktionalitet för att konfigurera en anslutning.

Eftersom anslutningen är permanent så vill man lägga till några kommandon i sin rc.inet1-fil. Det enda man egentligen behöver göra för en permanent anslutning är att konfigurera den seriella enheten till korrekt hastighet och ställa in den i SLIP-läge. Med slattach kan man göra detta i ett enda kommando. Man lägger till följande i sin rc.inet1-fil:

```bash
#
```
6. Nätverksspecifik Information.

# Attach a leased line static SLIP connection
#
# configure /dev/cua0 for 19.2kbps and cslip
/sbin/slattach -p cslip -s 19200 /dev/cua0 &
/sbin/ifconfig sl0 IPA.IPA.IPA.IPA pointopoint IPR.IPR.IPR.IPR up
#
# End static SLIP.

Där:

IPA.IPA.IPA.IPA

representerar IP-adressen.

IPR.IPR.IPR.IPR

representerar the IP-adressen i andra ändan.

slattach allokerar den första icke allokerade SLIP-enheten till den specifikerade seriella enheten. slattach börjar börjar med sl0. Därför så parar slattach ihop sl0 med den specifikerade seriella enheten och därefter sl1 osv.


6.28 SLIP (Serial Line Internet Protocol) server.

Om man har en maskin, som kanske är nätverksanslutet, som man vill att andra skall kunna ansluta till och använda nätverkstjänster, så skall man konfigurera sin maskin som en server. Om man vill använda SLIP som protokoll så har man för närvarande tre valmöjligheter för hur man skall konfigurera Linuxboxen som en SLIP-server. Mitt råd skulle vara att använda det som presenteras först, sliplogin, eftersom det verkar vara det enklaste att konfigurera och förstå, men jag kommer att presentera en sammanfattning av varje så att man kan bilda sig egen uppfattning.

6.28.1 SLIP-server med sliplogin.


De som ansluter kommer att logga in på vanligt vis med användarnamn och lösenord, men istället för att få ett shell efter de loggat in så exekveras sliplogin, som söker i sin konfigurationsfil (/etc/slip.hosts) efter en rad med ett loginnamn som motsvarar det som använts vid inloggningen. Om detta hittas så konfigurerar sliplogin linjen som en ren 8 bitars länk och använder ioctl för att konvertera länken till en SLIP-länk. Sedan återstår ett steg där sliplogin kör ett shellscript som konfigurerar SLIP-gränssnittet med relevant IP-adress, nätmask och routing. Scriptet heter normalt /etc/slip.login, men på liknande sätt som för getty om man har särskilda användare som behöver speciell initiering, så kan man skapa script som heter /etc/slip.login.loginname som körs istället för det allmänna.

Det finns antingen tre eller fyra filer att konfigurera för att sliplogin skall fungera. Jag skall i detalj gå igenom hur man får tag på programvara och hur man konfigurerar detta. Filerna är:

- /etc/passwd, för användarkonton.
• /etc/slip.hosts, för att innehålla information som är unik för varje användare.

• /etc/slip.login, vilken tar hand om konfiguration av routing.

• /etc/slip.tty, vilken endast behövs om man skall konfigurera en server som skall använda dynamisk adressallokering och innehåller en tabell med adresser att allokeras.

• /etc/slip.logout, vilken innehåller kommandon för att stänga upp efter det att en användare har lagt på eller loggat ut.


För att se till att bara auktoriserade användare skall kunna köra sliplogin, så bör man lägga till en rad i filen /etc/group, ungefär som följande:

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

När man installerar sliplogin-paketet så kommer Makefile att ändra ägargruppen för kommandot sliplogin till slip, och det betyder att endast användare som tillhör den gruppen kan köra programmet. I exemplet ovan kan endast radio och fred köra sliplogin.

För att installera binärfilerna i katalogen /sbin gör man följande:

```
# cd /usr/src
# gzip -dc .../sliplogin-2.1.1.tar.gz | tar xvf -
# cd sliplogin-2.1.1
# <..edit the Makefile if you don't use shadow passwords..>
# make install
```

Om man vill kompilera om binärfilerna innan man installerar så skall man göra make clean innan man kör make install. Om man vill installera filerna i någon annan katalog så får man ändra i filen Makefile under regeln install.

Läs filen README som följer med paketet för mer information.

Att konfigurera /etc/passwd för SLIP. Normalt skulle man skapa särskilda login för SLIP-användare i filen /etc/passwd. En konvention som ofta följs är att använda hostname för den anslutande datorns med ett stort ‘S’ i början. Så, till exempel, om den anslutande datorn heter radio så kan man lägga en rad i /etc/passwd som ser ut såhär:

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

Men det spelar egentligen ingen roll vad kontot heter, bara det betyder något för administratören av SLIP-servern.

6. Nätverksspecifik Information.

**Att konfigurera /etc/slip.hosts** Filen `/etc/slip.hosts` är filen som `sliplogin` söker igenom efter rader som stämmer överens med loginnamnet för att få tag på konfigurationsdetaljer för denna användare. Det är filen där man specificerar IP-adressen och nätmasken som kommer att tilldelas användaren. Exempelrader för två datorer, en statisk konfiguration för `radio` och en annan, dynamisk konfiguration för `albert` kan se ut så här:

```
# 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
#
```

Parametrarna i `/etc/slip.hosts` är:

1. loginnamnet för användaren.
2. IP-adress för servermaskinen (dvs denna dator)
3. IP-adress som användaren blir tilldelad. Om detta fält innehåller `DYNAMIC` så kommer IP-adressen att allokeras baserat på informationen som finns i filen `/etc/slip.tty`. **Notera:** man måste ha åtminstone version 1.3 av `sliplogin` för att detta skall fungera.
4. nätmasken som tilldelas den anslutande maskinen.
5. SLIP-läge där man kan sätta på/stänga av kompression och andra funktioner. Tillåtna värden är `normal` eller `compressed`.
6. en timeoutparameter som specificerar hur lång tid länken kan vara använd innan den automatiskt avbryts. Ett negativt värde stänger av funktionen.
7. valfria argument.


De vanligaste SLIP-lägena är:

- **normal**
  
  att slå på normal SLIP utan kompression.

- **compression**
  
  att slå på van Jacobsen header compression (cSLIP)

Man kan endast använda en åt gången. För mer information om de andra valmöjligheterna som finns, se manualbladen.
Att konfigurera `/etc/slip.login`. När `sliplogin` har hittat en passande rad i `/etc/slip.hosts` så kommer programmet att försöka exekvera filen `/etc/slip.login` för att konfigurera SLIP-gränsnittet med dess IP-adress och nätmask.

Exemplet på en `/etc/slip.login` som följer med `sliplogin`-paketet ser ut så här:

```bash
#!/bin/sh -
#
# @(#)slip.login 5.1 (Berkeley) 7/1/90
#
# generic login file for a SLIP line. sliplogin invokes this with
# the parameters:
# $1 $2 $3 $4, $5, $6 ...
# SLIPunit ttyspeed pid the arguments from the slip.host entry
#
/sbin/ifconfig $1 $5 pointopoint $6 mtu 1500 -trailers up
/sbin/route add $6
arp -s $6 <hw_addr> pub
exit 0
#
```


Notera också användandet av `Proxy ARP` för att se till att andra datorer på samma Ethernet som servern kommer att nå den anslutna datorn. Fältet `<hw_addr>` skall vara hårdvaruadressen på Ethernet-kortet i maskinen. Om servern inte är ansluten till ett Ethernet-nätverk kan man helt utelämna denna rad.

Att konfigurera `/etc/slip.logout`. När anslutningen avslutas, så vill man se till att den seriella enheten återställs till sitt normala tillstånd så att framtida anslutningar kan logga in ordentligt. Detta uppnås genom med hjälp av filen `/etc/slip.logout`. Dess format är ganska enkelt och används med samma argument som filen `/etc/slip.login`.

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

Allt den gör är ‘ta ner’ gränsnittet vilket kommer att ta bort routen som skapades tidigare. Den användar också kommandot `arp` för att ta bort eventuella `Proxy ARP`s som har skapats, återigen, man behöver inte kommandot `arp` ifall servern inte är ansluten till ett Ethernet-nätverk.

Att konfigurera `/etc/slip.tty`. Om man använder dynamisk IP-adressallokering (har någon rad konfigurerad med `DYNAMIC` i `/etc/slip.hosts`) så måste man konfigurera filen `/etc/slip.tty` för att lista vilka adresser som tilldelas vilken port. Man behöver endast denna fil om man vill att servern dynamiskt skall ge adresser till användare. Formatet är följande:

```bash
# slip.tty   tty -> IP address mappings for dynamic SLIP
```
6. Nätverksspecifik Information.

```markdown
# format: /dev/tty?? xxx.xxx.xxx.xxx
#
/dev/ttyS0  192.168.0.100
/dev/ttyS1  192.168.0.101
#
```

Vad denna fil säger är att användare som ansluter till porten `/dev/ttyS0` och som har sitt adressfält i filen `/etc/slip.hosts` satt till `DYNAMIC` kommer att tilldelas adressen `192.168.0.100`.

På detta sättet behöver man endast allokera en adress per port förutom för de användare som behöver en egen adress. Detta hjälper till att hålla nere antalet adresser man behöver till ett minimum och undviker slösande.

### 6.28.2 Slip Server med `dip`.

Låt mig börja med att säga att lite av informationen nedan kom ifrån manualbladet för `dip`, där det kortfattat beskrivs hur man kör Linux som en SLIP-server. Var också uppmärksam på att det som följer är baserat på paketet `dip337o-uri.tgz` och antagligen inte gäller för andra versioner av `dip`.

`dip` har ett läge när det automatiskt letar upp en rad för den användaren som körde programmet och konfigurerar den seriella linjen som en SLIP-länk baserat på informationen som den hittar i filen `/etc/diphosts`. Detta läge aktiveras genom att köra `dip` som `diplogin`. Detta är därför sättet på vilket man använder `dip` som en SLIP-server, genom att skapa särskilda konton där `diplogin` används som login-shell. Det första som skall göras är att göra en symbolisk länk som följer:

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

Sedan behöver man lägga till rader i filerna `/etc/passwd` och `/etc/diphosts`. Raderna är formaterade som följer:


Ett exempel på en rad för en SLIP-användare i `/etc/passwd` ser ut så här:

```
Sfredm:ij/SMxiTlGVCo:1004:10:Fred:/tmp:/usr/sbin/diplogin
```

```markdown
| | | | | | \__ diplogin as login shell
| | | | | \________ Home directory
| | | | \____________ User Full Name
| | | \____________ User Group ID
| | \____________ User ID
\___________________________ Encrypted User Password
\____________________________ Slip User Login Name
```

När användaren har loggat in så utför programmet `login`, om det hittar och verifierar användaren ok, kommandot `diplogin`. `dip`, när det körs som `diplogin`, vet automatiskt att det skall användas som ett login-shell. När det startas som `diplogin` så är det första programmet gör att använda funktionsanropet `getuid()` för att få användardir för den som körde programmet. Det letar sedan i filen `/etc/diphosts` efter den första raden som passar in på det användardir eller namnet på den `tty`-enhet som anslutningen kom från och konfigurerar sig själv därefter. Genom att välja om en användare skall få en rad i filen `/etc/diphosts` eller om användaren skall ges den generella konfigurationen så kan man bygga sin server på så sätt att man kan ha en blandning av statiskt och dynamiskt tilldelade adresser för användarna. `dip` kommer automatiskt att lägga till en Proxy ARP så detta behöver man inte bry sig om att göra manuellt.
Att konfigurera `/etc/diphosts` Filen `/etc/diphosts` används av dip för att hitta konfigurationer för avlägsna datorer. Dessa avlägsna datorer kan vara användare som ringer in till en Linuxbox eller de kan vara datorer som man ringer till ifrån Linuxboxen.

Det generella formatet på `/etc/diphosts` är som följer:

```
.. 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 ..
```

Fälten är:

1. **login name**: som returnerat av getpwuid(getuid()) eller tty namn.
2. **önanvänd**: kompatibilitet med `/etc/passwd`
3. **Remote Address**: IP-adress för den anslutande datorn, antingen numeriskt eller med namn
4. **Local Address**: IP-adress för denna dator, numeriskt eller med namn
5. **Netmask**: i punkterad decimal notation
6. **Comment field**: skriv vad du vill här.
7. **protocol**: Slip, CSLip osv.
8. **MTU**: decimalt tal

Ett exempel på en rad i `/etc/net/diphosts` för en SLIP-användare kan vara:

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

som specificerar en SLIP-länk med en avlägsen adress 145.71.34.2 och MTU på 296, eller:

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

som specificerar en cSLIP-länk med avlägsen adress 145.71.34.1 och MTU på 1006.

Därför så skall alla användare som man vill tillåta en statisktallokerad adress ha en rad i `/etc/diphosts`. Om man vill att användare som ansluter till en viss port skall få en dynamisk allokerad adress så måste man ha en rad för tty-enheten och ingen rad för användaren. Man skall komma ihåg att konfigurera åtminstone en rad för varje tty-enhet som används för SLIP så att det säkert finns en passande konfiguration oavsett vilket modem de ringer in på.

När användare loggar in får de en normal login och password prompt vid vilken de skall skriva in sitt SLIP-login och password. Om dessa verifieras ok så kommer användaren inte se några särskilda meddelanden och användarna skall endast byta till SLIP-läge på sin sida. Användaren bör ha möjlighet att ansluta ok och bli konfigurerad med relevanta parametrar från filen `diphosts`.

6.28.3 **SLIP server med paketet dSLIP**

Matt Dillon <dillon@apollo.west.oic.com> har skrivit ett programpaket som inte bara ringer in utan också ringer ut med SLIP. Matts paket är en kombination av små program och scripts som hanterar ens
6. Nätverksspecifik Information.

anslutningar. För detta måste man ha `tcsh` eftersom åtminstone ett av scripten kräver detta. Matt tillhandahåller en binär kopia av verktyet `expect` eftersom även det behövs av en del av scripten. Man behöver antagligen ha lite erfarenhet med kommandot `expect` för att lyckas länka in ok, men lät inte detta avskräcka.

Matt har skrivit bra med installationsinstruktioner i README-filen så jag tänker inte upprepa dem.

Man kan hämta `dSLIP` från dess hemsida på:

```
apollo.west.oic.com
/pub/linux/dillon_src/dSLIP203.tgz
```

eller från:

```
sunsite.unc.edu
/pub/Linux/system/Network/serial/dSLIP203.tgz
```

Läs filen README och skapa raderna i `/etc/passwd` och `/etc/group` innan `make install`.

6.29 STRIP (Starmode Radio IP)

Kompileringsalternativ för Kärnan:

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


6.30 Token Ring

Enhetsnamn för Token Ring är `tr0`, `tr1` osv. Token Ring är ett standardiserat LAN-protokoll från IBM som undviker kollisioner med en mekanism som endast tillåter en station på LANet att sända åt gången. En `token` innehas av en station åt gången och stationen som har `token` får lov att sända. När den har sändt sin data så skickar den `token` vidare till nästa station. Token går runt mellan alla aktiva stationer, därav namnet 'Token Ring'.

Kompileringsalternativ för Kärnan:
Konfigurationen av Token Ring är identisk med den som görs med Ethernet förutom att man använder andra enhetsnamn.

6.31 X.25


Jonathon Naylor jsn@cs.nott.ac.uk leder utvecklingen och det har skapats en mailinglista för att diskutera ämnen relaterade till Linux X.25. För att prenumerera skall man skicka ett meddelande till majordomo@vger.rutgers.edu med texten "subscribe linux-x25" i meddelandekroppen.


6.32 WaveLan

Enhetsnamn för WaveLan är ‘eth0’, ‘eth1’, osv.

Kompileringssalternativ för Kärnan:

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


7 Kablage

De som är händiga med en lödpenna kanske vill bygga egna kablar för att koppla ihop Linuxburkar. Följande figurer bör vara till hjälp för detta.

7.1 Seriell NULL-modem kabel.

 Alla NULL-modem kablar är inte likadana. Många NULL-modem kablar gör inte mer än att lura datorn att tro att alla signaler finns och byter sänd och ta emot data pinnarna. Detta fungerar men betyder att man måste använda flödeskontroll i mjukvara (XON/XOFF) vilket inte är så effektivt som flödeskontroll i hårdvara. Följande kabel har bästa möjliga signalering mellan datorer och tillåter användning av flödeskontroll i hårdvara (RTS/CTS).
7. Kablage

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin</th>
<th>Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tx Data</td>
<td>2</td>
<td>---</td>
</tr>
<tr>
<td>Rx Data</td>
<td>3</td>
<td>---</td>
</tr>
<tr>
<td>RTS</td>
<td>4</td>
<td>---</td>
</tr>
<tr>
<td>CTS</td>
<td>5</td>
<td>---</td>
</tr>
<tr>
<td>Ground</td>
<td>7</td>
<td>---</td>
</tr>
<tr>
<td>DTR</td>
<td>20</td>
<td>---</td>
</tr>
<tr>
<td>DSR</td>
<td>6</td>
<td>---</td>
</tr>
<tr>
<td>RLS/DCD</td>
<td>8</td>
<td>---</td>
</tr>
</tbody>
</table>

7.2 Parallellportskabel (PLIP kabel)

Om man tänker använda PLIP-protokollet mellan två maskiner så fungerar följande kabel oavsett vilken typ av parallellportar man har.

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>pin</th>
<th>pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>STROBE</td>
<td>1*</td>
<td></td>
</tr>
<tr>
<td>D0-&gt;ERROR</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>D1-&gt;SLCT</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>D2-&gt;PAPOUT</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>D3-&gt;ACK</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>D4-&gt;BUSY</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>D5</td>
<td>7*</td>
<td></td>
</tr>
<tr>
<td>D6</td>
<td>8*</td>
<td></td>
</tr>
<tr>
<td>D7</td>
<td>9*</td>
<td></td>
</tr>
<tr>
<td>ACK-&gt;D3</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>BUSY-&gt;D4</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>PAPOUT-&gt;D2</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>SLCT-&gt;D1</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>FEED</td>
<td>14*</td>
<td></td>
</tr>
<tr>
<td>ERROR-&gt;D0</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>INIT</td>
<td>16*</td>
<td></td>
</tr>
<tr>
<td>SLCTIN</td>
<td>17*</td>
<td></td>
</tr>
<tr>
<td>GROUND</td>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>

Anmärkningar:

- Anslut inte pinnarna som är markerade med en asterisk '*'.
- Extra jordning är 18,19,20,21,22,23 och 24.
- Om kabeln man använder har en metallskärmning, så skall den anslutas till metallskälet på DB-25 endast på en sida.


Även om man kan använda PLIP-kablar över långa avstånd så skall man undvika det om man kan. Specifikationerna för kabeln tillåter längder på ungefär en meter. Var försiktig med långa PLIP-kablar eftersom källor för stark elektromagnetisk strålning (tex blixtar, elkablar och radiosändare) kan störa och till och med skada kontrollerkortet. Om man verkligen vill ansluta två datorer över långa avstånd så bör man titta på möjligheterna med Ethernet och använda tunn koaxialkabel.
7.3 10base2 (tunn koax) Ethernet-kabel


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


7.4 Tvinnad tvåpar Ethernet-kabel.


8 Terminologi i detta dokument.

Följande är en lista med de viktigaste termerna som används i detta dokument.

**ARP**

Detta är en förkortning av *Address Resolution Protocol* och det är hur en maskin på ett nätverk associerar en IP-adress med en hårdvaruadress.

**ATM**

Detta är en förkortning av *Asynchronous Transfer Mode*. Ett ATM-nätverk packar data i block med en standardstorlek vilka det sedan kan skicka effektivt från punkt till punkt. ATM är ett kopplingsorienterat nätverk med virtuella kretsar.

**klient**


**datagram**

Ett datagram är ett diskret paket med data och huvuden, som innehåller adresser, som är en transmissionsenhet över ett IP-nätverk. Detta kan också kallas för ‘paket’ i vissa sammanhang.
9. Linux för en ISP?

Om man är intresserad av att använda Linux för ISP-syften så rekommenderar jag att titta på Linux ISP homepage <http://www.anime.net/linuxisp/> för en bra lista med pekare till information som kan vara användbar.

**DLCI**
Detta är en förkortning av Data Link Connection Identifier och används för att identifiera en unik virtuell punkt till punkt anslutning via ett Frame Relay nätverk. DLCIs delas normalt ut av den som tillhandahåller Frame Relay nätverket.

**Frame Relay**

**hårdvaruadress**

**ISDN**
Detta är en förkortning av Integrated Services Digital Network. ISDN tillhandahåller ett standardiserat sätt på vilket telebolag kan erbjuda antingen data- eller rösttjänster till en kund. Tekniskt sett är ISDN ett datanätverk med virtuella kretsar.

**ISP**
Detta är en förkortning av Internet Service Provider (internetleverantör). Detta är de organisationer eller företag som erbjuder nätverksanslutningar till Internet.

**IP-adress**
Detta är ett nummer som unikt identifierar en TCP/IP dator på ett nätverk. Adressen är 4 bytes lång och representeras ofta i vad som heter ”punkterad decimal notation”, där varje byte skrivs för sig och de olika bytearna är separerade med en ’.’.

**MTU**
Detta är en förkortning av Maximum Transmission Unit. MTU är en parameter som bestämmer den största datagramstorleken som kan överföras av ett IP-gränsnitt utan att det behöver delas upp i flera mindre delar. MTU bör vara större än det största datagram som man vill skicka ofragmenterat. Notera att detta bara hindrar lokal fragmentering, om datagrammet passerar en länk med en mindre MTU så kan det fragmenteras där. Typiska värden för MTU är 1500 bytes för Ethernet och 576 bytes för SLIP.

**route**
Routen (eller rutten) är den väg som datagrammen färdas genom nätverket för att nå sin destination.

**server**

#### 9 Linux för en ISP?

Om man är intresserad av att använda Linux för ISP-syften så rekommenderar jag att titta på Linux ISP homepage <http://www.anime.net/linuxisp/> för en bra lista med pekare till information som kan vara användbar.
10 Tillkännagivanden


11 Copyright.

NET-3-HOWTO, information om hur man installerar och konfigurerar nätverksstöd för Linux. Copyright (c) 1997 Terry Dawson.

Detta program är fri mjukvara; du kan distribuera det och/eller modifiera det under reglerna som finns i GNU General Public License så som de publiceras av Free Software Foundation; antingen version 2 av licensen, eller (ditt eget val) någon senare version.

Detta program distribueras med förhoppningen att det skall vara användbart, men UTAN NÄGON GARANTI; GNU General Public License för fler detaljer.

Du skall ha fått en kopia av GNU General Public License tillsammans med detta program; om inte, skriv till:

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