Configurazione di un collegamento PPP su un server Linux Red Hat
Configurazione della rete in Red Hat Linux

Il *software* di configurazione presente nella distribuzione *Red Hat Linux* consente una gestione semplificata delle interfacce di rete e delle funzionalità di *networking*. Le informazioni relative vengono tenute nei *file* di configurazione `/etc/sysconfig/network` e `/etc/sysconfig/network-scripts` e vengono utilizzate dagli *script* di *boot* che si occupano di attivare la rete (`/etc/rc.d/init.d/network`).

Nel caso si modifichino a mano tali *file*, per rendere attive le modifiche è necessario far ripartire i servizi di rete:

```
# /etc/rc.d/init.d/network restart
Shutting down interface eth0 [ OK ]
Disabling IPv4 packet forwarding [ OK ]
Disabling IPv4 automatic defragmentation [ OK ]
Enabling IPv4 packet forwarding [ OK ]
Bringing up interface lo [ OK ]
Bringing up interface eth0 [ OK ]
```

Volendo fermare o far ripartire singolarmente una determinata interfaccia di rete, si possono utilizzare i comandi `ifup` e `ifdown`:

```
# ifdown eth0
# ifup eth0
```

Il *file* `/etc/sysconfig/network` contiene le impostazioni generali del *software* di rete, come il nome del sistema, comprensivo del dominio di appartenenza (FQDN, *Full Qualified Domain Name*) e l'indirizzo del *gateway* di *default*.

```
NETWORKING=yes
FORWARD_IPV4=true
HOSTNAME=mionome.miodominio.com
DOMAINNAME=miodominio.com
GATEWAY=193.43.98.254
```

La linea *`FORWARD_IPV4=true`* indica che la macchina deve eseguire il *forwarding* dei pacchetti IP fra le diverse interfacce, operazione che non è abilitata per *default*.

Nella *directory* `/etc/sysconfig/network-scripts/` sono presenti i *file* contenenti i dati di configurazione relativi alle singole interfacce presenti nel sistema. Ad esempio `ifcfg-eth0` contiene i dati di configurazione della prima scheda *ethernet* presente nel sistema:

```
DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.1.255
IPADDR=192.168.1.23
NETMASK=255.255.255.0
NETWORK=192.168.1.0
ONBOOT=yes
```

L'esempio presentato si riferisce ad una configurazione statica (*BOOTPROTO=static*) della interfaccia `eth0`, in cui vengono specificati esplicitamente l'indirizzo IP, quello della rete, la *netmask* e l'indirizzo di *broadcast*. L'ultima riga del *file* indica che l'interfaccia deve essere attivata al *boot* del sistema.

È possibile in alternativa configurare l'interfaccia per acquisire il proprio indirizzo mediante il protocollo BOOTP (*BOOTPROTO=bootp*) oppure da un *server* DHCP (*BOOTPROTO=dhcp*).

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
```

Interfacce Point-to-Point

In *Linux* sono disponibili diversi protocolli di connessione punto-a-punto su collegamento seriale, di cui i due più usati sono PPP e SLIP. Analizzeremo solamente il primo in quanto più diffuso.

Trattandosi di un protocollo *peer-to-peer*, è possibile utilizzare PPP, oltre che per accedere ad Internet attraverso un Internet *Provider*, anche per connettere fra loro due reti oppure per offrire l'accesso ad altre macchine.

Generalmente il collegamento avviene via modem oppure mediante un cavo seriale null-modem, ma è anche possibile incapsulare una connessione PPP all'interno di pacchetti IP in modo da realizzare una VPN. Un caso particolare è dato dal protocollo PPPoE, utilizzato dai modem ADSL, il quale permette il *tunnelling* di PPP all'interno di *frame Ethernet*.

La configurazione di una interfaccia PPP è più complessa rispetto a quella di una interfaccia *Ethernet*, in quanto oltre alla connessione devono essere gestiti anche gli aspetti riguardanti la negoziazione degli indirizzi, l'autenticazione, la compressione, ...

PPP è composto da tre strati: un metodo per incapsulare pacchetti di dati su una linea seriale, un protocollo che gestisce il link (LCP, *Link Control Protocol*) ed uno strato che si occupa di gestire i differenti protocolli di rete che possono funzionare sopra PPP (nel caso del protocollo IP viene usato IPCP, IP *Control Protocol*).

In *Linux* l'incapsulamento dei dati avviene internamente al *kernel*, mentre il controllo della connessione viene gestito mediante il demone esterno `pppd`.

La presenza del supporto per il PPP nel *kernel* è confermata dal seguente messaggio al *boot*:

```
PPP: version 2.3.7 (demand dialling)
PPP line discipline registered.
```

La configurazione di `pppd` avviene specificando i parametri necessari nella linea di comando, oppure inserendoli nel *file* `/etc/ppp/options`. Nel caso di collegamento telefonico è possibile utilizzare il comando *chat* per inizializzare il modem, comporre il numero di telefono ed eseguire il *login* sul terminal *server* remoto. Esso utilizza un linguaggio di tipo *send-expect* per descrivere le stringhe da spedire in risposta a quelle ricevute dal modem.

Generalmente il collegamento PPP viene gestito mediante degli *script* in *shell* (ppp-*on*, ppp-*off*) contenenti i comandi necessari, tuttavia è possibile semplificare le operazioni utilizzando delle interfacce grafiche, come KPPP (compresa in KDE), WvDial, RP3 (*Red Hat Linux*). La configurazione e la gestione di PPP sono possibili anche mediante lo strumento standard *Linuxconf*.

Autenticazione

L'autenticazione in PPP può avvenire in diversi modi:

- eseguendo il *login* sull'*access server* e successivamente richiedendo una sessione in modalità

PPP: in questo caso le relative procedure possono essere automatizzate mediante l'utilizzo del comando *chat*.

- mediante i metodi standard PAP/CHAP/MSCHAP: in questo caso è sufficiente specificare nella linea di comando di PPP oppure in `/etc/ppp/options` il metodo prescelto ed il nome dell'*account* ed inserire la *password* (*secret*) in uno dei *file* `/etc/ppp/pap-secrets` o `/etc/ppp/chap-secrets`. Essi possono contenere una o più linee nel seguente formato: *client server secret IP addresses*. La *password* da utilizzare viene selezionata dal `pppd` utilizzando come chiavi di ricerca in *chap-secrets* il nome specificato nella opzione *user*, l'indirizzo che viene assegnato alla macchina dal *router* remoto e l'indirizzo all'altro capo della connessione. Nel caso alcuni di questi valori vengano gestiti in modo dinamico, ad e esempio l'indirizzo IP, è possibile sostituire la *wildcard* * al posto dei dati che non si conoscono: ** * secret **

Esempio di script PPP

Il seguente *script* mostra un esempio di come possa essere instaurato un collegamento PPP in modalità client. Per concludere la connessione è sufficiente terminare il processo `pppd`.

```
#!/bin/sh

TELEPHONE=0862123456 # Numero di telefono del provider
ACCOUNT=test # Username fornito dal provider
PASSWORD=test # Password fornita dal provider
LOCAL_IP=0.0.0.0
REMOTE_IP=0.0.0.0
export TELEPHONE ACCOUNT PASSWORD

exec /usr/sbin/pppd debug lock modem crtscts /dev/ttyS0 38400 \
asyncmap 20A0000 escape FF kdebug 0 $LOCAL_IP:$REMOTE_IP \
noipdefault netmask 255.255.255.0 defaultroute \
connect /etc/ppp/ppp-on-dialer
```

Le opzioni che nell'esempio sono specificate direttamente nella linea di comando di `pppd`, possono essere in alternativa inserite, una per riga, nel *file* `/etc/ppp/options`.

Nel caso si effettui una connessione con IP fissi, essi dovranno essere specificati nelle variabili `$LOCAL_IP` e `$REMOTE_IP`. Se si utilizza un indirizzamento dinamico, si possono invece lasciare gli indirizzi fittizi 0.0.0.0 e inserire fra le opzioni *noipdefault*.

Si noti che alcune delle variabili definite nello *script* vengono esportate in modo da renderle disponibili anche ai programmi lanciati dal `pppd`:

L'inizializzazione del modem e la composizione del numero di telefono del *provider* non vengono effettuati direttamente da `pppd`, ma vengono demandate allo *script* definito dalla opzione *connect*:

```
#!/bin/sh
exec chat -v "" AT OK ATZ OK ATM1L3X3 OK ATDT$TELEPHONE CONNECT
```

L'esempio precedente mostra una semplice sequenza *send-expect* necessaria a resettare il modem, inviargli una semplice stringa di inizializzazione e fargli comporre un numero di telefono.

Uno *script* come quello appena visto non è molto robusto, in quanto non è in grado di riconoscere eventuali errori, se non quello dovuto al *timeout*. Uno esempio più completo è il seguente, in cui vengono definiti alcuni dei messaggi di errore che il modem può ritornare.

```
exec chat -v TIMEOUT 3 ABORT '\nBUSY\r' ABORT '\nNO ANSWER\r'
ABORT '\nRINGING\r\n\r\nRINGING\r' '' \rAT 'OK-+++\c-OK' ATH0
TIMEOUT 30 OK ATDT$TELEPHONE CONNECT
```

È possibile utilizzare *chat*, oltre che per comporre il numero, anche per autenticare l'utente nel caso l'*access server* remoto preveda una procedura di *login*. Un esempio è il seguente:

```
chat -v ATZ OK ATD$TELEPHONE CONNECT \r ername: test ssword: test > ppp
```

Si noti che il carattere > è stato quotato per evitare che venga interpretato dalla *shell* come un operatore di ridirezione. Analogamente per il carattere \r, che serve per inviare un a capo.

Quelle appena viste solo solamente alcune delle opzioni usabili nel comando *chat*. Per ulteriori informazioni si può fare riferimento al manuale in linea (*man chat*).

Debugging

Inserendo fra le opzioni di `pppd` *debug* e k*debug* n è possibile tenere traccia nel log di sistema delle operazioni compiute da `pppd`. Essa può essere utilizzata per risolvere eventuali problemi di funzionamento.

```
Jan 24 15:56:28 freddy kernel: registered device ppp0
Jan 24 15:56:28 freddy pppd[6204]: pppd 2.3.10 started by root, uid 0
Jan 24 15:56:28 freddy pppd[6204]: Using interface ppp0
Jan 24 15:56:28 freddy pppd[6204]: Connect: ppp0 <--> /dev/modem
Jan 24 15:56:30 freddy kernel: PPP BSD Compression module registered
Jan 24 15:56:31 freddy kernel: PPP Deflate Compression module registered
Jan 24 15:56:31 freddy pppd[6204]: Unsupported protocol (0x803f) received
Jan 24 15:56:31 freddy pppd[6204]: local IP address 193.43.99.222
Jan 24 15:56:31 freddy pppd[6204]: remote IP address 193.43.99.1
...
Jan 24 16:27:49 freddy pppd[6204]: Terminating on signal 2.
Jan 24 16:27:49 freddy pppd[6204]: Connection terminated.
Jan 24 16:27:49 freddy pppd[6204]: Connect time 31.4 minutes.
Jan 24 16:27:49 freddy pppd[6204]: Sent 3370 bytes, received 4663 bytes.
Jan 24 16:27:49 freddy pppd[6204]: Exit.
```

Gli script /etc/ppp/ip-up e /etc/ppp/ip-down

Mediante lo *script* `/etc/ppp/ip-up` si possono eseguire automaticamente delle operazioni quando viene instaurato un collegamento PPP. Analogamente, *`ip-down`* permette di compiere delle operazioni quando esso termina.

In questo modo è per esempio possibile tenere un log della connessione, oppure prelevare la posta elettronica dal *provider* o aggiungere delle tabelle di *routing*.

Se fra le opzioni di `pppd` si inserisce `usepeerdns`, è possibile ottenere dall'*access server* remoto gli indirizzi dei *nameserver* da utilizzare: essi vengono passati allo *script* `/etc/ppp/ip-up`, come variabili d'ambiente DNS1 e DNS2, da utilizzare per aggiornare in modo automatico il *file* `/etc/resolv.conf`.

Un esempio di *script* *`ip-up`* è il seguente, che compie operazioni differenti a seconda che ci si connetta o meno alla rete aziendale (la variabile $5 contiene l'indirizzo assegnato alla connessione).

```
#!/bin/sh
# tiene un log della connessione
```

```
echo `/bin/date` INIZIO >>/var/log/isdn.log

case "$5" in
192.168.*) # se collegato alla LAN aziendale, aggiunge una route
/sbin/route add -net 192.168.0.0 dev ppp0
# e monta un disco dal server NT
/bin/mount -t smbfs //ntserver/share /share
;;

*) # se collegato all'ISP, scarica la posta
/usr/local/bin/SCAMBIA_POSTA &
# e aggiorna la configurazione dei DNS
if [ ! "$DNS" = "" ]
then
echo DNS1 >/etc/resolv.conf
echo DNS2 >>/etc/resolv.conf
fi
;;
esac
```

Collegamento on demand

Mediante il programma `diald` è possibile automatizzare il collegamento PPP in presenza di traffico ed abbattere la connessione allo scadere di un determinato *timeout*. Si tratta di un programma versatile, che consente di filtrare i pacchetti che possono instaurare il collegamento e di assegnare un *timeout* diverso a seconda della tipologia di traffico (ad esempio aumentandolo nel caso siano attive delle sessioni di lavoro il cui funzionamento possa essere pregiudicato da una eventuale interruzione della linea).

Linux come Access Server

Mediante opportune configurazioni è possibile utilizzare `pppd` anche per permettere collegamenti entranti da parte di altre macchine. Per far ciò è necessario configurare il programma `getty`, che gestisce i modem in ingresso, in modo che esso riconosca il tipo di connessione e lanci il PPP invece della normale richiesta di *login* in modo testo. Esistono diversi modi per fare ciò, che sono spiegati nel PPP *HOWTO*.

Volendo utilizzare *Linux* per sostituire un *access server* commerciale, conviene indirizzarsi verso il pacchetto *Portslave*, il quale permette l'autenticazione degli utenti utilizzando il metodo standard PAM, oppure appoggiandosi ad un *server* TACACS o RADIUS.

ISDN

Mentre i *Terminal Adapter* ISDN attivi possono essere in generale utilizzati mediante il demone `pppd` standard, quelli passivi necessitano di un opportuno supporto da parte del *kernel*, fornito dal sottosistema ISDN4*Linux*.

Se si utilizza il protocollo sincrono X.75, è necessario ricorrere al demone `ipppd` (recentemente i due programmi `pppd` e `ipppd` sono stati riuniti in un unico *software* in grado di gestire entrambi i protocolli). Salvo lievi differenze, la configurazione di `ipppd` è simile a quanto abbiamo visto per il `pppd`.

ADSL (PPPoE)

Il collegamento ADSL è possibile sia mediante il protocollo PPPoA (PPP-*over*-ATM), che mediante PPPoE (PPP-*over*-*Ethernet*). Nel primo caso è necessario compilare il *kernel* (versione 2.4) con il

supporto per ATM, applicando le apposite *patch* disponibili su:

http://linux-atm.sourceforge.net/

Le istruzioni per l'installazione di PPPoA si trovano nel sito:

http://www.sfgoth.com/~mitch/linux/atm/pppoatm/

Per utilizzare PPPoE è invece possibile utilizzare sia una soluzione *kernel-based*, sia una soluzione più semplice, basata su un apposito demone, che realizza il *tunnelling* di una connessione PPP all'interno di *frame ethernet* sfruttando il normale `pppd` (PPPoE della *Roaring Penguin*, disponibile per *Linux* 2.2). Il supporto necessario a livello di *kernel* è già incluso in *Linux* 2.4 (dettagli).

Per maggiori informazioni sulla configurazione di ADSL si può fare riferimento al DSL-*HOWTO*.

Comandi basilari per il test della rete

Per verificare che le tabelle di *routing* funzionino in modo corretto si può utilizzare il comando ping, che testa la raggiungibilità di un dato indirizzo spedendo dei pacchetti di tipo ICMP ed aspettando una risposta dalla macchina remota. Esso permette fra le altre cose di conoscere il tempo impiegato dai pacchetti per compiere il tragitto di andata e ritorno (RTT, *Round Trip Time*). Per terminare si premano i tasti CTRL-C:

```
# ping www.pluto.linux.it
PING www.pluto.linux.it (147.162.126.3) from 62.98.87.226 :
56(84) bytes of data.
64 bytes from 147.162.126.3: icmp_seq=0 ttl=242 time=105.4 ms
64 bytes from 147.162.126.3: icmp_seq=1 ttl=242 time=141.1 ms
64 bytes from 147.162.126.3: icmp_seq=2 ttl=242 time=100.5 ms

--- www.pluto.linux.it ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 100.5/115.6/141.1 ms
```

Se si desidera conoscere il percorso esatto che compiono i pacchetti per raggiungere una data destinazione, si può utilizzare invece il programma *traceroute* (*tracert* nella versione *Windows*). Esso spedisce un tipo particolare di pacchetto ICMP che fa in modo che tutti i *router* per cui il esso transita spediscano indietro una risposta. In questo modo è possibile generare una traccia del percorso del pacchetto, che permette di capire se e dove ci sono problemi di connettività o quale *router* è la causa di eventuali rallentamenti nella connessione.

```
# traceroute www.interbusiness.it
traceroute to www.cs.interbusiness.it (151.99.250.4), 30 hops max, 40 byte
packets
1 217.57.6.193 (217.57.6.193) 1.132 ms 1.806 ms 1.086 ms
2 62.86.50.61 (62.86.50.61) 7.795 ms 14.604 ms 5.071 ms
3 r-pd48-fa2.interbusiness.it (212.131.52.79) 7.945 ms 4.991 ms 4.998 ms
4 151.99.101.97 (151.99.101.97) 7.851 ms 14.313 ms 11.904 ms
5 r-mi213-fa4.interbusiness.it (151.99.75.218) 11.279 ms 12.783 ms 8.197 ms
6 r-rm99-ts21.interbusiness.it (151.99.98.109) 23.966 ms 17.211 ms 15.916 ms
7 r-rm179-fa4.interbusiness.it (151.99.29.213) 19.518 ms 17.472 ms 16.214 ms
8 r-rm80-fa2.interbusiness.it (151.99.29.8) 23.093 ms 28.061 ms 32.447 ms
9 212.131.81.38 (212.131.81.38) 40.216 ms 31.431 ms 62.16 ms
10 www.cs.interbusiness.it (151.99.250.4) 42.342 ms 34.359 ms 64.41 ms
```

Poiché in Internet il *routing* viene eseguito in modo dinamico, non è detto che ogni pacchetto segua necessariamente lo stesso percorso del precedente o di quello successivo (ad esempio perché una

tratta diventa congestionata o interrotta). Pertanto *traceroute* di solito spedisce 3 pacchetti ICMP e calcola una media dei tempi impiegati.