UNIVERSITA' DEGLI STUDI DI PAVIA
Sede distaccata di Mantova
Laboratorio di RETI DI CALCOLATORI
(Prof. Giuseppe Federico Rossi)

Manuale di configurazione del protocollo IPv6 nel sistema operativo Linux

ad uso degli Studenti del Corso di RETI TELEMATICHE
CdL II° livello in Ingegneria Informatica

A cura dell'Ing. Luca Anselmi ed Emanuele Goldoni

Versione 0.3

Mantova, li 21/04/2005

1 - PREMESSA

Il presente manuale è stato pensato quale ausilio per gli Studenti del CdL in Ingegneria Informatica, impegnati nello svolgimento di esercitazioni pratiche presso il *Laboratorio Reti* attivo presso la sede distaccata di Mantova dell'Università degli Studi di Pavia.
Più precisamente verrà illustrata la configurazione di rete nel sistema operativo Linux, con particolare riferimento al protocollo IPv6, ai protocolli EthernetV2 (e IEEE 802.3) e al più utilizzato PPP (*Point-to-Point Protocol*).

Da sempre il sistema operativo Linux, e anche i suoi predecessori UNIX e Minix, offrono un'infinita possibilità di configurazioni di rete.
E' da considerare che comandi, procedure e schemi simili a quelli di seguito descritti si trovano anche in altri sistemi operativi secondo svariate implementazioni, ma queste varianti non saranno prese in considerazione nella presente trattazione.
Tutto ciò che sarà visto è una implementazione degli standard *de facto* definiti a livello internazionale, gli RFC (*Request For Comment*), disponibili per consultazione all'URL *http://ww.ietf.org*. Per ogni riferimento dettagliato al funzionamento dei protocolli di rete, si rimanda pertanto a tali documenti.

Chiunque può provare ciò che viene descritto, avendo a disposizione una distribuzione Linux con kernel almeno 2.2.x.x (o superiore). In particolare con il kernel Linux 2.2.x.x (quale ad esempio Red Hat 6.2, 5.2, Mandrake 7.2, Debian) anche i vecchi 486 funzioneranno egregiamente!

I comandi e le procedure descritte sono stati testati con successo sulla distribuzione Fedora Core 3.

Sistemi operativi ancora più adatti per l'utilizzo in rete sono Solaris, NetBSD/OpenBSD e soprattutto FreeBSD; tuttavia questi discendono da un ramo di sviluppo Unix diverso da quello di Linux e pertanto i comandi e le funzionalità descritti in questo manuale potranno presentare alcune leggere differenze sui sistemi sopra indicati.

Nel sistema operativo Linux, la configurazione di base della rete si compone dei seguenti punti:

- Creazione delle connessioni hardware;
- Creazione dei dispositivi (interfacce) di rete;
- Impostazione dei parametri di rete;
- Configurazione dei principali parametri di routing;
- Verifica delle impostazioni inserite;
- Altre categorie non trattate in questo manuale.

Praticamente quasi tutto ciò che troverete in questo manuale presuppone l'utilizzo di un terminale testuale, la *shell* di Linux, ma è allo stesso modo realizzabile con la rispettiva interfaccia grafica (X-Windows, GNOME, KDE, ecc..). Questo non per rendere le cose più difficili (a volte non lo sono affatto), ma per dare una visione più standardizzata possibile della configurazione di rete.
Infatti l'interfaccia grafica non è uguale nelle diverse versioni di Linux e purtroppo, alle volte, non è pienamente collaudata e può quindi presentare dei malfunzionamenti.

- SICUREZZA -

Sono state istituite alcune varianti per mantenere la sicurezza nella gestione dei sistemi. In pratica, per l'esecuzione dei comandi specificati, è spesso necessario utilizzare l'utenza di amministratore (`root`); se ciò fosse concesso, un utente incauto o malintenzionato potrebbe danneggiare il sistema.
Per questo non è consentito utilizzare l'utenza root, ma sarà consentito utilizzare l'utenza `labreti#` (esempio `labreti1`, `labreti2`, ecc..) con password uguale al nome utente.

Per gestire i comandi di rete, sono ammessi solamente i comandi descritti con esecuzione tramite l'utility `sudo` secondo la sintassi:

`$ sudo <nome comando> <opzioni comando>`

In pratica ogni comando che richiede l'utenza `root` per poter funzionare deve essere preceduto da `sudo`.

Le esercitazioni sono impostate in modo tale che le configurazioni modificate non vengano memorizzate dal sistema; quindi, se il sistema viene riavviato, le impostazioni saranno perse.

2 - CONNESSIONI HARDWARE

I 5 PC che vi troverete di fronte sono dotati di 3 schede Ethernet 10/100 Mbps ognuno e di 2 interfacce seriali. La prima interfaccia Ethernet è chiamata `eth0`, la seconda `eth1`, la terza `eth2`. L'interfaccia seriale è chiamata `sl0`, le interfacce Token Ring (non presenti) sono chiamate `tr#` mentre con `ppp#` si indicano le interfacce Point-to-Point.

- GLOSSARIO -

È necessario avere molta familiarità con i termini **host**, **interfaccia e scheda** in quanto verranno frequentemente utilizzati in tutto il manuale.

Host: rappresenta il singolo PC o router interconnesso nella rete. Talvolta si fa confusione perchè si chiama l'host indicando il suo indirizzo IP. Occorre fare molta attenzione perchè un host può avere molti indirizzi IP (ed è questo il nostro caso). Nella rete Internet, di solito un host ha un solo indirizzo IP pubblico, quindi dall'esterno può essere indicato con l'indirizzo IP (o anche con il nome completo di dominio) ma i router, ad esempio, hanno molti indirizzi IP pubblici.

Interfaccia: rappresenta ciò che il sistema identifica come dispositivo di connessione di rete (scheda di rete, porta seriale e persino tunnel "virtuali"). Un host può avere più interfacce (i nostri hanno 3 schede di rete ciascuno più le due seriali) ed ogni interfaccia può avere più indirizzi IP associati.

Scheda: è il componente "hardware" che interconnette realmente alla rete, detto anche NIC (Network Interface Card). Per maggiori informazioni si consiglia di far riferimento alla *Guida all'Hardware di Rete* pubblicata sul sito del Laboratorio Reti.

Connessioni hardware per le schede Ethernet.

Le connessioni possono essere effettuate con i cavi dotati di PLUG RJ45 (simili a quelli telefonici) a 8 fili che possono essere "dritti" o "incrociati" (*cross-cable*). Ogni scheda di rete ha solitamente una presa conforme allo standard RJ45; se si usano i cavi dritti per interconnettere le schede è necessario utilizzare anche un HUB (concentratore) mentre, se si usano cavi incrociati, è possibile interconnettere con un cavo direttamente due schede di rete tra loro.
Nelle esercitazioni si useranno spesso gli HUB anche se, non dovendo effettuare sniffing del traffico di rete, andrebbero benissimo anche gli SWITCH .
Un HUB (o una cascata di HUB) può interconnettere anche decine di host e gli host connessi su un HUB si intendono come appartenenti ad una stessa sottorete. In teoria è possibile avere più sottoreti su uno stesso HUB, ma questa possibilità non sarà presa in considerazione (nella realtà non si interconnettono più sottoreti su uno stesso HUB).

3 - CREAZIONE DEI DISPOSITIVI DI RETE

In Linux ogni dispositivo (*device*) è visto come un file e i files che rappresentano tutti i dispositivi si trovano nella directory `/dev`.
Linux riconosce le schede di rete generalmente durante l'installazione del sistema; questo significa che solitamente non è necessario alcun tipo di operazione per creare i dispositivi di rete Ethernet.
I nomi utilizzati per i dispositivi Ethernet sono, come già detto, `eth<numero dispositivo>`.
Tra le schede di rete più diffuse molti ricorderanno quelle basate su chipset NE2000-compatibili, le schede 3Com Etherlink 3C509x o i tantissimi dispositivi Realtek RTL8129; la scheda di rete attualmente più diffusa è basata sul chip Realtek RTL8139 ed è la stessa utlizzata nei nostri PC.
Nel caso in cui il riconoscimento automatico di questo dispositivo fallisca può essere necessario ricorrere al caricamento manuale dell'apposito modulo del kernel; per fare questo è necessario inserire nel file `/etc/modules.conf` (kernel 2.4) o in `/etc/modprobe.conf` (kernel 2.6) la seguente riga per ogni scheda di rete:

```
alias     eth# 8139too
```

Utile può essere il comando `ifstatus`, che permette di verificare se il cavo di rete è connesso alla scheda (per le schede che supportano questa funzione); altri tool da provare sono `ifport`, `mii-diag` e `mii-tools`.
In generale, se non si riesce a far funzionare una scheda di rete o altro dispositivo installato, possono essere di aiuto i comandi `lspci` o `insmod`; tuttavia in diversi casi, specialmente se si tratta di hardware molto recente, l'unica soluzione è rappresentata dal passaggio ad una più recente versione del kernel.

Creazione delle interfacce ppp su porte seriali

Un cavo Ethernet incrociato non è l'unico modo possibile per realizzare una connessione punto-a-punto tra due PC; nel nostro caso infatti utilizzeremo un cavo seriale con connettore a 9 poli (DB9).
Assodato che i dati passeranno su una "rete" costituita da cavi seriali, si tratta ora di trovare un DLC in grado di supportare IPv6. Purtroppo il vecchio protocollo di comunicazione dati per le interfacce seriali SLIP non supporta ne mai supporterà IPv6 e quindi l'unica possibilità è rappresentata dal protocollo PPP (RFC2472 - IP Version 6 over PPP), lo stesso usato ad esempio per una normale connessione ad Internet tramite modem.
Fortunatamente il demone `pppd` permette di stabilire una connessione PPP semplicemente utilizzando il comando:

```
$ sudo pppd /dev/ttyS0 lock crtscts
```

`/dev/ttyS0` è il nome del dispositivo utilizzato da Linux per identificare la prima porta seriale (COM1 sotto DOS/Windos); sono validi anche `ttyS1` (COM2) ecc.
`lock` garantisce a `pppd` un accesso esclusivo alla seriale
`crtscts` abilita il controllo di flusso harware RTS/CTS sulla porta seriale

Questo comando creerà un nuovo dispositivo di rete `ppp0` attivabile e configurabile con il comando `ifconfig`, esattamente come accade per una comune scheda di rete.

Tra le moltissime opzioni disponibili di `pppd` (per visualizzarle tutte potete, come di consueto, consultare la pagina di manuale) segnaliamo:
`mtu n`: imposta il valore di *Maximum Transmit Unit* (MTU) a `n` byte; per IPv6 deve essere almeno di 1280
`ipv6 <local_interface_identifier>`: permette di impostare manualmente l'*Interface_Identifier* di 64 bit univoco sul link (es. `::dead:beef`) utilizzato poi per l'autoconfigurazione dell'indirizzo unicast link-local (con prefix cioè `fe80::/64`)
`ipv6cp-use-ipaddr`: per la generazione dell'indirizzo link-local IPv6 viene utilizzato l'indirizzo IPv4 assegnato all'interfaccia
NB: Se nessuna delle due opzioni precedenti viene specificata, come ad esempio nel comando da noi utilizzato durante l'esercitazione, l'*Interface_Identifier* viene generato casualmente.

4 - IMPOSTAZIONE DEI PARAMETRI DI RETE

Ogni interfaccia di rete ha alcuni parametri che è necessario configurare per far sì che possa funzionare, alcuni dei quali alcuni prevedono una configurazione automatica:

- Attivazione dello stack IPv6
- Indirizzo IPv6;
- Maschera di sottorete (Subnet Mask);
- MTU (Maximum Tranfer Unit);
- Indirizzo di broadcast;
- Eventuali Indirizzi alias;
- Attivazione e disattivazione dell'interfaccia.

Attivazione dello stack IPv6

Lo stack IPv6 all'interno di Linux viene normalmente fornito, insieme a moltissimi altri componenti del kernel, come modulo da caricare all'occorrenza.
Per visualizzare i moduli caricati è possibile utilizzare il comando:

```
$ lsmod
```

E' bene verificare che, all'interno della lista presentata, compaia anche il modulo `ipv6`.

Nel caso in cui un modulo necessario non sia ancora stato attivato, è possibile caricarlo in maniera esplicita utilizzando il comando `modprobe` seguito dal nome del modulo desiderato; per attivare ad esempio IPv6 è necessario digitare:

```
$ modprobe ipv6
```

Conviene poi verificare l'avvenuto caricamento sempre utilizzando `lsmod`.

Se si volesse infine rendere automatico il caricamento di questo modulo all'avvio del sistema, sarà sufficiente inserire nel file `/etc/modules.conf` (o `/etc/modprobe.conf`) la riga di configurazione:

```
alias     net-pf-10 ipv6
```

Configurazione interfacce Ethernet

Per le interfacce Ethernet (`eth#`) è necessario impostare indirizzo IPv6 di tipo global ed un prefisso e, in casi particolari, anche MTU o eventuali indirizzi di alias.
Ogni interfaccia Ethernet deve essere configurata con due tipi di indirizzi IPv6: uno di tipo link local, locale al link, ed un secondo di tipo global che deve essere univoco a livello mondiale.
Gli indirizzi link-local nel caso delle interfacce Ethernet sono assegnati dinamicamente per via algoritmica partendo dal MAC Address della scheda, mentre gli indirizzi di tipo Global devono essere assegnati in maniera esplicita.

Per configurare parametri di rete si usa il comando `ifconfig`.

Se si esegue il comando senza parametri, esso riporta l'elenco delle interfacce attive in questo momento. Di seguito un possibile esempio di output del comando ifconfig:

```
eth0      Link encap:Ethernet  HWaddr 00:50:FC:CD:20:2A
          inet6 addr: 3ffe:1001:1c0:2::/64 Scope:Global
          inet6 addr: fe80::250:fcff:fecd:202a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:566 (566.0 b)
          Interrupt:9 Base address:0xc000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1011 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1011 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1485416 (1.4 MiB)  TX bytes:1485416 (1.4 MiB)
```

Da questo esempio è possibile verificare che sono attive due interfacce: la `eth0` e quella di *loopback* (indicata con `lo`).
Per ogni interfaccia inoltre vengono riportate informazioni significative quali:

- Il MAC Address ddella scheda
- Gli eventuali indirizzi IPv4, e IPv6 (global e link-local)
- Il valore di MTU
- Informazioni varie riguardanti i pacchetti ricevuti o trasmessi.

Attivazione e disattivazione dell'interfaccia

Ogni interfaccia è attivabile in maniera esplicita tramite il comando:

`$ ifconfig <nome interfaccia> up`

Si consiglia di attivare un'interfaccia prima di assegnarle eventuali indirizzi IPv6 e si ricorda che i computer utilizzati per le esercitazioni all'avvio vedono tutte le interfacce disattivate.

Allo stesso modo è possibile disattivare in maniera esplicita un'interfaccia specifica digitando:

`$ ifconfig <nome interfaccia> down`

Se fosse necessario visualizzare tutte le interfacce disponibili, indipendentemente dal fatto che siano attive o meno, lo si può fare con:

`$ ifconfig –a`

Assegnazione di un indirizzo IPv6

Per impostare l'indirizzo IPv6 di una scheda di rete, basta scrivere:

`$ ifconfig eth# inet6 add <indirizzo IPv6>/<prefisso>`

dove **#** rappresenta il numero della scheda di rete.
Nel caso fosse necessario assegnare più indirizzi alla stessa interfaccia, basterà ripetere lo stesso comando ovviamente con un indirizzo diverso.

Allo stesso modo, se si volesse togliere un indirizzo IPv6 precedentemente assegnato:

`$ ifconfig eth# inet6 del <indirizzo IPv6>/<prefisso>`

laddove l'indirizzo e il relativo prefisso devo essere stati precedentemente assegnati alla scheda.

Si ricorda che l'indirizzo link-local, nel caso delle interfacce Ethernet, viene assegnato in maniera automatica e non ha quindi bisogno di essere esplicitato.

MTU (*Maximum Transmission Unit*)

MTU è la lunghezza massima in byte che può avere un pacchetto IP trasmesso sulla Ethernet. Per lo standard ethernet v2 (IEEE 802.3) si ha di norma `MTU = 1500`, che è il massimo valore impostabile.
Il minimo MTU è impostabile a 64 byte (per le Ethernet), ma il protocollo IPv6 non accetta MTU al di sotto di 1280 byte. Tuttavia `ifconfig` di Linux accetta valori di MTU tra 1 e 2000.
In condizioni normali questa impostazione viene sempre lasciata a 1500 bytes.
Per impostarlo:

`$ ifconfig eth# mtu <mtu>`

NB:
Le impostazioni di `ifconfig` vanno perse in caso di riavvio del sistema.
Per far sì che le impostazioni IP vengano mantenute al riavvio del sistema occorre modificare il file `/etc/sysconfig/network-scripts/ifcfg-eth#` (dove # rappresenta il numero dell'interfaccia) con un qualsiasi editor di testo.
A questo scopo può essere utile il comando `ifcfg`.

Indirizzo di *loopback*

Va detto che in ogni host è presente una interfaccia di *loopback* che rappresenta l'host stesso; solitamente viene chiamata `lo` e ha indirizzo IPv4 `127.0.0.1/8` e indirizzo IPv6 `::1/128`.
Questa interfaccia ha lo scopo di poter utilizzare lo stack di protocolli TCP/IP anche se il computer non è dotato di schede di rete. Rappresenta infatti una interfaccia di rete "virtuale", nel senso che i dati non vengono inviati a nessun dispositivo ma vengono ritornati all'host stesso. In questo modo è anche possibile verificare se il sistema operativo è in grado di far funzionare correttamente il modulo TCP/IP (chiamato *socket*) .

5 - CONFIGURAZIONE DEI PARAMETRI DI ROUTING

Il routing è un argomento estremamente vasto (protocolli di routing, *load balancing*, routing dinamico ecc.) che non può certamente essere esaurito in poche righe; per ovvi motivi noi ci limiteremo alle nozioni base necessarie per portare a termine l'esercitazione e per risolvere comunque le più diffuse problematiche.
Il comando principale che permette di gestire il routing tra interfacce è `route`. `route` permette di "manipolare" le tabelle di routing del sistema, inserendo o cancellando regole di routing statiche, che cioè che non cambiano mai. Durante l'esercitazione basterà quindi implementare le tabelle di routing già predisposte durante la soluzione dell'esercizio proposto utilizzando le opzioni del comando `route` presentati di seguito.

Per aggiungere una regola che riguarda una rete o una sottorete:
`$ route --inet6 add <dest ip>/<netmask> gw <gateway>`

Per aggiungere una regola riguardante un singolo host:
`$ route --inet6 add -host <dest ip> gw <gateway>`

Per aggiungere la regola di *default*:
`$ route --inet6 add default gw <ip gateway>`
oppure:
`$ route --inet6 add ::/0 gw <ip gateway>`

Per eliminare una regola è possibile utilizzare la stessa sintassi vista sopra sostituendo `add` con `del`.

Per visualizzare le regole inserite si digiti poi il comando
`$ route --inet6 -n`.

E' da tenere presente che nella tabella di routing per ogni interfaccia vengono già inserite di default una o più regole (che noi però trascuremo); queste possono essere contraddistinte dalla presenza all'interno della tabella di un "*" nel campo Gateway.

Con l'opzione `reject` invece è possibile eliminare tutti i pacchetti diretti verso una certa rete esterna, per evitare ad esempio che questa sua raggiungibile passando da router in questione.

Un'altra opzione che può avere qualche applicazione è `dev <interf>`, che permette di specificare su quale interfaccia deve essere inviato il pacchetto destinato ad una determinata rete.
Nella maggior parte dei casi non è necessario specificare questo parametro, in quanto l'interfaccia da utilizzare può essere identificata immediatamente in base al proprio indirizzo IP; tuttavia torna utile nel caso in cui ad esempio si voglia testare un host connesso alla stessa rete mediante due diverse interfacce. Un altro utilizzo si può trovare nei cosiddetti tunnel, ovvero quando un pacchetto non deve essere inviato direttamente ad una scheda ma deve essere prima "incapsulato" in un altro pacchetto, per poi essere immesso in un tunnel.

Ulteriori opzioni di `route` sono usate per scopi raffinati o diagnostici, nonché dai programmi di instradamento automatico dei pacchetti quali il demone `routed`**,** che implementa il protocollo RIPv2.

6 - ABILITAZIONE DEL FORWARDING

In Linux è possibile che un host fuonzioni da router, commutando i pacchetti da una interfaccia all'altra (*forwarding*). Per default, e per ragioni di sicurezza, il sistema operativo disabilita la funzione di forwarding; se si intende rendere attivo il forwarding bisogna pertanto abilitarlo esplicitamente andando a modificare le impostazioni del kernel stesso.

Molti dei parametri di funzionamento del kernel Linux possono essere modificati "al volo" senza dover ricorrere ad un riavvio della macchina ad ogni modifica. Tali modifiche possono essere apportate utilizzando il comando `sysctl` e salvate all'occorrenza nel file di configurazione `/etc/sysctl.conf`, per poterle così caricare ad ogni boot del sistema.
Queste "regolazioni" del kernel possono però essere apportate anche scrivendo direttamente all'interno dei file contenuti nel filesystem virtuale `proc`. Quest'ultimo rappresenta infatti un'astrazione del funzionamento del kernel e del sistema: al suo interno sono presenti moltissimi file, organizzati in cartelle e sottocartelle, il cui contenuto rappresenta lo stato dei processi, della memoria, della cpu e di tutte le impostazioni del kernel stesso.
Per visualizzare il contenuto dei file presenti nel `procfs` è sufficiente utilizzare il comando cat;
 # cat /proc/sys/vm/page-cluster visualizza ad esempio un'impostazione della memoria virtuale (nello specifico il numero di pagine che dalla RAM vengono atomicamente trasferite alla swap ad ogni richiesta del kernel)
Per modificare questa impostazione è possibile scrivere direttamento all'interno del file con il comando echo
 # echo "4" > cat /proc/sys/vm/page-cluster
o utilizzando il comando sysctl
 # sysctl -w vm.page-cluster = 8

Nei nostri casi il forwarding deve essere abilitato sui PC che fungono da Router.

Per abilitare (ON = `1`) il forwarding su una specifica interfaccia:
$ echo 1 > /proc/sys/net/ipv6/conf/eth#/forwarding
oppure
$ sysctl -w net.ipv6.conf.eth#.forwarding=1

Poiché in laboratorio utilizzeremo pressoché tutte le interfacce, conviene abilitare il forwarding su tutte con un unico comando:
$ sysctl -w net.ipv6.conf.all.forwarding=1

Per disabilitarlo ovviamente si procede inmaniera analoga, impostando il valore del parametro a `0`.

Per controllare che il forwarding sia veramente abilitato, si può scrivere:
$ sysctl -a | grep forwading
e verificare che alla riga:
net.ipv6.conf.all.forwarding
corrisponda il valore `1`.

7 - VERIFICHE DEL FUNZIONAMENTO DELLA RETE

Siamo giunti alla prova finale, ovvero la verifica delle connessioni di rete. Si tratta di verificare se ogni PC è in grado di "comunicare" in maniera esatta con gli altri PC.

Il comando utilizzato per verificare le connessioni IPv6 è `ping6`.

`ping6` ha la stessa funzione del noto `ping` per la versione IPv4. La risposta a questa domanda può assumere tre forme diverse:

1) Il `ping` va a buon fine e l'host interpellato risponde correttamente;
2) L'host contattato non risponde alla richiesta;
3) Il nostro host non riesce ad inviare il pacchetto di ping.

Un esempio di `ping6`:

```
$ ping6 3ffe:1001:1c0:ffff::
```

la cui risposta può essere:

```
PING 3ffe:1001:1c0:ffff::(3ffe:1001:1c0:ffff::) 56 data bytes
64 bytes from 3ffe:1001:1c0:ffff::: icmp_seq=0 ttl=64 time=0.187 ms
64 bytes from 3ffe:1001:1c0:ffff::: icmp_seq=1 ttl=64 time=0.181 ms
64 bytes from 3ffe:1001:1c0:ffff::: icmp_seq=2 ttl=64 time=0.179 ms
64 bytes from 3ffe:1001:1c0:ffff::: icmp_seq=3 ttl=64 time=0.179 ms
```

Questo indica non solo che l'interfaccia interpellata sta funzionando e risponde ai "ping", ma che abbiamo anche ricevuto 64 bytes in un tempo di andata e ritorno di circa 0.0180 ms.

NB: il comando ping senza opzioni esegue un test continuativo; per interromperlo è necessario premere i tasti CTRL + C.

Ping verso le interfacce locali

Una delle prime cose da verificare in seguito alla configurazione di un host, è la corretta configurazione delle interfacce locali. Per verificare la corretta assegnazione dell'indirizzo IP all'interfaccia è possibile eseguire verso di essa un `ping6`.

Esempio:
Supponiamo che un host abbia i seguenti indirizzi IP:
`eth0: 3ffe:1001:1c0:1::1`
`eth1: 3ffe:1001:1c0:3::2`

Se lanciamo i comandi

```
$ ping6 ::1      (interfaccia di loopback)
$ ping6 3ffe:1001:1c0:1::1
$ ping6 3ffe:1001:1c0:3::2
```

Se tutte le interfacce sono funzionanti, come risposta ad ogni ping specifico avremo:

```
64 bytes from xxx: icmp_seq=1 ttl=64 time=0.056 ms
64 bytes from xxx: icmp_seq=2 ttl=64 time=0.056 ms
```

Se tra questi comandi ce n'è uno (o più) che non dà la risposta corretta, si è sbagliato qualcosa durante l'impostazione dei parametri di rete oppure, più semplicemente, non si è digitato il comando
$ ifconfig eth# up.

Test di raggiungibilità degli altri host

Realizzata la rete e configurati i computer, occorre verificare se il nostro host è in grado di comunicare correttamente con gli altri host. Questo lo si fa "pingando" tutti gli altri indirizzi IP della rete.
Le possibili risposte del sistema sono:

a) *64 bytes from xxx: icmp_seq=1 ttl=64 time=0.056 ms*

Se tutto funziona correttamente, appaiono sempre righe simili a quelle sopra citate, dove xxx è l'indirizzo IPv6 puntato:

b) `connect: Network is unreachable`

Se uno dei nodi intermedi non riesce ad instradare il pacchetto di "ping" (non esiste una regola di routing per instradare quel pacchetto):

c) `From 192.168.0.1 icmp_seq=1 Destination Host Unreachable`

Se la rete è raggiungibile, ma l'interfaccia non risponde:

Possibili cause di errore possono essere:

1. Si è sbagliato l'indirizzo IP, o comunque l'indirizzo non può essere raggiunto, pur essendo logicamente adiacente all'host in questione.
2. Alcune tabelle di routing sono sbagliate o incomplete sul nodo stesso o su host intermedi.
3. Non si è configurato correttamente il percorso di andata e ritorno del pacchetto tramite il comando route sui vari host.
4. Se tra il nostro host e l'host da pingare c'è di mezzo un firewall (anche se non è il nostro caso), talvolta i ping possono non andare a buon fine; in tal caso, questo non significa che l'host non è raggiungibile, ma semplicemente che il firewall ha impedito il `ping` disabilitando il protocollo ICMP.

Verifica del percorso seguito da un pacchetto

Talvolta è necessario verificare quale percorso abbia seguito un pacchetto; il comando `traceroute` permette di registrare il percorso seguito sia all'andata che al ritorno visualizzando gli indirizzi IP coinvolti

`$ traceroute6 <indirizzo ip>`

Verifica della comunicazione

A rete configurata, una verifica molto importante consiste nell'osservare i pacchetti che transitano sul canale di comunicazione: tale procedura è in gergo chiamata "sniffing" della rete.
In parole povere, essa consiste nel vedere "cosa passa" nei cavi.

Tcpdump

Un programma utile per eseguire lo sniffing è `tcpdump`. Questo software permette di mettersi in "ascolto" su una certa interfaccia e vedere l'intestazione IP e TCP dei pacchetti che passano.
Se si lancia:
`$ tcpdump –i <nome interfaccia>`
ad esempio
`$ sudo tcpdump –i eth0`
Si ottengono delle righe simili:

```
...
17:21:17.662664 3ffe:1001:1c0::1 > 3ffe:1001:1c0::2: icmp6 echo request 0
17:21:17.662777 3ffe:1001:1c0::2 > 3ffe:1001:1c0::1: icmp6 echo reply 0
```

In pratica significa che alle `17:21:17` il sistema `3ffe:1001:1c0::1` ha lanciato una `icmp6 echo request` (`ping6`) verso `3ffe:1001:1c0::1`, il quale ha poi risposto (`echo reply`).

Dall'esempio proposto si vede come `tcpdump` sia in grado di fornire informazioni molto importanti sul transito dei pacchetti di rete.

Ethereal

Un altro tool, molto più potente di `tcpdump`, è `ethereal`. Per farlo funzionare è necessario accedere all'interfaccia grafica e scrivere a terminale grafico `sudo ethereal` (anche questo comando è normalmente disabilitato per impedire che qualsiasi utente senza particolari priveligi possa "sniffare" il traffico di rete della macchina su cui sta lavorando). A questo punto scegliendo `start` dal menù `capture` è possibile indicare il dispositivo di rete da "sniffare"; premendo poi `OK` il programma inizierà a raccogliere informazioni sui pacchetti in transito, classificandoli per protocollo.
Una volta concluse le operazioni di cattura, sarà possibile andare a vedere nel dettaglio tutti i pacchetti transitati ed il loro contenuto (e non solo le intestazioni, come accade invece con `tcpdump`).

Verifica delle comunicazioni TCP/UDP

Il comando `netstat` visualizza le connessioni TCP, le porte d'ascolto TCP e UDP, nonché lo stato degli stream Unix.
Se si digita:

```
# netstat -an |more
```

o meglio:

```
# netstat -an --inet6 | more
```

Si ottiene, nella parte iniziale della risposta, l'elenco completo delle porte TCP e UDP che attualmente il sistema sta "ascoltando".
Inoltre l'elenco contiene anche le connessioni TCP che sono instaurate (*established*), nonché l'elenco delle trasmissioni di pacchetti UDP che si sono appena verificate.

8 - IMPOSTAZIONE DEI NOMI DEGLI HOST

Ogni host connesso in rete ha un suo nome e a tale nome è associato un indirizzo IP (i nomi sono stati introdotti perchè molto più pratici e semplici da ricordare rispetto ad una sequenza di numeri!).
In teoria ogni host dovrebbe conoscere sempre l'indirizzo IP associato a ciascuno degli altri host connessi per poter comunicare con esso nel caso un cui venga specificato solamente il nome; questo significa che ogni host deve essere in grado di "risolvere" un nome, cioè trovare l'indirizzo IP associato ad un *hostname*.
Nelle reti piccole è possibile impostare staticamente in un tabella presente su ogni host i nomi degli altri host e l'indirizzo associato; nelle grandi reti (Internet) questa però non è più ovviamente una soluzione praticabile, poiché in reti di questo tipo

1) i nomi, gli indirizzi e le informazioni sono in continua evoluzione;
2) gli host connessi possono essere tantissimi (si parla di quasi 200 milioni di host connessi a Internet);
3) l'aggiornamento e la ricerca delle voci deve avvenire in tempi acceettabili

Quello che viene utilizzato in Internet è invece un servizio DNS che si occupa di risolvere i nomi dell'intera rete Internet; i servizi DNS rappresentano database distribuiti su migliaia di calcolatori connessi a Internet in grado di mantenere tutte le informazioni per i domini registrati.
Per fare delle prove, con un PC correttamente connesso a Internet si possono usare i comandi `nslookup`, `dig` e `host` per trovare l'indirizzo IP di un host specificando il suo nome. L'output del comando `dig` per un host IPv6-enabled è:

```
# dig www6.ipv6.polito.it

; <<>> DiG 9.2.4 <<>> www6.ipv6.polito.it
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37534
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;www6.ipv6.polito.it.         IN          A

;; ANSWER SECTION:
www6.ipv6.polito.it.         349059      IN    CNAME freebsd.ipv6.polito.it.
freebsd.ipv6.polito.it.      349059      IN    A     130.192.16.134

;; AUTHORITY SECTION:
ipv6.polito.it.              349059      IN    NS    freebsd.ipv6.polito.it.

;; ADDITIONAL SECTION:
freebsd.ipv6.polito.it. 349059      IN    AAAA  2001:760:400:1:201:2ff:fe07:a2d7

;; Query time: 432 msec
;; SERVER: 2001:760:2::a#53(2001:760:2:0::a)
;; WHEN: Mon Jan 17 16:40:24 2005
;; MSG SIZE  rcvd: 117
```

La risposta ci dice che il server `www6.ipv6.polito.it` (il nome della macchina che risponde è `freebsd.ipv6.polito.it`) ha per indirizzo IPv4 `130.192.16.134` (si noti come sia una risposta di tipo A ad una query DNS) e `2001:760:400:1:201:2ff:fe07:a2d7` come indirizzo IPv6 (risposta di tipo AAAA). Come facilmente si può intuire, in una piccola rete come quella realizzata da noi in laboratorio non è necessario impostare i servizi DNS; per i computer locali è sufficiente editare il file `/etc/hosts` e specificare:

```
IP   nome_completo_host   [nome_locale_host]
```

Ad esempio:

```
217.169.102.120     www.unimn.it    www
```

Una volta effettuata questa operazione su tutti i PC interessati, è possibile usare i nomi locali o i nomi completi per effettuare tutte le operazioni in sostituzione degli indirizzi IP (decisamente meno facili da ricordare!)

Per far sì che un sistema Linux connesso a Internet utilizzi il servizio DNS, basta inserire l'indirizzo IP del server DNS preferito nel file `/etc/resolv.conf`, nella seguente modalità:
`nameserver <DNS_IP_Addr>`

Ad esempio
`nameserver 2001:760:2:0::a`

Questo indica che il sistema utilizzerà server DNS indicato per trovare gli indirizzi IP degli host che non sono presenti nel file `/etc/hosts`. Nel file `/etc/resolv.conf` si possono specificare più server DNS, utilizzando per ogni riga la forma sopra indicata (se si specificano più server DNS, nel caso in cui il primo non risponda, si utilizzerà il secondo e così via…).

9 - CONNESSIONE ALLA RETE MONDIALE IPv6

Il laboratorio di reti telematiche presso il quale avviene l'esercitazione possiede una connessione verso uno dei progetti di rete mondiale IPv6 chiamato 6Bone; tuttavia non è disponibile alcuna connessione diretta verso la dorsale di 6Bone ma quest'ultima viene raggiunta tramite un punto di accesso che fornisce al laboratorio stesso il servizio.

Il punto di accesso consiste nel laboratorio TILab di TelecomItalia il quale, facendo parte sia della rete 6bone che della rete Internet IPv4, ha la possibilità di porre in comunicazione computer che si trovano sulle 2 reti.

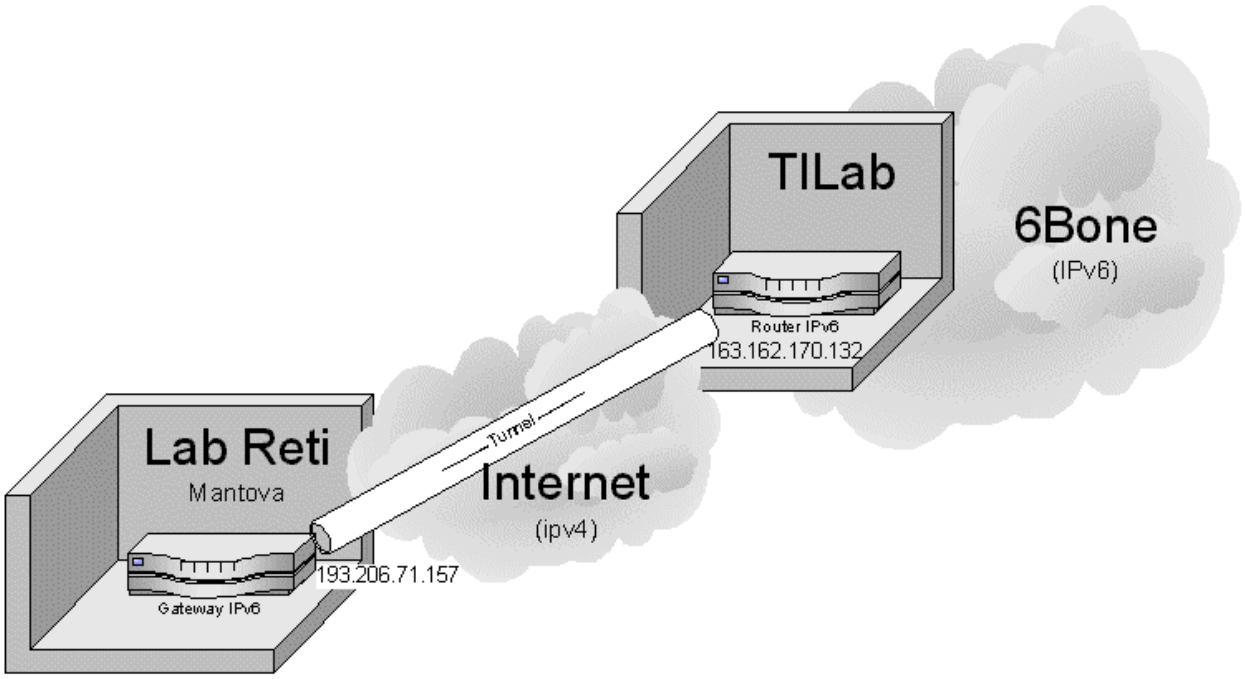
La connettività verso il TIlab avviene tramite un tunnel che incapsula IPv6 all'interno di IPv4 e che permette alle comunicazioni IPv6 provenienti dal (e dirette verso) il laboratorio di Mantova di raggiungere la rete 6Bone in maniera del tutto trasparente agli altri computer IPv6.

I due Router che si trovano ai capi del tunnel possiedono una numerazione pubblica IPv4 fissa di seguito riportata:

Lab Reti Mantova: 193.206.71.157

TILab: 163.162.170.132

Di seguito è riportato lo schema di principio che rappresenta quanto sopra descritto:



Una volta stabilita la comunicazione tra i 2 router tramite un tunnel, sarà possibile utilizzare il gateway IPv6 del laboratorio reti come nodo di accesso locale che apre la strada verso 6Bone.

Un qualsiasi computer con un protocollo IPv6 opportunamente configurato può entrare a far parte della rete 6Bone passando attraverso il tunnel.

Per questo motivo al Lab Reti è assegnato una spazio di indirizzamento pubblico IPv6 utilizzabile per effettuare i nostri esperiementi. Lo spazio è il seguente:

3ffe:1001:1c0::/48

A fronte di un indirizzo tale sarà possibile numerare le macchine locali al laboratorio anche creando eventuali sottoreti laddove ne fosse necessario.

Realizzazione pratica del Tunnel

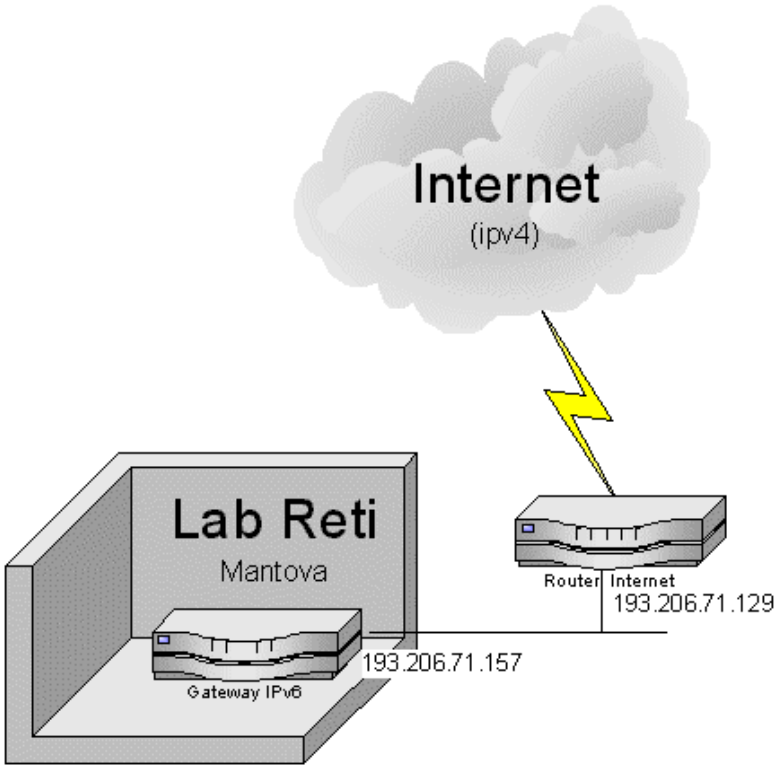
Come spiegato nella sezione precedente la connettività verso la rete mondiale 6Bone è realizzata da un tunnel che incapsula i pacchetti IPv6 all'interno della rete Internet IPv4. All'interno del laboratorio esiste un computer destinato esclusivamente a svolgere tale funzione e denominato Gateway IPv6.

Il Gateway IPv6 possiede un sistema operativo Linux Fedora Core 3, il quale è in grado di implementare la funzionalità di tunnelling di IPv6 in IPv4. Per rendere effettivo il tunnel sarà quindi necessario svolgere le seguenti operazioni:

a) Configurare un'interfaccia con numerazione pubblica verso la rete mondiale IPv4

b) Configurare il tunnel

c) Definire la una tabella di Routing capace di instradare verso il Tunnel tutte le comunicazioni che sono rivolte all'esterno.



a) Configurare un'interfaccia con numerazione pubblica verso la rete mondiale IPv4

Il Gateway IPv6 deve essere configurato in modo da avere un'interfaccia che comunica con la rete Internet IPv4 in modo da poter raggiungere la controparte del tunnel posta presso il TILab.

L'indirizzo pubblico IPv4 assegnato al Gateway IPv6 è `193.206.71.157`

Il comando per assegnarlo all'interfaccia di rete è:

```
$ ifconfig eth0 193.206.71.157
```

Come è possibile vedere dallo schema sopra riportato il laboratorio fa parte di una rete locale, il cui punto di accesso verso Internet è un router il cui indirizzo è `193.206.71.129`

L'interfaccia del router IPv4 deve essere impostato come *default gateway* nella tabella di routing con il comando:

```
$ route add default gw 193.206.71.129
```

Una volta eseguiti questi due comandi il computer fa parte della rete Internet, per cui è bene testare la connessione alla rete provando un comando di `ping` verso un indirizzo pubblico conosciuto:

```
$ ping  www.unipv.it
```

b) Configurare il tunnel

Il sistema operativo che definisce il tunnel verso il TILab deve tenere in considerazione che la controparte possiede l'indirizzo pubblico IPv4 `163.162.170.132`

Il comando per attivare il tunnel è:

```
$ ip tunnel add <nome tunnel> mode sit remote <ip remoto>
  local <ip locale>
```

Successivamente il tunnel deve essere attivato con il comando:

```
$ ip link set <nome tunnel> up
```

E' possibile verificare la presenza del tunnel mostrando la lista dei tunnel aperti con il comando:

```
$ ip -6 tunnel show
```

oppure assincerandosi che il tunnel compaia nell'elenco delle interfacce visualizzato da `ifconfig`

Una volta stabilito e attivato un canale capace di incapsulare il protocollo IPv6, sarà possibile assegnare al capo del tunnel che ci compete un indirizzo IPv6, naturalmente scelto all'interno dello spazio assegnato:

```
$ ifconfig <nome tunnel> inet6 add <indirizzo IPv6>
```

c) Definire la una tabella di Routing

Sul Router deve essere definita una tabella di routing capace di instradare verso il Tunnel tutte le comunicazioni che sono rivolte all'esterno con il comando:

```
$ ip route add ::/0 dev <nome tunnel>
```

In questo modo si crea una regola di default verso il tunnel stesso (questi, una volta instaurato, viene infatti visto come fosse un'interfaccia).

Essendo un Router, il sistema operativo deve aver abilitato il forwading.

10 - INFORMAZIONI AGGIUNTIVE SU IFCONFIG

Un riferimento molto striminzito alle opzioni di `ifconfig` si ottiene digitando
`$ ifconfig –help`
La pagina di manuale è consultabile in qualsiasi momento digitando:
`$ man ifconfig`
anche se molto spesso risultano più aggiornate le pagine di documentazione GNU/Info, navigabili con il comando:
`$ info ifconfig`

FAMIGLIE DI INDIRIZZI

Durante l'assegnazione di un'indirizzo ad una interfaccia è necessario anche specificare la tipologia dell'indirizzo stesso; l'opzione di default è `inet` (IPv4) ma è possibile specificare indirizzi `inet6` (IPV6), `atalk` (AppleTalk, usato su reti Machintosh), `ipx` (IPX/SPX Microsoft), `link` (MAC-address di livello 2), `lladdr`…

In pratica per assegnare un MAC address diverso da quello del dispositivo:
`# ifconfig eth0 link <MAC_Addr>`

ARP

Le opzioni `arp` e `-arp` servono per attivare o disattivare l'utilizzo di protocolli ARP per la risoluzione degli indirizzi MAC. Nelle Ethernet ARP è necessario, altrimenti si dovrebbero specificare manualmente tutte le voci dell'ARP-table con il comando `arp`.

PROMISCUOUS

Normalmente il sistema analizza solo i pacchetti che sono direttamente indirizzati al suo indirizzo IP. Come detto, se utilizziamo un HUB, ogni pacchetto è trasmesso su tutte le interfacce. Nella modalità *promiscuous* è possibile ricevere anche i pacchetti che non sono stati indirizzati a noi.
Si attiva nel seguente modo:
`$ ifconfig eth0 promisc`
e si disattiva nel seguente modo:
`$ ifconfig eth0 –promisc`
In altre implementazioni, `promisc` è simile ad `allmulti`.

DRIVER MODE

Serve per impostare la modalità di funzionamendo del driver; nelle lan wireless, ad esempio, può essere '`802.11a`', '`802.11b`', '`802.11g`'.

STATISTICHE DI FUNZIONAMENTO

`$ ifconfig –s` visualizza le statistiche di funzionamento di ciascuna interfaccia.

11 - ALTRI ARGOMENTI NON TRATTATI IN QUESTO MANUALE

Su Internet si trovano la maggior parte dei documenti informativi riguardanti questo argomento.

Per ulteriori informazioni sulle specifiche dei protocolli di rete:

http://www.ietf.org
http://www.ietf.org/rfc
http://www.faqs.org
http://www.linux-ipv6.org
http://www.ieee.org
http://www.iana.org
http://www.6bone.net
http://www.tuttoreti.com
http://www.mrtg.org
http://dast.nlanr.net

Per documentazione riguardante il sistema GNU/Linux o applicativi funzionanti sotto Linux:

http://www.gnu.org
http://www.kernel.org
http://www.linux.org
http://www.linux.it
http://www.lugman.org
http://www.google.it/linux
http://sourceforge.net

Inoltre in molte distribuzioni di GNU/Linux è presente la cartella `/usr/share/doc` che contiene varia documentazione sui tools installati nel sistema, fornendo manuali ed esempi.

Siti IPv6-enabled

www.ipv6forum.com
www.kame.net
www.6bone.net
www6.caspur.it
www6.ipv6.polito.it

DNS utilizzabili

IPv4 – `151.1.1.1`
IPv6 – `2001:760:2:0::a`