Es(p)NET

NetEmul, Netkit: esperienze di networking

2015.11

Indice

0 La rete dentro l'host

Nella teoria delle reti un host è un computer della rete, il titolo del paragrafo, come l'immagine della copertina, invece fa riferimento alla possibilità di emulare una rete e il suo funzionamento mediante un opportuno software che gira in un computer. In questo modo si possono fare esperienze di configurazione e gestione di una rete senza necessità di averne a disposizione una effettiva e senza timore di pasticciare troppo con le configurazioni.

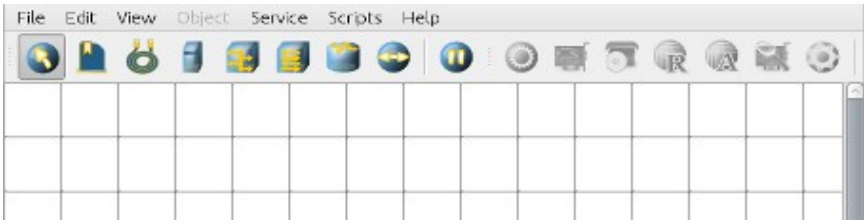
Questi appunti presentano due software grazie ai quali si possono effettuare esperienze di laboratorio di networking. Alcune di queste sono riportate come esempio. Le esperienze sono effettuate con i software:

- **NetEmul**: si tratta di un software grafico (sviluppato da Semenov Pavel e Omilaeva Anastasia) che permette di sperimentare i collegamenti fra hardware di rete (computer, hub, switch, router). Si progetta la rete utilizzando elementi grafici che rappresentano gli hardware della rete, si effettuano le configurazioni e il software con animazioni mostra il percorso dei dati nella rete. Principalmente, quindi, è uno strumento utile per imparare come configurare una rete, anche complessa, in modo che ogni nodo sia raggiungibile da parte degli altri. Si può installare dai repository delle distribuzioni.
- **Netkit**: in questo caso si tratta di un software più complesso e più realistico (sviluppato presso il Dipartimento di Informatica e Automazione dell'Università Roma Tre e mantenuto in collaborazione con il LUG di Roma Tre). Ogni computer della rete è accessibile e utilizzabile attraverso un terminale in cui è virtualizzata la macchina: ognuna è una macchina Linux a tutti gli effetti, in cui è installata una distribuzione minimale e in cui, se necessario, si possono installare ulteriori programmi (ad ogni computer della rete corrisponde un file sulla memoria di massa del computer in cui è in esecuzione il software). La rete, rappresentata dalle diverse istanze dei terminali, si può collegare ad Internet attraverso l'host in cui gira il software e con il quale la rete stessa può comunicare. Per il download e l'installazione: `http//wiki.netkit.com`

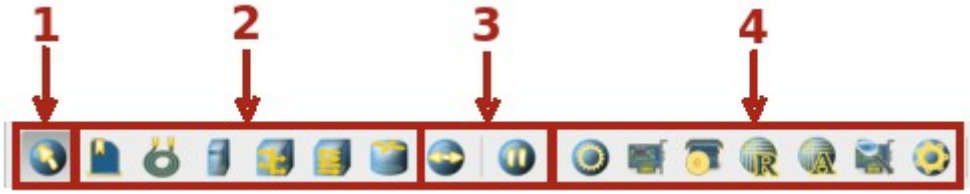
Per sperimentare con i software è necessario possedere conoscenze, cui non si fa riferimento in questi appunti, come i concetti di funzionamento e configurazione di una rete trattati negli appunti EsNET di cui questi appunti sono da considerarsi l'appendice di laboratorio. La somiglianza dei titoli ne richiama il legame.

1 Sperimentare con NetEmul

1.1 L'interfaccia grafica



Se dal menù *File* si seleziona *New*, viene mostrata l'area di lavoro, con una griglia per comodità di allineamento, dove si potranno posizionare i dispositivi. Nella barra degli strumenti utilizzabili, alcuni possono essere in grigio quando non disponibili in quel contesto.



Il primo a sinistra (**1**) è lo strumento di selezione. Quando bisogna configurare un dispositivo è necessario selezionarlo con il clic sinistro del mouse e per selezionarlo deve essere attivo lo strumento.

La serie di pulsanti **2** rappresentano oggetti che possono essere inseriti nel progetto della rete. Rispettivamente da sinistra verso destra: etichetta descrittiva, cavo di collegamento, computer, hub, switch, router.

Per inserire un dispositivo nella rete si seleziona il dispositivo dalla barra si porta il mouse nell'area di lavoro e si clicca con il tasto sinistro dove si vuole piazzare il dispositivo tante volte quanti dispositivi di quel tipo si vuole avere a disposizione. Scegliendo il pulsante di selezione il dispositivo si può spostare anche in altra posizione.

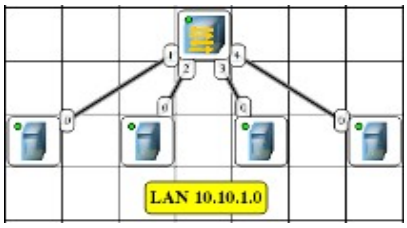
Per collegare due dispositivi basta selezionare lo strumento cavo di collegamento, fare clic sinistro sul primo dispositivo da collegare e trascinare fino a destinazione.

Il primo pulsante a sinistra, nel gruppo **3**, permettere di scegliere l'emittente e il ricevente dei dati. Effettuata la scelta una animazione mostra immediatamente il traffico presente nella rete. Il successivo pulsante permette di fermare o riavviare l'animazione. Per default l'animazione è attiva.

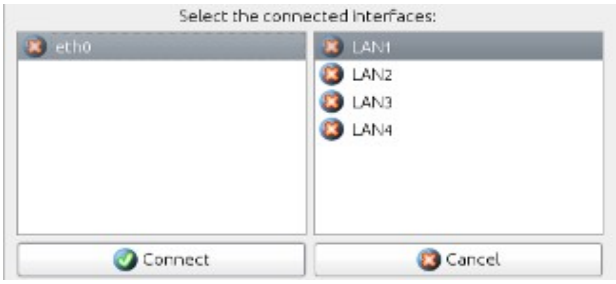
Se si seleziona un dispositivo inserito nell'area di lavoro, vengono attivati i pulsanti **4** che rappresentano gli strumenti di configurazione del dispositivo. Da sinistra a destra i pulsanti permettono di configurare o esaminare: le proprietà (per esempio il gateway), la/le scheda/e di rete, i programmi installati nel dispositivo, la tabella di routing per l'instradamento dei pacchetti, la ARP table per la corrispondenza fra indirizzi IP e MAC, il file di log dove mano a mano vengono registrati i pacchetti in transito nel dispositivo. L'ultimo a destra permette la generazione personalizzata di pacchetti (funzionalità non utilizzata in questi appunti).

Le configurazioni sono accessibili anche dal menù contestuale visualizzato in seguito al clic destro del mouse sul dispositivo o, anche, dal menu *Object*.

1.2 Rete gestita da switch



Per mostrare l'utilizzo degli strumenti base del programma ci si occuperà, come primo esempio, della configurazione di una semplice rete le cui comunicazioni sono gestite da uno switch. Nell'esempio proposto sono presenti 4 computer e uno switch. Inoltre una etichetta mostra l'indirizzo che si è scelto per la rete (naturalmente può essere qualsiasi).



Inseriti i 4 computer che faranno parte della rete, se si *stende* un cavo di collegamento fra ciascun computer e lo switch viene visualizzata una finestra per la scelta della porta dello switch cui collegare il cavo. Si può semplicemente confermare la scelta di default.

Le impostazioni di default prevedono 1 interfaccia ethernet per i computer (modificabile da *Service*, *Settings...*) e 4 porte per lo switch (modificabile dalle proprietà).

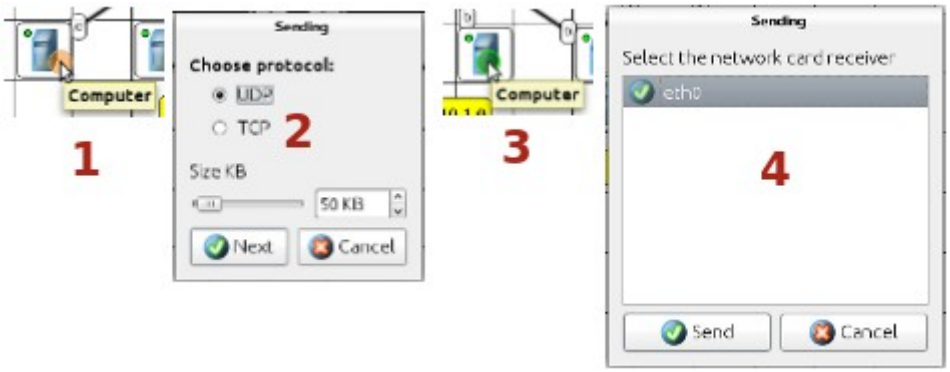
Quando il computer è collegato il pallino rosso visualizzato nell'icona diventa di colore giallo.

L'indirizzo IP di ogni scheda di rete si può configurare selezionando il dispositivo e premendo il pulsante dalla barra degli strumenti o scegliendo l'opzione dal menù contestuale.

Nell'esempio si è configurata la scheda di rete del computer con l'indirizzo 10.10.1.1 appartenete al range di indirizzi 10.10.1.0 scelto per la rete.

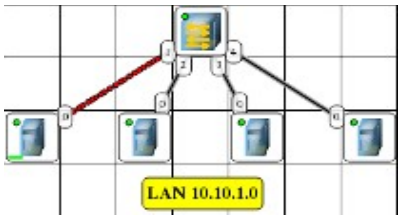
Mano a mano che si configurano gli indirizzi IP, il pallino di colore giallo che accompagna l'icona del computer diventa verde, ad indicare l'avvenuta configurazione e la possibilità di comunicazione.

Quando tutti i computer della rete sono configurati si può verificare l'esattezza delle configurazioni e la possibilità di comunicare.

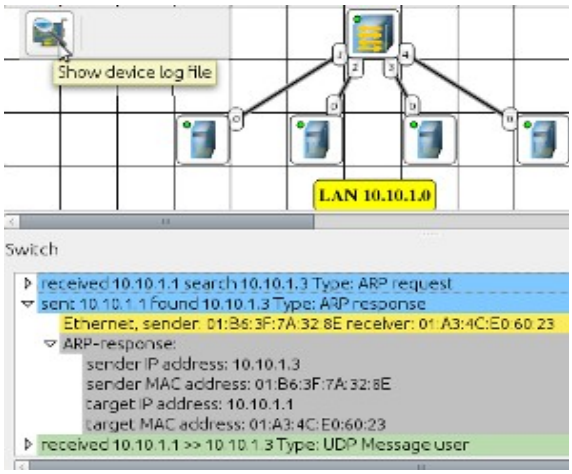


Se si seleziona lo strumento *Invia Dati* alla freccetta del mouse viene aggiunto un pallino arancione (**1**) e in questa modalità si può selezionare il computer che invierà i dati. La finestra di dialogo visualizzata dopo la selezione effettuata (**2**) consente di scegliere il tipo di dati da inviare ed, eventualmente, la dimensione. Si possono confermare le impostazioni proposte.

Dopo la conferma del tipo di dati la freccetta associata al mouse presenta un pallino verde (**3**) e in questa modalità si può scegliere il computer destinazione dei dati. Dopo la selezione della destinazione, la scelta presentata nella finestra di dialogo successiva (**4**) può essere confermata se nel computer è installata una sola scheda di rete. In caso contrario si sceglie quale scheda deve essere interessata a ricevere i dati.



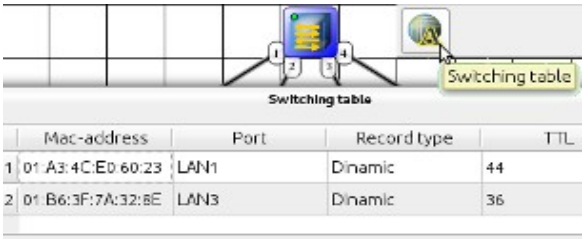
Selezionando il pulsante di avvio della simulazione (*Send*), il trasferimento di dati viene animato da dei pallini che percorrono i tratti della rete: dapprima una serie di pallini gialli indicano i pacchetti di interrogazione da parte dello switch degli indirizzi dei computer collegati. Il computer destinazione risponde con la conferma (pallino rosso) e i dati possono intraprendere il proprio cammino (serie di pallini rossi).



Se, dopo aver selezionato lo switch, si seleziona il pulsante per la visualizzazione del file di log, è possibile rendersi conto dei pacchetti transitati dal dispositivo che evidenziano la richiesta, secondo

il protocollo ARP, di informazioni sull'indirizzo MAC cui corrisponde quel determinato IP e la risposta da parte del computer interessato.

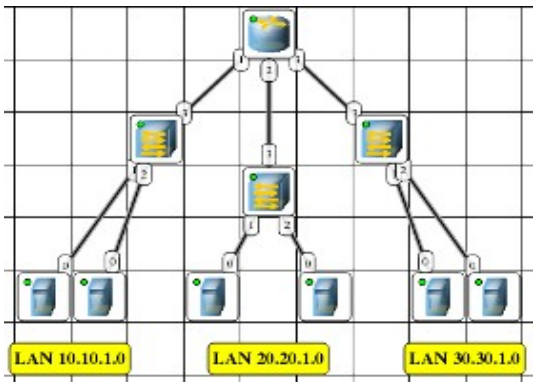
A questo punto può iniziare l'invio effettivo dei dati perché lo switch è in grado di stabilire in quale ramo di rete indirizzare i pacchetti.



Lo strumento *Switching table* mostra la ARP table con le corrispondenze fra indirizzi MAC e indirizzi IP. Lo switch è in grado di stabilire in quale ramo della rete indirizzare i dati in base a tale tabella.

L'esperienza mostrata può essere ripetuta, per esempio, sostituendo un hub allo switch e verificando il percorso che effettuano i dati in transito in entrata e in uscita dall'hub verso il computer destinazione.

1.3 Interconnessioni fra reti diverse



In questa esperienza ci si occuperà di far comunicare tre reti con range di indirizzi diversi e quindi sarà necessario utilizzare un router per l'instradamento dei pacchetti nella giusta direzione.

Oltre a configurare le varie schede di rete con l'indirizzo IP appartenente al range scelto per la rete cui appartiene il computer, è necessario configurare il gateway per l'*uscita* dei pacchetti di dati destinati a computer appartenenti a domini diversi.

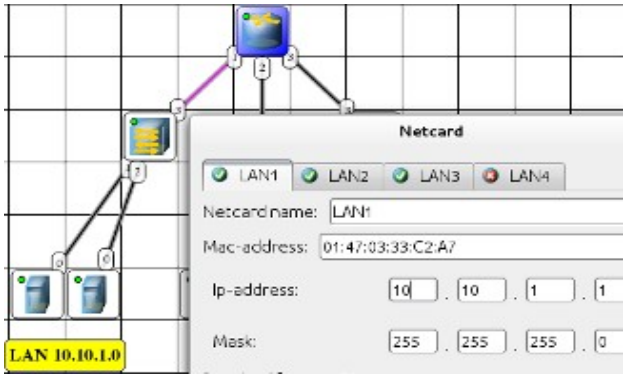
Dal pulsante delle proprietà si configura il gateway, che per comodità avrà come indirizzo il primo indirizzo della rete: per la rete 10.10.1.0 il gateway sarà 10.10.1.1. Ovviamente tutti i computer della rete verranno configurati con indirizzi successivi.

Routing table

	Destination	Mask	Gateway	Interface	Metric	Source
1	10.10.1.0	255.255.255.0	10.10.1.2	10.10.1.2	0	Connected
2	0.0.0.0	0.0.0.0	10.10.1.1	10.10.1.2	0	Static

La tabella di routing del computer indica, in modo corretto, che tutto il traffico in uscita dall'interfaccia 10.10.1.2 (IP del computer) va indirizzato al gateway con l'indirizzo specificato prima.

I computer delle reti 20.20.1.0 e 30.30.1.0 vanno configurati allo stesso modo.



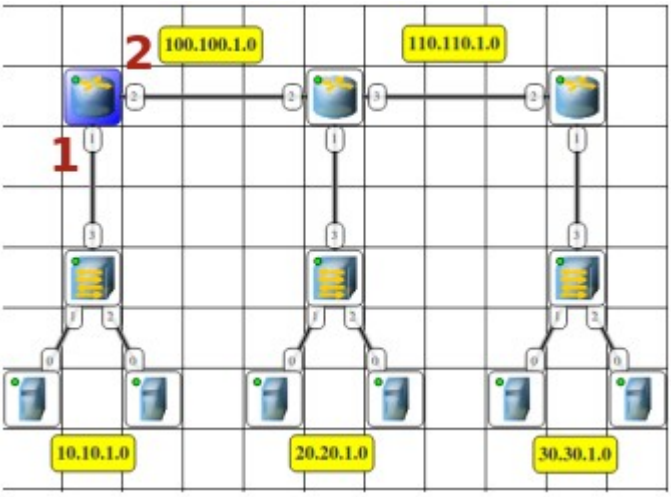
Il router va configurato in modo che sia abilitato il routing (nelle proprietà) e inoltre le varie schede di rete di cui è provvisto vanno configurate in modo che abbiano l'indirizzo corrispondente al gateway della rete cui sono collegate.

Si può controllare nella tabella di routing se le configurazioni indirizzano i pacchetti nella direzione corretta.

Se tutto è stato configurato in modo corretto i dati in partenza da qualsiasi computer di una delle tre reti dovrebbero poter raggiungere qualsiasi altra destinazione ed è possibile verificare la correttezza delle impostazioni avviando la simulazione di invio dati e specificando, al solito, origine e destinazione.

1.4 Routing: configurazione manuale

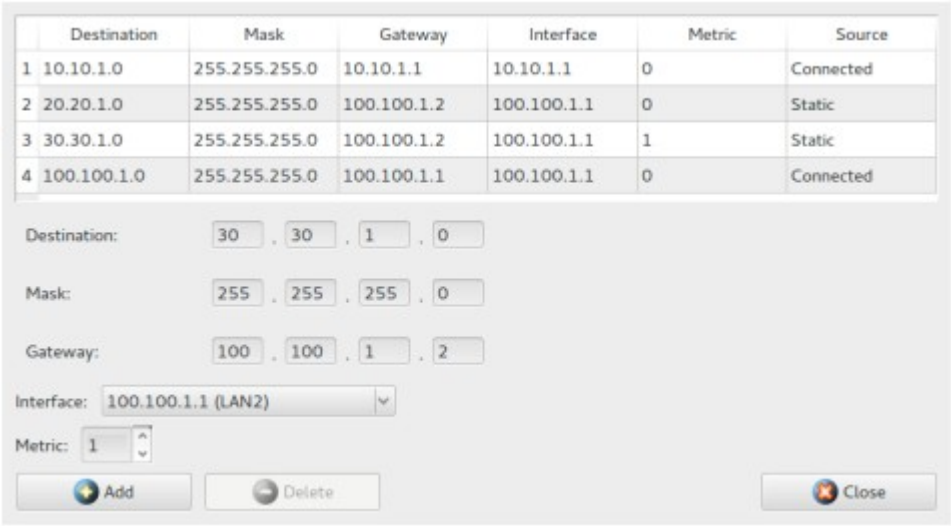
Nella precedente esperienza le diverse reti che dovevano comunicare erano collegate allo stesso router, in questa le reti dovranno comunicare per mezzo dei router che li gestiscono.



Nel router selezionato (il primo a sinistra) la tabella di routing dovrà essere configurata in modo che i pacchetti inviati dalla rete 10.10.1.0 possano raggiungere le reti 20.20.1.0 e 30.30.1.0.

Le schede dei computer, come descritto nell'esperienza precedente, vanno configurate scegliendo l'indirizzo in modo da appartenere al dominio specificato nell'etichetta.

Nel router: abilitare il routing dalla finestra delle proprietà in modo che il dispositivo possa inoltrare i pacchetti. Alla scheda di rete **1** va assegnato l'indirizzo 10.10.1.1 che sarà anche il gateway dei computer della rete 10.10.1.0. Alla scheda **2** va assegnato un indirizzo della rete 100.100.1.0, per esempio 100.100.1.1.



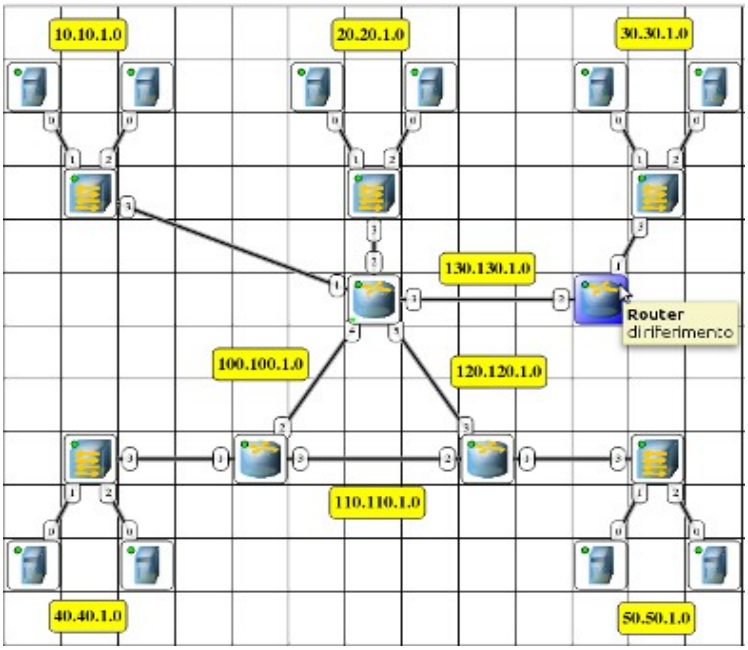
Abilitata la tabella di routing, affinché si possa raggiungere la rete 30.30.1.0 in *Destination* (**3**) si dovrà specificare l'indirizzo di rete, in *Mask* (**4**) la maschera di rete, in *Gateway* (**5**) la scheda che si dovrà occupare della gestione dei pacchetti inviati: si tratta della scheda del router centrale collegata con la 100.100.1.1 specificata in *Interface* (**6**) che quindi è la scheda da cui vengono inviati i pacchetti. Il parametro *Metric* di **7** indica che i pacchetti diretti alla rete specificata devono essere ulteriormente inviati avanti: il gateway sarà un punto di passaggio. Il numero specificato indica quanti passaggi necessitano affinché si possa raggiungere la destinazione (il numero di ***hop***). Con il pulsante *Add* si confermano le impostazioni e si aggiunge una riga alla tabella di routing (**8**) in cui è specificato l'instradamento dei vari pacchetti in transito nel router.

La configurazione del router a destra nella rete proposta dall'esperienza si configura in maniera simile: in questa saranno i pacchetti destinati alla rete 10.10.1.0 che dovranno effettuare un salto.

Il router rappresentato nella parte centrale del grafico ha una configurazione più semplice: tutte le reti possibili sono collegate direttamente:

	Destination	Mask	Gateway	Interface	Metric	Source
1	10.10.1.0	255.255.255.0	100.100.1.1	100.100.1.2	0	Static
2	20.20.1.0	255.255.255.0	20.20.1.1	20.20.1.1	0	Static
3	30.30.1.0	255.255.255.0	110.110.1.2	110.110.1.1	0	Static
4	100.100.1.0	255.255.255.0	100.100.1.2	100.100.1.2	0	Connected
5	110.110.1.0	255.255.255.0	110.110.1.1	110.110.1.1	0	Connected

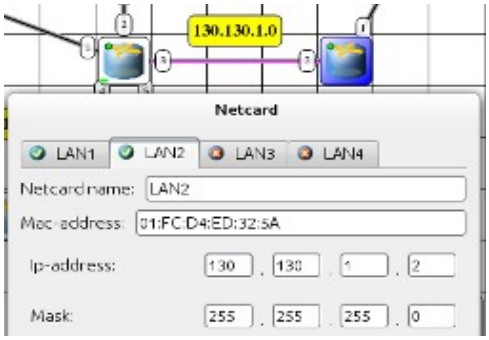
1.5 Rete dinamica



In questa esperienza ci saranno più reti gestite da router che dialogheranno fra di loro. I router sono collegati fra di loro (*communication subnet*). Inoltre si farà in modo, a differenza dell'esperienza precedente, che la rete possa crescere e i router possano adattarsi, automaticamente, in modo da sapere in ogni momento la strada da fare intraprendere ai dati che transitano da essi tenendo conto dei nuovi frammenti che si aggiungono. In pratica, in piccola scala, cosa avviene in Internet.

Per quanto riguarda i computer delle varie reti le configurazioni sono le stesse dell'esperienza precedente: indirizzo IP e gateway con il primo indirizzo della rete. Nella rete 10.10.1.0 il gateway sarà configurato, per esempio, con indirizzo 10.10.1.1 e ai vari computer verranno assegnati indirizzi da 10.10.1.2 in poi.

Gli indirizzi da assegnare alle varie reti sono riportati nelle etichette presenti nel grafico.

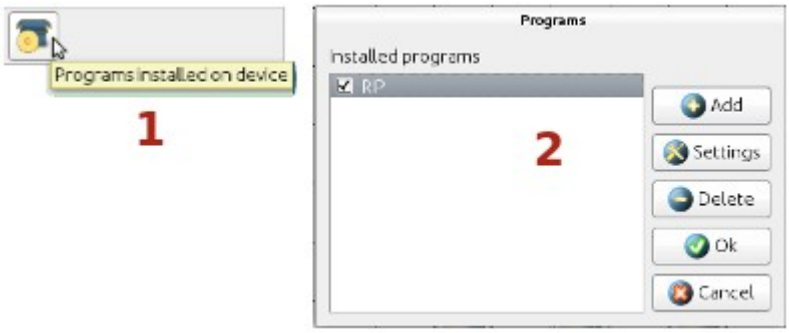


In ogni router sono configurate più schede di rete. Per esempio nel router di riferimento sono configurate due schede:

- una che *guarda* verso la rete locale (il gateway) a cui viene assegnato il primo indirizzo della rete che gestisce. Il router della rete 30.30.1.0 avrà la scheda cui è collegato lo switch configurata con l'indirizzo 30.30.1.1
- la scheda per il collegamento al router vicino ha indirizzo 130.130.1.2

Il router collegato ha 5 schede di rete. Ognuna è configurata in modo da avere un indirizzo appartenente al range specificato in figura per ogni segmento.

In ogni router dovranno essere abilitate le capacità di routing (pulsante delle proprietà)



Questa volta in ogni router bisogna installare un programma:

- Selezionare il router e quindi il pulsante per l'installazione di un servizio (**1**).
- Dalla finestra di dialogo dei programmi (**2**) premere il pulsante *Add*, scegliere **RIP** e confermare l'installazione.

Routing Information Protocol è un protocollo che consente ad un router di conoscere le vie per l'instradamento dei pacchetti. In pratica ogni router RIP trasmette, a intervalli di tempo (30 secondi per default), la propria tabella di routing agli altri router cui è collegato e così ognuno conosce le reti gestite dai vicini.

Routing table

	Destination	Mask	Gateway	Interface	Metric	Source
1	10.10.1.0	255.255.255.0	130.130.1.1	130.130.1.2	1	RIP
2	20.20.1.0	255.255.255.0	130.130.1.1	130.130.1.2	1	RIP
3	30.30.1.0	255.255.255.0	30.30.1.1	30.30.1.1	0	Connected
4	40.40.1.0	255.255.255.0	130.130.1.1	130.130.1.2	2	RIP
5	50.50.1.0	255.255.255.0	130.130.1.1	130.130.1.2	2	RIP
6	100.100.1.0	255.255.255.0	130.130.1.1	130.130.1.2	1	RIP
7	110.110.1.0	255.255.255.0	130.130.1.1	130.130.1.2	2	RIP
8	120.120.1.0	255.255.255.0	130.130.1.1	130.130.1.2	1	RIP
9	130.130.1.0	255.255.255.0	130.130.1.2	130.130.1.2	0	Connected

Selezionando il router di riferimento e facendo visualizzare la tabella di routing si notano i possibili cammini stabiliti:

- Se la destinazione è la rete 30.30.1.0 il cammino da percorre passa per la scheda configurata con indirizzo 30.30.1.1 (riga 3). È l'unica rete, assieme alla 130.130.1.0 (riga 9), collegata direttamente al router.
- Le informazioni relative al cammino per le altre sottoreti sono state inviate a questo router dagli altri router utilizzando il protocollo RIP, come notificato dall'ultima colonna della tabella. La colonna *Metric* indica il numero di salti (hop), il numero di stazioni a partire dalla posizione attuale, che i dati devono percorrere per raggiungere quella destinazione. La rete 10.10.1.0 e la rete 20.20.1.0 sono raggiungibili con un solo passaggio (righe 1 e 2), invece per raggiungere la 40.40.1.0 o la 50.50.1.0, così come si nota facilmente dal grafico, sono necessari due hop (righe 4 e 5). Il traffico diretto alla rete 10.10.1.0 passante per l'interfaccia 130.130.1.2 (riga 1) viene processato direttamente dal gateway 130.130.1.1 (la

scheda di rete del router cui è collegato questo router). Il traffico diretto alla 40.40.1.0 (riga 4) viene sempre processato da 130.130.1.1 ma per essere instradato ulteriormente alla prossima destinazione (*Metric* 2).

Se si prova ad inviare i dati, ora tutti i computer dovrebbero essere raggiungibili.

Si può provare inoltre ad aggiungere una ulteriore rete con il rispettivo router, sempre RIP, che ne gestisce il traffico, collegato ad uno o più router esistenti, e con le schede di rete configurate in modo da comunicare con la propria rete e con l'esterno con indirizzi appartenenti ad un range prefissato e osservare dopo un po' il pallino giallo che parte dal router e che indica l'invio della propria tabella di routing verso gli altri.

2 Sperimentare con Netkit

2.1 Caratteristiche generali e macchine virtuali

Netkit, ovvero *The poor man's system for experimenting computer networking* (come lo definiscono gli autori), è un sistema per emulare, quindi riprodurre le funzionalità di un sistema reale, una rete di computer. È basato su UML (User-Mode Linux) un kernel Linux che può girare come fosse un programma. Ogni apparecchiatura di rete emulata è una ***macchina virtuale*** Linux (**vm**). Il computer su cui girano le macchine virtuali è chiamato ***host***. Più macchine virtuali possono essere eseguite nello stesso tempo nello stesso host.

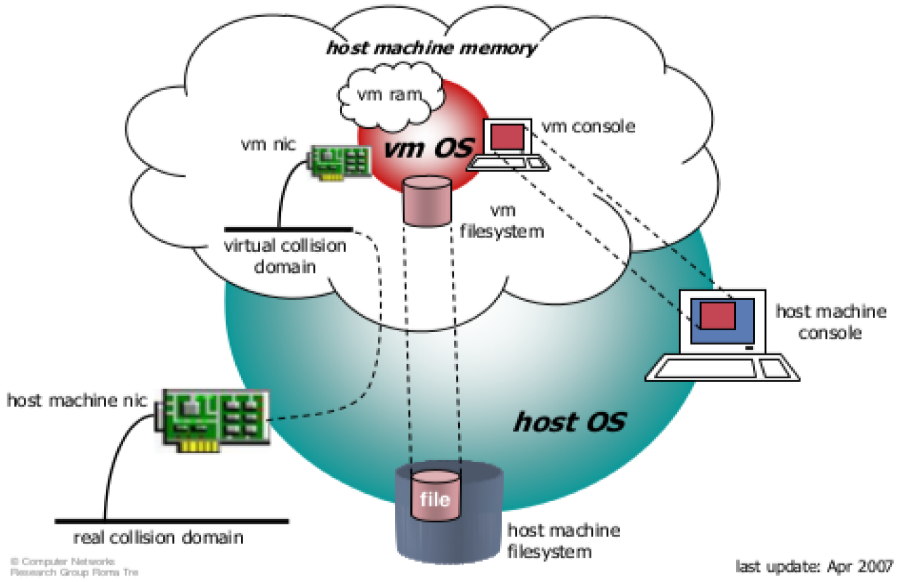


immagine da http://wiki.netkit.org

Ogni macchina virtuale è dotata di una console per controllarla (una finestra dentro cui gira una istanza del programma *Terminale*), di una memoria RAM (che è una parte della RAM dell'host), di un filesystem che è registrato in un file dell'HD dell'host e di una o più schede di rete. Ogni scheda di rete può connettersi ad un ***Dominio di Collisione***: *insieme di nodi che concorrono per accedere allo stesso mezzo trasmissivo e successivamente trasmettere* (da Wikipedia). In pratica si tratta di un hub virtuale che permette la comunicazione con gli altri dispositivi collegati ad esso.

Ogni macchina virtuale può essere configurata come l'host di una rete, uno router e, inoltre ogni macchina virtuale può comunicare con l'host e, attraverso questo, può collegarsi ad Internet.

Netkit rende disponibili un set di istruzioni per configurare e avviare le singole macchine virtuali (comandi che iniziano con la lettera v: *vcommands*) e un set di istruzioni per gestire facilmente, con singoli comandi, laboratori composti da più macchine virtuali (comandi che iniziano con la lettera l: *lcommands*). Tutti i comandi vanno inseriti da una finestra di Terminale attraverso il quale si controllano, dall'host, le macchine virtuali.

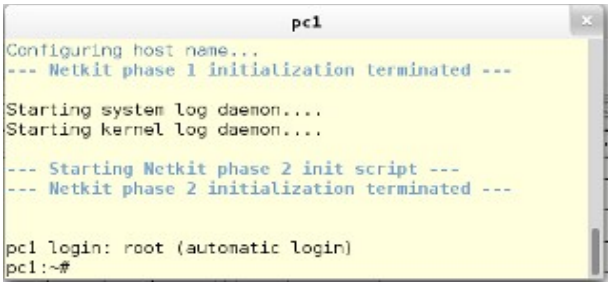
Comando	***Effetto***
`vstart`	Avvia una nuova macchina virtuale
`vlist`	Mostra le macchine virtuali attive

Comando	*Effetto*
`vhalt`	Ferma una macchina virtuale
`vcrash`	Forza il blocco di una macchina

La macchina virtuale si può fermare anche scrivendo il comando `halt` nella console della macchina stessa.

Una macchina virtuale si avvia, in generale, utilizzando da un terminale dell'host il comando:

```
host:~$ vstart pc1 --eth0=hub0
```



dove `pc1` è il nome che si vuole attribuire alla macchina virtuale e `hub0` è il dominio di collisione cui è collegata la macchina (questi due nomi si possono scegliere liberamente), `eth0` è l'unica scheda ethernet che è installata nella macchina. Altre schede possono essere definite aggiungendole alla riga di comando.

Nella macchina virtuale viene effettuato in automatico l'accesso come utente root e si possono fare tutte le operazioni permesse in una qualsiasi macchina Linux.

Più macchine collegate allo stesso dominio di collisione possono comunicare tra di loro. Per ogni macchina virtuale viene generato, sull'hard disk dell'host, un file con lo stesso nome della macchina virtuale che rappresenta l'hard disk della macchina. Nell'esempio il file si chiamerà `pc1.disk`. Essendo questo l'HD della macchina `pc1`, qui saranno conservate tutte le modifiche che si effettueranno e che si ritroveranno al prossimo avvio della macchina virtuale.

Il file generato è un file sparso (***sparse file***) ovvero un file registrato in modo da occupare lo spazio più efficientemente possibile: non vengono conservate le aree vuote ma informazioni per la gestione di tali aree.

```
host:~$ du -h --apparent-size pc1.disk
11G     pc1.disk
host:~$ du -h pc1.disk
1,1M    pc1.disk
```

Il comando `du` (Disk Usage) mostra l'occupazione di spazio del file. Il primo comando fornisce la dimensione *apparente* del file: è lo stesso valore che si ottiene se si esegue il comando `ls`. Il secondo comando mostra l'occupazione *effettiva* dello spazio su HD. Il parametro `-h` serve solamente a mostrare l'output in modo più comprensibile (human).

È possibile fare in modo che non venga generato il file:

```
host:~$ vstart pc1 --eth0=hub0 --hide-disk-file
```

Naturalmente in questo modo tutte le configurazioni effettuate sulla macchina virtuale vengono perdute.

Così come evidenziato dagli esempi, nei prossimi paragrafi, i comandi digitati da tastiera verranno scritti in **grassetto** e le risposte del sistema in carattere normale. Inoltre nel prompt verrà specificato, di volta in volta, in quale macchina viene eseguito il comando.

2.2 Rete locale: configurazioni di base

In questa esperienza ci si occuperà di configurare una rete locale.

La rete di esempio sarà composta da tre computer e quindi da un terminale dell'host si lanciano i comandi per la generazione di tre macchine virtuali con i nomi specificati.

Le tre macchine condividono lo stesso dominio di collisione e quindi possono comunicare.

```
host:~$ vstart pc1 --eth0=hub0
...
host:~$ vstart pc2 --eth0=hub0
...
host:~$ vstart pc3 --eth0=hub0
```

Ora si possono fare stampare informazioni su tutte le macchine virtuali in esecuzione:

```
host:~$ vlist
USER             VHOST               PID       SIZE  INTERFACES
tux              pc1                3889      11624  eth0 @ hub0
tux              pc2                4738      11624  eth0 @ hub0
tux              pc3                5451      11624  eth0 @ hub0

Total virtual machines:       3    (you),        3    (all users).
Total consumed memory:    34872 KB (you),    34872 KB (all users).
host:~$
```

Scelto come indirizzo di rete 10.10.1.0 le schede di rete di ognuno dei computer vengono configurate in modo da avere, ognuna, un indirizzo appartenente al range scelto.

```
pc1:~# ifconfig eth0 10.10.1.1 netmask 255.255.255.0
pc1:~#
pc2:~# ifconfig eth0 10.10.1.2 netmask 255.255.255.0
pc2:~#
pc3:~# ifconfig eth0 10.10.1.3 netmask 255.255.255.0
pc3:~#
```

Ora ogni macchina è accessibile da qualunque altra dello stesso dominio di collisione. Si può testare la connettività effettuando un *ping* da ogni macchina verso le altre:

```
pc2:~# ping 10.10.1.1
PING 10.10.1.1 (10.10.1.1) 56(84) bytes of data.
64 bytes from 10.10.1.1: icmp_seq=1 ttl=64 time=3.15 ms
64 bytes from 10.10.1.1: icmp_seq=2 ttl=64 time=1.15 ms
64 bytes from 10.10.1.1: icmp_seq=3 ttl=64 time=0.307 ms
^C
--- 10.10.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2011ms
rtt min/avg/max/mdev = 0.307/1.537/3.150/1.192 ms
pc2:~#
```

Si può ripetere il *ping* dalla macchina pc2 alla macchina pc1 ma stavolta avviando nella pc1 uno *sniffer di rete*. Lo sniffer è un programma che intercetta i pacchetti in transito da una interfaccia di rete. tcpdump è lo sniffer presente nelle distribuzioni Linux e quindi presente nella installazione

della macchina `pc1`.

```
pc1:~# tcpdump -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
```

il programma si è posto *in ascolto* della scheda `eth0`. Ripetendo il ping dalla macchina `pc2`, si possono osservare i risultati riportati da tcpdump:

```
19:24:23.780444 IP 10.10.1.2 > 10.10.1.1: ICMP echo request, id 2306, seq 1,
length 64
19:24:23.786964 IP 10.10.1.1 > 10.10.1.2: ICMP echo reply, id 2306, seq 1, length
64
19:24:24.781571 IP 10.10.1.2 > 10.10.1.1: ICMP echo request, id 2306, seq 2,
length 64
19:24:24.781610 IP 10.10.1.1 > 10.10.1.2: ICMP echo reply, id 2306, seq 2, length
64
19:24:25.791576 IP 10.10.1.2 > 10.10.1.1: ICMP echo request, id 2306, seq 3,
length 64
19:24:25.791616 IP 10.10.1.1 > 10.10.1.2: ICMP echo reply, id 2306, seq 3, length
64
19:24:28.772224 arp who-has 10.10.1.2 tell 10.10.1.1
19:24:28.772439 arp reply 10.10.1.2 is-at 2e:18:cc:d4:8c:e7 (oui Unknown)
^C
8 packets captured
8 packets received by filter
0 packets dropped by kernel
pc1:~#
```

Si possono notare i tre pacchetti provenienti dalla macchina con indirizzo 10.10.1.2 e le risposte, per ognuno. Sono presenti anche i pacchetti scambiati dal protocollo ARP per conoscere il MAC della scheda con indirizzo 10.10.1.2

Si può interrogare la tabella ARP di `pc2` per vedere come i pacchetti riescano a raggiungere la destinazione:

```
pc2:~# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.10.1.1                ether   6e:5f:98:37:0c:07   C                     eth0
pc2:~#
```

La cache ARP permane per breve tempo e se si interroga successivamente è probabile che non mostri informazioni.

2.3 Netkit lab

Quando si vuole emulare una rete complessa o si vogliono effettuare diverse elaborazioni anche in tempi diversi sarebbe noioso ripetere, ogni volta, i comandi per ogni macchina. Netkit mette a disposizione uno strumento che permette di configurare un laboratorio con le configurazioni di tutte le macchine, definite nel laboratorio, e lanciare l'*esecuzione del laboratorio* ovvero di tutte le macchine virtuali con le configurazioni impostate. Per i laboratori sono disponibili alcuni comandi, equivalenti a quelli per le macchine virtuali ma che riguardano tutte le macchine virtuali definite nel laboratorio e cominciano con la lettera l:

Comando	*Effetto*
`lstart`	Avvia un laboratorio
`linfo`	Mostra informazioni sul laboratorio senza avviarlo
`lhalt`	Ferma tutte le macchine virtuali del laboratorio
`lcrash`	Forza il blocco di tutte le macchine di un laboratorio

Il laboratorio, che in sostanza è una directory contenente alcune directory e file speciali, che si definirà ora, riprodurrà la configurazione definita nel paragrafo precedente.

- Per prima cosa bisogna creare una directory destinata al laboratorio:

```
host:~$ mkdir lab1
```

- nella direcotory `lab1` devono esistere tante sottodirectory quante sono le macchine virtuali che fanno parte del laboratorio e che devono avere lo stesso nome che si vuole assegnare alle macchine virtuali:

```
host:~$ cd lab1
host:~/lab1$ mkdir pc1
host:~/lab1$ mkdir pc2
host:~/lab1$ mkdir pc3
```

  nelle directory riservate alle singole macchine virtuali possono essere inseriti file che la macchina virtuale, la prima volta che viene avviata, copierà nel proprio filesystem. Se si vogliono inserire ulteriori file in un secondo momento è necessario cancellare l'HD associato alla macchina virtuale (per esempio `pc1.disk` per la macchina `pc1`) perché la sottodirectory viene letta solo la prima volta che la macchina viene avviata.

- Sempre nella `lab1` deve esistere il file di configurazione del laboratorio `lab.conf` dove vengono specificate le macchine virtuali che fanno parte del laboratorio. Nel caso di esempio il file conterrà le righe:

```
host:~/lab1$ cat lab.conf
pc1[0]=hub0
pc2[0]=hub0
pc3[0]=hub0

LAB_AUTHOR="Tux"
LAB_DESCRIPTION="Rete locale con 3 computer"
```

  le prime tre righe servono a definire i nomi delle macchine virtuali, le schede di rete (il numero fra parentesi quadra. Per esempio `0` vuol dire `eth0`) e il dominio di collisione cui le schede sono collegate. Le ultime righe servono solo come documentazione

- nella `lab1` possono essere presenti, per ogni macchina virtuale, due file in cui sono definiti i comandi da eseguire all'avvio della macchina virtuale e prima della chiusura della macchina virtuale. Ogni file ha il nome della macchina virtuale cui si riferisce e i suffissi `.startup` per i comandi di avvio e `.shutdown` per quelli di chiusura. Nell'esempio sono necessari solo i file per l'avvio che conterranno la configurazione degli indirizzi IP delle schede di rete:

```
host:~/lab1$ cat pc1.startup
ifconfig eth0 10.10.1.1 netmask 255.255.255.0
host:~/lab1$ cat pc2.startup
```

```
ifconfig eth0 10.10.1.2 netmask 255.255.255.0
host:~/lab1$ cat pc3.startup
ifconfig eth0 10.10.1.3 netmask 255.255.255.0
```

- nella lab1 possono essere presenti anche due altri file `shared.startup` e `shared.shutdown`, rispettivamente, per i comandi che sono eseguiti da tutte le macchine virtuali all'avvio e alla chiusura.

Ora si possono avere informazioni sul laboratorio anche se non è ancora in esecuzione:

```
host:~/lab1$ linfo

====================== Lab information ==========================
Lab directory: /home/tux/lab1
Version:       <unknown>
Author:        Tux
Email:         <unknown>
Web:           <unknown>
Description:
Rete locale con 3 computer

The lab is made up of 3 virtual machines ( pc1 pc2 pc3).
=================================================================

host:~/lab1$
```

Per avviare il laboratorio:

```
host:~/lab1$ lstart

======================== Starting lab ===========================
...
=================================================================
Starting "pc1"...
Starting "pc2"...
Starting "pc3"...

The lab has been started.
=================================================================

host:~/lab1$
```

Le macchine virtuali vengono avviate e ognuna avrà la propria scheda di rete già configurata.

Anche per fermare tutte le macchine è possibile utilizzare un comando per l'intero laboratorio:

```
host:~/lab1$ lhalt

======================== Halting lab ============================
...
=================================================================
Halting virtual machine "pc1" (PID 2987) owned by tux [...]
Halting virtual machine "pc2" (PID 3720) owned by tux [...]
Halting virtual machine "pc3" (PID 4490) owned by tux [...]
Removing readyfor.test...

Lab has been halted.
=================================================================
```

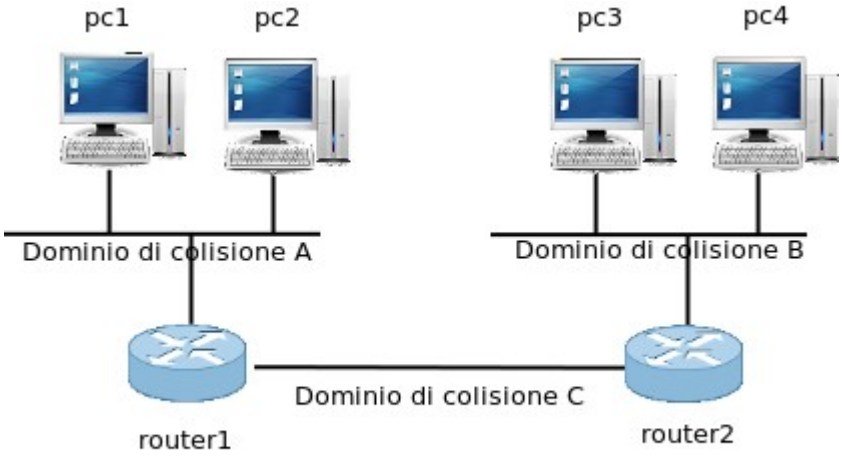
```
host:~/lab1$
```

Tutti i comandi possono essere inviati anche da una directory diversa, rispetto a quella dove è definito il laboratorio, specificando il parametro `-d` seguito dalla directory del laboratorio:

```
host:~$ lstart -d lab1
```

2.4 Comunicazione fra router

La configurazione proposta in questa esperienza riguarda la comunicazione fra due reti diverse gestite da router. Quando si tratta di configurare reti complesse è consigliabile prima disegnare la topologia che si vuole generare.



Per configurare il laboratorio, nella directory `lab2` si creano 6 directory (`pc1`, `pc2`, `pc3`, `pc4`, `router1`, `router2`).

Nel file `lab.conf` vengono configurate le schede di rete (una per ogni computer, due per ogni router):

```
host:~/lab2$ cat lab.conf
pc1[0]=A
pc2[0]=A

router1[0]=A
router1[1]=C

router2[1]=C
router2[0]=B

pc3[0]=B
pc4[0]=B

LAB_AUTHOR="Tux"
LAB_DESCRIPTION="Due reti che comunicano con router"
```

I computer del dominio A potrebbero, per esempio, appartenere alla rete 10.10.1.0, nel dominio B ci saranno indirizzi della rete 20.20.1.0 e nel dominio C gli indirizzi apparterranno alla rete 100.100.1.0.

I file `.startup` dei computer contengono oltre le solite righe di configurazione della scheda di rete anche la configurazione della tabella di routing con l'indicazione del gateway (la scheda eth0 del router cui sono collegati). Ai gateway, in conformità con gli altri esempi di questi appunti, verrà assegnato il primo indirizzo della rete di appartenenza:

```
host:~/lab2$ cat pc1.startup
ifconfig eth0 10.10.1.2 netmask 255.255.255.0
route add default gw 10.10.1.1
host:~/lab2$ cat pc2.startup
ifconfig eth0 10.10.1.3 netmask 255.255.255.0
route add default gw 10.10.1.1

host:~/lab2$ cat pc3.startup
ifconfig eth0 20.20.1.2 netmask 255.255.255.0
route add default gw 20.20.1.1
host:~/lab2$ cat pc4.startup
ifconfig eth0 20.20.1.3 netmask 255.255.255.0
route add default gw 20.20.1.1
```

Nei router oltre alle configurazioni delle due schede è necessario configurare i cammini per l'instradamento:

```
host:~/lab2$ cat router1.startup
ifconfig eth0 10.10.1.1 netmask 255.255.255.0
ifconfig eth1 100.100.1.1 netmask 255.255.255.0
route add -net 20.20.1.0 netmask 255.255.255.0 gw 100.100.1.2 dev eth1

host:~/lab2$ cat router2.startup
ifconfig eth0 20.20.1.1 netmask 255.255.255.0
ifconfig eth1 100.100.1.2 netmask 255.255.255.0
route add -net 10.10.1.0 netmask 255.255.255.0 gw 100.100.1.1 dev eth1
```

La tabella di routing di `router1` è predisposta in modo che le richieste di raggiungere la rete 20.20.1.0 passino dalla `eth1` e siano processati da 100.100.1.2. Quella di `router2` in modo che le richieste per la rete 10.10.1.0, passanti per la `eth1`, vengano processate dal gateway 100.100.1.1. Se ci fossero stati ulteriori router, come descritto per esempio in una esperienza precedente di configurazione manuale delle tabelle di routing nel simulatore NetEmul, ci sarebbe stato bisogno di aggiungere, nella riga del comando `route`, il parametro `metric` seguito dal numero di hop.

Avviato il laboratorio (`lstart`) ora, per esempio, da `pc1` della rete 10.10.1.0 si può raggiungere `pc3` della rete 20.20.1.0:

```
pc1:~# ping 20.20.1.2
PING 20.20.1.2 (20.20.1.2) 56(84) bytes of data.
64 bytes from 20.20.1.2: icmp_seq=1 ttl=62 time=21.1 ms
64 bytes from 20.20.1.2: icmp_seq=2 ttl=62 time=1.06 ms
64 bytes from 20.20.1.2: icmp_seq=3 ttl=62 time=0.505 ms
^C
--- 20.20.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2016ms
rtt min/avg/max/mdev = 0.505/7.562/21.115/9.586 ms
pc1:~#
```

2.5 Comunicazioni fra host e macchine virtuali

In ogni macchina virtuale esiste la directory `/hosthome`, in lettura e scrittura, che contiene la home

dell'utente che ha lanciato la macchina virtuale/il laboratorio e quindi sia l'host che la macchina virtuale possono scrivere, rispettivamente, nella directory `/home/tux` (per l'utente tux) e nella `/hosthome` nella macchina virtuale e condividerne il contenuto.

Attraverso il computer host, collegato ad Internet, è possibile fare in modo di collegare la rete virtuale con Internet

```
host:~$ vstart pc1 --eth0=tap,192.168.232.1,192.168.232.2
...
******** Starting Internet connected virtual hub ********
   192.168.232.1 (host side) - 192.168.232.2 (guest side)
********     (root privileges are required)      ********
...
Setting up tunnel...                          done.
Bringing up nk_tap_tux...                     done.
Setting permissions for /dev/net/tun...       done.
Enabling IP forwarding...                     done.
Enabling masquerading...                      done.
Opening firewall for tunnel...                done.
************** Abandoning root privileges ***************
```

Per attivare il `tap` è necessario inserire la password (sono necessari i diritti di root).

- Il dominio speciale di collisione `tap` consente di *attaccarsi* alla rete dell'host
- Il primo indirizzo (192.168.232.1) è l'indirizzo che nell'host consente di collegarsi ad Internet e che deve appartenere ad un range di indirizzi diverso da uno preesistente nella rete reale:

```
host:~$ ifconfig
eth0      ...
          indirizzo inet:192.168.1.100  Bcast:192.168.1.255
Maschera:255.255.255.0
...
nk_tap_tux Link encap:Ethernet  HWaddr 3e:54:71:3f:c4:ab
          indirizzo inet:192.168.232.1  Bcast:192.168.232.255
Maschera:255.255.255.0
...
```

Nell'host viene generato un nuovo device (`nk_tap_`*`nomeutente`*) che naturalmente deve avere indirizzo diverso e appartenere ad una rete diversa rispetto alla `eth0` dell'host stesso.

- Il secondo indirizzo (192.168.232.2) è l'indirizzo della `eth0` della macchina virtuale `pc1` che deve essere un indirizzo della stessa sotto-rete del primo.

Il nome `tap` tradotto letteralmente dall'inglese significa *rubinetto* ma è anche l'acronimo di Test Access Port. Nelle reti informatiche è un dispositivo hardware che si inserisce in una rete per monitorarne il traffico (il rubinetto rappresenta in senso figurato le funzionalità del dispositivo). Ne esiste un tipo chiamato V-Line TAP utilizzato quando ci sono dispositivi che necessitano di essere in-line in una rete per poter assolvere ai propri compiti (`http://www.networkcritical.com`). Nelle macchine virtuali di Netkit la funzionalità è la stessa di questo ultimo tipo da cui il nome del dominio di collisione (qui è un dispositivo virtuale).

Dalla macchina `pc1` configurata già con l'indirizzo specificato ora è possibile accedere ad Internet:

```
pc1:~# ping 173.194.35.191
PING 173.194.35.191 (173.194.35.191) 56(84) bytes of data.
```

```
64 bytes from 173.194.35.191: icmp_seq=1 ttl=53 time=56.3 ms
64 bytes from 173.194.35.191: icmp_seq=2 ttl=53 time=52.1 ms
64 bytes from 173.194.35.191: icmp_seq=3 ttl=53 time=52.6 ms
^C
--- 173.194.35.191 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2017ms
rtt min/avg/max/mdev = 52.194/53.731/56.395/1.909 ms
pc1:~#
```

L'output del ping verso un indirizzo di `www.google.it` ne mostra la raggiungibilità. Per poter utilizzare direttamente il nome di dominio è necessario definire i nameserver per la risoluzione dei nomi nella macchina `pc1`:

```
pc1:~# cat /etc/resolv.conf
nameserver 208.67.222.222
nameserver 208.67.220.220
```

Gli indirizzi fanno riferimento ad OpenDNS. Ora si può fare il `ping` direttamente a `www.google.it`.

Si può configurare un laboratorio con una rete virtuale le cui macchine accedono a Internet:

```
host:~/lab3$ cat lab.conf
pc1[0]=tap,192.168.232.1,192.168.232.2
pc2[0]=tap,192.168.232.1,192.168.232.3

LAB_AUTHOR="Tux"
LAB_DESCRIPTION="Rete collegata ad Internet via host"
```

in questo caso non sono necessari i file `.startup` per la configurazione delle schede di rete delle due macchine perché già definite dalle definizioni comprese nel `lab.conf`. Si può invece inserire il file `resolv.conf` contenete le righe prima specificate, in `pc1/etc` e `pc2/etc` e quando viene generato il filesystem delle macchine virtuali il file viene ricopiato nella directory `/etc` delle macchine.

Quando viene fermata la macchina virtuale o il laboratorio (`lhalt`) è necessario togliere il dispositivo `tap` che non viene eliminato in automatico (anche in questo caso sono necessari i diritti di root). È necessario inserire il comando:

```
host:~/lab3$ vclean -T
************* Removing tap configurations ************
   This will affect tap configurations for user tux.
******** This operation requires root privileges ********
Running ==> /opt/netkit/bin/manage_tuntap stop
tux's password:
Closing firewall...                       done.
Disabling masquerading...                 done.
Disabling IP forwarding...                done.
Bringing down tap devices and tunnels:
        nk_tap_tux...                     done.
        nk_tap_tux tunnel...           done.
Done.
Resetting permissions for /dev/net/tun... done.
************** Abandoning root privileges ***************
```

2.6 Rete Client/Server

Fra i servizi installati in ogni macchina virtuale è presente il web server apache2. Si possono quindi emulare reti C/S con server che gestiscono servizi vari.

Intanto si possono configurare, per esempio, tre computer collegati in rete:

```
host:~/lab4$ cat lab.conf
pc1[0]=hub0
pc2[0]=hub0
pc3[0]=hub0

LAB_AUTHOR="Tux"
LAB_DESCRIPTION="Rete locale con web server"
```

Se si sceglie la macchina `pc1` come quella in cui girerà il server il file con i comandi di avvio della macchina virtuale prevederà anche l'avvio del server. Per le altre macchine i file saranno i soliti con la sola configurazione delle interfacce di rete:

```
host:~/lab4$ cat pc1.startup
ifconfig eth0 10.10.1.1 netmask 255.255.255.0
/etc/init.d/apache2 start
host:~/lab4$ cat pc2.startup
ifconfig eth0 10.10.1.2 netmask 255.255.255.0
host:~/lab4$ cat pc3.startup
ifconfig eth0 10.10.1.3 netmask 255.255.255.0
```

Avviato il laboratorio ora dalle altre macchine ci si può collegare al server in esecuzione nella macchina 10.10.1.1:

```
pc2:~# links http://10.10.1.1
```

viene visualizzata la pagina di default con la scritta *It works!* Ad indicare il corretto funzionamento del collegamento e di apache.

Le pagine web del sito che si vuole testare vanno inserite, come impostazione di default del server Apache, nella directory `/var/www/` di `pc1` e permarranno nell'HD della macchina virtuale (il file `pc1.disk` della directory dell'host da cui sono state avviate le macchine virtuali).

Nel sistema presente nelle macchine virtuali non sono installati né il server mysql né php, ma essendo una installazione Linux normale, se serve, può essere installato qualsiasi pacchetto e l'installazione permarrà per i prossimi avvii. Per installare pacchetti dai repository, naturalmente, è necessario che la macchina sia collegata ad Internet (2.5).

2.7 Accessi controllati da firewall

In questa esperienza ci si occuperà di configurare un firewall sulla macchina virtuale che esegue il web server in modo da consentirne gli accessi solo alle macchine appartenenti ad una determinata zona.

Nelle macchine virtuali è installato il software Shorewall per l'impostazione delle catene da passare al kernel. Per questa esperienza si utilizzerà una configurazione simile (è stata tolta la macchina `pc4` per semplificare) a quella già esaminata in 2.4 con alcune modifiche.

```
host:~/lab5$ cat lab.conf
pc1[0]=A
```

```
pc1[mem]=256
pc2[0]=A

router1[0]=A
router1[1]=C

router2[1]=C
router2[0]=B

pc3[0]=B

LAB_AUTHOR="Tux"
LAB_DESCRIPTION="Configurazione firewall per accesso al web server"
```

Nel file di configurazione alla macchina `pc1` nella quale girerà il server vengono assegnati 256M di memoria centrale (seconda riga) in modo da poter permettere la compilazione delle regole di firewalling. Per default (come riportato dal file `netkit.conf`) la memoria assegnata ad ogni macchina virtuale è 32M.

Per la configurazione iniziale delle macchine:

```
host:~/lab5$ cat pc1.startup
ifconfig eth0 10.10.1.2 netmask 255.255.255.0
route add default gw 10.10.1.1
/etc/init.d/apache2 start

host:~/lab5$ cat pc2.startup
ifconfig eth0 10.10.1.3 netmask 255.255.255.0
route add default gw 10.10.1.1

host:~/lab5$ cat pc3.startup
ifconfig eth0 20.20.1.2 netmask 255.255.255.0
route add default gw 20.20.1.1

host:~/lab5$ cat router1.startup
ifconfig eth0 10.10.1.1 netmask 255.255.255.0
ifconfig eth1 100.100.1.1 netmask 255.255.255.0
route add -net 20.20.1.0 netmask 255.255.255.0 gw 100.100.1.2 dev eth1

host:~/lab5$ cat router2.startup
ifconfig eth0 20.20.1.1 netmask 255.255.255.0
ifconfig eth1 100.100.1.2 netmask 255.255.255.0
route add -net 10.10.1.0 netmask 255.255.255.0 gw 100.100.1.1 dev eth1
```

Rispetto alla configurazione di 2.4 nella macchina `pc1`, ora, si avvia il web server.

Si può verificare che dalle macchine `pc2` (10.10.1.3) e `pc3` (20.20.1.2) si può accedere al server web e navigare fra le pagine HTML con il comando:

```
links http://10.10.1.2
```

Ora si configurerà il firewall della macchina `pc1` in modo che al server possano avere accesso le macchine della rete 10.10.1.0 ma non le macchine della rete 20.20.1.0

Intanto occorre editare dalla `/usr/share/doc/shorewall-common/default-config/` alcuni file, aggiungere le configurazioni che servono e salvare le modifiche in `/etc/shorewall`:

```
pc1:/etc/shorewall# cat zones
```

```
...
#############################################################################
#ZONE   TYPE            OPTIONS         IN                      OUT
#                                       OPTIONS                 OPTIONS
fw      firewall

net1    ipv4
net2    ipv4

#LAST LINE - ADD YOUR ENTRIES ABOVE THIS ONE - DO NOT REMOVE
```

Oltre la zona `fw`, gestita per default da Shorewall e che rappresenta la macchina su cui si installano le regole di firewalling, vengono definite due zone, in cui saranno definite regole diverse, chiamate `net1` e `net2`.

```
pc1:/etc/shorewall# cat hosts
...
#############################################################################
#ZONE   HOST(S)                                 OPTIONS

net1    eth0:10.10.1.0/24
net2    eth0:20.20.1.0/24

#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS LINE -- DO NOT REMOVE
```

Il file `hosts` è necessario perché le due zone definite utilizzano la stessa interfaccia (`eth0`). La notazione per identificare la rete (es: 10.10.1.0/24) indica l'indirizzo di rete/quantità di bit utilizzati per l'indirizzo di rete (equivale a netmask 255.255.255.0).

```
pc1:/etc/shorewall# cat interfaces
...
#############################################################################
#ZONE   INTERFACE       BROADCAST       OPTIONS

-       eth0

#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

Il file `interfaces` indica che tutte le zone (il trattino nella colonna `ZONE`) comunicano attraverso l'unica interfaccia `eth0`.

A questo punto si possono definire le politiche di accesso:

```
pc1:/etc/shorewall# cat policy
...
#############################################################################
#SOURCE         DEST            POLICY          LOG             LIMIT:BURST
#                                               LEVEL

all             all             ACCEPT

#LAST LINE -- DO NOT REMOVE
```

Si permette il traffico in qualunque direzione (`all` come `fw` di prima è un nome predefinito da Shorewall e indica *tutti*). Si adotta una politica generale di tipo *permissivo*: tutto ciò che non viene espressamente proibito è permesso.

```
pc1:/etc/shorewall# cat rules
```

```
...
################################################################################
##########################################
#ACTION         SOURCE          DEST            PROTO   DEST    SOURCE
ORIGINAL        RATE            USER/   MARK
#                                                       PORT    PORT(S)
DEST            LIMIT           GROUP
#SECTION ESTABLISHED
#SECTION RELATED
SECTION NEW

HTTP/REJECT     net2            fw

#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

I pacchetti che utilizzano il protocollo HTTP, provenienti dalla rete 20.20.1.0 e destinati a `pc1` (diretti al server web), sono respinti.

Adesso si può avviare il servizio nella `pc1`:

```
pc1:~# shorewall start
Compiling...
Initializing...
Determining Zones...
...
Starting Shorewall....
Initializing...
...
Creating action chain Drop
Creating action chain Reject
Creating action chain dropBcast
Creating action chain dropInvalid
Creating action chain dropNotSyn
Applying Policies...
Activating Rules...
done.
pc1:~#
```

Dalla macchina `pc3` (20.20.1.2) è possibile effettuare il ping verso la 10.10.1.2 che risponde ma se si prova, dalla stessa macchina, ad accedere al server web si riceve un messaggio di `Connection refused`.

```
pc3:~# telnet 10.10.1.2 80
Trying 10.10.1.2...
telnet: Unable to connect to remote host: Connection refused
pc3:~#
```