Facultat d'Informàtica de Barcelona - Sistemes de Transport de Dades (STD)

Comandes bàsiques per a la configuració del nivell IP en una xarxa de màquines UNIX

Autor: Llorenç Cerdà Alabern

PARTE 1

1. Creació d'un *floppy* amb Trinux

Per fer aquesta pràctica hi ha que botar les màquines amb UNIX en comptes de DOS. Per això crearem dos discs amb Trinux (una versió de Linux que cap en un *floppy* de 1.44 MB) i botarem el PC des del *floppy*. Per crear els *floppy* amb Trinux::

- Botar el PC amb DOS des del servidor Novell i entrar com usuari STD,
- canviar al directori TRINUX del disc T:,
- posar el floppy formatejat en la unitat A:,
- executar rawrite. Els noms de les imatges a copiar als floppy son boot81.raw i pkg81.raw.

2. La interfície loopback

La primera interfície que hi ha que activar al configurar el nivell IP és la interfície loopback. Aquesta interfície és una mena de curtcircuit, és a dir, els datagrames que s'envien en aquesta interfície no abandonen mai la màquina, sinó que retornen immediatament al nivell IP que els envia. El nom estàndard d'aquesta interfície és lo, i la seva adreça és 127.0.0.1.

3. Comandes bàsiques

3.1. Comanda `ifconfig`

Descobreix una interfície al nivell de xarxa del kernel del SO. Això implica: assignar una adreça IP a la interfície i activar-la. Les maneres típiques d'invocar aquesta comanda són:

`ifconfig` `interfície adreça_IP [`**`netmask`** `màscara]`[1]

Activa una interfície i l'hi assigna una adreça. Si no es dóna la màscara, s'assigna la que correspon segons la classe de l'adreça IP. El nom estàndard per designar una tarja ethernet en UNIX és ethi, on i val 0 per la primera tarja, 1 per la segona etc. Per exemple, per activar el loopback i una tarja ethernet:

```
trinux> ifconfig lo 127.0.0.1
trinux> ifconfig eth0 192.176.6.1 netmask 255.255.255.0
```

Exemple 1: Configuració de la interfície `lo` i `eth0`.

ifconfig

Mostra les interfícies conegudes pel nivell de xarxa i la seva configuració. Per exemple:

```
trinux> ifconfig
lo    Link encap:Local Loopback
      inet addr:127.0.0.1 Bcast:127.255.255.255 Mask:255.0.0.0
      UP BROADCAST LOOPBACK RUNNING MTU:3584 Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0
      TX packets:0 errors:0 dropped:0 overruns:0
eth0  Link encap:Ethernet HWaddr 00:10:5A:F7:E3:22
      inet addr:192.176.6.1 Bcast:192.176.6.255 Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0
      TX packets:0 errors:0 dropped:0 overruns:0
      Interrupt:5 Base address:0x6800
```

Exemple 2: Llistat de les interfícies configurades.

3.2. Comanda route

Permet afegir/esborrar entrades a la taula d'encaminament i mostrar el seu contingut. Les invocacions típiques són[1]:

route add|**del** -**net**|-**host** destí [**netmask** màscara] [**gw** gateway] [**dev** intf.]

Afegeix una entrada. Per exemple::

```
trinux> route add -net 192.176.6.0
```

Exemple 3: Adició d'una entrada a la taula d'encaminament.

Fixeu-vos que en l'exemple anterior no es dóna la màscara ni la interfície. Aquesta informació no és necesaria si ja s'ha configurat una de les interfícies de la màquina que estigui en la mateixa xarxa (per exemple, la interfície eth0 en l'exemple 1).

route [**-n**][2]

Mostra el contingut de la taula d'encaminament. Amb l'opció `-n` mostra les adreces IP en forma numèrica. Per exemple:

---

[1] Conveni: les paraules clau estan en negreta i els paràmetres que dóna l'usuari no.

[2] En alguns UNIX `route` només permet afegir entrades a la taula d'encaminament. Per llistar la taula hi ha que executar `netstat -nr`

```
trinux> route -n
Kernel IP routing table
Destination   Gateway   Genmask    Flags   Metric  Ref  Use  Iface
192.176.6.0     0.0.0.0  255.255.255.0   U       0       0    0     eth0
127.0.0.0       0.0.0.0   255.0.0.0      U       0       0    0     lo
```

Exemple 4: Llistat de la taula d'encaminament.

3.3. Comanda arp

La comanda arp permet veure i modificar manualment la taula que manté el mòdul ARP (Address Ressolution Protocol). En aquesta taula hi ha la correspondència entre les adreces IP i les adreces hardware. Les invocacions típiques són:

`arp -a`

Mostra la taula ARP.

`arp -s` `adreça_IP adreça_hw`

Assigna l'adreça *hardware* `adreça_hw` a l'adreça IP `adreça_IP`.

`arp -d` `adreça_IP`

Esborra l'entrada adreça_IP de la taula.

3.4. Comanda ping

La comanda ping és una mena de sonar que permet verificar si una certa interfície està a l'abast del nivell de xarxa, i per mesurar el retard d'anada i tornada que hi ha fins el destí. Ping envia periòdicament un paquet a l'adreça que es dóna com a paràmetre que provoca la resposta del destí. Per parar el ping hi ha que fer un CONTROL-C. Per exemple, per saber si podem accedir a la màquina 192.176.6.114:

```
trinux> ping 192.176.6.114
PING 192.176.6.114 (192.176.6.114): 56 data bytes
64 bytes from 192.176.6.114: icmp_seq=0 ttl=255 time=0.6 ms
64 bytes from 192.176.6.114: icmp_seq=1 ttl=255 time=0.6 ms
64 bytes from 192.176.6.114: icmp_seq=2 ttl=255 time=0.6 ms
^C
--- 192.176.6.114 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.6/0.6/0.6 ms
```

Exemple 5: Ping a una màquina remota.

També podem fer un *ping* a una interfície de la mateixa màquina, per exemple el *loopback*:

```
trinux> ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
```

```
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.1 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.0 ms
^C
--- 127.0.0.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.0/0.0/0.1 ms
```

Exemple 6: Ping a la interfície de loopback.

També es pot fer un ping a l'adreça de broadcast (la que té el camp de host amb tots els bits a 1) d'una interfície per saber quines altres màquines hi ha connectades a la mateixa xarxa:

```
trinux> ping 192.176.6.255
PING 192.176.6.255 (192.176.6.255): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.1 ms
64 bytes from 192.176.6.7: icmp_seq=0 ttl=255 time=0.6 ms (DUP!)
64 bytes from 192.176.6.3: icmp_seq=0 ttl=255 time=0.7 ms (DUP!)
...
^C
--- 192.176.6.255 ping statistics ---
1 packets transmitted, 1 packets received, +12 duplicates, 0% packet loss
round-trip min/avg/max = 0.1/1.9/7.0 ms
```

Exemple 7: Ping broadcast.

Els DUPs indiquen que s'ha rebut més d'un paquet de resposta a un mateix *ping*.

4. Els fitxer /etc/hosts

Per no haver d'usar sempre les adreces IP, una màquina UNIX permet assignar etiquetes a les adreces de lesinterfícies amb el fitxer `/etc/hosts`. Per exemple, el contingut d'aquests fitxer podria ser:

```
trinux> cat /etc/hosts
127.0.0.1 localhost
192.176.6.2 linux
192.176.6.201 pc1
```

Exemple 8: Contingut del fitxer /etc/hosts.

Amb el fitxer anterior, els exemples 1 i 5 es podrien haver fet amb les següents comandes:

```
trinux> ifconfig lo localhost
trinux> ifconfig eth0 pc1 netmask 255.255.255.0
trinux> ping linux
```

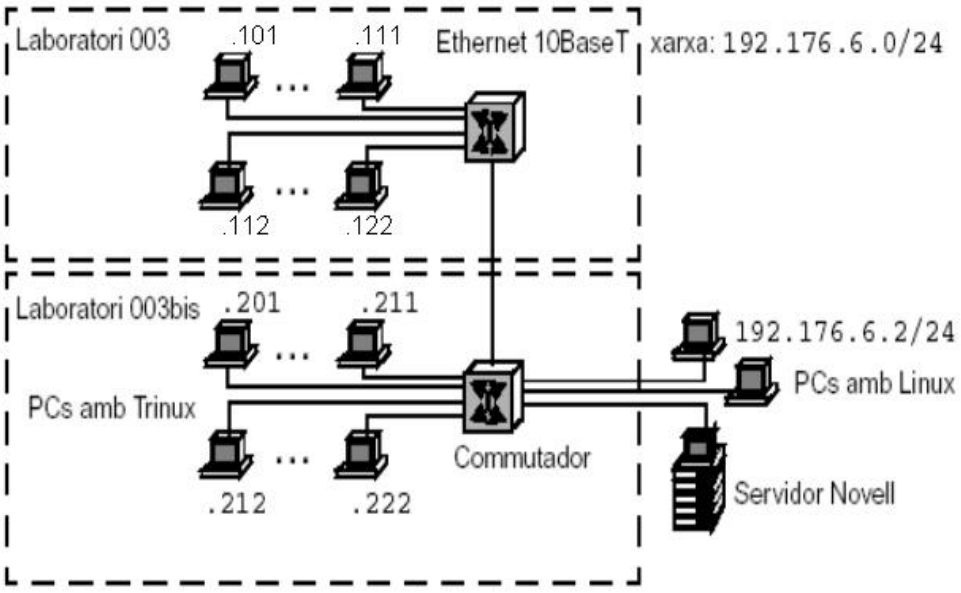
Exemple 9: Ús d'etiquetes en comptes d'adreces IP.

5. Realització de la pràctica

L'objectiu de la pràctica és familiaritzar-se amb les comandes anteriors. Per això configurarem la xarxa del laboratori tal com es mostra en la figura 1.

És a dir, hi haurà una xarxa IP 192.176.6.0.

Per assignar les adreces IP als PC farem servir el següent conveni: 192.176.6.<número PC+200>. Per exemple, si el número que hi ha en l'etiqueta del PC és 3, el PC tindrà com adreça 192.176.6.203.



Per fer la pràctica seguiu el següents passos:

1. Botar el PC amb el disc de trinux.
2. Llistar les interfícies (igual que en l'exemple 2) per comprovar que no hi cap interfície configurada. Llistar la taula d'encaminament (exemple 4) per comprovar que la taula està buida. Llistar la taula ARP (executant arp -a) per comprovar que també està buida.
3. Assignar l'adreça 127.0.0.1 a la interfície de loopback, i llistar les interfícies per comprovar que s'ha afegit.
4. Provar de fer un *ping* al `127.0.0.1`.

5. Assignar l'adreça IP a la tarja *ethernet* fent servir el conveni explicat anteriorment. Comprovar que la interfície s'ha activat llistant les interfícies.

6. Proveu de fer un *ping* a la tarja *ethernet* per assegurar-vos de que és accessible.

7. Crear el fitxer `/etc/hosts` amb l'entrada `linux 192.176.6.2`. Ho podeu fer amb l'editor `vi` o simplement executant: `echo 192.176.6.2 linux > /etc/hosts`.

8. Provar de fer `ping linux` per comprovar que la màquina és accessible. Llistar la taula ARP per comprovar que l'adreça *hardware* de `192.176.6.2` s'ha afegit a la taula.

9. Fer un *ping broadcast* per descobrir quines altres màquines hi ha connectades a la xarxa. Llistar la taula ARP per veure les adreces *hardware* d'aquestes màquines.

10. Suposa que en la mateixa xarxa *ethernet* hi ha dos PC amb adreces que tenen diferents prefixos de xarxa. Per exemple: `192.176.6.a` i `172.16.0.b` (on `a` i `b` és el número assignat al PCs segons el criteri explicat anteriorment). Quines entrades s'haurien de posar en les taules d'encaminament dels dos PC perquè es poguessin comunicar? Provar que la vostra solució funciona fent servir dos PCs del laboratori (ajudant-vos amb el *ping*).

PARTE 2

1. Introducción.

Tcpdump es una herramienta de red elaborada en la Universidad de California (Lawrence Berkeley National Laboratory, LBNL) para la monitorización de redes. Permite monitorizar tanto una interficie Ethernet como una comunicación punto a punto (e.g. puerto serie).

Tcpdump es un programa de red que pone la tarjeta de red en *modo promiscuo:* recibe todos los paquetes/tramas que pasan por esa interficie independientemente de que la dirección MAC destino sea la de la tarjeta (normalmente la tarjeta solo captura aquellas tramas que contienen su dirección destino). Eso significa que tcpdump captura todas las tramas que pasan por la interficie a la que esta conectada. Tcpdump usa un software llamado BPF (BSD Packet Filter) que le permite poner la tarjeta en modo promiscuo, de forma que cada vez que se transmite o se recibe una trama por el driver de la tarjeta de red, BPF recibe una copia de esa trama. De esta forma BPF puede procesar y filtrar los paquetes de red que el usuario considera más interesantes: el comando tcpdump permite filtrar por protocolos (e.g.; ip, tcp, udp, arp, ether, fddi, …), por interficie (e.g.; eth0, lo, …), por puerto TCP/UDP o dirección IP, por dirección (e.g.; origen, destino, ambos, …), etc.

Para ver el funcionamiento del comando tcpdump podeis usar el comando *man* de UNIX: *man tcpdump*.

El formato general de salida del tcpdump es el siguiente:

fecha src > dst: flags data-sqno ack window urgent options

donde:

- **fecha** nos da la hora en que se produjo el evento en formato **hora:minutos:segundos.microsegundos** (o milisegundos dependiendo de la resolución del reloj).

- **src y dst** son las direcciones IP y puertos TCP/UDP de las conexiones fuente y destino.

- **flags** son una combinación de los posibles flgas de un segmento/datagrama TCP/UDP: S (SYN), F (FIN), P (PUSH), R (RST) y '.' (no flags).
- **data-sqno** describe el número de secuencia de la porción de datos.
- **ack** es el número de secuencia del próximo byte que espera recibir el otro extremo TCP/UDP.
- **window** es el tamaño de la ventana que advierte el receptor al transmisor.
- urgent indica que hay datos urgentes en ese segmento/datagrama.
- **options** son las opciones TCP que suelen estar entre corchetes del tipo < >, por ejemplo el tamaño máximo del segmento (e.g. <mss 1024>)

Por ejemplo ejecutando el comando:

galatzo% tcpdump -i eth0

Observo todo lo que pasa por la tarjeta de red de mi PC. Una línea de salida de tcpdump sería:

17:58:58.470000 giralt.ac.upc.es.6000 > galatzo.ac.upc.es.4228: P 1191656817:1191656849(32) ack 2435973568 win 61320 (DF)

donde se indica que a las 17:58:58.470000 se recibio una trama con origen giralt (nombre de una máquina) y destino galatzo (hay una opción para que escriba las direcciones IP en vez de los nombres). Los puertos TCP usados son el 6000 en origen y el 4228 en destino. P indica que el flag PUSH del segmento TCP estaba activado. Giralt ha enviado 32 bytes de datos con números de secuencia comprendidos entre el 1191656817 y el 1191656849. Como la conexión estaba ya empezada cuando se monitorizo la conexión, los números de secuencia son absolutos. Si tcpdump monitoriza una conexión desde el inicio, los números de secuencia son relativos al número de secuencia inicial. Giralt reconoce haber recibido de galatzo el dato 2435973567 y espera que el próximo que envíe sea el 2435973568. Finalmente, giralt anuncia una ventana de 61320 y avisa que no se fragmenten los datagramas IP.

Si queremos ver el contenido del paquete en hexadecimal podemos ejecutar el comando: *tcpdump -x -s100 -i eth0* que nos indica que capturemos los primeros 100 bytes (incluidos cabeceras IP y TCP/UDP y excluidas las cabeceras Ethernet) del paquete. De esta forma también podemos observar valores de la cabecera IP (e.g. TTL, tipo, …).