

Administración de redes Linux

Iria Estévez Ayres & Andrés Marín López

Abril 2025

Resumen

Vamos a montar unos servidores dhcp y tftp para instalar una máquina cliente desde la red sin USB ni CD ni necesidad de ninguna configuración de la máquina cliente. El resto de la sesión será ahondar un poco en el filtrado de tráfico en linux con nftables.

1. Configurando un servidor para instalar Debian con configuración cero

Vamos a crear la infraestructura necesaria para que cualquier máquina en la red LAN que vamos a definir pueda instalarse Debian sin ninguna configuración previa. Lo único que vamos a necesitar es configurar su BIOS para que arranque desde la red (PXEboot). Para ello vamos a usar como base la guía Debian <https://wiki.debian.org/PXEBootInstall>.

1.1. Crear interfaces en red interna

En primer lugar paramos nuestra MV y en la configuración de la MV en el apartado de la red añadimos una segunda interfaz y seleccionamos “Internal network” (o “Red Interna”, en castellano). Esta será la interfaz a través de la cual nos conectaremos a la nueva máquina que vamos a instalar por red.

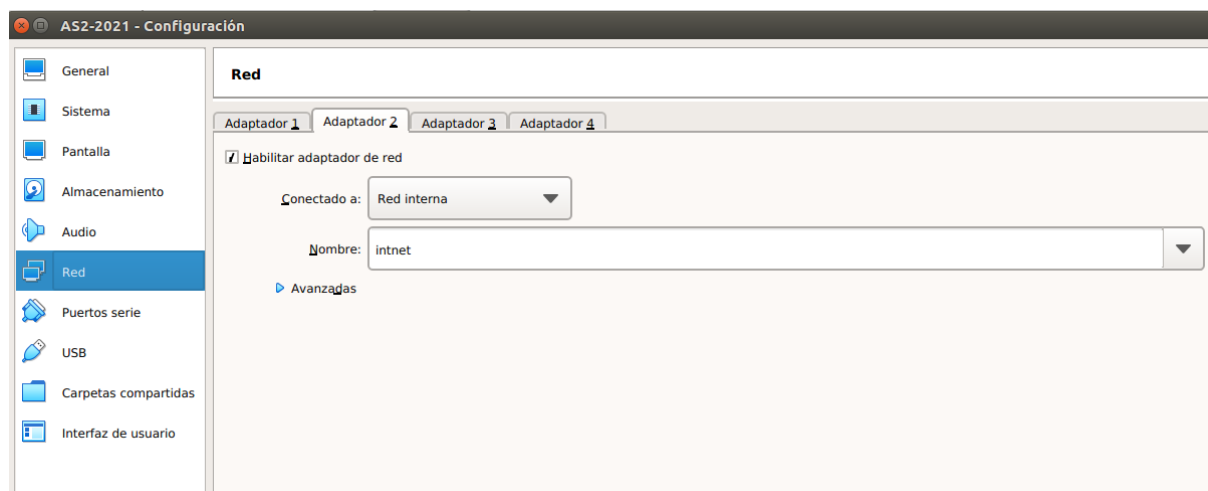


Figura 1: Configuración de red de nuestra MV

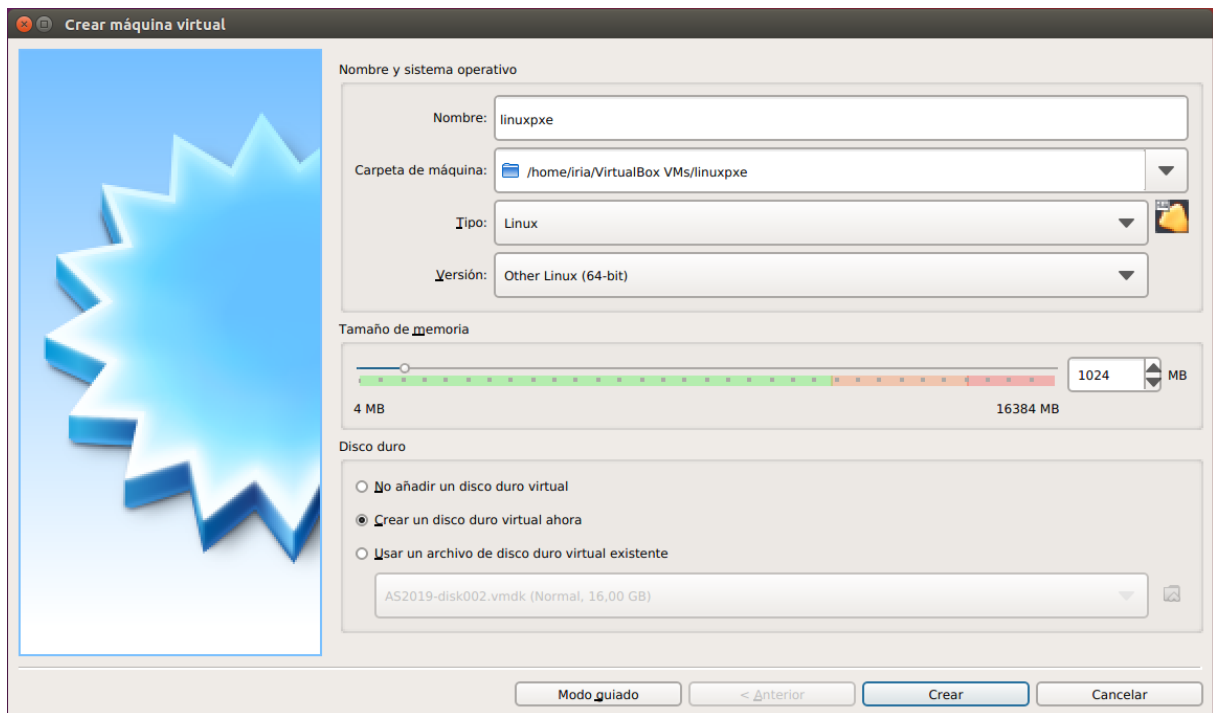


Figura 2: Creación de la MV linuxpxe. Paso 1

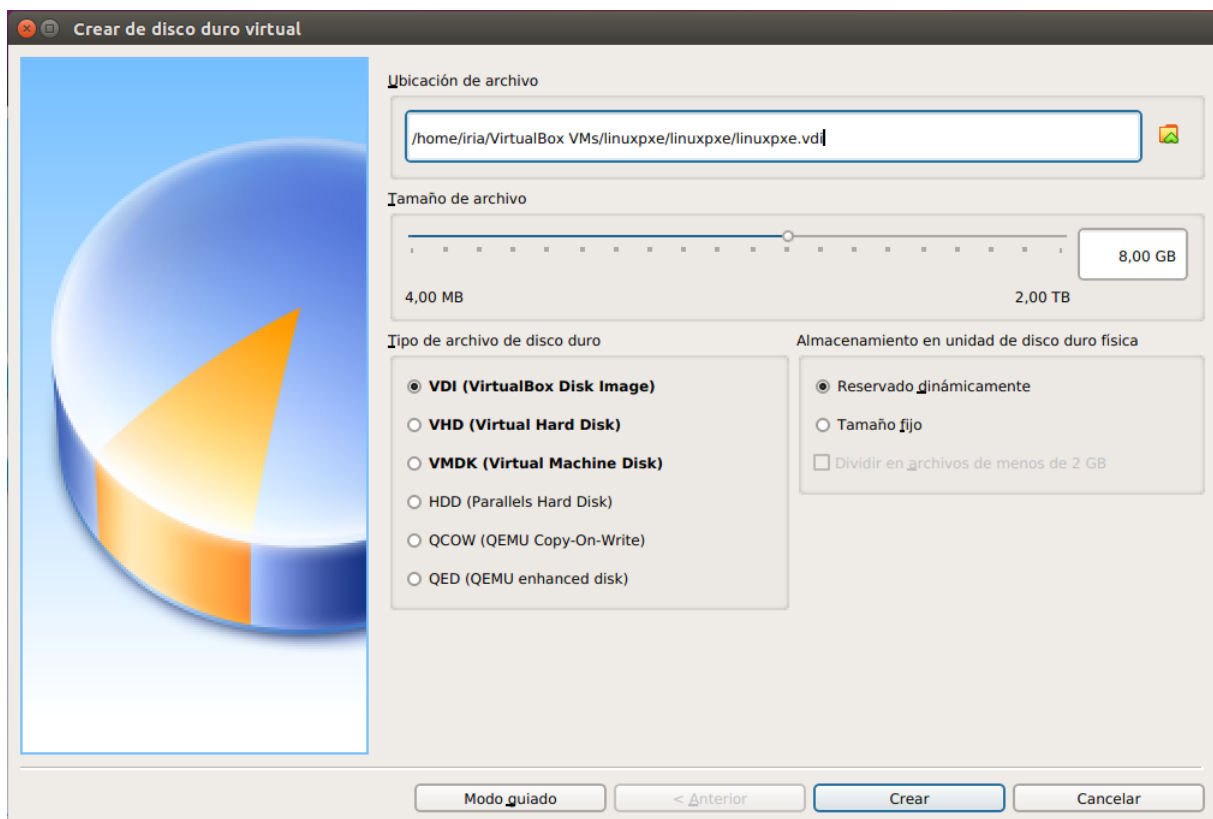


Figura 3: Creación de la MV linuxpxe. Paso 2

En segundo lugar vamos a crear la máquina cliente, a la que vamos a llamar linuxpxe. La creamos como una nueva máquina en VirtualBox, yendo al menú Máquina → Nueva). Si pulsáis en modo experto os saldrá el menú para indicar todas las opciones (como se puede ver en las figuras 2 y 3):

- Nombre: linuxpxe.
- Tipo: Linux.
- Versión: Other Linux (64-bits).
- Tamaño de memoria: 1 GB.
- Crear disco duro virtual ahora, con 8 Gigas es suficiente (si no tenemos espacio podemos crear el disco en un pendrive).

Lo más importante es configurarle una interfaz de red conectado a “Red Interna”, a la misma red que hemos conectado nuestra MV, como vemos en la figura 4. Esto es porque queremos configurar nuestra MV para que haga de router para linuxpxe.

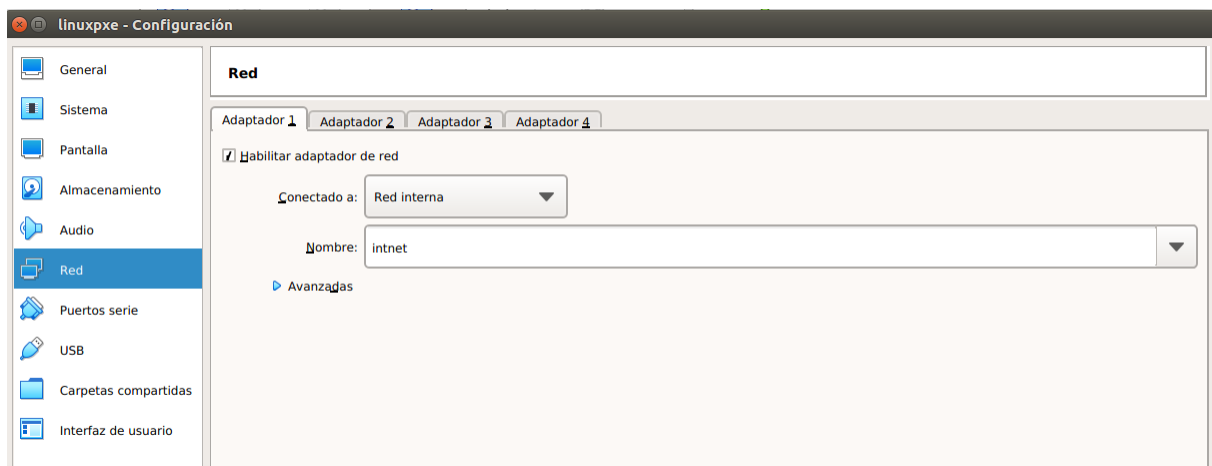


Figura 4: Configuración de red de linuxpxe

Ahora tenemos que configurar la nueva interfaz de la MV, convertir nuestra máquina en un router para linuxpxe, montar un servidor DHCP para configurar la interfaz de red de linuxpxe, y por último montar también un servidor TFTP para servir la imagen para que arranque linuxpxe.

1.2. Configuración estática de enp0s8 en nuestra máquina virtual

Vamos a elegir las IPs para la MV y linuxpxe. Por ejemplo 10.1.0.3 (para nuestra MV) y alguna IP de esa red para linuxpxe. Primero tengo que configurar una interfaz estática en nuestra MV con esa IP, para ello editamos `/etc/network/interfaces`:

```
iface enp0s8 inet static
    address 10.1.0.3
    netmask 255.255.255.0
```

Como no hemos dicho que la interfaz sea hotplug, no lo configura automáticamente NetworkManager y entonces lo tenemos que hacer nosotros manualmente. Utilizamos `ifup enp0s8` para subir y configurar la interfaz y `ifdown enp0s8` para bajarlo. Subimos y configuramos la interfaz:

```
sudo ifup enp0s8
```

Fijaos que si reinicio la máquina virtual, debo otra vez que a mano subir la nueva interfaz. Esto se soluciona haciendo que sea automático el subirla poniendo `auto enp0s8` antes de la descripción de la interfaz en el fichero `/etc/network/interfaces`.

1.3. Servidor dhcp

Instalamos el paquete `isc-dhcp-server` con `apt-get`.

```
$ sudo apt install isc-dhcp-server
```

Al terminar de instalarlo da un conjunto de errores que son producto de que no está configurado el servidor dhcp (fig 5).

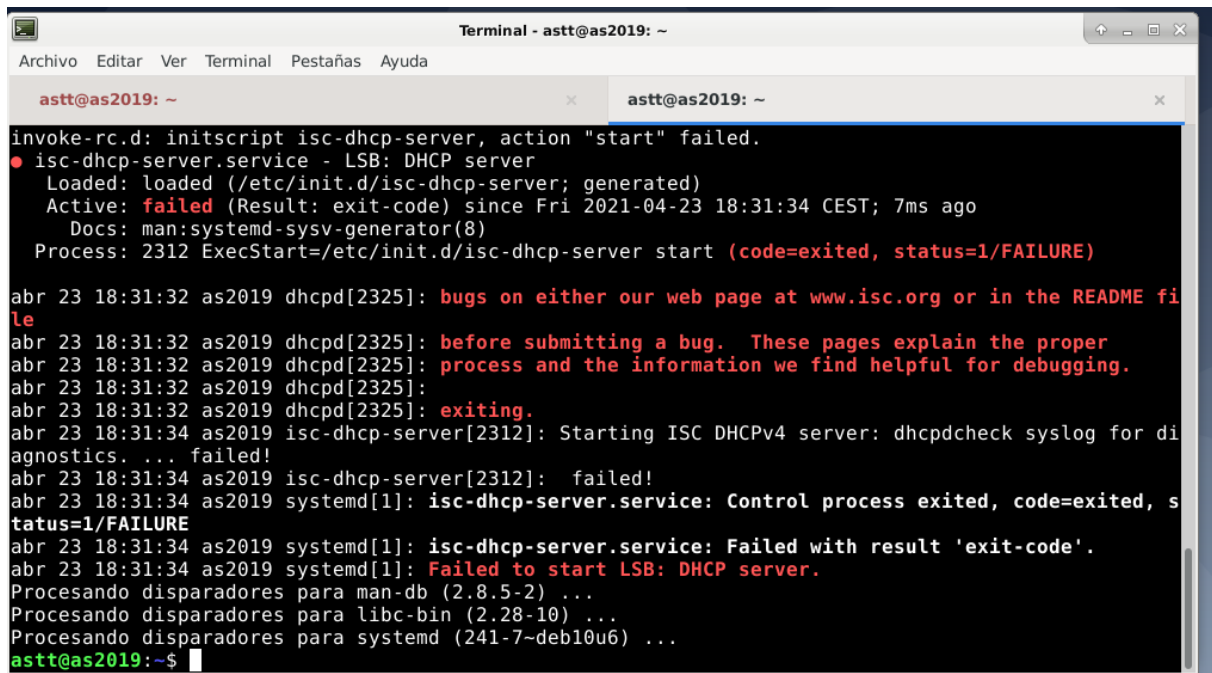


Figura 5: Errores al final de la instalación del dhcp

Vamos a configurarlo, indicando el rango estático para dhcp en (10.1.0.103-10.1.0.250). Si nadie se adelanta, linuxpxe recibirá la IP 10.1.0.103.

Vamos a configurar el servidor DHCP copiando el contenido siguiente en `/etc/dhcp/dhcpd.conf`

```
default-lease-time 600;
max-lease-time 7200;
allow booting;

subnet 10.1.0.0 netmask 255.255.255.0 {
    range 10.1.0.103 10.1.0.250;
    option routers 10.1.0.3;           # ip del router: de nuestra MV
    option domain-name-servers 8.8.8.8; # servidor dns abierto público
    option broadcast-address 10.1.0.255;
    filename "pxelinux.0";
    next-server 10.1.0.3;
}

group {
    next-server 10.1.0.3;           # ip del servidor tftp
    host tftpclient {
        filename "pxelinux.0";
    }
}
```

Después editamos `/etc/default/isc-dhcp-server` para contener:

```
DHCPDv4_CONF=/etc/dhcp/dhcpd.conf
INTERFACESv4="enp0s8"
```

Ya podemos arrancar el servidor dhcpd:

```
sudo systemctl start isc-dhcp-server
```

Arrancamos ahora el cliente linuxpxe, justo cuando está iniciando debemos indicar con F12 que queremos que nos salga el menú de selección de boot device (ver figura 6).

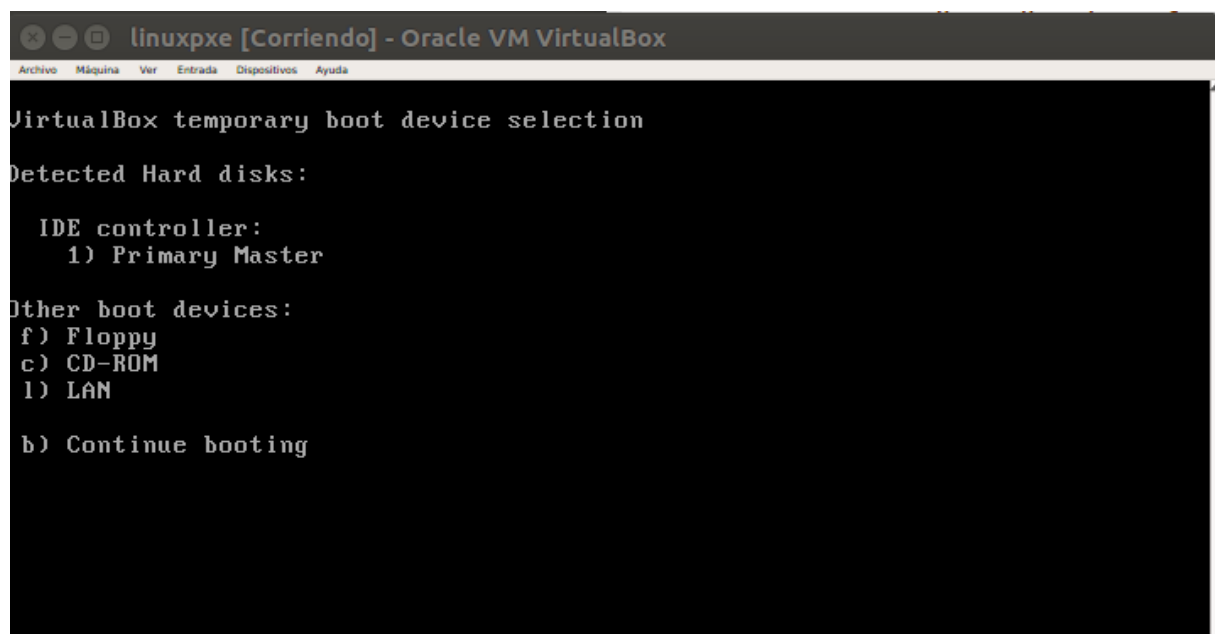


Figura 6: Menú selección boot device

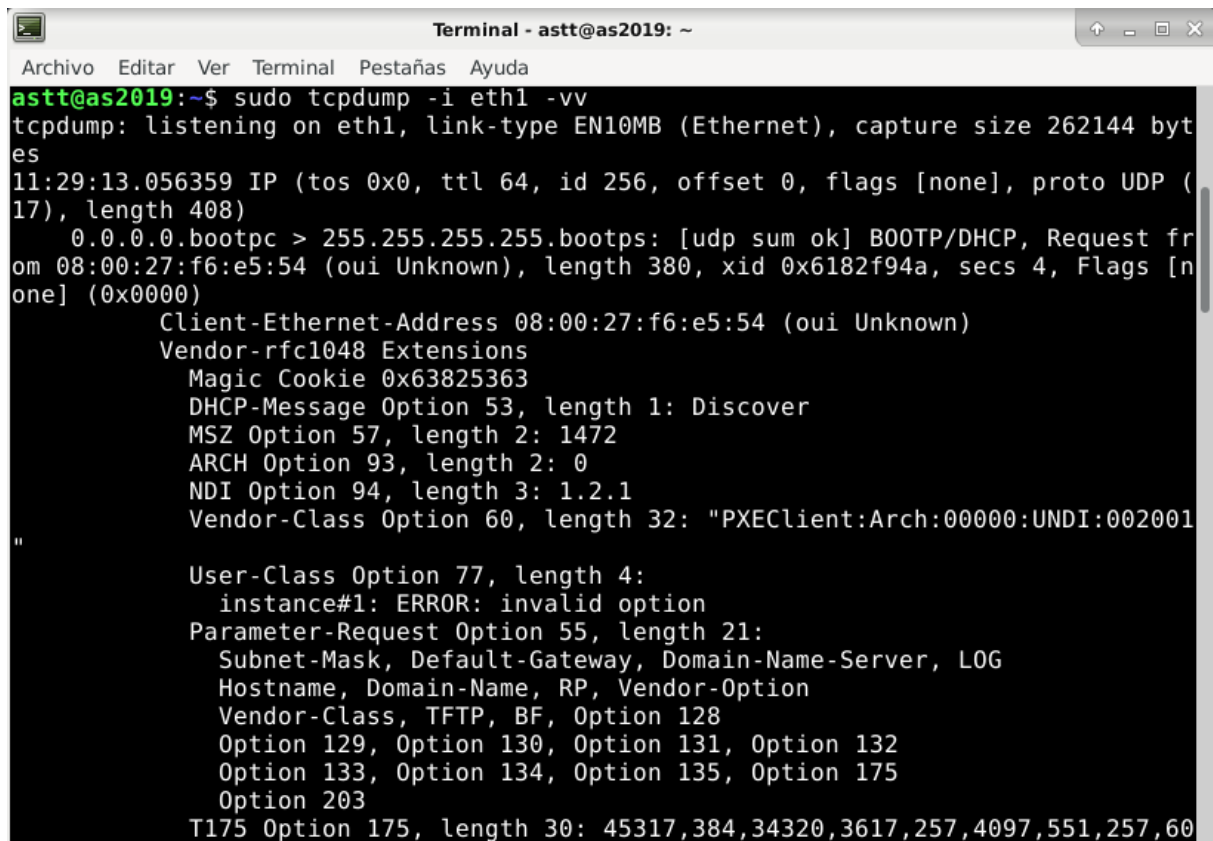
Indicamos que queremos que se inicie por la LAN, veremos como le asigna la dirección IP 10.1.0.103, pero que el sistema se para pues no ha podido bajar el fichero pxelinux.0 por TFTP (ver figura 7).

Figura 7: Configuración dhcp de linuxpxe

En nuestra MV podemos instalar tcpdump (podríamos también instalar wireshark), lo arrancamos y lo ponemos a escuchar en eth1.

```
$ sudo tcpdump -i eth1 -vv
```

Arrancamos otra vez linuxpxe (recordad que en el arranque tenemos que pulsar F12 para seleccionar que arranque desde la LAN). Se pueden ver las peticiones de DHCP, con las respuestas, así como la última petición del fichero, que nadie le sirve a linuxpxe. En la figura 8 puede verse una captura de la solicitud de configuración.



```
Terminal - astt@as2019: ~
Archivo  Editar  Ver  Terminal  Pestañas  Ayuda
astt@as2019:~$ sudo tcpdump -i eth1 -vv
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
11:29:13.056359 IP (tos 0x0, ttl 64, id 256, offset 0, flags [none], proto UDP (17), length 408)
  0.0.0.0.bootpc > 255.255.255.255.bootps: [udp sum ok] BOOTP/DHCP, Request from 08:00:27:f6:e5:54 (oui Unknown), length 380, xid 0x6182f94a, secs 4, Flags [none] (0x0000)
    Client-Ethernet-Address 08:00:27:f6:e5:54 (oui Unknown)
    Vendor-rfc1048 Extensions
      Magic Cookie 0x63825363
      DHCP-Message Option 53, length 1: Discover
      MSZ Option 57, length 2: 1472
      ARCH Option 93, length 2: 0
      NDI Option 94, length 3: 1.2.1
      Vendor-Class Option 60, length 32: "PXEClient:Arch:000000:UNDI:002001"
    User-Class Option 77, length 4:
      instance#1: ERROR: invalid option
    Parameter-Request Option 55, length 21:
      Subnet-Mask, Default-Gateway, Domain-Name-Server, LOG
      Hostname, Domain-Name, RP, Vendor-Option
      Vendor-Class, TFTP, BF, Option 128
      Option 129, Option 130, Option 131, Option 132
      Option 133, Option 134, Option 135, Option 175
      Option 203
      T175 Option 175, length 30: 45317,384,34320,3617,257,4097,551,257,60
```

Figura 8: Captura de solicitud de configuración de linuxpxe

1.4. Configuración del router

En este punto tenemos ya un servidor DHCP funcionando, pero para que el sistema reencamine los paquetes es necesario activar la opción en el kernel. Editamos el fichero `/etc/sysctl.conf` y descomentamos la línea que dice:

```
# net.ipv4.ip_forward=1
```

y ejecutamos el comando siguiente para aplicar el cambio.

```
sudo sysctl -p
```

Ahora vamos a configurar el routing en el kernel para utilizar `enp0s3` para paquetes salientes con NAT POSTROUTING, y que se utilice masquerading, es decir que antes de mandar un paquete de salida, se le cambie la IP por la pública del router. Y también para permitir los paquetes en la cadena FORWARD con entrada en `enp0s8` y salida en `enp0s3`, y los entrantes en `enp0s3` y salida en `enp0s8` si pertenecen a una conexión ya establecida. Para esto, usaremos `nft`, que substituye a `iptables` en las últimas versiones de Debian:

```
$ sudo nft add table ip nat
$ sudo nft add chain ip nat postrouting '{ type nat hook postrouting priority 100; }'
$ sudo nft add rule ip nat postrouting oifname "enp0s3" masquerade
$ sudo nft add table ip filter
$ sudo nft add chain ip filter forward '{ type filter hook forward priority 0; }'
$ sudo nft add rule ip filter forward iifname "enp0s3" oifname "enp0s8" \
  ct state { related, established } accept
$ sudo nft add rule ip filter forward iifname "enp0s8" oifname "enp0s3" accept
```

Si comprobamos la lista de reglas establecidas tendríamos:

```
$ sudo nft list ruleset
table ip nat {
    chain postrouting {
        type nat hook postrouting priority srcnat; policy accept;
        oifname "enp0s3" masquerade
    }
}
table ip filter {
    chain forward {
        type filter hook forward priority filter; policy accept;
        iifname "enp0s3" oifname "enp0s8" ct state { established, related } accept
        iifname "enp0s8" oifname "enp0s3" accept
    }
}
```

1.5. Servidor TFTP

A continuación nos instalamos el paquete `tftpd-hpa`, que es nuestro servidor de TFTP y, además, aprovechamos e instalaremos el instalador de Debian, que es la imagen que le mandaremos al cliente por TFTP, que está en el paquete `debian-installer-12-netboot-amd64`.

```
$ sudo apt install tftpd-hpa debian-installer-12-netboot-amd64
```

En el caso hipotético de no tener espacio en el disco, podríamos limpiar la caché de paquetes que hay en `/var/cache/apt/archives/`

Configuramos el servidor tftpd en el fichero `/etc/default/tftpd-hpa`

```
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/srv/tftp"
TFTP_ADDRESS="10.1.0.3:69"
TFTP_OPTIONS="--secure"
```

Y reiniciamos el servidor TFTP:

```
$ sudo /etc/init.d/tftpd-hpa restart
```

o

```
sudo systemctl restart tftpd-hpa
```

Podemos comprobar con un cliente TFTP si está funcionando nuestro servidor, para ello instalamos el paquete `tftp-hpa`.

```
$ sudo apt-get install tftp-hpa
```

Crearemos un fichero en el directorio en `/srv/tftp` y después lo bajaremos con el comando `get` de tftp.

```
$ cd /tmp
$ uname -a > test
$ sudo mv test /srv/tftp/
$ tftp 10.1.0.3
tftp> get test
tftp> quit
$ diff test /srv/tftp/test
$
```


En un servidor TFTP si quisiéramos dejar que el cliente cree ficheros en el mismo, debemos cambiar la configuración en el fichero `/etc/default/tftpd-hpa` de la última opción añadiendo `--create`:

```
TFTP_OPTIONS="--secure --create"
```

En esta práctica no lo hacemos pues no nos interesa que los clientes creen nada en el servidor.

1.6. Terminando de configurar el PXE (Preboot Execution Environment)

Ahora está prácticamente todo, pero aún falta la configuración para que el cliente se pueda bajar la imagen. Así, creamos el enlace simbólico. Podemos elegir la instalación gráfica o textual, elegimos la textual dado que la mandamos por la red y enlazamos el directorio entero:

```
$ sudo rm -r /srv/tftp
$ sudo ln -s /usr/lib/debian-installer/images/12/amd64/text /srv/tftp
$ ls -l /srv/tftp
lrwxrwxrwx 1 root root 46 abr 12 16:06 /srv/tftp -> \
  /usr/lib/debian-installer/images/12/amd64/text
```

Fijaos que dentro del directorio `/srv/tftp` está el fichero `pxelinux.0` que es el que va a pedir el cliente.

```
$ ls /srv/tftp
debian-installer  ldlinux.c32  pxelinux.0  pxelinux.cfg  version.info
```

Volvemos a arrancar el servicio de TFTP

```
$ sudo systemctl restart tftpd-hpa
```

Probamos a arrancar el cliente y esta vez debe aparecer el menu de arranque como muestra la figura 9.

1.7. Ampliación

Si añadimos la base de datos de respuestas de `debconf` en el `init`, podemos tener una instalación totalmente automatizada. Ver <https://wiki.debian.org/DebianInstaller/Preseed>. Esto que hemos realizado existe también como un proyecto en si mismo, el proyecto Fully Automatic Instalation <https://fai-project.org>.

2. Firewalling and NATting

2.1. Netfilter

El proyecto netfilter <https://www.netfilter.org/> es el metaproyecto donde se han realizado los distintos marcos de filtrado de tráfico y traducción entre IPs y puertos en el kernel de linux.

En los kernels 2.0.x se utilizaba `ipfwadm`¹ en los kernels 2.x se utilizaban las `ip-`

¹<https://web.archive.org/web/20080714230323/http://www.xos.nl/resources/ipfwadm/>

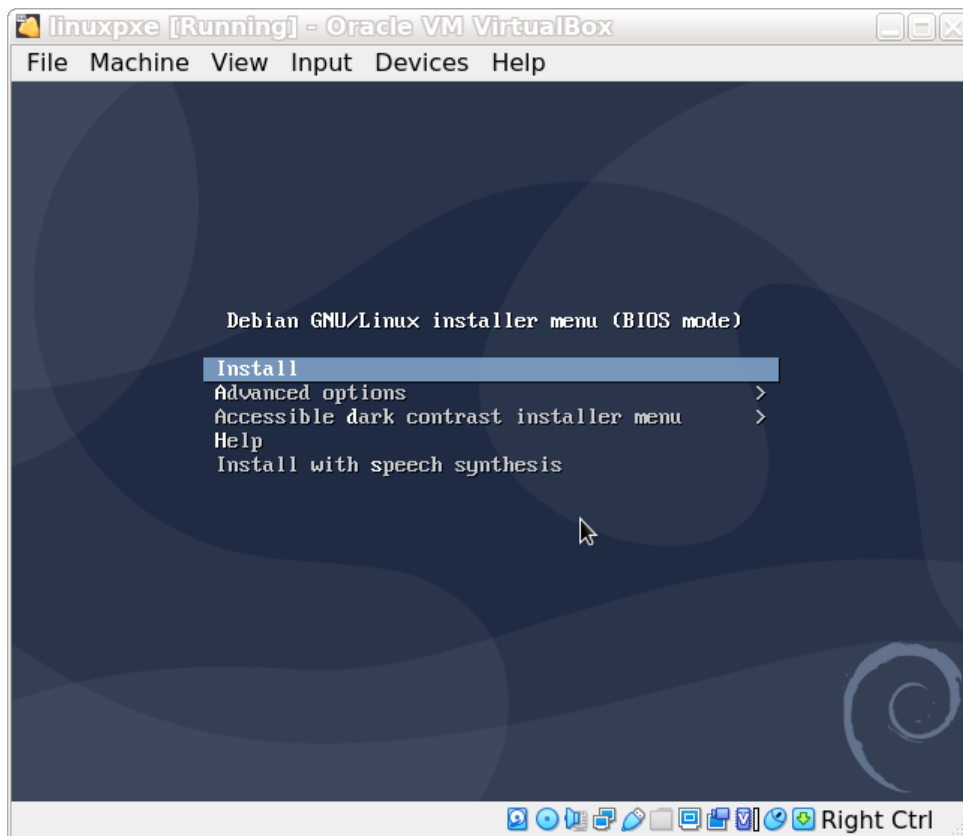


Figura 9: Linuxpxe carga su imagen por tftp

chains (ver <https://www.netfilter.org/ipchains>) y en los kernels 2.4.x las iptables (ver <https://www.netfilter.org/projects/iptables>). Desde los kernels 3.13 se ofrece la evolución nftables <https://www.netfilter.org/projects/nftables>. En debian 11 por defecto se van a sustituir las iptables por las nftables.

2.2. Nftables

nftables reemplaza a iptables, ip6tables, arptables y ebtables. Este nuevo marco de clasificación de paquetes en el kernel está basado en una máquina virtual específica de red y una nueva herramienta de línea de comando: nft.

Reutiliza partes clásicas de la infraestructura Netfilter, como el *connection tracking system* (*conntrack*, sistema de seguimiento de conexiones), el subsistema de envío de paquetes a espacio de usuario (*nf_queue*) y el subsistema de registro (*nf_log*), entre otros. También existe una capa de traducción y compatibilidad para facilitar el trabajo sobre reglas ya existentes de iptables.

Hay varias mejoras de nftables frente a iptables, como por ejemplo:

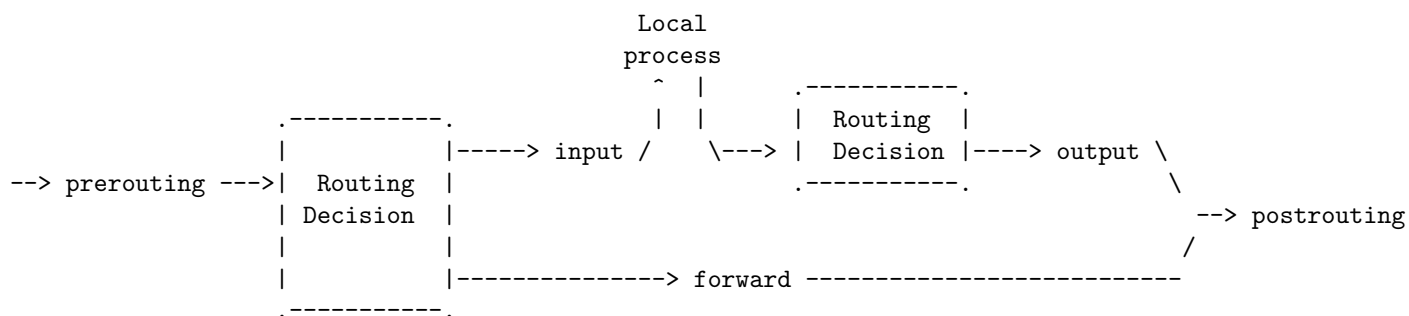
- evitar duplicidad e inconsistencia en el código fuente. Muchas extensiones de iptables estaban duplicadas con pequeños cambios para interactuar con diferentes protocolos de red.
- mejorar soporte para conjuntos y mapeo de datos.
- simplificar usabilidad en entornos IPv4/IPv6.

- etc.

Hay documentación de nftables:

- el how to en <http://wiki.nftables.org/wiki-nftables/>
- <https://home.regit.org/netfilter-en/nftables-quick-howto/>
- la FAQ de debian <https://wiki.debian.org/nftables>
- Introducción y audio en castellano <https://www.eduardocollado.com/2019/07/12/introduccion-a-nftables/>
- video muy introductorio también en castellano sobre iptables, nftables y bpfilter <https://www.youtube.com/watch?v=4x66uucq0EU>

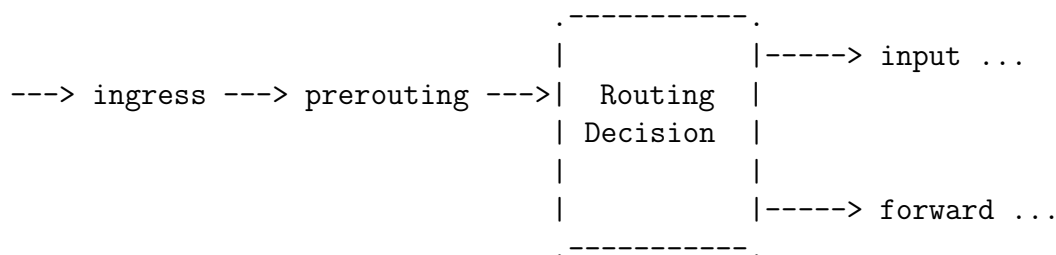
Veámos una representación visual en ASCII art de como funciona el sistema de netfilter y el encaminamiento de los paquetes al/del kernel y a/de los procesos: https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks.



Cada protocolo define unos puntos bien definidos por los que pasan los paquetes denominados *hooks* en cada uno de esos puntos el protocolo puede llamar a netfilter con el paquete y el número del hook. IPv4 definió inicialmente 5 hooks: prerouting, input, output, forward y postrouting.

Los paquetes vienen desde un adaptador siguiendo el hook de prerouting, y el de input a los procesos que los consumen. Si la máquina está configurada como router (para hacer forwarding) el tráfico seguirá el hook de forwarding y el de postrouting. El tráfico generado localmente seguirá el hook de output y el de postrouting.

Además desde el kernel 4.2 se definió un hook adicional de ingress filtering, previo al de prerouting:



La sintaxis de nftables junto con distintos ejemplos la podemos ver en el apartado “Basic rule handling” de <https://home.regit.org/netfilter-en/nftables-quick-howto/>. Ahí podemos ver como:

- añadir, quitar y listar tablas, como quitar todas las reglas y cadenas de una tabla. https://wiki.nftables.org/wiki-nftables/index.php/Configuring_tables
- También como añadir, quitar y listar cadenas en tablas (chains/hooks). https://wiki.nftables.org/wiki-nftables/index.php/Configuring_chains En nftables, las cadenas base no vienen definidas por defecto. Las cadenas son de tipo filter, route o nat, y puedes especificar el hook (donde aplicar la cadena), la prioridad y la política: accept o drop.
- Y por último muestra como añadir, quitar y listar reglas en cadenas. https://wiki.nftables.org/wiki-nftables/index.php/Configuring_chains
- nft permite operar de forma atómica insertando todas las reglas en una única operación con

```
nft -f file
```

Podemos comenzar instalando y activando nftables y añadiendo una regla para aceptar y contar tráfico ssh entrante:

```
sudo su
apt-get install nftables
systemctl enable nftables.service
nft add table ip filter
nft add chain ip filter INPUT
nft add rule ip filter INPUT tcp dport 22 ct state new counter accept
nft list table ip filter
```

Si queremos eliminar por completo cualquier actividad de nftables, podemos tirar todas las reglas:

```
# nft flush ruleset
```

Y para que no se carguen en el arranque:

```
# systemctl mask nftables.service
```

Este video explica como configurar nftables en un servidor https://www.youtube.com/watch?v=_A-Q6yTMX0g (está en inglés).