PRÁCTICA 4: TÚNELES MULTICAST DVMRP

Autores: Santiago Felici
Rogelio Montañana

1.- Objetivo

Configurar en un conjunto de hosts Linux la implementación del protocolo multicast DVMRP conocida como mrouted. Establecer entre ellos túneles IP-IP para permitir el intercambio de tráfico multicast a través de routers unicast, no preparados para tráfico multicast. Esta práctica es una continuación de la práctica 3 en la que los alumnos han probado algunas de las herramientas Mbone (SDR, VIC y RAT).

2.- Introducción

Cuando un datagrama multicast llega a un router que no soporta multicast, el router simplemente lo ignora. La propagación de tráfico multicast requiere por tanto habilitar en los routers protocolos de routing multicast, adaptados a la naturaleza singular de dicho tipo de tráfico.

El protocolo de routing multicast que utilizaremos en esta práctica es el **DVMRP** (*Distance Vector Multicast Routing Protocol*), que es el protocolo de routing multicast más antiguo, especificado en el RFC 1075. Para probar su funcionamiento emplearemos las herramientas de MBone ya utilizadas en la práctica anterior (SDR, VIC y RAT). Las características principales de DVMRP son las siguientes:

- Para cada posible fuente se construye un árbol de distribución broadcast óptimo usando como base la información de rutas unicast calculada a partir de la información suminstrada en la configuración de los routers DVMRP.

- Trabaja en modo denso, es decir inicialmente los paquetes multicast se propagan por toda la red siguiendo un árbol de distribución broadcast. Si algún router no está interesado en la emisión envía un comando de poda del árbol (comando 'prune'); el árbol de distribución broadcast se reconstruye cada 2 minutos, por lo que el podado tiene que repetirse periódicamente. La información de podado es almacenada en los routers, de forma que se puede restablecer alguna rama de distribución previamente podada si algún router lo solicita mediante el comando 'graft' (injertar).

- Se basa en el algoritmo del vector distancia usando como métrica el número de saltos (*hops*). Los vectores distancia se actualizan cada 60 segundos. DVMRP calcula sus propias rutas unicast óptimas, independientemente de la información que se disponga de rutas unicast.

- Emplea la técnica denominada Reverse Path Multicast para construir un árbol de distribución broadcast truncado sobe el cual los paquetes se encaminan siguiendo rutas óptimas.

- Normalmente se utiliza sobre **túneles IP-IP**, es decir el router DVMRP encapsula el datagrama multicast en un datagrama unicast antes de enviarlo al router DVMRP remoto. Los túneles permiten conectar 'islas' multicast a través de routers unicast. De esta forma se evita mezclar ambos tráficos (unicast y multicast) y sus protocolos de routing, de forma que los posibles problemas del routing multicast no afectarán al tráfico unicast.

La implementación más extendida de DVMRP es un programa UNIX denominado mrouted (multicast route daemon). Los hosts normalmente disponen de una única interfaz LAN sobre la que se define una interfaz virtual para cada túnel que se quiere configurar.

El protocolo DVMRP se utilizó en MBone desde los inicios en 1992 hasta que se produjo la migración al protocolo PIM-SM entre 2000-2002. PIM-SM se utiliza normalmente en modo nativo (sin túneles).

El ámbito de difusión de las emisiones multicast se delimita mediante alguno de los siguientes mecanismos:

a) Ajustando el valor del TTL.

b) Asignando rangos de direcciones a determinados ámbitos según el RFC 2365.
c) Utilizando un protocolo de delimitación de ámbitos, como el MZAP (Multicast Zone Announcement Protocol) especificado en el RFC 2776.

En esta práctica utilizaremos el primer sistema, es decir ajustaremos el valor del TTL configurando las interfaces de los routers con un valor umbral (TTL-Threshold) para limitar el ámbito o alcance del tráfico multicast. Por ejemplo si configuramos la interfaz del router DVMRP con un valor umbral del TTL de 16 cualquier paquete multicast que llegue a dicho router solo será propagado al exterior si su valor de TTL es igual o superior a 16. Igual que ocurre con los paquetes unicast los routers multicast restan 1 al valor del TTL de los paquetes multicast que pasan por ellos, descartándolos cuando el TTL vale cero. La única diferencia respecto a los paquetes unicast es que cuando se descarta un paquete no se genera ningún mensaje ICMP.

Cuando se configura en el *mrouted* un túnel se especifican tres parámetros: *el mrouter destino, el coste y el umbral o 'threshold'* (el umbral establece el TTL mínmo requerido para que el paquete sea propagado al otro lado del túnel, de forma que es posible controlar el ámbito de la emisión, como hemos comentado antes). Cuando un *mrouter* recibe por una de sus interfaces un paquete multicast el árbol de distribución activo en ese momento le dirá si lo debe reenviar por alguna otra interfaz; en ese caso el renvío solo se producirá el umbral asociado a esa interfaz es menor que el TTL del paquete.

3.- Software requerido

Todos los programas necesarios se encuentran en el directorio de la asignatura (*/iilabs/AR*) en el servidor *ftp* de *slabii.informatica.uv.es*. Todas las transferencias deben hacerse en modo binario.

Los elementos de software necesarios para esta práctica son:

- Software DVMRP (*mrouted*). Directorio */iilabs/AR/p4*.
- Software de videoconferencia MBone (SDR, VIC y RAT). Directorio */iilabs/AR/mbone*.
- Analizador de protocolos (Ethereal). Directorio */iilabs/AR/ethereal*.

La instalación del programa *mrouted* es el primer paso de esta práctica.

El software de videoconferencia MBone debe estar instalado en el ordenador como resultado de la práctica anterior. En caso de que no se encuentre o presente problemas se puede volver a instalar.

El analizador de protocolos Ethereal también debe estar instalado en el ordenador. En caso de que no se encuentre o presente problemas se puede volver a instalar.

4.- Hardware requerido

- Cinco routers con 2 interfaces LAN cada uno. Normalmente estos routers serán 1721 con una interfaz Ethernet y una Fast Ethernet, aunque el modelo puede variar en función de la disponibilidad.
- Cinco hubs de al menos cuatro puertos.
- Cinco ordenadores Linux con interfaz Ethernet en los que se instalará el programa *mrouted*
- Cinco ordenadores con interfaz Ethernet, puerto USB y tarjeta de sonido para la instalación de las herramientas de videoconferencia MBone (SDR, VIC, RAT)
- Cinco cámaras y cinco auriculares con micrófono.
- 20 latiguillos Ethernet normales (no cruzados).
- Cinco cables negros de consola para los routers

En caso de no disponer de todo el material la maqueta se adaptará a los recursos disponibles.

5. - Desarrollo

Se trata de constituir una red, según se muestra en la figura 1, formada por cinco LANs Ethernet interconectadas en topología de anillo mediante cinco routers, cada uno con dos interfaces Ethernet. En cada LAN habrá dos ordenadores y dos routers, para lo que utilizaremos un hub. En uno de los

ordenadores, que llamaremos 'mrouted', instalaremos el programa *mrouted* y hará la función de router multicast. El otro ordenador, que llamaremos 'SDR', se utilizará para ejecutar las aplicaciones de videoconferencia MBone (SDR, VIC y RAT).

Paso 1: Instalación del software de videoconferencia MBone (SDR, VIC, RAT) en los ordenaodres SDR

En primer lugar arrancaremos los ordenadores SDR con el sistema operativo linux redes y entraremos con el usuario 'root' y la password que nos indique el profesor. A continuación conectaremos la cámara USB, el auricular y el micrófono.

Es posible que el software Mbone esté ya correctamente instalado de la práctica anterior. Por tanto lo primero que haremos es intentar arrancar el programa SDR, crear una sesión e intentar emitir audio y vídeo. Si todo funciona correctamente daremos por finalizado este paso, si no instalaremos (o reinstalaremos) las herramientas que no funcionen o sobre las que tengamos dudas.

Para instalar el software MBone crearemos un directorio de nombre **mbone** en el directorio home de root (comando `'mkdir mbone'`). En este directorio instalaremos las herramientas multicast de MBone (SDR, VIC y RAT). Copiaremos todos los ficheros que se encuentran en el directorio **/iilabs/AR/mbone** de **slabii.informat.uv.es**[1]. Los ficheros se deben copiar al ordenador por *ftp* en modo binario. Normalmente la sesión ftp se establece en modo binario, pero si no fuera así para pasar a modo binario deberemos teclear el comando 'bin' (una vez dentro de la sesión ftp).

Paso 2 (ordenadores mrouted): Instalación del software de routing multicast (mrouted)

En primer lugar arrancaremos los ordenadores mbone con el sistema operativo linux redes y entraremos con el usuario 'root' y la password que nos indique el profesor. A continuación crearemos un directorio de nombre mrouted en el directorio home de root (comando `'mkdir mrouted'`). En este directorio instalaremos el programa *mrouted*. Copiaremos allí los siguientes ficheros:

- **mrouted** (ejecutable del programa DVMRP)
- **mrouted.conf** (un fichero de configuración de ejemplo)

Estos ficheros se encuentran en el directorio `/iilabs/AR/p4` de **slabii.informat.uv.es.** Los ficheros se deben copiar al ordenador por `sftp` en modo binario. Normalmente la sesión sftp se establece en modo binario, pero si no fuera así para pasar a modo binario deberemos teclear el comando `'bin'` (una vez dentro de la sesión sftp). Una vez copiado el fichero hay que darle permisos de ejecución mediante el comando:

```
chmod 555 mrouted
```

El programa mrouted es la parte principal de esta práctica. El man correspondiente se encuentra como apéndice al final de este guión. Para más información también podemos consultar el URL *http://mice.ed.ac.uk/mice/archive/mrouted.html*.

Paso 3 (todos): Interconexión de los equipos

En este paso interconectaremos los equipos de acuerdo con el esquema de la figura 1. Los routers 1721 tienen una interfaz Ethernet (Ethernet 0) y una Fast Ethernet (FastEthernet 0), aunque al conectarse a hubs de 10 Mb/s las interfaces Fast Ethernet funcionan también a 10 Mb/s. Para interconectar en cada LAN los hosts y los routers utilizaremos el hub y cuatro latiguillos normales (no cruzados). Los ordenadores mrouted servirán además como consola del router correspondiente (ver figura 1), por lo que se conectará

[1] El acceso a slabii.uv.es solo puede hacerse de forma segura, es decir mediante el comando 'sftp' (no 'ftp'). Para realizar descargas por tanto ejecutaremos el comando `'sftp cuenta@slabii.uv.es'` siendo `'cuenta'` la cuenta o identificador de usuario del alumno en slabii.uv.es. Si es la primera vez que se realiza el acceso al servidor éste pide la aprobación para descargar la clave con la cual cifrará las futuras conexiones

el puerto COM1 del ordenador al puerto CONSOLE del router correspondiente. El router tiene un conector RJ-45 mientras que el PC tiene en el puerto COM1 (tty0) un adaptador de DB-9 a RJ45; para la conexión se debe utilizar un cable negro plano con dos conectores RJ45 (no se debe utilizar un latiguillo Ethernet).

La disposición de los equipos será la siguiente:

Mesas centrales:

- Router R-M
- Mrouted M
- Hub M
- SDR M
- Router M-P
- Mrouted P
- Hub P
- SDR P

Mesas laterales:

- Router P-B
- Mrouted B
- Hub B
- SDR B
- Router B-G
- Mrouted G
- Hub G
- SDR G
- Router G-R
- Mrouted R
- Hub R
- SDR R

Para realizar las uniones entre el Router R-M y el Hub R y entre el Router P-B y el Hub P sin tener que cruzar cables de la mesa central a la lateral utilizaremos los puentes que hay entre las rosetas números 538 y 557 y entre la 547 y 559.

Paso 4 (todos los hosts): Configuración de la red unicast en los hosts

Para el desarrollo de la práctica crearemos primero una red unicast con la topología que aparece en la Figura 1.

En primer lugar desactivaremos en todos los hosts los servicios de cortafuegos IPtables para permitir el libre intercambio de paquetes con la red. Para ello utilizaremos el comando:

```
Service iptables stop
```

Después cambiaremos el nombre al fichero de resolución de nombres para que el linux trabaje sin DNS, ya que si existe este fichero algunos comandos intentan acceder al DNS lo cual retrasa considerablemente su ejecución. Para ello utilizaremos el comando:

```
mv /etc/resolv.conf /etc/resolv.conf.old
```

A continuación configuramos la dirección IP del host mediante el comando ifconfig:

```
ifconfig eth0 inet 10.0.x.y netmask 255.255.255.0
```

donde 10.0.x.y es la dirección IP que corresponde a la máquina, según el esquema que aparece en la Figura 1.

Después definimos la ruta por defecto. Esto lo hacemos mediante el comando:

```
route add default gw 10.0.x.1
```

donde x valdrá 1, 2, 3, 4 o 5 según la LAN en la que nos encontremos. Obsérvese que, aunque hay dos routers accesibles en cada LAN, utilizamos el de dirección más baja como router por defecto.

Una vez configurado el host deberemos comprobar que todas estas definiciones se han procesado correctamente mediante los comandos `'ifconfig'` y `'route -n'`. La respuesta obtenida al comando `'route –n'` debe ser similar a la siguiente:

```
route -n
Destination  Gateway      Genmask          Flags  Metric Ref    Use    Iface
10.0.3.0     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        UG     0      0      0      lo
0.0.0.0      0.0.0.0      0.0.0.0          UG     0      0      0      eth0
```

Paso 5 (routers): Configuración de los routers unicast

Cda router se configurará desde ele host que tiene la conexión de consola, según se muestra en el esquema de la figura 1. así por ejemplo el router P-B se configurará desde el host Mrouted B.

Para configurar los routers 1721 utilizaremos la conexión de consola de los ordenadores mrouted mediante el programa de emulación de terminal minicom (comando '**minicom –s**'). Los routers tienen, además de las interfaces LAN, dos interfaces Serie que no se utilizan en esta práctica. La secuencia a seguir para configurar los routers es la siguiente:

1. Abrir en la máquina Linux una ventana de shell y arrancar en ella el programa de emulación de terminal (comando `'minicom –s'`). La configuración del programa de emulación debe ser la siguiente:

   - Velocidad 9600 bits/s
   - Sin paridad
   - 8 bits de datos
   - bit de parada (8N1)
   - Dispositivo de entrada: ttyS0

2. Encender el router. En la ventana de shell debe aparecer la secuencia de mensajes de arranque. Esto nos confirma que la comunicación por el puerto de consola es correcta.

3. Una vez ha arrancado el router debe aparecer el prompt `'Router>'`; teclear el comando **`'enable'`** para pasar a modo Privilegiado. En caso de que pida una password consultar al profesor.

4. Una vez en modo Privilegiado entraremos en modo Configuración Global para introducir la configuración que corresponde a cada router. Por ejemplo en el caso del router B-G (Burjassot-Godella) la configuración que se debe introducir es la siguiente:

```
Router>ENable
Router#CONFigure Terminal
Router(config)#HOstname B-G
B-G(config)#NO IP DOMAIN-LOokup
B-G(config)#INterface Ethernet 0
B-G(config-if)#DEScription Conexion a Godella
B-G(config-if)#Ip ADdress 10.0.4.1 255.255.255.0
B-G(config-if)#NO SHutdown
B-G(config-if)#INterface FastEthernet 0
B-G(config-if)#DEScription Conexion a Burjassot
B-G(config-if)#Ip ADdress 10.0.3.2 255.255.255.0
B-G(config-if)#NO SHutdown
B-G(config-if)#INterface Serial 0
B-G(config-if)#SHutdown
B-G(config-if)#INterface Serial 1
B-G(config-if)#SHutdown
B-G(config-if)#ROUTER Eigrp 65000
B-G(config-router)#NETwork 10.0.0.0
B-G(config-router)#LIne Vty 0 4
B-G(config-line)#PASsword genios
B-G(config-line)#CTRL/Z
B-G#
```

La configuración de un protocolo de routing nos evita tener que definir rutas estáticas para cada una de las redes remotas.

Al haber configurado una password en el modo Configuración de Línea es posible acceder vía telnet al router utilizando cualquiera de sus direcciones IP (en este caso la 10.0.4.1 o la 10.0.3.2); sin embargo al no definir una password para acceso Privilegiado este solo será posible desde la consola del router, con lo que evitaremos posibles interferencias por modificación de la configuración desde más de un ordenador.

Al finalizar la configuración de los cinco routers debemos tener conectividad a nivel unicast en toda la red, cosa que comprobaremos haciendo ping a hosts arbitrarios. Si no tenemos conectividad completa deberemos hacer pruebas progresivas y revisar las conexiones y las configuraciones hasta determinar donde se encuentra el error y corregirlo, ya que mientras la conectividad unicast no sea perfecta será imposible que funcione la multicast.

Paso 6 (hosts mrouted): creación de los túneles multicast DVMRP

En primer lugar editaremos el fichero de configuración `mrouted.conf` para la generación de los túneles necesarios. Dicho fichero se encuentra en el directorio mrouted en una configuración de ejemplo que usaremos como base editándolo con algún editor de texto (vi por ejemplo).

En principio cada mrouted establecerá dos túneles, uno con cada uno de sus vecinos, siguiendo la topología lógica en anillo que se muestra en la Figura 2. Así por ejemplo mrouted-M deberá establecer dos túneles, uno con mrouted-P y otro con mrouted-R.

En la configuración se debe especificar cada túnel mediante una instrucción 'tunnel'. En ella se indican las direcciones IP unicast de los dos extremos del túnel, el local (que siempre será el propio mrouted) y el remoto. Además se indican mediante palabras clave tres parámetros:

- **Metric**: indica la métrica para el cálculo de distancias del protocolo de routing. DVMRP utiliza número de saltos únicamente. Por ejemplo en el caso del túnel entre Burjassot y Paterna el threshold debería ser 2 para reflejar el hecho de que entre estos routers hay dos saltos de distancia. Si se indica 1 se está suministrando al protocolo de routing información errónea, con lo que el cálculo de rutas óptimas no será correcto. Obsérvese que el DVMRP no aprovecha para nada la información sobre métricas que elabora el protocolo de routing unicats (en este caso EIGRP).

- **Threshold**: este parámetro indica el valor de TTL mínimo que ha de tener un paquete mutlicast para que sea enviado por ese túnel. Vamos a configurar todos los túneles con 16 de forma que solo las sesiones creadas en SDR con valor superior podrán ser vistas fuera de la LAN de origen. Normalmente 16 es el valor mínimo requerido para emisiones que abarcan más de una organización dentro de un país.

- **Rate_limit**: este parámetro fija un límite al tráfico multicast que se va a permitir en la red, a fin de evitar que una emisión incontrolada de tráfico multicast pueda saturar una red. Este parámetro no funciona como mecanismo de reserva o garantía de recursos, sino como limitación para evitar que el usuario supere el caudal establecido. Si se especifica el valor 0 se entiende que no se está imponiendo ninguna restricción.

Por ejemplo el túnel entre Manises y Paterna se declararía en Mrouted-M mediante la siguiente línea en el fichero mrouted.conf:

```
tunnel 10.0.1.3 10.0.2.3 metric 1 threshold 16 rate_limit 0
```

Ahora debemos comprobar que el módulo para encapsular IP sobre IP está habilitado en el Kernel de Linux (que normalmente no lo estará). Para ello ejecutamos el comando:

```
lsmod
```

En la lista debe aparecer '`ipip`'. Si no se encuentra significa que no está instalado. Para instalarlo ejecutar el comando:

```
insmod ipip
```

A continuación debemos mirar si la tarjeta ethernet del host mrouted tiene soporte multicast, ya que en caso contrario debemos activarlo (normalmente debe estar activado). Para esto utilizamos el comando '`ifconfig eth0`' que debe devolver algo parecido a lo siguiente:

```
[root@lab3inf05 /root]# ifconfig eth0
eth0   Link encap:Ethernet    Hwaddr 52:54:00:E1:47:4C
       inet addr:10.0.1.      Bcast: 10.0.1.255    Mask:255.255.255.0
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          . . .
```

En este ejemplo la tercera línea nos indica que está activado el routing multicast. Si en vez de eso se obtiene una respuesta parecida a la siguiente:

```
[root@lab3inf04 /root]# ifconfig eth0
eth0   Link encap:Ethernet    Hwaddr 52:54:00:E1:47:4C
       inet addr:10.0.1.3     Bcast: 10.0.1.255    Mask:255.255.255.0
       UP BROADCAST RUNNING  MTU:1500  Metric:1
          . . .
```

significa que no está activada la posibilidad de tráfico multicast en esa interfaz, y el mrouted no podrá establecer los túneles. En ese caso para activar multicast en la interfaz hay que teclear el comando:

```
ifconfig eth0 multicast
```

con lo que se activará el routing multicast (comprobar esto repitiendo el comando `'ifconfig eth0'`).

Una vez hecho todo lo anterior ya podemos arrancar el mrouted ejecutando desde el directorio mrouted el comando:

```
./mrouted –c mrouted.conf –d
```

Si los túneles se establecen correctamente la secuencia normal de mensajes que debe aparecer será similar a la siguiente:

```
[root@lab3inf05 /root]# ./mrouted –c mrouted.conf –d
debug level 2
mrouted version 3.8
Getting vifs from kernel interfaces
Installing eth0 (10.0.1.3 on subnet 10.0.1.0/24) as vif # 0 – rate=0
Getting vifs from mrouted.conf
installing tunnel from 10.0.1.3 to 10.0.2.3 as vif #1 – rate=0
Installing vifs in mrouted…
vif #0, phyint 10.0.1.3
vif #1, tunnel 10.0.1.3 -> 10.0.2.3
pruning on
Installing vifs in kernel…
vif #0, phyint 10.0.1.3
vif #1, tunnel 10.0.1.3 -> 10.0.2.3
vifs_with_neighbors = 0

Virtual Interface Table
. . .
```

A continuación aparece por consola información de las interfaces virtuales creadas (*vifs*) y de los túneles establecidos. En unos pocos minutos empezaremos a recibir en cada SDR los anuncios de las sesiones creadas en las otras LANs, lo cual demuestra que el tráfico multicast ya está atravesando los túneles. En caso de que esto no ocurra debemos revisar las direcciones IP y las rutas de cada host (recordemos comprobar la defínción de la ruta multicast).

Posibles problemas:

- Si al intentar ejecutar el programa aparece el mensaje 'Permiso denegado' se debe a que el ejecutable mrouted no tiene permiso de ejecución. En ese caso teclearemos el comando:

  ```
  chmod 555 mrouted
  ```

  para cambiar dicho permiso y poder ejecutarlo.

- Si al arrancar las intefaces tunel la ejecución aborta y aparece el mensaje:

  ```
  Setsockopt MRT_ADD_VIF on vif 1: No buffer space available
  ```

  significa que no está instalado el módulo ipip. Instalarlo tecleando el comando `'insmod ipip'`, según se ha indicado más arriba, y repetir el comando `'lsmod'` para comprobar que efectivamente se ha instalado.

- Si durante la ejecución del mrouted aparecen por consola los mensajes:

  ```
  ```

```
warning – kernel entry already exists for (10.0.2.3 224.2.158.46)
warning – setsockopt MRT_ADD_MFC: Invalid argument
warning – setsockopt MRT_DEL_MFC of (10.0.2.3 224.2.158.46): Invalid argument
```

seguramente se debe a que se esta ejecutando una version incorrecta del mrouted (probablemente una que se encuentra en algun directorio del path y que no es la que se acaba de instalar). En este caso comprobar que al invocar el porgrama mrouted se utilizo la sintaxis `'./mrouted'` para estar seguros de estar ejecutando la versión correcta. Para comprobar esto se puede emplear el comando:

```
ps –ef|grep mrouted
```

que devuelve la sintaxis exacta utilizada al invocar el comando.

- Si al arrancar el mrouted aparece:

```
[root@lab3inf05 /root]# ./mrouted –c mrouted.conf –d
debug level 2
mrouted version 3.8
Getting vifs from kernel interfaces
Getting vifs from mrouted.conf
installing tunnel from 10.0.3.3 to 10.0.2.3 as vif #1 – rate=0
can't forward: only one enabled vif
[root@lab3inf05 /root]#
```

Se debe a que la interfaz ethernet (eth0) no tiene habilitado el soporte multicast. Utilizar en este caso el comando `'ifconfig eth0 multicast'` según se ha indicado anteriormente.

Paso 7 (hosts SDR): Emisión multicast en LAN

En este paso nos prepararemos para iniciar una sesión SDR. Para ello vamos a editar el fichero **/etc/hosts**, que contiene una tabla de resolución para el nombre del host local. Esto es necesario ya que estamos funcionando sin servidor de nombres y el SDR al arrancar si no encuentra su dirección IP en este fichero intenta por resolución inversa averiguar el nombre del host donde se está ejecutando, por lo que queda bolqueado. Con la inclusión de nuestra dirección IP en el fichero `/etc/hosts` evitamos que el SDR haga la consulta al DNS y se quede bloqueado. En el fichero `/etc/hosts` debemos introducir una línea que contenga la dirección IP y nombre de nuestro host. El nombre debe ser exactamente el que aparece al ejecutar el comando `'hostname'`. Por ejemplo si el nombre que aparece en mrouted-R al ejecutar el comando hostname es lab3inf.informat.uv.es, entonces incluiremos en el fichero `/etc/hosts` una línea que diga:

```
10.0.5.3 lab3inf.informat.uv.es
```

A continuación borraremos la cache de sesiones anteriores que el SDR pueda tener almacenadas en disco. Dicha cache se guarda en el directorio `.sdr/cache` (desde el directorio home de root). Debemos borrar todos los ficheros que aparezcan en ese directorio, pero no el directorio cache mismo. Para borrarla teclearemos (desde el directorio home de root) el siguiente comando:

```
rm .sdr/cache/* -f
```

Ahora en los hosts SDR (los que tienen las cámaras) abriremos una ventana de shell y estando en el directorio home de root teclearemos el comando:

```
./mbone/sdr
```

para ejecutar el programa SDR (Session Directory). Esto ya convierte a ese host en emisor (y receptor) del grupo multicast 224.2.127.254, que corresponde a los anuncios de sesiones SAP del SDR. Podemos comprobar esto desde el host mrouted enviando un ping a esa dirección multicast y viendo como se recibe respuesta.

Posibles problemas:

- Si al arrancar el SDR aborta la ejecución y da el mensaje

```
Gethostbyname failed (hostname='lab3inf05.informat.uv.es'!
```

significa que no se ha introducido el nombre correcto en el fichero `/etc/hosts`.

Definiremos a continuación mediante el SDR una sesión (audio y vídeo) en cada LAN: para ello en la ventana de sesiones del SDR seleccionaremos 'New' y a continuación 'Create advertised session'). Entramos entonces en un diálogo con varias etapas para crear la sesión:

0. En la etapa 0 asignaremos a cada sesión un nombre que coincidirá con el de su LAN correspondiente (Burjassot, Godella, etc.); además le asignaremos la descripción que queramos (es obligatorio asignar una).
1. En la etapa 1 elegiremos el valor por defecto (sesión tipo Test).
2. En la etapa 2 también elegiremos los valores por defecto (sesión de dos horas de duración a empezar de forma inmediata).
3. En la etapa 3, 'Select the Distribution Scope', elegiremos el valor 'IPv4 World (ttl 127)' para asegurarnos de que la emisión atraviesa todos los túneles.
4. En la etapa 4 elegiremos audio y vídeo. Si no nos permite seleccionar el audio es que no ha reconocido la tarjeta de sonido.
5. En la etapa 5 (Provide Contact Details) podemos dejar los valores por defecto.
6. En la etapa 6 (Select security parameters for this session) también podemos dejar los valores por defecto.

A continuación aparece una pantalla resumen ('Review session details') donde ya podemos ver las direcciones multicast que el SDR ha asignado al flujo de vídeo y de audio que hemos creado. Todas las aplicaciones arrancadas con esta aplicación reciben direcciones del rango 224.2/16, que es el que está reservado por el IANA para el SDR.

A nivel unicast hemos constituido un anillo, lo cual junto al uso de EIGRP nos permite asegurar conectividad aún en caso de fallo de uno de los routers. Dado que a nivel multicast también hemos constituído los túneles en forma de anillo podemos también garantizar el funcionamiento en caso de fallo de uno de los routers multicast. Por ejemplo, la comunicación entre mrouted-M y mrouted-B se hará normalmente vía mrouted-P, ya que esta la ruta que ofrece una métrica menor (2 saltos). Pero si falla mrouted-P la comunicación entre mrouted-M y mrouted-B se restablecerá vía mrouted-R y mrouted-G. En caso de que fallara alguno de estos mrouted la comunicación multicast entre Manises y Burjassot ya no sería posible.

Paso 8 (hosts mrouted): pruebas adicionales

Durante la ejecución del proceso mrouted cualquier comunicación con él se realiza mandando diversas señales mediante el comando kill. Por ejemplo para terminar el proceso mrouted se utilizaría el comando:

```
kill –INT pid_mrouted
```

donde *pid_mrouted* es el identificador que el sistema ha asignado al proceso mrouted. Cuando arranaca el mrouted escribe su pid en el fichero /etc/mrouted.pid. También podemos averiguar el pid mediante el comando **'ps –ef'**, pero como la lista suele ser bastante larga resulta más práctico teclear el comando:

```
ps –ef|grep mrouted
```

que es un 'pipe' que solo nos muestra la línea correspondiente al proceso mrouted (y al grep que en ese instante estamos ejecutando). Una forma equivalente (y mucho más sencilla) de abortar el proceso mrouted es teclear CTRL/C en la consola del mrouted.

Pero el comando `'kill'` permite mandar al proceso mrouted otras señales mucho más interesante que el INT. Por ejemplo el comando:

```
kill –USR1 pid_mrouted
```

nos vuelca las tablas de routing en el fichero `/usr/tmp/mrouted.dump`. En el *man* del mrouted (ver apéndice) se explican con ejemplos todas la señales que se pueden utilizar.

Podemos probar a cambiar los valores de los parámetros asociados con los túneles (métricas o TTLs) y ver que consecuencias tiene esto en las rutas elegidas. Para cambiar algún parámetro editaremos el fichero de configuración (`mrouted.conf`) y una vez realizados los cambios enviaremos la señal HUP (comando `'kill –HUP pid_mrouted'`) para que el mrouted reinicie y cargue el nuevo fichero.

Por ejemplo intentemos predecir que ocurrirá si declaramos métrica 1 en vez de métrica 2 para el túnel Burjassot-Paterna. Hagamos el cambio y comprobemos si nuestra hipótesis es correcta.

También podemos arrancar el mrouted con un mayor nivel de deputración (debug). El nivel por defecto es 2, si se utiliza nivel 3 nos mostrará por consola todos los paquetes que enruta. Para esto se debe arrancar el proceso con el comando

```
./mrouted –c mrouted.conf –d 3
```

Utilizando el analizador de protocolos **ethereal** podemos ver los paquetes encapsulados que viajan por la red. Los túneles mrouted utilizan encapsulado IP en IP (valor 4 en el campo protocolo de la cabecera IP). Podremos ver como los paquetes encapsulados viajan en tramas Ethernet unicast.

También podemos utilizar el programa **tcpdump** para analizar el tráfico multicast que se genera, se recibe o que transita por un host.. A pesar de su nombre tcpdump permite analizar todo tipo de tráfico IP, no únicamente tráfico TCP. Por ejemplo si en un host cualquiera queremos analizar el tráfico que está llegando con dirección de origen la de SDR-G teclearemos el comando:

```
tcpdump –i eth0 –n src host 10.0.4.4
```

Los filtros de tcpdump utilizan la misma sitaxis que los de ethereal.

Si exploramos el tráfico generado por un SDR que no esté emitiendo en ningún grupo pero que esté recibiendo en alguno observaremos como de vez en cuando envía un paquete al grupo multicast correspondiente. Se trata de los informes RTCP que envía periódicamente al grupo multicast en el que se encuentra.

También es interesante observar que cuando un host hace ping a un grupo multicast las respuestas las recibe de los emisores activos en dicho grupo. Esta puede ser una manera fácil de averiguar si un host está

recibiendo un determinado grupo multicast: se le mandan desde dicho host pings al grupo deseado y se comprueba si hay o no respuesta.

El comando traceroute ejecutado en un host SDR para ver el tráfico de la dirección mutlicast en la que esté participando dicho host (como emisor o receptor) también puede dar pie a resultados interesantes.

Paso 9 (hosts mrouted): Establecimiento de túneles en una topología en estrella

Una vez comprobada la operatividad básica de mrouted podemos establecer túneles adicionales para mejorar la fiabilidad de la red. Podríamos establecer hasta cuatro túneles desde cada mrouted, llegando a tener así hasta 10 túneles configurados en toda la red. En vez de eso vamos a configurar ahora túneles con una topología en estrella centrada en Paterna, según lo representado en la Figura 3. Para ello mrouted-P establecerá túneles con los otros cuatro mrouted, y éstos solo establecerán solo un túnel con mrouted-P. De esta forma obligaremos a que todo el tráfico multicast entre cualquier par de redes pase a través de mrouted-P.

Los nuevos túneles a definir deberán tener métricas que reflejen la topología unicast de la red, así por ejemplo el túnel mrouted-P – mrouted-R deberá tener una métrica de 2, ya que este es el valor de la métrica que les corresponde en la red unicast. Obsérvese que en este caso el uso de métrica basada en número de saltos no es ningún problema puesto que todas las LAN son de las mismas características, pero la situación habría sido más difícil si se hubiera tratado de enlaces de diferente tipo (por ejemplo interfaces Ethernet mezcladas con líneas serie).

Paso 10: Finalización

Una vez finalizada la práctica los alumnos deberán realizar las siguientes tareas:

1. Volverán a poner el nombre habitual al fichero de los DNS, comando **'mv /etc/resolv.conf.old /etc/resolv.conf'**.

2. Cerrarán ordenadamente el sistema operativo Linux de los hosts (comando **'shutdown –h 0'**).

3. Apagarán los routers y las regletas de enchufes que tengan interruptores.

4. Devolverán las conexiones de red de los hosts a las tomas de la pared en las que se encontraban inicialmente, utilizando para ello los mismos latiguillos paralelos que tenían en un principio.

7.- Observaciones:

- Debido a que el programa VIC (soporte de vídeo), no incluye la opción Loopback, el mrouter y el VIC no pueden ejecutarse simultáneamente en el mismo host. Por ello es necesario que un host haga de *mrouter* y otro haga de emisor/receptor multicast.

8.- Agradecimientos:

Se agradece a Ulisses Alonso su colaboración en la puesta a punto de esta práctica y en la resolución de la multitud de problemas técnicos que la han rodeado.

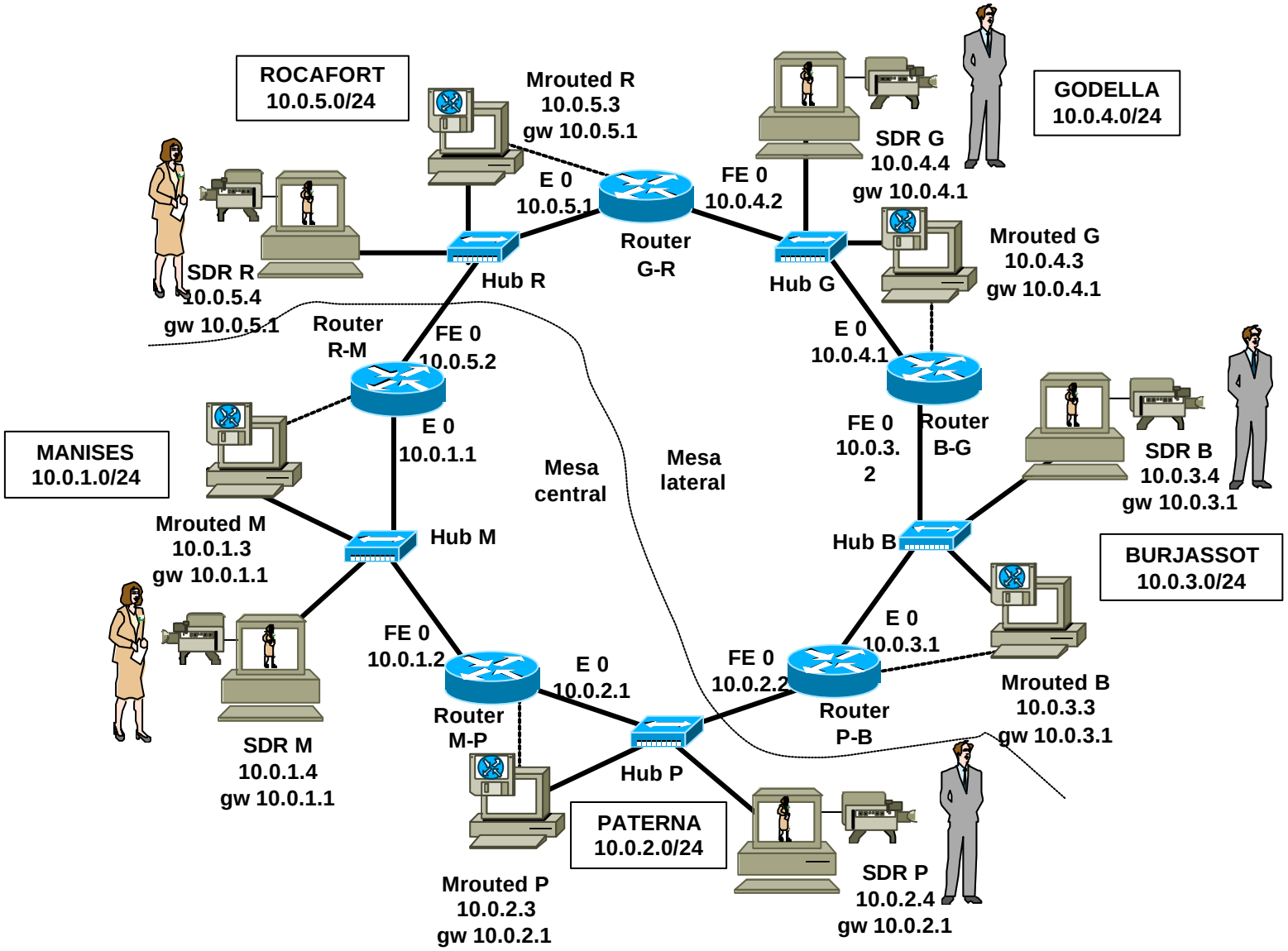
ROCAFORT
10.0.5.0/24
Mrouted R
10.0.5.3
gw 10.0.5.1
E 0
10.0.5.1
FE 0
10.0.4.2
SDR G
10.0.4.4
gw 10.0.4.1
GODELLA
10.0.4.0/24
SDR R
10.0.5.4
gw 10.0.5.1
Hub R
Router
G-R
Hub G
Mrouted G
10.0.4.3
gw 10.0.4.1
Router
R-M
FE 0
10.0.5.2
E 0
10.0.4.1
MANISES
10.0.1.0/24
E 0
10.0.1.1
Mesa
central
Mesa
lateral
FE 0
10.0.3.
2
Router
B-G
SDR B
10.0.3.4
gw 10.0.3.1
Mrouted M
10.0.1.3
gw 10.0.1.1
Hub M
Hub B
BURJASSOT
10.0.3.0/24
FE 0
10.0.1.2
E 0
10.0.2.1
FE 0
10.0.2.2
E 0
10.0.3.1
Router
M-P
Router
P-B
Mrouted B
10.0.3.3
gw 10.0.3.1
SDR M
10.0.1.4
gw 10.0.1.1
Hub P
PATERNA
10.0.2.0/24
Mrouted P
10.0.2.3
gw 10.0.2.1
SDR P
10.0.2.4
gw 10.0.2.1

Figura 1: Topología de la red física con detalle de las direcciones IP de los equipos

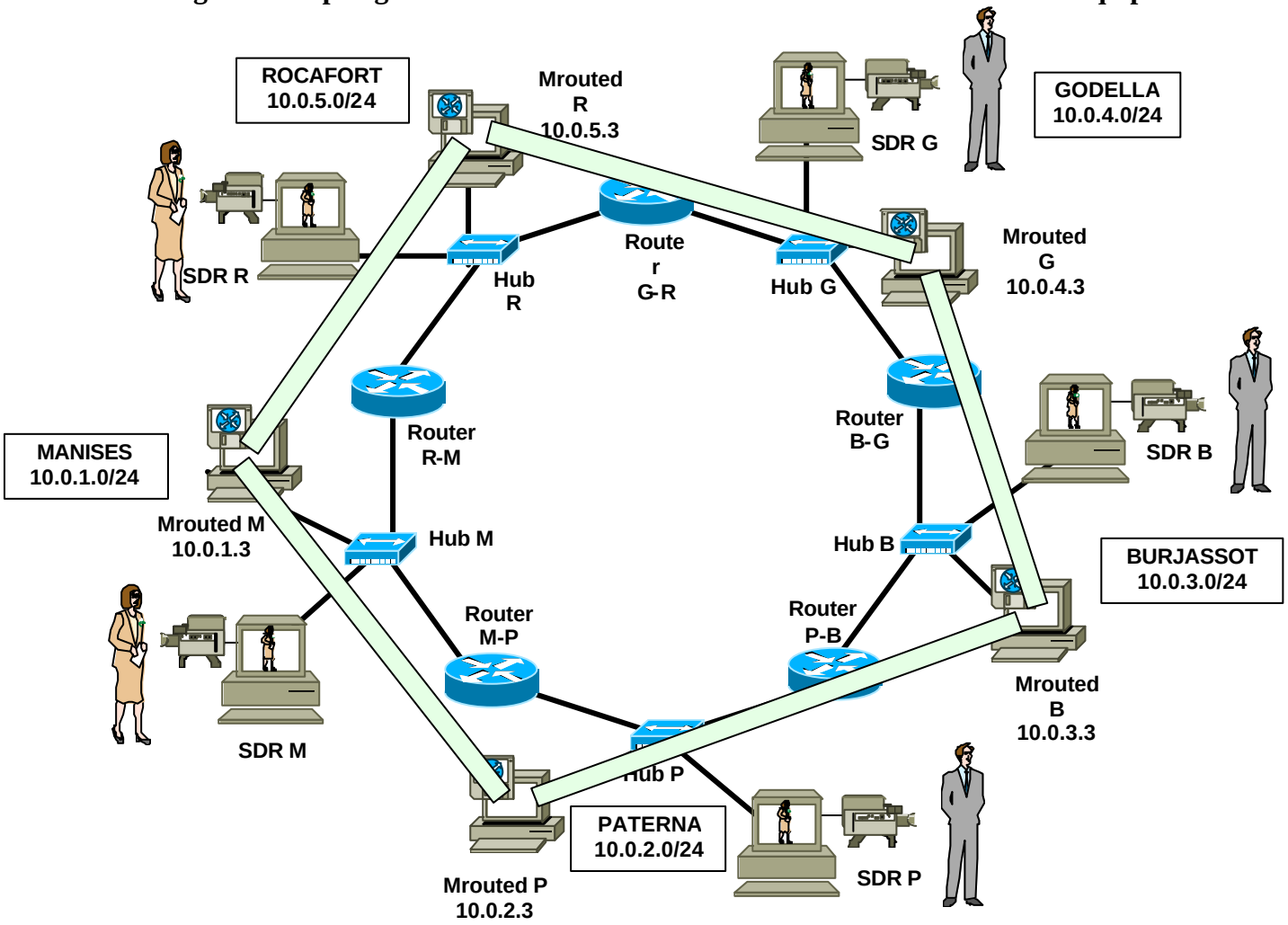

Figura 2: Topología de la red lógica de túneles DVMRP en anillo

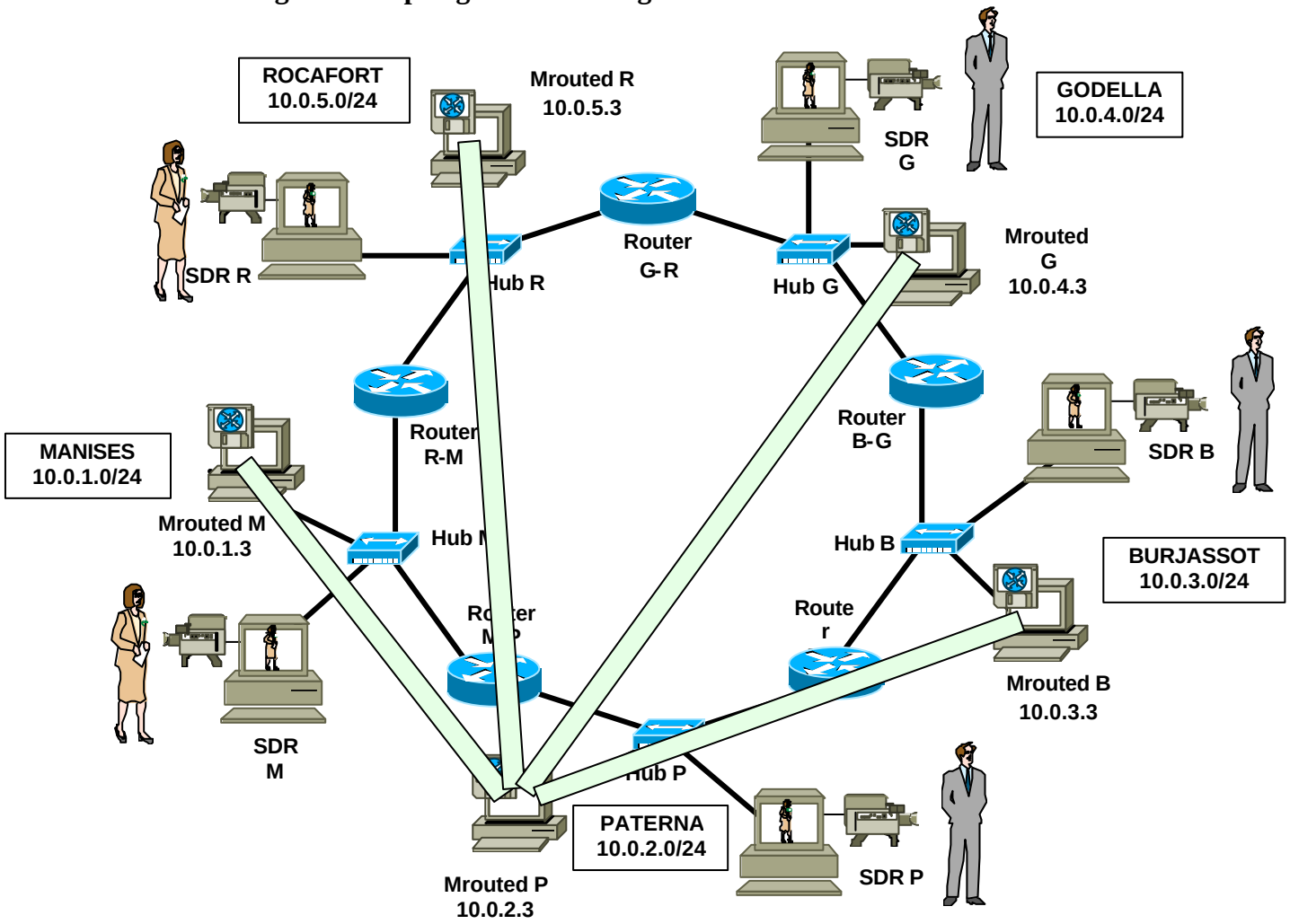

Figura 3: Topología de la red lógica de túneles DVMRP en estrella

APÉNDICE: MAN DEL COMANDO MROUTED

NAME

mrouted - IP multicast routing daemon

SYNOPSIS

/usr/sbin/mrouted [**-c** *config_file*] [**-d** [*debug_level*]]

DESCRIPTION

Mrouted is an implementation of the Distance Vector Multicast Routing Protocol (DVMRP), an earlier version of which is specified in RFC 1075. It maintains topological knowledge via a distance vector routing protocol (like RIP, described in RFC 1058), upon which it implements a multicast datagram forwarding algorithm called Reverse Path Multicasting.

Mrouted forwards a multicast datagram along a shortest (reverse) path tree rooted at the subnet on which the datagram originates. The multicast delivery tree may be thought of as a broadcast delivery tree that has been pruned back so that it does not extend beyond those subnetworks that have members of the destination group. Hence, datagrams are not forwarded along those branches which have no listeners of the multicast group. The IP time to live of a multicast datagram can be used to limit the range of multicast datagrams.

In order to support multicasting among subnets that are separated by (unicast) routers that do not support IP multicasting, *mrouted* includes support for "tunnels", which are virtual point to point links between pairs of multicast routers located anywhere in an internet. IP multicast packets are encapsulated for transmission through tunnels, so that they look like normal unicast datagrams to intervening routers and subnets. The encapsulation is added on entry to a tunnel, and stripped off on exit from a tunnel. The packets are encapsulated using the IP in IP protocol (IP protocol number 4). Older versions of *mrouted* tunneled using IP source routing, which puts a heavy load on some types of routers. This version does not support IP source route tunnelling.

The tunnelling mechanism allows *mrouted* to establish a virtual internet, for the purpose of multicasting only, which is independent of the physical internet, and which may span multiple Autonomous Systems. This capability is intended for experimental support of internet multicasting only, pending widespread support for multicast routing by the regular (unicast) routers. *Mrouted* suffers from the well known scaling problems of any distance vector routing protocol, and does not (yet) support hierarchical multicast routing.

Mrouted handles multicast routing only; there may or may not be unicast routing software running on the same machine as *mrouted*. With the use of tunnels, it is not necessary for *mrouted* to have access to more than one physical subnet in order to perform multicast forwarding.

OPTIONS

-c *config_file*

Specifies an alternate configuration file to read (normally /etc/mrouted.conf)

-d *debug_level*

Turn on debugging; *debug_level* is a comma separated list of subsections to debug.

INVOCATION

If no "-d" option is given, *mrouted* detaches from the invoking terminal. Otherwise, it remains attached to the invoking terminal and responsive to signals from that terminal. Regardless of the debug level, *mrouted* always writes warning and error messages to the system log daemon. The *debug level* argument is a comma separated list of any of the following:

packet	Display the type, source and destination of all packets sent or received.
pruning	Display more information about prunes sent or received.
routing	Display more information about routing update packets sent or received.
route_detail	Display routing updates in excruciating detail. This is generally way too much information.
Neighbors	Display information about neighbor discovery.
Cache	Display insertions, deletions and refreshes of entries in the kernel forwarding cache.
Timeout	Debug timeouts and periodic processes.
interface	Display information about interfaces and their configuration.
Membership	Display information about group memberships on physical interfaces.
traceroute	Display information about multicast traceroute requests passing through this router.
Igmp	Display IGMP operation including group membership and querier election. Monitor ICMP handling.
.	
icmp rsrr	Monitor RSRR operation.

Upon startup, mrouted writes its pid to the file /var/run/mrouted.pid.

CONFIGURATION

Mrouted automatically configures itself to forward on all multicast capable interfaces, i.e., interfaces that have the IFF_MULTICAST flag set (excluding the loopback "interface"), and it finds other DVMRP routers directly reachable via those interfaces. To override the default configuration, or to add tunnel links to other multicast routers, configuration commands may be placed in /etc/mrouted.conf (or an alternative file, specified by the "-c" option).

The file format is free form; whitespace (including newlines) is not significant. Begins with commands that apply to *mrouted*'s overall operation or set defaults.

cache_lifetime *secs*

Specifies, in seconds, the lifetime of a multicast forwarding cache entry in the kernel. Multicast forwarding cache entries in the kernel are checked every *secs* seconds, and are refreshed if the source is still active or deleted if not. Care should be taken when setting this value, as a low value can keep the kernel cache small at the cost of "thrashing" the cache for periodic senders, but high values can cause the kernel cache to grow unacceptably large. The default is 300 seconds (5 minutes).

prune_lifetime *secs*

Specifies, in seconds, the average lifetime of prunes that are sent towards parents. The actual lifetimes will be randomized in the range [.5*secs*,1.5*secs*]. The default is 7200 (2 hours). Smaller values cause less state to be kept both at this router and the parent, at the cost of more frequent broadcasts. However, some routers (e.g. mrouted <3.3 and all currently known versions of cisco's IOS) do not use the DVMRP generation ID to determine that a neighbor has rebooted. Prunes sent towards these neighbors should be kept short, in order to shorten the time to recover from a reboot. For use in this situation, the prune_lifetime keyword may be specified on an interface as described below.

noflood *Mrouted* uses a DVMRP optimization to prevent having to keep individual routing tables for each neighbor; part of this optimization is that *mrouted* assumes that it is the forwarder for each of its attached subnets on startup. This can cause duplicates for a short period (approximately one full route report interval), since both the router that just started up and the proper forwarder will be forwarding traffic. This behavior can be turned off with the noflood keyword; *mrouted* will not assume that it is the forwarder on startup. Turning on noflood can cause black holes on restart, which will generally last approximately one full route report interval. The noflood keyword can also be specified on individual interfaces.

rexmit_prunes *[on|off]*

Mrouted's default is to retransmit prunes on all point to point interfaces (including tunnels) but no multiaccess interfaces. This option may be used to make the default on (or off) for all interfaces. The rexmit_prunes keyword can also be specified on individual interfaces.

name *boundary-name scoped-addr/mask-len*

Associates *boundary-name* with the boundary described by *scoped-addr/mask-len*, to help make interface configurations more readable and reduce repetition in the configuration file.

The second section of the configuration file, which may optionally be empty, describes options that apply to physical interfaces.

phyint *localaddr|ifname*

The phyint command does nothing by itself; it is simply a place holder which interface specific commands may follow. An interface address or name may be specified.

Disable Disables multicast forwarding on this interface. By default, *mrouted* discovers all locally attached multicast capable interfaces and forwards on all of them.

netmask *netmask*

If the kernel's netmask does not accurately reflect the subnet (e.g. you're using proxy ARP in lieu of IP subnetting), use the netmask command to describe the real netmask.

altnet*network/mask-len*

If a phyint is attached to multiple IP subnets, describe each additional subnet with the altnet keyword. This command may be specified multiple times to describe multiple subnets.

igmpv1 If there are any IGMPv1 routers on the phyint, use the **igmpv1** keyword to force *mrouted* into IGMPv1 mode. All routers on the phyint must use the same version of IGMP.

force_leaf Force *mrouted* to ignore other routers on this interface. mrouted will never send or accept neighbor probes or route reports on this interface.

In addition, the common vif commands described later may all be used on a phyint.

The third section of the configuration file, also optional, describes the configuration of any DVMRP tunnels this router might have.

tunnel *local-addr|ifname remote-addr|remote-hostname*

This command establishes a DVMRP tunnel between this host (on the interface described by *local-addr* or *ifname*) and a remote host (identified by *remote-addr* or *remote-hostname*). A remote hostname may only be used if it maps to a single IP address. A tunnel must be configured on both routers before it can be used.

Be careful that the unicast route to the remote address goes out the interface specified by the *local-addr|ifname* argument. Some UNIX kernels rewrite the source address of *mrouted*'s packets on their way out to contain the address of the transmission interface. This is best assured via a static host route.

The common vif commands described below may all be used on tunnels or phyints.

metric *m* The metric is the "cost" associated with receiving a datagram on the given interface or tunnel; it may be used to influence the choice of routes. The metric defaults to 1. Metrics should be kept as small as possible, because DVMRP cannot route along paths with a sum of metrics greater than 31.

advert_metric *m*

The advert_metric is the "cost" associated with sending a datagram on the given interface or tunnel; it may be used to influence the choice of routes. The advert_metric defaults to 0. Note that the effective metric of a link is one end's metric plus the other end's advert_metric.

threshold*t* The threshold is the minimum IP time to live required for a multicast datagram to be forwarded to the given interface or tunnel. It is used to control the scope of multicast datagrams. (The TTL of forwarded packets is only compared to the threshold, it is not decremented by the threshold. Every multicast router decrements the TTL by exactly 1.) The default threshold is 1.

In general, all multicast routers connected to a particular subnet or tunnel should use the same metric and threshold for that subnet or tunnel.

rate_limit *r* The rate_limit option allows the network administrator to specify a certain bandwidth in Kbits/second which would be allocated to multicast traffic. It defaults 0 (unlimited).

boundary*boundary-name|scoped-addr/mask-len*

The boundary option allows an interface to be configured as an administrative boundary for the specified scoped address. Packets belonging to this address will not be forwarded on a scoped interface. The boundary option accepts either a name or a boundary spec. This command may be specified several times on an interface in order to describe multiple boundaries.

passive No packets will be sent on this link or tunnel until we hear from the other end. This is useful for the "server" end of a tunnel that goes over a dial-on-demand link; configure the "server" end as passive and it will not send its periodic probes until it hears one from the other side, so will not keep the link up. If this option is specified on both ends of a tunnel, the tunnel will never come up.

noflood As described above, but only applicable to this interface/tunnel.

prune_lifetime *secs*

As described above, but only applicable to this interface/tunnel.

rexmit_prunes *[on|off]*

As described above, but only applicable to this interface/tunnel. Recall that prune retransmission defaults to on on point to point links and tunnels, and off on multiaccess links.

allow_nonpruners

By default, *mrouted* refuses to peer with DVMRP neighbors that do not claim to support pruning. This option allows such peerings on this interface.

notransit A specialized case of route filtering; no route learned from an interface marked "notransit" will be advertised on another interface marked "notransit". Marking only a single interface "notransit" has no meaning.

accept|**deny** *(route/mask-len [exact])+ [bidir]*

The **accept** and **deny** commands allow rudimentary route filtering. The **accept** command causes *mrouted* to accept only the listed routes on the configured interface; the **deny** command causes *mrouted* to accept all but the listed routes. Only one of **accept** or **deny** commands may be used on a given interface.

The list of routes follows the **accept** or **deny** keyword. If the keyword *exact* follows a route, then only that route is matched; otherwise, that route and any more specific route is matched. For example, **deny 0/0** denys all routes, while **deny 0/0 exact** denys only the default route. The default route may also be specified with the **default** keyword.

The **bidir** keyword enables bidirectional route filtering; the filter will be applied to routes on both output and input. Without the **bidir** keyword, **accept** and **deny** filters are only applied on input. Poison reverse routes are never filtered out.

Mrouted will not initiate execution if it has fewer than two enabled vifs, where a vif (virtual interface) is either a physical multicast capable interface or a tunnel. It will log a warning if all of its vifs are tunnels; such an *mrouted* configuration would be better replaced by more direct tunnels (i.e. eliminate the middle man).

EXAMPLE CONFIGURATION

This is an example configuration for a mythical multicast router at a big school.

```
#
# mrouted.conf example
#
# Name our boundaries to make it easier
name LOCAL 239.255.0.0/16
name EE 239.254.0.0/16
#
# le1 is our gateway to compsci, don't forward our
# local groups to them
phyint le1 boundary EE
#

# le2 is our interface on the classroom net, it has four
# different length subnets on it.
# note that you can use either an ip address or an
# interface name
phyint 172.16.12.38 boundary EE altnet 172.16.15.0/26

altnet 172.16.15.128/26 altnet 172.16.48.0/24


#
```

```
# atm0 is our ATM interface, which doesn't properly
# support multicasting.
phyint atm0 disable
#
# This is an internal tunnel to another EE subnet
# Remove the default tunnel rate limit, since this
# tunnel is over ethernets
tunnel 192.168.5.4 192.168.55.101 metric 1 threshold 1

rate_limit 0


#
# This is our tunnel to the outside world.
# Careful with those boundaries, Eugene.
tunnel 192.168.5.4 10.11.12.13 metric 1 threshold 32

boundary LOCAL boundary EE
```

SIGNALS

Mrouted responds to the following signals:

HUP restarts *mrouted* . The configuration file is reread every time this signal is evoked.

INT terminates execution gracefully (i.e., by sending goodbye messages to all neighboring routers).

TERM same as INT

USR1 dumps the internal routing tables to /var/tmp/mrouted.dump.

USR2 dumps the internal cache tables to /var/tmp/mrouted.cache.

QUIT dumps the internal routing tables to stderr (only if *mrouted* was invoked with a nonzero debug level).

For convenience in sending signals, *mrouted* writes its pid to /var/run/mrouted.pid upon startup.

EXAMPLE

The routing tables dumped in /var/tmp/mrouted.dump look like this:

```
Virtual Interface Table

Vif LocalAddress
0   36.2.0.8     subnet: 36.2/16      1      1   querier
                 groups: 224.0.2.1
                 224.0.0.4


pkts in: 3456

                pkts out: 2322323
1   36.11.0.1   subnet: 36.11/16 1           1   querier
                groups: 224.0.2.1
                224.0.1.0
```

```
              224.0.0.4
              pkts in: 345
              pkts out: 3456
2   36.2.0.8  tunnel: 36.8.0.77 3          1
              peers: 36.8.0.77 (3.255)
3   boundaries: 239.0.1/24
    : 239.1.2/24
    pkts in: 34545433
    pkts out: 234342

    36.2.0.8 tunnel: 36.6.8.23 3

Multicast Routing Table (1136 entries)

Origin-        FromGateway    Metric Tmr InVif OutVifs
Subnet 36.2                   1 45 0 1* 2 3*
36.8           36.8.0.77      4 15 2 0* 1* 3* 1 20 1 0*
36.11 .                       2 3*
.
.
```

In this example, there are four vifs connecting to two subnets and two tunnels. The vif 3 tunnel is not in use (no peer address). The vif 0 and vif 1 subnets have some groups present; tunnels never have any groups. This instance of *mrouted* is the one responsible for sending periodic group membership queries on the vif 0 and vif 1 subnets, as indicated by the "querier" flags. The list of boundaries indicate the scoped addresses on that interface. A count of the no. of incoming and outgoing packets is also shown at each interface.

Associated with each subnet from which a multicast datagram can originate is the address of the previous hop router (unless the subnet is directly connected), the metric of the path back to the origin, the amount of time since we last received an update for this subnet, the incoming vif for multicasts from that origin, and a list of outgoing vifs. "*" means that the outgoing vif is connected to a leaf of the broadcast tree rooted at the origin, and a multicast datagram from that origin will be forwarded on that outgoing vif only if there are members of the destination group on that leaf.

Mrouted also maintains a copy of the kernel forwarding cache table. Entries are created and deleted by *mrouted*.

The cache tables dumped in /var/tmp/mrouted.cache look like this:

```
Multicast Routing Cache Table (147 entries)

Origin
13.2.116/22       224.2.127.255    3m  2m    0     1
>13.2.116.19
>13.2.116.196
138.96.48/21      224.2.127.255    5m  2m    0     1
>138.96.48.108
128.9.160/20      224.2.127.255    3m  2m    0     1
>128.9.160.45


198.106.194/24 224.2.135.190 9m 28s 9m 0P
>198.106.194.22
```

Each entry is characterized by the origin subnet number and mask and the destination multicast group. The 'CTmr' field indicates the lifetime of the entry. The entry is deleted from the cache table (or refreshed, if traffic is flowing) when the timer decrements to zero. The 'Age' field is the time since this cache entry was originally created. Since cache entries get refreshed if traffic is flowing, routing entries can grow very old. The 'Ptmr' field is simply a dash if no prune was sent upstream, or the amount of time until the upstream prune will time out. The 'Ivif' field indicates the incoming vif for multicast packets from that origin. Each router also maintains a record of the number of prunes received from neighboring routers for a particular source and group. If there are no members of a multicast group on any downward link of the multicast tree for a subnet, a prune message is sent to the upstream router. They are indicated by a "P" after the vif number. The Forwvifs field shows the interfaces along which datagrams belonging to the sourcegroup are forwarded. A "p" indicates that no datagrams are being forwarded along that interface. An unlisted interface is a leaf subnet with no members of the particular group on that subnet. A "b" on an interface indicates that it is a boundary interface, i.e. traffic will not be forwarded on the scoped address on that interface. An additional line with a ">" as the first character is printed for each source on the subnet. Note that there can be many sources in one subnet. An additional line with a "<" as the first character is printed describing any prunes received from downstream dependent neighbors for this subnet and group.

FILES

/etc/mrouted.conf *mrouted*'s configuration file.

/var/run/mrouted.pid *mrouted*'s PID file.

/var/tmp/mrouted.dump Where *mrouted* dumps its routing table when sent a SIGUSR1.

/var/tmp/mrouted.cache Where *mrouted* dumps its forwarding cache when sent
SIGUSR2.

Note that these files are located in the following places on pre-4.4BSD systems:

/etc/mrouted.conf *mrouted*'s configuration file.

/etc/mrouted.pid*mrouted*'s PID file.

/usr/tmp/mrouted.dump Where *mrouted* dumps its routing table when sent a SIGUSR1.

/usr/tmp/mrouted.cache Where *mrouted* dumps its forwarding cache when sent SIGUSR2.

SEE ALSO

mrinfo(8)**, mtrace**(8)**, mapmbone**(8)

DVMRP is described, along with other multicast routing algorithms, in the paper "Multicast Routing in Internetworks and Extended LANs" by S. Deering, in the Proceedings of the ACM SIGCOMM '88 Conference.

AUTHORS

Steve Deering, Ajit Thyagarajan, Bill Fenner