
UNIVERSIDAD DE GRANADA

MASTER PROFESIONAL EN INGENIERÍA INFORMÁTICA

PRÁCTICA 1

IP Móvil

Autor:

Manuel Jesús García Manday
(nickter@correo.ugr.es)

Master en Ingeniería Informática

20 de abril de 2017

Índice

1. Objetivo.	3
2. Configuración inicial del entorno.	3
3. IP Móvil.	19
3.1. Instalación de IP Móvil	20
3.2. Simulación de IP Móvil	26

1. Objetivo.

El objetivo de esta práctica es conocer y trabajar con el protocolo **IP Móvil**, mediante el cual es posible implementar la movilidad (que no es lo mismo que nomacidad) de un dispositivo móvil sobre diferentes redes.

2. Configuración inicial del entorno.

Vamos a montar un entorno compuesto por tres redes (LAN1, LAN2 y LAN3) y seis hosts (FA, HA, Host, R1, R2, MN) distribuidos entre las diferentes redes, cada uno de ellos con un claro objetivo como se muestra en la imagen de a continuación.

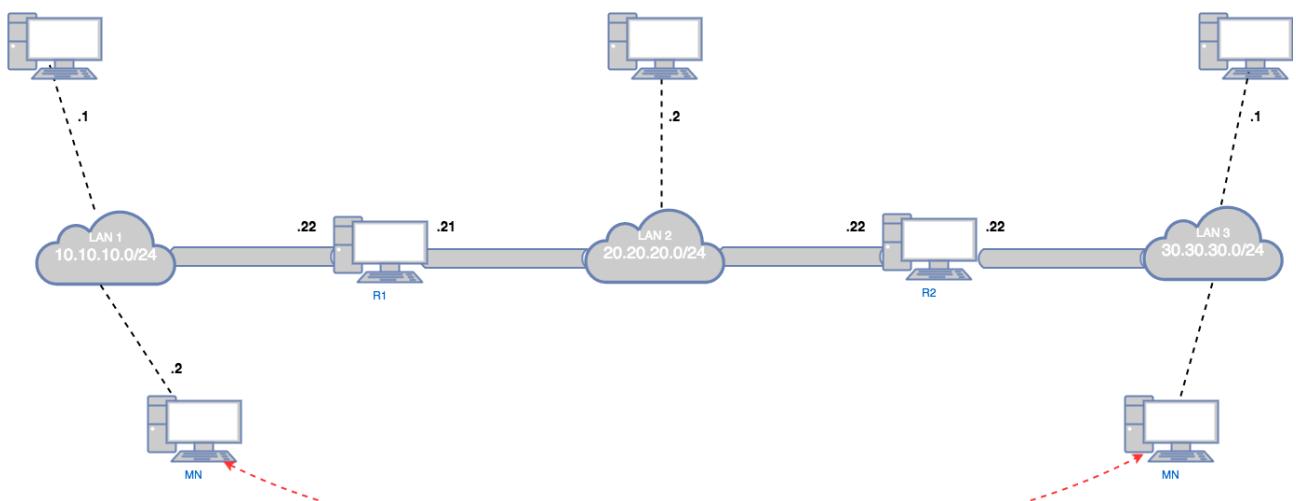


Figura 1: Entorno de redes.

La red **LAN1** es la red base del dispositivo móvil y va a contar con el siguiente rango de red **10.10.10.0/24**. La red **LAN2** es la red intermedia y va a contar con el rango de red **20.20.20.0/24**. La red **LAN3** es la red visitada por el dispositivo móvil y va contar con un rango de red **30.30.30.0/24**.

Una vez definidas las redes que se van a montar pasamos a describir los diferentes hosts:

- **FA:** es el host que hace de agente local y pertenece a la red **LAN1** con la dirección ip **10.10.10.1**. Su función es la de anunciar continuamente su presencia en dicha red. También es capaz de redirigir información a otra red haciendo **tunneling** si es necesario.
- **HA:** es el host que hace de agente externo y pertenece a la red **LAN3** con la dirección ip **30.30.30.1**. Al igual que el **FA** posee también la funcionalidad de anunciar continuamente su presencia en la red a la que pertenece. También es capaz de redirigir información a otra red haciendo **tunneling** si es necesario.
- **HostIntermedio:** es el host que se encuentra en la red intermedia **LAN2**
- **R1:** es el host que hará de router que se encuentra conectado a la red **LAN1** a través de una interfaz de red con ip **10.10.10.22** y a la red **LAN2** mediante la interfaz de red con ip **20.20.20.21**.
- **R2:** es el host que hará de roter que se encuentra conectado a la red **LAN2** a través de una interfaz de red con ip **20.20.20.22** y a la red **LAN3** mediante la interfaz de red con ip **30.30.30.21**.
- **MN:** es el host que hace dispositivo móvil que inicialmente se encuentra situado en la red base **LAN1** con la dirección ip **10.10.10.2** y que se desplazará hacia la red visitada **LAN3** con la dirección ip **30.30.30.2**.

Para la creación y configuración de los hosts vamos a utilizar el hipervisor **VirtualBox**, mediante el cual se van a definir las seis instancias necesarias con las mismas prestaciones hardware y con una distribución **Ubuntu 14.04** por cada cada una de ellas.

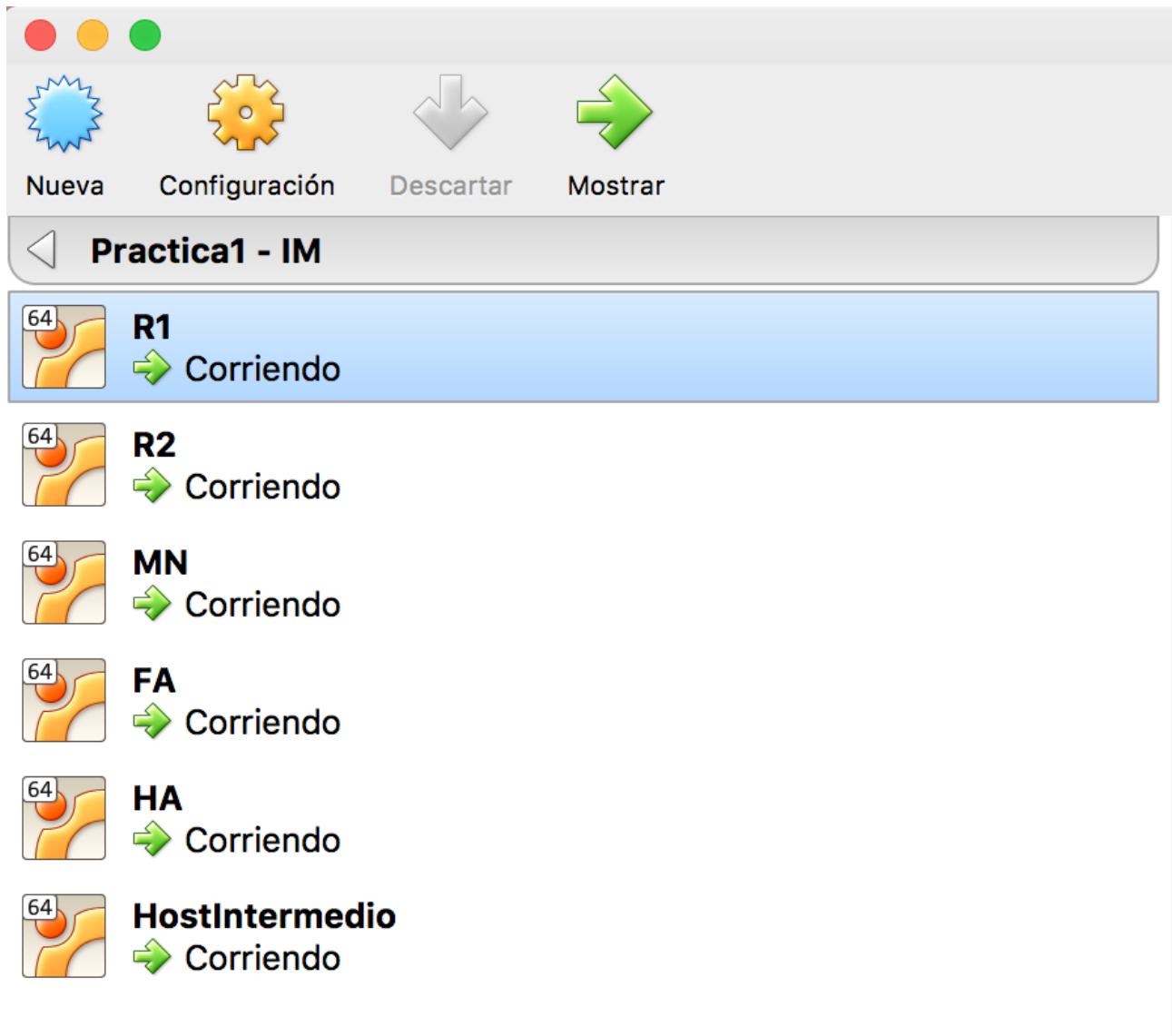


Figura 2: Definición de los hosts.

Con los hosts ya creados y corriendo como muestra la imagen anterior, lo siguiente es pasar a configurarlos de acorde al esquema de la **Figura 1**, asignándoles la red a la que pertenece el adaptador de red correspondiente y la dirección ip que tomará cada interfaz de red.

El host **HA** como se puede ver en la **Figura 1** tiene una interfaz de red **eth0** a la cual se le asignará la dirección ip **10.10.10.1** y que se encontrará situada en la red base **LAN1**. Se le añade como router el host **R1** mediante el cual podrá comunicarse con la red intermedia **LAN2**, el cual será el que tomará por defecto. La dirección ip de la interfaz de red será asignada a través de un script como se puede ver a continuación.

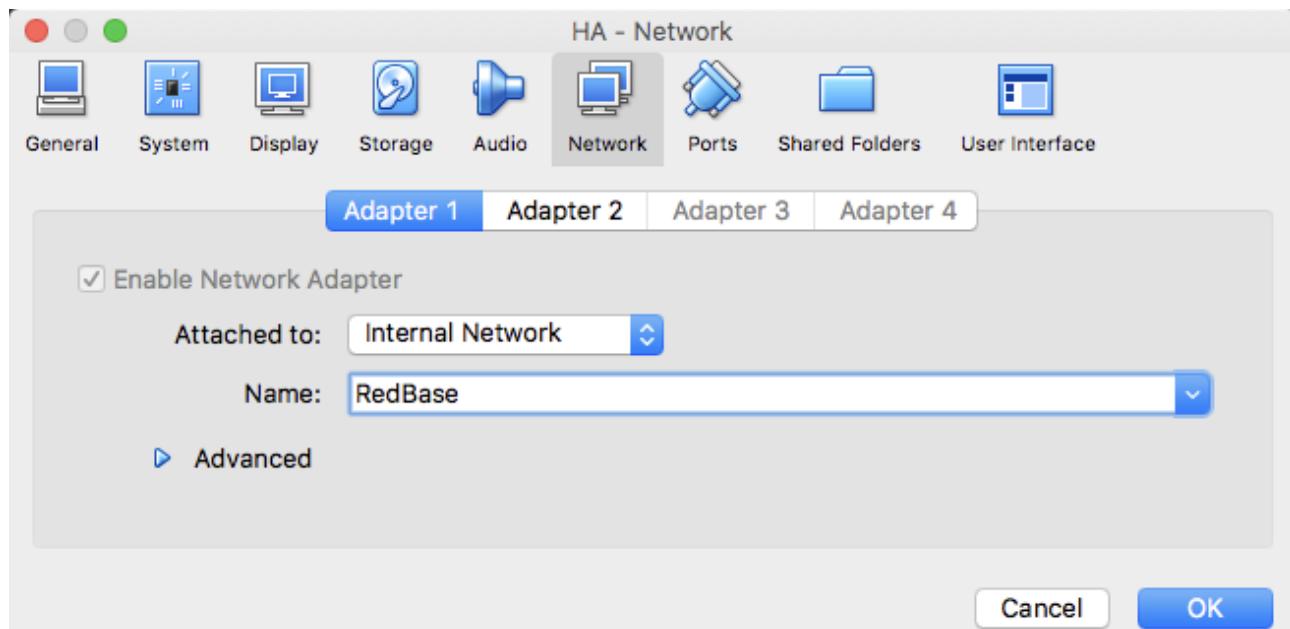


Figura 3: Interfaz de red **eth0** de **HA**.

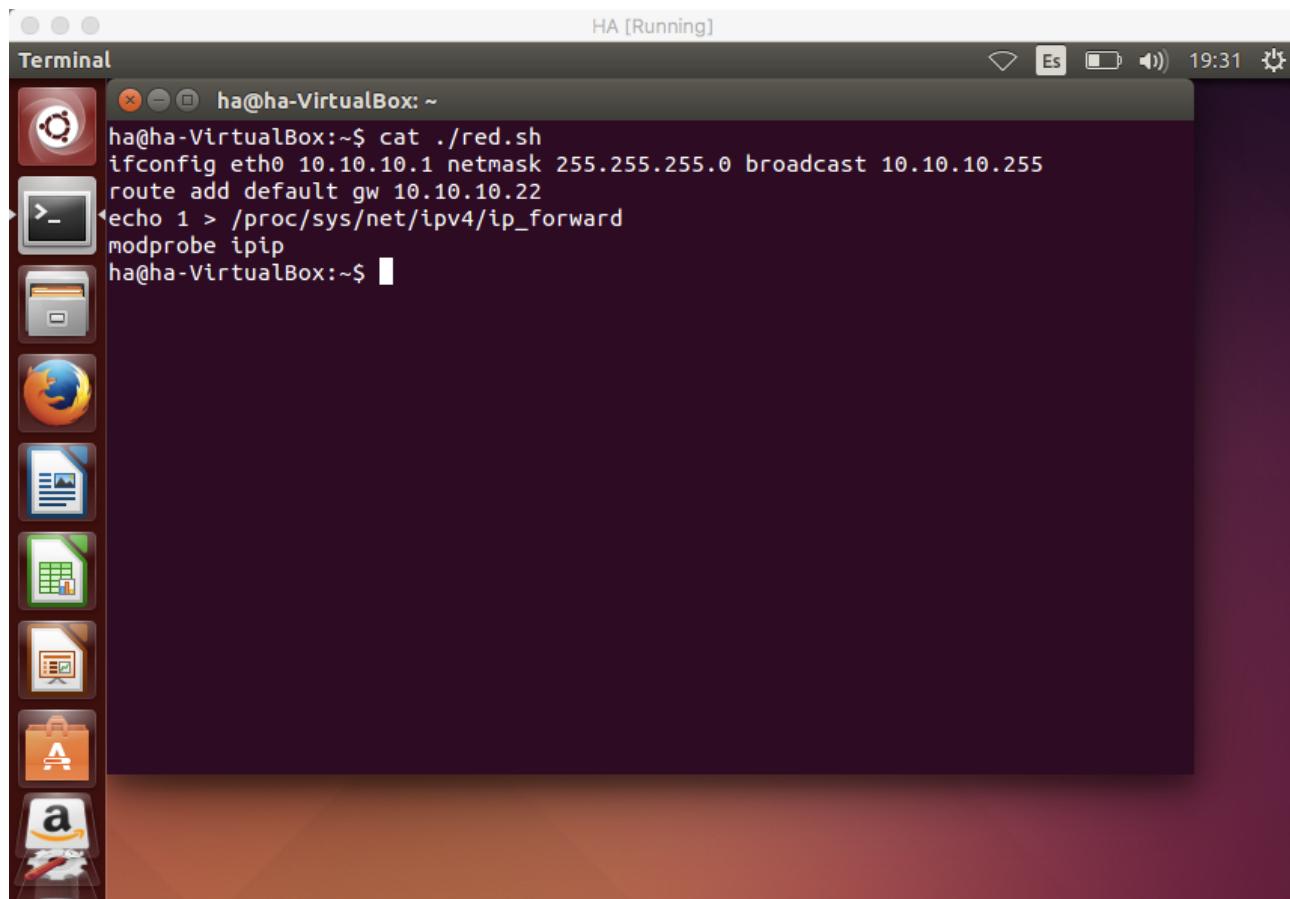


Figura 4: Script para asignar la dirección ip a **eth0** en **HA**.

```
Bytes RX:2480 (2.4 KB) TX bytes:2480 (2.4 KB)

ha@ha-VirtualBox:~$ sudo ./red.sh
[sudo] password for ha:
ha@ha-VirtualBox:~$ ifconfig
eth0      Link encap:Ethernet  direcciónHW 08:00:27:ba:9b:8c
          Direc. inet:10.10.10.1  Difus.:10.10.10.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:feba:9b8c/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:0 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:17 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:0 (0.0 B) TX bytes:2289 (2.2 KB)

lo        Link encap:Bucle local
          Direc. inet:127.0.0.1  Másc:255.0.0.0
          Dirección inet6: ::1/128 Alcance:Anfitrión
          ACTIVO BUCLE FUNCIONANDO  MTU:65536  Métrica:1
          Paquetes RX:80 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:80 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1
          Bytes RX:6128 (6.1 KB) TX bytes:6128 (6.1 KB)

ha@ha-VirtualBox:~$
```

Figura 5: Asignación de la dirección ip a **eth0** dentro de la red base **LAN1**.

El host **R1** se compone de dos interfaces de redes como se aprecia en la figura del esquema debido a que hace de router entre la red base (**LAN1**) y la red intermedia (**LAN2**). Se añade el host **R2** como router por defecto mediante el cual podrá comunicarse con la red visitada **LAN3**. Al igual que con el host anterior, a través de un script se asignarán las diferentes direcciones ip para cada interfaz de red, siendo la dirección **10.10.10.22** para la interfaz de red **eth0** que se encuentra en la red base (**LAN1**) y la dirección **20.20.20.21** para la interfaz de red **eth1** que está en la red intermedia (**LAN2**) como se puede apreciar en las imágenes.

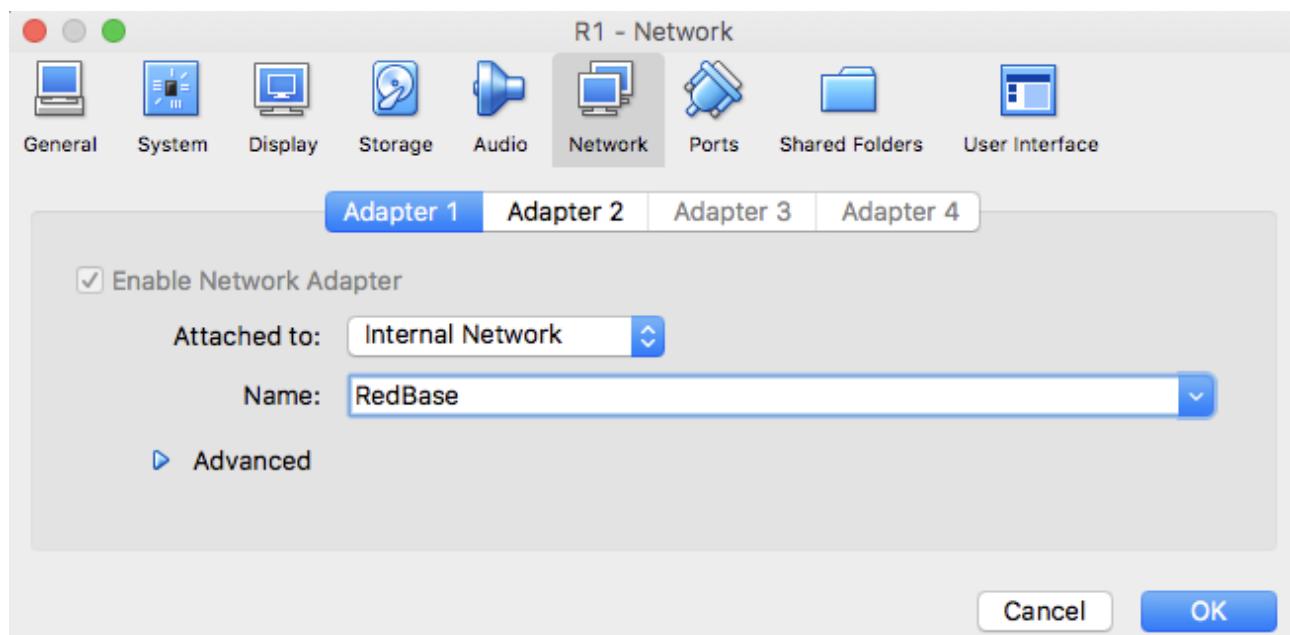


Figura 6: Interfaz de red **eth0** de R1.

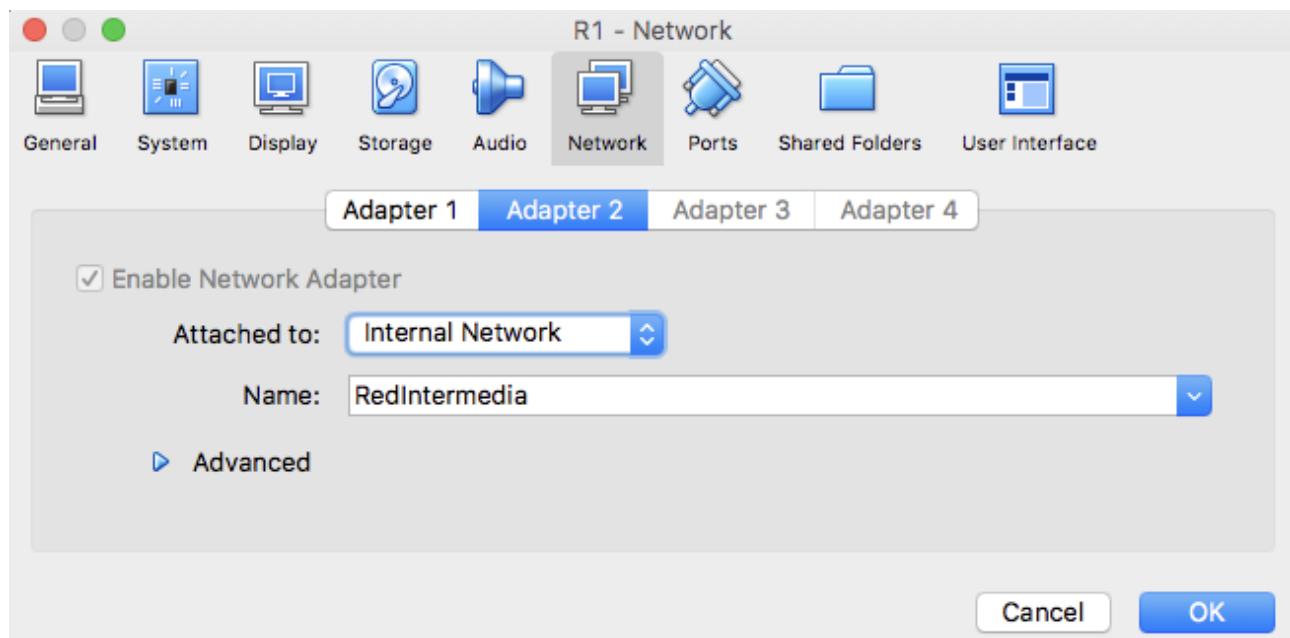


Figura 7: Interfaz de red **eth1** de R1.

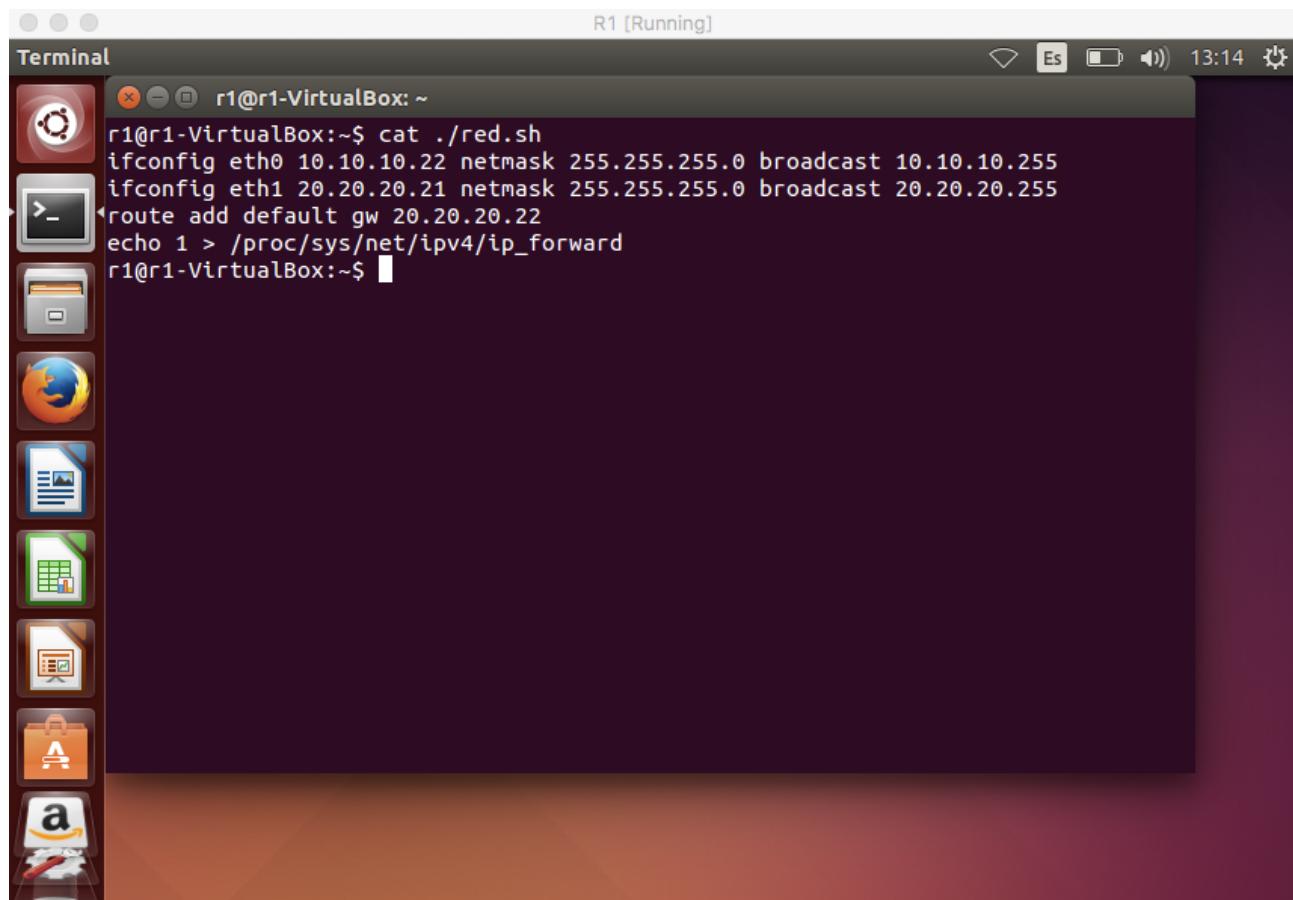


Figura 8: Script para asignar las direcciones ip a **eth0** y **eth1** en **R1**.

```
r1@r1-VirtualBox:~$ sudo ./red.sh
[sudo] password for r1:
r1@r1-VirtualBox:~$ ifconfig
eth0      Link encap:Ethernet  direcciónHW 08:00:27:f7:62:8f
          Direc. inet:10.10.10.22  Difus.:10.10.10.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:fe7:628f/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:0 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:318 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:0 (0.0 B)  TX bytes:60912 (60.9 KB)

eth1      Link encap:Ethernet  direcciónHW 08:00:27:95:01:5d
          Direc. inet:20.20.20.21  Difus.:20.20.20.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:fe95:15d/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:17 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:319 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:3421 (3.4 KB)  TX bytes:61755 (61.7 KB)

lo        Link encap:Bucle local
          Direc. inet:127.0.0.1  Másc:255.0.0.0
          Dirección inet6: ::1/128 Alcance:Anfitrión
```

Figura 9: Asignación de las direcciones ip a **eth0** y **eth1** dentro de la red base **LAN1** y de la red intermedia **LAN2** respectivamente.

Para el host **R2** sucede lo mismo que para el host anterior (**R1**), con la diferencia de que este host hace de router entre la red intermedia (**LAN2**) y la red visitada (**LAN3**). Se añade el host **R1** como router por defecto mediante el cual podrá comunicarse con la red base **LAN1**. A través de un nuevo script se asignarán las diferentes direcciones ip para cada una de las dos interfaces de red de las que dispone este host, siendo la dirección **20.20.20.22** para la interfaz de red **eth1** que se encuentra en la red intermedia (**LAN2**) y la dirección **30.30.30.22** para la interfaz de red **eth0** que se encuentra en la red visitada (**LAN3**).

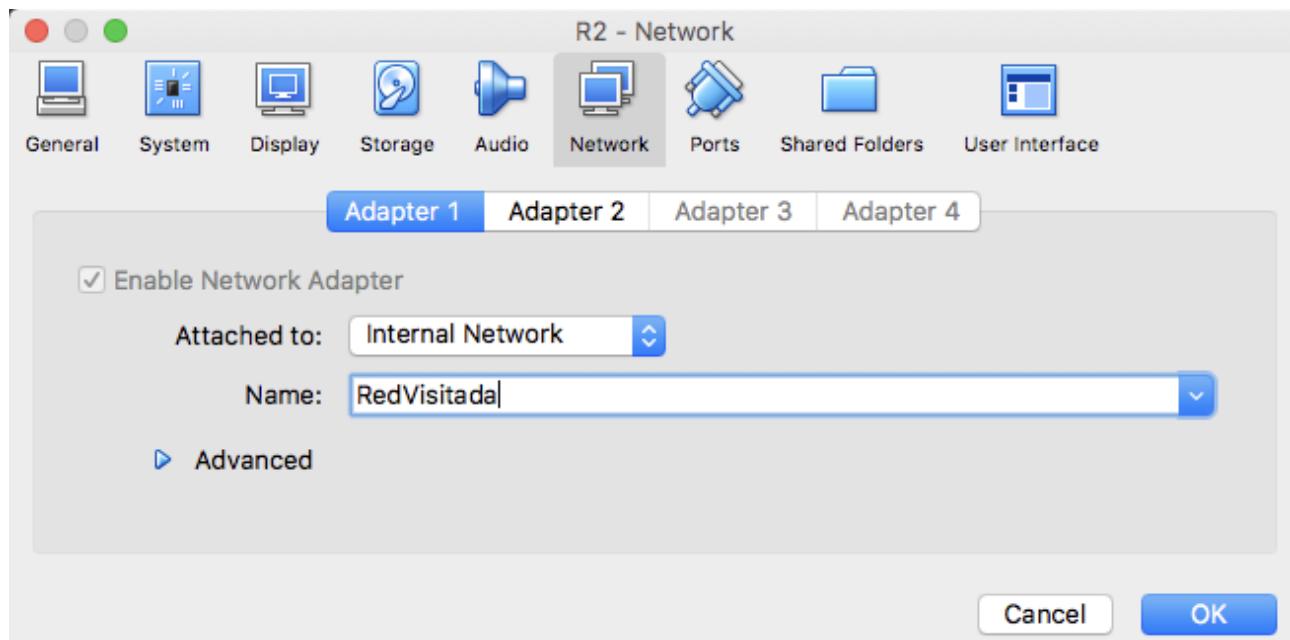


Figura 10: Interfaz de red **eth0** de R2.

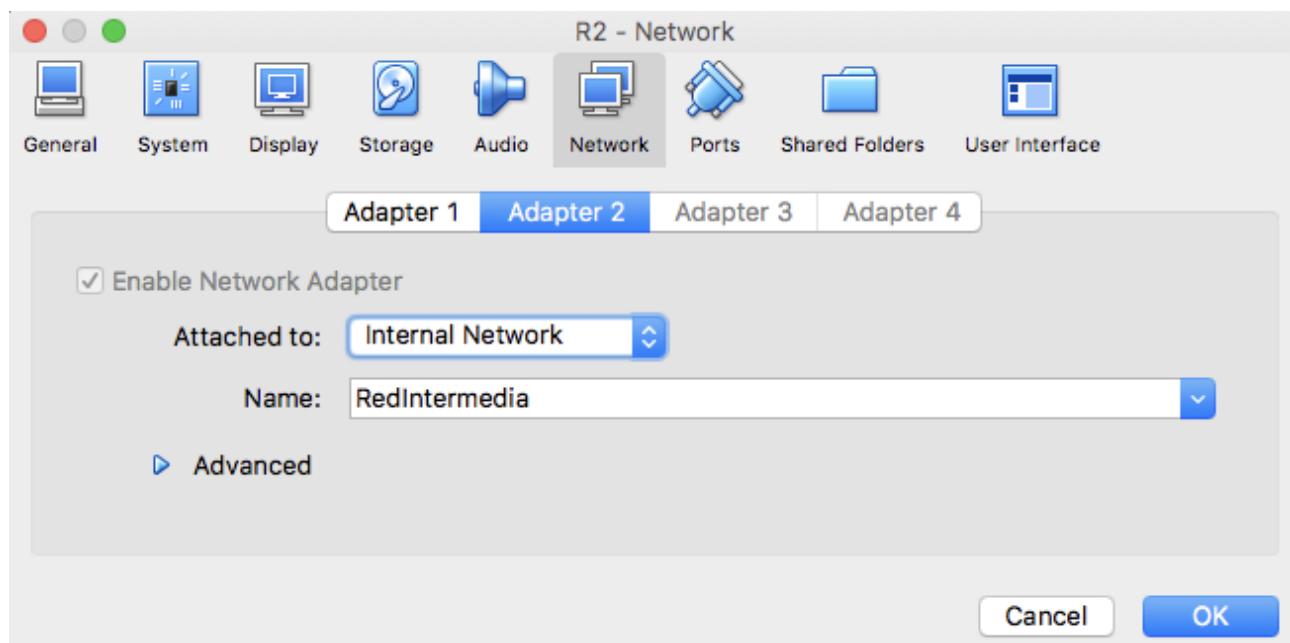


Figura 11: Interfaz de red **eth1** de R2.

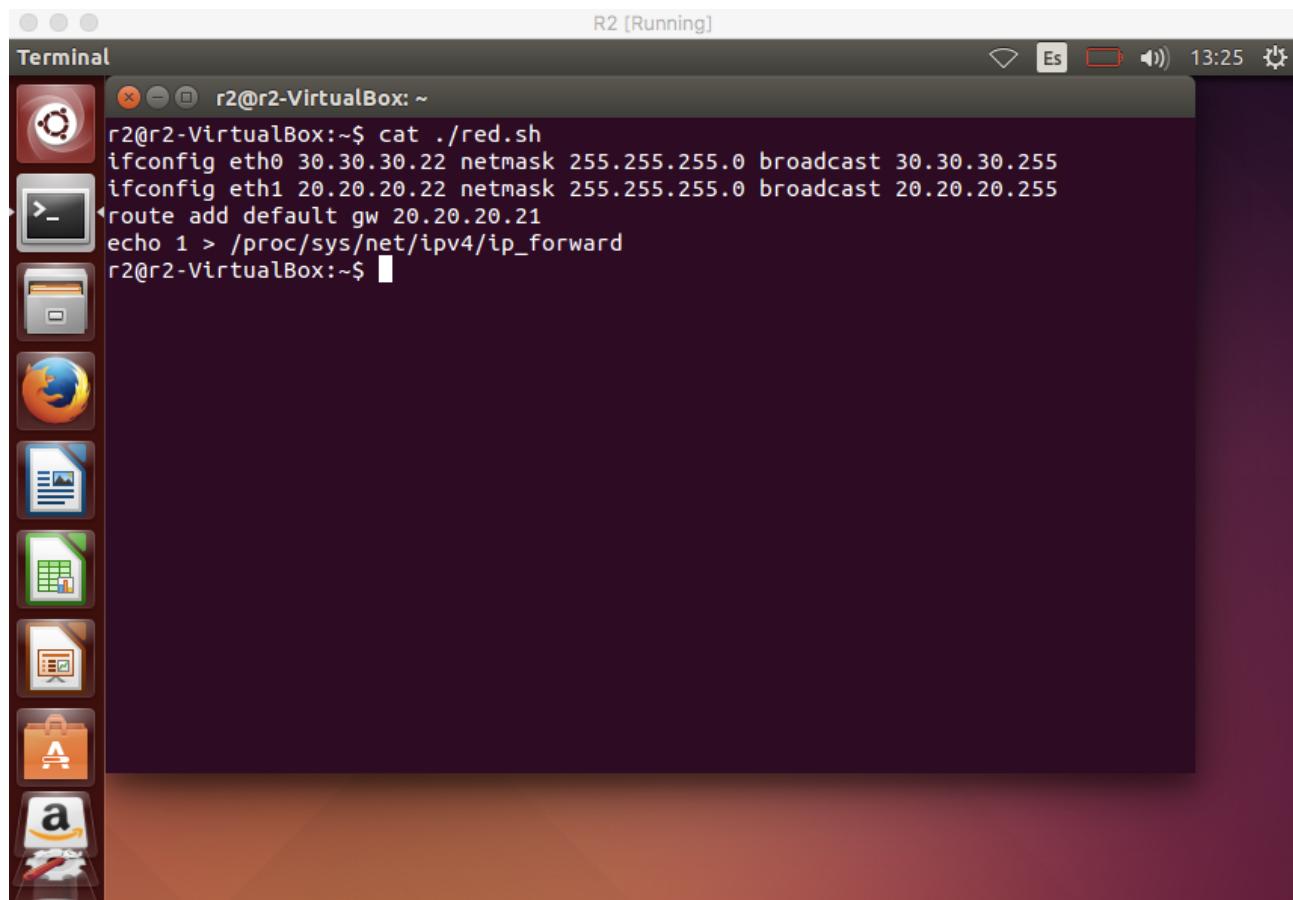


Figura 12: Script para asignar las direcciones ip a **eth0** y **eth1** en **R2**.

```
r2@r2-VirtualBox:~$ sudo ./red.sh
[sudo] password for r2:
r2@r2-VirtualBox:~$ ifconfig
eth0      Link encap:Ethernet  direcciónHW 08:00:27:40:7c:66
          Direc. inet:30.30.30.22  Difus.:30.30.30.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:fe40:7c66/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500  Métrica:1
          Paquetes RX:1 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:36 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:107 (107.0 B)  TX bytes:7036 (7.0 KB)

eth1      Link encap:Ethernet  direcciónHW 08:00:27:cf:7d:dc
          Direc. inet:20.20.20.22  Difus.:20.20.20.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:fecf:7ddc/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500  Métrica:1
          Paquetes RX:0 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:36 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:0 (0.0 B)  TX bytes:7036 (7.0 KB)

lo        Link encap:Bucle local
          Direc. inet:127.0.0.1  Másc:255.0.0.0
          Dirección inet6: ::1/128 Alcance:Anfitrión
```

Figura 13: Asignación de las direcciones ip a **eth0** y **eth1** dentro de la red visitada **LAN3** y de la red intermedia **LAN2** respectivamente.

El host **FA** tiene una configuración muy parecida a la del host **HA** ya que ambos disponen de una sola interfaz de red, salvo que para este host ésta pertenece a la red visitada (**LAN3**) y según el esquema de la **Figura1** se le asigna la dirección ip **30.30.30.1** a la interfaz de red **eth0**. El host **R2** se le asigna como router por defecto mediante el cual podrá comunicarse con la red intermedia **LAN2**.

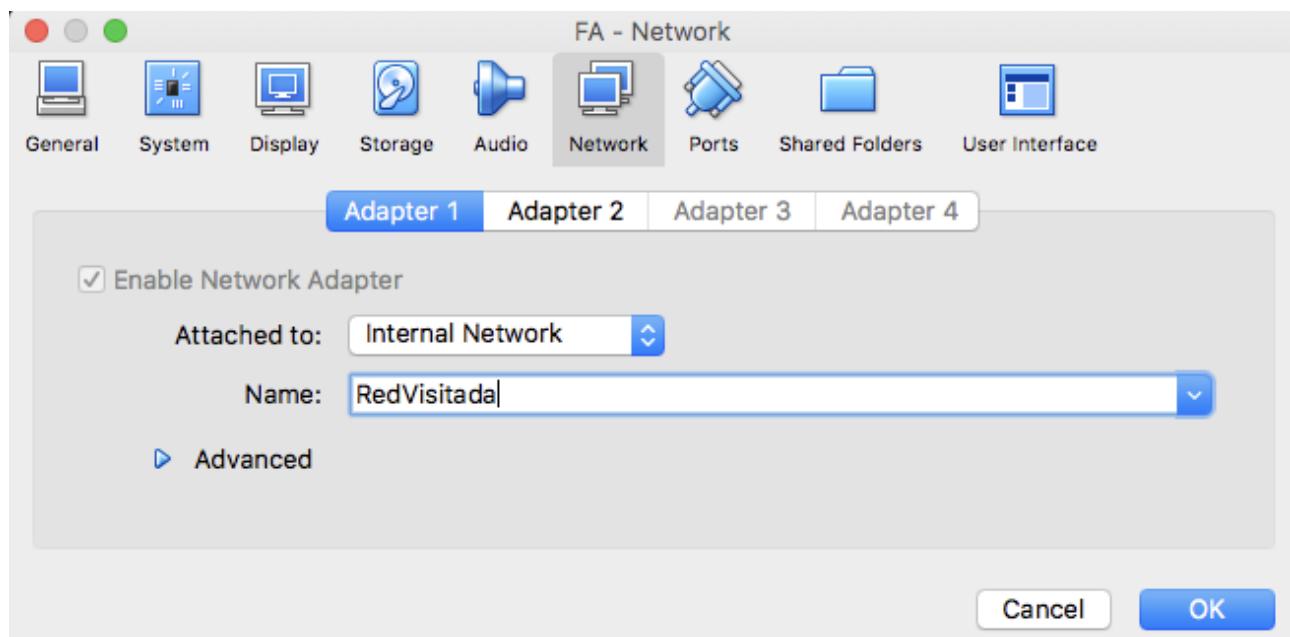


Figura 14: Interfaz de red **eth0** de FA.

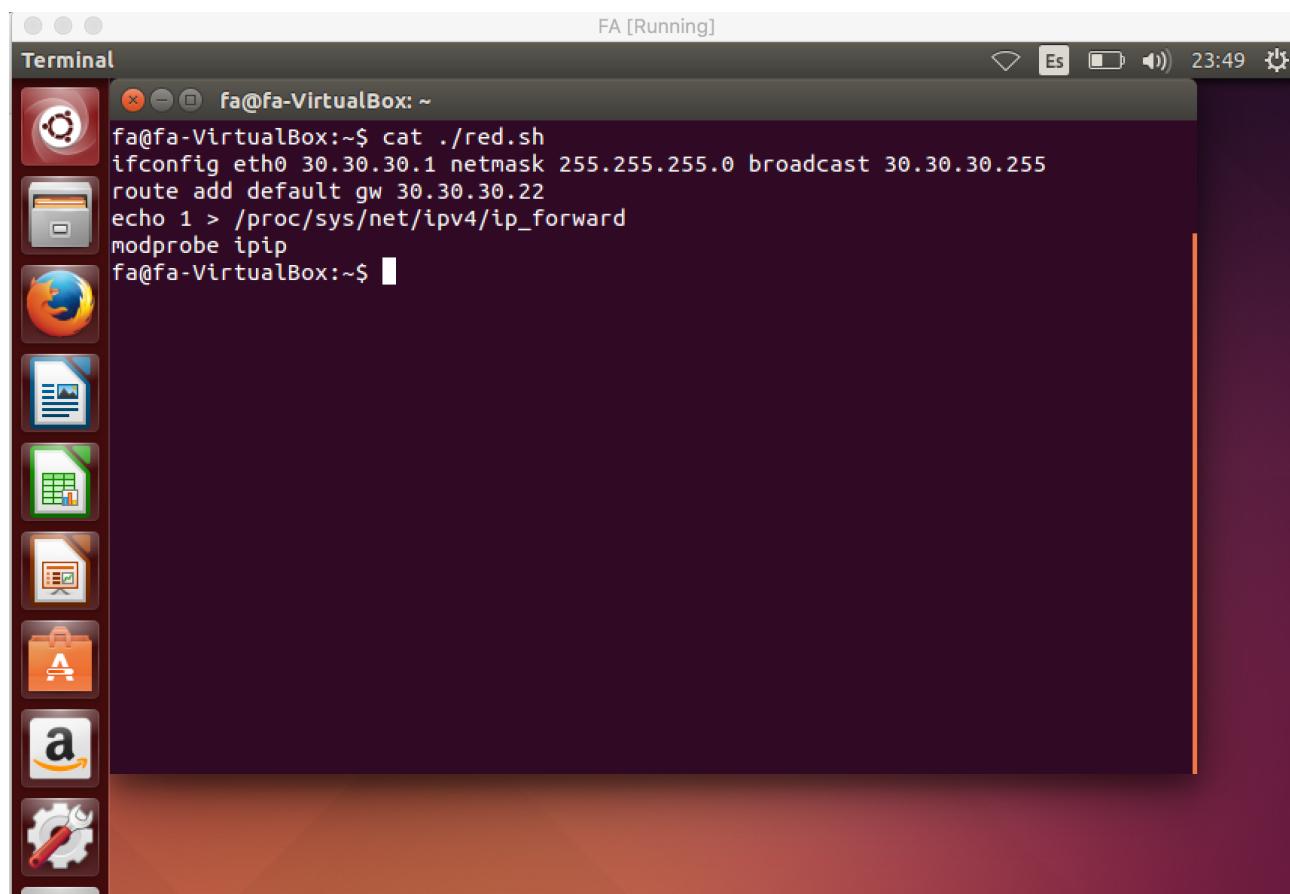
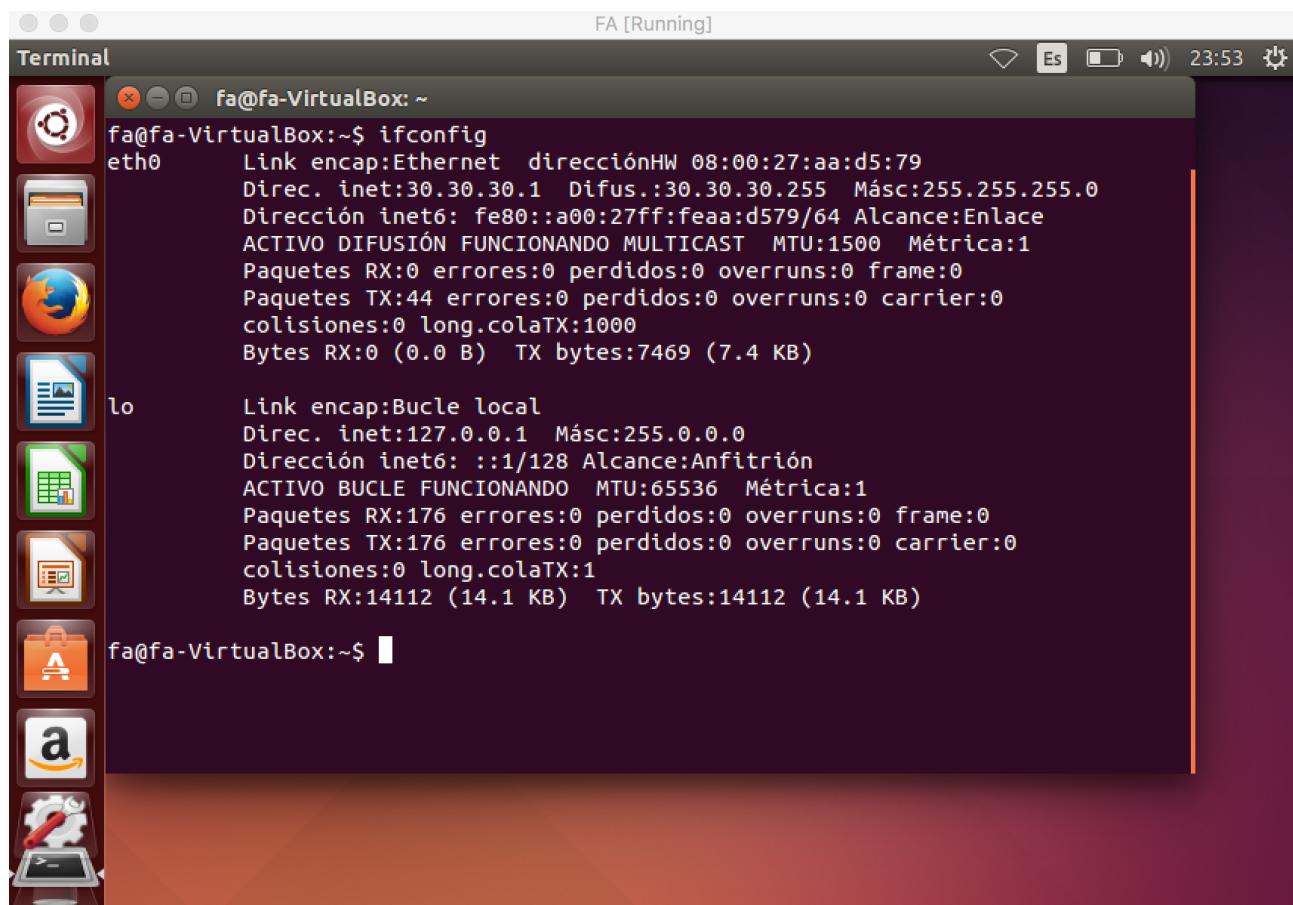


Figura 15: Script para asignar la dirección ip a **eth0** de FA.



```
fa@fa-VirtualBox:~$ ifconfig
eth0      Link encap:Ethernet  direcciónHW 08:00:27:aa:d5:79
          Direc. inet:30.30.30.1  Difus.:30.30.30.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:fea:d579/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:0 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:44 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colatX:1000
          Bytes RX:0 (0.0 B)  TX bytes:7469 (7.4 KB)

lo        Link encap:Bucle local
          Direc. inet:127.0.0.1  Másc:255.0.0.0
          Dirección inet6: ::1/128 Alcance:Anfitrión
          ACTIVO BUCLE FUNCIONANDO  MTU:65536  Métrica:1
          Paquetes RX:176 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:176 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colatX:1
          Bytes RX:14112 (14.1 KB)  TX bytes:14112 (14.1 KB)

fa@fa-VirtualBox:~$
```

Figura 16: Asignación de la dirección ip a **eth0** dentro de la red visitada **LAN3**.

Para la máquina **HostIntermedio** se realiza un proceso similar al que se ha echo con los hosts **HA** y **FA**. Este host pertenece a la red intermedia **LAN2** y dispone de una interfaz de red **eth0** a la que se le asigna la dirección ip **20.20.20.2** mediante su corresponsiente script. En este caso son dos routers los que se le añaden, uno para acceder a la red base **LAN1** y otro para acceder a la red visitada **LAN3**

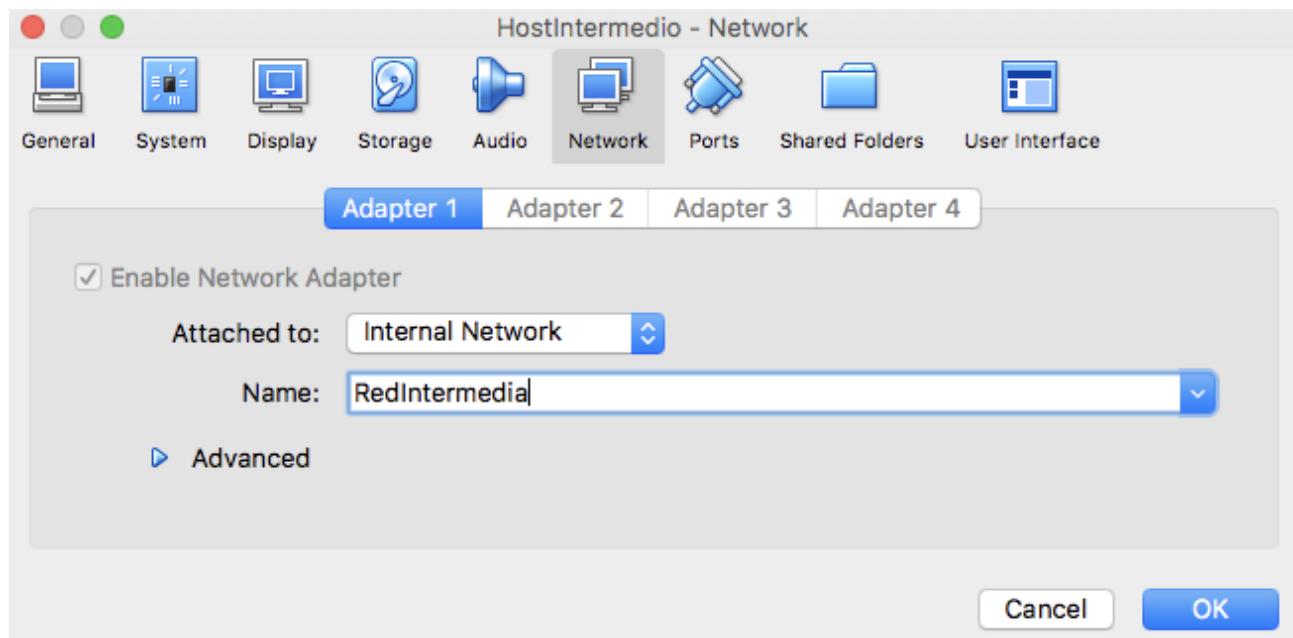


Figura 17: Interfaz de red de **eth0** de **HostIntermedio**.

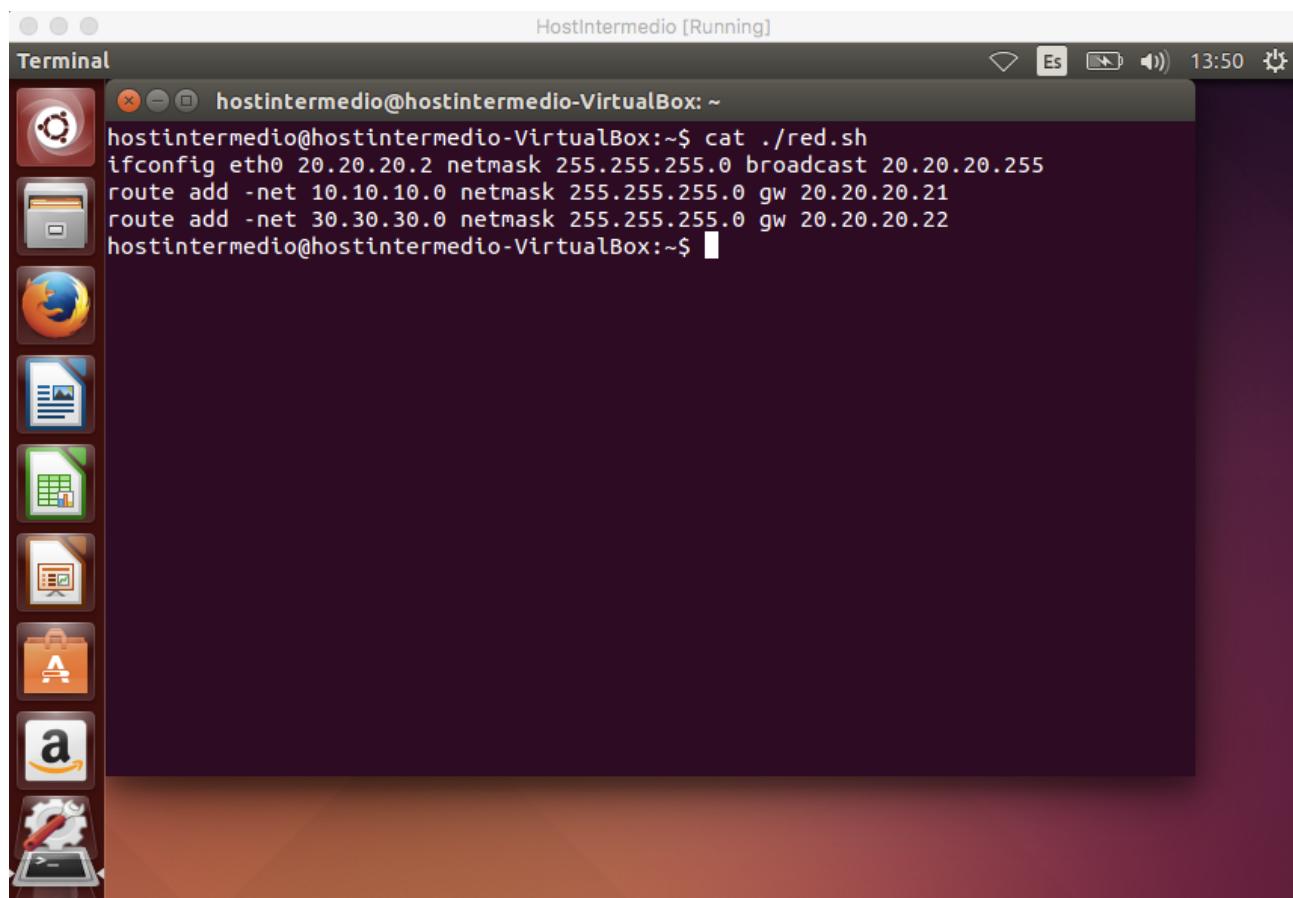


Figura 18: Script para asignar la dirección ip a **eth0** de **HostIntermedio**.

```
hostintermedio@hostintermedio-VirtualBox:~$ sudo ./red.sh
[sudo] password for hostintermedio:
hostintermedio@hostintermedio-VirtualBox:~$ ifconfig
eth0      Link encap:Ethernet  direcciónHW 08:00:27:bc:d3:33
          Direc. inet:20.20.20.2  Difus.:20.20.20.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:febc:d33/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:0 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:43 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:0 (0.0 B)  TX bytes:7905 (7.9 KB)

lo        Link encap:Bucle local
          Direc. inet:127.0.0.1  Másc:255.0.0.0
          Dirección inet6: ::1/128 Alcance:Anfitrión
          ACTIVO BUCLE FUNCIONANDO  MTU:65536  Métrica:1
          Paquetes RX:64 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:64 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1
          Bytes RX:5120 (5.1 KB)  TX bytes:5120 (5.1 KB)

hostintermedio@hostintermedio-VirtualBox:~$
```

Figura 19: Asignación de la dirección ip a **eth0** dentro de la red intermedia **LAN2**.

El host **MN** será el que hará las veces de dispositivo móvil. En un primer lugar como se muestra en la **Figura 1**, dispone de una interfaz de red **eth0** que se encuentra conectada a la red base **LAN1** con la dirección ip **10.10.10.2**. El host **R1** se le añade como router por defecto para poder comunicarse con la red intermedia **LAN2**.

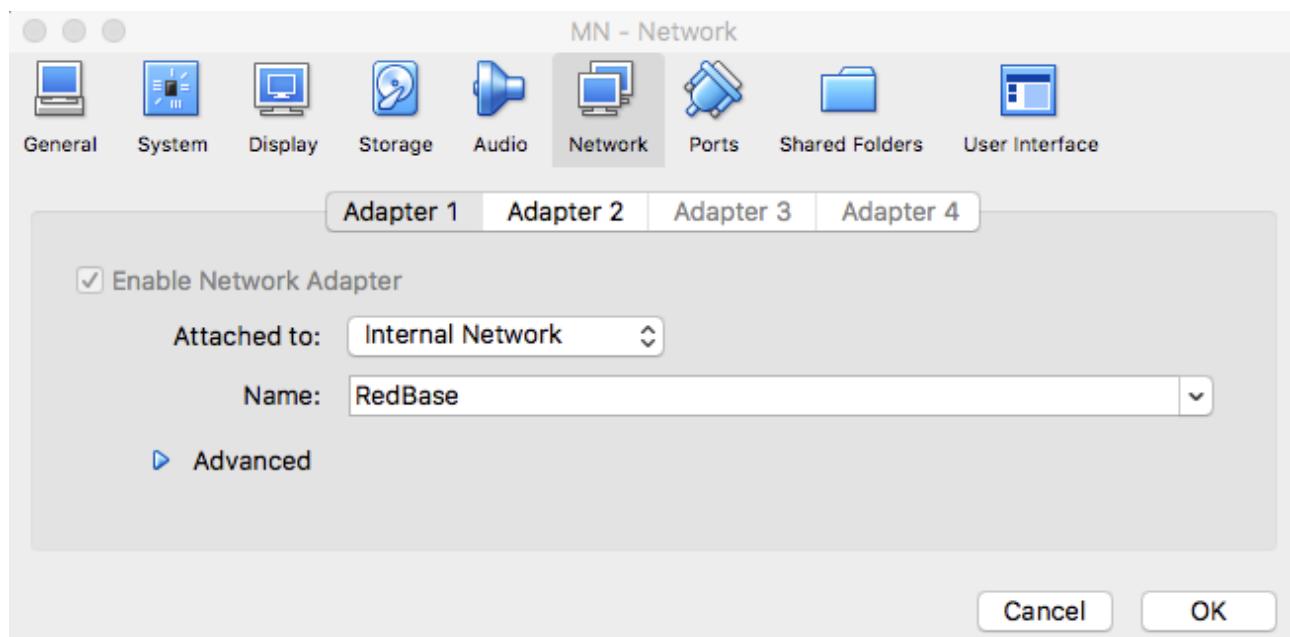


Figura 20: Interfaz de red **eth0** de MN.

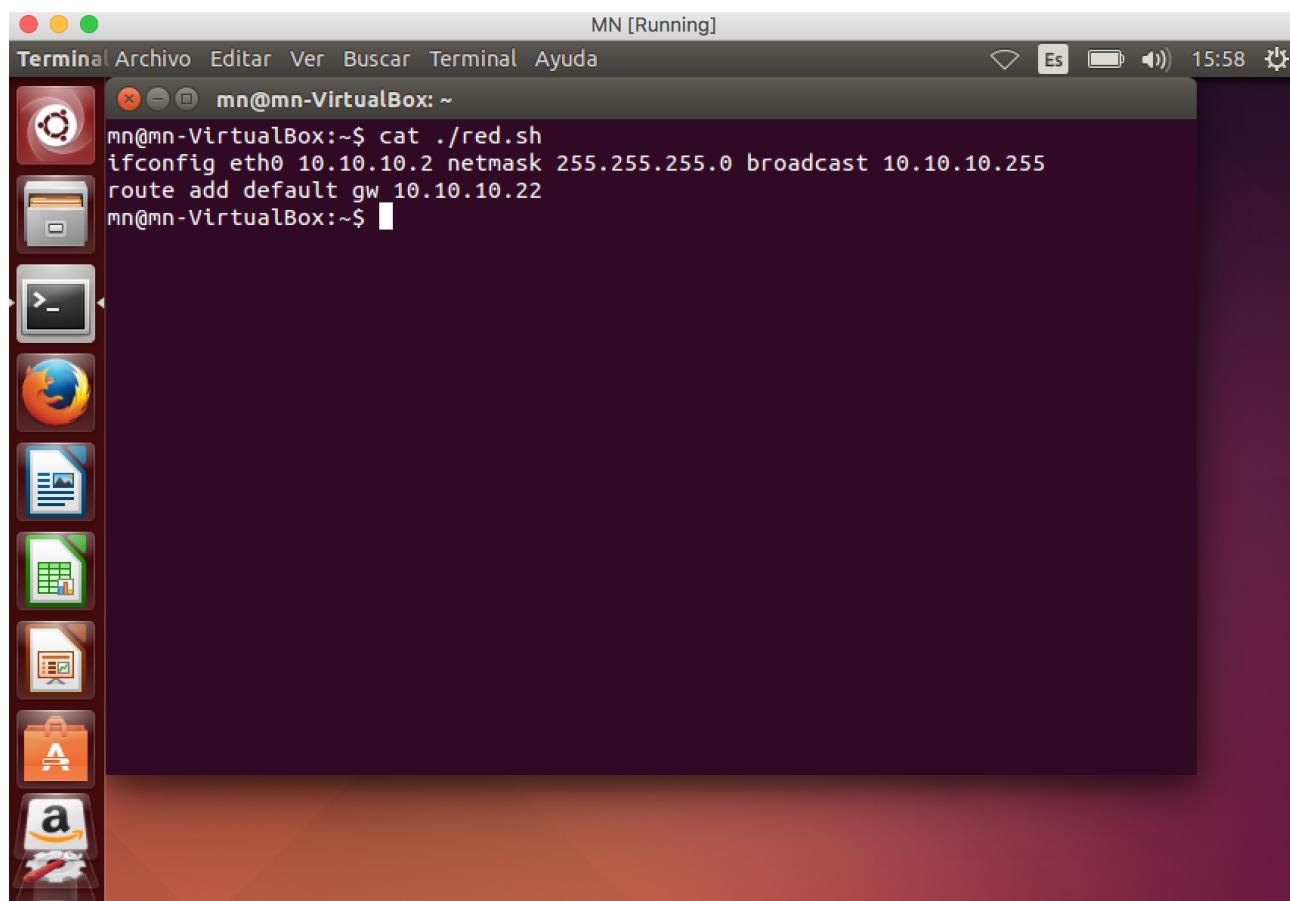


Figura 21: Script para asignar la dirección ip a **eth0** de MN.

```
MN [Running]
Terminal mn@mn-VirtualBox: ~
mn@mn-VirtualBox:~$ sudo ./red.sh
[sudo] password for mn:
mn@mn-VirtualBox:~$ ifconfig
eth0      Link encap:Ethernet  direcciónHW 08:00:27:cc:26:96
          Direc. inet:10.10.10.2  Difus.:10.10.10.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:fecc:2696/64  Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:0 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:89 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:0 (0.0 B)  TX bytes:16140 (16.1 KB)

lo        Link encap:Bucle local
          Direc. inet:127.0.0.1  Másc:255.0.0.0
          Dirección inet6: ::1/128  Alcance:Anfitrión
          ACTIVO BUCLE FUNCIONANDO  MTU:65536  Métrica:1
          Paquetes RX:152 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:152 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1
          Bytes RX:12030 (12.0 KB)  TX bytes:12030 (12.0 KB)

mn@mn-VirtualBox:~$
```

Figura 22: Asignación de la dirección ip a **eth0** dentro de la red base **LAN1** y del router por defecto **R1**.

Para comprobar que las configuraciones de los diferentes hosts se han realizado correctamente, vamos a ejecutar un **ping** desde el agente local **HA** situado en la red base **LAN1** al agente externo **FA** situado en la red visitada **LAN3**.

```
ha@ha-VirtualBox:~$ ping 30.30.30.1
PING 30.30.30.1 (30.30.30.1) 56(84) bytes of data.
64 bytes from 30.30.30.1: icmp_seq=1 ttl=62 time=0.901 ms
64 bytes from 30.30.30.1: icmp_seq=2 ttl=62 time=0.785 ms
64 bytes from 30.30.30.1: icmp_seq=3 ttl=62 time=1.11 ms
64 bytes from 30.30.30.1: icmp_seq=4 ttl=62 time=0.620 ms
64 bytes from 30.30.30.1: icmp_seq=5 ttl=62 time=0.994 ms
64 bytes from 30.30.30.1: icmp_seq=6 ttl=62 time=0.634 ms
64 bytes from 30.30.30.1: icmp_seq=7 ttl=62 time=0.638 ms
64 bytes from 30.30.30.1: icmp_seq=8 ttl=62 time=0.596 ms
64 bytes from 30.30.30.1: icmp_seq=9 ttl=62 time=0.921 ms
64 bytes from 30.30.30.1: icmp_seq=10 ttl=62 time=1.23 ms
64 bytes from 30.30.30.1: icmp_seq=11 ttl=62 time=0.806 ms
64 bytes from 30.30.30.1: icmp_seq=12 ttl=62 time=1.21 ms
64 bytes from 30.30.30.1: icmp_seq=13 ttl=62 time=1.01 ms
64 bytes from 30.30.30.1: icmp_seq=14 ttl=62 time=1.25 ms
64 bytes from 30.30.30.1: icmp_seq=15 ttl=62 time=0.793 ms
```

Figura 23: Ping de **HA** a **FA**.

Como se ha podido apreciar, la configuración se ha realizado correctamente conforme al esquema de la **Figura 1**, ya que en la anterior imagen ambas máquinas responden al **ping** desde diferentes redes.

Hemos podido ver que la mayoría de los scripts ejecutados en los hosts para asignarles una dirección ip a sus interfaces de red y un router tenían prácticamente la misma composición y estructura, sin embargo, había además otras órdenes para solamente algunos hosts. Para **R1**, **R2**, **FA** y **HA** se les añadían la orden **echo 1 > /proc/sys/net/ipv4/ip_forward** a través de la cual puedan ser capaces de redirigir información. Otra orden que en este caso se incluía en los scripts de **HA** y de **FA** les va a permitir hacer el tunneling necesario para el protocolo **IP Móvil**, el comando **modprobe ipip** es el que se debe ejecutar en cada uno de los dos agentes.

3. IP Móvil.

Para probar el funcionamiento del protocolo **IP Móvil** en la estructura de redes montada se van a realizar las siguientes tareas:

- Comprobar que los agentes **HA** y **FA** se están anunciando.
- Comprobar que el host **MN** se encuentra en la red base.
- Mostrar un ping desde el host **MN** al **HostIntermedio**.
- Comprobar que el host **MN** se encuentra en la red visitada (a través del CA de ésta).

- Comprobar que se produce el registro.
- Mostrar el tunneling.
- Comprobar que el host MN ha vuelto de nuevo a la red base.

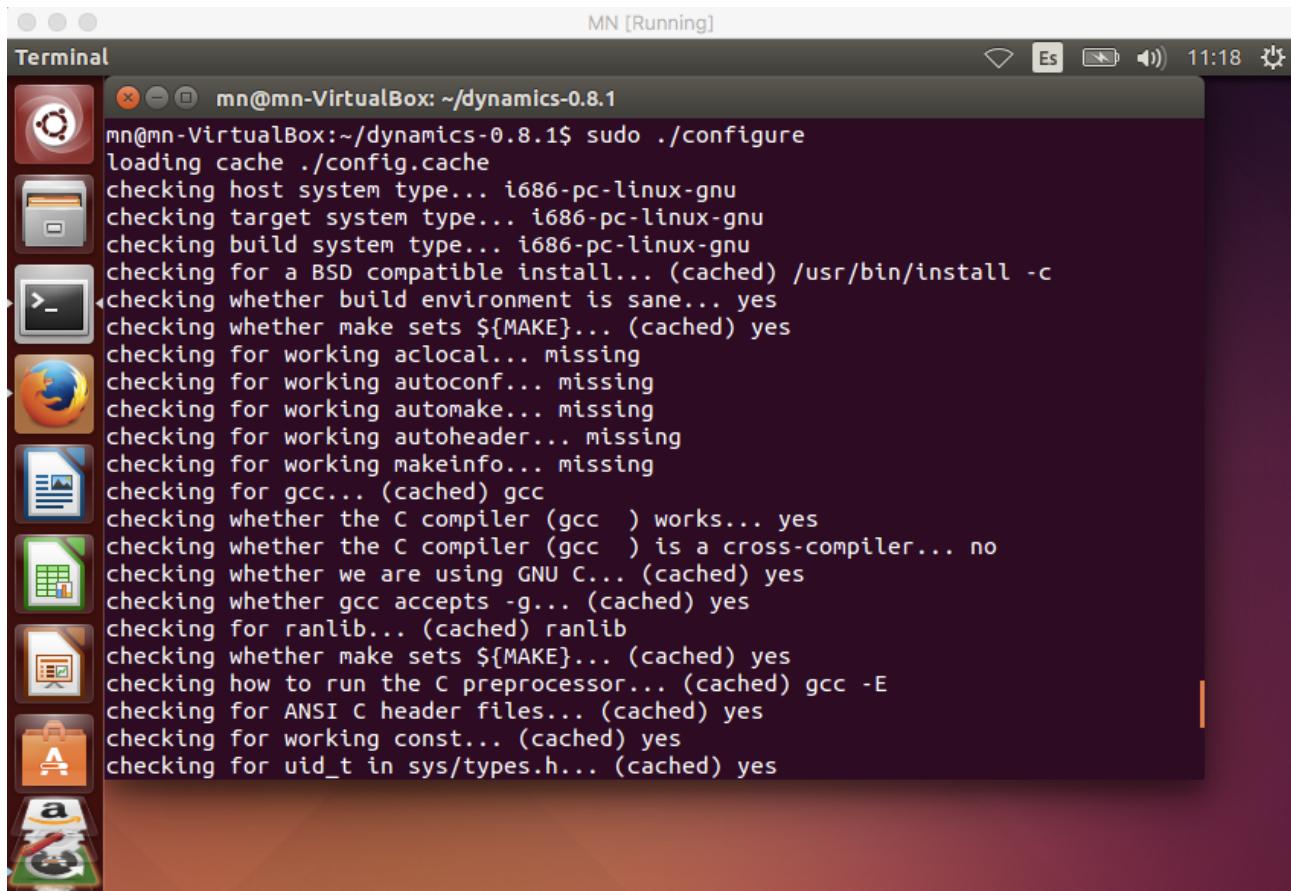
Podemos decir que la demostración del protocolo **IP Móvil** se basa en tres funcionalidades: **descubrimiento** de la dirección **CA**, **registro** de la dirección **CA** y **tunneling** de la dirección **CA**.

3.1. Instalación de IP Móvil

El primer paso a realizar es instalar el paquete de dicho protocolo en los hosts **HA**, **FA** y **MN**. La forma para instalarlo es la misma en los tres casos, y es que una vez que tengamos el paquete descomprimido y nos encontremos dentro del directorio hay que ejecutar los siguientes comandos:

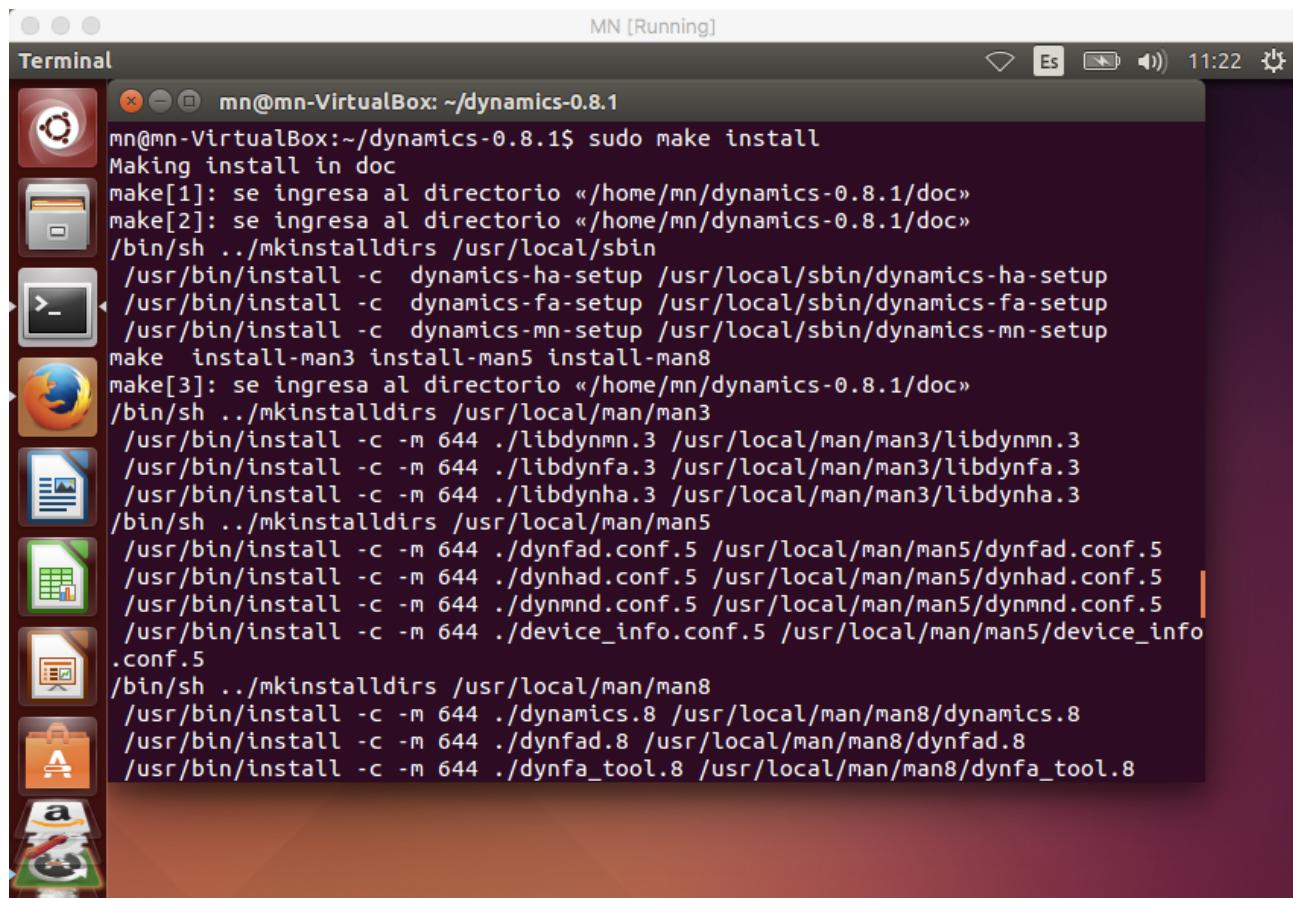
- `./configure`
- `make`
- `make install`

Es probable que se presenten algunos problemas de incompatibilidades debido a la distribución de linux que se está usando en los hosts y para la que fue desarrollada la herramienta, por lo que si en el momento de compilar nos arrojase algún tipo de error relacionado con la librería **gmp**, este se solucionaría instalando un paquete de la misma con el siguiente comando `sudo apt-get install libgmp3-dev`.



```
mn@mn-VirtualBox:~/dynamics-0.8.1$ sudo ./configure
loading cache ./config.cache
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu
checking build system type... i686-pc-linux-gnu
checking for a BSD compatible install... (cached) /usr/bin/install -c
checking whether build environment is sane... yes
checking whether make sets ${MAKE}... (cached) yes
checking for working aclocal... missing
checking for working autoconf... missing
checking for working automake... missing
checking for working autoheader... missing
checking for working makeinfo... missing
checking for gcc... (cached) gcc
checking whether the C compiler (gcc ) works... yes
checking whether the C compiler (gcc ) is a cross-compiler... no
checking whether we are using GNU C... (cached) yes
checking whether gcc accepts -g... (cached) yes
checking for ranlib... (cached) ranlib
checking whether make sets ${MAKE}... (cached) yes
checking how to run the C preprocessor... (cached) gcc -E
checking for ANSI C header files... (cached) yes
checking for working const... (cached) yes
checking for uid_t in sys/types.h... (cached) yes
```

Figura 24: Instalación IP Móvil en MN (I).



The screenshot shows a terminal window titled "MN [Running]" running on a Linux desktop. The terminal is displaying the output of a "make install" command for the "dynamics" package. The output shows the process of installing various files into the "/usr/local" directory, including configuration files like "dynfad.conf.5", "dynhad.conf.5", and "dynmnd.conf.5", as well as man pages like "dynamics.8", "dynfad.8", and "dynfa_tool.8". The desktop background is orange, and there are several icons in the dock on the left.

```
mn@mn-VirtualBox: ~/dynamics-0.8.1$ sudo make install
Making install in doc
make[1]: se ingresa al directorio «/home/mn/dynamics-0.8.1/doc»
make[2]: se ingresa al directorio «/home/mn/dynamics-0.8.1/doc»
/bin/sh ../_mkinstalldirs /usr/local/sbin
/usr/bin/install -c dynamics-ha-setup /usr/local/sbin/dynamics-ha-setup
/usr/bin/install -c dynamics-fa-setup /usr/local/sbin/dynamics-fa-setup
/usr/bin/install -c dynamics-mn-setup /usr/local/sbin/dynamics-mn-setup
make install-man3 install-man5 install-man8
make[3]: se ingresa al directorio «/home/mn/dynamics-0.8.1/doc»
/bin/sh ../_mkinstalldirs /usr/local/man/man3
/usr/bin/install -c -m 644 ./libdynmn.3 /usr/local/man/man3/libdynmn.3
/usr/bin/install -c -m 644 ./libdynfa.3 /usr/local/man/man3/libdynfa.3
/usr/bin/install -c -m 644 ./libdynha.3 /usr/local/man/man3/libdynha.3
/bin/sh ../_mkinstalldirs /usr/local/man/man5
/usr/bin/install -c -m 644 ./dynfad.conf.5 /usr/local/man/man5/dynfad.conf.5
/usr/bin/install -c -m 644 ./dynhad.conf.5 /usr/local/man/man5/dynhad.conf.5
/usr/bin/install -c -m 644 ./dynmnd.conf.5 /usr/local/man/man5/dynmnd.conf.5
/usr/bin/install -c -m 644 ./device_info.conf.5 /usr/local/man/man5/device_info.conf.5
/bin/sh ../_mkinstalldirs /usr/local/man/man8
/usr/bin/install -c -m 644 ./dynamics.8 /usr/local/man/man8/dynamics.8
/usr/bin/install -c -m 644 ./dynfad.8 /usr/local/man/man8/dynfad.8
/usr/bin/install -c -m 644 ./dynfa_tool.8 /usr/local/man/man8/dynfa_tool.8
```

Figura 25: Instalación IP Móvil en MN (II).

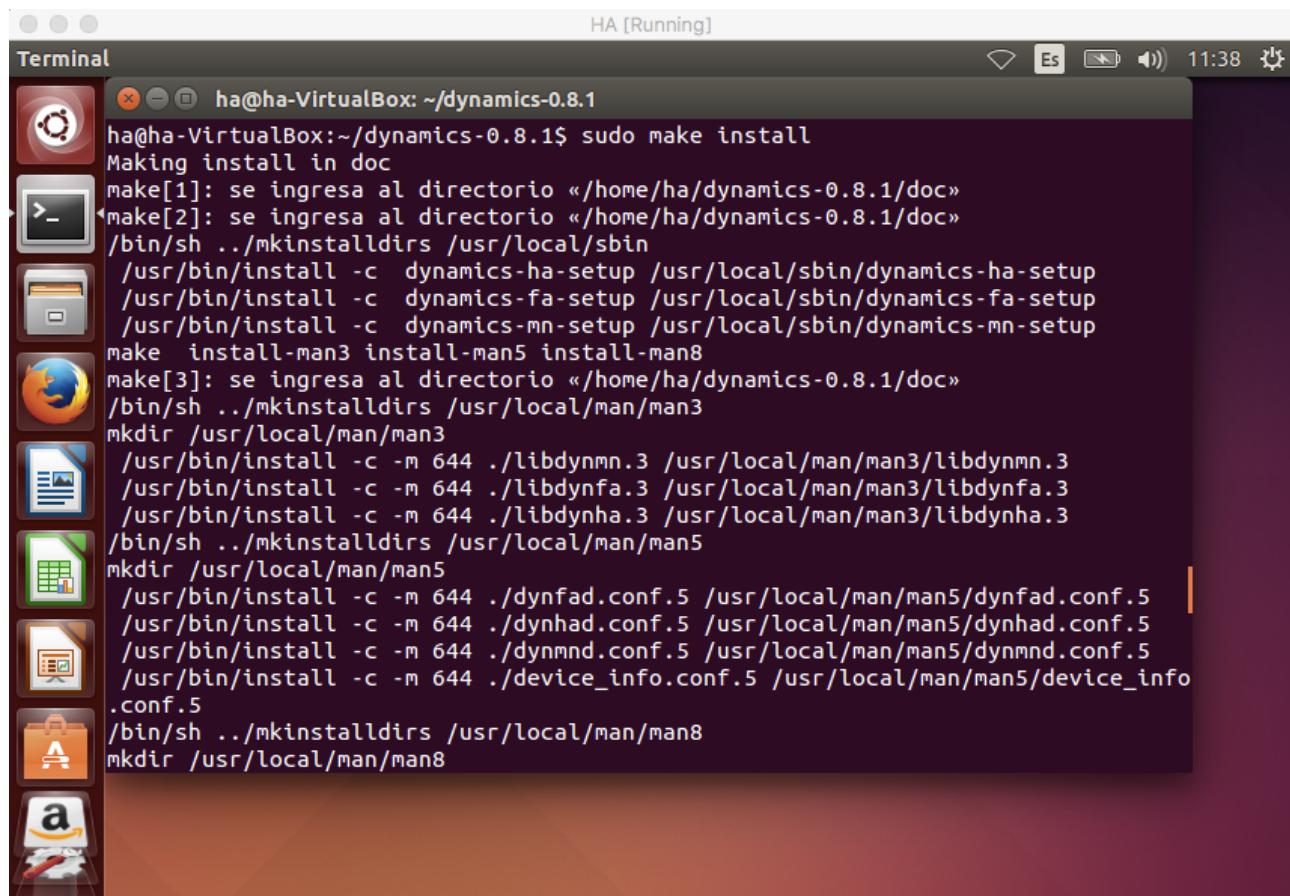
The screenshot shows a terminal window titled "HA [Running]" running on an Ubuntu desktop environment. The terminal command being executed is:

```
ha@ha-VirtualBox:~/dynamics-0.8.1$ sudo ./configure
```

The output of the command is a series of "checking" messages, indicating the configuration process for the "dynamics-0.8.1" package. The messages include:

- creating cache ./config.cache
- checking host system type... i686-pc-linux-gnu
- checking target system type... i686-pc-linux-gnu
- checking build system type... i686-pc-linux-gnu
- checking for a BSD compatible install... /usr/bin/install -c
- checking whether build environment is sane... yes
- checking whether make sets \${MAKE}... yes
- checking for working aclocal... missing
- checking for working autoconf... missing
- checking for working automake... missing
- checking for working autoheader... missing
- checking for working makeinfo... missing
- checking for gcc... gcc
- checking whether the C compiler (gcc) works... yes
- checking whether the C compiler (gcc) is a cross-compiler... no
- checking whether we are using GNU C... yes
- checking whether gcc accepts -g... yes
- checking for ranlib... ranlib
- checking whether make sets \${MAKE}... (cached) yes
- checking how to run the C preprocessor... gcc -E
- checking for ANSI C header files... yes
- checking for working const... yes
- checking for uid_t in sys/types.h... yes

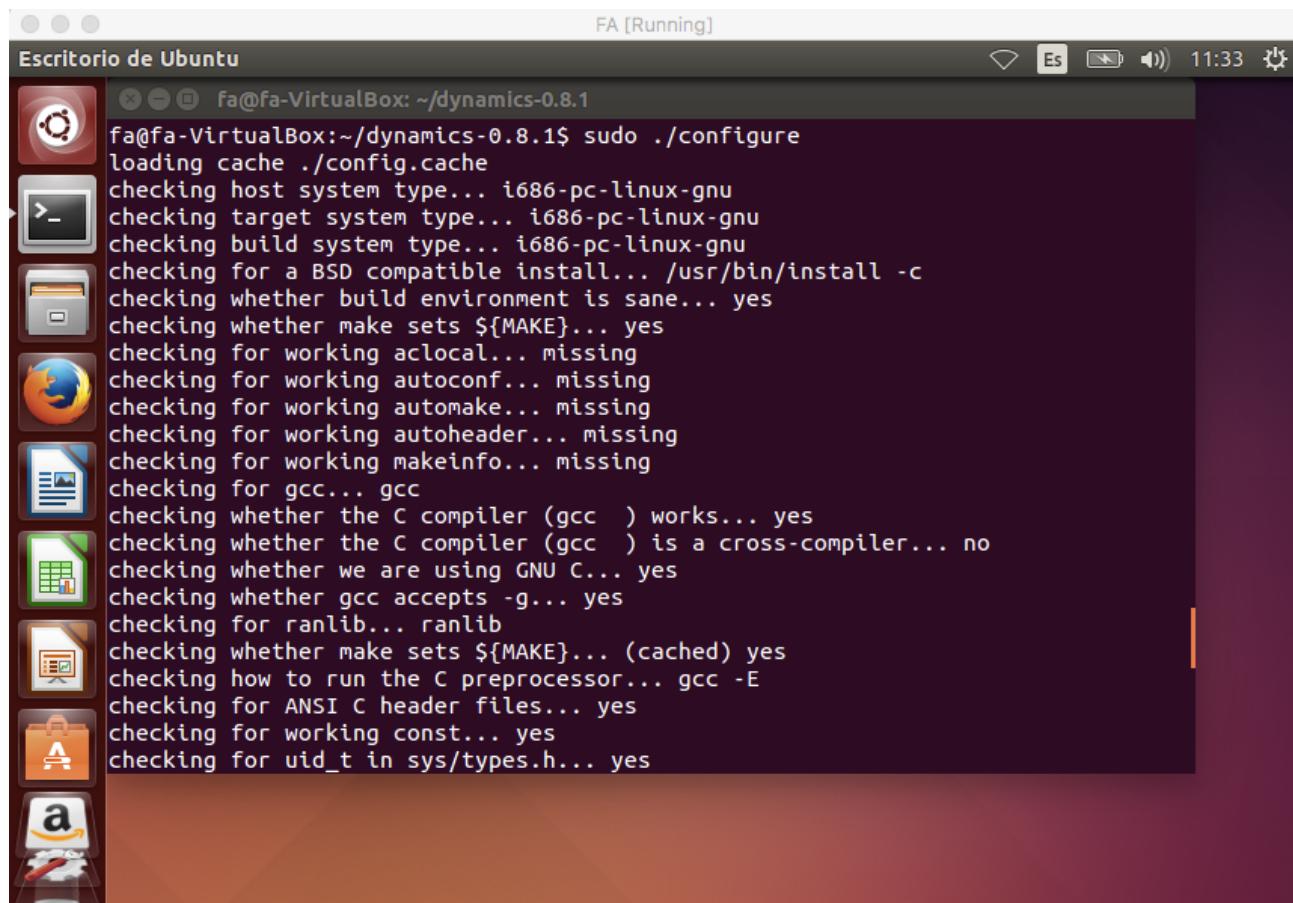
Figura 26: Instalación IP Móvil en **HA** (I).



The screenshot shows a terminal window titled "HA [Running]" running on a Linux desktop. The terminal contains the following command-line session:

```
ha@ha-VirtualBox:~/dynamics-0.8.1$ sudo make install
Making install in doc
make[1]: se ingresa al directorio «/home/ha/dynamics-0.8.1/doc»
make[2]: se ingresa al directorio «/home/ha/dynamics-0.8.1/doc»
/bin/sh ..../mkinstalldirs /usr/local/sbin
/usr/bin/install -c dynamics-ha-setup /usr/local/sbin/dynamics-ha-setup
/usr/bin/install -c dynamics-fa-setup /usr/local/sbin/dynamics-fa-setup
/usr/bin/install -c dynamics-mn-setup /usr/local/sbin/dynamics-mn-setup
make install-man3 install-man5 install-man8
make[3]: se ingresa al directorio «/home/ha/dynamics-0.8.1/doc»
/bin/sh ..../mkinstalldirs /usr/local/man/man3
mkdir /usr/local/man/man3
/usr/bin/install -c -m 644 ./libdynmn.3 /usr/local/man/man3/libdynmn.3
/usr/bin/install -c -m 644 ./libdynfa.3 /usr/local/man/man3/libdynfa.3
/usr/bin/install -c -m 644 ./libdynha.3 /usr/local/man/man3/libdynha.3
/bin/sh ..../mkinstalldirs /usr/local/man/man5
mkdir /usr/local/man/man5
/usr/bin/install -c -m 644 ./dynfad.conf.5 /usr/local/man/man5/dynfad.conf.5
/usr/bin/install -c -m 644 ./dynhad.conf.5 /usr/local/man/man5/dynhad.conf.5
/usr/bin/install -c -m 644 ./dynmnd.conf.5 /usr/local/man/man5/dynmnd.conf.5
/usr/bin/install -c -m 644 ./device_info.conf.5 /usr/local/man/man5/device_info
.conf.5
/bin/sh ..../mkinstalldirs /usr/local/man/man8
mkdir /usr/local/man/man8
```

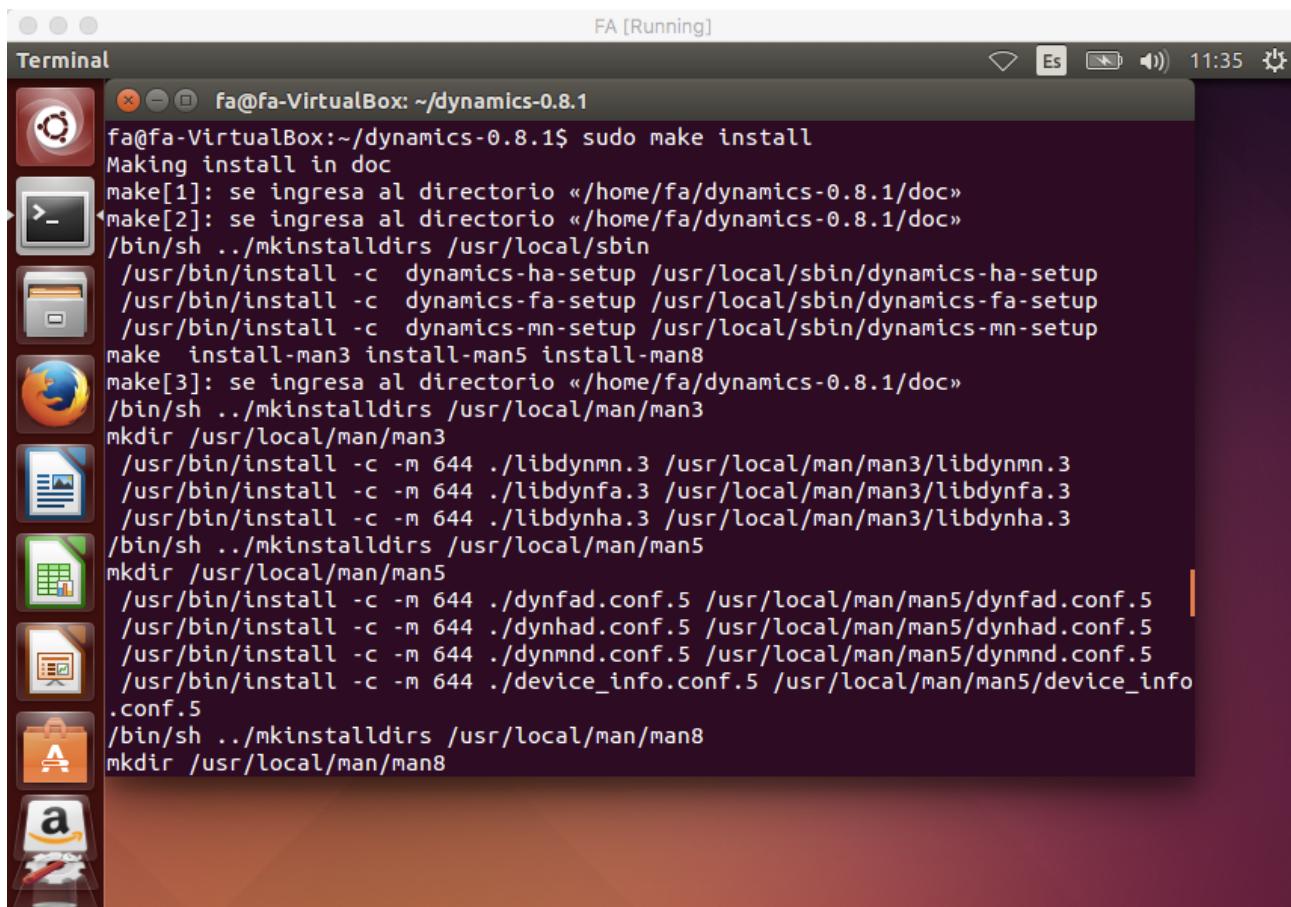
Figura 27: Instalación IP Móvil en **HA** (II).



The screenshot shows a terminal window titled "FA [Running]" running on an Ubuntu desktop environment. The terminal output is as follows:

```
fa@fa-VirtualBox:~/dynamics-0.8.1$ sudo ./configure
loading cache ./config.cache
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu
checking build system type... i686-pc-linux-gnu
checking for a BSD compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking whether make sets ${MAKE}... yes
checking for working aclocal... missing
checking for working autoconf... missing
checking for working automake... missing
checking for working autoheader... missing
checking for working makeinfo... missing
checking for gcc... gcc
checking whether the C compiler (gcc ) works... yes
checking whether the C compiler (gcc ) is a cross-compiler... no
checking whether we are using GNU C... yes
checking whether gcc accepts -g... yes
checking for ranlib... ranlib
checking whether make sets ${MAKE}... (cached) yes
checking how to run the C preprocessor... gcc -E
checking for ANSI C header files... yes
checking for working const... yes
checking for uid_t in sys/types.h... yes
```

Figura 28: Instalación IP Móvil en FA (I).



```

fa@fa-VirtualBox:~/dynamics-0.8.1$ sudo make install
Making install in doc
make[1]: se ingresa al directorio «/home/fa/dynamics-0.8.1/doc»
make[2]: se ingresa al directorio «/home/fa/dynamics-0.8.1/doc»
/bin/sh ..../mkinstalldirs /usr/local/sbin
/usr/bin/install -c dynamics-ha-setup /usr/local/sbin/dynamics-ha-setup
/usr/bin/install -c dynamics-fa-setup /usr/local/sbin/dynamics-fa-setup
/usr/bin/install -c dynamics-mn-setup /usr/local/sbin/dynamics-mn-setup
make install-man3 install-man5 install-man8
make[3]: se ingresa al directorio «/home/fa/dynamics-0.8.1/doc»
/bin/sh ..../mkinstalldirs /usr/local/man/man3
mkdir /usr/local/man/man3
/usr/bin/install -c -m 644 ./libdynmn.3 /usr/local/man/man3/libdynmn.3
/usr/bin/install -c -m 644 ./libdynfa.3 /usr/local/man/man3/libdynfa.3
/usr/bin/install -c -m 644 ./libdynha.3 /usr/local/man/man3/libdynha.3
/bin/sh ..../mkinstalldirs /usr/local/man/man5
mkdir /usr/local/man/man5
/usr/bin/install -c -m 644 ./dynfad.conf.5 /usr/local/man/man5/dynfad.conf.5
/usr/bin/install -c -m 644 ./dynhad.conf.5 /usr/local/man/man5/dynhad.conf.5
/usr/bin/install -c -m 644 ./dynamnd.conf.5 /usr/local/man/man5/dynamnd.conf.5
/usr/bin/install -c -m 644 ./device_info.conf.5 /usr/local/man/man5/device_info
.conf.5
/bin/sh ..../mkinstalldirs /usr/local/man/man8
mkdir /usr/local/man/man8

```

Figura 29: Instalación IP Móvil en FA (II).

Con el paquete del protocolo **IP Móvil** instalado en los tres hosts, toca configurar algunos parámetros, para ello hay que dirigirse a la ruta **/usr/local/etc/** dentro de cada host y editar los ficheros **dynamnd.conf** para el host **MN**, **dynhad.conf** para el host **HA** y **dynfad.conf** para el host **FA** realizando las siguientes modificaciones.

En el fichero **dynamnd.conf** se editarán los siguientes parámetros:

- MNHomeIPAddress: 10.10.10.2
- HAIPAddress: 10.10.10.1
- HomeNetPrefix: 10.10.10.0/24

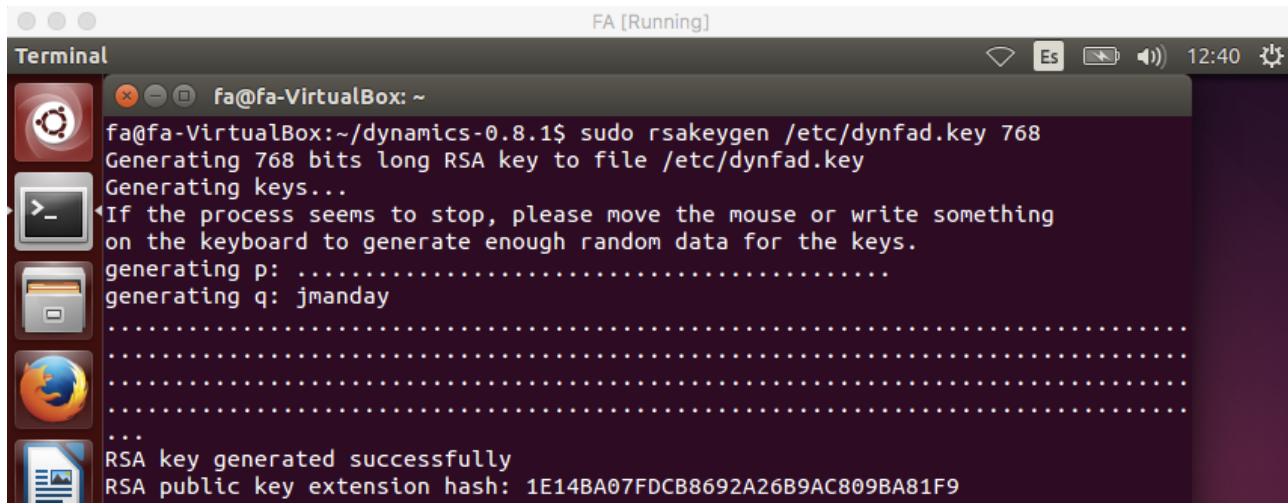
Para el fichero **dynhad.conf** se editarán los siguientes parámetros:

- AUTHORIZEDLIST-BEGIN (habilita a cualquier dispositivo de la red a solicitar los servicios del agente local)
 - 1000 10.10.10.0/24
- AUTHORIZEDLIST-END
- INTERFACES-BEGIN (interfaz de red a través de la cual se accede a la red)
 - eth0 1 1 10
- INTERFACES-END

Por último, en el fichero **dynfad.conf** se modificarán los siguientes parámetros:

- INTERFACES-BEGIN (interfaz de red a través de la cual se accede a la red)
 - eth0 1 1 30
- INTERFACES-END
- HighestFAIPAddress/UpperFAIPAddress: 30.30.30.1

Es necesario crear el fichero **/etc/dynfad.key** en el agente externo **FA**, para ello ejecutamos el comando **rsakeygen /etc/dynfad.key 768** en dicho host.

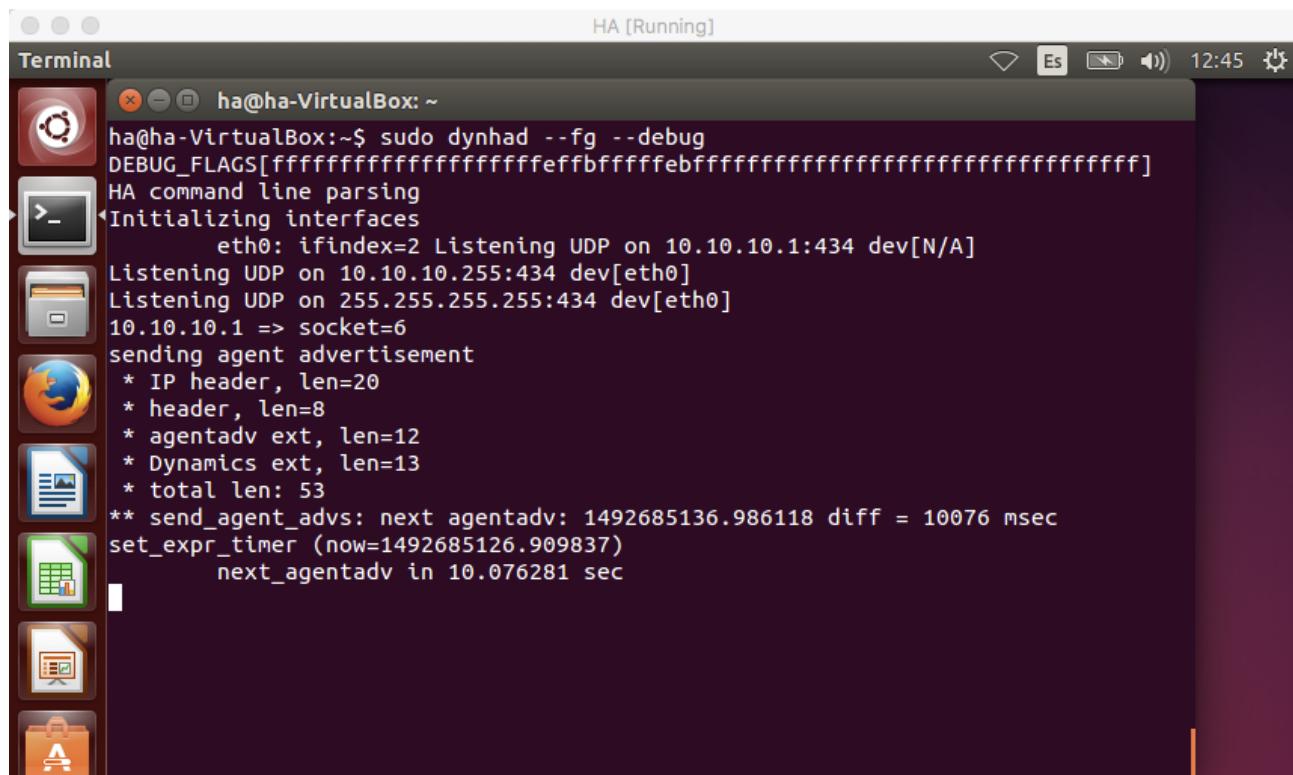


```
fa@fa-VirtualBox:~/dynamics-0.8.1$ sudo rsakeygen /etc/dynfad.key 768
Generating 768 bits long RSA key to file /etc/dynfad.key
Generating keys...
If the process seems to stop, please move the mouse or write something
on the keyboard to generate enough random data for the keys.
generating p: .....
generating q: jmanday
.....
.....
.....
RSA key generated successfully
RSA public key extension hash: 1E14BA07FDCB8692A26B9AC809BA81F9
```

Figura 30: Semilla para la generación del fichero en **FA**.

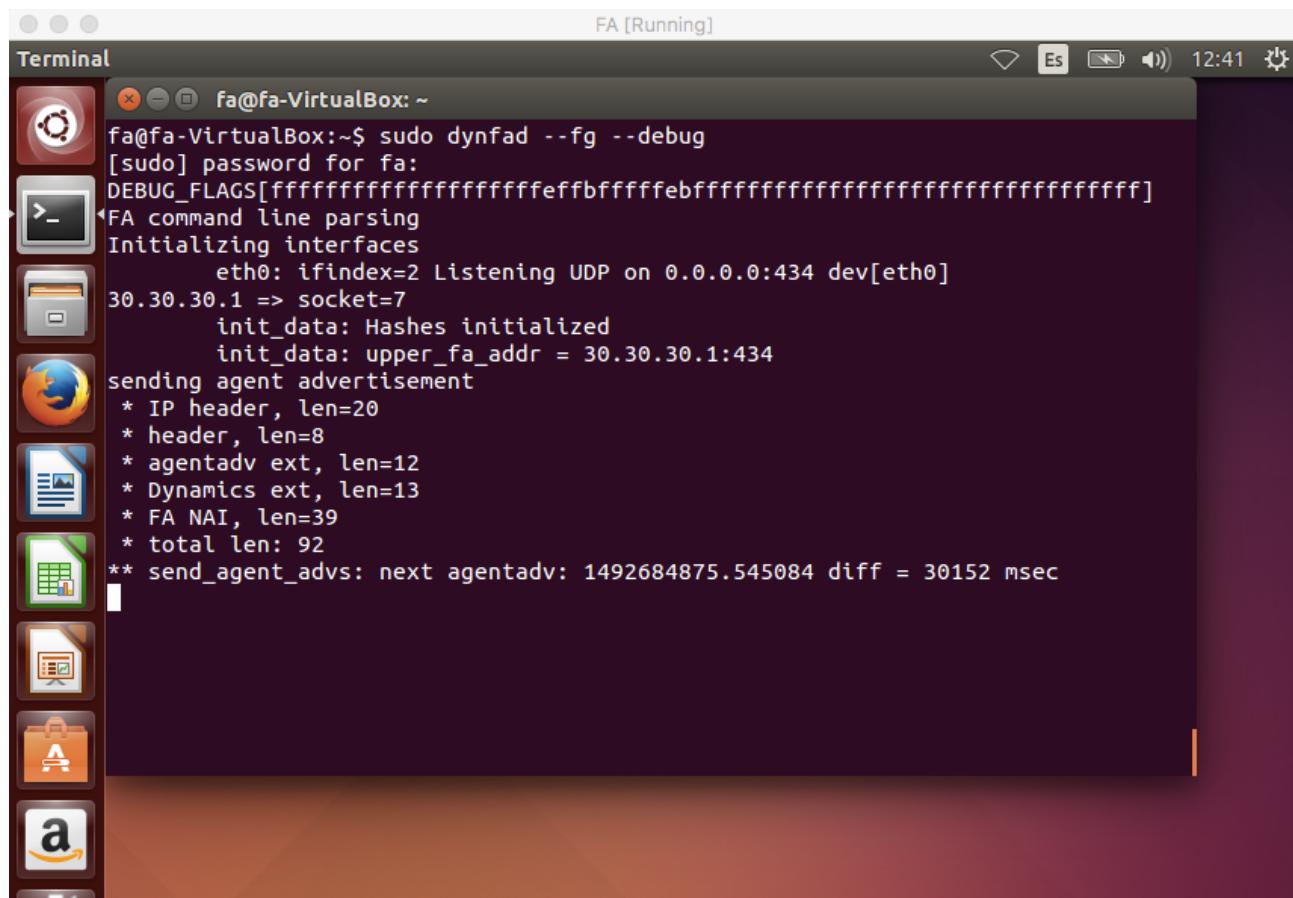
3.2. Simulación de IP Móvil

Una vez que se ha configurado todo lo referente al protocolo **IP Móvil** en los correspondientes hosts, vamos a comenzar a probar el funcionamiento de este. Lo primero que se va a realizar es comprobar que los agentes **HA** y **FA** se están anunciando, para ello es necesario ejecutar el comando **dynhad -fg -debug** en el agente **HA** y el comando **dynfad -fg -debug** en el agente **FA**.



```
ha@ha-VirtualBox:~$ sudo dynhad --fg --debug
DEBUG_FLAGS[fffffffffffffeffbfffffebfffff]
HA command line parsing
Initializing interfaces
    eth0: ifindex=2 Listening UDP on 10.10.10.1:434 dev[N/A]
Listening UDP on 10.10.10.255:434 dev[eth0]
Listening UDP on 255.255.255.255:434 dev[eth0]
10.10.10.1 => socket=6
sending agent advertisement
    * IP header, len=20
    * header, len=8
    * agentadv ext, len=12
    * Dynamics ext, len=13
    * total len: 53
** send_agent_advs: next agentadv: 1492685136.986118 diff = 10076 msec
set_expr_timer (now=1492685126.909837)
            next_agentadv in 10.076281 sec
```

Figura 31: Descubrimiento de **HA**.



```
fa@fa-VirtualBox:~$ sudo dynfad --fg --debug
[sudo] password for fa:
DEBUG_FLAGS[fffffffffffffffffffffeffbfffebfffffffffffff]
FA command line parsing
Initializing interfaces
    eth0: ifindex=2 Listening UDP on 0.0.0.0:434 dev[eth0]
30.30.30.1 => socket=7
        init_data: Hashes initialized
        init_data: upper_fa_addr = 30.30.30.1:434
sending agent advertisement
 * IP header, len=20
 * header, len=8
 * agentadv ext, len=12
 * Dynamics ext, len=13
 * FA NAI, len=39
 * total len: 92
** send_agent_advs: next agentadv: 1492684875.545084 diff = 30152 msec
```

Figura 32: Descubrimiento de **FA**.

Con los dos agentes anunciándose en sus respectivas redes, vamos a comprobar que el host **MN** se encuentra en la red base. Esto se puede ver en la siguiente imagen, donde el host **MN** esta recibiendo paquetes ICMP desde el agente local **HA** por lo que podemos determinar que el host **MN** se encuentra 'at home'.

```

Terminal
mn@mn-VirtualBox: ~
handle_icmp_adv: ifindex=2 hatype=0x0100 pkttype=0 halen=6 addr=08:00:27:ba:9b:8
c:00:00
ICMP from 10.10.10.1, len=53
* Dynamics agentadv ext
    agent adv (sent to 10.10.10.2)
Added new agentadv entry for 10.10.10.1
    now=1492685331.650833, expire=1492685362.650833
Home advertisement
    from our own HA - MN is at home
Deregistering due to the heard own HA agent advertisement
close_for_home(0)
close_for_home - not registered, but trying to deregister possible old bindings
at HA
FA[10.10.10.1] reg_retry_time: 0 => 1 (close_for_home)
proxyarp_gratuitous: sending gratuitous ARP - IPAddr: 10.10.10.2, HWaddr: 08:00:
27:CC:26:96, family=1
Setting in_use=0 for FA 10.10.10.1 (send reg)
Sending registration message:
* header
FA did not advertise support for Dynamics extensions - not using them
* mh_auth
Registration Request
    type 1, opts 0, lifetime 0
    home_addr 10.10.10.2, ha_addr 10.10.10.1

```

Figura 33: Comprobando que MN está en la red base.

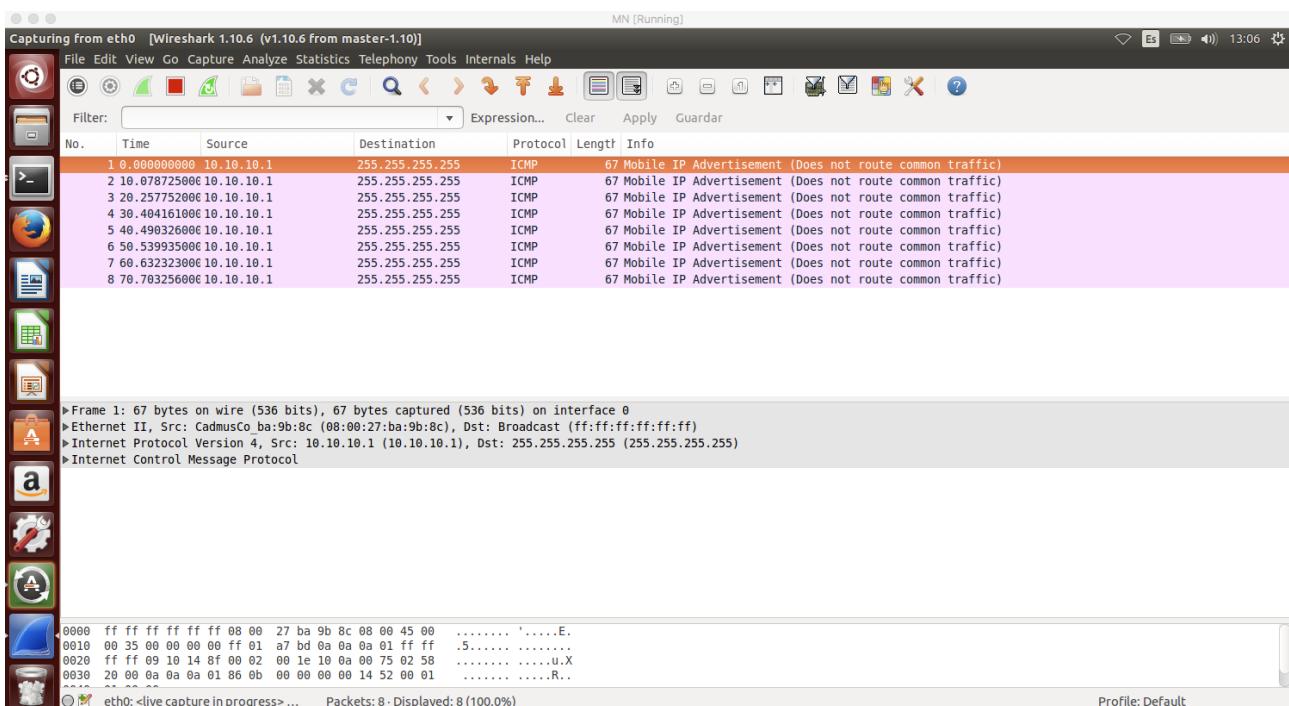


Figura 34: Comprobando que MN recibe los paquetes del agente HA.

Si desde el host **MN** hacemos un **traceroute** para conocer la ruta que hay que seguir para llegar al host **FA** nos mostrará la siguiente secuencia de saltos entre redes.

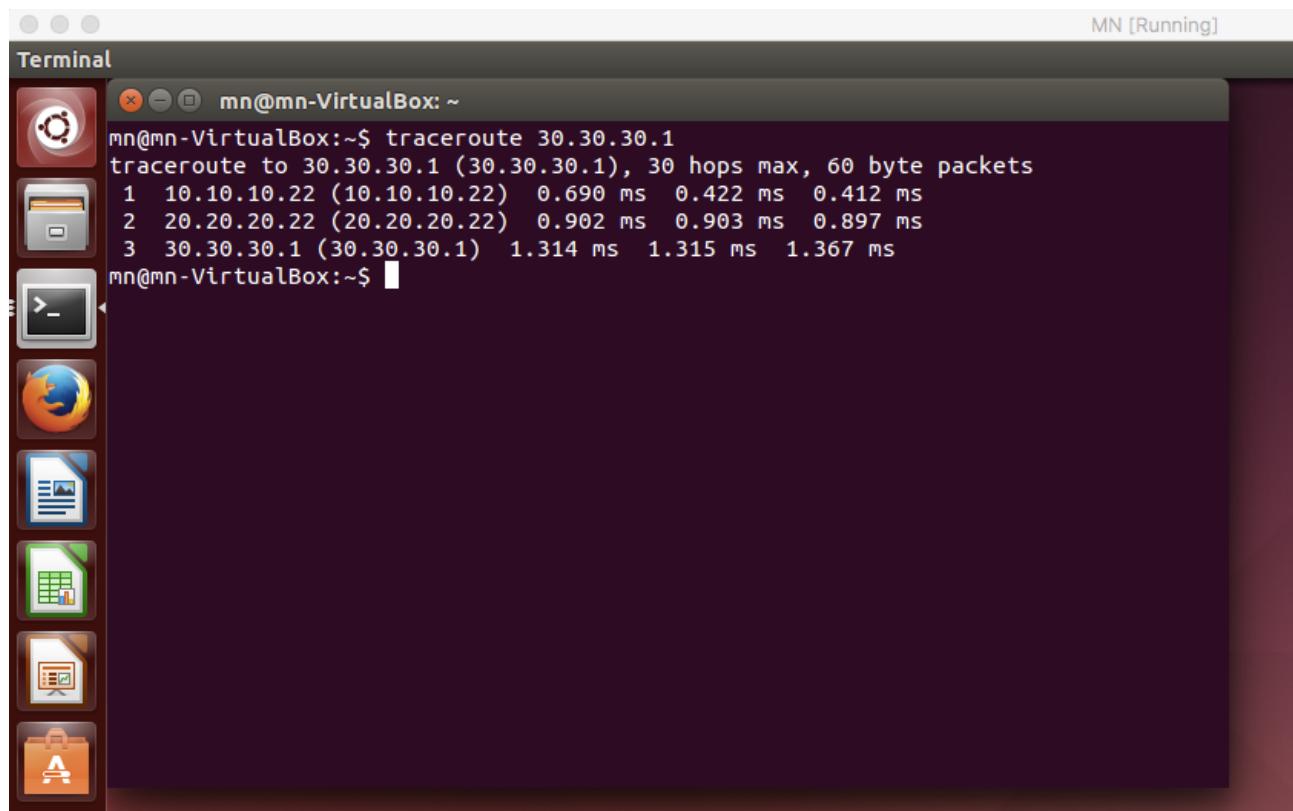


Figura 35: Traceroute para llegar a FA desde la red base LAN1.

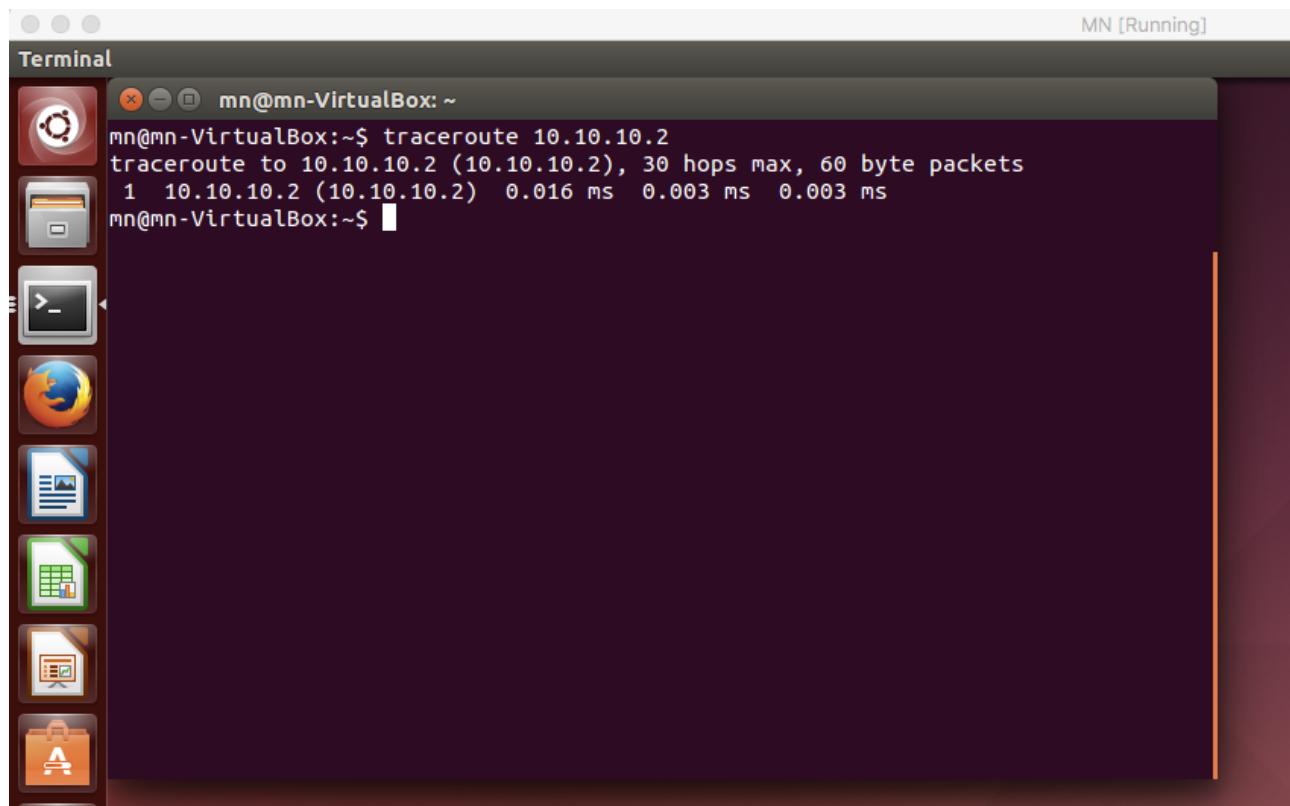


Figura 36: Traceroute para llegar a MN desde la red base **LAN1**.

Se puede ver en la última imagen como para llegar al host **MN** desde la red base **LAN1** solo es necesario su dirección ip, es decir, no hay que dar ningún salto entre redes. Ahora que hemos comprobado tanto los descubrimientos de los agentes **HA** y **FA** como que el host **MN** se encuentra en la red base **LAN1**, vamos a cambiar a dicho host a la red visitada **LAN3** para ver en un primer lugar el registro del **CA** en la red base y el **tunneling** creado entre los dos agentes para comunicar con el host **MN** desde otro host situado en la red intermedia **LAN2**.

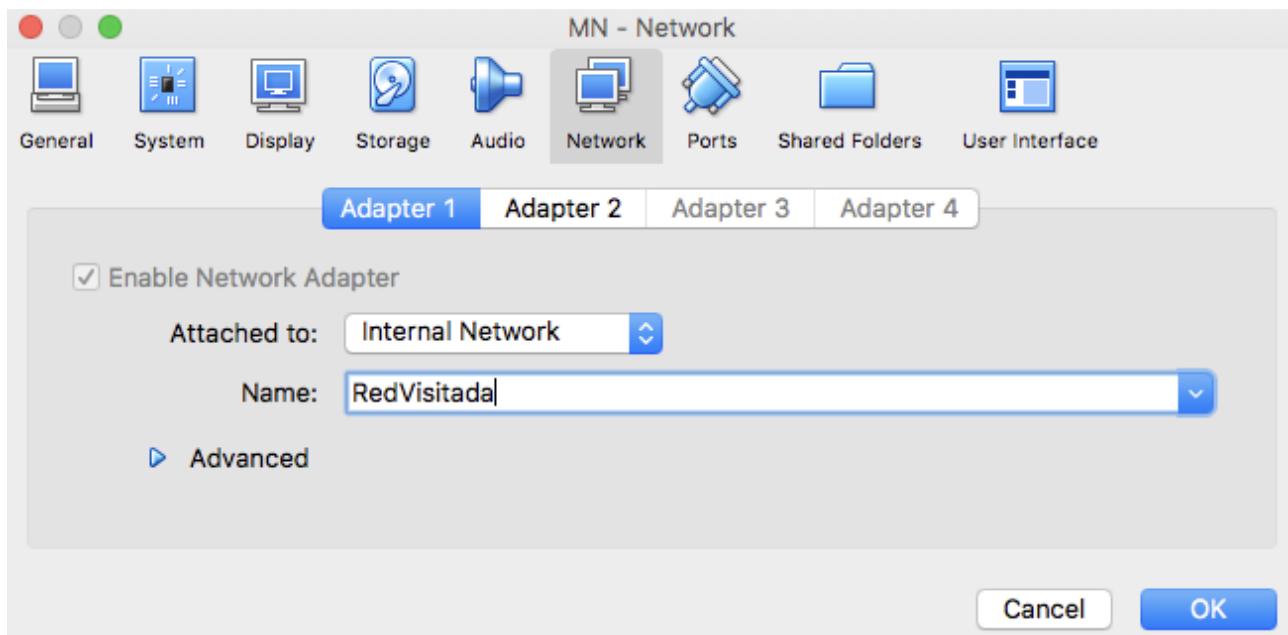


Figura 37: Cambiando al host MN a la red visitada **LAN3**.

Cuando se ha producido el cambio de red del host **MN** de la red base **LAN1** en la que se encontraba a la red visitada **LAN3**, podemos ver en las siguientes imágenes el procedimiento de registro de la dirección **CA** que se lleva a cabo.

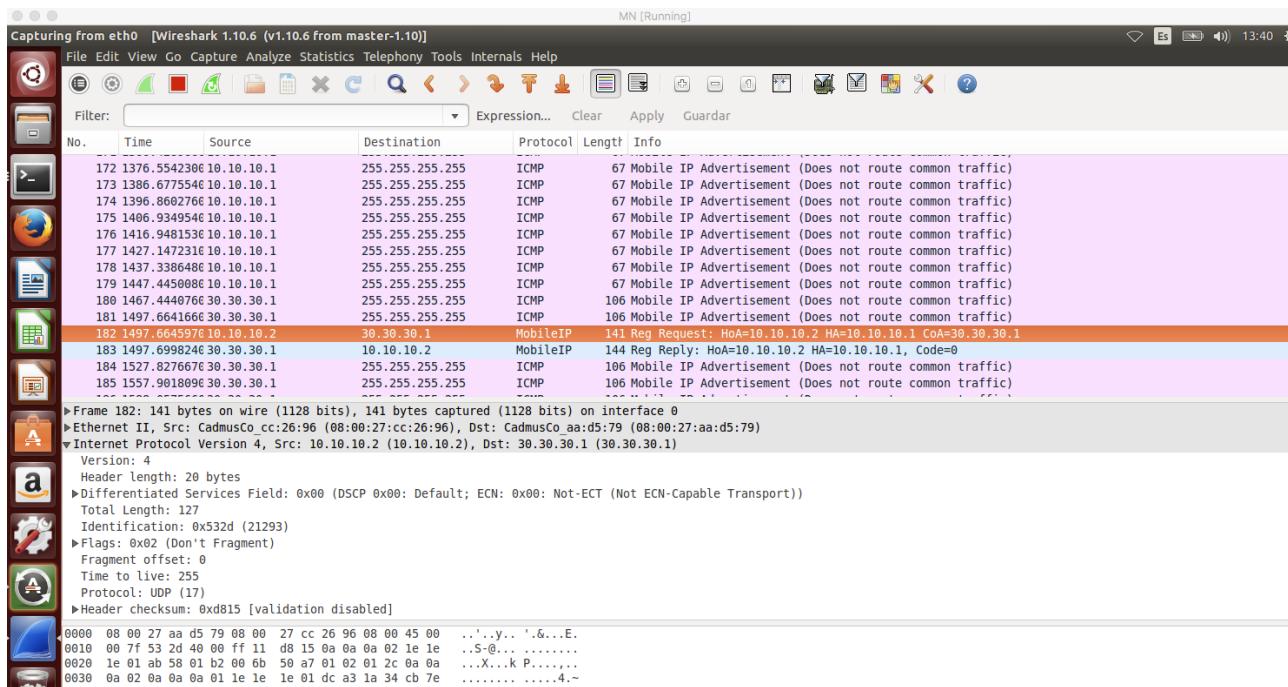


Figura 38: **MN** envía un mensaje de RRq al **FA**.

En esta primera imagen se ve como el host **MN** envía un mensaje de petición de registro (RRq) al host **FA**. En dicho mensaje le indica la dirección del agente **HA** al que el agente **FA** debe retransmitirle el mensaje una

vez que lo procese. El agente **HA** responderá con un mensaje RR_p al agente **FA** en el que concederá o denegará la petición, a su vez éste le retransmitirá la respuesta al host **MN** una vez que la haya procesado.

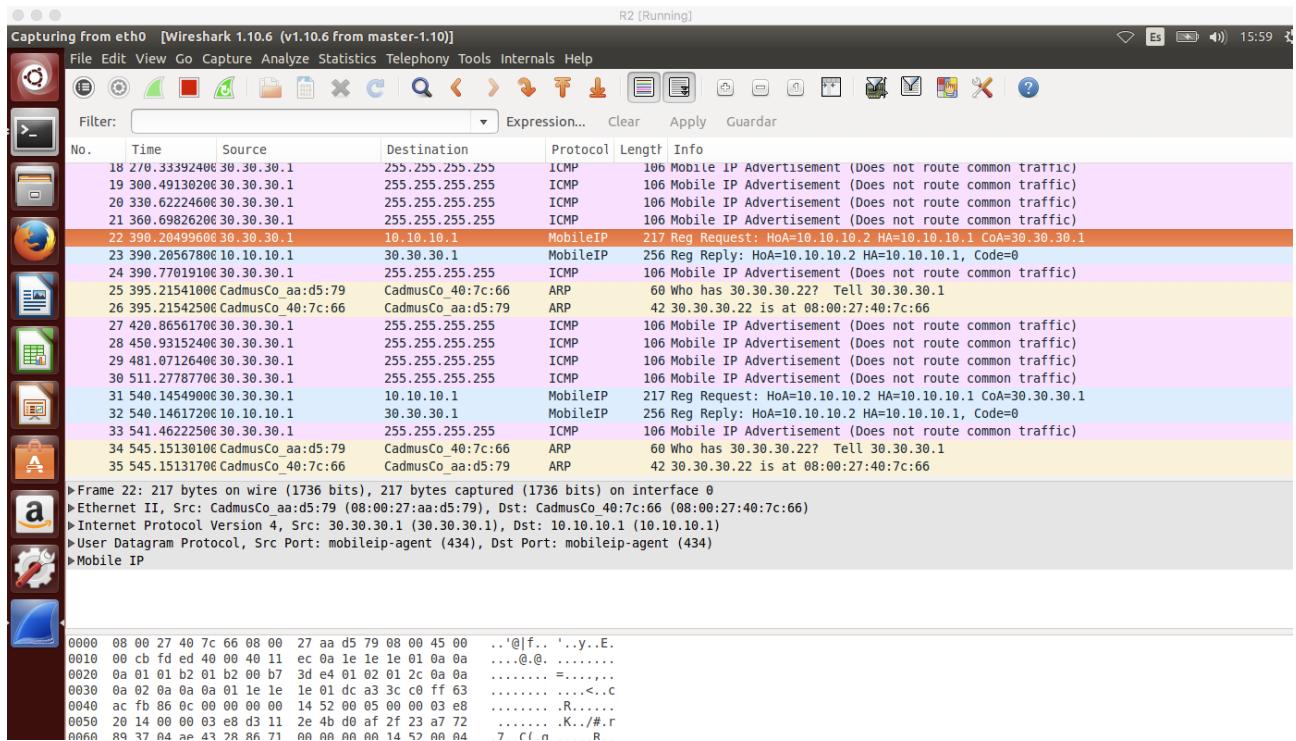
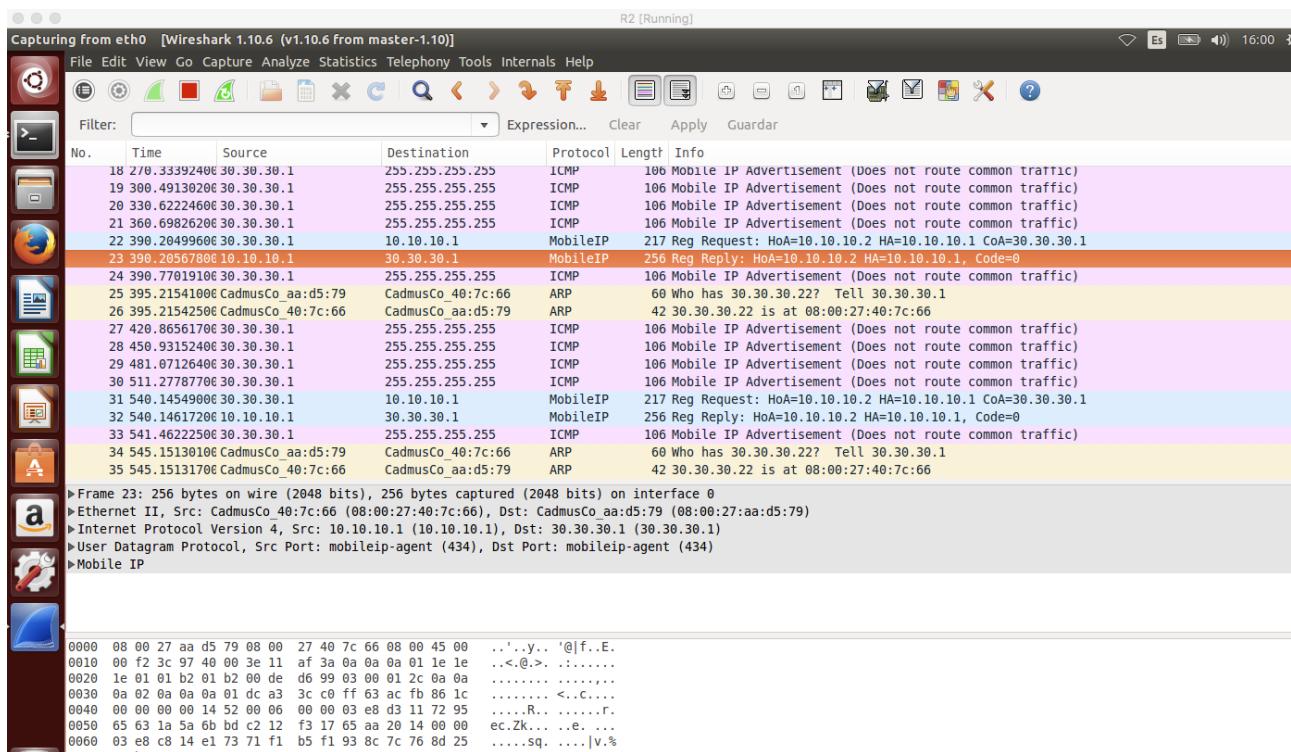
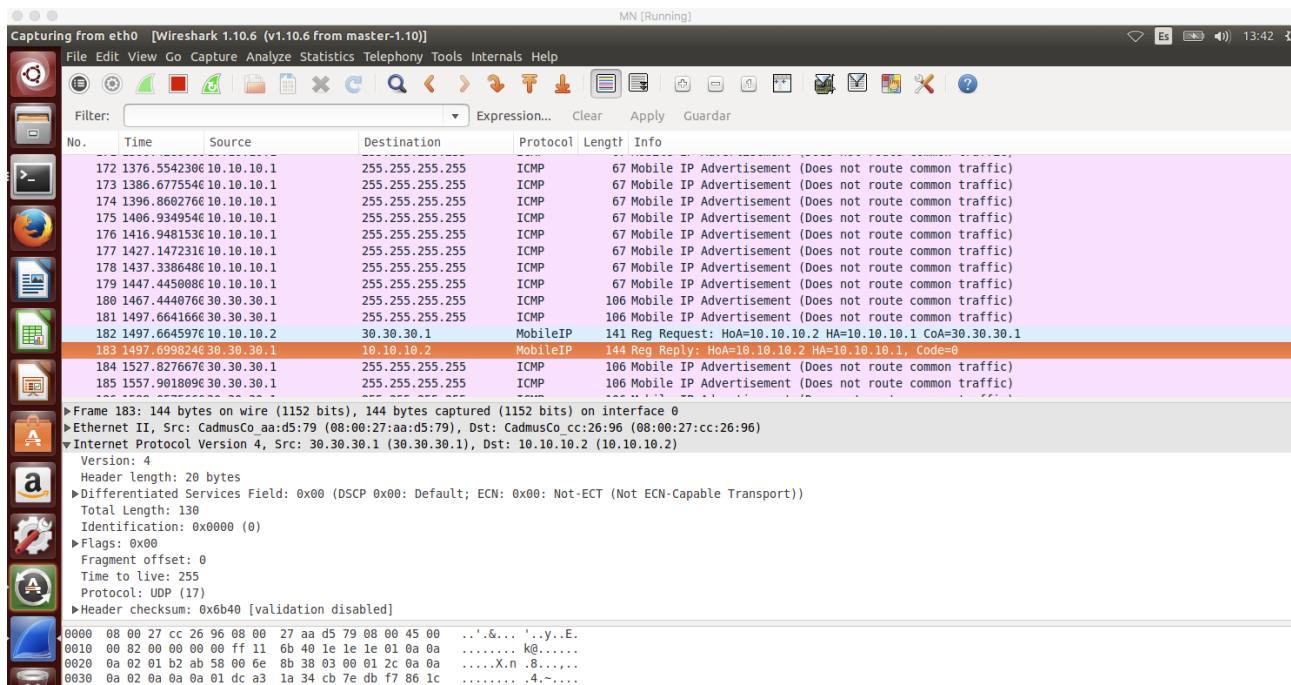
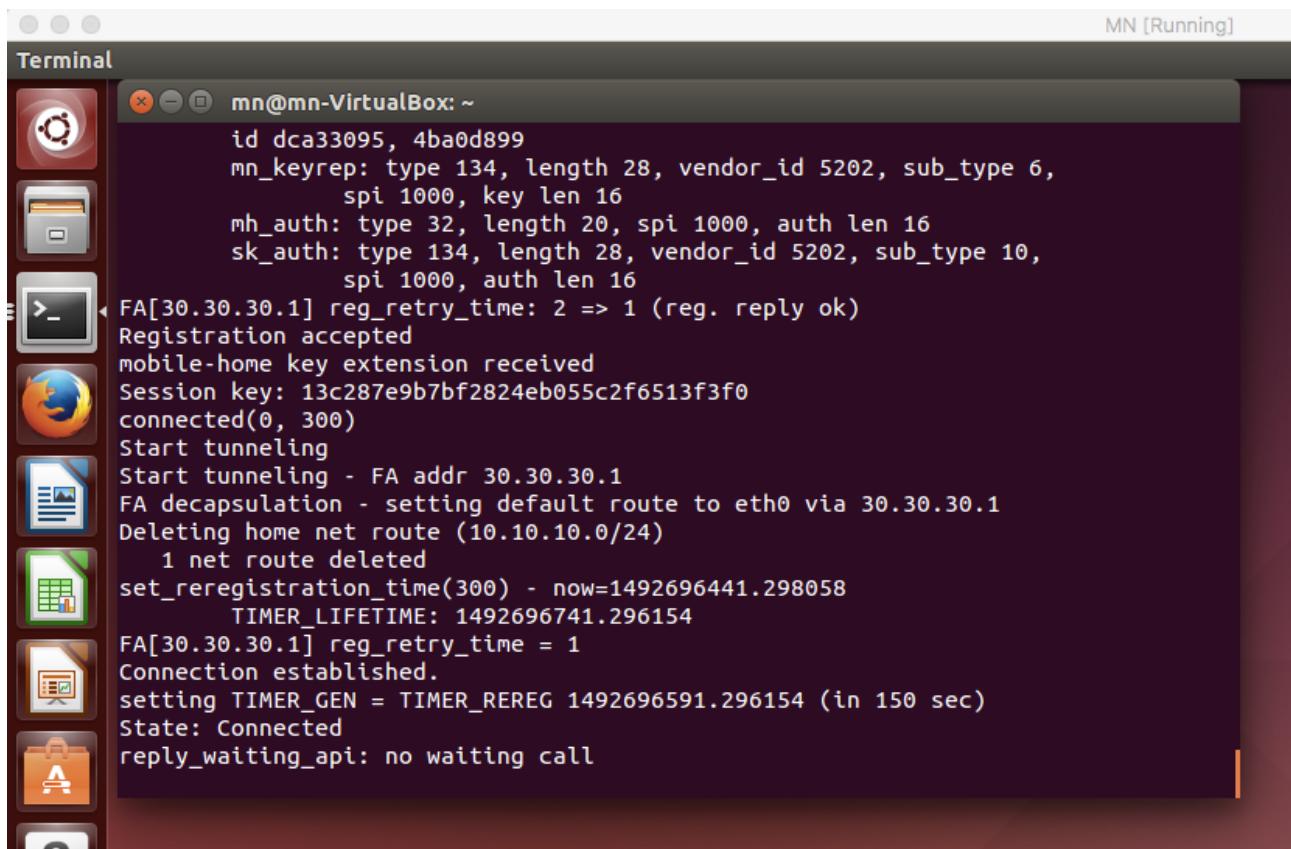


Figura 39: Envío del mensaje RR_q del agente **FA** al agente **HA**.

Figura 40: Envío del mensaje RRp del agente **HA** al agente **FA**.Figura 41: Envío del mensaje RRp del agente **FA** al host **MN**.



```
id dca33095, 4ba0d899
mn_keyrep: type 134, length 28, vendor_id 5202, sub_type 6,
           spi 1000, key len 16
mh_auth: type 32, length 20, spi 1000, auth len 16
sk_auth: type 134, length 28, vendor_id 5202, sub_type 10,
           spi 1000, auth len 16
FA[30.30.30.1] reg_retry_time: 2 => 1 (reg. reply ok)
Registration accepted
mobile-home key extension received
Session key: 13c287e9b7bf2824eb055c2f6513f3f0
connected(0, 300)
Start tunneling
Start tunneling - FA addr 30.30.30.1
FA decapsulation - setting default route to eth0 via 30.30.30.1
Deleting home net route (10.10.10.0/24)
  1 net route deleted
set_reregistration_time(300) - now=1492696441.298058
  TIMER_LIFETIME: 1492696741.296154
FA[30.30.30.1] reg_retry_time = 1
Connection established.
setting TIMER_GEN = TIMER_REREG 1492696591.296154 (in 150 sec)
State: Connected
reply_waiting_api: no waiting call
```

Figura 42: Registro aceptado y túnel iniciado.

Como se ha podido ver en las imágenes anteriores, la respuesta que el agente **HA** envía es positiva, por lo que el registro del **CA** se ha llevado con éxito y se ha iniciado el túnel. Estos mensajes se repiten periódicamente ya que el registro del **CA** tiene un tiempo de vida que al caducar hace que se vuelva a solicitar. Para una mayor comprobación, se puede ver como el host **MN** recibe paquetes de anuncio del agente externo **FA** como muestran las siguientes imágenes.

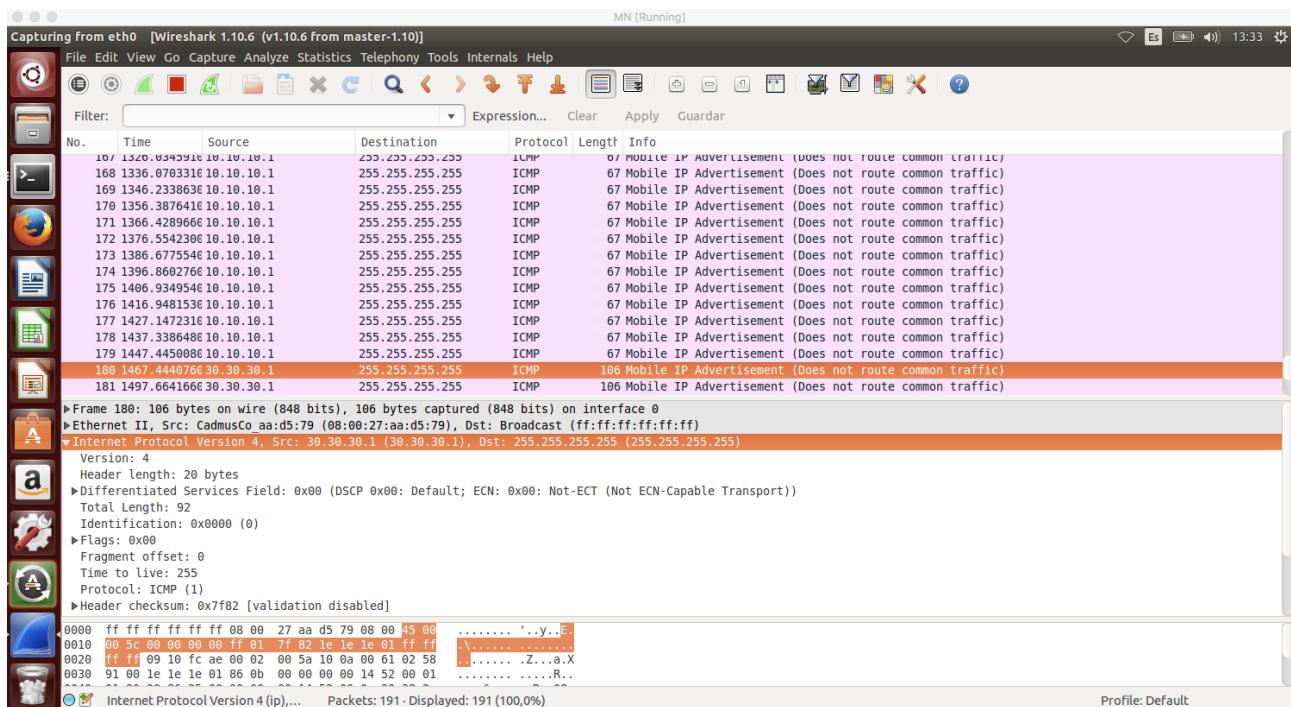


Figura 43: Descubrimiento del agente FA capturado por el host MN (I).

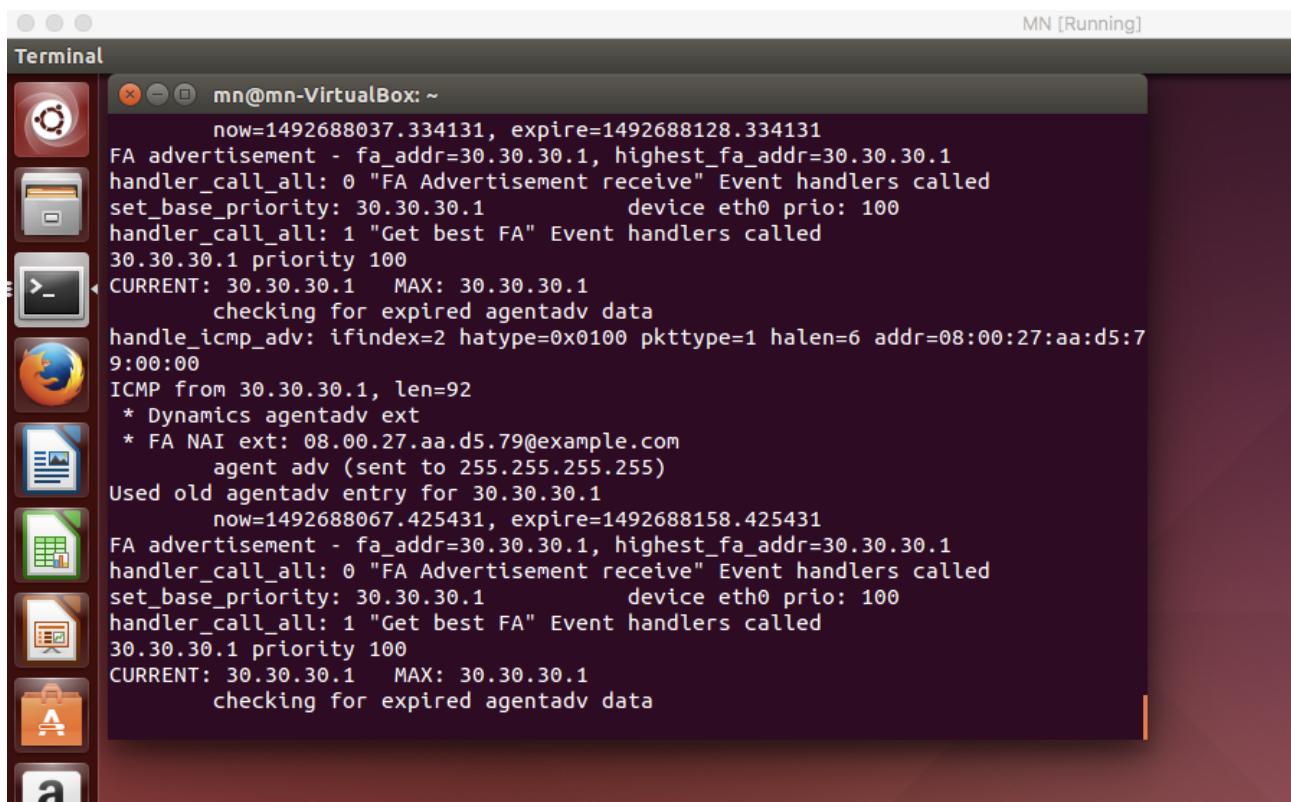


Figura 44: Descubrimiento del agente FA capturado por el host MN (II).

Siguiendo con la comprobación del protocolo **IP Móvil**, ahora nos queda por comprobar que se realiza el **tunneling** entre los agentes **HA** y **FA**. Para ello vamos a realizar un **ping** y un **traceroute** desde el host **HostIntermedio** al host **MN** para comprobar los saltos que son necesarios dar para comunicar con ese nodo. Cabe recordar que cualquier paquete que vaya destinado a **MN** será interceptado por **HA** y enviado por éste a **FA** sobre el túnel, sin embargo, los paquetes generados por **MN** no tienen por qué pasar por **HA**.

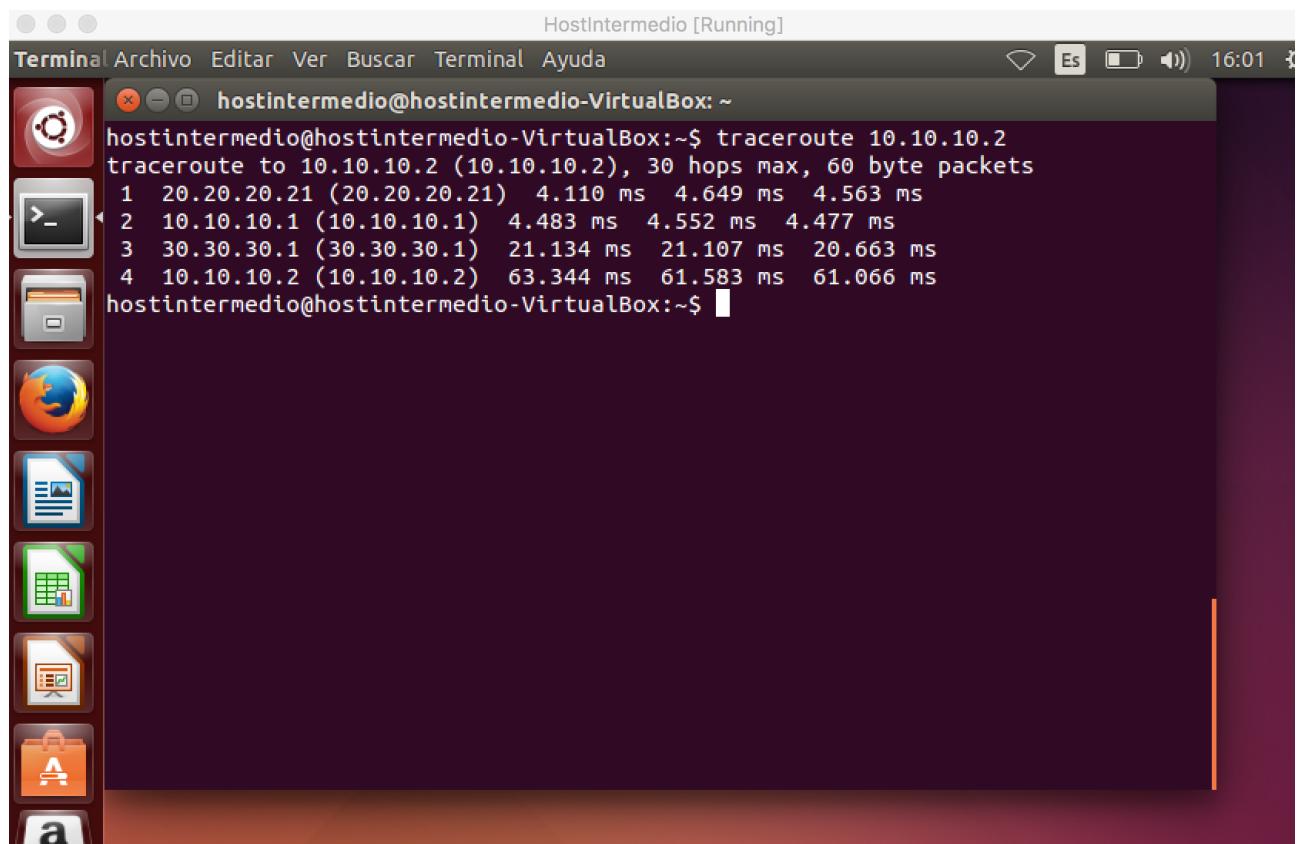


Figura 45: Traceroute desde **HostIntermedio** a **MN**.

Con los saltos que se debe dar para llegar desde **HostIntermedio** a **MN** mostrados mediante **traceroute**, vamos a ejecutar **ping** para ver que efectivamente el paquete a **MN** le llega de **FA** que a su vez le llega desde **HA** y no directamente desde **HostIntermedio** y viceversa para la respuesta de **MN** a **HostIntermedio**.

```

hostintermedio@hostintermedio-VirtualBox:~$ ping 10.10.10.2
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.
64 bytes from 10.10.10.2: icmp_seq=1 ttl=61 time=8.60 ms
64 bytes from 10.10.10.2: icmp_seq=2 ttl=61 time=11.3 ms
64 bytes from 10.10.10.2: icmp_seq=3 ttl=61 time=1.64 ms
64 bytes from 10.10.10.2: icmp_seq=4 ttl=61 time=31.6 ms
64 bytes from 10.10.10.2: icmp_seq=5 ttl=61 time=4.34 ms
64 bytes from 10.10.10.2: icmp_seq=6 ttl=61 time=2.51 ms
64 bytes from 10.10.10.2: icmp_seq=7 ttl=61 time=125 ms
64 bytes from 10.10.10.2: icmp_seq=8 ttl=61 time=133 ms
64 bytes from 10.10.10.2: icmp_seq=9 ttl=61 time=29.5 ms
64 bytes from 10.10.10.2: icmp_seq=10 ttl=61 time=16.8 ms
64 bytes from 10.10.10.2: icmp_seq=11 ttl=61 time=14.2 ms
64 bytes from 10.10.10.2: icmp_seq=12 ttl=61 time=9.32 ms
64 bytes from 10.10.10.2: icmp_seq=13 ttl=61 time=4.55 ms
64 bytes from 10.10.10.2: icmp_seq=14 ttl=61 time=16.2 ms
64 bytes from 10.10.10.2: icmp_seq=15 ttl=61 time=1.71 ms
64 bytes from 10.10.10.2: icmp_seq=16 ttl=61 time=2.63 ms
64 bytes from 10.10.10.2: icmp_seq=17 ttl=61 time=4.42 ms
64 bytes from 10.10.10.2: icmp_seq=18 ttl=61 time=9.98 ms
64 bytes from 10.10.10.2: icmp_seq=19 ttl=61 time=2.03 ms
64 bytes from 10.10.10.2: icmp_seq=20 ttl=61 time=50.2 ms
64 bytes from 10.10.10.2: icmp_seq=21 ttl=61 time=1.59 ms

```

Figura 46: Pin de HostIntermedio a MN (I).

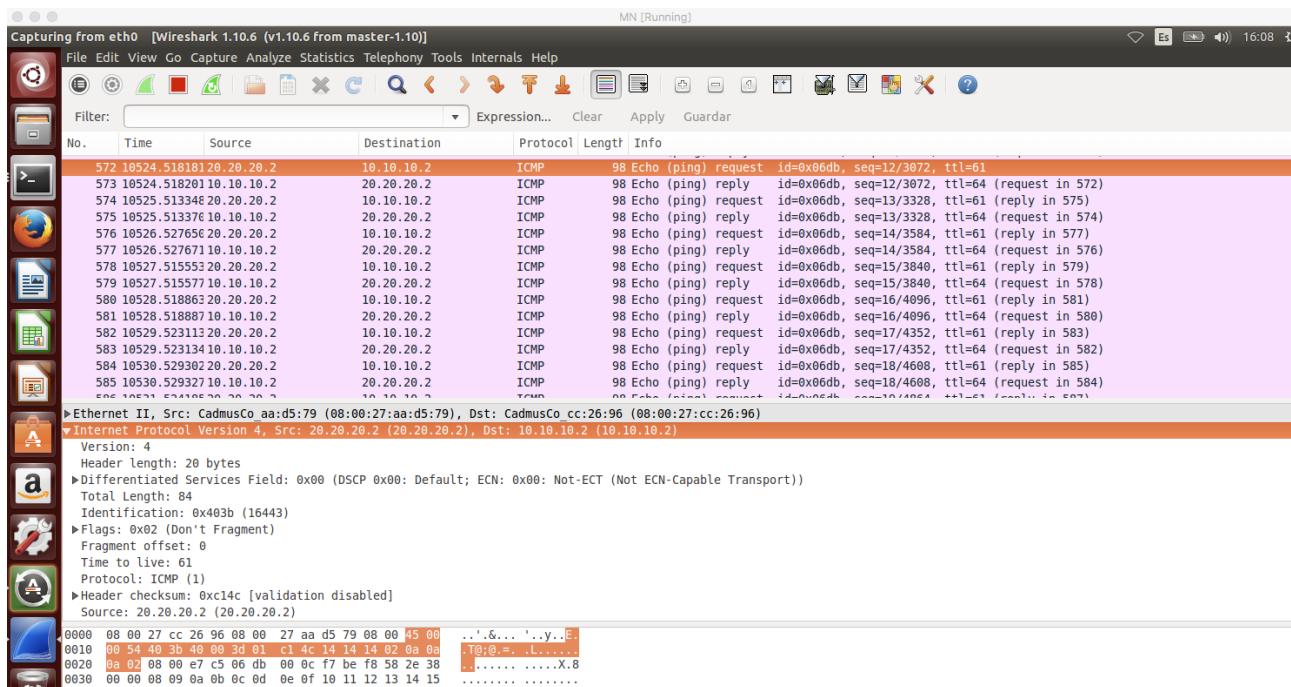


Figura 47: Pin de HostIntermedio a MN (II).

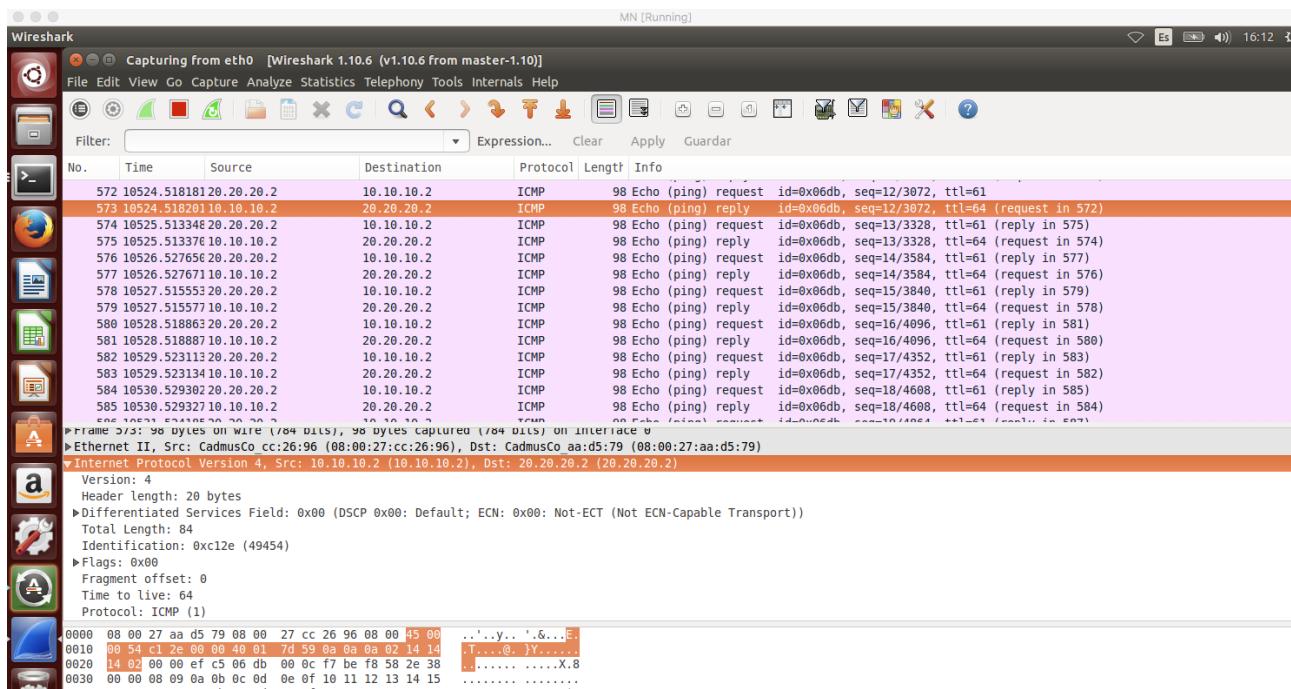


Figura 48: Respuesta del pin de MN a HostIntermedio.

Una vez que se ha comprobado el correcto funcionamiento del **tunneling**, se va a volver a conectar el host **MN** a la red base **LAN1**. Veremos entonces como el agente de dicho host cambiará al agente local **HA** y los paquetes de anuncio que verá serán de éste, y como ahora el traceroute para llegar desde **HostIntermedio** a **MN** es diferente ya que el **tunneling** ha desaparecido, por lo que no serán necesarios tantos saltos. Recordar también que al encontrarse **MN** en su red, no será necesario ningún tipo de registro de **CA**.

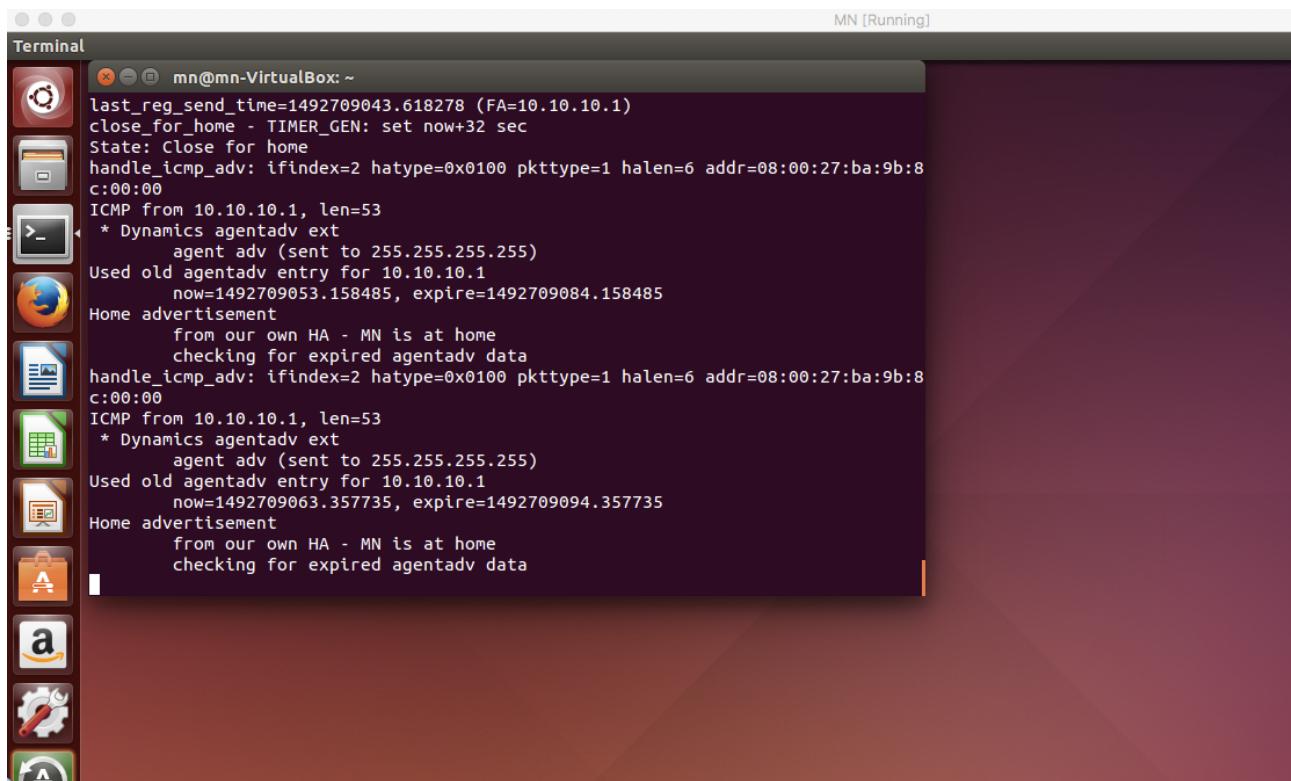


Figura 49: MN recibe paquetes de aviso de HA al cambiar a la red base LAN1 (I).

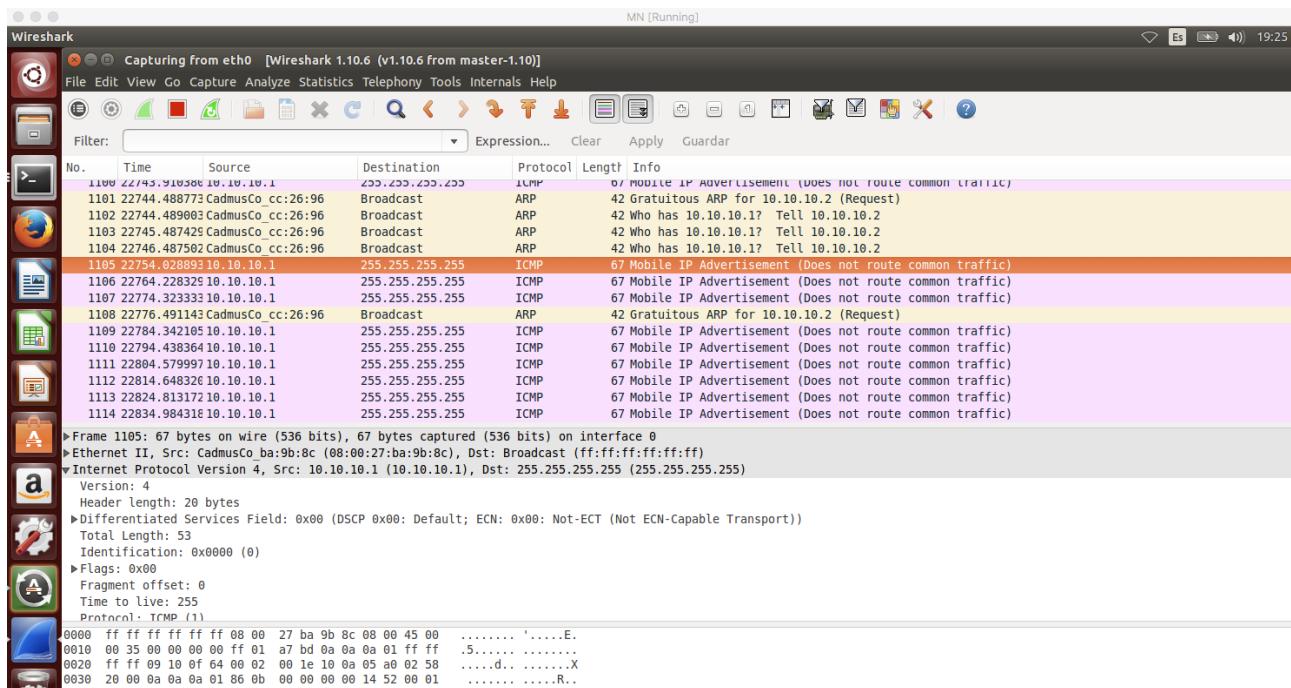


Figura 50: MN recibe paquetes de aviso de HA al cambiar a la red base LAN1 (II).

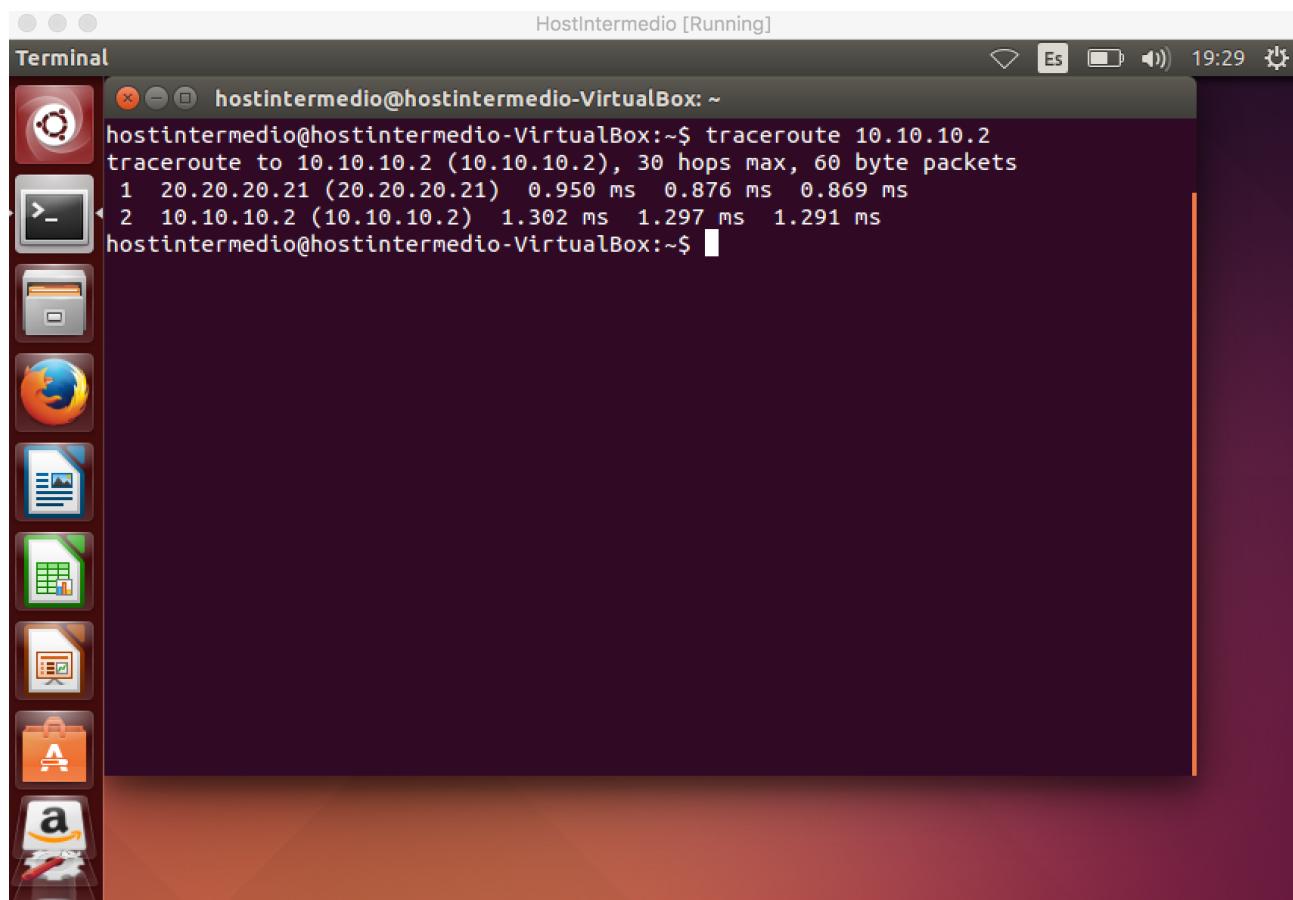


Figura 51: Traceroute de **HostIntermedio** a **MN**.

Como se ha podido ver en la anterior imagen, la ruta para ir de **HostIntermedio** a **MN** ha cambiado debido al cambio de red de este último, que como se ha mencionado antes no se necesita realizar **tunneling** al encontrarse en su red base **LAN1**.

Se ha probado correctamente por tanto el funcionamiento del protocolo de movilidad **IP Móvil (IPv4)**.