


Lab 3: Desarrollo de una aplicación SDN	 Universidad Carlos III de Madrid
Redes Software Última modificación: 13-04-2020	2019-20

Introducción y objetivos

1 Objetivo

En este laboratorio desarrollará de manera independiente y basándose en los conocimientos adquiridos en los laboratorios anteriores una aplicación Software Defined Networking (SDN) que implemente la funcionalidad de un encaminador IPv4 y que se ejecute en el controlador RYU. Para demostrar el funcionamiento se apoyará en un entorno de emulación de red sobre Mininet.

2 Normas básicas

Estudie este documento en profundidad antes de empezar el laboratorio. Asegúrese que lo entiende en su totalidad. El laboratorio está pensado para realizarse en grupos de dos estudiantes.

Aclare posibles dudas con los profesores en el laboratorio.

El proceso de desarrollo se ha dividido en cuatro etapas. Cree un subdirectorío **distinto** para el desarrollo de cada una de las etapas. A partir de la segunda etapa, copie los programas que haya desarrollado en la etapa anterior y modifíquelos para satisfacer los criterios de la etapa correspondiente.

El laboratorio se evaluará en un test escrito en el que se le harán preguntas sobre el desarrollo realizado en este laboratorio.

3 Entorno de trabajo

Utilice la máquina virtual instalada en los equipos del laboratorio. Esta máquina virtual está preparada para el hipervisor VirtualBox [1]. Internamente, la máquina virtual esta basada en XUbuntu versión 18.04 [2]. Utiliza el escritorio XFCE [3], que por su bajo consumo de recursos, es muy recomendable en entornos virtualizados. Además del software necesario para los laboratorios, la máquina incluye un número suficiente de programas auxiliares para ayudar en la ejecución de este y los demás laboratorios. **No instale más paquetes a no ser que se lo indiquen los profesores de laboratorio.**

La máquina virtual incluye:

1. El controlador `ryu` en su versión 4.34
2. El simulador de redes `mininet` en su versión 2.3.0d3
3. El editor de textos `atom`

Nota : Debido a una incompatibilidad surgida en versiones recientes de Virtualbox, puede que la tecla `AltGr` no se pase correctamente al entorno emulado en algunas combinaciones de sistemas operativos. Para prevenir este problema, esta tecla se mapea la tecla de menu del teclado con el fichero `/home/student/.Xmodmap` dentro de la máquina virtual.

En caso de no ser necesaria esta adaptación, se puede renombrar el fichero con el comando

```
mv ~/.Xmodmap ~/.dot.Xmodmap
```

para conservarlo para entornos en los que sea necesario.

3.1 Arranque de la Máquina Virtual y usuario por defecto

Creación de la imagen de máquina virtual en el laboratorio

La Máquina Virtual (VM) se genera desde un terminal con el comando:

```
/usr/dist/bin/rs20.sh start
```

Creacion de la máquina virtual en un PC local

Acceda al enlace de la imagen OVA publicado en Aula Global y descárguela. Una vez descargada, abra el fichero con el programa VirtualBox e importe la máquina virtual.

Arranque de la máquina virtual

Arrancar la interfaz gráfica del programa VirtualBox. Una vez abierta, aparece una máquina virtual llamada **rs-2020**, que hay que seleccionar y arrancar.

El usuario por defecto es **student** y la clave de acceso es **5tud3nt**. Esta clave se necesitará para poder ejecutar comandos elevando privilegios con el comando `sudo`.

4 Utilización de los ejemplos de código

Este manual incluye ejemplos de código que se deberían utilizar en las prácticas. Se pueden utilizar cortando y pegando el código con un visor PDF arrancado en la máquina virtual o ajustando el funcionamiento de dicha función entre la máquina virtual y el anfitrión (e.d. el PC de laboratorio). Es posible que tenga que eliminar algún carácter especial y reajustar la indentación.

Un ejemplo de código que habría que editar después de ser pegado en el editor de textos es:

```
def function(self,argumento):  
    print('Esto es un ejemplo de mensaje')  
    print('en Python')  
    print('con argumento formateado %s' % argumento)
```



Nota : Intente copiar y pegar este texto e identifique si hay que algún carácter especial y cómo hay que reindentar el código en el editor. La “chincheta” indica que el código está incluido en el fichero PDF. Un visor que soporte adjuntos (*attachments*) permitirá extraer el código a un fichero.

Nota : En esta práctica, se tiene que utilizar el modo *Python* para todo el código que se edite. **Se recomienda** partir de los ejemplos desarrollados en el laboratorio de *Introducción a Mininet y Ryu*.

1 Implementación básica de la conmutación de paquetes IPv4

El objetivo de esta práctica es reproducir los procesos de conmutación de paquetes de nivel 3 (IPv4) que tienen lugar en un encaminador. Para implementar el entorno completo, vamos a proceder de manera incremental:

- I. Implementación de la tabla de encaminamiento en el conmutador
- II. Implementación de las respuestas al protocolo ICMP en el conmutador
- III. Implementación de las respuestas al protocolo ARP en el conmutador

1.1 Introducción al entorno

En este laboratorio pretendemos emular el siguiente entorno:

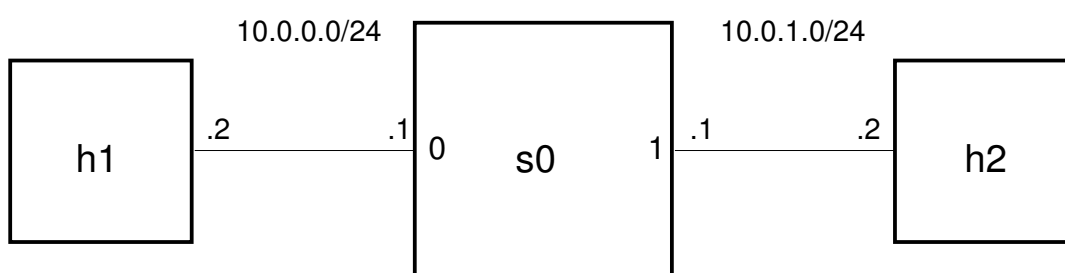


Figure 1.1: Escenario Mininet del desarrollo

Adicionalmente, se supone la siguiente correspondencia entre direcciones IPv4 y MACs:

Dirección IPv4	MAC	Dirección IPv4	MAC
10.0.0.1	77:88:99:00:00:01	10.0.1.1	77:88:99:00:00:02
10.0.0.2	00:00:00:00:00:01	10.0.1.2	00:00:00:00:00:02

Table 1.1: Correspondencia entre direcciones IPv4 y MACs

Recordemos que para que los equipos finales (*h1* y *h2*) envíen paquetes IPv4, deben de conocer la dirección MAC del destino si está en la misma subred o de la puerta por defecto (*default gateway*). Además, los equipos finales sólo recibirán paquetes cuya MAC destino sea la del equipo o la dirección de *broadcast* (FF:FF:FF:FF:FF:FF)¹

¹En funcionamiento normal. Se puede forzar la recepción de **todos** los paquetes, poniendo el equipo en modo

1.2 Gestión de MACs de los equipos finales

Como se ha expuesto en la introducción, los equipos finales requieren tener la información MAC del siguiente salto para poder enviar paquetes. Esta información la proporciona normalmente el protocolo de resolución de direcciones o Address Resolution Protocol (Protocolo de resolución de direcciones) (ARP). Para casos excepcionales, Linux permite definir la traducción de direcciones de modo estático.

Para el primer paso, la información ARP se programará de manera estática tanto en los equipos finales como en el encaminador. En los equipos finales, la puerta por defecto será la interface del encaminador a que está conectado. En un equipo Linux real, la configuración se realizaría con los comandos:

```
ip link set dev eth0 up
ip address add <dirección IPv4 de equipo>/24 dev eth0
ip route add default <dirección IPv4 del router>
arp -s <dirección IPv4 del router> <MAC del router>
```

Busque en la definición de la clase **Node** de Mininet cómo se tiene que implementar. Puede encontrar información adicional en [4] y en [5].

Es recomendable fijar el protocolo a OpenFlow13 en la definición del conmutador. Esto se puede realizar con el parámetro adicional `protocols='OpenFlow13'` a la hora de crearlo.

Recuerde que el entorno necesita que el controlador sea de tipo `RemoteController` para poder utilizar el controlador Ryu que tiene instalado en el entorno de desarrollo.

Proceso de conmutación en el conmutador

El proceso de conmutación en los conmutadores deberá emular el proceso que se realiza en un encaminador de nivel 3.

La selección de paquetes se realiza mediante protocolo y dirección destino. Para ello se tendrá que implementar un objeto `OFPMatch()` que detecte el código de tipo de paquete IPv4 en la cabecera MAC y filtre por prefijo IPv4 en la dirección de destino IPv4 para cada una de las rutas:

```
match = parser.OFPMatch(eth_type=ether_types.ETH_TYPE_IP,
                        ipv4_dst=('10.0.0.0', '255.255.255.0'))
```

promiscuo. Esto queda fuera del alcance de esta práctica.

Los paquetes se emiten una vez ajustados los campos de las cabeceras de la siguiente forma:

- i La dirección MAC origen es la de la interface del encaminador por la que salen
- ii La dirección MAC destino es la del equipo final al que van destinados
- iii El campo TTL de la cabecera IPv4 se decrementa en una unidad

```
actions = [  
    parser.OFPACTIONOutput(1),  
    parser.OFPACTIONDecNwTtl(),  
    parser.OFPACTIONSetField(eth_src='<MAC interface>'),  
    parser.OFPACTIONSetField(eth_dst='<MAC host>')  
]
```

Nota : Para mantener el código sea legible, es recomendable crear funciones que devuelvan las acciones. Partiendo del ejemplo anterior, se pueden crear la siguientes funciones para la conmutación de paquetes y el descarte de paquetes:

```
def forwardActions(self, parser, ofproto, port, src, dst):  
    return [  
        parser.OFPACTIONOutput(port),  
        parser.OFPACTIONDecNwTtl(),  
        parser.OFPACTIONSetField(eth_src=src),  
        parser.OFPACTIONSetField(eth_dst=dst),  
    ]  
def dropActions(self, parser, ofproto):  
    return [ ]
```

Y luego utilizarlas en el código. Observe que se pasan siempre los parámetros `parser` y `ofproto` aunque no se utilicen. De esta manera, el código resulta más homogéneo y, por tanto, legible.

La **tabla de conmutación** que se debería implementar es la siguiente:

prioridad	filtro	acción
10000	Paquete LLDP	Descartar
10000	Paquete IPv6	Descartar
1000	prefijo IP == 10.0.0.0/24	Conmutar a la 1
1000	prefijo IP == 10.0.1.0/24	Conmutar a la 2
0	REST	Send to controller

Table 1.2: Tabla de conmutación inicial

Las primeras entradas (las de mayor prioridad) harán que se descarten los paquetes que quedan fuera del objetivo de este laboratorio de manera automática.

Hito 1 : Ejecute el entorno y demuestre a los profesores que se puede realizar ping entre los dos equipos. Capture el tráfico en el extremo de *h1* (con Wireshark) y demuestre que el campo TTL se decrementa en el conmutador. Para ello haga ping de *h1* a *h2* y compare el campo TTL de las preguntas y respuestas ICMP.

Nota : En el código del conmutador se recomienda proceder de la siguiente manera:

Desarrollo de una aplicación SDN

1. Partir de cualquiera de los ejemplos de la sesión de SDN o del ejemplo del conmutador simple para OpenFlow1.3 (\$HOME/Install/ryu/ryu/app/simple_switch13.py)
2. Mantener el envío al controlador como último recurso en la tabla de conmutación
3. Eliminar el código de respuesta a paquetes recibidos del conmutador salvo la parte que imprime la información básica del paquete recibido (p.ej interface, Ethertype y MACs)

2 Gestión del protocolo ICMP

En este paso introducimos la funcionalidad de envío de paquetes desde el controlador. Para ello, haremos que los equipos finales puedan comprobar el estado de las puertas del conmutador enviando mensajes ICMP a la dirección IP correspondiente. Para ello, el controlador deberá enviar paquetes de respuesta ICMP a los paquetes de tipo `ICMP_REQUEST` que reciba.

Nota : Se recomienda partir del código que se desarrolló en el capítulo anterior. Duplíquelo en un directorio nuevo¹. Idealmente, el contenido del directorio de desarrollo de la VM se debería parecer a :

```
$HOME/Devel
├── chap1
│   ├── simple_router.py
│   └── scenario.py
└── chap2
    ├── simple_router.py
    └── scenario.py
```

Para implementar este paso tendrá que modificar el código del controlador (fichero `simple_router.py`) exclusivamente.

2.1 Función de envío de paquetes de respuesta ICMP

Parta de la captura de tráfico que realizó en el paso anterior y observe cómo se derivan los campos del paquete `ICMP_REPLY` a partir del paquete `ICMP_REQUEST`. Utilizando la librería de generación de paquetes de Ryu, tendrá que implementar una función que reciba un paquete `ICMP_REQUEST` y el puerto por el que se recibió y envíe la respuesta `ICMP_REPLY` correspondiente por el mismo puerto. En la figura 2.1 se ve como hacer los intercambios de *algunos* de los campos para generar el paquete de respuesta ICMP.

¹Desde `$HOME/Devel` puede ejecutar el comando `cp -rv chap1 chap2`.

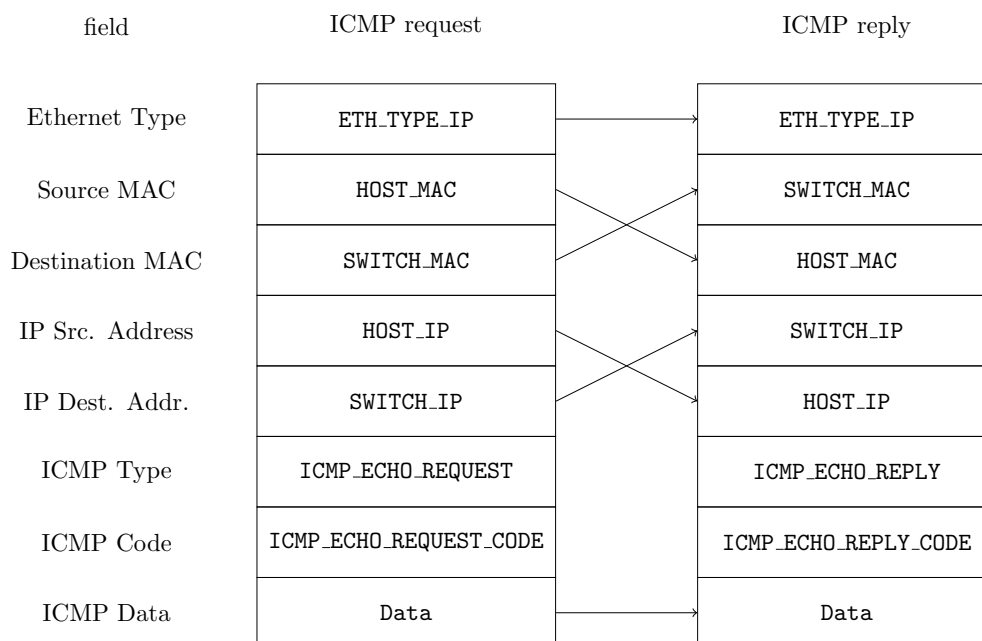


Figure 2.1: Construcción del paquete de respuesta ICMP

Para implementar esta función, utilice la documentación de la librería de paquetes de Ryu, que se describe en la documentación de Ryu [6] y en el libro de Ryu [7].

2.2 Integración en el código del controlador

Una vez implementada esta función, incluya el filtrado necesario en la tabla de conmutación del controlador, de manera que los paquetes dirigidos a las direcciones asignadas a las puertas del conmutador sean enviados al controlador. Piense la prioridad que deben de tener estas entradas para que este tráfico en concreto sea atendido por el controlador.

Finalmente, modifique la función de procesamiento de eventos para que los paquetes ICMP que lleguen al controlador sean tratados con la función de generación de respuestas **exclusivamente**.

Hito 2 : Demuestre mediante el comando ping que todos los equipos finales pueden detectar ambos puertos del 'encaminador' mediante el protocolo ICMP.

3 Integración del protocolo ARP

En este paso eliminaremos la programación estática del protocolo ARP en los equipos finales y haremos que el controlador gestione la respuesta a este protocolo. Para ello, añadiremos código para detectar y decodificar paquetes ARP y una función de respuesta a paquetes ARP_REQUEST en la gestión de los paquetes enviados por el conmutador al controlador.

Nota : Se recomienda partir del código que se desarrolló en el capítulo anterior. Duplíquelo en un directorio nuevo. Idealmente, el contenido del directorio de desarrollo de la VM se debería parecer a :

```
$HOME/Devel
├── chap1
│   ├── simple_router.py
│   └── scenario.py
├── chap2
│   ├── simple_router.py
│   └── scenario.py
└── chap3
    ├── simple_router.py
    └── scenario.py
```

Para implementar este paso tendrá que modificar tanto el escenario (`scenario.py`) como el código del controlador (`simple_router.py`).

3.1 Eliminar la configuración estática del protocolo ARP

En el código del escenario, elimine *exclusivamente* el código relacionado con la programación estática del escenario.

Ejecute el escenario con el código del controlador utilizado en el capítulo anterior **sin modificar**. Compruebe que se ha dejado de poder hacer ping desde cualquier equipo final a cualquier dirección IP remota (*e.d.* las direcciones de los puertos del encaminador y la del otro equipo final).

3.2 Decodificación y respuesta al protocolo ARP en el controlador

Para poder trabajar con el protocolo ARP, tendrá que empezar por importar el módulo `arp` de la librería `ryu.lib.packet`. Una posibilidad de mantener el código legible es incluir todos los módulos de protocolos en una única línea en el bloque de importación de módulos:

```
from ryu.lib.packet import ethernet, ipv4, icmp, arp
```

Esto permitirá intentar leer la información del protocolo ARP más adelante y reaccionar adecuadamente si está presente:

```
arp_info = pkt.get_protocol(arp.arp)
if arp_info is not None:
```

y mandar el paquete a una función que lo trate adecuadamente.

Respuesta a paquetes ARP

El controlador deberá responder a paquetes de tipo ARP_REQUEST. La siguiente figura muestra cómo se debe transformar el paquete ARP_REQUEST en un paquete ARP_REPLY:

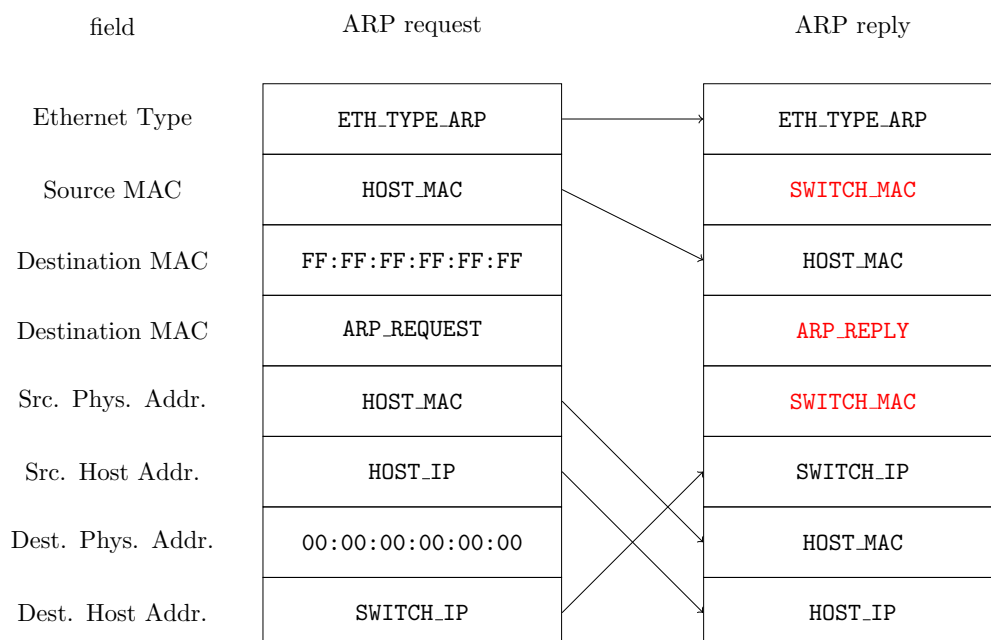


Figure 3.1: Como crear la respuesta ARP a partir de la petición

Al igual que para la implementación de la respuesta del protocolo ICMP, utilice la documentación de la librería de paquetes de Ryu, que se describe en la documentación de Ryu [6] y en el libro de Ryu [7], para implementar el código que genere y envíe el paquete de respuesta ARP.

Acrónimos

ARP	Address Resolution Protocol (Protocolo de resolución de direcciones)
ICMP	Internet Control Message Protocol (Protocol de mensajes de control de Internet)
MAC	Medium Access Control (Control de acceso al medio)
OF	OpenFlow
SDN	Software Defined Networking
VM	Máquina Virtual

Referencias

- [1] *VirtualBox*. URL: <https://www.virtualbox.org>.
- [2] *xubuntu*. URL: <https://xubuntu.org>.
- [3] *XFCE*. URL: <https://xfce.org>.
- [4] *Host configuration methods*. URL: <https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#config>.
- [5] *Introduction to Mininet: additional examples*. URL: <https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#examples>.
- [6] *Ryu documentation*. URL: <https://ryu.readthedocs.io/en>.
- [7] *Ryu SDN Framework*. URL: <https://osrg.github.io/ryu-book/en/html/index.html#>.