Guía de Laboratorio 1

Introducción a SDN con Mininet y POX

Asignatura: Redes Programables / SDN & NFV

Duración: 90–120 minutos

Nivel: Principiante

Objetivo general

Comprender la separación entre el **plano de control** y el **plano de datos** en una red SDN, utilizando **Mininet** y el **controlador POX** para observar cómo se instalan reglas de flujo y cómo se gestionan los paquetes de forma centralizada.

Objetivos específicos

1. Construir una topología simple en Mininet y verificar conectividad entre hosts.
2. Ejecutar POX con el módulo de aprendizaje de capa 2 (l2_learning).
3. Inspeccionar reglas de flujo instaladas en Open vSwitch (OVS).
4. Comparar el comportamiento de un hub (forwarding.hub) frente a un switch con aprendizaje (l2_learning).
5. Aplicar una política simple de bloqueo de tráfico entre dos hosts.

Materiales y requisitos previos

- Ubuntu 20.04 o superior (nativo o en máquina virtual).
- Mininet instalado: `sudo apt update && sudo apt install -y mininet`.
- POX disponible (incluido con Mininet en muchas distribuciones) o clonado desde GitHub: `git clone https://github.com/noxrepo/pox.git`.
- Wireshark (opcional) para observar tráfico entre switch y controlador.

Marco teórico breve

Componente	Descripción
Plano de datos	Conjunto de switches (Open vSwitch) que reenvían paquetes según reglas de flujo.
Plano de control	Controlador central (POX) que recibe eventos (Packet-In), decide rutas y programa los
Plano de aplicación	Lógica de alto nivel: políticas, seguridad, balanceo, QoS, etc., que se ejecutan sobre e

1) Crear una topología simple en Mininet

En una terminal:

```
sudo mn --topo single,3 --controller=remote --switch ovsk
```

Esto crea un switch (s1) y tres hosts (h1, h2, h3). Mantén esta consola abierta; aparecerá el prompt *mininet*>.

2) Lanzar el controlador POX (capa 2 con aprendizaje)

En una segunda terminal, dentro de la carpeta *pox/*:

```
./pox.py forwarding.l2_learning log.level --DEBUG
```

El módulo **l2_learning** aprende direcciones MAC y programa reglas de reenvío en el switch.

3) Probar conectividad y observar el aprendizaje

En la consola de Mininet:

```
mininet> pingall
```

La primera ejecución generará *Packet-In* al controlador; la segunda será más rápida porque ya existen reglas instaladas.

Para inspeccionar las reglas en el switch:

```
mininet> dpctl dump-flows
```

4) Comparar hub vs. switch con aprendizaje

Detén POX (Ctrl+C) y vuelve a iniciarlo como hub:

```
./pox.py forwarding.hub log.level --DEBUG
```

Repite `pingall` y compara. El hub reenvía por todos los puertos (flooding); el switch con aprendizaje reenvía por el puerto específico.

5) Política simple: bloquear tráfico entre h1 y h3

Crea un módulo básico de POX (por ejemplo, *blocking.py*) que instale reglas para evitar tráfico entre MAC/IP de h1 y h3. Fragmento ilustrativo:

```
from pox.core import core
import pox.openflow.libopenflow_01 as of

log = core.getLogger()

H1_MAC = "00:00:00:00:00:01"
H3_MAC = "00:00:00:00:00:03"

def _handle_ConnectionUp (event):
    log.info("Switch %s conectado", event.dpid)

    # Regla 1: bloquear h1 -> h3
    fm1 = of.ofp_flow_mod()
    fm1.match.dl_src = H1_MAC
    fm1.match.dl_dst = H3_MAC
    # Sin acciones => drop
    event.connection.send(fm1)

    # Regla 2: bloquear h3 -> h1
    fm2 = of.ofp_flow_mod()
    fm2.match.dl_src = H3_MAC
    fm2.match.dl_dst = H1_MAC
```

```
    event.connection.send(fm2)

def launch():
    core.openflow.addListenerByName("ConnectionUp", _handle_ConnectionUp)
    log.info("Módulo blocking.py cargado")
```

Ejecuta POX con el módulo:

```
./pox.py blocking
```

Prueba conectividad selectiva con `h1 ping -c 2 h3`.

6) Extensión opcional: topología por script y captura de tráfico

Topología Python para Mininet (archivo *topo_simple.py*):

```
from mininet.topo import Topo

class MyTopo( Topo ):
    def build(self):
        s1 = self.addSwitch('s1')
        h1 = self.addHost('h1')
        h2 = self.addHost('h2')
        h3 = self.addHost('h3')
        self.addLink(h1, s1)
        self.addLink(h2, s1)
        self.addLink(h3, s1)

topos = { 'mytopo': MyTopo }
```

Ejecuta:

```
sudo mn --custom topo_simple.py --topo=mytopo --controller=remote
```

Con Wireshark, observa tramas *OFPT_PACKET_IN* entre el switch y el controlador.

Rúbrica de evaluación sugerida

Criterio	Descripción
Conceptual (30%)	Explica planos SDN y el rol de POX vs. OVS.
Práctica (50%)	Construye la topología, ejecuta POX, inspecciona flows y demuestra el bloqueo.
Reflexión (20%)	Compara hub vs. switch con aprendizaje; discute ventajas de SDN.

Preguntas de reflexión

- ¿Qué ocurre en el controlador cuando llega un paquete sin coincidencia en las tablas del switch?
- ¿Cómo evidencia el experimento la separación entre plano de control y plano de datos?
- ¿Qué implicaciones tiene el uso de un hub para la seguridad y el rendimiento?
- ¿Cómo extenderías esta práctica para priorizar tráfico (QoS) o recolectar métricas (telemetría)?

Notas y buenas prácticas

• Usa instantáneas (snapshots) de la VM antes del laboratorio. • Si el controlador no conecta, revisa el puerto por defecto (6633/6653) o usa `--controller=remote,ip=127.0.0.1,port=6633`. • Para limpiar estado de OVS y Mininet: `sudo mn -c`.