Informe N°2:

“Lista de control de acceso (ACL)”

Grupo: Controladores SDN

Controlador usado: Ryu

INTEGRANTES:

Aldana Lacapmesure

Jose Luis Cahuana Lázaro

Matias D’amico

Access Control Lists (ACL)

Una lista de control de acceso permite controlar el flujo de tráfico en equipos como pueden ser enrutadores y conmutadores. El principal objetivo es filtrar tráfico, permitiendo o prohibiendo el tráfico de red de acuerdo a una condición dada. Además, puede servir para dar algunos usos adicionales como, por ejemplo, distinguir un tráfico más “importante” sobre otro.

Maqueta de Pruebas:

La prueba que realizaremos será sobre la topología definida en anteriores informes. 5 switch conectados entre sí, cada uno con un host. La topología sería la siguiente:

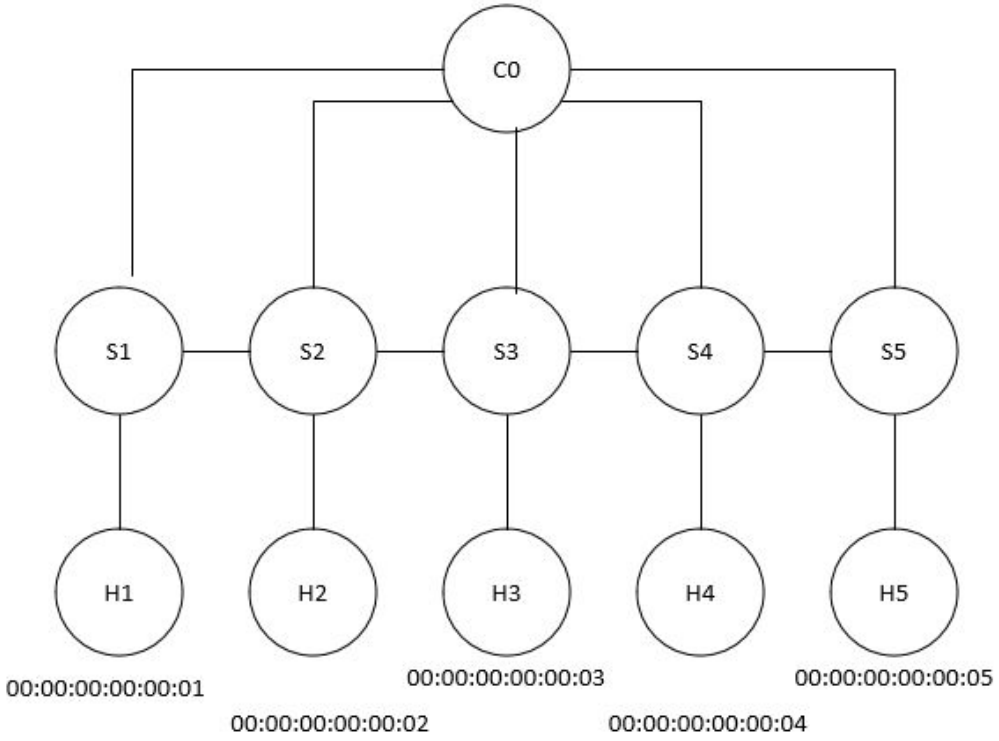
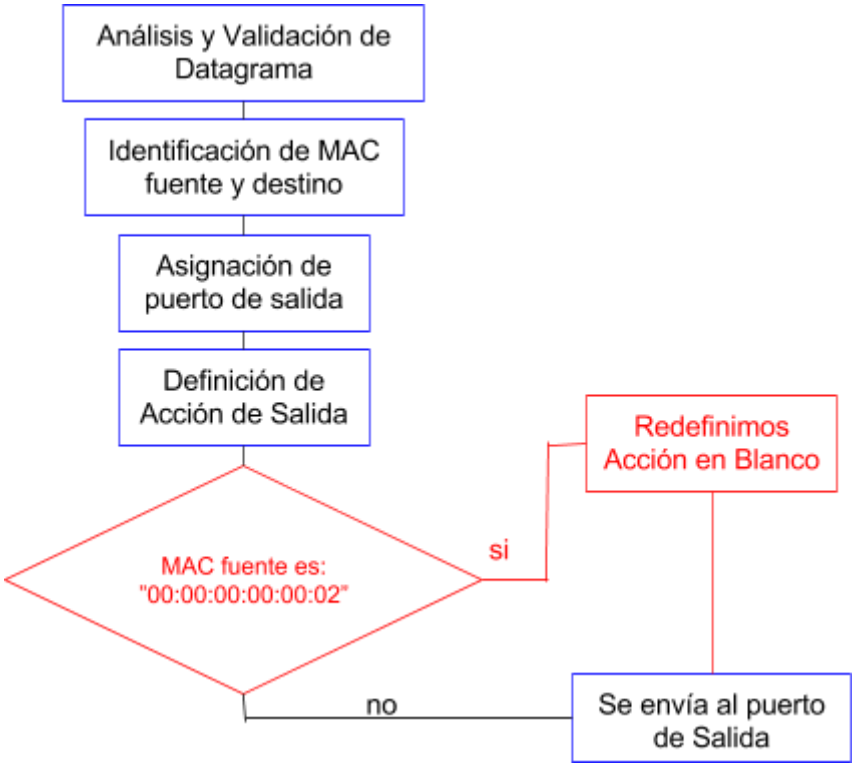


Diagrama de Flujo de ACL:



Código Propuesto:

Lo que buscamos hacer en esta topología es filtrar mediante dirección MAC el segundo host, ósea, la dirección 00:00:00:00:00:02.

Para lograr el filtrado utilizamos como controlador a Ryu modificando el código de **L2Switch**:

```
from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER, CONFIG_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.ofproto import ofproto_v1_3
from ryu.lib.packet import packet
from ryu.lib.packet import ethernet

class SimpleSwitch13v2(app_manager.RyuApp):
	OFP_VERSIONS= [ofproto_v1_3.OFP_VERSION]

	def __init__(self, *args, **kwargs):
		super(SimpleSwitch13v2,self).__init__(*args,**kwargs)

		self.mac_to_port = {}
		print (self.mac_to_port)
	@set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
	def switch_features_handler(self,ev):
		datapath = ev.msg.datapath
		ofproto = datapath.ofproto
		parser = datapath.ofproto_parser
```

```
        match = parser.OFPMatch()
        actions = [parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,
        ofproto.OFPCML_NO_BUFFER)]

        self.add_flow(datapath,0,match,actions)

def add_flow(self,datapath,priority,match,actions):
        ofproto=datapath.ofproto
        parser = datapath.ofproto_parser

        inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
        actions)]

        mod = parser.OFPFlowMod(datapath=datapath,priority=priority,

        match=match,instructions=inst)

        datapath.send_msg(mod)

@set_ev_cls(ofp_event.EventOFPPacketIn,MAIN_DISPATCHER)
def _packet_in_handler(self,ev):
        msg= ev.msg
        datapath = msg.datapath
        ofproto= datapath.ofproto
        parser = datapath.ofproto_parser

        dpid= datapath.id
        self.mac_to_port.setdefault(dpid, {})

        pkt = packet.Packet(msg.data)
        eth_pkt = pkt.get_protocol(ethernet.ethernet)

        dst =eth_pkt.dst
        src = eth_pkt.src

        in_port = msg.match["in_port"]

                self.mac_to_port[dpid][src]= in_port

        if dst in self.mac_to_port[dpid]:
                out_port = self.mac_to_port[dpid][dst]
        else:
                out_port = ofproto.OFPP_FLOOD

        actions = [parser.OFPActionOutput(out_port)]


        if src=="00:00:00:00:00:02":
                actions = []


        if out_port!=ofproto.OFPP_FLOOD:
                match = parser.OFPMatch(in_port=in_port,eth_dst=dst,
                                eth_src=src)

        self.add_flow(datapath,1,match,actions)

        out = parser.OFPPacketOut(datapath=datapath,

        buffer_id=ofproto.OFP_NO_BUFFER,
        in_port=in_port,
        actions=actions,data=msg.data)

        datapath.send_msg(out)
```

Para modificar la dirección MAC a filtrar habría que modificar el condicional “if” resaltado con turqueza con la dirección deseada a filtrar.

Resultados:

Como resultado al querer filtrar la dirección del segundo host en mininet probamos mediante el comando pingall y vemos como el host número 2 que tiene la dirección MAC 00:00:00:00:00:02 no puede ni enviar ni recibir paquetes.

```
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1 s2 s3 s4 s5
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (s2, s1) (s3, s2) (s4, s3) (s5, s4)

*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 5 switches
s1 s2 s3 s4 s5 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3 h4 h5
h2 -> X X X X
h3 -> h1 X h4 h5
h4 -> h1 X h3 h5
h5 -> h1 X h3 h4
*** Results: 40% dropped (12/20 received)
mininet>
```

A futuro:

En un futuro se podría modificar la manera de ingresar la dirección MAC a filtrar para evitar tener que entrar a editar el codigo.