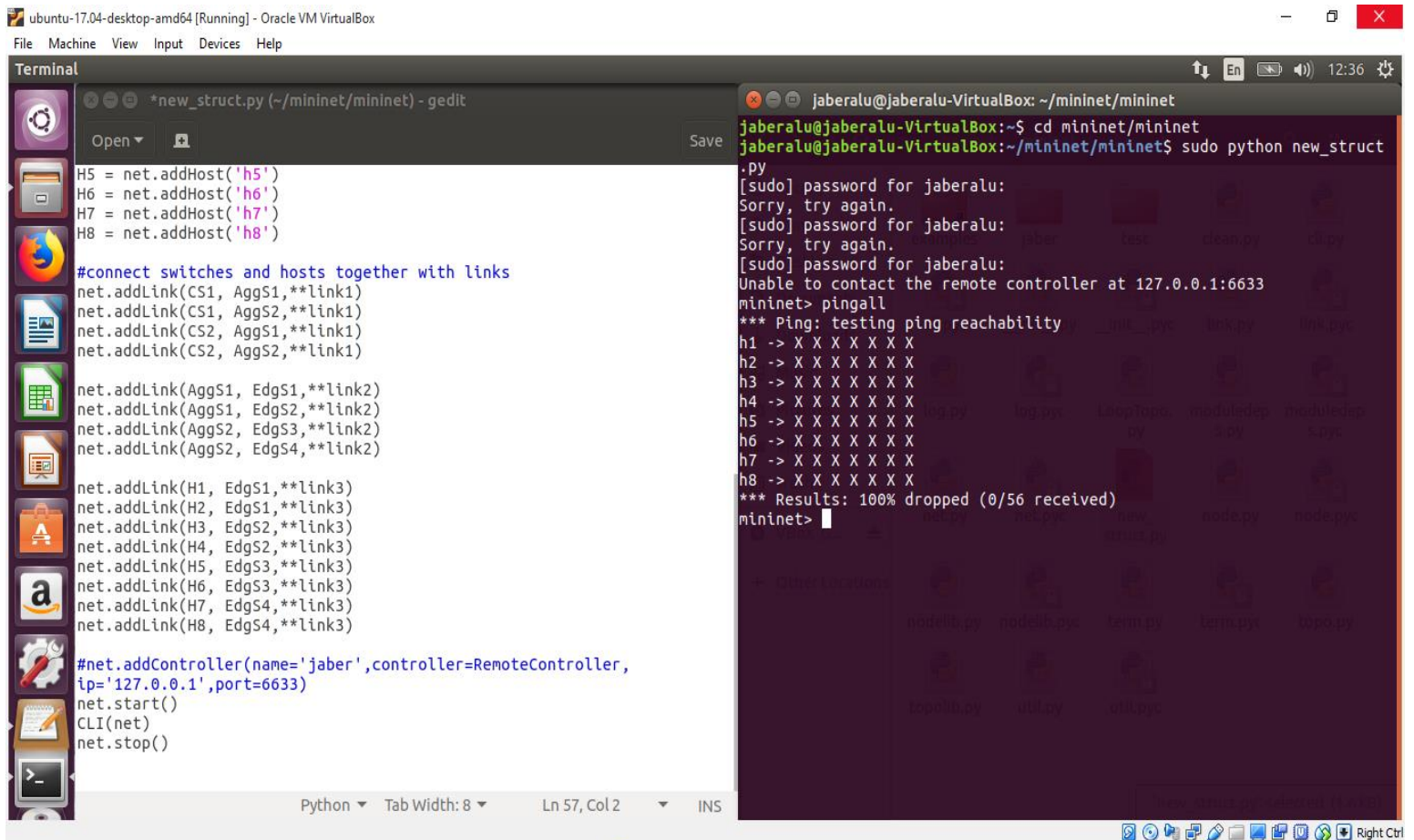


به نام خدا

جابر بابکی ۹۶۱۳۱۰۲۰

شبکه های پیشرفته کامپیوتری

بزرگ فکر کن ، هوشمندانه تصمیم بگیر، اما کوچک شروع کن



```
ubuntu-17.04-desktop-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Terminal
*new_struct.py (~/.mininet/mininet) - gedit
Open Save

H5 = net.addHost('h5')
H6 = net.addHost('h6')
H7 = net.addHost('h7')
H8 = net.addHost('h8')

#connect switches and hosts together with links
net.addLink(CS1, AggS1,**link1)
net.addLink(CS1, AggS2,**link1)
net.addLink(CS2, AggS1,**link1)
net.addLink(CS2, AggS2,**link1)

net.addLink(AggS1, EdgS1,**link2)
net.addLink(AggS1, EdgS2,**link2)
net.addLink(AggS2, EdgS3,**link2)
net.addLink(AggS2, EdgS4,**link2)

net.addLink(H1, EdgS1,**link3)
net.addLink(H2, EdgS1,**link3)
net.addLink(H3, EdgS2,**link3)
net.addLink(H4, EdgS2,**link3)
net.addLink(H5, EdgS3,**link3)
net.addLink(H6, EdgS3,**link3)
net.addLink(H7, EdgS4,**link3)
net.addLink(H8, EdgS4,**link3)

#net.addController(name='jaber',controller=RemoteController,
ip='127.0.0.1',port=6633)
net.start()
CLI(net)
net.stop()

Python Tab Width: 8 Ln 57, Col 2 INS

jaberalu@jaberalu-VirtualBox: ~/mininet/mininet
jaberalu@jaberalu-VirtualBox:~/mininet/mininet$ sudo python new_struct
.py
[sudo] password for jaberalu:
Sorry, try again.
[sudo] password for jaberalu:
Sorry, try again.
[sudo] password for jaberalu:
Unable to contact the remote controller at 127.0.0.1:6633
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X X X X
h2 -> X X X X X X
h3 -> X X X X X X
h4 -> X X X X X X
h5 -> X X X X X X
h6 -> X X X X X X
h7 -> X X X X X X
h8 -> X X X X X X
*** Results: 100% dropped (0/56 received)
mininet>
```

سوال اول گفته شده fat tree را اجرا کنید که در تصویر کد ها به همراه اجرا دیده می شود و از آنجایی که توپولوژی دارای حلقه می باشد در نتیجه ping ها جواب نمی دهد

```
ubuntu-17.04-desktop-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Terminal
jaberalu@jaberalu-VirtualBox: ~/pox
ket:(dns) parsing answers: Bad address format ***Jqc
ket:(dns) parsing answers: Bad address format ***{***}
ket:(dns) parsing answers: Bad address format ***{***}
flow.discovery:link detected: 00-00-00-00-00-03.3 -> 00-00-00-00-00-05.
flow.spanning_tree:1 ports changed
flow.discovery:link detected: 00-00-00-00-00-04.4 -> 00-00-00-00-00-08.
forwarding.l2_learning:Same port for packet from 96:86:57:94:07:72 -> 76
:8f:14 on 00-00-00-00-00-08.1. Drop.
flow.discovery:link detected: 00-00-00-00-00-04.1 -> 00-00-00-00-00-01.
flow.spanning_tree:5 ports changed
flow.discovery:link detected: 00-00-00-00-00-04.2 -> 00-00-00-00-00-02.
forwarding.l2_learning:Same port for packet from 3e:cb:ce:f0:a1:81 -> 2e
:60:8f on 00-00-00-00-00-07.1. Drop.
flow.discovery:link detected: 00-00-00-00-00-04.3 -> 00-00-00-00-00-07.
flow.spanning_tree:1 ports changed
flow.discovery:link detected: 00-00-00-00-00-05.1 -> 00-00-00-00-00-03.
flow.spanning_tree:4 ports changed
forwarding.l2_learning:Same port for packet from 96:86:57:94:07:72 -> 76
:8f:14 on 00-00-00-00-00-08.1. Drop.
flow.discovery:link detected: 00-00-00-00-00-06.1 -> 00-00-00-00-00-03.
flow.spanning_tree:4 ports changed
flow.discovery:link detected: 00-00-00-00-00-07.1 -> 00-00-00-00-00-04.
flow.spanning_tree:4 ports changed
flow.discovery:link detected: 00-00-00-00-00-08.1 -> 00-00-00-00-00-04.
flow.spanning_tree:4 ports changed

jaberalu@jaberalu-VirtualBox: ~/mininet/mininet
jaberalu@jaberalu-VirtualBox:~/mininet/mininet$ sudo python new_struct
.py
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 h5 h6 h7 h8
h3 -> h1 h2 h4 h5 h6 h7 h8
h4 -> h1 h2 h3 h5 h6 h7 h8
h5 -> h1 h2 h3 h4 h6 h7 h8
h6 -> h1 h2 h3 h4 h5 h7 h8
h7 -> h1 h2 h3 h4 h5 h6 h8
h8 -> h1 h2 h3 h4 h5 h6 h7
*** Results: 0% dropped (56/56 received)
mininet>
```

سپس pox را با دستور زیر اجرا کردم

```
./pox.py openflow.spanning_tree forwarding.l2_learning openflow.discovery
```

که spanning tree اجرا شده (البته نیاز هست تا discovery هم اجرا شود) و حلقه های توپولوژی را از بین می برد و در نتیجه همانطور که در تصویر دیده می شود ping ها کاملاً درست اجرا می شود


```
Terminal
jaberalu@jaberalu-VirtualBox: ~/pox
ERROR:packet:(dns) parsing answers: Bad address format 00'000000
ERROR:packet:(dns) parsing answers: Bad address format 00'000000
ERROR:packet:(dns) parsing answers: Bad address format 00(w 0080
ERROR:packet:(dns) parsing answers: Bad address format 00(w 0080
000,R:packet:(dns) parsing answers: Bad address format 000z
000,R:packet:(dns) parsing answers: Bad address format 000z
ERROR:packet:(dns) parsing answers: Bad address format 000000|J0
ERROR:packet:(dns) parsing answers: Bad address format 000000|J0
ERROR:packet:(dns) parsing answers: Bad address format 00S00000r
ERROR:packet:(dns) parsing answers: Bad address format 00S00000r
ERROR:packet:(dns) parsing answers: Bad address format 00p000'7
ERROR:packet:(dns) parsing answers: Bad address format 00p000'7
INFO:openflow.discovery:link detected: 00-00-00-00-00-07.1 -> 00-00-00-00-00-04.3
INFO:openflow.spanning_tree:4 ports changed
ERROR:packet:(dns) parsing answers: Bad address format 00(000000
ERROR:packet:(dns) parsing answers: Bad address format 00(000000
ERROR:packet:(dns) parsing answers: Bad address format 000000006
ERROR:packet:(dns) parsing answers: Bad address format 000000006
ERROR:packet:(dns) parsing answers: Bad address format 0000-000000
ERROR:packet:(dns) parsing answers: Bad address format 0000-000000
ERROR:packet:(dns) parsing answers: Bad address format 00t0}000H(
ERROR:packet:(dns) parsing answers: Bad address format 00t0}000H(
ERROR:packet:(dns) parsing answers: Bad address format 00000000m
ERROR:packet:(dns) parsing answers: Bad address format 00000000m
ERROR:packet:(dns) parsing answers: Bad address format 000j000000
ERROR:packet:(dns) parsing answers: Bad address format 000j000000
ERROR:packet:(dns) parsing answers: Bad address format 00kR0000]
ERROR:packet:(dns) parsing answers: Bad address format 00kR0000]
INFO:openflow.discovery:link detected: 00-00-00-00-00-08.1 -> 00-00-00-00-00-04.4
INFO:openflow.spanning_tree:4 ports changed

jaberalu@jaberalu-VirtualBox: ~/mininet/mininet
[sudo] password for jaberalu:
mininet> h2 ping h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
^@^C
--- 10.0.0.3 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4100ms

mininet> h1 ping h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=282 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=102 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=101 ms
^C
--- 10.0.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 101.318/162.285/282.741/85.177 ms
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=174 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=61.0 ms
^@64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=61.1 ms
^C
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 61.073/99.031/174.875/53.630 ms
mininet> h2 ping h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=143 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=101 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=101 ms
^C
--- 10.0.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 101.591/115.729/143.857/19.893 ms
mininet>
```

در این سوال گفته شده است که که کنترل لیستی تهیه شود به این صورت که هاستی که می خواهد Ping بزند باید ابتدا به h1 دستور ping بزند تا اعتبار سنجی شود و سپس بتواند عملیات خودش را انجام دهد

همانطور که در تصاویر دیده می شد h2 با h3 دستور ping زده شده است اما چون هیچ یک اعتبار سنجی نشده اند ۱۰۰ درصد packet ها loss شده اند، سپس h1 به h2 دستور ping زده شده است تا اعتبار سنجی انجام شود و همینطور برای h3 این اعتبار سنجی را انجام دادیم در نهایت با ping کردن h2 به h2 این عملیات به درستی انجام می شود.

```

def _handle_PacketIn(self, event):
    packet = event.parsed
    def drop(duration=None):
        if duration is not None:
            if not isinstance(duration, tuple):
                duration = (duration, duration)
            msg = of.ofp_flow_mod()
            msg.match = of.ofp_match.from_packet(packet)
            msg.idle_timeout = 14
            msg.hard_timeout = 94
            msg.buffer_id = event.ofp.buffer_id
            self.connection.send(msg)
        elif event.ofp.buffer_id is not None:
            msg = of.ofp_packet_out()
            msg.buffer_id = event.ofp.buffer_id
            msg.in_port = event.port
            self.connection.send(msg)
    if packet.type == packet.ARP_TYPE:
        return
    if packet.src == EthAddr('00:00:00:00:00:01'):
        police_mac.append(packet.dst)
        return
    if (packet.dst in police_mac and packet.src in police_mac) or packet.dst == '00:00:00:00:00:01':
        return
    if (packet.dst not in police_mac) or (packet.src not in police_mac):
        drop()
    return EventHalt

```

توضیحات قسمت اصلی کد AccessControl

در ابتدا یک لیست از هاست های شده را ایجاد کردیم که هاست های دیگر برای اعتبار سنجی باید ابتدا به این هاست درخواستی بزنند در این لیست فقط h1 با آدرس مک ۰۰:۰۰:۰۰:۰۰:۰۰:۰۱ قرار دارد. در کلاس ACModule ما یک event را میگیریم و از connection آن استفاده میکنیم. هنگامی که این event فعال شود ماژول ACModule این رخداد را مدیریت میکند. در این ماژول تابع handle_packet_in هنگامی که بسته ای به کنترلر وارد می شود صدا زده میشود و event به عنوان پارامتر به آن داده می شود. در این تابع ابتدا اطلاعات بسته ورودی را از روی event بدست می آوریم. سپس بر اساس نوع بسته بررسی میکنیم که آیا این بسته قابلیت ارسال دارد یا خیر. سپس تابع drop را نوشتیم که در آن یک پیغام drop را ایجاد کرده و به سویچ ارسال می کند. در آنجایی که در این تمرین نیازی به پیاده سازی flood نداریم آنرا پیاده نکردیم. در صورتی که در این تابع return خالی استفاده کنیم بدین معناست که بسته بصورت عادی مسیر یابی شود .

شرط های تابع handle_packet_in اینگونه است که ابتدا بررسی میکند که اگر بسته از نوع ARP است، هیچگونه کنترل دسترسی روی آن اعمال نشود و بسته بتواند از مبدا به مقصد برسد. در صورتی که مبدا بسته از هاست ۱ بود به معنای آن است که مقصد نیز authenticate شده است پس در صورتی که آدرس مقصد داخل لیست هاست های authenticate شده نبود باید آنرا به آن لیست اضافه کرد و به بسته اجازه داد تا به مقصد برسد. در صورتی که آدرس بسته ورودی داخل لیست هاست های مجوز دار باشد و یا مقصد آن هاست ۱ باشد، بسته میتواند به مسیر خود ادامه دهد و در صورتی که آدرس فرستنده در لیست آدرس های مجاز نبود باید بسته drop شود که این کار با صدا زدن تابع drop() انجام می شود.

همچنین با قرار دادن اولویت عددی (به غیر از عددی صفر) در قسمت connection.addListeners(self,priority=23) کنترلر متوجه می شود که ابتدا باید فایروال را اجرا کند و با این کار اولویت تعیین کردیم.


```

1 id,mac_0,mac_1
2 1,00:00:00:00:00:02,00:00:00:00:00:04
3 2,00:00:00:00:00:03,00:00:00:00:00:05
4 3,00:00:00:00:00:04,00:00:00:00:00:06
5 4,00:00:00:00:00:05,00:00:00:00:00:07

```

```

from pyretic.lib.corelib import *
from pyretic.lib.std import *
from pyretic.modules.mac_learner import mac_learner as act_like_switch
import csv, os
policy_file = "%s/pyretic/pyretic/examples/blocked_hosts.csv" % os.environ[ 'HOME' ]

def main():
    not_allowed = none
    with open(policy_file, 'rb') as f:
        reader = csv.DictReader(f)
        for row in reader:
            not_allowed = not_allowed match(srcmac=MAC(row['mac_0']), dstmac=MAC(row['mac_1'])) match(srcmac=MAC(row['mac_1']), dstmac=MAC(row['mac_0']))

    allowed = ~not_allowed
    return allowed>>act_like_switch()

```

توضیح کد این قسمت به این صورت است که در ابتدا فایل blocked_hosts.csv که داده شده است را میخوانیم با تعریف متغیر global به نام policy_file و همچنین import کردن فرمت CSV برای خواندن این نوع فایل، سپس آنرا در قالب dictionary در متغیر reader میگذاریم. سپس در یک حلقه for این reader را پیمایش میکنیم و در واقع هر سطر فایل CSV را به این صورت بررسی می کنیم که اگر بسته ای با srcmac ستون mac_0 و dstmac ستون mac_1 مطابق بود آن بسته اجازه ی عبور ندارد و همچنین برعکس این قضیه هم بررسی می شود، بعد از اجرای حلقه ما نیاز داریم برعکس not_allowed را داشته باشیم برای با علامت مد آن را not می کنیم و درون allowed می ریزیم و در نهایت return میکنیم.

The image shows two terminal windows. The left window displays a series of error messages: "ERROR:packet:(dns) parsing answers: Bad address format" followed by several lines of hex data. It also shows "INFO:openflow.discovery:link detected: 00-00-00-00-00-07.1 -> 00-00-00-00-00-04.3" and "INFO:openflow.spanning_tree:4 ports changed". The right window shows the output of the command "sudo mn --topo=single,5 --controller=remote --mac", which includes steps for creating a network, adding a controller, adding hosts (h1-h5), adding switches (s1), adding links, configuring hosts, starting the controller and switches, and starting the CLI. It concludes with a ping test: "mininet> pingall" showing "Ping: testing ping reachability" and "Results: 20% dropped (16/20 received)".