

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

ЗВІТ

з лабораторної роботи №3
з дисципліни ”Програмування комп’ютерних та віртуальних мереж”

Тема: Створення лінійної SDN мережі на базі OpenFlow

Варіант №5

Виконав:
Студент 1 курсу, групи ІМ-51мн
Ковальов Олександр

Перевірів:
доцент, Долголенко Олександр Миколайович

Дата здачі: 16.10.2025

Мета роботи. Налаштувати та дослідити лінійну топологію SDN мережі з кількох комутаторів і хостів, перевірити її працездатність та проаналізувати обмін OpenFlow-повідомленнями за допомогою Wireshark dissector.

Завдання: Створити лінійну топологію SDN мережі, що складається з $i + 2$ (де i - номер в списку групи) поєднаних між собою OpenFlow комутаторів, до кожного з котрих підключено по одному хосту та продемонструвати її працездатність з використанням OpenFlow Wireshark dissector.

Хід роботи.

Для початку, встановимо (перевіримо чи встановлені) додаткові пакети, які можуть знадобитися для роботи.

```
mininet@mininet-vm:~$ sudo apt install -y mininet openvswitch-switch wireshark tcpdump traceroute curl
Reading package lists... Done
Building dependency tree
Reading state information... Done
mininet is already the newest version (2.2.2-5ubuntu1).
traceroute is already the newest version (1:2.1.0-2).
wireshark is already the newest version (3.2.3-1).
curl is already the newest version (7.68.0-1ubuntu2.25).
openvswitch-switch is already the newest version (2.13.8-0ubuntu1.4).
tcpdump is already the newest version (4.9.3-4ubuntu0.3).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
mininet@mininet-vm:~$
```

За основу можна взяти скрипт з минулої лабораторної роботи. Частина з описом топології після змін виглядає так:

```
1  def run():
2      net = Mininet(controller=Controller, switch=OVSSwitch)
3      c0 = net.addController('c0')
4
5      variant = 5 + 2
6      for i in range(variant):
7          index = i + 1
8          switch = net.addSwitch(f"s{index}")
9          connected_host = net.addHost(f"h{index}", ip=f"10.0.0.{index}/24")
10         net.addLink(switch, connected_host)
11
12         if index > 1:
13             previous_switch = net.getNodeByName(f"s{index-1}")
14             net.addLink(previous_switch, switch)
15
16     net.start()
17
```

Так як варіант = 5, то потрібно створити 7 комутаторів та стільки ж хостів. Для цього використовується цикл, в якому створюються пристрої з відповідним ім'ям, а потім з'єднуються. Відповідно, була створена лінійна топологія.

Далі, щоб спростити процес запуску, ще минулого разу першим рядком скрипту був встановлений "шебанг" – вказівка, яку утиліту використовувати для інтерпретації коду:

```
1  #!/usr/bin/env python3
```

Якщо надати права на запуск за допомогою утиліти `chmod` та прапорця `-x` (executable), не треба кожного разу вказувати інтерпретатор.

```
mininet@mininet-vm:~/Labs/Lab3$ vim script.py
mininet@mininet-vm:~/Labs/Lab3$ chmod +x ./script.py
mininet@mininet-vm:~/Labs/Lab3$ sudo -E ./script.py
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7
*** Starting controller
50
```

Налаштована топологія:

```
mininet> nodes
available nodes are:
c0 h1 h2 h3 h4 h5 h6 h7 s1 s2 s3 s4 s5 s6 s7
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
h4 h4-eth0:s4-eth1
h5 h5-eth0:s5-eth1
h6 h6-eth0:s6-eth1
h7 h7-eth0:s7-eth1
s1 lo: s1-eth1:h1-eth0 s1-eth2:s2-eth2
s2 lo: s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
s3 lo: s3-eth1:h3-eth0 s3-eth2:s2-eth3 s3-eth3:s4-eth2
s4 lo: s4-eth1:h4-eth0 s4-eth2:s3-eth3 s4-eth3:s5-eth2
s5 lo: s5-eth1:h5-eth0 s5-eth2:s4-eth3 s5-eth3:s6-eth2
s6 lo: s6-eth1:h6-eth0 s6-eth2:s5-eth3 s6-eth3:s7-eth2
s7 lo: s7-eth1:h7-eth0 s7-eth2:s6-eth3
c0
mininet> █
```

Wireshark можна запустити командою:

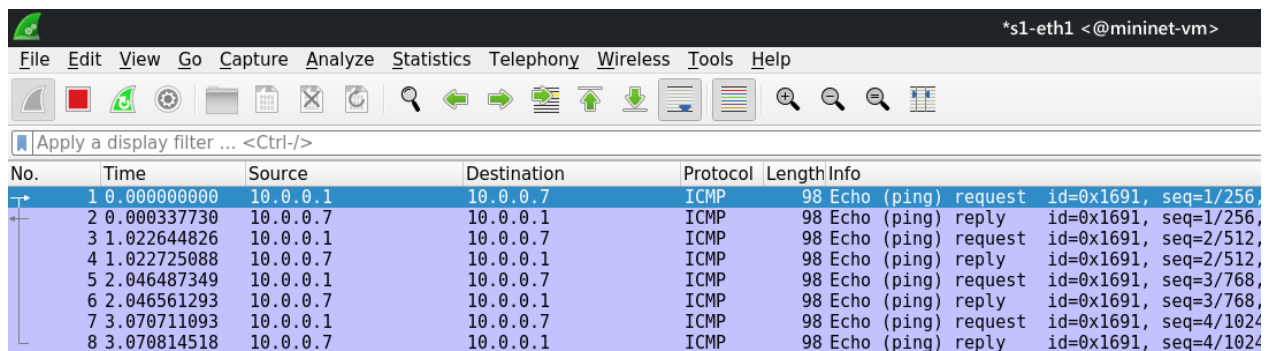
```
1  sudo -E wireshark &
```

Спочатку перевіримо чи взагалі працює можливість відстежувати пакети за допомогою Wireshark. Пропінгуємо сьомий хост з першого:

```
mininet>
mininet> h1 ping -c 4 h7
PING 10.0.0.7 (10.0.0.7) 56(84) bytes of data.
64 bytes from 10.0.0.7: icmp_seq=1 ttl=64 time=2.28 ms
64 bytes from 10.0.0.7: icmp_seq=2 ttl=64 time=4.60 ms
64 bytes from 10.0.0.7: icmp_seq=3 ttl=64 time=0.767 ms
64 bytes from 10.0.0.7: icmp_seq=4 ttl=64 time=0.228 ms

--- 10.0.0.7 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3009ms
rtt min/avg/max/mdev = 0.228/1.968/4.596/1.693 ms
mininet>
```

Почали з'являтися пакети – ICMP ЕЧНО запити та відповіді. В окремих колонках можна побачити, що адреси хостів співпадають.



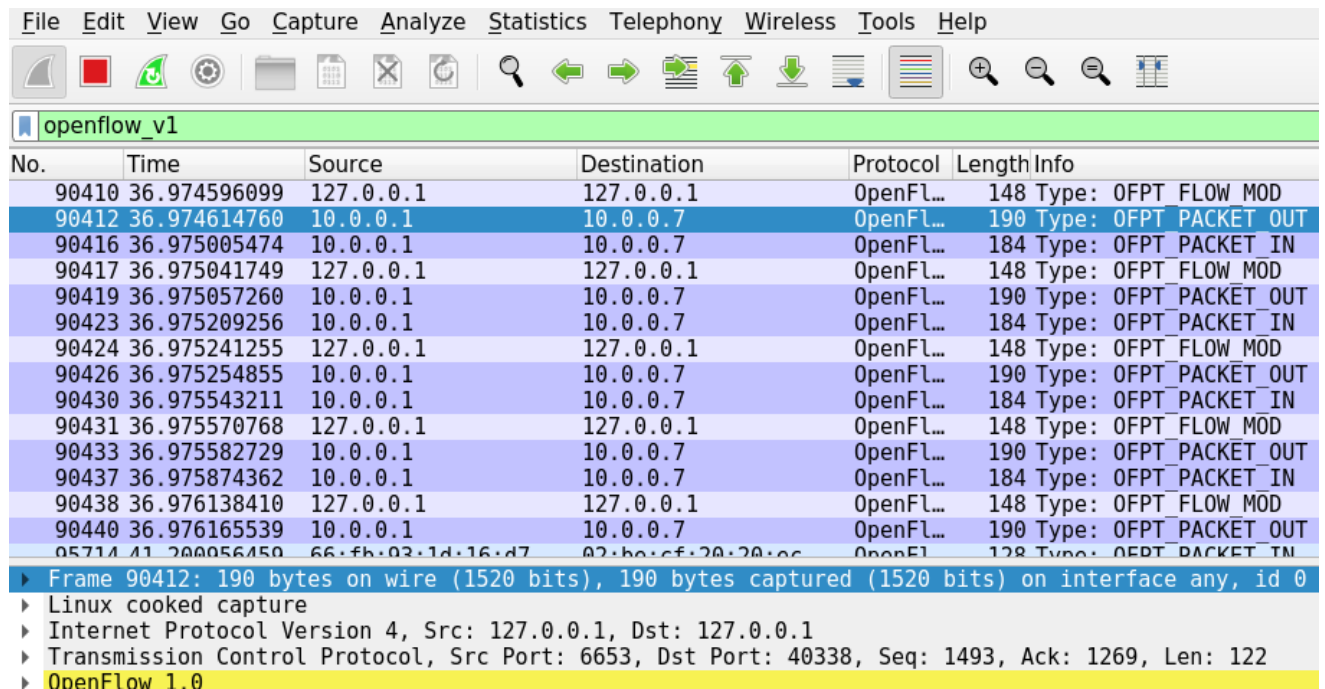
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request id=0x1691, seq=1/256
2	0.000337730	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply id=0x1691, seq=1/256
3	1.022644826	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request id=0x1691, seq=2/512
4	1.022725088	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply id=0x1691, seq=2/512
5	2.046487349	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request id=0x1691, seq=3/768
6	2.046561293	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply id=0x1691, seq=3/768
7	3.070711093	10.0.0.1	10.0.0.7	ICMP	98	Echo (ping) request id=0x1691, seq=4/1024
8	3.070814518	10.0.0.7	10.0.0.1	ICMP	98	Echo (ping) reply id=0x1691, seq=4/1024

Пропінгуємо знову, щоб виконати останнє завдання – провести інспекцію пакетів, надісланих за протоколом OpenFlow. Відповідно нього, якщо на хост надходить або з нього відправляються пакети — інформація про це надсилається за допомогою контролера.

```
mininet> h1 ping -c 4 h7
PING 10.0.0.7 (10.0.0.7) 56(84) bytes of data.
64 bytes from 10.0.0.7: icmp_seq=1 ttl=64 time=3.11 ms
64 bytes from 10.0.0.7: icmp_seq=2 ttl=64 time=4.71 ms
64 bytes from 10.0.0.7: icmp_seq=3 ttl=64 time=0.723 ms
64 bytes from 10.0.0.7: icmp_seq=4 ttl=64 time=0.125 ms

--- 10.0.0.7 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3021ms
rtt min/avg/max/mdev = 0.125/2.165/4.706/1.842 ms
mininet>
```

Насамкінець, маємо результат – чотири пари пакетів з типом "OFPT_PACKET_IN" та "OFPT_PACKET_OUT", які були відправлені через роботу утиліти ping. Також, видно джерело та призначення – IP-адреси хостів h1 та h7.



The image shows a Wireshark network traffic capture titled 'openflow_v1'. The main display area shows a list of captured packets. The selected packet is number 90412, which is an OpenFlow 'PACKET_OUT' message. The packet details pane on the right shows the structure of this message: it is an OpenFlow 1.0 message, an Internet Protocol Version 4 packet from 127.0.0.1 to 127.0.0.1, and a Transmission Control Protocol segment from port 6653 to port 40338. The packet length is 190 bytes.

No.	Time	Source	Destination	Protocol	Length	Info
90410	36.974596099	127.0.0.1	127.0.0.1	OpenFl...	148	Type: OFPT_FLOW_MOD
90412	36.974614760	10.0.0.1	10.0.0.7	OpenFl...	190	Type: OFPT_PACKET_OUT
90416	36.975005474	10.0.0.1	10.0.0.7	OpenFl...	184	Type: OFPT_PACKET_IN
90417	36.975041749	127.0.0.1	127.0.0.1	OpenFl...	148	Type: OFPT_FLOW_MOD
90419	36.975057260	10.0.0.1	10.0.0.7	OpenFl...	190	Type: OFPT_PACKET_OUT
90423	36.975209256	10.0.0.1	10.0.0.7	OpenFl...	184	Type: OFPT_PACKET_IN
90424	36.975241255	127.0.0.1	127.0.0.1	OpenFl...	148	Type: OFPT_FLOW_MOD
90426	36.975254855	10.0.0.1	10.0.0.7	OpenFl...	190	Type: OFPT_PACKET_OUT
90430	36.975543211	10.0.0.1	10.0.0.7	OpenFl...	184	Type: OFPT_PACKET_IN
90431	36.975570768	127.0.0.1	127.0.0.1	OpenFl...	148	Type: OFPT_FLOW_MOD
90433	36.975582729	10.0.0.1	10.0.0.7	OpenFl...	190	Type: OFPT_PACKET_OUT
90437	36.975874362	10.0.0.1	10.0.0.7	OpenFl...	184	Type: OFPT_PACKET_IN
90438	36.976138410	127.0.0.1	127.0.0.1	OpenFl...	148	Type: OFPT_FLOW_MOD
90440	36.976165539	10.0.0.1	10.0.0.7	OpenFl...	190	Type: OFPT_PACKET_OUT
95714	41.200056450	66.fh.03.1d.16.d7	02.ba.cf.20.20.ac	OpenFl...	128	Type: OFPT_PACKET_IN

Frame 90412: 190 bytes on wire (1520 bits), 190 bytes captured (1520 bits) on interface any, id 0

- Linux cooked capture
- Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
- Transmission Control Protocol, Src Port: 6653, Dst Port: 40338, Seq: 1493, Ack: 1269, Len: 122
- OpenFlow 1.0

Висновок. У ході лабораторної роботи було створено та досліджено лінійну топологію SDN-мережі, що складається із семи OpenFlow-комутаторів і семи хостів. За допомогою контролера було забезпечено зв'язок між вузлами та передано трафік між крайніми хостами. Робота мережі була перевірена за допомогою утиліти Wireshark, де вдалося проаналізувати обмін OpenFlow-повідомленнями типу "PACKET_IN" та "PACKET_OUT". Отримані результати підтверджують коректність налаштування SDN-топології та функціонування контролера OpenFlow.

Лістинг програми.

```
1  #!/usr/bin/env python3
2  from mininet.net import Mininet
3  from mininet.node import Controller, OVSSwitch
4  from mininet.cli import CLI
5  from mininet.log import setLogLevel, info
6  import os
7
8  def run():
9      net = Mininet(controller=Controller, switch=OVSSwitch)
10     c0 = net.addController('c0')
11
12     variant = 7
13     for i in range(variant):
14         index = i + 1
15         switch = net.addSwitch(f"s{index}")
16         connected_host = net.addHost(f"h{index}", ip=f"10.0.0.{index}/24")
17         net.addLink(switch, connected_host)
18
19         if index > 1:
20             previous_switch = net.getNodeByName(f"s{index-1}")
21             net.addLink(previous_switch, switch)
22
23     net.start()
24
25     info('*** Adding internal management port s1-mgmt with IP 10.0.0.254/24\n')
26     os.system('ovs-vsctl add-port s1 s1-mgmt -- set interface s1-mgmt type=internal')
27
28     os.system('ip addr add 10.0.0.254/24 dev s1-mgmt || true')
29     os.system('ip link set s1-mgmt up || true')
30
31     info('*** Setup complete - entering CLI\n')
32     CLI(net)
33     net.stop()
34
35 if __name__ == "__main__":
36     setLogLevel('info')
37     run()
38
```