

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

Факультет інформатики та обчислювальної техніки  
Кафедра обчислювальної техніки

## **ЗВІТ**

**з лабораторної роботи №1**  
**з дисципліни ”Програмування комп’ютерних та віртуальних мереж”**

**Тема: Ознайомлення з Mininet**

**Варіант №5**

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

Перевірів:  
доцент, Долголенко Олександр Миколайович

Дата здачі: 24.09.2025

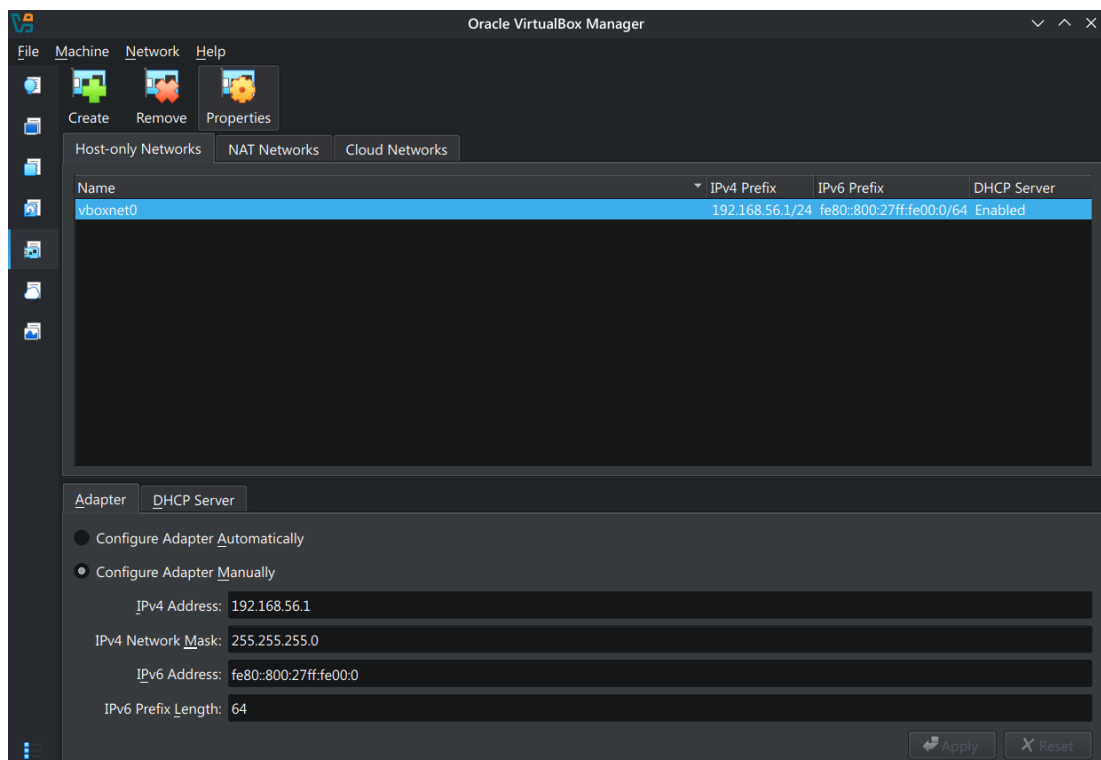
**Мета роботи.** Ознайомитись з Mininet - середовищем для моделювання комп'ютерної SDN мережі.

**Завдання:**

1. Встановити Mininet;
2. Створити мінімальну готову топологію SDN мережі;
3. Продемонструвати (з коментарями) використання основних команд: `nodes`, `net`, `ifconfig`, `down`, `up`, `route`, `ping`, `pingall`, `iperf`, `ps`, `xterm`.

**Хід роботи.**

Для використання Mininet був встановлений гіпервізор VirtualBox. Після імпортування віртуальної машини, її можна запустити з наданим логіном та паролем. Але, щоб мати змогу підключитися за допомогою протоколу SSH, треба додати "Host-only Network" ресурс. Цей процес продемонстрований на зображенні. Далі, в налаштуваннях віртуальної машини, цей адаптер треба встановити під номером 2, який на системі повинен відображатися як `eth1`.



Але, скоріш за все, після запуску машини, мережевий інтерфейс не буде встановленим. Для виправлення цієї проблеми треба застосувати команду:

```
1 sudo dhclient eth1
```

Після чого вже можна встановити зв'язок між хостом та гостьовою машиною. Щоб кожного разу не вводити пароль `mininet` при вході в гостьову систему, бажано

додати публічний ключ RSA на сервер. Для цього треба згенерувати ключ якщо його немає, скопіювати на гостьову машину за допомогою утиліти `scp`, створити файл в якому SSH сервер перевіряє "авторизовані" ключі, надати відповідні права файлам. Всі виконані дії продемонстровані на зображенні.

```
xairaven@laptop ~
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/xairaven/.ssh/id_rsa):
/home/xairaven/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase for "/home/xairaven/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/xairaven/.ssh/id_rsa
Your public key has been saved in /home/xairaven/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:390riat6HNeHb+EAYHz3G5WpieVj4xSNzc1A++XvqF8 xairaven@laptop
The key's randomart image is:
+---[RSA 3072]-----+
|      .Bo+|
|      + . . + 0o|
|      . o . = * .|
|      . . % o .|
|      S . * * o|
|      o = * . .|
|      . + * + E|
|      o . * + .|
|      .o...+o+..|
+---[SHA256]-----+
xairaven@laptop ~
$ scp ~/.ssh/id_rsa.pub mininet@192.168.56.101:~/
mininet@192.168.56.101's password:
id_rsa.pub                                100% 56
9      2.6MB/s   00:00
xairaven@laptop ~
$ ssh -p 22 mininet@192.168.56.101
mininet@192.168.56.101's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-216-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Tue Sep 23 13:34:01 2025
mininet@mininet-vm:~$ cd ~/ && mkdir -p .ssh && chmod 700 .ssh && cd .ssh && touch authorized_keys2 &&
chmod 600 authorized_keys2 && cat ../id_rsa.pub >> authorized_keys2 && rm ../id_rsa.pub && cd ..
mininet@mininet-vm:~$
```

Далі, треба запустити Wireshark. Щоб мати змогу користуватись GUI застосунками через SSH, треба налаштувати X11 тунелінг, де X11 - віконна система. Щоб все працювало справно, треба щоб на хостовій системі працював X сервер. Так як лабораторна робота виконується на дистрибутиві Fedora 42 з використанням середовища робочого столу KDE Plasma, де за замовчуванням встановлена сесія протоколу Wayland, треба довстановити потрібні пакети, а саме `xorg-x11-xauth` та `xorg-x11-server-Xwayland`.

```
xairaven@laptop ~
$ sudo dnf install -y xorg-x11-xauth xorg-x11-server-Xwayland
Updating and loading repositories:
Repositories loaded.
Package "xorg-x11-xauth-1:1.1.3-3.fc42.x86_64" is already installed.
Package "xorg-x11-server-Xwayland-24.1.8-1.fc42.x86_64" is already installed.

Nothing to do.
xairaven@laptop ~
$
```

Далі, вже на гостьовій машині, треба змінити налаштування SSH серверу, тобто відредагувати файл `/etc/ssh/sshd_config`. Якщо потрібних налаштувань стосовно X11 немає, то треба їх додати.

```
$OpenBSD: sshd_config,v 1.103 2018/04/09 20:41:22 tj Exp $
# This is the sshd server system-wide configuration file. See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented. Uncommented options override the
# default value.

Include /etc/ssh/sshd_config.d/*.conf

#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

X11Forwarding yes
X11DisplayOffset 10
X11UseLocalhost yes

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
#PermitRootLogin prohibit-password
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

"/etc/ssh/sshd_config" [readonly] 128L, 3367C
```

Далі, можна перевірити роботу механізму тунелінгу. По-перше, треба підключитися до гостьової машини використовуючи ключ `-X`, що дозволяє працювати даному механізму:


```
1 ssh -X -p 22 mininet@192.168.56.101
```

Для перевірки, можна вивести змінну `$DISPLAY` (вказує, куди переводити відображення графічних додатків), або ж запустити додаток `xclock`.

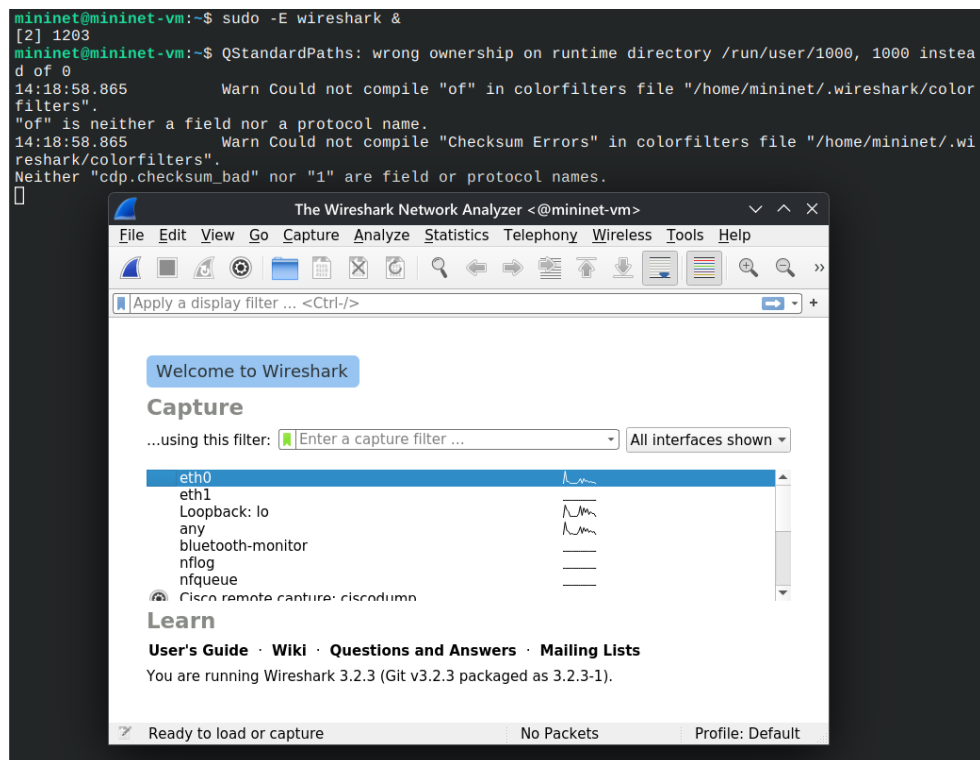
```
xairaven@laptop ~
$ ssh -X -p 22 mininet@192.168.56.101
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-216-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/pro
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Tue Sep 23 14:10:10 2025 from 192.168.56.1
mininet@mininet-vm:~$ echo $DISPLAY
localhost:10.0
mininet@mininet-vm:~$ xclock
```



Далі можна запустити Wireshark, який успішно запуститься, якщо використати команду `sudo -E wireshark &`. Тобто, утиліта запускається у фоні щоб не перекривати потік введення, і окремо використовується прапорець `-E`, який потрібен щоб при використанні `sudo` було збережене оточення користувача, а саме вищезазначена змінна `$DISPLAY`.



Була запущена утиліта `mininet`, тобто, водночас, була створена мінімальна топологія - OpenFlow контролер, два пристрої та коммутатор.

```
mininet@mininet-vm:~$ sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> help
```

Відобразити всі вузли і зв'язки між ними можна відповідно за допомогою команд `nodes` та `net`. Можна побачити мінімальну топологію та створені зв'язки: першого хоста зі свічем, другого хоста з ним, та зв'язки комутатора з ними обома.

```
mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet>
```

Стан мережевих інтерфейсів можна перевірити за допомогою команди `ifconfig` -а. Відповідно, на зображенні - мережеві інтерфейси двох хостів.

```
mininet> h1 ifconfig -a
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.1 netmask 255.0.0.0 broadcast 10.255.255.255
    ether 02:67:5c:8f:1b:14 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

mininet> h2 ifconfig -a
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.2 netmask 255.0.0.0 broadcast 10.255.255.255
    ether 5a:3a:ed:8c:41:4e txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

mininet>
```

Стан комутатора майже повністю співпадає зі станом інтерфейсів системи - це пов'язано з тим, що він працює в просторі імен кореневої мережі.

```
(mininet) 192.168.56.101 — Konsole

(mininet) 192.168.56.101
mininet>
mininet> s1 ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.56.101 netmask 255.255.255.0 broadcast 192.168.56.255
    ether 08:00:27:2d:19:50 txqueuelen 1000 (Ethernet)
    RX packets 6871 bytes 1395045 (1.3 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 60545 bytes 85711970 (85.7 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    ether 08:00:27:9f:47:2b txqueuelen 1000 (Ethernet)
    RX packets 369 bytes 36868 (36.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 370 bytes 38914 (38.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 7260 bytes 82554020 (82.5 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 7260 bytes 82554020 (82.5 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ovs-system: flags=4098<BROADCAST,MULTICAST> mtu 1500
    ether f6:38:ab:25:42:cc txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

s1: flags=4098<BROADCAST,MULTICAST> mtu 1500
    ether 32:ee:77:33:c9:46 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

s1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    ether 9e:0c:76:b3:2b:03 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(mininet) 192.168.56.101
mininet> s1 ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.56.101 netmask 255.255.255.0 broadcast 192.168.56.255
    ether 08:00:27:2d:19:50 txqueuelen 1000 (Ethernet)
    RX packets 7046 bytes 1416609 (1.4 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 60687 bytes 85737322 (85.7 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    ether 08:00:27:9f:47:2b txqueuelen 1000 (Ethernet)
    RX packets 372 bytes 37112 (37.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 373 bytes 31154 (31.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 7284 bytes 82555396 (82.5 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 7284 bytes 82555396 (82.5 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

s1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    ether 9e:0c:76:b3:2b:03 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

s1-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    ether 3a:ed:7e:81:0f:01 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

mininet@mininet-vm:~$
```

Список процесів, запущених на пристрої, можна переглянути використовуючи команду ps.

```
mininet> h1 ps -a
  PID TTY          TIME CMD
  586 tty1          00:00:00 bash
 1257 pts/0          00:00:00 sudo
 1258 pts/0          00:00:00 mn
 1298 pts/1          00:00:00 controller
 1432 pts/0          00:00:00 stty
 1433 pts/2          00:00:00 ps
mininet> s1 ps -a
  PID TTY          TIME CMD
  586 tty1          00:00:00 bash
 1257 pts/0          00:00:00 sudo
 1258 pts/0          00:00:00 mn
 1298 pts/1          00:00:00 controller
 1435 pts/4          00:00:00 ps
mininet>
```

Для перевірки доступності вузлів використовується команда ping. Перший хост запитує ARP для MAC-адреси другого, що призводить до надсилання повідомлення packet\_in до контролера. Потім контролер надсилає повідомлення packet\_out, щоб переслати широкомовний пакет на інші порти комутатора (у цьому прикладі



єдиний інший порт даних). Другий хост бачить ARP-запит і надсилає відповідь. Ця відповідь надходить до контролера, який надсилає її до першого хоста.

Тепер перший хост знає MAC-адресу другого та може надіслати свій ping через ICMP Echo Request. Цей запит разом із відповідною відповіддю від другого хоста надсилається до контролера та призводить до виводу запису потоку (разом із фактичним надсиленням пакетів).

Видно, що час роботи утиліти набагато менший час для другої спроби. Це відбувається через повторення операції, тобто пакети одразу проходять через комунікатор.

Простіший спосіб виконати цей тест - скористатися вбудованою командою pingall в інтерфейсі командного рядка Mininet, яка виконує ping для всіх пар.

```
mininet> h1 ping -c 1 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=1.29 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.290/1.290/1.290/0.000 ms
mininet> h1 ping -c 1 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=0.172 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.172/0.172/0.172/0.000 ms
mininet> h2 ping -c 1 h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.444 ms

--- 10.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.444/0.444/0.444/0.000 ms
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> 
```



Для перевірки пропускної здатності мережі, можна запустити тест за допомогою команди `iperf`. Це можна зробити і безпосередньо в консолі самої системи, а не застосунку.

```
*** Starting CLI:
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['48.4 Gbits/sec', '48.7 Gbits/sec']
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 19.922 seconds
mininet@mininet-vm:~$ sudo mn --test iperf
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['46.8 Gbits/sec', '47.1 Gbits/sec']
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 10.916 seconds
mininet@mininet-vm:~$
```

Для перегляду або проведення маніпуляцій з таблицею IP-маршрутизації використовується команда `route`. Можна помітити, що у комутатора є шлях до інтерфейсу системи - пояснення вже є вище.

```
mininet> h1 route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
10.0.0.0         0.0.0.0         255.0.0.0        U        0      0      0 h1-eth0
mininet> h2 route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
10.0.0.0         0.0.0.0         255.0.0.0        U        0      0      0 h2-eth0
mininet> s1 route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
default          _gateway         0.0.0.0          UG        0      0      0 eth1
10.0.2.0         0.0.0.0         255.255.255.0    U        0      0      0 eth1
192.168.56.0     0.0.0.0         255.255.255.0    U       100     0      0 eth0
mininet> █
```

Для тестування перенаправлення шляхів чи інших параметрів, можна вмикати або вимикати з'єднання за допомогою команд `link up` та `link down`. На зображенні можна спостерігати, що після вимкнення з'єднання не можливо передати ICMP пакет з одного пристрою на інший.

```
mininet> h1 ping -c 4 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=1.11 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.572 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.090 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.077 ms

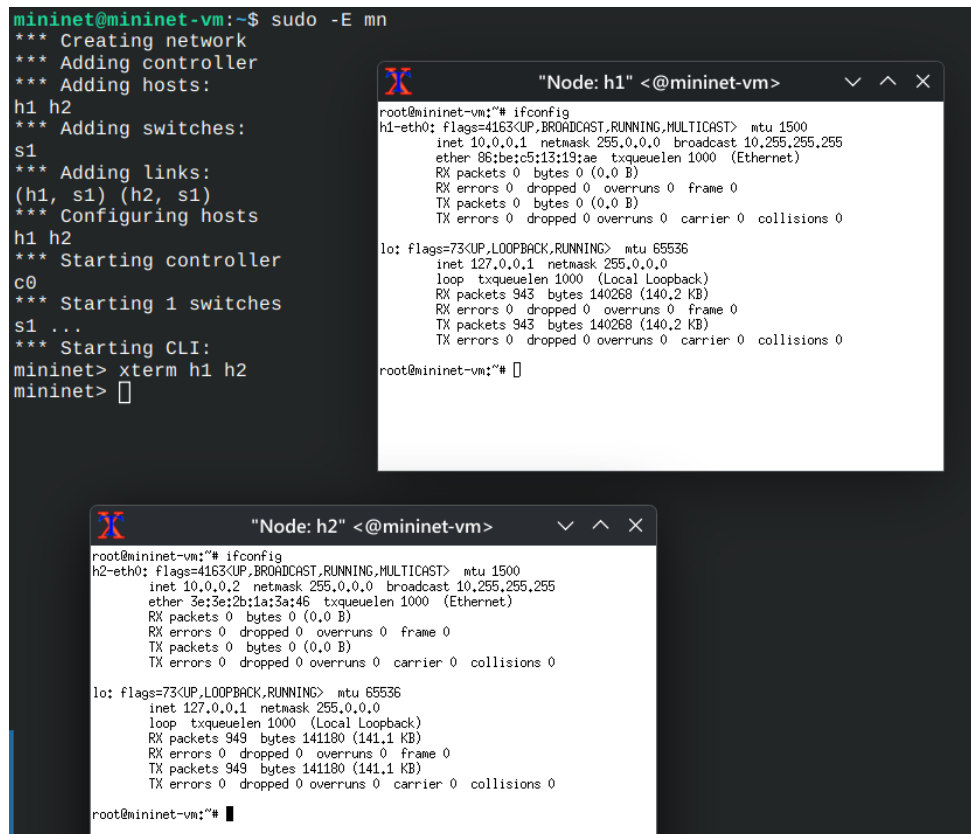
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3050ms
rtt min/avg/max/mdev = 0.077/0.462/1.110/0.423 ms
mininet> link s1 h1 down
mininet> h1 ping -c 4 h2
ping: connect: Network is unreachable
mininet> link s1 h1 up
mininet> h1 ping -c 1 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.390 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.390/0.390/0.390/0.000 ms
mininet> █
```

Насамкінець, за допомогою X11-тунелінгу можна запустити термінали пристроїв. Команда `sudo -E mn -x` дає доступ до всіх терміналів відразу, але можна отримати, наприклад, до двох:

```
1 mininet> xterm h1 h2
```

На зображенні можна побачити роботу вищезазначеної команди.



The image shows a terminal window for Mininet and two separate terminal windows for nodes h1 and h2. The main terminal window shows the following commands and output:

```
mininet@mininet-vm:~$ sudo -E mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> xterm h1 h2
mininet>
```

The two node terminal windows show the output of the `ifconfig` command for each node:

```
root@mininet-vm:~# ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.1 netmask 255.0.0.0 broadcast 10.255.255.255
    ether 86:be:c5:13:19:ae txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 943 bytes 140268 (140.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 943 bytes 140268 (140.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@mininet-vm:~#
```

```
root@mininet-vm:~# ifconfig
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.2 netmask 255.0.0.0 broadcast 10.255.255.255
    ether 3e:3e:2b:1a:3a:46 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 949 bytes 141180 (141.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 949 bytes 141180 (141.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@mininet-vm:~#
```

**Висновок.** Було ознайомлено з Mininet, зокрема з встановленням віртуальної машини, налаштуванням SSH і X11 та запуском Wireshark. Було створено мінімальну топологію та відпрацьовано базові команди (`nodes`, `net`, `ifconfig`, `ping`, `iperf`, `xterm`). Було проведено спостереження за обміном через контролер і за поведінкою при `link down/up`; зроблено висновок, що Mininet ефективний для навчальної симуляції SDN.