

# **Лабораторная работа №1**

**Введение в Mininet**

Замбалова Дина Владимировна

# Содержание

Цель работы	4
Теоретическое введение	5
Выполнение лабораторной работы	6
Выводы	23
Список литературы	24

# Список иллюстраций

1	Импорт конфигураций . . . . .	6
2	Параметры импорта . . . . .	7
3	Настройка сети . . . . .	8
4	Настройка сети . . . . .	8
5	Запуск mininet . . . . .	9
6	Подключение к mininet через SSH . . . . .	9
7	Просмотр IP-адресов машины . . . . .	10
8	Файл /etc/netplan/01-netcfg.yaml . . . . .	10
9	Обновление Mininet . . . . .	11
10	Номер установленной версии mininet . . . . .	11
11	Настройка соединения X11 для суперпользователя . . . . .	12
12	Работа с Mininet с помощью командной строки . . . . .	13
13	Работа с Mininet с помощью командной строки . . . . .	13
14	Работа с Mininet с помощью командной строки . . . . .	14
15	Проверка связности хостов . . . . .	15
16	Очистка предыдущего экземпляра Mininet . . . . .	15
17	Настройка параметров XTerm . . . . .	16
18	Запуск и настройка Xserver . . . . .	16
19	Запуск putty и добавление опции перенаправления X11 . . . . .	17
20	Запуск putty и добавление опции перенаправления X11 . . . . .	18
21	Добавление двух хостов и одного коммутатора . . . . .	19
22	Настройка IP-адреса на хостах . . . . .	19
23	Проверка назначенных IP-адресов для h2 и проверка соединения между хостами . . . . .	20
24	Проверка автоматического назначения адресов . . . . .	21
25	Отображение IP-адреса, назначенного хосту h1 . . . . .	21
26	Сохранение топологии . . . . .	22
27	Изменение прав доступа к файлам в каталоге проекта . . . . .	22

## Цель работы

Основной целью работы является развёртывание в системе виртуализации (например, в VirtualBox) mininet, знакомство с основными командами для работы с Mininet через командную строку и через графический интерфейс.

# Теоретическое введение

Mininet[1] – это эмулятор компьютерной сети. Под компьютерной сетью подразумеваются простые компьютеры — хосты, коммутаторы, а так же OpenFlow-контроллеры. С помощью простейшего синтаксиса в примитивном интерпретаторе команд можно разворачивать сети из произвольного количества хостов, коммутаторов в различных топологиях и все это в рамках одной виртуальной машины(VM). На всех хостах можно изменять сетевую конфигурацию, пользоваться стандартными утилитами(`ifconfig`, `ping`) и даже получать доступ к терминалу. На коммутаторы можно добавлять различные правила и маршрутизировать трафик.

Mininet создает реалистичную виртуальную сеть, выполняя реальный код ядра, коммутатора и приложения на одной машине (VM, облачной или собственной) за считанные секунды с помощью одной команды `sudo mn`.

# Выполнение лабораторной работы

Перейдем в репозиторий Mininet, скачаем актуальный релиз ovf-образа виртуальной машины. Запустим систему виртуализации и импортируем файл .ovf и укажем параметры импорта (рис. 1;2).

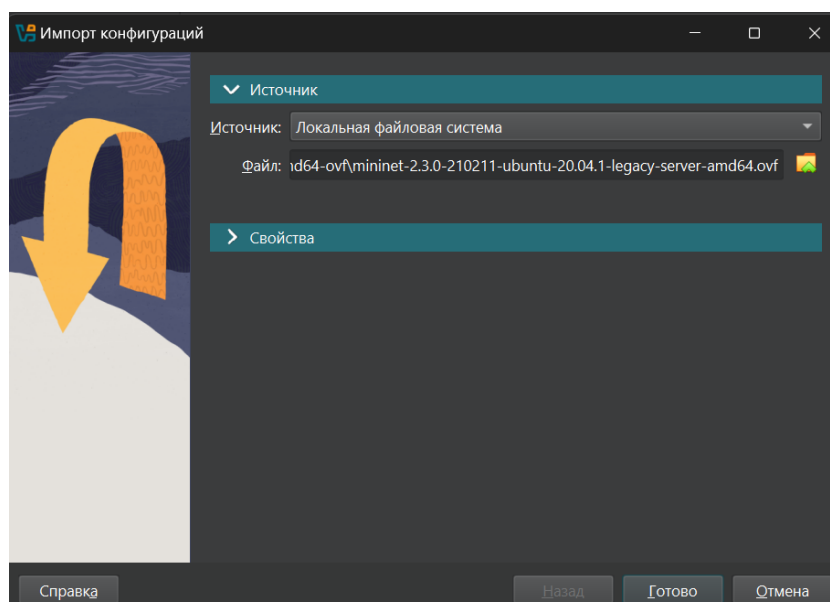


Рис. 1: Импорт конфигураций

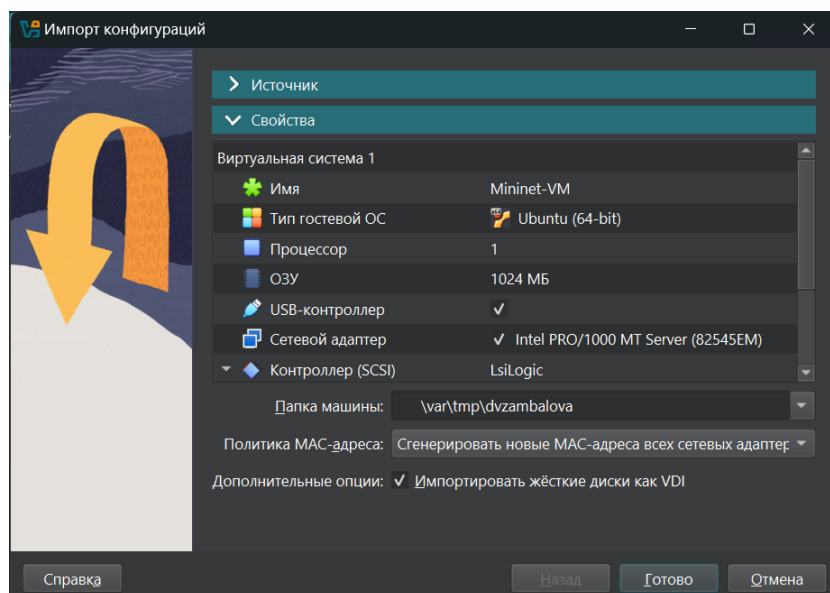


Рис. 2: Параметры импорта

Перейдем в настройки системы виртуализации и уточним параметры настройки виртуальной машины. В частности, для VirtualBox выберем импортированную виртуальную машину и перейдите в меню “Машина -> Настроить”. Перейдем к опции «Система». Если внизу этого окна есть сообщение об обнаружении неправильных настроек, то, следуя рекомендациям, внесем исправления (изменим тип графического контроллера на рекомендуемый). В настройках сети первый адаптер должен иметь подключение типа NAT (рис. 3). Для второго адаптера укажите тип подключения host-only network adapter (виртуальный адаптер хоста), который в дальнейшем вы будете использовать для входа в образ виртуальной машины (рис. 4).

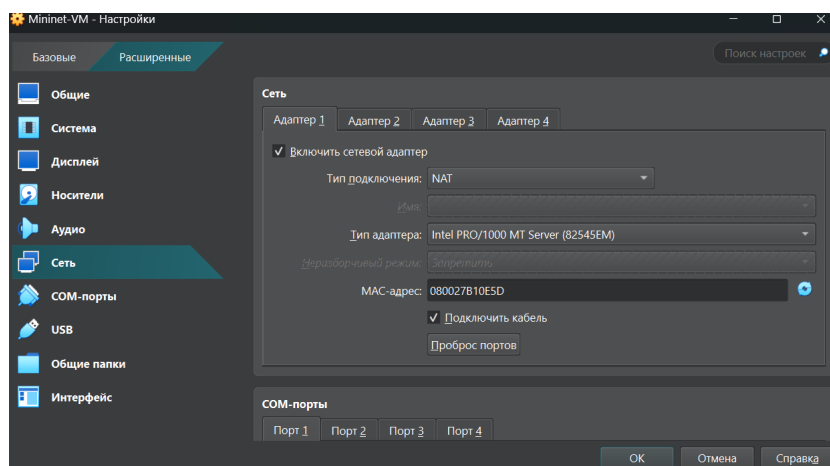


Рис. 3: Настройка сети

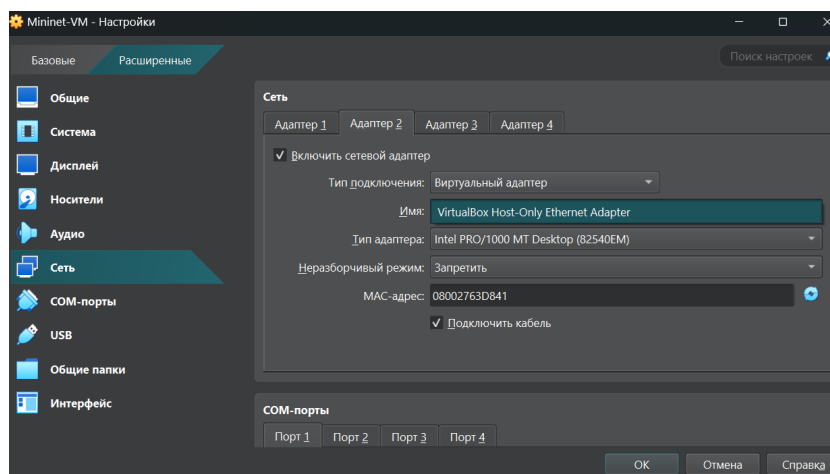
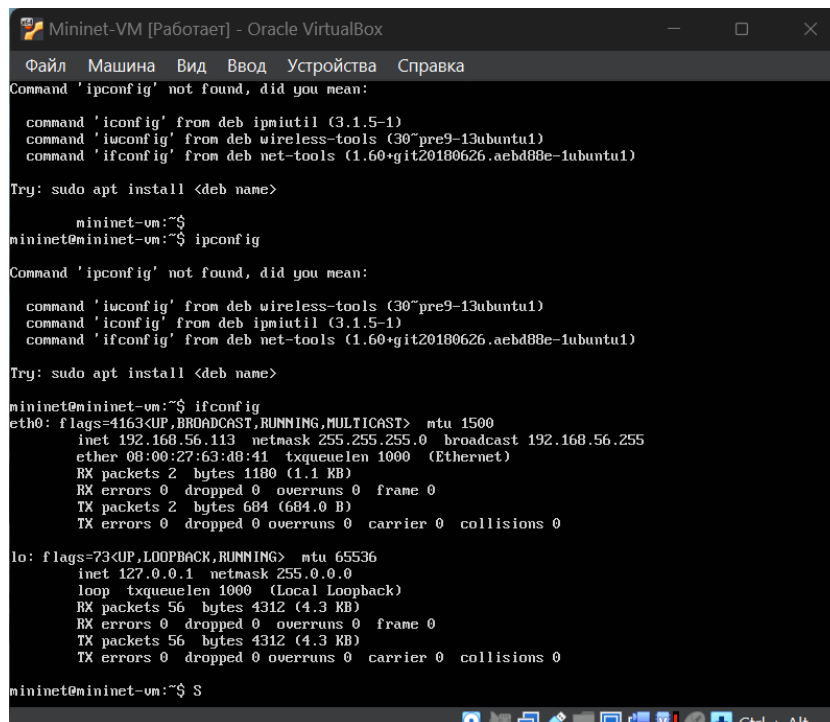


Рис. 4: Настройка сети

Запустим виртуальную машину с Mininet. Залогинимся в виртуальную машину: - login: mininet - password: mininet

Посмотрите адрес машины с помощью ifconfig (рис. 5).





```
Mininet-VM [Работает] - Oracle VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Command 'ipconfig' not found, did you mean:

command 'iconfig' from deb ipmiutil (3.1.5-1)
command 'iwconfig' from deb wireless-tools (30~pre9-13ubuntu1)
command 'ifconfig' from deb net-tools (1.60+git20180626.aebd88e-1ubuntu1)

Try: sudo apt install <deb name>

mininet-vm:~$
mininet@mininet-vm:~$ ipconfig

Command 'ipconfig' not found, did you mean:

command 'iwconfig' from deb wireless-tools (30~pre9-13ubuntu1)
command 'iconfig' from deb ipmiutil (3.1.5-1)
command 'ifconfig' from deb net-tools (1.60+git20180626.aebd88e-1ubuntu1)

Try: sudo apt install <deb name>

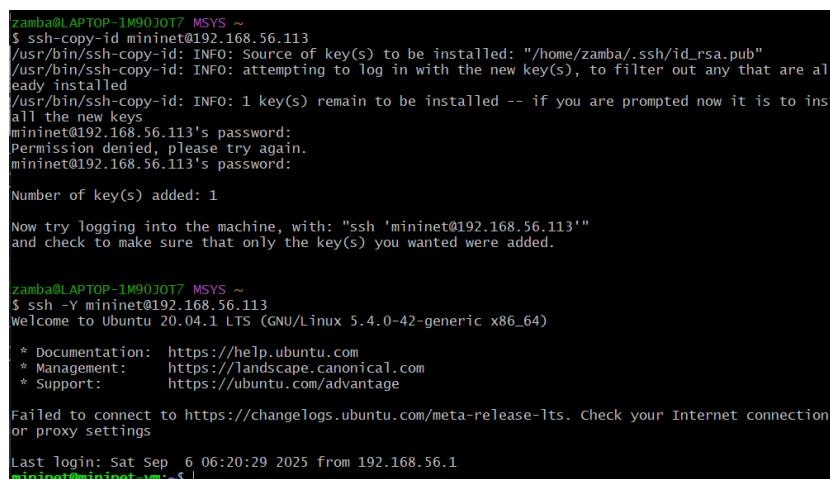
mininet@mininet-vm:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.56.113  netmask 255.255.255.0  broadcast 192.168.56.255
    ether 08:00:27:63:d8:41  txqueuelen 1000  (Ethernet)
    RX packets 2  bytes 1180 (1.1 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 2  bytes 684 (684.0 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 56  bytes 4312 (4.3 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 56  bytes 4312 (4.3 KB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

mininet@mininet-vm:~$ S
```

Рис. 5: Запуск mininet

Подключимся к виртуальной машине (из терминала хостовой машины). Настроим ssh-подсоединение по ключу к виртуальной машине. Вновь подключимся к виртуальной машине и убедимся, что подсоединение происходит успешно и без ввода пароля (рис. 6).



```
zamba@LAPTOP-1M90JOT7: MSYS ~
$ ssh-copy-id mininet@192.168.56.113
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/zamba/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install all the new keys
mininet@192.168.56.113's password:
Permission denied, please try again.
mininet@192.168.56.113's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'mininet@192.168.56.113'"
and check to make sure that only the key(s) you wanted were added.

zamba@LAPTOP-1M90JOT7: MSYS ~
$ ssh -Y mininet@192.168.56.113
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-42-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Sat Sep  6 06:20:29 2025 from 192.168.56.1
mininet@mininet-vm:~$
```

Рис. 6: Подключение к mininet через SSH

После подключения к виртуальной машине mininet посмотрим IP-адреса машины. Активен только внутренний адрес машины вида 192.168.x.y, поэтому активируем второй интерфейс (рис. 7).

```
mininet@mininet-vm:~$ sudo dhclient eth1
mininet@mininet-vm:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.56.113 netmask 255.255.255.0 broadcast 192.168.56.255
    ether 08:00:27:63:d8:41 txqueuelen 1000 (Ethernet)
    RX packets 928 bytes 111199 (111.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 294 bytes 47359 (47.3 KB)
    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:b1:0e:5d txqueuelen 1000 (Ethernet)
    RX packets 2 bytes 1180 (1.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2 bytes 684 (684.0 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 1840 bytes 141000 (141.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1840 bytes 141000 (141.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Рис. 7: Просмотр IP-адресов машины

Для удобства дальнейшей работы добавим для mininet указание на использование двух адаптеров при запуске. Для этого требуется перейти в режим суперпользователя и внести изменения в файл /etc/netplan/01-netcfg.yaml виртуальной машины mininet. В результате файл /etc/netplan/01-netcfg.yaml должен иметь следующий вид (рис. 8).

```
/etc/netplan/01-netcfg.yaml [-M--] 16 L:[ 1+ 9 10/ 11] *(219 / 220b)
# This file describes the network interfaces available on your system
# For more information, see netplan(5).
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: yes
    eth1:
      dhcp4: yes|
```

Рис. 8: Файл /etc/netplan/01-netcfg.yaml

В виртуальной машине mininet переименуем предыдущую установку Mininet. Скачаем новую версию Mininet. Обновим исполняемые файлы (рис. 9).

```

mininet@mininet-vm:~$ cd ~
mininet@mininet-vm:~$ git clone https://github.com/mininet/mininet.git
Cloning into 'mininet'...
remote: Enumerating objects: 10388, done.
remote: Counting objects: 100% (128/128), done.
remote: Compressing objects: 100% (59/59), done.
remote: Total 10388 (delta 102), reused 69 (delta 69), pack-reused 10260 (from 3)
Receiving objects: 100% (10388/10388), 3.36 MiB | 2.91 MiB/s, done.
Resolving deltas: 100% (6906/6906), done.
mininet@mininet-vm:~$ cd ~/mininet
mininet@mininet-vm:~/mininet$ sudo make install
cc -Wall -Wextra \
-DVERSION=\\"PYTHONPATH=. python -B bin/mn --version 2>&1\\" mnexec.c -o mnexec
install -D mnexec /usr/bin/mnexec
PYTHONPATH=. help2man -N -n "create a Mininet network." \
--no-discard-stderr "python -B bin/mn" -o mn.1
help2man -N -n "execution utility for Mininet." \
-h "-h" -v "-v" --no-discard-stderr ./mnexec -o mnexec.1
install -D -t /usr/share/man/man1 mn.1 mnexec.1
python -m pip uninstall -y mininet || true
Found existing installation: mininet 2.3.0
Uninstalling mininet-2.3.0:
  Successfully uninstalled mininet-2.3.0
python -m pip install .
Processing /home/mininet/mininet
Requirement already satisfied: setuptools in /usr/lib/python3/dist-packages (from mininet==2.3.1b4) (45.2.0)
Building wheels for collected packages: mininet
  Building wheel for mininet (setup.py) ... done
  Created wheel for mininet: filename=mininet-2.3.1b4-py3-none-any.whl size=160942 sha256=274a120e36ceb0b7e204b1a8bb2aa767f8728cb479499d89fc5f4b760ae70d23
  Stored in directory: /tmp/pip-ephem-wheel-cache-j887q0b/_wheels/cd/7d/a7/aafe1b3eaff31ef6ba4e2ea6c9690a717bdf739db6cfe8d45
Successfully built mininet
Installing collected packages: mininet
Successfully installed mininet-2.3.1b4
mininet@mininet-vm:~/mininet$ _

```

Рис. 9: Обновление Mininet

Проверим номер установленной версии mininet (рис. 10).

```

mininet@mininet-vm:~/mininet$ mn --version
2.3.1b4
mininet@mininet-vm:~/mininet$

```

Рис. 10: Номер установленной версии mininet

При попытке запуска приложения из-под суперпользователя возникает ошибка: X11 connection rejected because of wrong authentication. Ошибка возникает из-за того, что X-соединение выполняется от имени пользователя mininet, а приложение запускается от имени пользователя root с использованием sudo. Для исправления этой ситуации необходимо заполнить файл полномочий /root/.Xauthority, используя утилиту xauth. Скопируем значение куки (MIT magic cookie)<sup>1</sup> пользователя mininet в файл для пользователя root (рис. 11).

```

mininet@mininet-vm:~$ xauth list $DISPLAY
mininet-vm/unix:12 MIT-MAGIC-COOKIE-1 4cbb57874d9b6a08ee805a5258a8c2f8
mininet@mininet-vm:~$ sudo -i
root@mininet-vm:~# xauth list
mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 cfd80afdc298ef84233473ad48a3e0bd
root@mininet-vm:~#
root@mininet-vm:~# xauth add mininet-vm/unix:12 MIT-MAGIC-COOKIE-1 4cbb57874d9b6a08ee805a5258a8c2f8
root@mininet-vm:~# xauth list $DISPLAY
mininet-vm/unix:12 MIT-MAGIC-COOKIE-1 4cbb57874d9b6a08ee805a5258a8c2f8
root@mininet-vm:~# logout
mininet@mininet-vm:~$ sudo ~/mininet/mininet/examples/miniedit.py
topo=None

```

Рис. 11: Настройка соединения X11 для суперпользователя

Для запуска минимальной топологии введем в командной строке (рис. 12): `sudo mn`. Эта команда запускает Mininet с минимальной топологией, состоящей из коммутатора, подключённого к двум хостам. Для отображения списка команд интерфейса командной строки Mininet и примеров их использования введем команду в интерфейсе командной строки Mininet: `help` Для отображения доступных узлов введем: `nodes` Вывод этой команды показывает, что есть два хоста (хост h1 и хост h2) и коммутатор (s1). Иногда бывает полезно отобразить связи между устройствами в Mininet, чтобы понять топологию. Введем команду `net` в интерфейсе командной строки Mininet, чтобы просмотреть доступные линки: `net` Вывод этой команды показывает: - Хост h1 подключён через свой сетевой интерфейс h1-eth0 к коммутатору на интерфейсе s1-eth1. - Хост h2 подключён через свой сетевой интерфейс h2-eth0 к коммутатору на интерфейсе s1-eth2. - Коммутатор s1: - имеет петлевой интерфейс lo. - подключается к h1-eth0 через интерфейс s1-eth1. - подключается к h2-eth0 через интерфейс s1-eth2.

```

root@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

Documented commands (type help <topic>):
=====
EOF      gterm   iperfudp  nodes    pingpair  py       switch  xterm
dpctl    help    link      noecho   pingpairfull  quit    time
dump     intfs   links     pingall  ports     sh       wait
exit     iperf   net       pingallfull  px       source  x

You may also send a command to a node using:
<node> command {args}
For example:
mininet> h1 ifconfig

The interpreter automatically substitutes IP addresses
for node names when a node is the first arg, so commands
like
mininet> h2 ping h3
should work.

Some character-oriented interactive commands require
noecho:
mininet> noecho h2 vi foo.py
However, starting up an xterm/gterm is generally better:
mininet> xterm h2

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

```

Рис. 12: Работа с Mininet с помощью командной строки

Mininet позволяет выполнять команды на конкретном устройстве. Чтобы выполнить команду для определенного узла, необходимо сначала указать устройство, а затем команду, например: `h1 ifconfig`

Эта запись выполняет команду `ifconfig` на хосте `h1` и показывает интерфейсы хоста `h1` — хост `h1` имеет интерфейс `h1-eth0`, настроенный с IP-адресом `10.0.0.1`, и другой интерфейс `lo`, настроенный с IP-адресом `127.0.0.1` (рис. 13).

```

mininet> h1 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 ca:f4:8b:f1:5e:4d 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

```

Рис. 13: Работа с Mininet с помощью командной строки

Посмотрим конфигурацию всех узлов (рис. 14).

```
mininet> h2 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 8a:6f:fe:4d:39:79 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> s1 ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.56.113 netmask 255.255.255.0 broadcast 192.168.56.255
        ether 08:00:27:63:d8:41 txqueuelen 1000 (Ethernet)
        RX packets 1933 bytes 191867 (191.8 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 988 bytes 136591 (136.5 KB)
        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:b1:0e:5d txqueuelen 1000 (Ethernet)
        RX packets 4934 bytes 5917564 (5.9 MB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 2863 bytes 184542 (184.5 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 1983 bytes 150926 (150.9 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 1983 bytes 150926 (150.9 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

s1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        ether 3a:85:4e:26:60:6b 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
```

Рис. 14: Работа с Mininet с помощью командной строки

По умолчанию узлам h1 и h2 назначаются IP-адреса 10.0.0.1/8 и 10.0.0.2/8 соответственно. Чтобы проверить связь между ними, используем команду `ping`. Команда `ping` работает, отправляя сообщения эхо-запроса протокола управляющих сообщений Интернета (ICMP) на удалённый компьютер и ожидая ответа. Например, команда `h1 ping 10.0.0.2` проверяет соединение между хостами h1 и h2 (рис. 15).

```

mininet> h1 ping 10.0.0.2
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=2.39 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.277 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.080 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.077 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.080 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.086 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.069 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.068 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.074 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.122 ms
--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9175ms
rtt min/avg/max/mdev = 0.068/0.332/2.394/0.689 ms
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 281.211 seconds

```

Рис. 15: Проверка связности хостов

Очистим предыдущий экземпляр Mininet (рис. 16).

```

root@mininet-vm:~# sudo mn -c
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-controller ovs-testcontroller udprawtest mnexec ixs ryu-manager 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-controller ovs-testcontroller udprawtest mnexec ixs ryu-manager 2> /dev/null
killall -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old x11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
*** Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([_[:alnum:]]+-eth[[:digit:]]+)'
ip link show
*** Killing stale mininet node processes
killall -9 -f mininet:
*** Shutting down stale tunnels
killall -9 -f Tunnel=Ethernet
killall -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.

```

Рис. 16: Очистка предыдущего экземпляра Mininet

По умолчанию XTerm использует растровые шрифты малого кегля. Для увеличения размера шрифта и применения векторных шрифтов вместо растровых необходимо внести изменения в файл `/etc/X11/app-defaults/XTerm`. Для этого можно воспользоваться следующей командой: `sudo mcedit /etc/X11/app-defaults/XTerm` и затем в конце файла добавить строки: `xtermfaceName: Monospace`, `xtermfaceSize: 12`. Здесь выбран системный моноширинный шрифт, кегль шрифта — 12 пунктов (рис. 17).

```
/etc/X11/app-defaults/XTerm  [M~] 15 L:[232*34 266/266] *(10373/10376b) 32 0x020 [~]IXI
! Set a border for the menus to make them simpler to distinguish against the
! vt100 widget:
*SimpleMenu=borderWidth: 2

! xterm can switch at runtime between bitmap (default) and TrueType fonts.
! The "faceSize" resource controls the size of the latter. However, it was
! originally given with a size that makes the two types of fonts different
! sizes. Uncomment this line to use the same size as "fixed".
!*faceSize: 8

! Here is a pattern that is useful for double-clicking on a URL:
*charClass: 33:48,35:48,37-38:48,43-47:48,58:48,61:48,63-64:48,95:48,126:48
!
! Alternatively,
!*on2Clicks: regex [[[:alpha:]]+://([[:alnum:]]#+,./=?@_~)~]([[:xdigit:]]|[:xdigit:]]|)+

! VT100s and similar terminals recognize escape sequences and control
! characters to which they reply to the host with other escape sequences,
! to provide information. The "resize" program uses this feature.
!
! In addition, xterm recognizes several escape sequences which can be used to
! set fonts, window properties, return settings via escape sequences. Some
! find these useful; others are concerned with the possibility of unexpected
! inputs.
!
! All of these features can be enabled or disabled via menus.
!
! Depending on your environment, you may wish to disable those by default by
! uncommenting one or more of the resource settings below:
!*allowFontOps: false
!*allowTcapOps: false
!*allowTitleOps: false
!*allowWindowOps: false
xterm*faceName: Monospace
xterm*faceSize: 12
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit
```

Рис. 17: Настройка параметров XTerm

Запустим Xserver. Выберем опции: multiple windows, display number: -1, start no client. Сохраним параметры, тогда при следующем запуске не нужно будет отмечать эти опции (рис. 18).

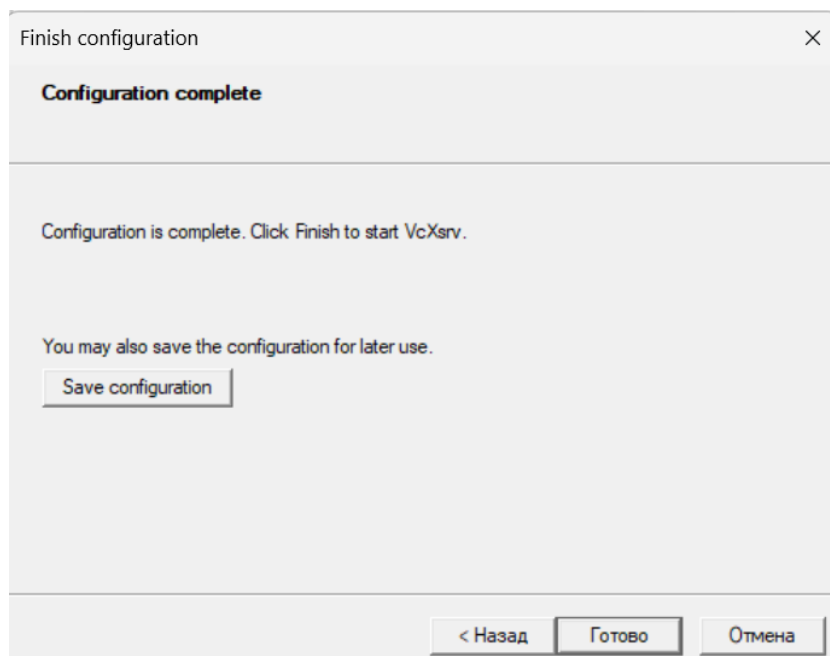


Рис. 18: Запуск и настройка Xserver



Запустим putty. При подключении добавим опцию перенаправления X11 (рис. 19;20).

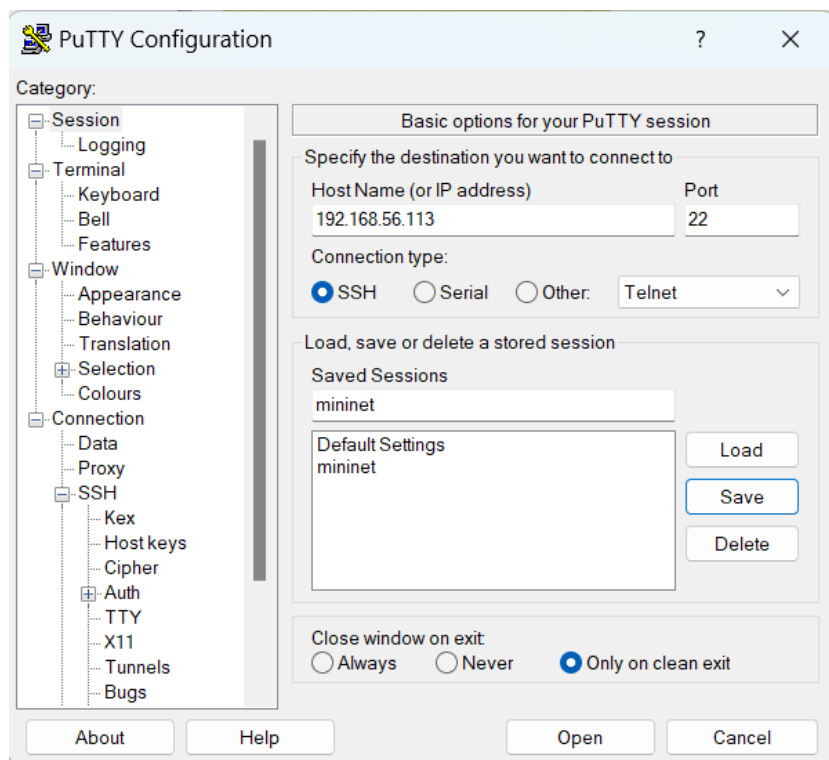


Рис. 19: Запуск putty и добавление опции перенаправления X11

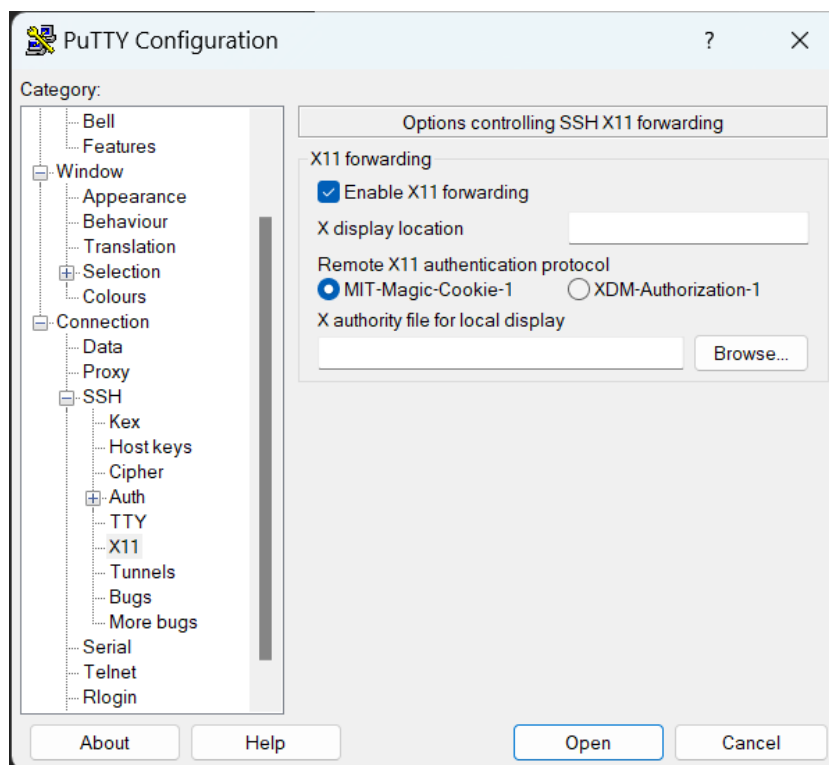


Рис. 20: Запуск putty и добавление опции перенаправления X11

В терминале виртуальной машины mininet запустим MiniEdit: `sudo ~/mininet/mininet/examples/miniedit.py`.

Добавим два хоста и один коммутатор, соединим хосты с коммутатором (рис. 21). Настроим IP-адреса на хостах h1 и h2. Для этого удерживая правую кнопку мыши на устройстве выберем свойства. Для хоста h1 укажем IP-адрес 10.0.0.1/8, а для хоста h2— 10.0.0.2/8(рис. 22).

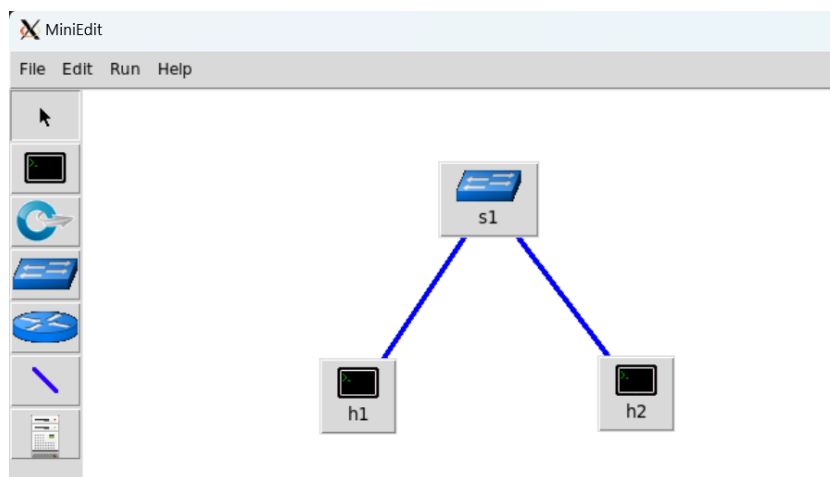


Рис. 21: Добавление двух хостов и одного коммутатора

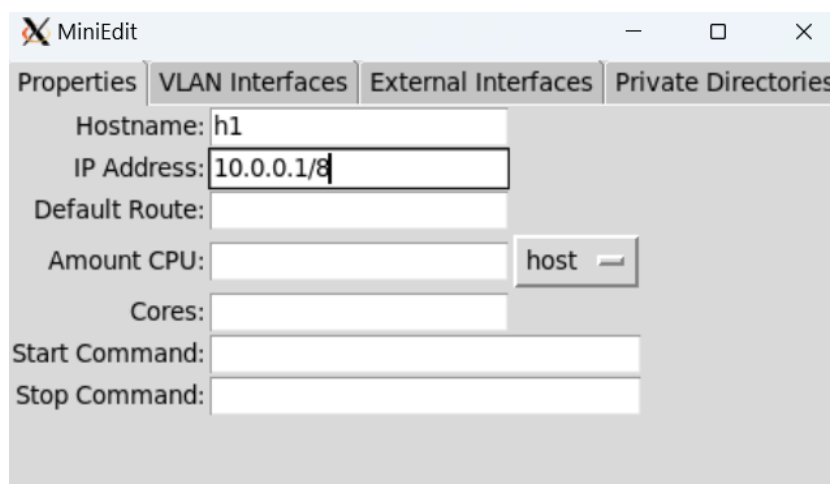
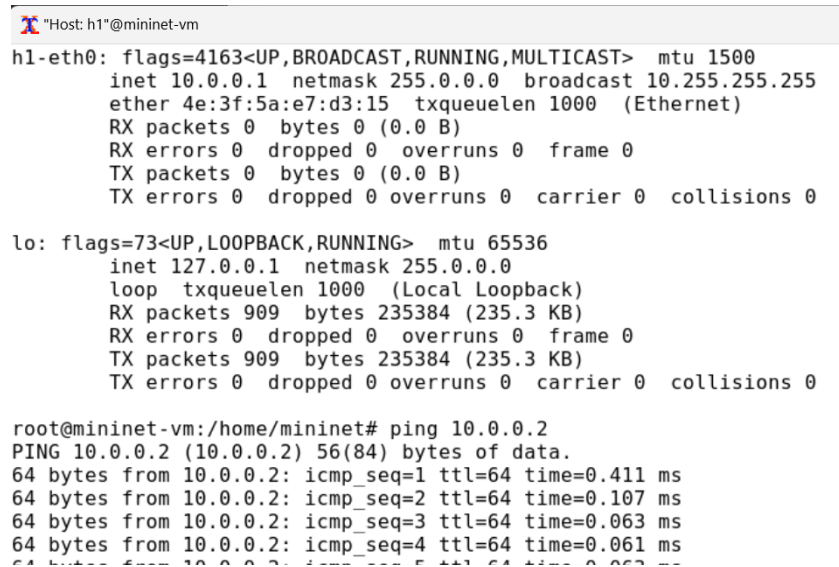


Рис. 22: Настройка IP-адреса на хостах

Перед проверкой соединения между хостом h1 и хостом h2 необходимо запустить эмуляцию. Для запуска эмуляции нажмём кнопку Run. После начала эмуляции кнопки панели MiniEdit стали серыми, указывая на то, что в настоящее время они отключены. Откроем терминал на хосте h2. На терминале хоста h1 введём команду `ifconfig`, чтобы отобразить назначенные ему IP-адреса. Интерфейс h1-eth0 на хосте h1 настроен с IP-адресом 10.0.0.1 и маской подсети 255.0.0.0. Повторим эти действия на хосте h2. Его интерфейс h2-eth0 настроен с IP-адресом 10.0.0.2 и маской подсети 255.0.0.0. Проверим соединение между хо-

стами, введя в терминале хоста h2 команду `ping 10.0.0.1`. Для остановки теста нажмём `Ctrl + c`. Остановим эмуляцию, нажав кнопку Stop (рис. 23).



```
"Host: h1"@mininet-vm
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 4e:3f:5a:e7:d3:15 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 909 bytes 235384 (235.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 909 bytes 235384 (235.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@mininet-vm:/home/mininet# ping 10.0.0.2
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=0.411 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.107 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.063 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.061 ms
^C
```

Рис. 23: Проверка назначенных IP-адресов для h2 и проверка соединения между хостами

Ранее IP-адреса узлам h1 и h2 были назначены вручную. В качестве альтернативы можно полагаться на Mininet для автоматического назначения IP-адресов. Для этого удалим назначенный вручную IP-адрес с хостов h1 и h2. В MiniEdit нажмём Edit Preferences. По умолчанию в поле базовые значения IP-адресов (IP Base) установлено 10.0.0.0/8. Изменим это значение на 15.0.0.0/8. Затем запустим эмуляцию, нажав кнопку Run (рис. 24).

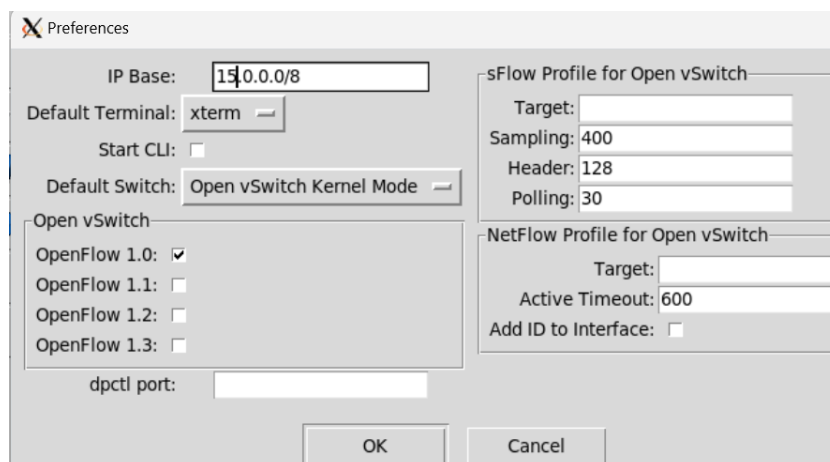


Рис. 24: Проверка автоматического назначения адресов

Откроем терминал на хосте h1, удерживая правую кнопку мыши на хосте h1 и выбрав Terminal. Отобразим IP-адреса, назначенные хосту h1. Интерфейс h1-eth0 на узле h1 теперь имеет IP-адрес 15.0.0.1 и маску подсети 255.0.0.0 (рис. 25)

```

Host: h1@mininet-vm
root@mininet-vm:/home/mininet# ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 15.0.0.1 netmask 255.0.0.0 broadcast 15.255.255.255
    ether 76:19:85:ea:ed:2b 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 807 bytes 226356 (226.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 807 bytes 226356 (226.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@mininet-vm:/home/mininet#

```

Рис. 25: Отображение IP-адреса, назначенного хосту h1

В домашнем каталоге виртуальной машины mininet создадим каталог для работы с проектами mininet: `mkdir ~/work`. Для сохранения топологии сети в файл нажмём в MiniEdit “File”-“Save”. Укажем имя для топологии и сохраним на своём компьютере (рис. 26).

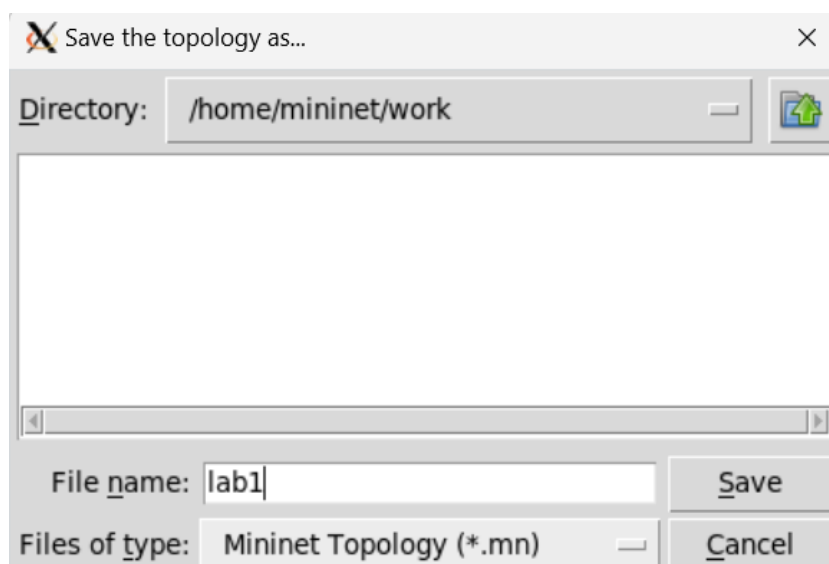


Рис. 26: Сохранение топологии

После сохранения проекта поменяем права доступа к файлам в каталоге проекта (рис. 27).

```
mininet@mininet-vm:~$ ls -Al ~/work
total 4
-rw-r--r-- 1 root root 1655 Sep  7 05:32 lab1.mn
mininet@mininet-vm:~$ sudo chown -R mininet ~/work/
mininet@mininet-vm:~$
```

Рис. 27: Изменение прав доступа к файлам в каталоге проекта

## Выводы

В результате выполнения данной лабораторной работы я развёрнула mininet в системе виртуализации VirtualBox, а также ознакомилась с основными командами для работы с Mininet через командную строку и через графический интерфейс.

## Список литературы

1. Mininet [Электронный ресурс]. Mininet Project Contributors. URL: <http://mininet.org/> (дата обращения: 07.10.2025).