

# Отчёт по лабораторной работе №1

## Моделирование сетей передачи данных

### Введение в Mininet

Выполнил: Исаев Булат Абубакарович,  
НПИбд-01-22, 1132227131

### Содержание

1	Цель работы .....	1
2	Выполнение лабораторной работы.....	1
2.1	Настройка образа VirtualBox.....	1
2.2	Подключение к виртуальной машине.....	2
2.3	Работа с Mininet из-под Windows.....	4
2.4	Настройка параметров XTerm.....	7
2.5	Настройка соединения X11 для суперпользователя.....	8
2.6	Работа с Mininet с помощью командной строки .....	8
2.7	Построение и эмуляция сети в Mininet с использованием графического интерфейса .....	12
3	Вывод.....	17
4	Список литературы. Библиография .....	17

## 1 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.

## 2 Выполнение лабораторной работы

### 2.1 Настройка образа VirtualBox

Для начала перейдём в репозиторий Mininet и скачаем актуальный релиз ovf-образа виртуальной машины. После чего запустим систему виртуализации и импортируем файл .ovf. Перейдём в настройки системы виртуализации и уточним параметры настройки виртуальной машины. В частности, для VirtualBox выберем импортированную виртуальную машину и перейдите в меню “Машина”-“Настроить”. Перейдём к опции «Система». Внизу этого окна есть сообщение об обнаружении

неправильных настроек, следуя рекомендациям, внесём исправления. В настройках сети первый адаптер должен иметь тип подключения host-only network adapter (виртуальный адаптер хоста), который в дальнейшем мы будем использовать для входа в образ виртуальной машины. В этом режиме адаптер хоста использует специальное устройство vboxnet0, создает подсеть и назначает IP-адрес сетевой карте гостевой операционной системы. Запустим виртуальную машину с Mininet (рис. 1):

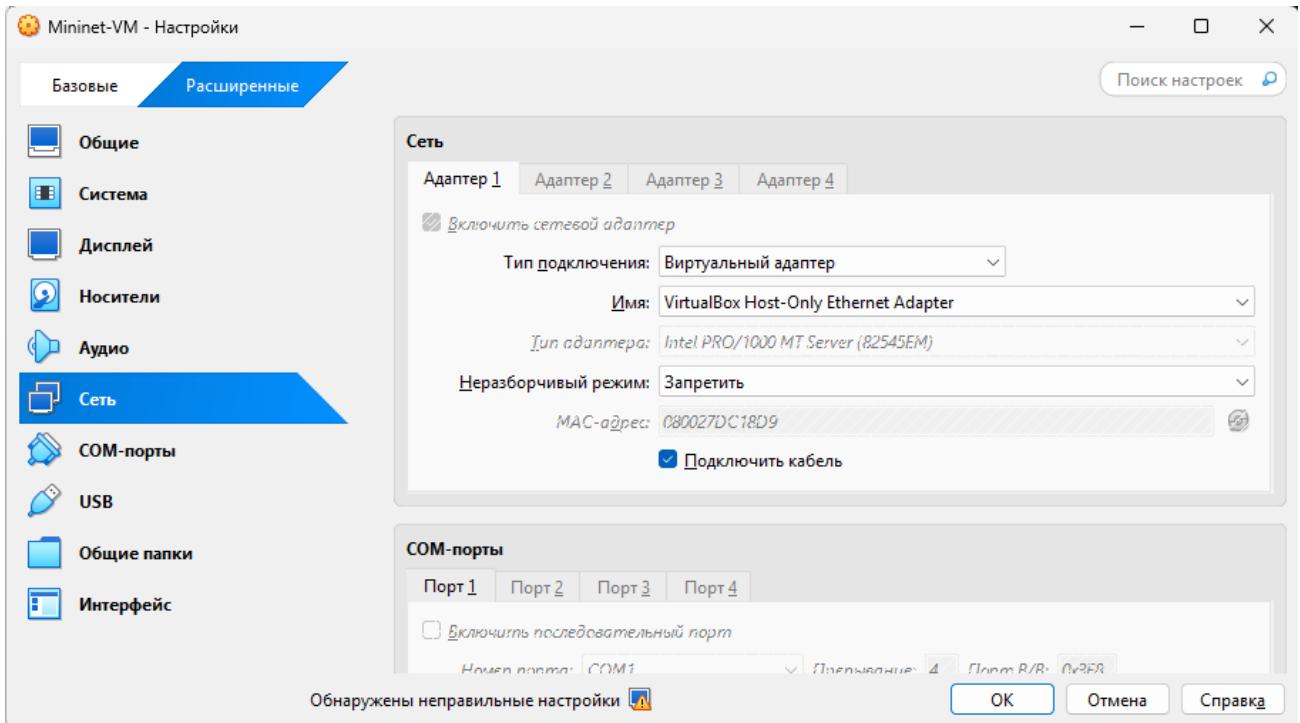


Рис. 1: Установка и настройка виртуальной машины

## 2.2 Подключение к виртуальной машине

Залогинемся в виртуальной машине и посмотрим её адрес (рис. 2):

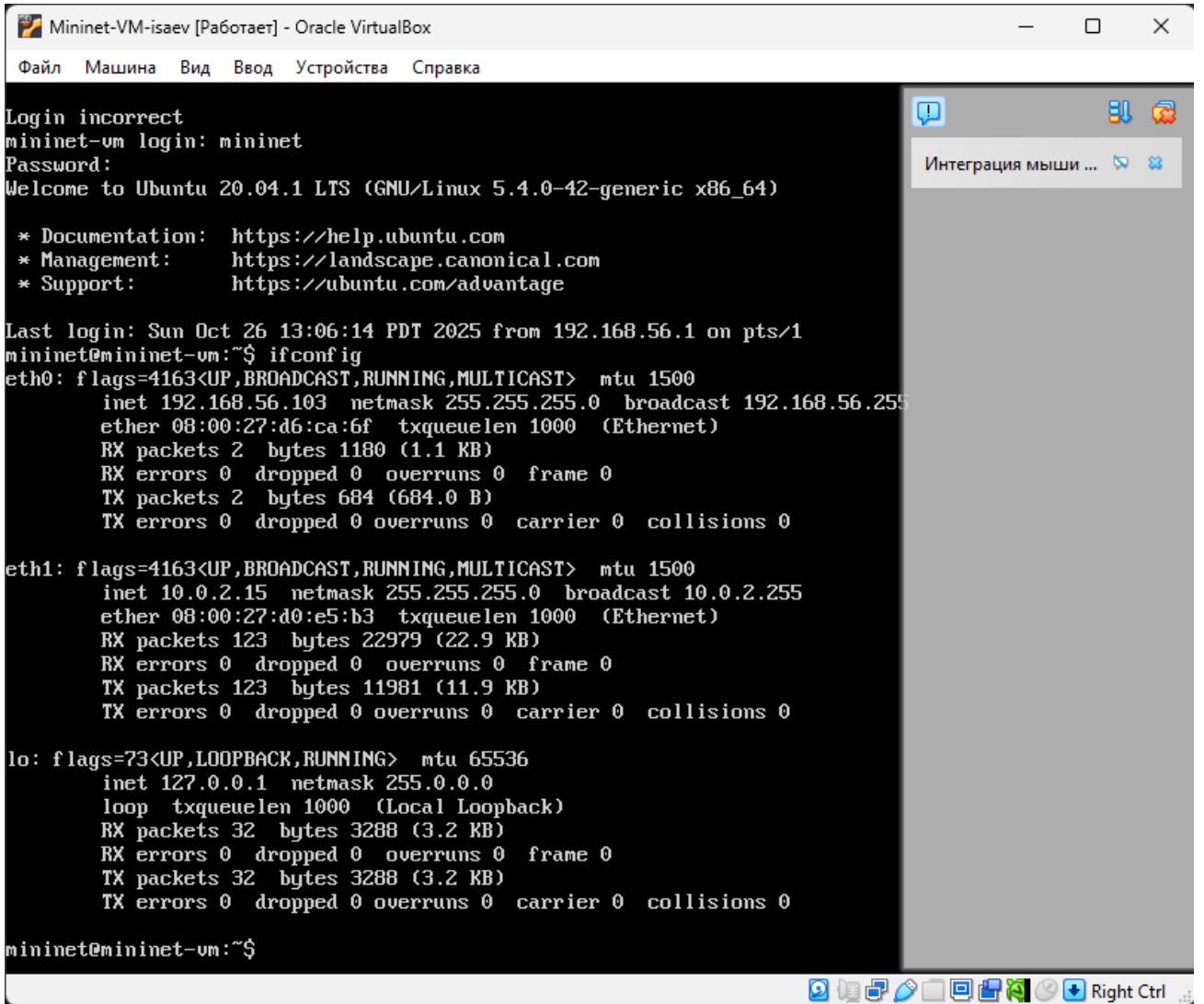
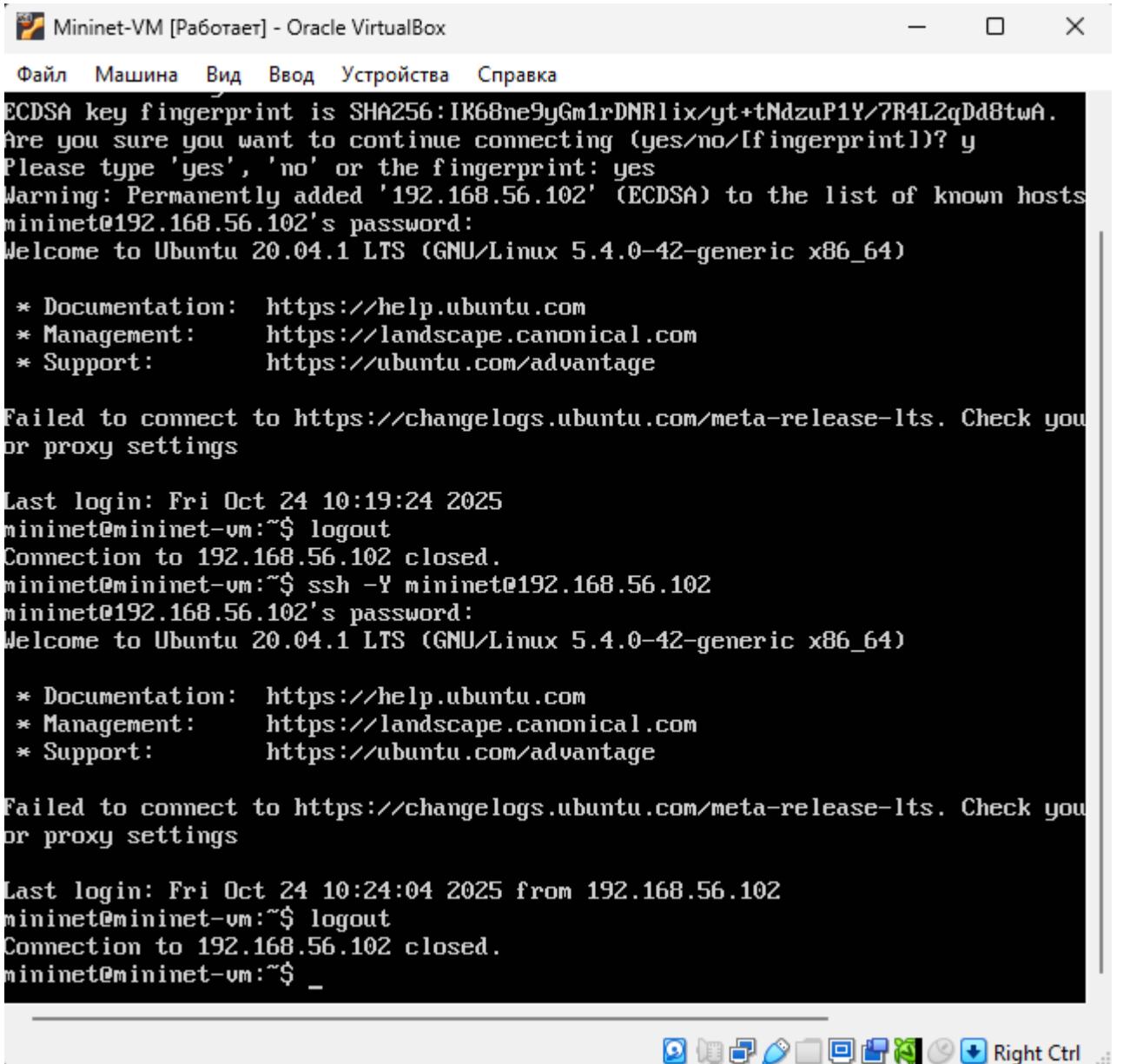


Рис. 2: Вход и просмотр адреса виртуальной машины

Внутренний адрес машины 192.168.56.102, подключимся к виртуальной машине (из терминала хостовой машины). Для отключения ssh-соединения с виртуальной машиной нажмём Ctrl + d (рис. 3):



```
Mininet-VM [Работает] - Oracle VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
ECDSA key fingerprint is SHA256:IK68ne9yGm1rDNR1ix/yt+tNdzuP1Y/7R4L2qDd8twA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '192.168.56.102' (ECDSA) to the list of known hosts
mininet@192.168.56.102's password:
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
or proxy settings

Last login: Fri Oct 24 10:19:24 2025
mininet@mininet-vm:~$ logout
Connection to 192.168.56.102 closed.
mininet@mininet-vm:~$ ssh -Y mininet@192.168.56.102
mininet@192.168.56.102's password:
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
or proxy settings

Last login: Fri Oct 24 10:24:04 2025 from 192.168.56.102
mininet@mininet-vm:~$ logout
Connection to 192.168.56.102 closed.
mininet@mininet-vm:~$
```

Рис. 3: Подключение к виртуальной машине из терминала хостовой машины

### 2.3 Работа с Mininet из-под Windows

Установим putty (рис. 4) и VcXsrv Windows X Server (рис. 5):

```
Administrator: Windows PowerShell
Windows PowerShell
(С) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Установите последнюю версию PowerShell для новых функций и улучшения! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> choco install putty -y
Chocolatey v2.4.1
3 validations performed. 2 success(es), 1 warning(s), and 0 error(s).

Validation Warnings:
- A pending system reboot request has been detected, however, this is
  being ignored due to the current Chocolatey configuration. If you
  want to halt when this occurs, then either set the global feature
  using:
    choco feature enable --name="exitOnRebootDetected"
  or pass the option --exit-when-reboot-detected.

Installing the following packages:
putty
By installing, you accept licenses for the packages.
Downloading package from source 'https://community.chocolatey.org/api/v2/'
Progress: Downloading putty.portable 0.83.0... 100%

putty.portable v0.83.0 [Approved]
putty.portable package files install completed. Performing other installation steps.
Extracting 64-bit C:\ProgramData\chocolatey\lib\putty.portable\tools\putty_x64.zip to C:\ProgramData\chocolatey\lib\putty.portable\tools...
C:\ProgramData\chocolatey\lib\putty.portable\tools
ShimGen has successfully created a gui shim for PAGEANT.EXE
ShimGen has successfully created a shim for PLINK.EXE
ShimGen has successfully created a shim for PSCP.EXE
ShimGen has successfully created a shim for PSFTP.EXE
ShimGen has successfully created a gui shim for PUTTY.EXE
ShimGen has successfully created a gui shim for PUTTYGEN.EXE
The install of putty.portable was successful.
  Deployed to 'C:\ProgramData\chocolatey\lib\putty.portable\tools'
Downloading package from source 'https://community.chocolatey.org/api/v2/'
Progress: Downloading putty 0.83.0... 100%

putty v0.83.0 [Approved]
putty package files install completed. Performing other installation steps.
The install of putty was successful.
  Deployed to 'C:\ProgramData\chocolatey\lib\putty'

Chocolatey installed 2/2 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
PS C:\WINDOWS\system32>
```

Рис. 4: Установка putty

```
Administrator: Windows PowerShell
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
PS C:\WINDOWS\system32> choco install vcxsrv -y
Chocolatey v2.4.1
3 validations performed. 2 success(es), 1 warning(s), and 0 error(s).

Validation Warnings:
- A pending system reboot request has been detected, however, this is
being ignored due to the current Chocolatey configuration. If you
want to halt when this occurs, then either set the global feature
using:
  choco feature enable --name="exitOnRebootDetected"
or pass the option --exit-when-reboot-detected.

Installing the following packages:
vcxsrv
By installing, you accept licenses for the packages.
Downloading package from source 'https://community.chocolatey.org/api/v2/'
Progress: Downloading vcxsrv 21.1.10... 100%
vcxsrv v21.1.10 [Approved]
vcxsrv package files install completed. Performing other installation steps.
Installing 64-bit vcxsrv...
vcxsrv has been installed.
  vcxsrv may be able to be automatically uninstalled.
The install of vcxsrv was successful.
  Software installed as 'EXE', install location is likely default.

Chocolatey installed 1/1 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
PS C:\WINDOWS\system32>
```

Рис. 5: Установка putty VcXsrv Windows X Server

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. 6):

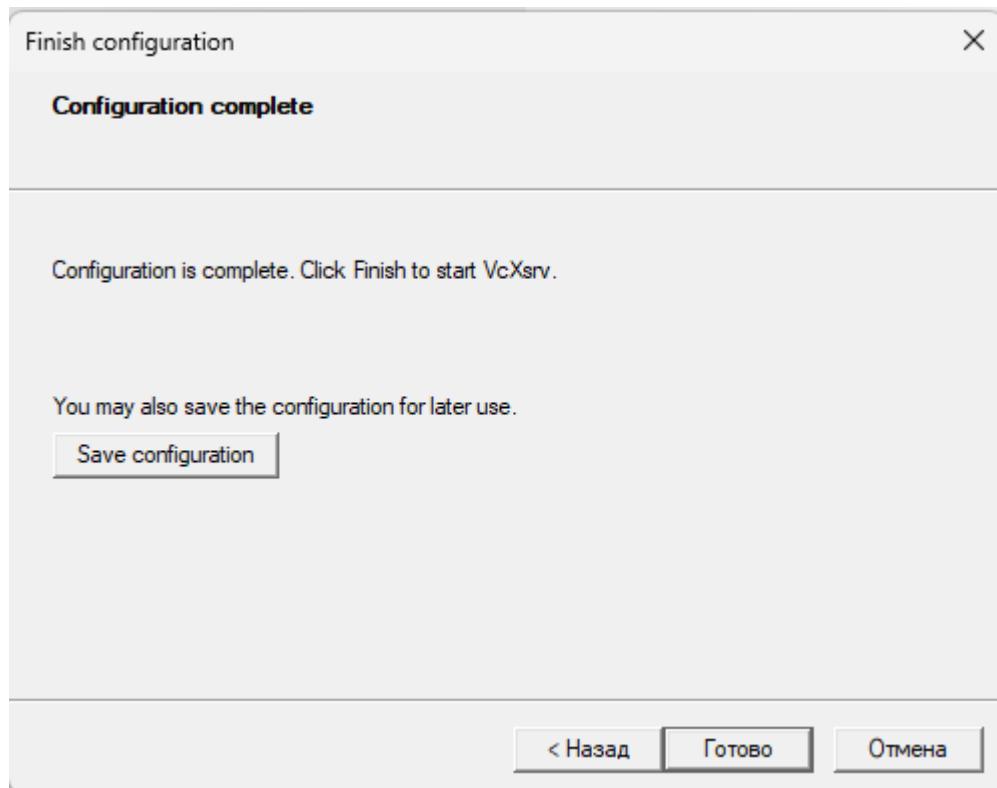


Рис. 6: Запуск и настройка Xserver

Запустим putty. При подключении добавим опцию перенаправления X11 (рис. 7):

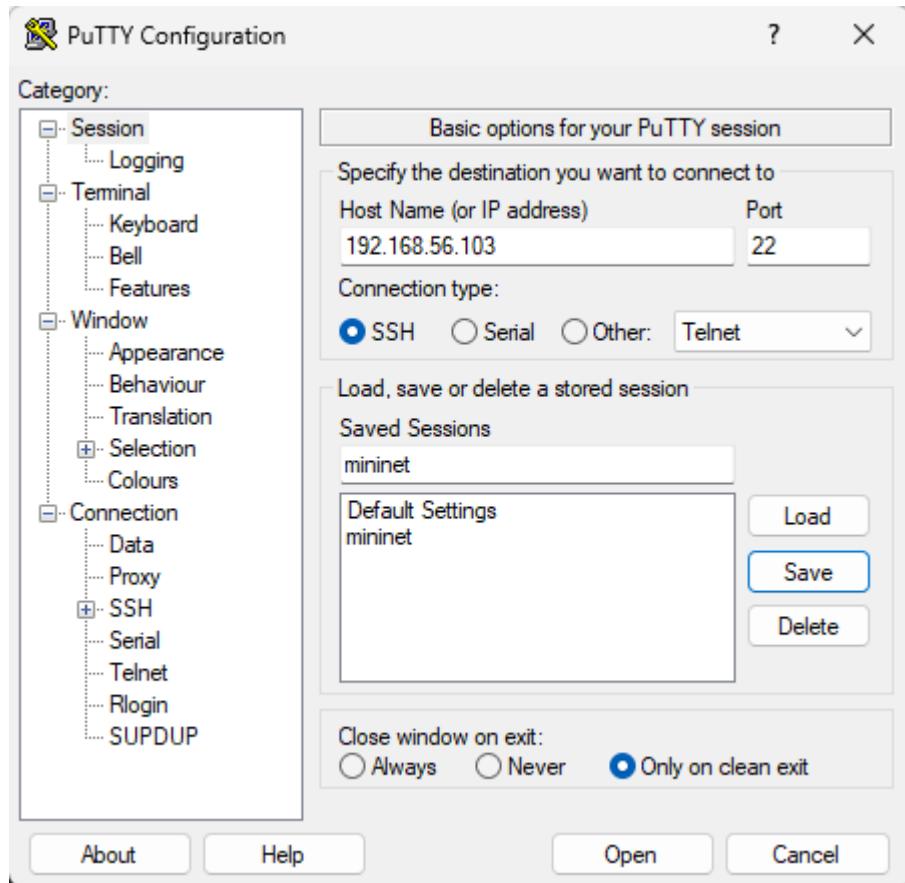


Рис. 7: Запуск putty и добавление опции перенаправления X11

## 2.4 Настройка параметров XTerm

По умолчанию XTerm использует растревые шрифты малого кегля. Для увеличения размера шрифта и применения векторных шрифтов вместо растревых необходимо внести изменения в файл /etc/X11/app-defaults/XTerm и в конце файла добавить нужные строки. Перед этим установим текстовый редактор mcedit (рис. 8):

```
mininet@mininet-vm: ~
/etc/X11/app-defaults/XTerm  [---] 18 L:[239+27 266/266] *(10376/10376b) <EOF>
! 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
!*allowCapOps: false
!*allowTitleOps: false
!*allowWindowOps: false
xterm*faceName: Monospace
xterm*faceSize: 12
```

Рис. 8: Увеличение размера шрифта и применение векторного шрифта

## 2.5 Настройка соединения X11 для суперпользователя

При попытке запуска приложения из-под суперпользователя возникает ошибка. 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)1 пользователя mininet в файл для пользователя root (рис. 9):

```
mininet@mininet-vm:~$ xauth list $DISPLAY
mininet-vm/unix:10  MIT-MAGIC-COOKIE-1  69fa6826576af937fcefe92dc91d92ad
mininet@mininet-vm:~$ sudo -i
root@mininet-vm:~# xauth list
mininet-vm/unix:10  MIT-MAGIC-COOKIE-1  69fa6826576af937fcefe92dc91d92ad
root@mininet-vm:~#
```

Рис. 9: Заполнение файла полномочий /root/.Xauthority

## 2.6 Работа с Mininet с помощью командной строки

Запустим минимальную топологию, состоящую из коммутатора, подключённого к двум хостам (рис. 10):

```
root@mininet-vm: ~
mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 69fa6826576af937fcfe92dc91d92ad
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> -
```

Рис. 10: Вызов Mininet с использованием топологии по умолчанию

Для отображения списка команд интерфейса командной строки Mininet и примеров их использования введём команду: help (рис. 11):

```
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> -
```

Рис. 11: Отображение списка команд и примеров их использования

Для отображения доступных узлов введём: nodes. Вывод этой команды показывает, что есть два хоста (хост h1 и хост h2) и коммутатор (s1) (рис. 12):

```
mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet>
```

Рис. 12: Отображение доступных узлов

Иногда бывает полезно отобразить связи между устройствами в Mininet, чтобы понять топологию. Введём команду net в интерфейсе командной строки Mininet, чтобы просмотреть доступные линки (рис. 13):

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

Рис. 13: Просмотр доступных линков

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.

Mininet позволяет выполнять команды на конкретном устройстве. Чтобы выполнить команду для определенного узла, необходимо сначала указать устройство, а затем команду (рис. 14):

```
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:29:e3:72:ab:3b  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>
```

Рис. 14: Выполнение команды для устройства h1

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.

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, мы можем использовать команду ping (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=0.329 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.128 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.121 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.102 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.061 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.082 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.093 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.403 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.098 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.088 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.229 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.305 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.086 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.073 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.128 ms
^C
--- 10.0.0.2 ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 14319ms
rtt min/avg/max/mdev = 0.061/0.155/0.403/0.104 ms
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 185.246 seconds
root@mininet-vm:~#
```

Рис. 15: Проверка связи между узлами h1 и h2

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 udpbw
test mnexec ivs ryu-manager 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-controller ovs-testcontroller ud
pbwtest mnexec ivs ryu-manager 2> /dev/null
pkill -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
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
root@mininet-vm:~#
```

Рис. 16: Очистка предыдущего экземпляра Mininet

## 2.7 Построение и эмуляция сети в Mininet с использованием графического интерфейса

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

Добавим два хоста и один коммутатор, соединим хосты с коммутатором (рис. 17). 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 (рис. 18), а для хоста h2 — 10.0.0.2/8 (рис. 19):

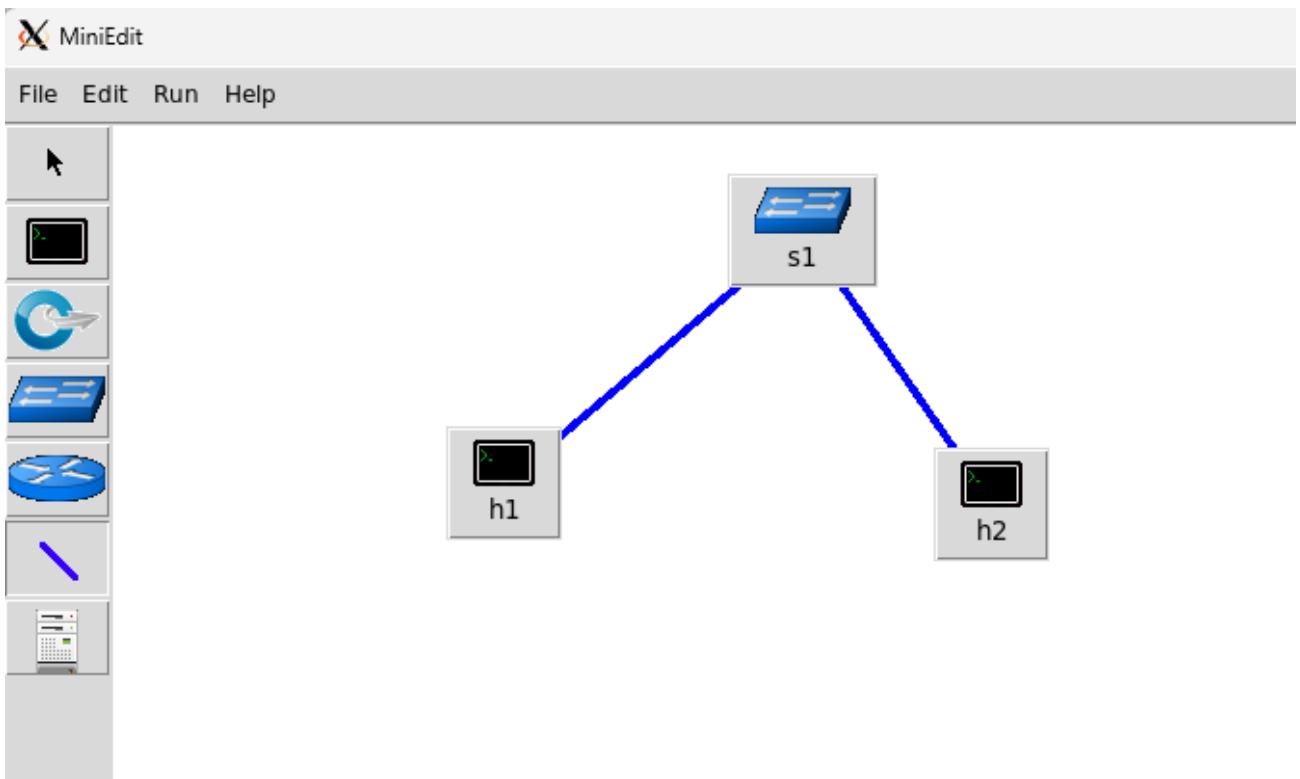


Рис. 17: Добавление двух хостов и одного коммутатора

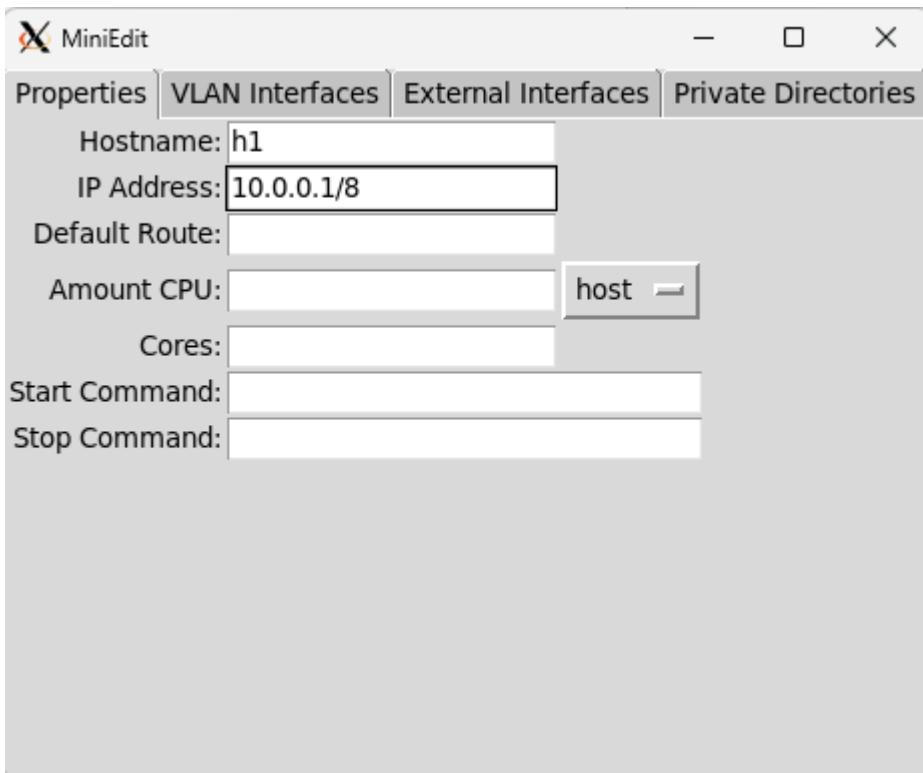


Рис. 18: Настройка IP-адреса на хосте h1

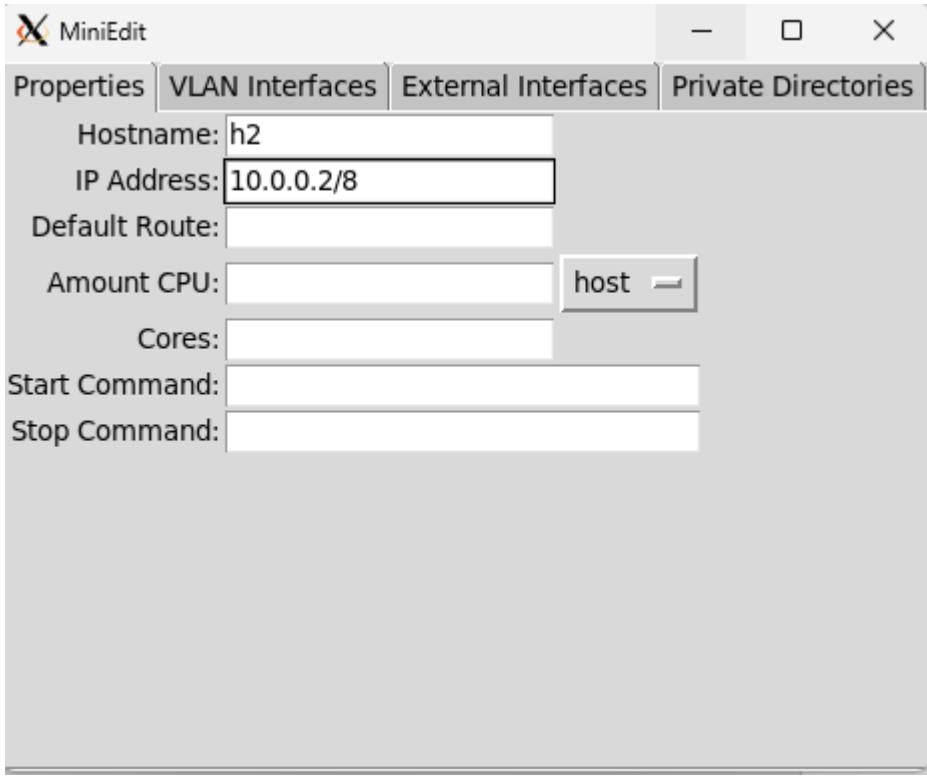
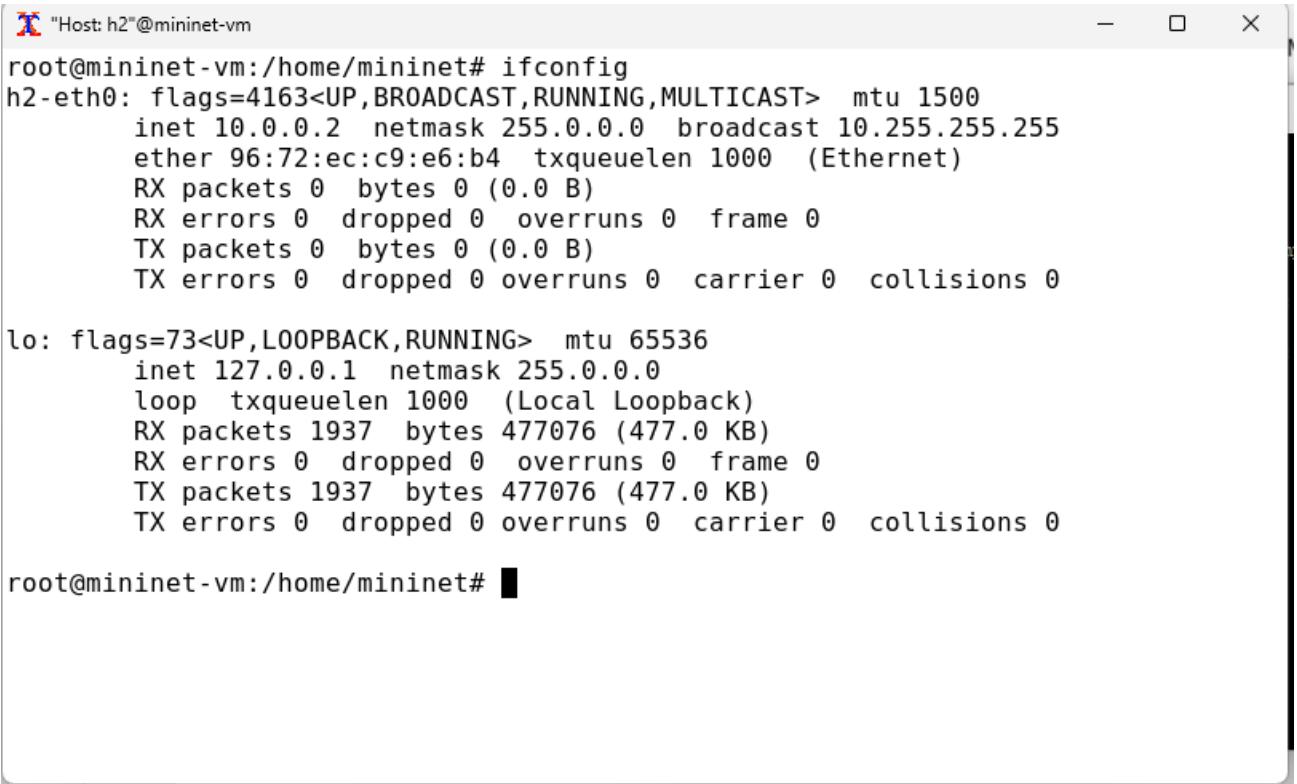


Рис. 19: Настройка IP-адреса на хосте h2

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 хостами, введя в терминале хоста h2 команду ping 10.0.0.1. Для остановки теста нажмём Ctrl + c. Остановим эмуляцию, нажав кнопку Stop (рис. 20):



```
"Host: h2" @mininet-vm
root@mininet-vm:/home/mininet# 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 96:72:ec:c9:e6:b4 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 1937 bytes 477076 (477.0 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 1937 bytes 477076 (477.0 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@mininet-vm:/home/mininet#
```

Рис. 20: 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. 21):

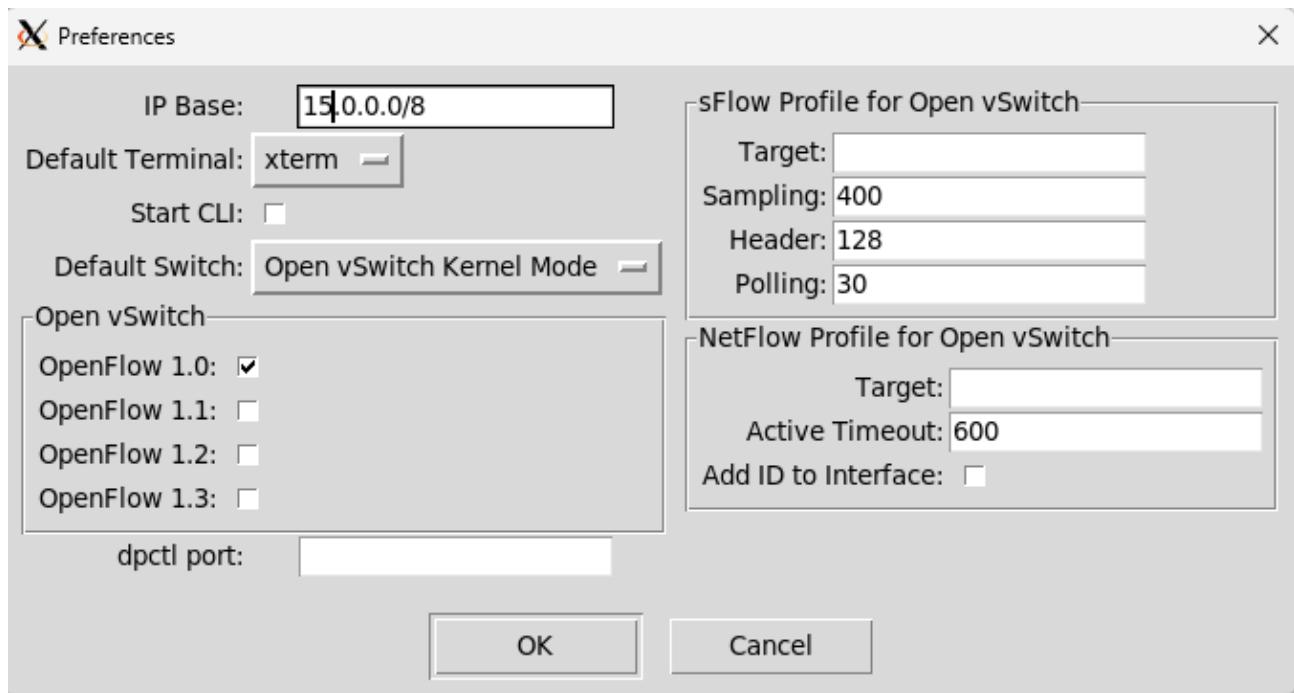


Рис. 21: Проверка автоматического назначения адресов

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. 22):

```
"Host: h1"@mininet-vm
root@mininet-vm:/home/mininet# 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 82:b5:d4:9f:08:c9 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 3023 bytes 424900 (424.9 KB)
                    RX errors 0 dropped 0 overruns 0 frame 0
                    TX packets 3023 bytes 424900 (424.9 KB)
                    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@mininet-vm:/home/mininet#"
```

Рис. 22: Отображение IP-адреса, назначенного хосту h1

В домашнем каталоге виртуальной машины mininet создадим каталог для работы с проектами mininet (рис. 23):

```
mininet@mininet-vm:~$ mkdir ~work
mkdir: cannot create directory '/home/mininet/work': File exists
mininet@mininet-vm:~$ ls
mininet mininet.orig oflops oftest openflow pox work
mininet@mininet-vm:~$
```

Рис. 23: Создание нового каталога

Для сохранения топологии сети в файл нажмём в MiniEdit “File”-“Save”. Укажем имя для топологии и сохраним на своём компьютере (рис. 24):

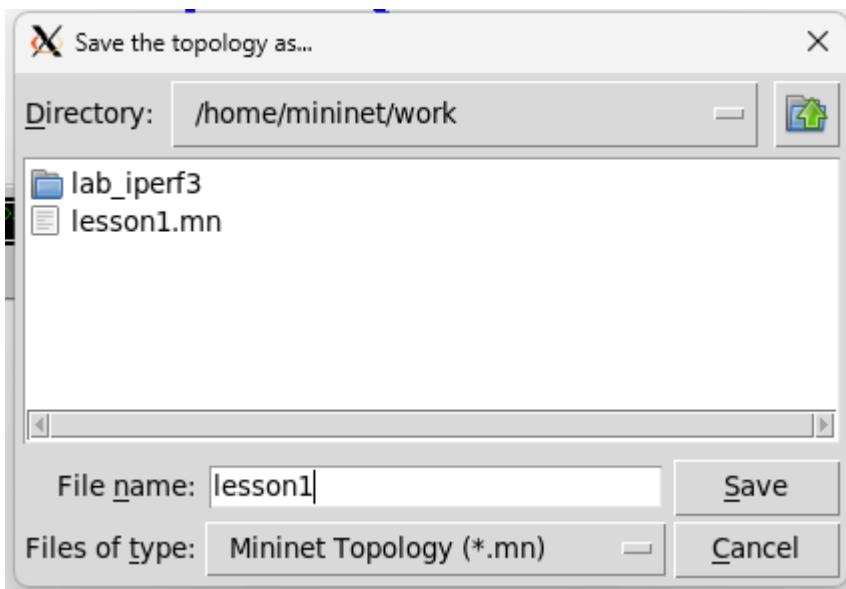


Рис. 24: Сохранение топологии

После сохранения проекта поменяем права доступа к файлам в каталоге проекта (рис. 25):

```
mininet@mininet-vm:~/work$ ls
lab_iperf3  lesson1.mn
mininet@mininet-vm:~/work$
```

Рис. 25: Изменение прав доступа к файлам в каталоге проекта

### 3 Вывод

В ходе выполнения лабораторной работы были получены навыки по развертыванию в системе виртуализации (например, в VirtualBox) mininet, а также познакомились с основными командами для работы с Mininet через командную строку и через графический интерфейс.

### 4 Список литературы. Библиография

[1] Mininet: <https://mininet.org/>