

Лабораторная работа №1

Введение в Mininet

Беличева Дарья Михайловна

Содержание

1	Цель работы	4
2	Теоретическое введение	5
3	Выполнение лабораторной работы	6
4	Выводы	17
	Список литературы	18

Список иллюстраций

3.1	Импорт конфигураций	6
3.2	Параметры импорта	7
3.3	Настройка сети	8
3.4	Настройка сети	8
3.5	Запуск mininet	9
3.6	Подключение к mininet через SSH	9
3.7	Просмотр IP-адресов машины	10
3.8	Файл /etc/netplan/01-netcfg.yaml	11
3.9	Обновление Mininet	11
3.10	Номер установленной версии mininet	11
3.11	Настройка соединения X11 для суперпользователя	12
3.12	Работа с Mininet с помощью командной строки	13
3.13	Работа с Mininet с помощью командной строки	14
3.14	Работа с Mininet с помощью командной строки	15
3.15	Проверка связности хостов	16

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 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(ВМ). 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`.

3 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. 3.1;3.2).

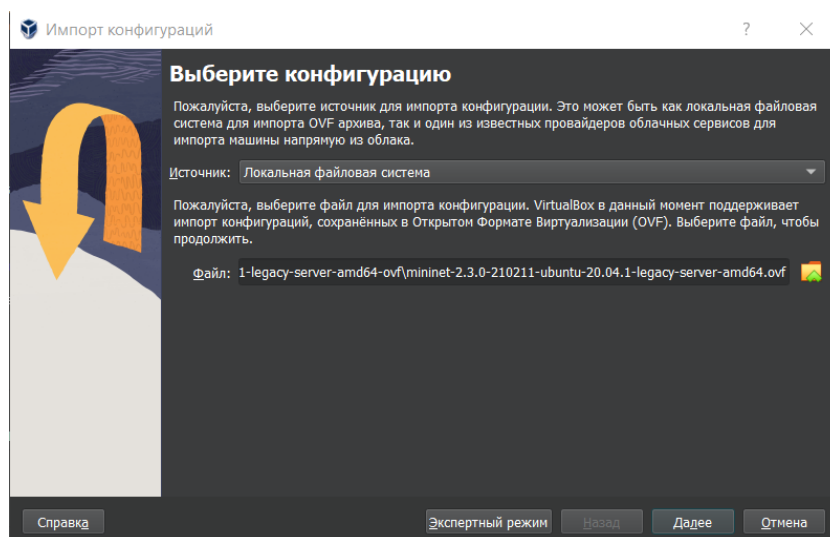


Рис. 3.1: Импорт конфигураций

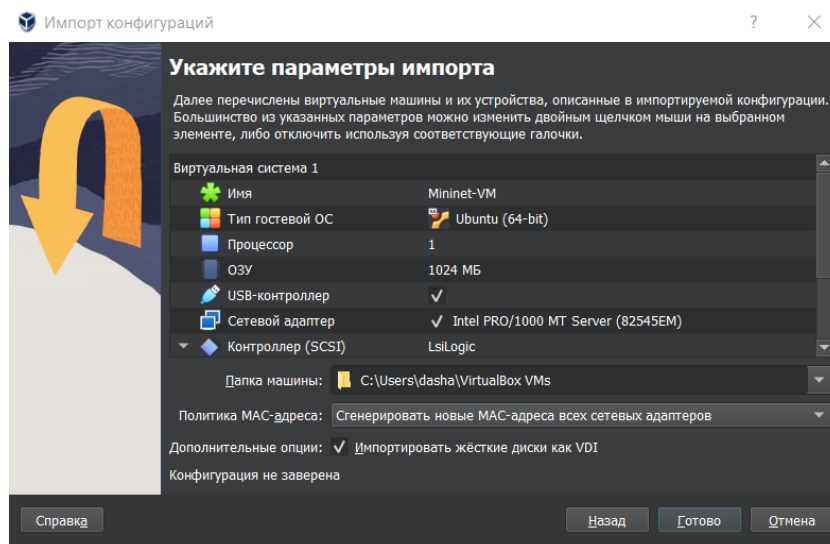


Рис. 3.2: Параметры импорта

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. 3.3). 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. 3.4).

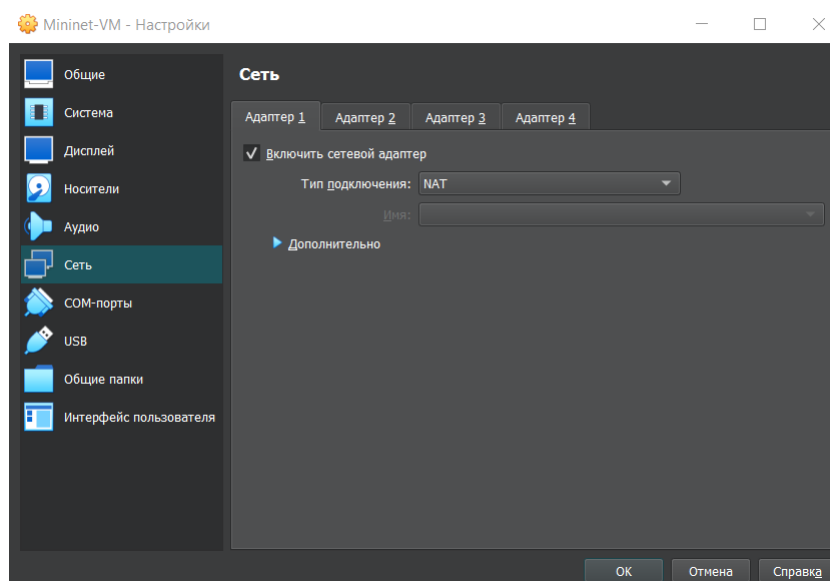


Рис. 3.3: Настройка сети

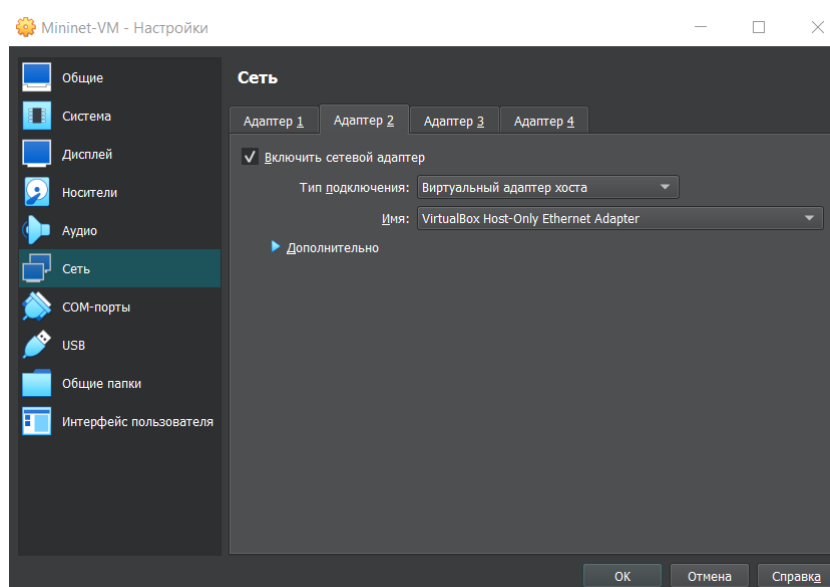
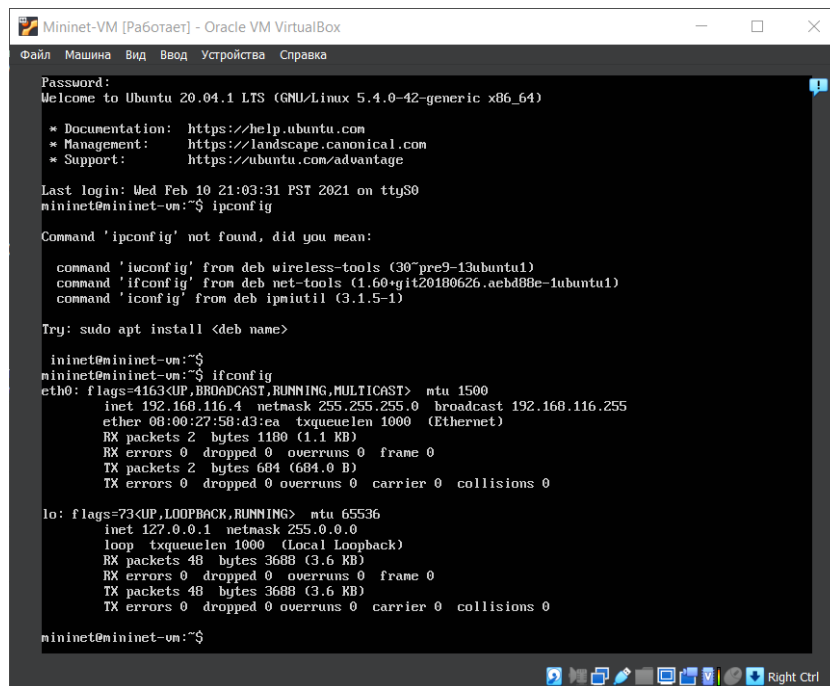


Рис. 3.4: Настройка сети

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

Посмотрите адрес машины с помощью ifconfig (рис. 3.5).



```
Mininet-VM [Работаer] - Oracle VM VirtualBox
Файл Машина Вид Ввод Устройства Справка

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

Last login: Wed Feb 10 21:03:31 PST 2021 on ttyS0
mininet@mininet-vm:~$ ipconfig

Command 'ipconfig' not found, did you mean:

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

Try: sudo apt install <deb name>

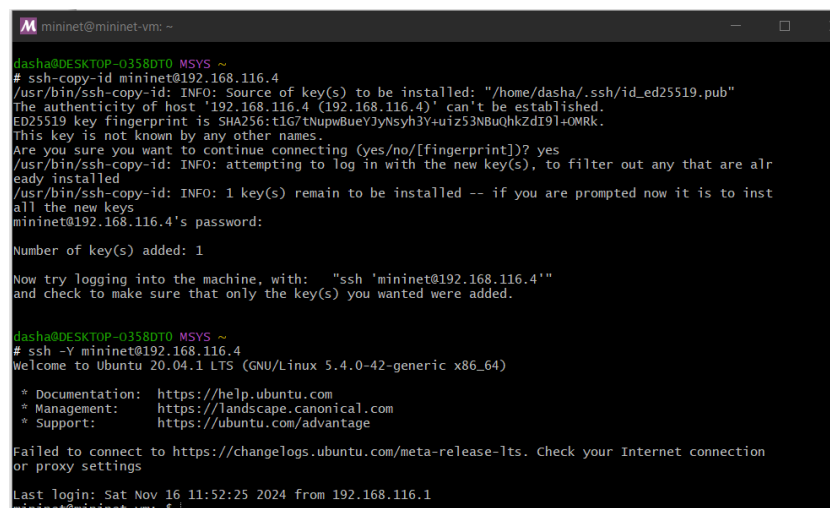
mininet@mininet-vm:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.116.4  netmask 255.255.255.0  broadcast 192.168.116.255
    ether 08:00:27:58:d3:ea  txqueuelen 1000  (Ethernet)
    RX packets 2  bytes 1180 (1.1 KB)
    RX errors 0  dropped 0  overruns 0  frame 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 48  bytes 3688 (3.6 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 48  bytes 3688 (3.6 KB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

mininet@mininet-vm:~$
```

Рис. 3.5: 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. 3.6).



```
mininet@mininet-vm: ~
dasha@DESKTOP-0358D70 MSYS ~
# ssh-copy-id mininet@192.168.116.4
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/dasha/.ssh/id_ed25519.pub"
The authenticity of host '192.168.116.4 (192.168.116.4)' can't be established.
ED25519 key fingerprint is SHA256:tlG7tnupwBueYjyNsyh3Y+uiz53NbuQhkZdI9l+QMRK.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are alr
eady installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to inst
all the new keys
mininet@192.168.116.4's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'mininet@192.168.116.4'"
and check to make sure that only the key(s) you wanted were added.

dasha@DESKTOP-0358D70 MSYS ~
# ssh -Y mininet@192.168.116.4
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 Nov 16 11:52:25 2024 from 192.168.116.1
mininet@mininet-vm:~$
```

Рис. 3.6: 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. 3.7).

```
mininet@mininet-vm:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.116.4 netmask 255.255.255.0 broadcast 192.168.116.255
    ether 08:00:27:58:d3:ea txqueuelen 1000 (Ethernet)
    RX packets 579 bytes 60169 (60.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 789 bytes 119161 (119.1 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 1808 bytes 142352 (142.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1808 bytes 142352 (142.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

mininet@mininet-vm:~$ sudo dhclient eth1
mininet@mininet-vm:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.116.4 netmask 255.255.255.0 broadcast 192.168.116.255
    ether 08:00:27:58:d3:ea txqueuelen 1000 (Ethernet)
    RX packets 595 bytes 63046 (63.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 802 bytes 121883 (121.8 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:bb:30:c0 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 2344 bytes 183376 (183.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2344 bytes 183376 (183.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

mininet@mininet-vm:~$
```

Рис. 3.7: 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. 3.8).

```
mininet@mininet-vm: ~  
/etc/netplan/01-netcfg.yaml [-M-] 16 L: [ 1+ 9 10/ 11] *(219 / 220b) 10 0x00A  
# This file describes the network interfaces available on your system  
# For more information, see netplan(5).  
network:  
  version: 2  
  renderer: networkd  
  ethernet:  
    eth0:  
      dhcp4: yes  
    eth1:  
      dhcp4: yes
```

Рис. 3.8: Файл /etc/netplan/01-netcfg.yaml

В виртуальной машине mininet переименуем предыдущую установку Mininet. Скачаем новую версию Mininet. Обновим исполняемые файлы (рис. 3.9).

```
Mininet-VM [Работает] - Oracle VM VirtualBox  
Файл Машина Вид Ввод Устройства Справка  
mininet@mininet-vm:~$ mv ~/mininet ~/mininet.orig  
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% (234/234), done.  
remote: Compressing objects: 100% (140/140), done.  
remote: Total 10388 (delta 129), reused 174 (delta 92), pack-reused 10154 (from 1)  
Receiving objects: 100% (10388/10388), 3.36 MiB | 9.07 MiB/s, done.  
Resolving deltas: 100% (6911/6911), done.  
mininet@mininet-vm:~$ cd ~/mininet  
mininet@mininet-vm:~/mininet$ sudo make install  
cc -Wall -Wextra -c \n  
-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.\" \n  
--no-discard-stderr \"python -B bin/mn\" -o mn.1  
help2man -N -n \"execution utility for Mininet.\" \n  
-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=559efd9077  
273f465e22e4a85bb2c33a29982cce7836c10a7eae5307972f734d  
  Stored in directory: /tmp/pip-ephem-wheel-cache-vhiuvtq/wheels/cd/7d/a7/aafef13eaff31ef46ba4e2ea6  
c9690a717bdf739db6cfe8d45  
Successfully built mininet
```

Рис. 3.9: Обновление Mininet

Проверим номер установленной версии mininet (рис. 3.10).

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

Рис. 3.10: 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)1 пользователя mininet в файл для пользователя root (рис. 3.11).

```
mininet@mininet-vm:~/mininet$ cd ~
mininet@mininet-vm:~$ xauth list $DISPLAY
mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 a39c1b15252480c385b5440b07af8a36
mininet@mininet-vm:~$ sudo -i
root@mininet-vm:~# xauth list
xauth: file /root/.Xauthority does not exist
root@mininet-vm:~# xauth add mininet-vm/unix:10
xauth: file /root/.Xauthority does not exist
xauth: (argv):1: bad "add" command line
root@mininet-vm:~# xauth add mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 a39c1b15252480c385b5440b07af8a36
xauth: file /root/.Xauthority does not exist
root@mininet-vm:~# xauth list $DISPLAY
mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 a39c1b15252480c385b5440b07af8a36
root@mininet-vm:~# logout
mininet@mininet-vm:~$ sudo -i
root@mininet-vm:~# xauth list
mininet-vm/unix:10 MIT-MAGIC-COOKIE-1 a39c1b15252480c385b5440b07af8a36
root@mininet-vm:~#
```

Рис. 3.11: Настройка соединения X11 для суперпользователя

Для запуска минимальной топологии введем в командной строке (рис. 3.12): `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.

```

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

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
mininet>

```

Рис. 3.12: 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`.

```
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 d6:13:98:2e:1f:5f 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. 3.13: Работа с Mininet с помощью командной строки

Посмотрим конфигурацию всех узлов.

```

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 82:aa:90:75:69:1f 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 192.168.116.4 netmask 255.255.255.0 broadcast 192.168.116.255
        ether 08:00:27:58:d3:ea txqueuelen 1000 (Ethernet)
        RX packets 1056 bytes 100323 (100.3 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 1274 bytes 185531 (185.5 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:bb:30:c0 txqueuelen 1000 (Ethernet)
        RX packets 4078 bytes 5864974 (5.8 MB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 959 bytes 66278 (66.2 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 2508 bytes 194466 (194.4 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 2508 bytes 194466 (194.4 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

s1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        ether ae:9f:5c:52:d5:52 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 06:d3:d8:3b:85:a8 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)

```

Рис. 3.14: 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.

```
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=19.5 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.338 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.088 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.092 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.095 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.087 ms
^C
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5105ms
rtt min/avg/max/mdev = 0.087/3.373/19.540/7.230 ms
mininet>
```

Рис. 3.15: Проверка связности хостов

4 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: 11.12.2024).