

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

дисциплина: Сетевые технологии

Студент: Сахно Никита

Группа: НФИбд-02-23

МОСКВА

2025 г.

Цель работы

Построить простейшие модели сетей на базе коммутатора и маршрутизаторов FRR и VyOS в GNS3, проанализировать трафик посредством Wireshark.

Задание

1. Смоделировать простейшую сеть на базе коммутатора в GNS3;
2. Проанализировать трафик в GNS3 посредством Wireshark;
3. Смоделировать простейшую сеть на базе маршрутизатора FRR в GNS3;
4. Смоделировать простейшую сеть на базе маршрутизатора VyOS в GNS3.

Выполнение лабораторной работы

№1

Для начала запустим GNS3 VM и GNS3, а также создадим новый проект.

В рабочей области GNS3 разместим коммутатор Ethernet и два VPCS. В меню Configure изменим название устройства, включив в имя устройства имя моей учётной записи. Коммутатору присвоим название msk-nvsakhno-sw-01. Соединим VPCS с коммутатором и отобразим обозначение интерфейсов соединения.

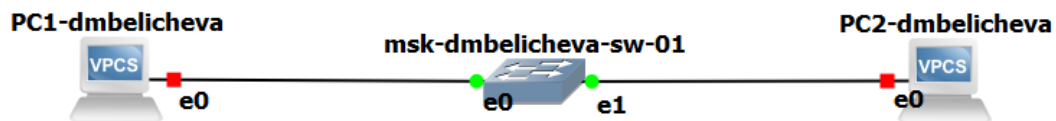


Рисунок 1. Топология простейшей сети в GNS3

Зададим IP-адреса VPCS. Для этого с помощью меню, вызываемого правой кнопкой мыши, запустим Start PC-1, затем вызовем его терминал Console. Для просмотра синтаксиса возможных для ввода команд можно набрать /?.

```
Welcome to Virtual PC Simulator, version 0.6.2
Dedicated to Daling.
Build time: Apr 10 2019 02:42:20
Copyright (c) 2007-2014, Paul Meng (mirnshi@gmail.com)
All rights reserved.

VPCS is free software, distributed under the terms of the "BSD" licence.
Source code and license can be found at vpcs.sf.net.
For more information, please visit wiki.freecode.com.cn.

Press '?' to get help.

Executing the startup file

Checking for duplicate address...
PC1 : 192.168.1.11 255.255.255.0 gateway 192.168.1.1

VPCS> /?

?                Print help
! COMMAND [ARG ...] Invoke an OS COMMAND with optional ARG(s)
                  Shortcut for: show arp. Show arp table
arp              Clear IPv4/IPv6, arp/neighbor cache, command history
clear ARG        Shortcut for: ip dhcp. Get IPv4 address via DHCP
dhcp [OPTION]    Exit the telnet session (daemon mode)
disconnect       Display TEXT in output. See also set echo ?
echo TEXT        Print help
help             Shortcut for: show history. List the command history
history          Configure the current VPC's IP settings. See ip ?
ip ARG ... [OPTION] Load the configuration/script from the file FILENAME
load [FILENAME]  Ping HOST with ICMP (default) or TCP/UDP. See ping ?
ping HOST [OPTION ...] Quit program
quit            Configure packet relay between UDP ports. See relay ?
relay ARG ...   Telnet to port on host at ip (relative to host PC)
rlogin [ip] port Save the configuration to the file FILENAME
save [FILENAME]  Set VPC name and other options. Try set ?
set ARG ...      Print the information of VPCS (default). See show ?
show [ARG ...]   Print TEXT and pause running script for seconds
sleep [seconds] [TEXT] Print the path packets take to network HOST
trace HOST [OPTION ...] Shortcut for: show version
version

To get command syntax help, please enter '?' as an argument of the command.

VPCS> █
```

Рисунок 2. Окно терминала PC1

Для задания IP-адреса 192.168.1.11 в сети 192.168.1.0/24 введем:

ip 192.168.1.11/24 192.168.1.1

Для сохранения конфигурации введем команду save.

```
VPCS> ip 192.168.1.11/24 192.168.1.1
Checking for duplicate address...
PC1 : 192.168.1.11 255.255.255.0 gateway 192.168.1.1

VPCS> save
Saving startup configuration to startup.vpc
. done
```

Рисунок 3. Задание IP-адреса для PC-1

Аналогичным образом зададим IP-адрес 192.168.1.12 для PC-2.

```
VPCS> ip 192.168.1.12/24 192.168.1.1
Checking for duplicate address...
PC1 : 192.168.1.12 255.255.255.0 gateway 192.168.1.1

VPCS> save
Saving startup configuration to startup.vpc
. done

VPCS> █
```

Рисунок 4. Задание IP-адреса для PC-2

Проверим работоспособность соединения между PC-1 и PC-2 с помощью команды ping.

В терминале PC-1 введем команду ping и IP-адрес, присвоенный PC-2. Получаем эхо-ответ от PC-2 (возвращены четыре пакета).

```
VPCS> ping 192.168.1.12
84 bytes from 192.168.1.12 icmp_seq=1 ttl=64 time=1.174 ms
84 bytes from 192.168.1.12 icmp_seq=2 ttl=64 time=1.268 ms
84 bytes from 192.168.1.12 icmp_seq=3 ttl=64 time=0.981 ms
84 bytes from 192.168.1.12 icmp_seq=4 ttl=64 time=1.651 ms
84 bytes from 192.168.1.12 icmp_seq=5 ttl=64 time=0.932 ms

VPCS> █
```

Рисунок 5. Пингование PC-2

То же самое делаем из терминала PC-2, пингуем соответственно IP-адрес PC-1. Получаем эхо-ответ от PC-1. Значит соединение наших PC работоспособно.

```
VPCS> ping 192.168.1.11
84 bytes from 192.168.1.11 icmp_seq=1 ttl=64 time=1.063 ms
84 bytes from 192.168.1.11 icmp_seq=2 ttl=64 time=1.139 ms
84 bytes from 192.168.1.11 icmp_seq=3 ttl=64 time=1.197 ms
84 bytes from 192.168.1.11 icmp_seq=4 ttl=64 time=1.198 ms
84 bytes from 192.168.1.11 icmp_seq=5 ttl=64 time=1.127 ms

VPCS> █
```

Рисунок 6. Пингование PC-1

Остановим в проекте все узлы.

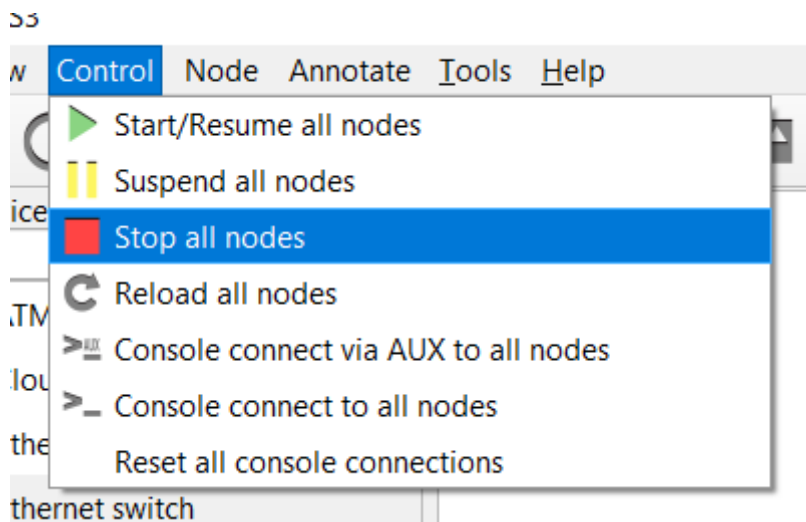


Рисунок 7. Остановка всех узлов

№2

Запустим на соединении между PC-1 и коммутатором анализатор трафика. Для этого щёлкнем правой кнопкой мыши на соединении, выберем в меню Start capture. После этого запустился Wireshark, а в проекте GNS3 на соединении появился значок лупы. В проекте GNS3 стартуем все узлы.

3

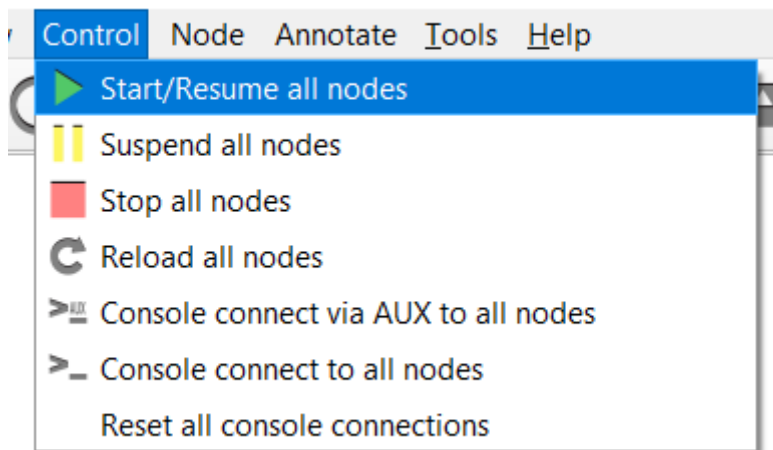


Рисунок 8. Старт всех узлов

В окне Wireshark отобразится информация по протоколу ARP.

В поле кадра физического уровня мы можем узнать длину кадра (в моем случае было 64 бита). В поле канального уровня можем посмотреть mac-адреса источника и получателя. По нулевому и первому битам можем определить тип mac-адресов (получатель – локально администрируемый и широковещательный; источник - глобально администрируемый и одиночный).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	::	ff02::2	ICMPv6	62	Router Solicitation
2	0.063350	Private_66:68:00	Broadcast	ARP	64	Gratuitous ARP for 192.168.1.11 (Request)
3	0.100734	::	ff02::2	ICMPv6	62	Router Solicitation
4	0.163877	Private_66:68:01	Broadcast	ARP	64	Gratuitous ARP for 192.168.1.12 (Request)
5	1.063804	Private_66:68:00	Broadcast	ARP	64	Gratuitous ARP for 192.168.1.11 (Request)
6	1.164420	Private_66:68:01	Broadcast	ARP	64	Gratuitous ARP for 192.168.1.12 (Request)
7	2.065362	Private_66:68:00	Broadcast	ARP	64	Gratuitous ARP for 192.168.1.11 (Request)
8	2.165133	Private_66:68:01	Broadcast	ARP	64	Gratuitous ARP for 192.168.1.12 (Request)
9	624.660173	Private_66:68:01	Broadcast	ARP	64	Who has 192.168.1.11? Tell 192.168.1.12
10	624.661213	Private_66:68:00	Private_66:68:01	ARP	64	192.168.1.11 is at 00:50:79:66:68:00
11	624.662837	192.168.1.12	192.168.1.11	ICMP	98	Echo (ping) request id=0x6262, seq=1/256, tt
12	624.663931	192.168.1.11	192.168.1.12	ICMP	98	Echo (ping) reply id=0x6262, seq=1/256, tt

[Coloring Rule Name: ARP]
[Coloring Rule String: arp]
▼ Ethernet II, Src: Private_66:68:00 (00:50:79:66:68:00), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 ▼ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
 Address: Broadcast (ff:ff:ff:ff:ff:ff)
 1. = LG bit: Locally administered address (this is NOT the factory default)
 1. = IG bit: Group address (multicast/broadcast)
 ▼ Source: Private_66:68:00 (00:50:79:66:68:00)
 Address: Private_66:68:00 (00:50:79:66:68:00)
 0. = LG bit: Globally unique address (factory default)
 0. = IG bit: Individual address (unicast)
 Type: ARP (0x0806)
 Padding: 00000000000000000000000000000000
 Frame check sequence: 0x00000000 [unverified]
 [FCS Status: Unverified]
 ▼ Address Resolution Protocol (request/gratuitous ARP)
 Hardware type: Ethernet (1)
 Protocol type: IPv4 (0x0800)
 Hardware size: 6
 Protocol size: 4
 Opcode: request (1)
 [Is gratuitous: True]
 Sender MAC address: Private_66:68:00 (00:50:79:66:68:00)
 Sender IP address: 192.168.1.11
 Target MAC address: Broadcast (ff:ff:ff:ff:ff:ff)
 Target IP address: 192.168.1.11

Рисунок 9. Информация по протоколу ARP

В терминале PC-2 посмотрим информацию по опциям команды ping. Затем сделаем один эхо-запрос в ICMP-моду к узлу PC-1. Для этого используем опцию -1.

```
VPCS> ping 192.168.1.11 -1
84 bytes from 192.168.1.11 icmp_seq=1 ttl=64 time=1.610 ms
84 bytes from 192.168.1.11 icmp_seq=2 ttl=64 time=1.630 ms
84 bytes from 192.168.1.11 icmp_seq=3 ttl=64 time=1.831 ms
84 bytes from 192.168.1.11 icmp_seq=4 ttl=64 time=1.621 ms
84 bytes from 192.168.1.11 icmp_seq=5 ttl=64 time=1.610 ms
```

Рисунок 10. Эхо-запрос в ICMP-моду

Далее откроем Wireshark и проанализируем эхо-запрос по протоколу ICMP. В поле канального уровня можем посмотреть mac-адреса источника и получателя. По нулевому и первому битам можем определить тип mac-адресов (получатель и источник - глобально администрируемые и одиночные, так как биты равны 0). В поле сетевого уровня указан протокол ICMP и IP-адреса источника (192.168.1.12, то есть PC-2) и получателя (192.168.1.11, то есть PC-2).

No.	Time	Source	Destination	Protocol	Length	Info
11	624.662837	192.168.1.12	192.168.1.11	ICMP	98	Echo (ping) request id=0x6262, seq=1/256, ttl=64 (reply in 12)
12	624.663931	192.168.1.11	192.168.1.12	ICMP	98	Echo (ping) reply id=0x6262, seq=1/256, ttl=64 (request in 11)
13	625.665766	192.168.1.12	192.168.1.11	ICMP	98	Echo (ping) request id=0x6362, seq=2/512, ttl=64 (reply in 14)
14	625.666306	192.168.1.11	192.168.1.12	ICMP	98	Echo (ping) reply id=0x6362, seq=2/512, ttl=64 (request in 13)
15	626.668282	192.168.1.12	192.168.1.11	ICMP	98	Echo (ping) request id=0x6462, seq=3/768, ttl=64 (reply in 16)
16	626.668846	192.168.1.11	192.168.1.12	ICMP	98	Echo (ping) reply id=0x6462, seq=3/768, ttl=64 (request in 15)
17	627.670081	192.168.1.12	192.168.1.11	ICMP	98	Echo (ping) request id=0x6562, seq=4/1024, ttl=64 (reply in 18)
18	627.671168	192.168.1.11	192.168.1.12	ICMP	98	Echo (ping) reply id=0x6562, seq=4/1024, ttl=64 (request in 17)
19	628.672298	192.168.1.12	192.168.1.11	ICMP	98	Echo (ping) request id=0x6662, seq=5/1280, ttl=64 (reply in 20)
20	628.673385	192.168.1.11	192.168.1.12	ICMP	98	Echo (ping) reply id=0x6662, seq=5/1280, ttl=64 (request in 19)
21	1190.613747	Private_66:68:01	Broadcast	ARP	64	Who has 192.168.1.11? Tell 192.168.1.12
22	1190.614816	Private_66:68:00	Private_66:68:01	ARP	64	192.168.1.11 is at 00:50:79:66:68:00

[Coloring Rule String: icmp || icmpv6]

- ✓ Ethernet II, Src: Private_66:68:01 (00:50:79:66:68:01), Dst: Private_66:68:00 (00:50:79:66:68:00)
 - ✓ Destination: Private_66:68:00 (00:50:79:66:68:00)
 - Address: Private_66:68:00 (00:50:79:66:68:00)
 - 0. = LG bit: Globally unique address (factory default)
 - 0. = IG bit: Individual address (unicast)
 - ✓ Source: Private_66:68:01 (00:50:79:66:68:01)
 - Address: Private_66:68:01 (00:50:79:66:68:01)
 - 0. = LG bit: Globally unique address (factory default)
 - 0. = IG bit: Individual address (unicast)
 - Type: IPv4 (0x0800)
- ✓ Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.11
 - 0100 = Version: 4
 - 0101 = Header Length: 20 bytes (5)
 - > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 - Total Length: 84
 - Identification: 0x6262 (25186)
 - > 000. = Flags: 0x0
 - ...0 0000 0000 0000 = Fragment Offset: 0
 - Time to Live: 64
 - Protocol: ICMP (1)
 - Header Checksum: 0x94df [validation disabled]
 - [Header checksum status: Unverified]
 - Source Address: 192.168.1.12
 - Destination Address: 192.168.1.11

Рисунок 11. Полученная информация по эхо-запросу в ICMP-моду к узлу PC-1

- ✓ **Internet Control Message Protocol**
 - Type: 8 (Echo (ping) request)
 - Code: 0
 - Checksum: 0xbda8 [correct]
 - [Checksum Status: Good]
 - Identifier (BE): 25186 (0x6262)
 - Identifier (LE): 25186 (0x6262)
 - Sequence Number (BE): 1 (0x0001)
 - Sequence Number (LE): 256 (0x0100)
 - [Response frame: 12]
 - > Data (56 bytes)

Рисунок 12 Поле уровня протокола ARP

Сделаем один эхо-запрос в UDP-моду к узлу PC-1. Для этого используем опцию -2.

```
VPCS> ping 192.168.1.11 -2
84 bytes from 192.168.1.11 udp_seq=1 ttl=64 time=1.429 ms
84 bytes from 192.168.1.11 udp_seq=2 ttl=64 time=1.537 ms
84 bytes from 192.168.1.11 udp_seq=3 ttl=64 time=1.523 ms
84 bytes from 192.168.1.11 udp_seq=4 ttl=64 time=1.614 ms
84 bytes from 192.168.1.11 udp_seq=5 ttl=64 time=1.479 ms
```

Рисунок 13. Эхо-запрос в UDP-моду

Сделаем один эхо-запрос в TCP-моду к узлу PC-1. Для этого используем опцию -3.

```
VPCS> ping 192.168.1.11 -3
Connect 7@192.168.1.11 seq=1 ttl=64 time=2.169 ms
SendData 7@192.168.1.11 seq=1 ttl=64 time=2.162 ms
Close 7@192.168.1.11 seq=1 ttl=64 time=3.491 ms
Connect 7@192.168.1.11 seq=2 ttl=64 time=2.198 ms
SendData 7@192.168.1.11 seq=2 ttl=64 time=2.178 ms
Close 7@192.168.1.11 seq=2 ttl=64 time=3.282 ms
Connect 7@192.168.1.11 seq=3 ttl=64 time=2.211 ms
SendData 7@192.168.1.11 seq=3 ttl=64 time=2.209 ms
Close 7@192.168.1.11 seq=3 ttl=64 time=3.296 ms
Connect 7@192.168.1.11 seq=4 ttl=64 time=2.215 ms
SendData 7@192.168.1.11 seq=4 ttl=64 time=2.192 ms
Close 7@192.168.1.11 seq=4 ttl=64 time=3.291 ms
Connect 7@192.168.1.11 seq=5 ttl=64 time=2.201 ms
SendData 7@192.168.1.11 seq=5 ttl=64 time=2.177 ms
Close 7@192.168.1.11 seq=5 ttl=64 time=3.198 ms
```

Рисунок 15. Эхо-запрос в TCP-моду

Далее откроем Wireshark и проанализируем эхо-запрос по протоколу TCP. В поле канального уровня можем посмотреть mac-адреса источника и получателя. По нулевому и первому битам можем определить тип mac-адресов (получатель и источник - глобально администрируемые и одиночные, так как биты равны 0). В поле сетевого уровня указан протокол TCP и IP-адреса источника (192.168.1.12, то есть PC-2) и получателя (192.168.1.11, то есть PC-2).

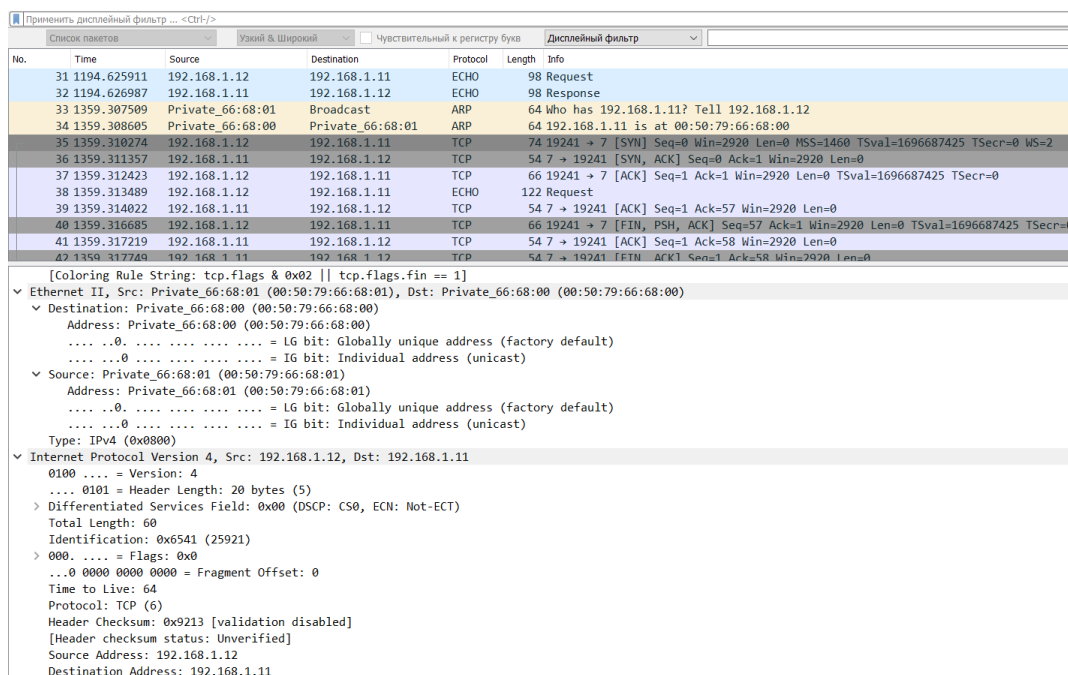


Рисунок 16. Полученная информация по эхо-запросу в TCP-моду к узлу PC-1

В поле протокола TCP можем узнать порты источника (19241) и получателя (7). А также посмотреть, как работает handshake протокола TCP. На первом шаге установлен флаг SYN, а также Порядковому номеру (Sequence Number) присвоено начальное 32-битное значение ISSa (в нашем случае 1741785218).

```
-----
v Transmission Control Protocol, Src Port: 19241, Dst Port: 7, Seq: 0, Len: 0
  Source Port: 19241
  Destination Port: 7
  [Stream index: 0]
  [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 0]
  Sequence Number: 0      (relative sequence number)
  Sequence Number (raw): 1741785218
  [Next Sequence Number: 1      (relative sequence number)]
  Acknowledgment Number: 0
  Acknowledgment number (raw): 0
  1010 .... = Header Length: 40 bytes (10)
> Flags: 0x002 (SYN)
  Window: 2920
  [Calculated window size: 2920]
  Checksum: 0xb650 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
> Options: (20 bytes), Maximum segment size, No-Operation (NOP), No-Operation (
```

Рисунок 17. Информация по протоколу TCP

Теперь можно остановить захват трафика в Wireshark.

№3

Теперь нам нужно построить в GNS3 топологию сети, состоящей из маршрутизатора FRR, коммутатора Ethernet и оконечного устройства.

К сожалению, у меня возникают проблемы после запуска терминала маршрутизатора FRR, и нет возможности вводить команды. Видимо это из-за того, что у меня операционная система Windows. Поэтому выполнить задание №3 я не могу.

```
FRR8.2.2-1 - PuTTY
[ 1.276106] kernel_init+0x11/0x120
[ 1.276106] ret_from_fork+0x22/0x30
[ 1.276106] </TASK>
[ 1.276106] Modules linked in:
[ 1.276106] ---[ end trace f27f28dd31d11e78 ]---
[ 1.276106] RIP: 0010:native_write_msr+0x6/0x30
[ 1.276106] Code: b8 40 00 00 00 0f 00 d8 89 ef e8 a5 5
78 00 66 66 2e 0f 1f 84 00 00 00 00 00 0f 1f 00 89 f0 89
89 c2 89 c1 89 c6 89 c7 c3 cc cc cc cc 48 c1 e2 20 48
[ 1.276106] RSP: 0018:ffffbb1640013e30 EFLAGS: 00010056
[ 1.276106] RAX: 0000000000000400 RBX: 0000000000000000
[ 1.276106] RDX: 0000000000000000 RSI: 0000000000000400
[ 1.276106] RBP: 0000000000000400 R08: 0000000000000000
[ 1.276106] R10: 0000000000000000 R11: 0000000000000000
[ 1.276106] R13: 0000000000080000 R14: 00000000000151c0
[ 1.276106] FS: 0000000000000000(0000) GS:ffff9efd0f20
000000000000
[ 1.276106] CS: 0010 DS: 0000 ES: 0000 CR0: 0000000080
[ 1.276106] CR2: ffff9efd06801000 CR3: 000000000600a000
[ 1.276106] Kernel panic - not syncing: Attempted to ki
000000b
[ 1.276106] ---[ end Kernel panic - not syncing: Attempt
code=0x0000000b ]---
```

Рисунок 18. Терминал маршрутизатора FRR

№4

Запустим GNS3 VM и GNS3. Создадим новый проект.

В рабочей области GNS3 разместим VPCS, коммутатор Ethernet и маршрутизатор VyOS.

Изменим отображаемые названия устройств. Коммутатору присвоим название msk-dmbelicheva-sw-01, маршрутизатору — по принципу msk-

dmbelicheva-gw-01, VPCS — по принципу PC1-dmbelicheva.

Включим захват трафика на соединении между коммутатором и маршрутизатором (появится значок лупы).

Запустим все устройства проекта и откроем консоль всех устройств проекта. Откроем окно терминала PC-1 и настроим IP-адресацию для интерфейса этого узла.

```
VPCS> ip 192.168.1.10/24 192.168.1.1
Checking for duplicate address...
PC1 : 192.168.1.10 255.255.255.0 gateway 192.168.1.1

VPCS> save
Saving startup configuration to startup.vpc
. done

VPCS> show ip

NAME           : VPCS[1]
IP/MASK         : 192.168.1.10/24
GATEWAY         : 192.168.1.1
DNS             :
MAC             : 00:50:79:66:68:00
LPORT          : 10003
RHOST:PORT      : 127.0.0.1:10004
MTU:            : 1500
```

Рисунок 19. Настройка IP-адресации для интерфейса узла PC-1

Настроим маршрутизатор VyOS:

Во-первых, после загрузки нужно ввести логин vyos и пароль vyos.

В рабочем режиме в командной строке отобразится символ \$.

```
Welcome to VyOS - vyos ttyS0

vyos login: vyos
Password:
Welcome to VyOS!

Check out project news at https://blog.vyos.io
and feel free to report bugs at https://vyos.dev

You can change this banner using "set system login banner post-login" command.

VyOS is a free software distribution that includes multiple components,
you can check individual component licenses under /usr/share/doc/*/copyright
vyos@vyos:~$ configure
```

Рисунок 20. Настройка маршрутизатора VyOS

Далее надо установить систему на диск с помощью команды *install image*, но у меня эта система уже была.

Следующим шагом перейдем в режим конфигурирования с помощью команды *configure*. Изменим имя устройства командой *set system host-name msk-nvsakhno-gw-01*

Зададим IP-адрес на интерфейсе *eth0* командой *set interfaces ethernet eth0 address 192.168.1.1/24*

Посмотрим внесённые в конфигурацию изменения с помощью команды *compare*. Далее применим изменения в конфигурации и сохраним саму конфигурацию с помощью команд *commit* и *save*.

Посмотрим информацию об интерфейсах маршрутизатора с помощью команды *show interfaces*.

Выйдем из режима конфигурирования, используя команду *exit*.

```
vyos@vyos# commit
[edit]
vyos@vyos# save
Saving configuration to '/config/config.boot'...
Done
[edit]
vyos@vyos# show interfaces
  ethernet eth0 {
    address 192.168.1.1/24
    hw-id 0c:0b:4c:9a:00:00
  }
  ethernet eth1 {
    hw-id 0c:0b:4c:9a:00:01
  }
  ethernet eth2 {
    hw-id 0c:0b:4c:9a:00:02
  }
  loopback lo {
  }
[edit]
vyos@vyos# exit
exit
```

Рисунок 21. Режим конфигурирования

Теперь проверим подключение. Узел PC1 должен успешно отправлять эхо-запросы на адрес маршрутизатора 192.168.1.1. Для проверки пингуем

маршрутизатор. Получили эхо-ответ (4 пакета).

```
VPCS> ping 192.168.1.1
84 bytes from 192.168.1.1 icmp_seq=1 ttl=64 time=2.537 ms
84 bytes from 192.168.1.1 icmp_seq=2 ttl=64 time=2.525 ms
84 bytes from 192.168.1.1 icmp_seq=3 ttl=64 time=2.428 ms
84 bytes from 192.168.1.1 icmp_seq=4 ttl=64 time=2.882 ms
84 bytes from 192.168.1.1 icmp_seq=5 ttl=64 time=2.530 ms
```

Рисунок 22. Пингование маршрутизатора

В окне Wireshark проанализируем полученную информацию. В поле канального уровня можем посмотреть mac-адреса источника и получателя. По нулевому и первому битам можем определить тип mac-адресов (получатель и источник - глобально администрируемые и одиночные, так как биты равны 0). В поле сетевого уровня указан протокол ICMP и IP-адреса источника (192.168.1.10, то есть PC-1) и получателя (192.168.1.1, то есть маршрутизатор VyOS).

No.	Time	Source	Destination	Protocol	Length	Info
618	2408.799133	0c:0b:4c:9a:00:00	Private_66:68:00	ARP	60	192.168.1.1 is at 0c:0b:4c:9a:00:00
619	2408.800211	192.168.1.10	192.168.1.1	ICMP	98	Echo (ping) request id=0x6783, seq=1/256, ttl=64 (reply in 620)
620	2408.802370	192.168.1.1	192.168.1.10	ICMP	98	Echo (ping) reply id=0x6783, seq=1/256, ttl=64 (request in 619)
621	2409.803711	192.168.1.10	192.168.1.1	ICMP	98	Echo (ping) request id=0x6883, seq=2/512, ttl=64 (reply in 622)
622	2409.805848	192.168.1.1	192.168.1.10	ICMP	98	Echo (ping) reply id=0x6883, seq=2/512, ttl=64 (request in 621)
623	2410.807115	192.168.1.10	192.168.1.1	ICMP	98	Echo (ping) request id=0x6983, seq=3/768, ttl=64 (reply in 624)
624	2410.808775	192.168.1.1	192.168.1.10	ICMP	98	Echo (ping) reply id=0x6983, seq=3/768, ttl=64 (request in 623)
625	2411.810818	192.168.1.10	192.168.1.1	ICMP	98	Echo (ping) request id=0x6a83, seq=4/1024, ttl=64 (reply in 626)
626	2411.812988	192.168.1.1	192.168.1.10	ICMP	98	Echo (ping) reply id=0x6a83, seq=4/1024, ttl=64 (request in 625)

[Time delta from previous displayed frame: 0.001078000 seconds]
[Time since reference or first frame: 2408.800211000 seconds]
Frame Number: 619
Frame Length: 98 bytes (784 bits)
Capture Length: 98 bytes (784 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ethertype:ip:icmp:data]
[Coloring Rule Name: ICMP]
[Coloring Rule String: icmp || icmpv6]

▼ Ethernet II, Src: Private_66:68:00 (00:50:79:66:68:00), Dst: 0c:0b:4c:9a:00:00 (0c:0b:4c:9a:00:00)
 ▼ Destination: 0c:0b:4c:9a:00:00 (0c:0b:4c:9a:00:00)
 Address: 0c:0b:4c:9a:00:00 (0c:0b:4c:9a:00:00)
 ...0. = LG bit: Globally unique address (factory default)
 ...0. = IG bit: Individual address (unicast)

 ▼ Source: Private_66:68:00 (00:50:79:66:68:00)
 Address: Private_66:68:00 (00:50:79:66:68:00)
 ...0. = LG bit: Globally unique address (factory default)
 ...0. = IG bit: Individual address (unicast)
 Type: IPv4 (0x0800)

▼ Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.1.1
 0100 = Version: 4
 0101 = Header Length: 20 bytes (5)
 > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 Total Length: 84
 Identification: 0x8367 (33639)
 > 000. = Flags: 0x0
 ...0 0000 0000 0000 = Fragment Offset: 0
 Time to Live: 64
 Protocol: ICMP (1)

Рисунок 23. Полученная информация в Wireshark по ICMP-сообщению

Выводы:

В процессе выполнения данной лабораторной работы я построил простейшие модели сети на базе коммутатора и маршрутизатора VyOS в GNS3, проанализировал трафик посредством Wireshark.