МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра «Вычислительные системы и технологии»

Сети и телекоммуникации

Лабораторная работа №1

Работа с анализаторами протоколов tcpdump и wireshark

ПРОВЕРИЛ:

_____ Гай В.Е.

СТУДЕНТ:

Козменкова Е.П.
18 В-2

Нижний Новгород
2021 г.

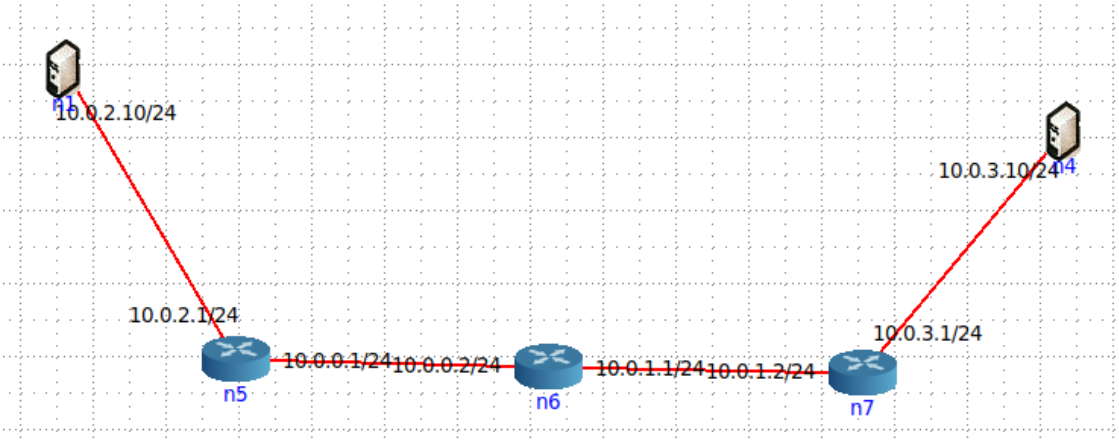
Цель:

Познакомиться с анализатором протоколов tcpdump и сравнить его работу с работой анализатора wireshark.

Ход работы:

tcpdump

Сеть (для всех, кроме п.2):



1. Запустить tcpdump в режиме захвата всех пакетов, проходящих по сети. Количество захватываемых пакетов ограничить 10. Результаты протоколировать в файл.

Команда:

tcpdump -c 10 -l | tee out.log

Для начала запустим команду ping 10.0.2.10 с компьютера 10.0.3.10:

```
root@n4:/tmp/pycore.41679/n4.conf# ping 10.0.2.10
PING 10.0.2.10 (10.0.2.10) 56(84) bytes of data.
64 bytes from 10.0.2.10: icmp_seq=1 ttl=61 time=0.124 ms
64 bytes from 10.0.2.10: icmp_seq=2 ttl=61 time=0.132 ms
64 bytes from 10.0.2.10: icmp_seq=3 ttl=61 time=0.172 ms
64 bytes from 10.0.2.10: icmp_seq=4 ttl=61 time=0.095 ms
64 bytes from 10.0.2.10: icmp_seq=5 ttl=61 time=0.135 ms
```

Теперь захватим пакеты с компьютера 10.0.2.10:

```
root@n1:/tmp/pycore.41679/n1.conf# tcpdump -c 10 -l | tee out.log
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
10:08:05.363409 IP 10.0.3.10 > n1: ICMP echo request, id 35, seq 14, length 64
10:08:05.363435 IP n1 > 10.0.3.10: ICMP echo reply, id 35, seq 14, length 64
10:08:06.387215 IP 10.0.3.10 > n1: ICMP echo request, id 35, seq 15, length 64
10:08:06.387231 IP n1 > 10.0.3.10: ICMP echo reply, id 35, seq 15, length 64
10:08:06.807283 IP _gateway > 224.0.0.5: OSPFv2, Hello, length 44
10:08:06.832737 IP6 fe80::200:ff:feaa:4 > ff02::5: OSPFv3, Hello, length 36
10:08:07.412229 IP 10.0.3.10 > n1: ICMP echo request, id 35, seq 16, length 64
10:08:07.412255 IP n1 > 10.0.3.10: ICMP echo reply, id 35, seq 16, length 64
10:08:08.435660 IP 10.0.3.10 > n1: ICMP echo request, id 35, seq 17, length 64
10:08:08.435686 IP n1 > 10.0.3.10: ICMP echo reply, id 35, seq 17, length 64
10 packets captured
11 packets received by filter
0 packets dropped by kernel
```

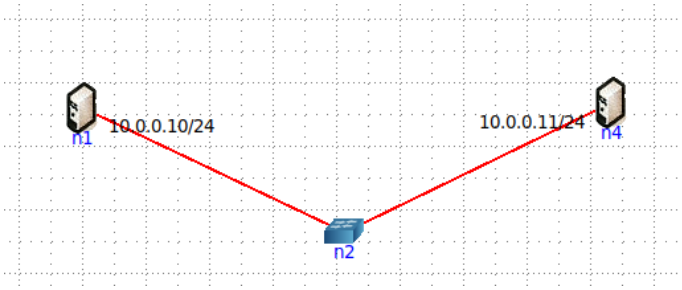
Проверим содержимое файла:

Открыть | out.log /tmp/pycore.41679/n1.conf | Сохранить

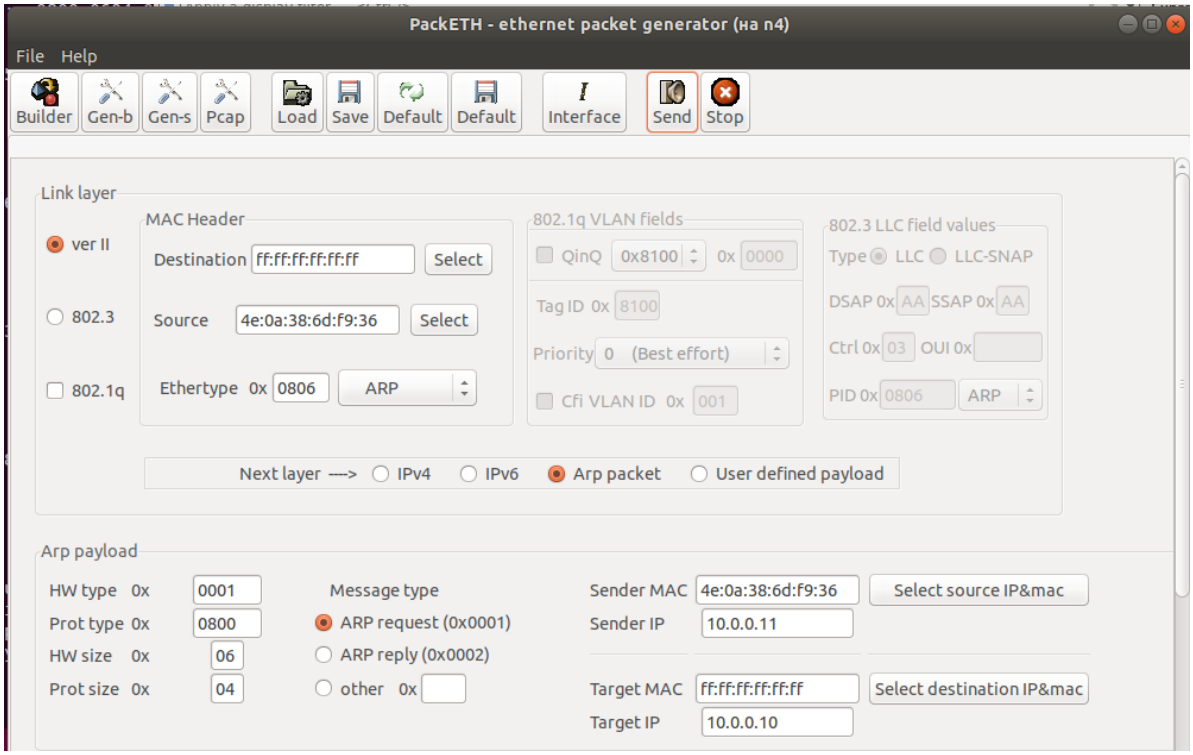
```
10:08:05.363409 IP 10.0.3.10 > n1: ICMP echo request, id 35, seq 14, length 64
10:08:05.363435 IP n1 > 10.0.3.10: ICMP echo reply, id 35, seq 14, length 64
10:08:06.387215 IP 10.0.3.10 > n1: ICMP echo request, id 35, seq 15, length 64
10:08:06.387231 IP n1 > 10.0.3.10: ICMP echo reply, id 35, seq 15, length 64
10:08:06.807283 IP _gateway > 224.0.0.5: OSPFv2, Hello, length 44
10:08:06.832737 IP6 fe80::200:ff:feaa:4 > ff02::5: OSPFv3, Hello, length 36
10:08:07.412229 IP 10.0.3.10 > n1: ICMP echo request, id 35, seq 16, length 64
10:08:07.412255 IP n1 > 10.0.3.10: ICMP echo reply, id 35, seq 16, length 64
10:08:08.435660 IP 10.0.3.10 > n1: ICMP echo request, id 35, seq 17, length 64
10:08:08.435686 IP n1 > 10.0.3.10: ICMP echo reply, id 35, seq 17, length 64
```

2. Запустить tcpdump в режиме перехвата широковещательного трафика (фильтр по MAC-адресу). Количество захватываемых пакетов ограничить 5. Включить распечатку пакета в шестнадцатеричной системе (включая заголовок канального уровня).
Команда:
tcpdump -c 5 -l -xx ether broadcast

Сеть:



Составлю кадр ARP запроса:



Результат:

```
root@n1:/tmp/pycore.39467/n1.conf# tcpdump -c 5 -l -xx ether broadcast
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
19:12:46.870011 ARP, Request who-has n1 (Broadcast) tell 10.0.0.11, length 46
        0x0000:  ffff ffff ffff 4e0a 386d f936 0806 0001
        0x0010:  0800 0604 0001 4e0a 386d f936 0a00 000b
        0x0020:  ffff ffff ffff 0a00 000a 0000 0000 0000
        0x0030:  0000 0000 0000 0000 0000 0000
19:12:47.894279 ARP, Request who-has n1 (Broadcast) tell 10.0.0.11, length 46
        0x0000:  ffff ffff ffff 4e0a 386d f936 0806 0001
        0x0010:  0800 0604 0001 4e0a 386d f936 0a00 000b
        0x0020:  ffff ffff ffff 0a00 000a 0000 0000 0000
        0x0030:  0000 0000 0000 0000 0000 0000
19:12:48.390388 ARP, Request who-has n1 (Broadcast) tell 10.0.0.11, length 46
        0x0000:  ffff ffff ffff 4e0a 386d f936 0806 0001
        0x0010:  0800 0604 0001 4e0a 386d f936 0a00 000b
        0x0020:  ffff ffff ffff 0a00 000a 0000 0000 0000
        0x0030:  0000 0000 0000 0000 0000 0000
19:12:48.776500 ARP, Request who-has n1 (Broadcast) tell 10.0.0.11, length 46
        0x0000:  ffff ffff ffff 4e0a 386d f936 0806 0001
        0x0010:  0800 0604 0001 4e0a 386d f936 0a00 000b
        0x0020:  ffff ffff ffff 0a00 000a 0000 0000 0000
        0x0030:  0000 0000 0000 0000 0000 0000
19:12:49.231083 ARP, Request who-has n1 (Broadcast) tell 10.0.0.11, length 46
        0x0000:  ffff ffff ffff 4e0a 386d f936 0806 0001
        0x0010:  0800 0604 0001 4e0a 386d f936 0a00 000b
        0x0020:  ffff ffff ffff 0a00 000a 0000 0000 0000
        0x0030:  0000 0000 0000 0000 0000 0000
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

3. Запустить tcpdump так, чтобы он перехватывал только пакеты протокола ICMP, отправленные на определенный IP-адрес. При этом включить распечатку пакета в шестнадцатеричной системе и ASCII-формате (включая заголовок канального уровня). Количество захватываемых пакетов ограничить 3. Для генерирования пакетов возпользоваться утилитой ping.

Команда:
tcpdump -c 3 -XX ‘dst host 10.0.0.13 and ip proto \icmp’

Запустим ping с компьютера 10.0.3.10:

```
root@n4:/tmp/pycore.41679/n4.conf# ping 10.0.2.10
PING 10.0.2.10 (10.0.2.10) 56(84) bytes of data.
64 bytes from 10.0.2.10: icmp_seq=1 ttl=61 time=0.121 ms
64 bytes from 10.0.2.10: icmp_seq=2 ttl=61 time=0.178 ms
64 bytes from 10.0.2.10: icmp_seq=3 ttl=61 time=0.171 ms
64 bytes from 10.0.2.10: icmp_seq=4 ttl=61 time=0.145 ms
^C
--- 10.0.2.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3038ms
rtt min/avg/max/mdev = 0.121/0.153/0.178/0.027 ms
```

Запустим tcpdump на компьютере 10.0.2.10:

```
<1.conf# tcpdump -c 3 -XX 'dst host 10.0.2.10 and ip proto \icmp'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
10:51:03.476575 IP 10.0.3.10 > n1: ICMP echo request, id 35, seq 1, length 64
        0x0000:  0000 00aa 0005 0000 00aa 0004 0800 4500  ..............E.
        0x0010:  0054 669e 4000 3d01 bdf7 0a00 030a 0a00  .Tf.@.=.........
        0x0020:  020a 0800 b6c5 0023 0001 e79d 5960 0000  .......#....Y`..
        0x0030:  0000 3a45 0700 0000 0000 1011 1213 1415  ..:E............
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
10:51:04.478667 IP 10.0.3.10 > n1: ICMP echo request, id 35, seq 2, length 64
        0x0000:  0000 00aa 0005 0000 00aa 0004 0800 4500  ..............E.
        0x0010:  0054 66dc 4000 3d01 bdb9 0a00 030a 0a00  .Tf.@.=.........
        0x0020:  020a 0800 96bc 0023 0002 e89d 5960 0000  .......#....Y`..
        0x0030:  0000 594d 0700 0000 0000 1011 1213 1415  ..YM............
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
10:51:05.491790 IP 10.0.3.10 > n1: ICMP echo request, id 35, seq 3, length 64
        0x0000:  0000 00aa 0005 0000 00aa 0004 0800 4500  ..............E.
        0x0010:  0054 6728 4000 3d01 bd6d 0a00 030a 0a00  .Tg(@.=..m......
        0x0020:  020a 0800 4588 0023 0003 e99d 5960 0000  ....E..#....Y`..
        0x0030:  0000 a980 0700 0000 0000 1011 1213 1415  ................
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
3 packets captured
3 packets received by filter
0 packets dropped by kernel
```

4. Запустить tcpdump в режиме сохранения данных в двоичном режиме так, чтобы он перехватывал пакеты, созданные утилитой traceroute для определения маршрута к заданному в варианте узлу. Включить распечатку пакета в шестнадцатеричной системе и ASCII-формате (включая заголовок канального уровня). Количество захватываемых пакетов ограничить 7. Результат работы программы писать в файл.

Команда:

tcpdump -c 7 -XX -w out.log "ip proto \icmp or ip proto \udp"

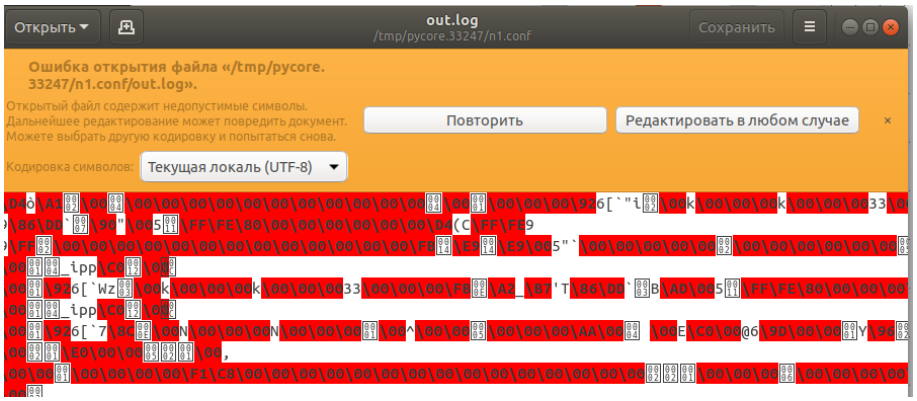
Запуск tcpdump:

```
<1.conf# tcpdump -c 7 -XX -l -w out.log "ip proto \icmp or ip proto \udp"
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 byt
es
7 packets captured
13 packets received by filter
0 packets dropped by kernel
```

Запуск traceroute:

```
root@n4:/tmp/pycore.38763/n4.conf# traceroute 10.0.2.10
traceroute to 10.0.2.10 (10.0.2.10), 30 hops max, 60 byte packets
 1  _gateway (10.0.3.1)  0.259 ms  0.028 ms  0.017 ms
 2  10.0.1.1 (10.0.1.1)  0.085 ms  0.027 ms  0.024 ms
 3  10.0.0.1 (10.0.0.1)  0.083 ms  0.034 ms  0.032 ms
 4  10.0.2.10 (10.0.2.10)  0.097 ms  0.113 ms  0.046 ms
```

Содержание файла:



Ошибка открытия, так как мы указали сохранение в двоичном формате.

5. Теперь попробуем прочитать ранее созданный файл.
tcpdump -r out.log
Результат:

```
root@n1:/tmp/pycore.33247/n1.conf# tcpdump -r out.log
reading from file out.log, link-type EN10MB (Ethernet)
15:59:35.148692 IP _gateway > 224.0.0.5: OSPFv2, Hello, length 44
15:59:37.149551 IP _gateway > 224.0.0.5: OSPFv2, Hello, length 44
15:59:38.582542 IP 10.0.3.10.49640 > n1.33443: UDP, length 32
15:59:38.582550 IP n1 > 10.0.3.10: ICMP n1 udp port 33443 unreachable, length 68
15:59:38.582575 IP 10.0.3.10.48742 > n1.33444: UDP, length 32
15:59:38.582599 IP n1 > 10.0.3.10: ICMP n1 udp port 33444 unreachable, length 68
15:59:38.582630 IP 10.0.3.10.52410 > n1.33445: UDP, length 32
```

6. Придумать три задания для фильтрации пакетов на основе протоколов ARP, TCP, UDP, ICMP

- Захватить 3 пакета протокола UDP, отправленных на определенный MAC-адрес. Отобразить данные канального уровня и включить распечатку пакета в шестнадцатеричной системе. Для генерирования пакетов воспользоваться утилитой traceroute.
  tcpdump -c 3 -e -x “ether dst 00:00:aa:00:00:01 and ip proto \udp”
  Результат:

```
root@n1:/tmp/pycore.42861/n1.conf# tcpdump -c 3 -e -x "ether dst 00:00:aa:00:0>
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
16:55:32.311457 00:00:00:aa:00:04 (oui Ethernet) > 00:00:aa:00:00:01 (oui Unknown), ethertype IPv4
 (0x0800), length 74: 10.0.3.10.44904 > n1.33443: UDP, length 32
        0x0000:  4500 003c 0d5f 0000 0111 933f 0a00 030a
        0x0010:  0a00 020a af68 82a3 0028 bf79 4041 4243
        0x0020:  4445 4647 4849 4a4b 4c4d 4e4f 5051 5253
        0x0030:  5455 5657 5859 5a5b 5c5d 5e5f
16:55:32.311580 00:00:00:aa:00:04 (oui Ethernet) > 00:00:aa:00:00:01 (oui Unknown), ethertype IPv4
 (0x0800), length 74: 10.0.3.10.43876 > n1.33444: UDP, length 32
        0x0000:  4500 003c 0d60 0000 0111 933e 0a00 030a
        0x0010:  0a00 020a ab64 82a4 0028 c37c 4041 4243
        0x0020:  4445 4647 4849 4a4b 4c4d 4e4f 5051 5253
        0x0030:  5455 5657 5859 5a5b 5c5d 5e5f
16:55:32.311607 00:00:00:aa:00:04 (oui Ethernet) > 00:00:aa:00:00:01 (oui Unknown), ethertype IPv4
 (0x0800), length 74: 10.0.3.10.51011 > n1.33445: UDP, length 32
        0x0000:  4500 003c 0d61 0000 0111 933d 0a00 030a
        0x0010:  0a00 020a c743 82a5 0028 a79c 4041 4243
        0x0020:  4445 4647 4849 4a4b 4c4d 4e4f 5051 5253
        0x0030:  5455 5657 5859 5a5b 5c5d 5e5f
3 packets captured
7 packets received by filter
0 packets dropped by kernel
root@n1:/tmp/pycore.42861/n1.conf#
```

– Перехватить 5 пакетов протокола ICMP, отправленных с определенного IP-адреса, отобразив минимум информации и в реальном времени. Для генерирования пакетов воспользоваться утилитой ping.
tcpdump -c 5 -q -l "src host 10.0.3.10 and ip proto \icmp"
Результат:

```
root@n1:/tmp/pycore.42861/n1.conf# tcpdump -c 5 -q -l "src host 10.0.3.10 and ip proto \icmp"
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:07:25.393673 IP 10.0.3.10 > n1: ICMP echo request, id 37, seq 1, length 64
17:07:26.395741 IP 10.0.3.10 > n1: ICMP echo request, id 37, seq 2, length 64
17:07:27.413648 IP 10.0.3.10 > n1: ICMP echo request, id 37, seq 3, length 64
17:07:28.437804 IP 10.0.3.10 > n1: ICMP echo request, id 37, seq 4, length 64
17:07:29.461748 IP 10.0.3.10 > n1: ICMP echo request, id 37, seq 5, length 64
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

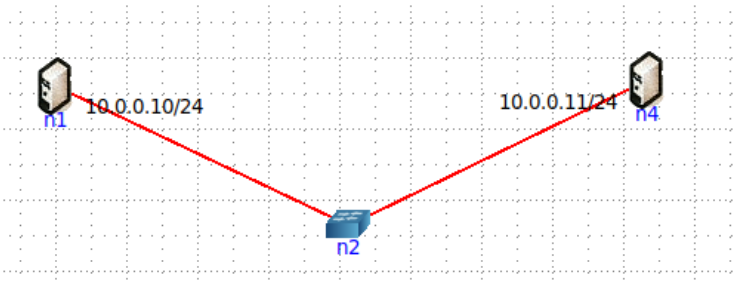
– Запустить tcpdump в режиме захвата пакетов так, чтобы он перехватывал только пакеты протокола TCP (созданные утилитой netcat), отображал данные в шестнадцатеричной системе и ASCII-формате. Количество пакетов ограничить 5.
tcpdump -c 5 -l -X "ip proto \tcp"
Результат:

```
123
543
hello
```

```
root@n1:/tmp/pycore.42861/n1.conf# tcpdump -c 5 -l -X "ip proto \tcp"
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
18:12:49.538327 IP n1.2399 > 10.0.3.10.46402: Flags [P.], seq 3327937100:3327937104, ack 28677
6492, win 510, options [nop,nop,TS val 1702954909 ecr 1628988388], length 4
        0x0000:  4500 0038 7eec 4000 4006 a2c0 0a00 020a  E..8~.@.@.......
        0x0010:  0a00 030a 095f b542 c65c 4a4c 1117 dcac  ....._.B.\JL....
        0x0020:  8018 01fe 083a 0000 0101 080a 6581 079d  .....:......e...
        0x0030:  6118 63e4 3132 330a                      a.c.123.
18:12:49.538389 IP 10.0.3.10.46402 > n1.2399: Flags [.], ack 4, win 502, options [nop,nop,TS v
al 1629021776 ecr 1702954909], length 0
        0x0000:  4500 0034 7baf 4000 3d06 a901 0a00 030a  E..4{.@.=.......
        0x0010:  0a00 020a b542 095f 1117 dcac c65c 4a50  .....B._.....\JP
        0x0020:  8010 01f6 ea19 0000 0101 080a 6118 e650  ............a..P
        0x0030:  6581 079d                                e...
18:12:51.853878 IP n1.2399 > 10.0.3.10.46402: Flags [P.], seq 4:8, ack 1, win 510, options [no
p,nop,TS val 1702957225 ecr 1629021776], length 4
        0x0000:  4500 0038 7eed 4000 4006 a2bf 0a00 020a  E..8~.@.@.......
        0x0010:  0a00 030a 095f b542 c65c 4a50 1117 dcac  ....._.B.\JP....
        0x0020:  8018 01fe 78bb 0000 0101 080a 6581 10a9  ....x.......e...
        0x0030:  6118 e650 3534 330a                      a..P543.
18:12:51.853961 IP 10.0.3.10.46402 > n1.2399: Flags [.], ack 8, win 502, options [nop,nop,TS v
al 1629024092 ecr 1702957225], length 0
        0x0000:  4500 0034 7bb0 4000 3d06 a900 0a00 030a  E..4{.@.=.......
        0x0010:  0a00 020a b542 095f 1117 dcac c65c 4a54  .....B._.....\JT
        0x0020:  8010 01f6 d7fd 0000 0101 080a 6118 ef5c  ............a..\
        0x0030:  6581 10a9                                e...
18:12:55.085988 IP n1.2399 > 10.0.3.10.46402: Flags [P.], seq 8:14, ack 1, win 510, options [n
op,nop,TS val 1702960457 ecr 1629024092], length 6
        0x0000:  4500 003a 7eee 4000 4006 a2bc 0a00 020a  E..:~.@.@.......
        0x0010:  0a00 030a 095f b542 c65c 4a54 1117 dcac  ....._.B.\JT....
        0x0020:  8018 01fe 876b 0000 0101 080a 6581 1d49  .....k......e..I
        0x0030:  6118 ef5c 6865 6c6c 6f0a                 a..\hello.
5 packets captured
6 packets received by filter
0 packets dropped by kernel
```

wireshark

Сеть:



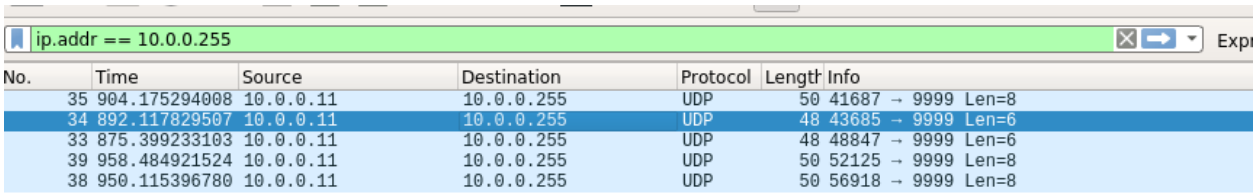
1. Захватить 5-7 пакетов широковещательного трафика (фильтр по IP-адресу). Результат сохранить в текстовый файл.

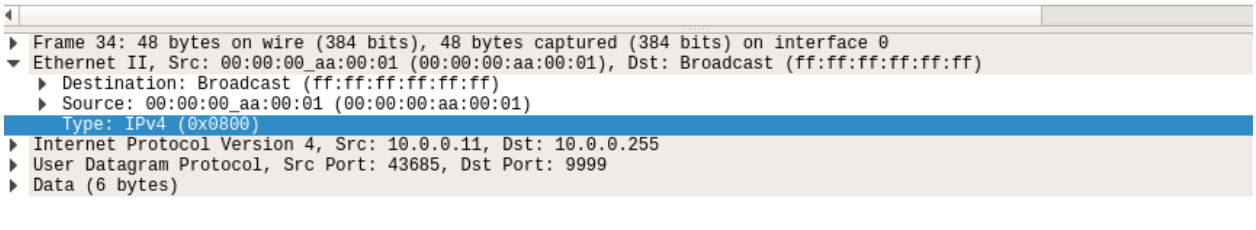
Фильтр:
ip.addr == 10.0.0.255

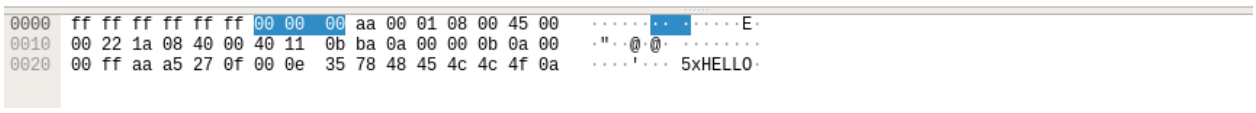
Команда:

```
<4.conf# echo "HELLO" | socat - UDP4-DATAGRAM:10.0.0.255:9999,broadcast
<" | socat - UDP4-DATAGRAM:10.0.0.255:9999,broadcast
root@n4:/tmp/pycore.39467/n4.conf# echo "HELLOxx" | socat - UDP4-DATAGRAM:10.0>
root@n4:/tmp/pycore.39467/n4.conf# echo "HELLOx1" | socat - UDP4-DATAGRAM:10.0.0.255:9999,broadcast
root@n4:/tmp/pycore.39467/n4.conf# echo "HELLOx2" | socat - UDP4-DATAGRAM:10.0.0.255:9999,broadcast
root@n4:/tmp/pycore.39467/n4.conf#
```

Whireshark:







2. Захватить 3-4 пакета ICMP, полученных от определенного узла. Для генерирования пакетов возпользоваться утилитой ping. Результат сохранить в текстовый файл.

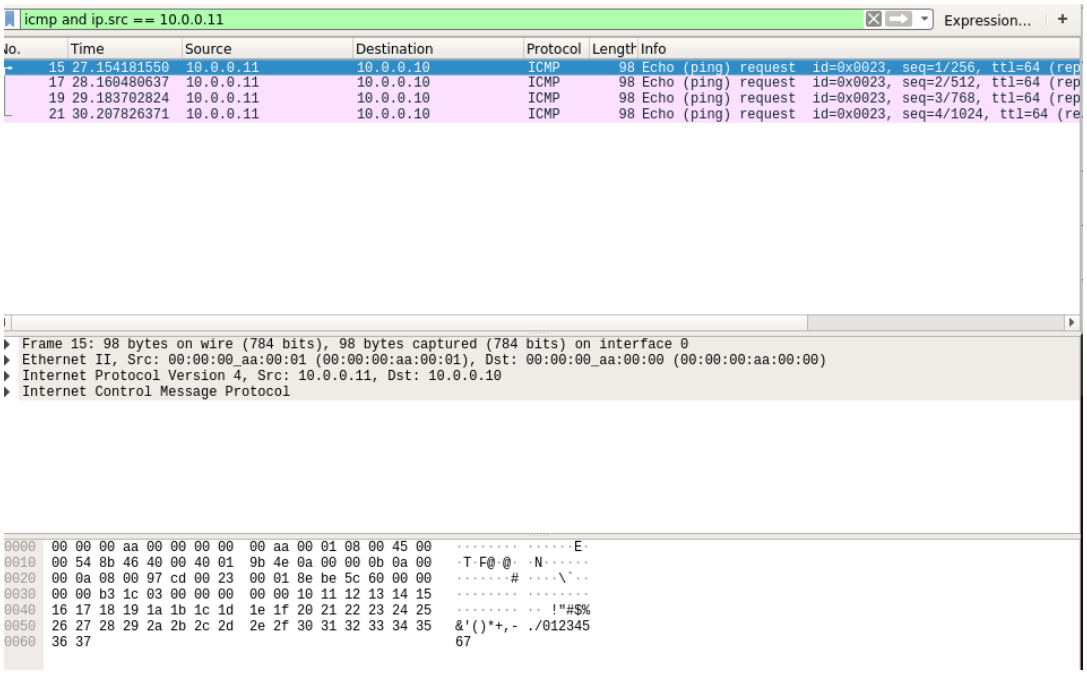
Фильтр:

icmp and ip.addr == 10.0.0.11

Запущу ping на компьютере 10.0.0.11:

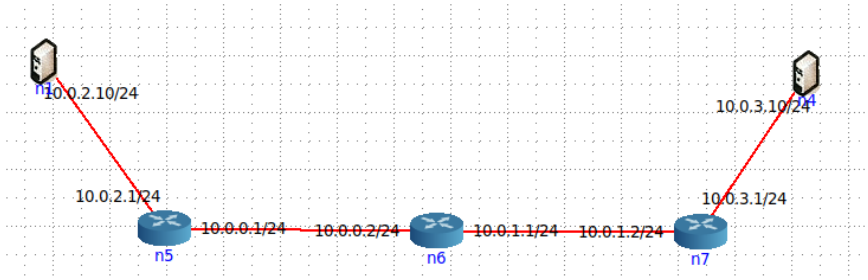
```
root@n4:/tmp/pycore.39467/n4.conf# ping 10.0.0.10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data.
64 bytes from 10.0.0.10: icmp_seq=1 ttl=64 time=0.064 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=64 time=0.095 ms
64 bytes from 10.0.0.10: icmp_seq=3 ttl=64 time=0.101 ms
64 bytes from 10.0.0.10: icmp_seq=4 ttl=64 time=0.099 ms
^C
--- 10.0.0.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3053ms
rtt min/avg/max/mdev = 0.064/0.089/0.101/0.018 ms
```

Whireshark на компьютере 10.0.0.10:



3. Перехватить пакеты, созданные утилитой traceroute для определения маршрута к заданному в варианте узлу. По результатам построить диаграмму Flow Graph. Диаграмму сохранить либо в виде текстового файла либо в виде изображения.

Сеть:

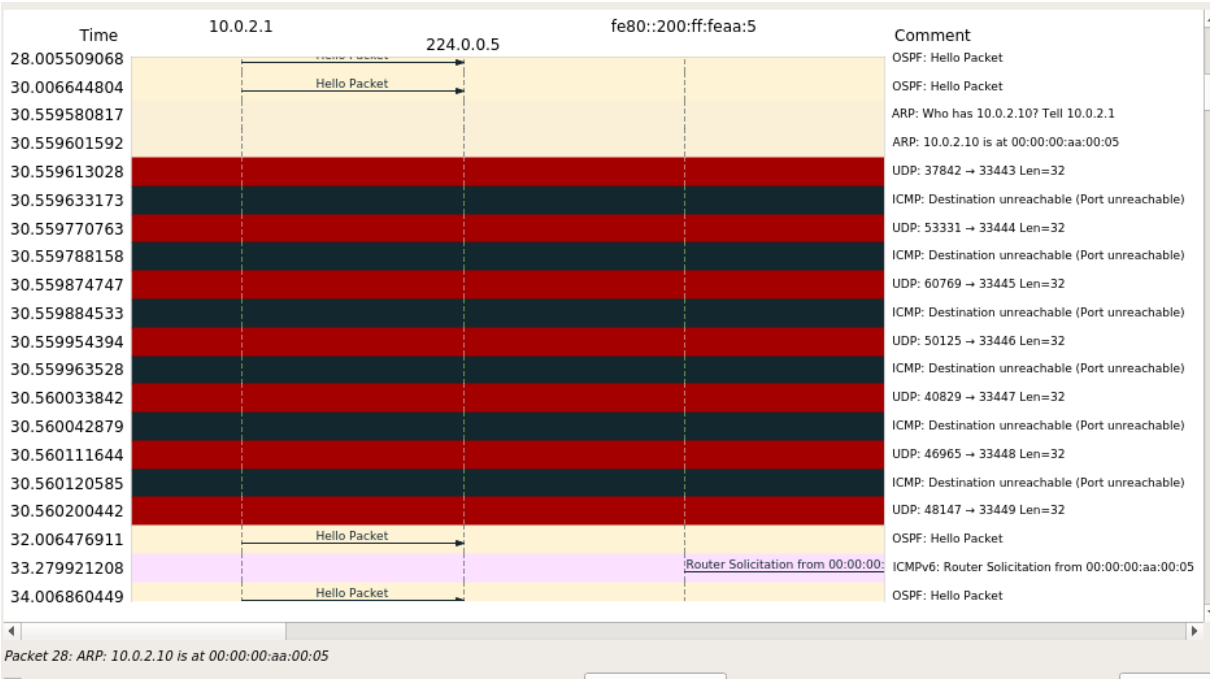


Запущу traceroute на хосте 10.0.3.10 и wireshark на хосте 10.0.2.10:

```
root@n4:/tmp/pycore.38763/n4.conf# traceroute 10.0.2.10
traceroute to 10.0.2.10 (10.0.2.10), 30 hops max, 60 byte packets
 1  _gateway (10.0.3.1)  0.259 ms  0.028 ms  0.017 ms
 2  10.0.1.1 (10.0.1.1)  0.085 ms  0.027 ms  0.024 ms
 3  10.0.0.1 (10.0.0.1)  0.083 ms  0.034 ms  0.032 ms
 4  10.0.2.10 (10.0.2.10)  0.097 ms  0.113 ms  0.046 ms
root@n4:/tmp/pycore.38763/n4.conf#
```

28	30.559601592	00:00:00_aa:00:05	00:00:00_aa:00:04	ARP	42 10.0.2.10 is at 00:00:00:aa:00:05
29	30.559613028	10.0.3.10	10.0.2.10	UDP	74 37842 → 33443 Len=32
30	30.559633173	10.0.2.10	10.0.3.10	ICMP	102 Destination unreachable (Port unreachable)
31	30.559770763	10.0.3.10	10.0.2.10	UDP	74 53331 → 33444 Len=32
32	30.559788158	10.0.2.10	10.0.3.10	ICMP	102 Destination unreachable (Port unreachable)
33	30.559874747	10.0.3.10	10.0.2.10	UDP	74 60769 → 33445 Len=32
34	30.559884533	10.0.2.10	10.0.3.10	ICMP	102 Destination unreachable (Port unreachable)
35	30.559954394	10.0.3.10	10.0.2.10	UDP	74 50125 → 33446 Len=32
36	30.559963528	10.0.2.10	10.0.3.10	ICMP	102 Destination unreachable (Port unreachable)
37	30.560033842	10.0.3.10	10.0.2.10	UDP	74 40829 → 33447 Len=32
38	30.560042879	10.0.2.10	10.0.3.10	ICMP	102 Destination unreachable (Port unreachable)
39	30.560111644	10.0.3.10	10.0.2.10	UDP	74 46965 → 33448 Len=32
40	30.560120585	10.0.2.10	10.0.3.10	ICMP	102 Destination unreachable (Port unreachable)
41	30.560200442	10.0.3.10	10.0.2.10	UDP	74 48147 → 33449 Len=32
42	32.006476911	10.0.2.1	224.0.0.5	OSPF	78 Hello Packet

FlowGraph:

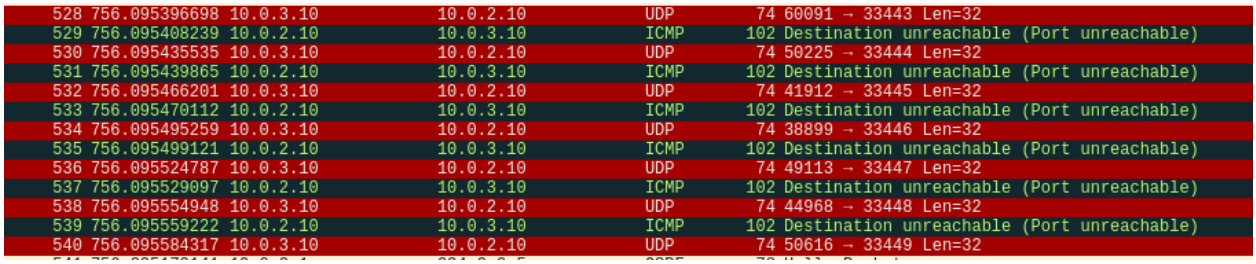


4. Прочесть файл, созданный программой tcpdump. Сравнить с тем, что было получено утилитой wireshark.

tcpdump:

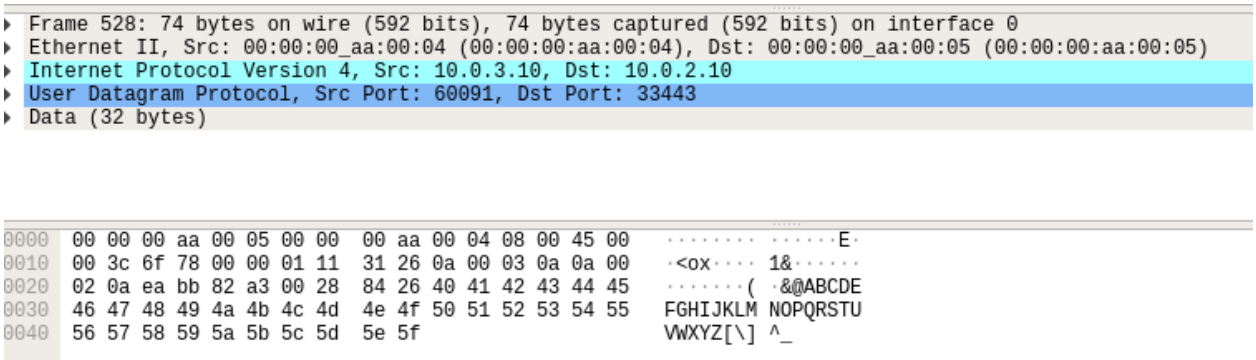
```
<1.conf# tcpdump -c 7 -XX -l -w out.log "ip proto \icmp or ip proto \udp"
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 byt
es
7 packets captured
13 packets received by filter
0 packets dropped by kernel
root@n1:/tmp/pycore.38763/n1.conf# tcpdump -r out.log
reading from file out.log, link-type EN10MB (Ethernet)
20:03:11.848977 IP 10.0.3.10.60091 > n1.33443: UDP, length 32
20:03:11.848984 IP n1 > 10.0.3.10: ICMP n1 udp port 33443 unreachable, length 68
20:03:11.849013 IP 10.0.3.10.50225 > n1.33444: UDP, length 32
20:03:11.849016 IP n1 > 10.0.3.10: ICMP n1 udp port 33444 unreachable, length 68
20:03:11.849044 IP 10.0.3.10.41912 > n1.33445: UDP, length 32
20:03:11.849047 IP n1 > 10.0.3.10: ICMP n1 udp port 33445 unreachable, length 68
20:03:11.849073 IP 10.0.3.10.38899 > n1.33446: UDP, length 32
```

wireshark:



```
528 756.095396698 10.0.3.10  10.0.2.10  UDP   74 60091 → 33443 Len=32
529 756.095408239 10.0.2.10  10.0.3.10  ICMP 102 Destination unreachable (Port unreachable)
530 756.095435535 10.0.3.10  10.0.2.10  UDP   74 50225 → 33444 Len=32
531 756.095439865 10.0.2.10  10.0.3.10  ICMP 102 Destination unreachable (Port unreachable)
532 756.095466201 10.0.3.10  10.0.2.10  UDP   74 41912 → 33445 Len=32
533 756.095470112 10.0.2.10  10.0.3.10  ICMP 102 Destination unreachable (Port unreachable)
534 756.095495259 10.0.3.10  10.0.2.10  UDP   74 38899 → 33446 Len=32
535 756.095499121 10.0.2.10  10.0.3.10  ICMP 102 Destination unreachable (Port unreachable)
536 756.095524787 10.0.3.10  10.0.2.10  UDP   74 49113 → 33447 Len=32
537 756.095529097 10.0.2.10  10.0.3.10  ICMP 102 Destination unreachable (Port unreachable)
538 756.095554948 10.0.3.10  10.0.2.10  UDP   74 44968 → 33448 Len=32
539 756.095559222 10.0.2.10  10.0.3.10  ICMP 102 Destination unreachable (Port unreachable)
540 756.095584317 10.0.3.10  10.0.2.10  UDP   74 50616 → 33449 Len=32
```

Информация схожа, но в wireshark мы так же можем более подробно изучить данные пересылаемых пакетов:



```
▸ Frame 528: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_aa:00:04 (00:00:00:aa:00:04), Dst: 00:00:00_aa:00:05 (00:00:00:aa:00:05)
▸ Internet Protocol Version 4, Src: 10.0.3.10, Dst: 10.0.2.10
▸ User Datagram Protocol, Src Port: 60091, Dst Port: 33443
▸ Data (32 bytes)
```



```
0000  00 00 00 aa 00 05 00 00  00 aa 00 04 08 00 45 00   ........ ......E·
0010  00 3c 6f 78 00 00 01 11  31 26 0a 00 03 0a 0a 00   ·<ox···· 1&·····
0020  02 0a ea bb 82 a3 00 28  84 26 40 41 42 43 44 45   ·......( ·&@ABCDE
0030  46 47 48 49 4a 4b 4c 4d  4e 4f 50 51 52 53 54 55   FGHIJKLM NOPQRSTU
0040  56 57 58 59 5a 5b 5c 5d  5e 5f                     VWXYZ[\] ^_
```

Тогда как в записанным tcmdump файле только та информация, которую мы получили, применив фильтр.