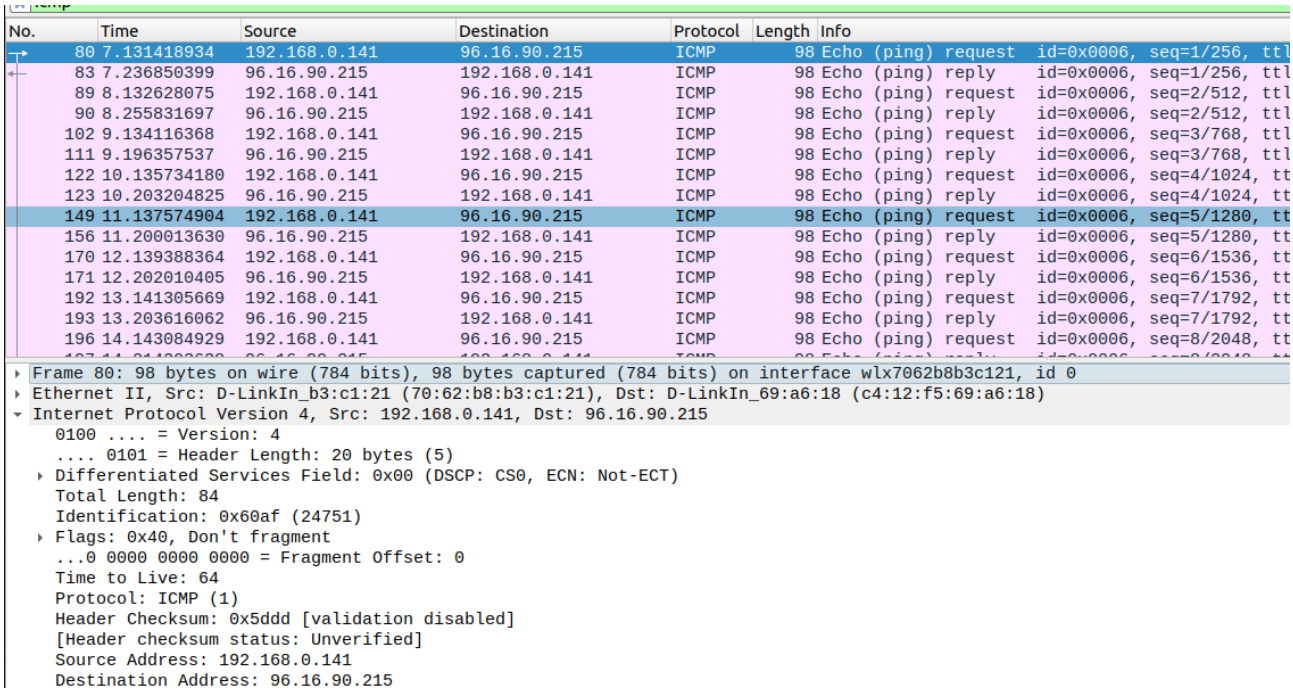
Курбатов Ярослав, РПО МКН

1. Wireshark: ICMP

в unix в ping опция для задания кол-ва запросов -n

1. Ping



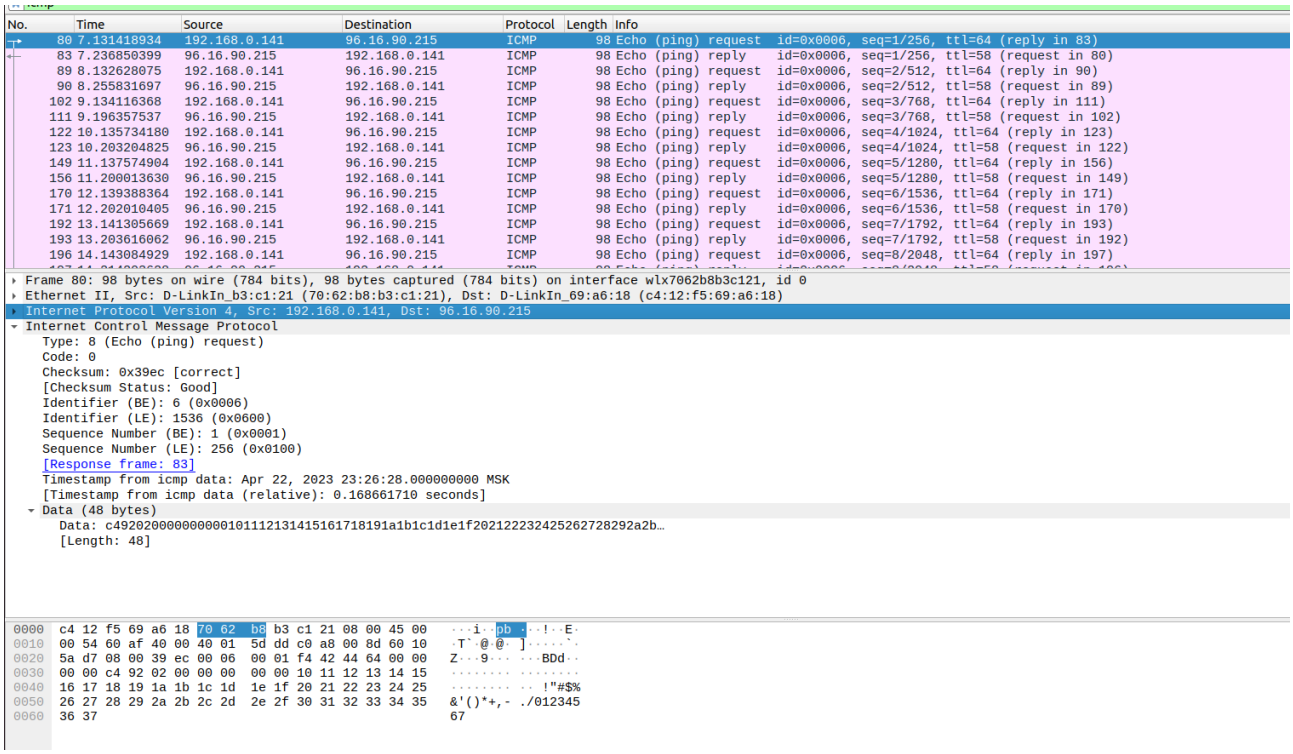
No.	Time	Source	Destination	Protocol	Length	Info
80	7.131418934	192.168.0.141	96.16.90.215	ICMP	98	Echo (ping) request id=0x0006, seq=1/256, ttl
83	7.236850399	96.16.90.215	192.168.0.141	ICMP	98	Echo (ping) reply id=0x0006, seq=1/256, ttl
89	8.132628075	192.168.0.141	96.16.90.215	ICMP	98	Echo (ping) request id=0x0006, seq=2/512, ttl
90	8.255831697	96.16.90.215	192.168.0.141	ICMP	98	Echo (ping) reply id=0x0006, seq=2/512, ttl
102	9.134116368	192.168.0.141	96.16.90.215	ICMP	98	Echo (ping) request id=0x0006, seq=3/768, ttl
111	9.196357537	96.16.90.215	192.168.0.141	ICMP	98	Echo (ping) reply id=0x0006, seq=3/768, ttl
122	10.135734180	192.168.0.141	96.16.90.215	ICMP	98	Echo (ping) request id=0x0006, seq=4/1024, tt
123	10.203204825	96.16.90.215	192.168.0.141	ICMP	98	Echo (ping) reply id=0x0006, seq=4/1024, tt
149	11.137574904	192.168.0.141	96.16.90.215	ICMP	98	Echo (ping) request id=0x0006, seq=5/1280, tt
156	11.200013630	96.16.90.215	192.168.0.141	ICMP	98	Echo (ping) reply id=0x0006, seq=5/1280, tt
170	12.139388364	192.168.0.141	96.16.90.215	ICMP	98	Echo (ping) request id=0x0006, seq=6/1536, tt
171	12.202010405	96.16.90.215	192.168.0.141	ICMP	98	Echo (ping) reply id=0x0006, seq=6/1536, tt
192	13.141305669	192.168.0.141	96.16.90.215	ICMP	98	Echo (ping) request id=0x0006, seq=7/1792, tt
193	13.203616062	96.16.90.215	192.168.0.141	ICMP	98	Echo (ping) reply id=0x0006, seq=7/1792, tt
196	14.143084929	192.168.0.141	96.16.90.215	ICMP	98	Echo (ping) request id=0x0006, seq=8/2048, tt

```
Frame 80: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface wlx7062b8b3c121, id 0
Ethernet II, Src: D-LinkIn_b3:c1:21 (70:62:b8:b3:c1:21), Dst: D-LinkIn_69:a6:18 (c4:12:f5:69:a6:18)
Internet Protocol Version 4, Src: 192.168.0.141, Dst: 96.16.90.215
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x60af (24751)
  ▸ Flags: 0x40, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0x5ddd [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.0.141
    Destination Address: 96.16.90.215
```

1) IP-адрес моего хоста -- 192.168.0.141. IP-адрес назначения – 96.16.90.215 (см. описание IP-пакета)

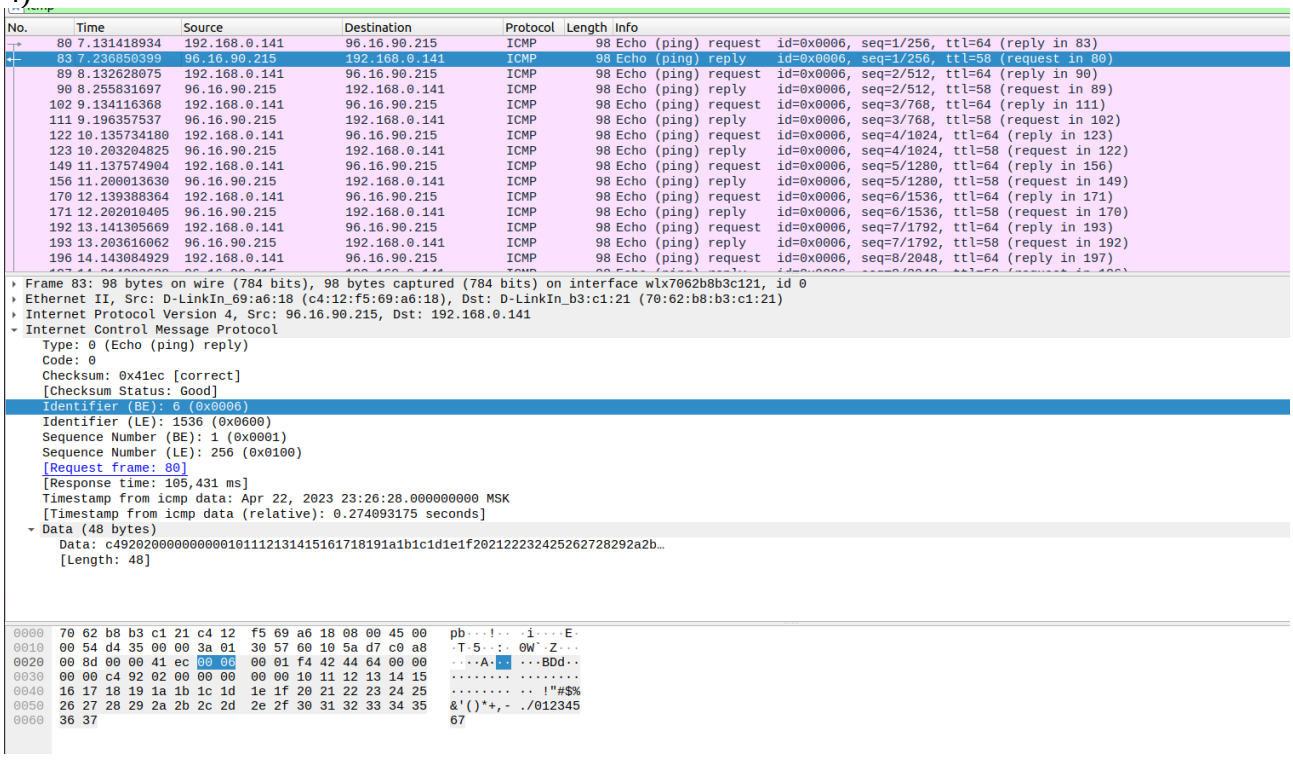
2) Протокол ICMP (хотя и работает поверх протокола IP) по сути находится на 3-ем уровне модели ISO/OSI, под транспортным уровнем, поэтому понятия порта тут нет.

3)



ICMP-тип первого запроса – echo (значение 8 в поле Type). Значение подтипа – 0 (единственный возможный при type=8). Также в пакете есть поле identifier, sequence number, checksum, timestamp. Размер identifier, checksum, SN – 2 байта.

4)

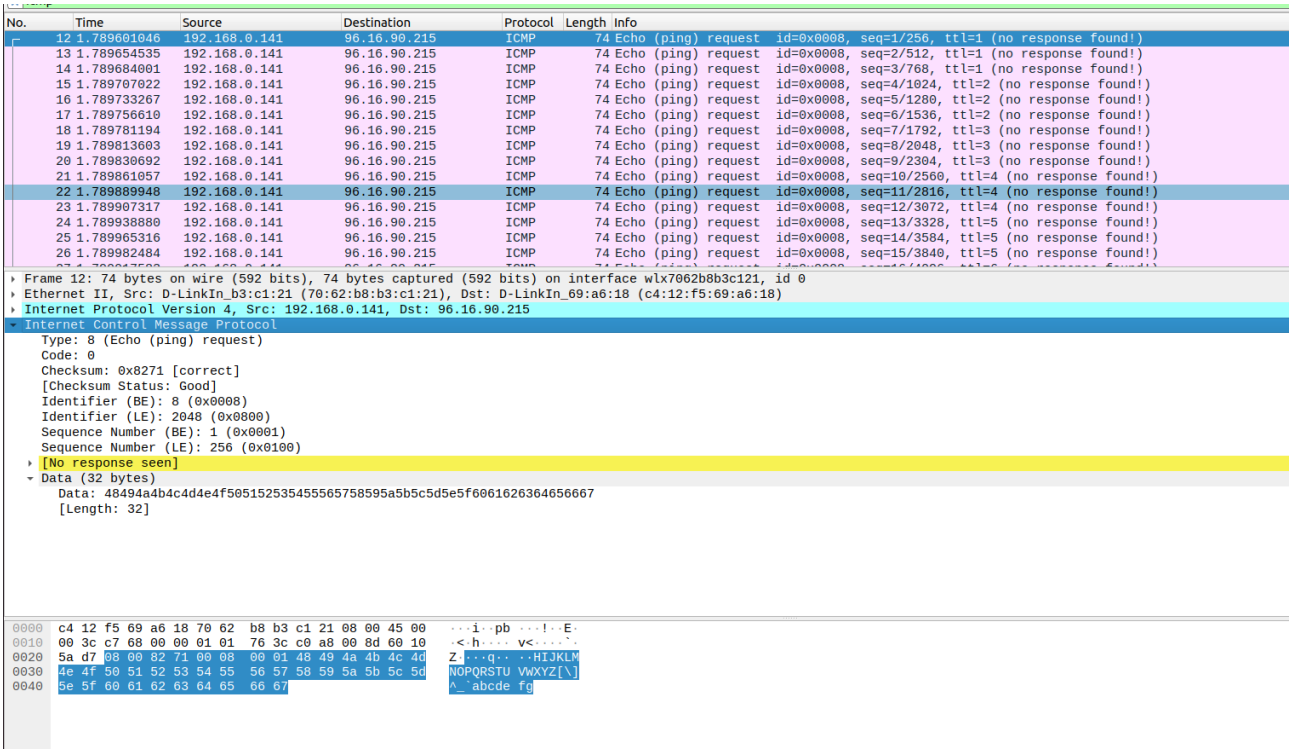


IC<P-тип первого ответа – echo reply (значение 0 в поле Type). Значение подтипа – 0 (единственный возможный при type=0). Пакет имеет те же поля, что и запрос. SN, checksum, identifier занимают 2 байта.

2. Traceroute

traceroute -I nsa.gov (-I чтобы использовать ICMP)

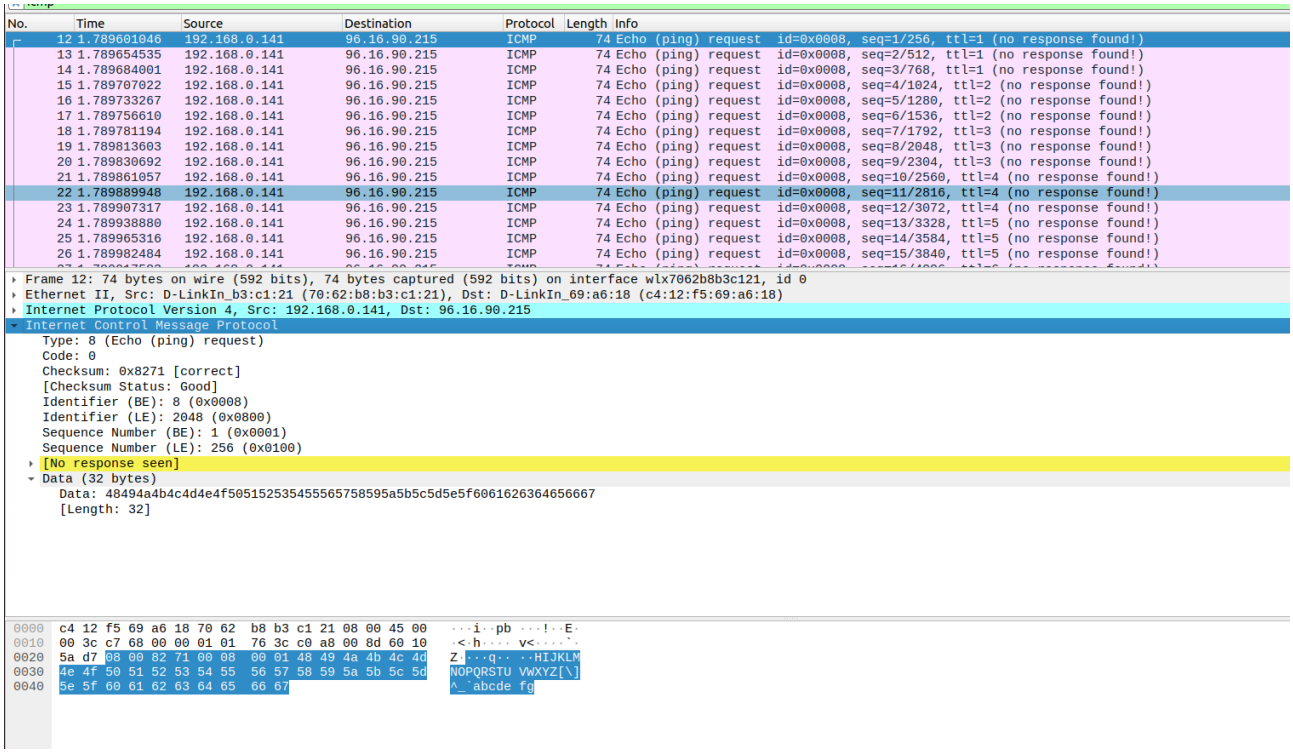
1)



можно видеть, что сам ICMP-пакет ничем не отличается от ICMP-пакета из первого задания. Есть различие на уровне IP-протокола (TTL=1).

2)

Ответ с сообщением об ошибке:

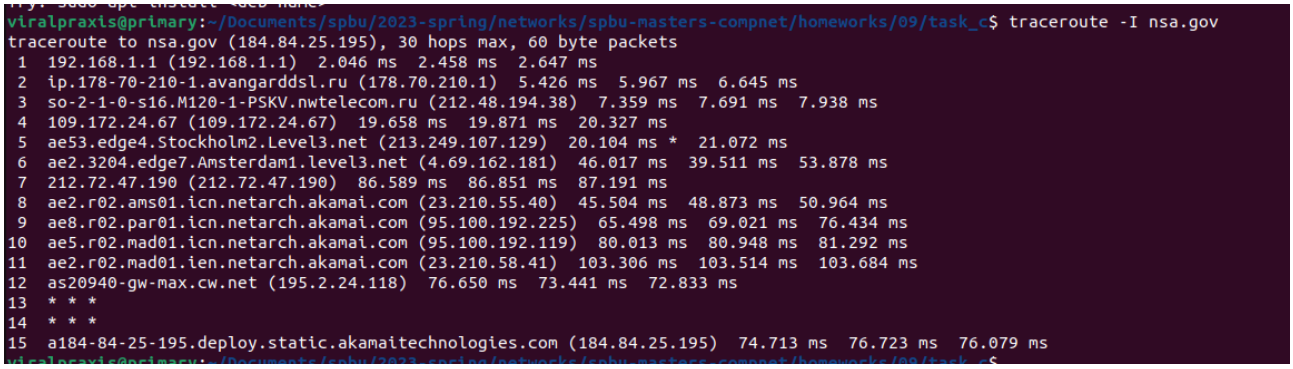


видно, что тип ICMP-пакета – 1 – TTL Exceeded. Кроме того, пакет содержит данные ICMP-запроса. Также среди полей пакета – только подтип и checksum.

3)

последние 3 пакета, которые пришли пакета на исходный (мой) хост, это ICMP echo reply от искомого хоста (nsa.gov, 184.84.25.195). Собственно, это последние пакеты, потому что ICMP-пакеты дошли до искомого хоста и он ответил на 3 наших запроса.

4)



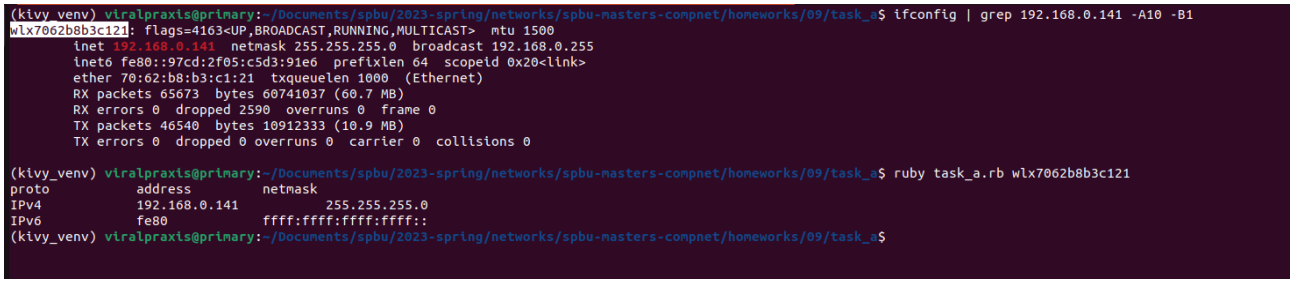
относительно большная задержка между ae53.edge4.Stockholm2.Level3.net (213.249.107.129) и ae2.3204.edge7.Amsterdam1.level3.net (213.249.107.129) (видимо, Стокгольм и Амстердам)

Программирование

Ruby MRI ~> 3.0.0

Задание A

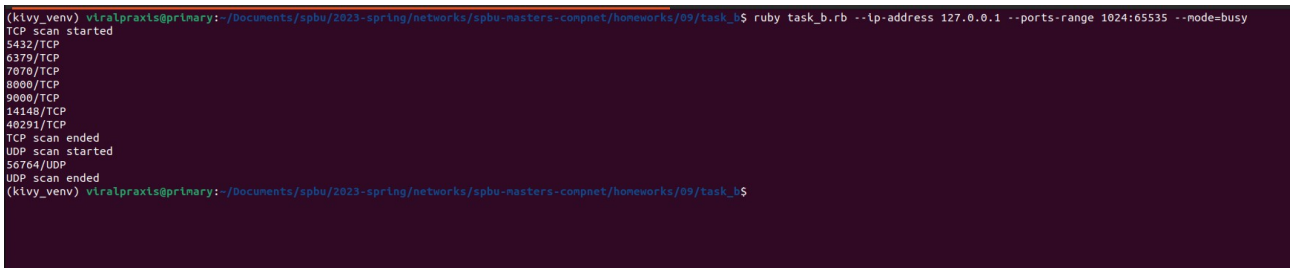
ruby task_a/task_a.rb <network-interface-identifier>



Задание B

ruby task_b/task_b.rb –ip-address=<...> --ports-range=<...> --mode=<...>

--ip-address: IP-адрес для скана
--ports-range: Отрезок портов для скана в формате start:end
--mode: open или busy, busy – занятые порты, open – свободные (по поводу работы с UDP см. Комментарии в коде)



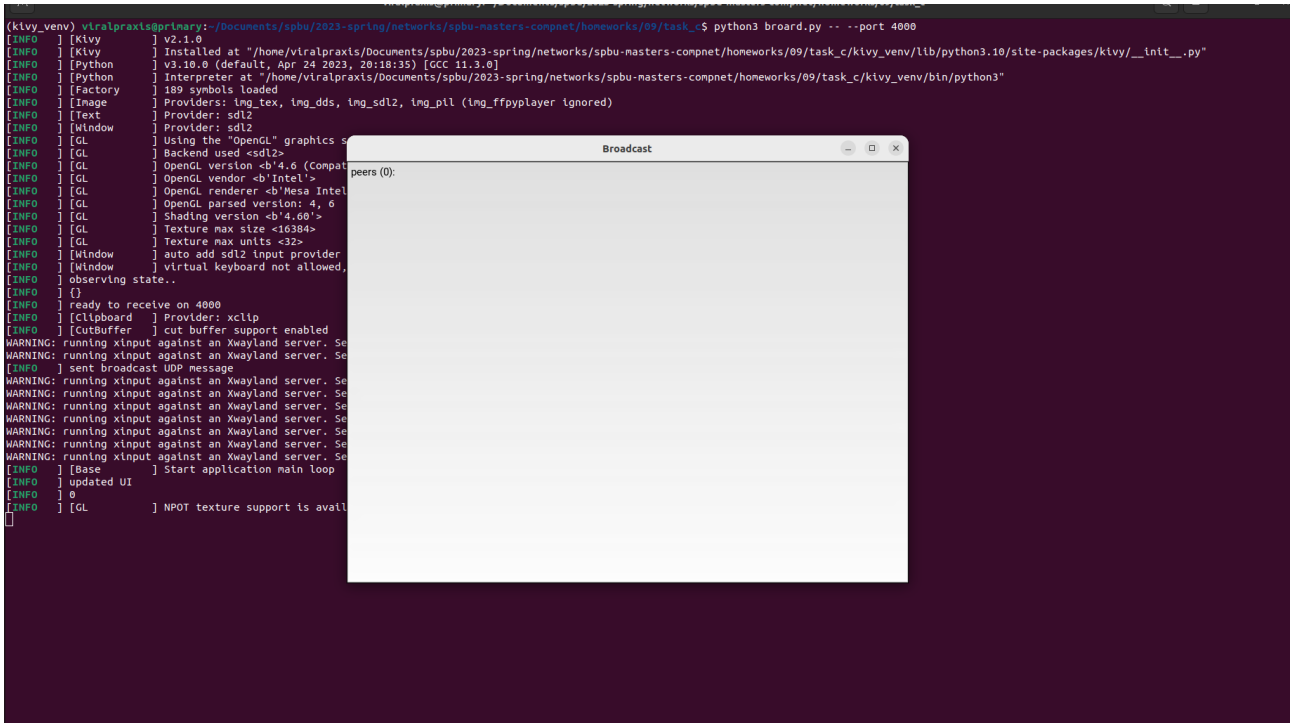
Задание C

python 3.10.0
зависимости: task_c/requirements.txt (сгенерировано pip freeze)

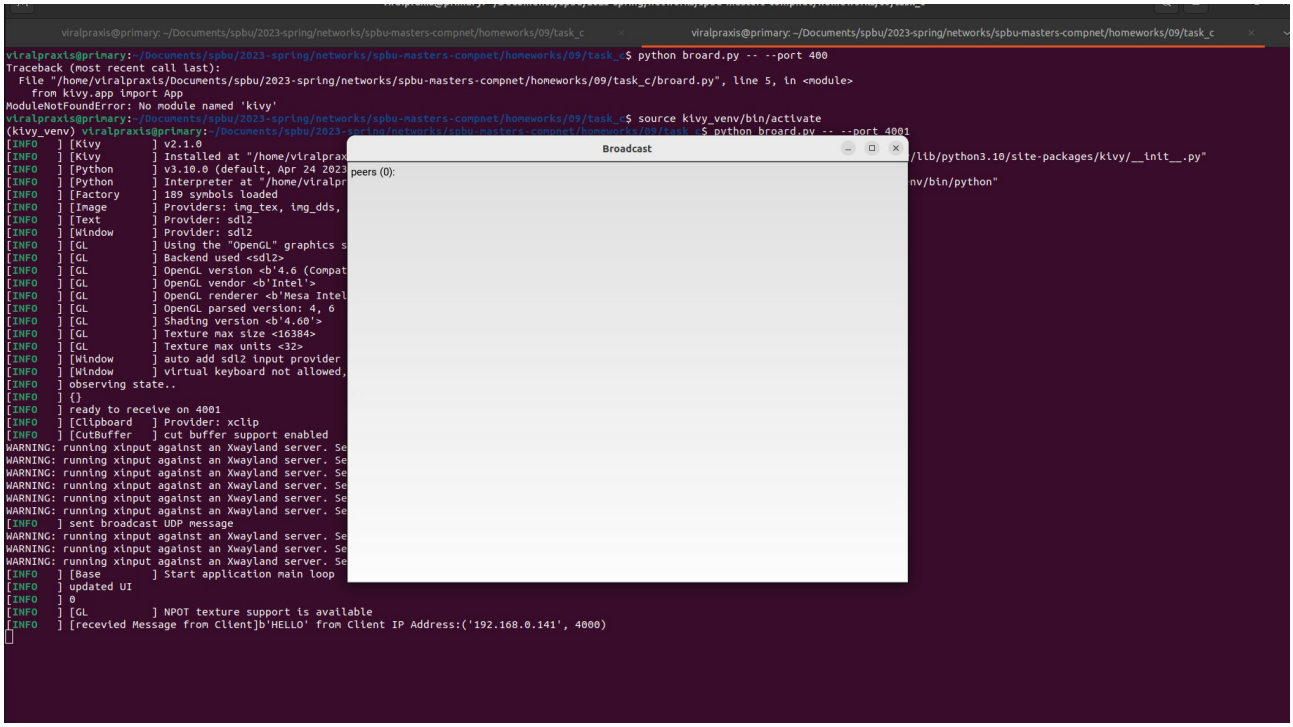
python task_c/broad.py -- –port <port> --broadcast-period <float-seconds> --inactive-peer-period <float-seconds>

Параметры:
port: число, порт на который биндится приложение
broadcast-period: float, кол-во секунд, раз в которое делается бродкаст
inactive-peer-period: float, кол-во секунд такое, что если в течение большего периода не получены UDP-пакеты от хоста, он считается неактивным

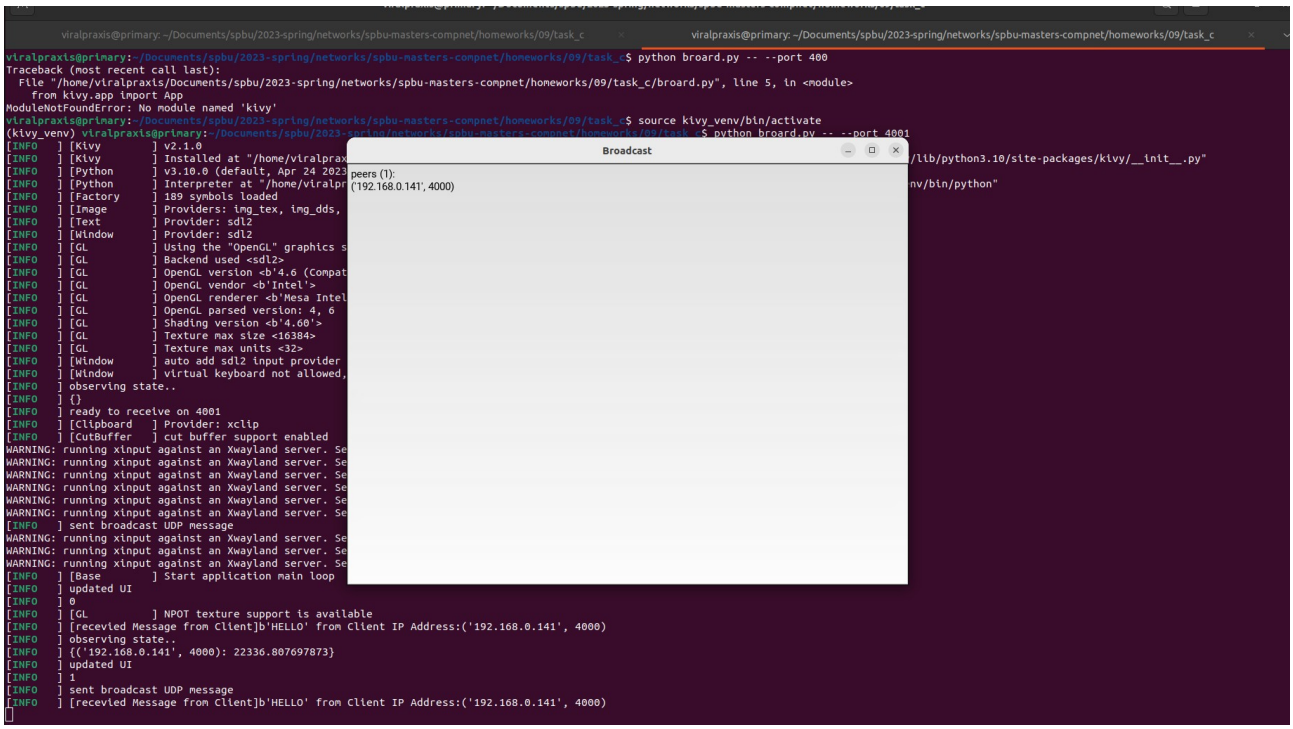
Запустим инстанс приложения на порту 4000 (python3 broard.py -- --port 4000):



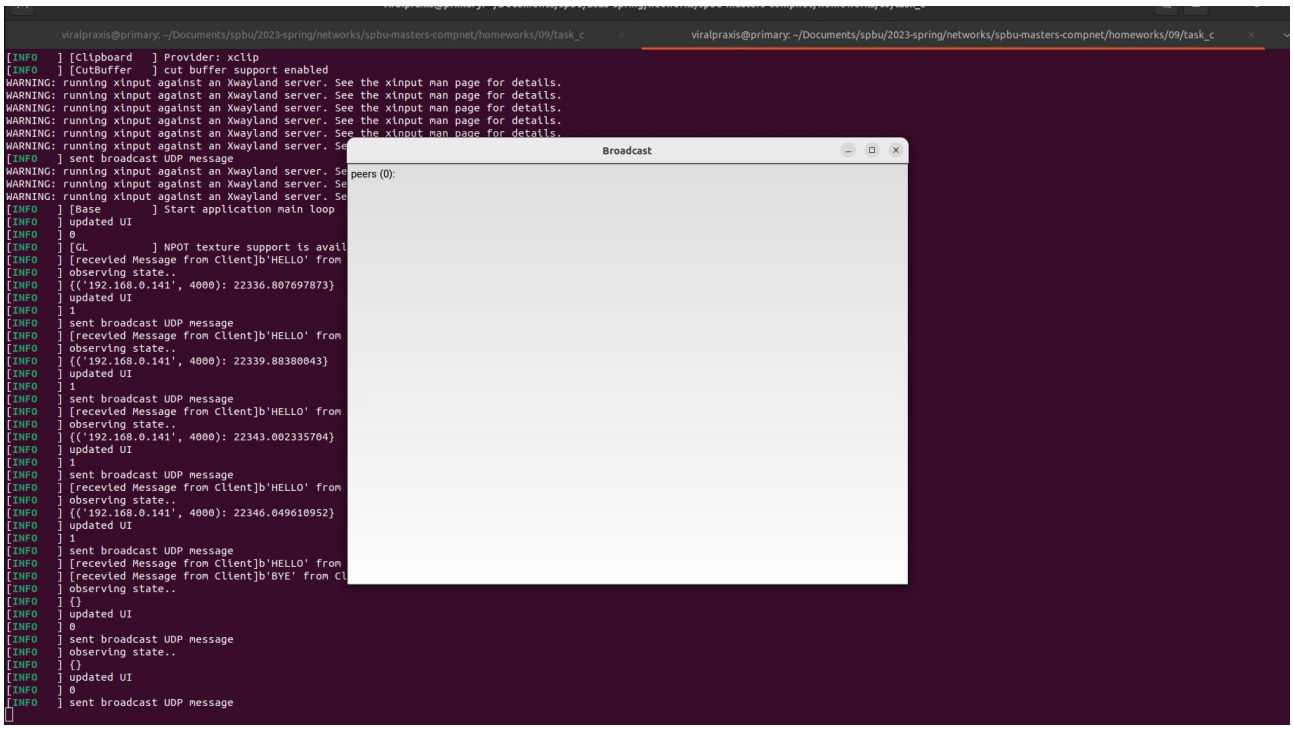
Запустим еще один инстанс на порту 4001 (python3 broard.py -- --port 4001):



Через какое-то время второе приложение получит broadcast-пакет от первого и добавит его в список:



Теперь остановим первое приложение. Перед завершением работы оно пошлет пакет, сигнализирующий о завершении работы. Второе приложение пометит первое как неактивное и уберет его из UI:



То же самое произодет, если первое приложение не будет отсылать пакет перед остановкой, через ~12 секунд (дефолтное значение) второе приложение пометит первое как неактивное.