

Санкт-Петербургский Политехнический Университет Петра Великого  
Институт компьютерных наук и технологий

**Кафедра компьютерных систем и программных технологий**

**Отчёт по лабораторной работе №1**

**Дисциплина:** Защита информации

**Тема:** Исследование сетевого трафика

Выполнил студент группы 43501/3

\_\_\_\_\_ Круминьш Д.В.  
(подпись)

Преподаватель

\_\_\_\_\_ Новопашенный А.Г.  
(подпись)

Санкт-Петербург  
2017 г.

# Содержание

<b>1</b>	<b>Лабораторная работа №1</b>	<b>2</b>
1.1	Цель работы . . . . .	2
1.2	Программа работы . . . . .	2
1.3	Конфигурация сети . . . . .	3
1.4	Ход работы . . . . .	4
14.1	Утилита ping . . . . .	4
14.2	Утилита tracert . . . . .	7
14.3	Протокол ARP . . . . .	10
14.4	Протокол ICMP . . . . .	12
14.5	Протокол UDP . . . . .	14
14.6	Протокол TCP . . . . .	14
1.5	Вывод . . . . .	19
	Приложение 1 . . . . .	20
	Приложение 2 . . . . .	20
	Приложение 3 . . . . .	21
	Приложение 4 . . . . .	23
	Приложение 5 . . . . .	24

# Лабораторная работа №1

## 1.1 Цель работы

Получение навыков по исследованию сетевого трафика.

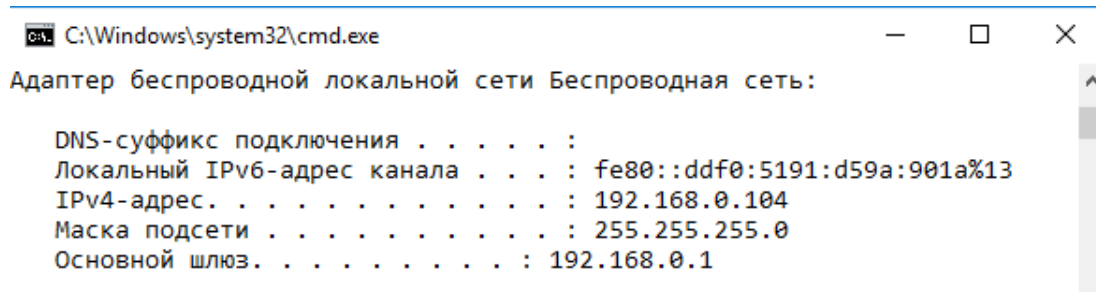
## 1.2 Программа работы

При помощи анализатора сетевого трафика Wireshark продемонстрировать в сети работу, следующих протоколов и утилит:

1. Утилиты ping:
  - без фрагментации,
  - с фрагментацией.
2. Утилиты tracer;
3. Протокола ARP:
  - запрос,
  - ответ.
4. Протокола ICMP:
  - пронаблюдать ошибку типа 3.
5. Протокола UDP:
  - попытка отправить udp пакет на несуществующий порт.
6. Протокола TCP:
  - установка соединения,
  - разрыв соединения,
  - попытка соединения на отсутствующий порт.

## 1.3 Конфигурация сети

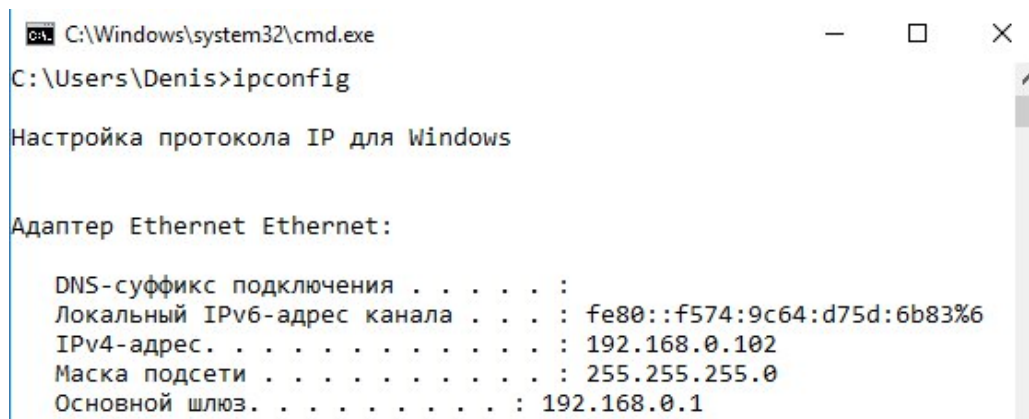
Опыты проводились используя два ПК, конфигурации которых представлены ниже. ПК находились в одной сети.



```
C:\Windows\system32\cmd.exe
Адаптер беспроводной локальной сети Беспроводная сеть:

DNS-суффикс подключения . . . . . :
Локальный IPv6-адрес канала . . . : fe80::ddf0:5191:d59a:901a%13
IPv4-адрес. . . . . : 192.168.0.104
Маска подсети . . . . . : 255.255.255.0
Основной шлюз. . . . . : 192.168.0.1
```

Рис. 1.1: Конфигурация ПК 1



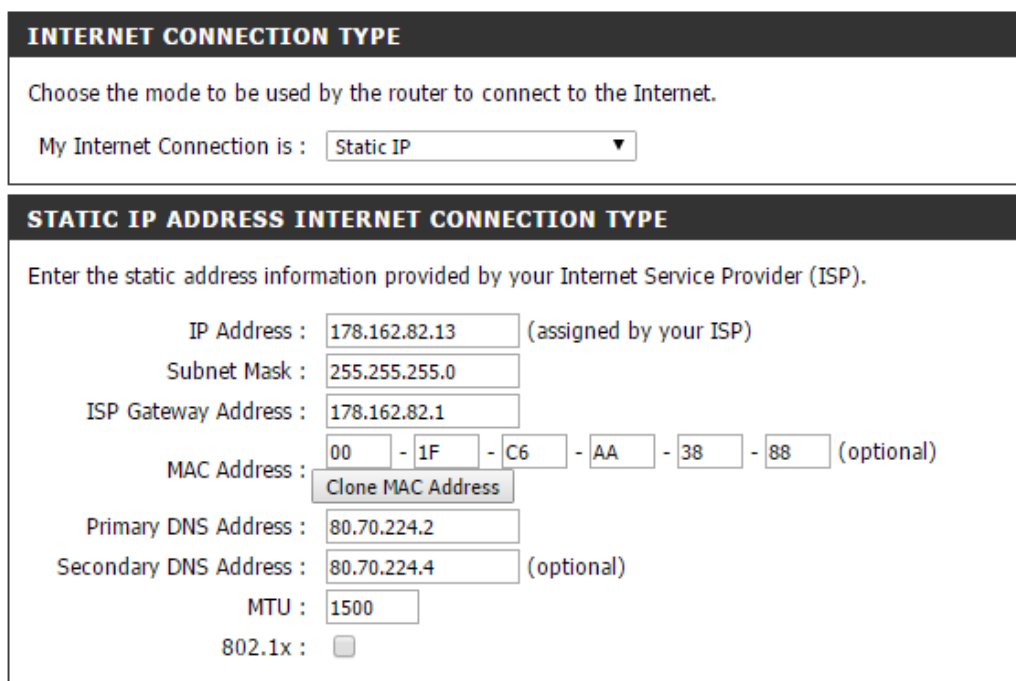
```
C:\Windows\system32\cmd.exe
C:\Users\Denis>ipconfig

Настройка протокола IP для Windows

Адаптер Ethernet Ethernet:

DNS-суффикс подключения . . . . . :
Локальный IPv6-адрес канала . . . : fe80::f574:9c64:d75d:6b83%6
IPv4-адрес. . . . . : 192.168.0.102
Маска подсети . . . . . : 255.255.255.0
Основной шлюз. . . . . : 192.168.0.1
```

Рис. 1.2: Конфигурация ПК 2



**INTERNET CONNECTION TYPE**

Choose the mode to be used by the router to connect to the Internet.

My Internet Connection is :

**STATIC IP ADDRESS INTERNET CONNECTION TYPE**

Enter the static address information provided by your Internet Service Provider (ISP).

IP Address :  (assigned by your ISP)

Subnet Mask :

ISP Gateway Address :

MAC Address :  -  -  -  -  -  (optional)

Primary DNS Address :

Secondary DNS Address :  (optional)

MTU :

802.1x : ☐

Рис. 1.3: WAN настройки роутера

# 1.4 Ход работы

## 1.4.1 Утилита ping

Утилита Ping отправляет эхо-запрос ICMP, после чего, в случае успеха должен прийти симметричный эхо-ответ ICMP. Если пакет не пришел за некоторое TTL, то удаленный сервер считается недостижимым. По умолчанию производится четыре попытки.

### Ping без фрагментации

Трафик утилиты Ping со стандартными параметрами (bytes = 32, TTL = 128):

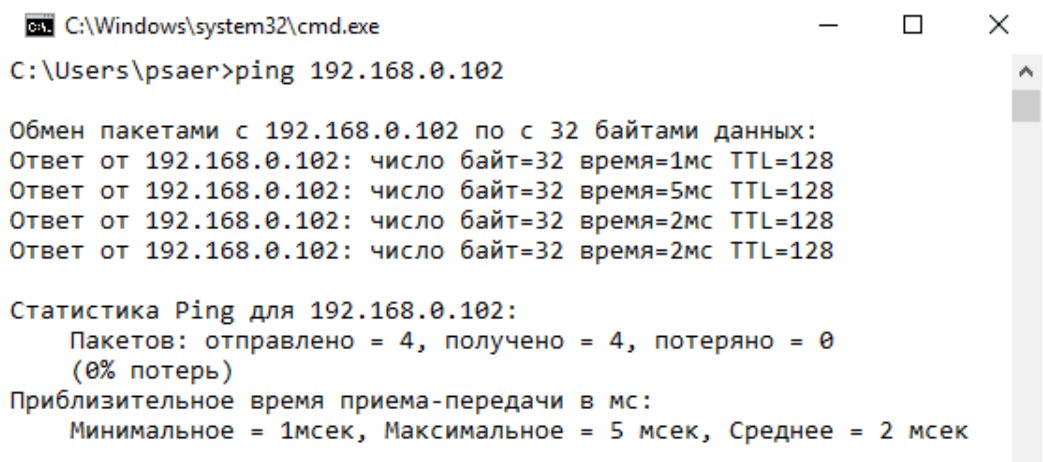


Рис. 1.4: Вызов утилиты в командной строке

No.	Time	Source	Destination	Protocol	Length	Info
→	1 0.000000	192.168.0.104	192.168.0.102	ICMP	74	Echo (ping) request
←	2 0.001806	192.168.0.102	192.168.0.104	ICMP	74	Echo (ping) reply
	3 1.009068	192.168.0.104	192.168.0.102	ICMP	74	Echo (ping) request
> Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0						
> Ethernet II, Src: HonHaiPr_49:6d:bd (14:2d:27:49:6d:bd), Dst: Giga-Byt_24:4a:24 (						
> Internet Protocol Version 4, Src: 192.168.0.104, Dst: 192.168.0.102						
v Internet Control Message Protocol						
Type: 8 (Echo (ping) request)						
Code: 0						
0000	94 de 80 24 4a 24 14 2d 27 49 6d bd 08 00 45 00	...\$J\$.- 'Im...E.				
0010	00 3c 31 f9 00 00 80 01 86 a9 c0 a8 00 68 c0 a8	.<1.....h..				
0020	00 66 08 00 4d 5a 00 01 00 01 61 62 63 64 65 66	.f.MZ.. ..abcdef				
0030	67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76	ghijklmn opqrstuv				
0040	77 61 62 63 64 65 66 67 68 69	wabcdefg hi				

Рис. 1.5: ICMP эхо-запрос

No.	Time	Source	Destination	Protocol	Length	Info
→ 1	0.000000	192.168.0.104	192.168.0.102	ICMP	74	Echo (ping) request
← 2	0.001806	192.168.0.102	192.168.0.104	ICMP	74	Echo (ping) reply
3	1.009068	192.168.0.104	192.168.0.102	ICMP	74	Echo (ping) request

> Frame 2: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0  
 > Ethernet II, Src: Giga-Byt\_24:4a:24 (94:de:80:24:4a:24), Dst: HonHaiPr\_49:6d:bd (  
 > Internet Protocol Version 4, Src: 192.168.0.102, Dst: 192.168.0.104  
 > Internet Control Message Protocol  
   Type: 0 (Echo (ping) reply)  
   Code: 0

0000	14 2d 27 49 6d bd 94 de	80 24 4a 24 08 00 45 00	.. 'Im... . \$J\$.E.
0010	00 3c 31 7e 00 00 80 01	87 24 c0 a8 00 66 c0 a8	.<1~.... . \$...f..
0020	00 68 00 00 55 5a 00 01	00 01 61 62 63 64 65 66	.h. .UZ.. ..abcdef
0030	67 68 69 6a 6b 6c 6d 6e	6f 70 71 72 73 74 75 76	ghijklmn opqrstuv
0040	77 61 62 63 64 65 66 67	68 69	wabcdefg hi

Рис. 1.6: ICMP эхо-ответ

Пакеты были распознаны как ICMP с пометкой "Echo (ping) reply/request" что означает эхо запрос/ответ.

Тип сообщения равный 0 означает эхо-запрос, а тип 0 означает эхо-ответ.

Графа Destination показывает IP адрес удаленного сервера, который мы пингуем, Source показывает IP адрес текущего компьютера.

В качестве передаваемой информации передаются коды символов a-w.

## Ping с фрагментацией

Для фрагментации пакета необходимо указать его размер, превышающий MTU (maximum transmission unit) - максимальный размер полезного блока данных одного пакета, который может быть передан протоколом без фрагментации. Максимальный размер одного блока данных, который может быть передан без фрагментации составляет 1480 байт.

С помощью утилиты ping изменяем размер буфера отправки на 8192 байт.

```

C:\Windows\system32\cmd.exe
C:\Users\psaer>ping 192.168.0.102 -l 8192

Обмен пакетами с 192.168.0.102 по с 8192 байтами данных:
Ответ от 192.168.0.102: число байт=8192 время=6мс TTL=128
Ответ от 192.168.0.102: число байт=8192 время=34мс TTL=128
Ответ от 192.168.0.102: число байт=8192 время=7мс TTL=128
Ответ от 192.168.0.102: число байт=8192 время=7мс TTL=128

Статистика Ping для 192.168.0.102:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0
    (0% потерь)
    Приблизительное время приема-передачи в мс:
    Минимальное = 6мсек, Максимальное = 34 мсек, Среднее = 13 мсек
  
```

Рис. 1.7: Вызов утилиты в командной строке

Из рисунка 1.8 видно что первый пакет был отправлен по протоколу ICMP, следующие фрагментированные пакеты передавались по протоколу IPv4 уже без заголовка ICMP. Каждый следующий пакет имеет fragment offset увеличенный на 1480, по сравнению с предыдущим.

No.	Time	Source	Destination	Protocol	Length	Info
→ 1	0.000000	192.168.0.104	192.168.0.102	ICMP	1514	Echo (ping) request id=0x0001, seq=155/39680, ttl=128 (reply in 7)
2	0.000019	192.168.0.104	192.168.0.102	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=26d8)
3	0.000029	192.168.0.104	192.168.0.102	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=26d8)
4	0.000039	192.168.0.104	192.168.0.102	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=4440, ID=26d8)
5	0.000049	192.168.0.104	192.168.0.102	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=5920, ID=26d8)
6	0.000058	192.168.0.104	192.168.0.102	IPv4	834	Fragmented IP protocol (proto=ICMP 1, off=7400, ID=26d8)
← 7	0.007056	192.168.0.102	192.168.0.104	ICMP	1514	Echo (ping) reply id=0x0001, seq=155/39680, ttl=128 (request in 1)
8	0.007059	192.168.0.102	192.168.0.104	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=69d3)
9	0.007060	192.168.0.102	192.168.0.104	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=69d3)
10	0.007064	192.168.0.102	192.168.0.104	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=4440, ID=69d3)
11	0.009510	192.168.0.102	192.168.0.104	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=5920, ID=69d3)
12	0.009512	192.168.0.102	192.168.0.104	IPv4	834	Fragmented IP protocol (proto=ICMP 1, off=7400, ID=69d3)

Рис. 1.8: Сегментирование пакетов

No.	Time	Source	Destination	Protocol	Length	Info
→ 1	0.000000	192.168.0.104	192.168.0.102	ICMP	1514	Echo (ping) request id=0x0001, seq=155/39680, ttl=128
2	0.000019	192.168.0.104	192.168.0.102	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=26d8)
3	0.000029	192.168.0.104	192.168.0.102	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=26d8)

>	Frame 1: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
>	Ethernet II, Src: HonHaiPr_49:6d:bd (14:2d:27:49:6d:bd), Dst: Giga-Byt_24:4a:24 (94:de:80:24:4a:24)
✓	Internet Protocol Version 4, Src: 192.168.0.104, Dst: 192.168.0.102
	0100 .... = Version: 4
	.... 0101 = Header Length: 20 bytes (5)
>	Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
	Total Length: 1500
	Identification: 0x26d8 (9944)
>	Flags: 0x01 (More Fragments)
	Fragment offset: 0
	Time to live: 128
	Protocol: ICMP (1)
	Header checksum: 0x6c2a [validation disabled]
	[Header checksum status: Unverified]
	Source: 192.168.0.104
	Destination: 192.168.0.102
	[Source GeoIP: Unknown]
	[Destination GeoIP: Unknown]
✓	Internet Control Message Protocol
	Type: 8 (Echo (ping) request)
	Code: 0
	Checksum: 0x44af [unverified] [fragmented datagram]
	[Checksum Status: Unverified]
	Identifier (BE): 1 (0x0001)
	Identifier (LE): 256 (0x0100)
	Sequence number (BE): 155 (0x009b)
	Sequence number (LE): 39680 (0x9b00)
	<a href="#">[Response frame: 7]</a>
>	Data (1472 bytes)

Рис. 1.9: Первый фрагмент пакета ping-запроса

Первый пакет имеет заголовок ICMP(8 байт) и данные(1472 байта). О фрагментированности пакета свидетельствуют флаги пакета IP (0x01 – имеются еще фрагменты). О том, что это первый пакет из фрагментированных, свидетельствует нулевое смещение фрагмента.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.104	192.168.0.102	ICMP	1514	Echo (ping) request id=0x0001, seq=155/39680, ttl=128 (
2	0.000019	192.168.0.104	192.168.0.102	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=26d8)
3	0.000029	192.168.0.104	192.168.0.102	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=26d8)

```

> Frame 2: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
> Ethernet II, Src: HonHaiPr_49:6d:bd (14:2d:27:49:6d:bd), Dst: Giga-Byt_24:4a:24 (94:de:80:24:4a:24)
▼ Internet Protocol Version 4, Src: 192.168.0.104, Dst: 192.168.0.102
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
        Total Length: 1500
        Identification: 0x26d8 (9944)
    > Flags: 0x01 (More Fragments)
        Fragment offset: 1480
        Time to live: 128
        Protocol: ICMP (1)
        Header checksum: 0x6b71 [validation disabled]
        [Header checksum status: Unverified]
        Source: 192.168.0.104
        Destination: 192.168.0.102
        [Source GeoIP: Unknown]
        [Destination GeoIP: Unknown]
    > Data (1480 bytes)

```

Рис. 1.10: Второй фрагмент пакета ping-запроса

Второй пакет все так же имеет флаг в заголовке IP-пакета (0x01), свидетельствующий о наличии пакетов кроме данного. От первого пакета его отличает ненулевое смещение фрагмента, а также отсутствие заголовка ICMP пакета, в следствии чего количество передаваемых данных увеличилось с 1472 до 1480 байт, по сравнению с предыдущим пакетом.

No.	Time	Source	Destination	Protocol	Length	Info
5	0.000049	192.168.0.104	192.168.0.102	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=5920, ID=26d8)
6	0.000058	192.168.0.104	192.168.0.102	IPv4	834	Fragmented IP protocol (proto=ICMP 1, off=7400, ID=26d8)
7	0.007056	192.168.0.102	192.168.0.104	ICMP	1514	Echo (ping) reply id=0x0001, seq=155/39680, ttl=128 (

```

> Frame 6: 834 bytes on wire (6672 bits), 834 bytes captured (6672 bits) on interface 0
> Ethernet II, Src: HonHaiPr_49:6d:bd (14:2d:27:49:6d:bd), Dst: Giga-Byt_24:4a:24 (94:de:80:24:4a:24)
▼ Internet Protocol Version 4, Src: 192.168.0.104, Dst: 192.168.0.102
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
        Total Length: 820
        Identification: 0x26d8 (9944)
    > Flags: 0x00
        Fragment offset: 7400
        Time to live: 128
        Protocol: ICMP (1)
        Header checksum: 0x8b35 [validation disabled]
        [Header checksum status: Unverified]
        Source: 192.168.0.104
        Destination: 192.168.0.102
        [Source GeoIP: Unknown]
        [Destination GeoIP: Unknown]
    > Data (800 bytes)

```

Рис. 1.11: Последний фрагмент пакета ping-запроса

Пакет, содержащий последний фрагмент ping-запроса, уже не имеет флагов в заголовке IP-пакета, о наличии дополнительных пакетов. Также имеется смещение фрагмента, а также оставшиеся для передачи данные.

После вышеприведенного фрагментированного эхо-запроса, следует такой-же эхо-ответ.

## 14.2 Утилита tracert

Пронаблюдаем трассировку маршрута пакетов до узла **kspt.icc.spbstu.ru** при помощи протокола ICMP и утилиты tracert.



```

C:\Windows\system32\cmd.exe
C:\Users\psaer>tracert kspt.icc.spbstu.ru

Трассировка маршрута к kspt.icc.spbstu.ru [91.151.191.13]
с максимальным числом прыжков 30:

 1      2 ms      1 ms      4 ms  lan-82-001.users.mns.ru [178.162.82.1]
 2      4 ms      2 ms      4 ms  df-1-142.users.mns.ru [80.70.224.142]
 3      5 ms      5 ms      5 ms  gw.mns.ru [80.70.239.254]
 4      4 ms      5 ms      5 ms  as5433.ix.dataix.ru [178.18.224.121]
 5      4 ms      4 ms      5 ms  gw-politech.nw.ru [91.151.178.42]
 6      4 ms      8 ms      4 ms  white.ftk.spbstu.ru [91.151.191.13]

Трассировка завершена.

```

Рис. 1.12: Результат трассировки маршрута в консоли

Как видно из рисунка 1.12, первый прыжок произошел на узел **178.162.82.1**, а не на адрес роутера **192.168.0.1**, указанный на рисунке 1.1. Это связано с тем, что из-за особенностей прошивки роутера, уменьшение ttl не происходит, в следствии чего:

- пакет проходит на следующий узел;
- роутер не отображается в списке узлов.

Первый пакет трассировки маршрута отправляется с TTL равным 1. Это значит, что на первом же маршрутизаторе, который проверит значение ttl, пакет будет уничтожен и нам придет сообщение об ошибке.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.104	91.151.191.13	ICMP	106	Echo (ping) request id=0x0001, seq=177/45312, ttl=1 (no
2	0.002582	178.162.82.1	192.168.0.104	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
3	0.003814	192.168.0.104	91.151.191.13	ICMP	106	Echo (ping) request id=0x0001, seq=178/45568, ttl=1 (no

```

> Frame 1: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface 0
> Ethernet II, Src: HonHaiPr_49:6d:bd (14:2d:27:49:6d:bd), Dst: D-Link_3e:a6:d6 (f0:7d:68:3e:a6:d6)
v Internet Protocol Version 4, Src: 192.168.0.104, Dst: 91.151.191.13
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 92
    Identification: 0x2fa4 (12196)
  > Flags: 0x00
    Fragment offset: 0
  > Time to live: 1
    Protocol: ICMP (1)
    Header checksum: 0xae48 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.0.104
    Destination: 91.151.191.13
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
v Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0xf74d [correct]
  [Checksum Status: Good]
  Identifier (BE): 1 (0x0001)
  Identifier (LE): 256 (0x0100)
  Sequence number (BE): 177 (0x00b1)
  Sequence number (LE): 45312 (0xb100)
  > [No response seen]
  > Data (64 bytes)

```

Рис. 1.13: Отправка первого пакета трассировки

- Internet Protocol Version 4, Src: 192.168.0.104, Dst: 91.151.191.13
  - 0100 .... = Version: 4
  - .... 0101 = Header Length: 20 bytes (5)
  - > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  - Total Length: 92
  - Identification: 0x2fa4 (12196)
  - > Flags: 0x00
  - Fragment offset: 0
  - > Time to live: 1
  - Protocol: ICMP (1)
  - Header checksum: 0xae48 [validation disabled]
  - [Header checksum status: Unverified]
  - Source: 192.168.0.104
  - Destination: 91.151.191.13
  - [Source GeoIP: Unknown]
  - [Destination GeoIP: Unknown]
- Internet Control Message Protocol
  - Type: 8 (Echo (ping) request)
  - Code: 0
  - Checksum: 0xf74d [correct]
  - [Checksum Status: Good]
  - Identifier (BE): 1 (0x0001)
  - Identifier (LE): 256 (0x0100)
  - Sequence number (BE): 177 (0x00b1)
  - Sequence number (LE): 45312 (0xb100)
  - > [No response seen]
  - > Data (64 bytes)

Рис. 114: Часть первого пакета трассировки

- Internet Control Message Protocol
  - Type: 11 (Time-to-live exceeded)
  - Code: 0 (Time to live exceeded in transit)
  - Checksum: 0xf4ff [correct]
  - [Checksum Status: Good]
- Internet Protocol Version 4, Src: 192.168.0.104, Dst: 91.151.191.13
  - 0100 .... = Version: 4
  - .... 0101 = Header Length: 20 bytes (5)
  - > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  - Total Length: 92
  - Identification: 0x2fa4 (12196)
  - > Flags: 0x00
  - Fragment offset: 0
  - > Time to live: 1
  - Protocol: ICMP (1)
  - Header checksum: 0xae48 [validation disabled]
  - [Header checksum status: Unverified]
  - Source: 192.168.0.104
  - Destination: 91.151.191.13
  - [Source GeoIP: Unknown]
  - [Destination GeoIP: Unknown]
- Internet Control Message Protocol
  - Type: 8 (Echo (ping) request)
  - Code: 0
  - Checksum: 0xf74d [unverified] [in ICMP error packet]
  - [Checksum Status: Unverified]
  - Identifier (BE): 1 (0x0001)
  - Identifier (LE): 256 (0x0100)
  - Sequence number (BE): 177 (0x00b1)
  - Sequence number (LE): 45312 (0xb100)

Рис. 115: Часть ответа на первый пакет трассировки

No.	Time	Source	Destination	Protocol	Length	Info
4	0.004983	178.162.82.1	192.168.0.104	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
5	0.005552	192.168.0.104	91.151.191.13	ICMP	106	Echo (ping) request id=0x0001, seq=78/19968, ttl=1 (no response found!)
6	0.007434	178.162.82.1	192.168.0.104	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
7	1.025660	192.168.0.104	91.151.191.13	ICMP	106	Echo (ping) request id=0x0001, seq=79/20224, ttl=2 (no response found!)
8	1.028936	80.70.224.142	192.168.0.104	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)

> Frame 6: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0

> Ethernet II, Src: D-Link\_3e:a6:d6 (f0:7d:68:3e:a6:d6), Dst: HonHaiPr\_49:6d:bd (14:2d:27:49:6d:bd)

- Internet Protocol Version 4, Src: 178.162.82.1, Dst: 192.168.0.104
  - 0100 .... = Version: 4
  - .... 0101 = Header Length: 20 bytes (5)
  - > Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)
  - Total Length: 56
  - Identification: 0x271f (10015)
  - > Flags: 0x00
  - Fragment offset: 0
  - Time to live: 254
  - Protocol: ICMP (1)
  - Header checksum: 0xcf31 [validation disabled]
  - [Header checksum status: Unverified]
  - Source: 178.162.82.1
  - Destination: 192.168.0.104
  - [Source GeoIP: Unknown]
  - [Destination GeoIP: Unknown]
- Internet Control Message Protocol
  - Type: 11 (Time-to-live exceeded)
  - Code: 0 (Time to live exceeded in transit)
  - Checksum: 0xf4ff [correct]
  - [Checksum Status: Good]
- Internet Protocol Version 4, Src: 192.168.0.104, Dst: 91.151.191.13
  - 0100 .... = Version: 4
  - .... 0101 = Header Length: 20 bytes (5)
  - > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  - Total Length: 92
  - Identification: 0x2fa6 (12198)
  - > Flags: 0x00
  - Fragment offset: 0
  - > Time to live: 1
  - Protocol: ICMP (1)
  - Header checksum: 0xae46 [validation disabled]
  - [Header checksum status: Unverified]
  - Source: 192.168.0.104
  - Destination: 91.151.191.13
  - [Source GeoIP: Unknown]
  - [Destination GeoIP: Unknown]
- Internet Control Message Protocol
  - Type: 8 (Echo (ping) request)
  - Code: 0
  - Checksum: 0xf74b [unverified] [in ICMP error packet]
  - [Checksum Status: Unverified]
  - Identifier (BE): 1 (0x0001)
  - Identifier (LE): 256 (0x0100)
  - Sequence number (BE): 179 (0x00b3)
  - Sequence number (LE): 45824 (0xb300)

Рис. 116: Ответ на первый пакет трассировки

На рисунке 1.16 видно что, сообщение пришло от маршрутизатора сети, который имеет адрес **178.162.82.1**. В сообщении об ошибке указан тип ICMP-пакета – 11.0, что означает, что время жизни пакета истекло.

На рисунках 1.14 и 1.15 происходит сравнение содержимого пакетов при запросе и ответе. При ответе, в пакете также пристыковывается IP заголовок запроса.

6	0.007434	178.162.82.1	192.168.0.104	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
7	1.025660	192.168.0.104	91.151.191.13	ICMP	106	Echo (ping) request id=0x0001, seq=79/20224, ttl=2 (no response found!)
8	1.028936	80.70.224.142	192.168.0.104	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)

```

> Frame 7: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface 0
> Ethernet II, Src: HonHaiPr_49:6d:bd (14:2d:27:49:6d:bd), Dst: D-Link_3e:a6:d6 (f0:7d:68:3e:a6:d6)
▼ Internet Protocol Version 4, Src: 192.168.0.104, Dst: 91.151.191.13
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 92
    Identification: 0x5e05 (24069)
    > Flags: 0x00
    Fragment offset: 0
    > Time to live: 2
    Protocol: ICMP (1)
    Header checksum: 0x7ee7 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.0.104
    Destination: 91.151.191.13
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▼ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0

```

Рис. 1.17: Второй пакет трассировки маршрута

Следующий пакет будет отправлен на тот же адрес, что и ранее, но параметр TTL будет установлен в 2, чтобы пройти маршрутизатор по адресу 178.162.82.1 и попасть в следующий пункт передачи.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.004983	178.162.82.1	192.168.0.104	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
5	0.005552	192.168.0.104	91.151.191.13	ICMP	106	Echo (ping) request id=0x0001, seq=78/19968, ttl=1 (no response found!)
6	0.007434	178.162.82.1	192.168.0.104	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
7	1.025660	192.168.0.104	91.151.191.13	ICMP	106	Echo (ping) request id=0x0001, seq=79/20224, ttl=2 (no response found!)
8	1.028936	80.70.224.142	192.168.0.104	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)

```

> Frame 8: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: D-Link_3e:a6:d6 (f0:7d:68:3e:a6:d6), Dst: HonHaiPr_49:6d:bd (14:2d:27:49:6d:bd)
▼ Internet Protocol Version 4, Src: 80.70.224.142, Dst: 192.168.0.104
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x9746 (38726)
    > Flags: 0x00
    Fragment offset: 0
    Time to live: 254
    Protocol: ICMP (1)
    Header checksum: 0x32d9 [validation disabled]
    [Header checksum status: Unverified]
    Source: 80.70.224.142
    Destination: 192.168.0.104
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▼ Internet Control Message Protocol
    Type: 11 (Time-to-live exceeded)

```

Рис. 1.18: Ответ на второй пакет трассировки

В ответе на второй пакет трассировки маршрута в качестве отправителя сообщения об ошибке истечения жизни пакета указан адрес 80.70.224.142. Значит, это и есть следующий пункт передачи.

Аналогично продолжается трассировка маршрута дальше с постепенным инкрементом параметра TTL. Таким образом составляется примерный маршрут прохождения IP-пакета до узла с адресом **kspt.icc.spbstu.ru**.

## 14.3 Протокол ARP

Рассмотрим пару APR пакетов, которая демонстрирует работу протокола

No.	Time	Source	Destination	Protocol	Length	Info
39	23.5943...	D-Link_3e:a6:d6	HonHaiPr_49:6d:bd	ARP	42	Who has 192.168.0.104? Tell 192.168.0.1
40	23.5944...	HonHaiPr_49:6d:bd	D-Link_3e:a6:d6	ARP	42	192.168.0.104 is at 14:2d:27:49:6d:bd
41	24.2564...	fe80::ffff:ffff::	ff02::2	ICMPv6	103	Router Solicitation

> Frame 39: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0  
 > Ethernet II, Src: D-Link\_3e:a6:d6 (f0:7d:68:3e:a6:d6), Dst: HonHaiPr\_49:6d:bd (14:2d:27:49:6d:bd)  
 > Address Resolution Protocol (request)  
     Hardware type: Ethernet (1)  
     Protocol type: IPv4 (0x0800)  
     Hardware size: 6  
     Protocol size: 4  
     Opcode: request (1)  
     Sender MAC address: D-Link\_3e:a6:d6 (f0:7d:68:3e:a6:d6)  
     Sender IP address: 192.168.0.1  
     Target MAC address: 00:00:00\_00:00:00 (00:00:00:00:00:00)  
     Target IP address: 192.168.0.104

Рис. 119: ARP запрос

В пакете ARP-запроса указывается его тип (поле Opcode) – 0x1 для запроса и 0x2 для ответа. Указан целевой IP-адрес для которого запрашивается MAC-адрес, MAC-адрес цели при этом обнулен.

No.	Time	Source	Destination	Protocol	Length	Info
39	23.5943...	D-Link_3e:a6:d6	HonHaiPr_49:6d:bd	ARP	42	Who has 192.168.0.104? Tell 192.168.0.1
40	23.5944...	HonHaiPr_49:6d:bd	D-Link_3e:a6:d6	ARP	42	192.168.0.104 is at 14:2d:27:49:6d:bd
41	24.2564...	fe80::ffff:ffff::	ff02::2	ICMPv6	103	Router Solicitation

> Frame 40: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0  
 > Ethernet II, Src: HonHaiPr\_49:6d:bd (14:2d:27:49:6d:bd), Dst: D-Link\_3e:a6:d6 (f0:7d:68:3e:a6:d6)  
 > Address Resolution Protocol (reply)  
     Hardware type: Ethernet (1)  
     Protocol type: IPv4 (0x0800)  
     Hardware size: 6  
     Protocol size: 4  
     Opcode: reply (2)  
     Sender MAC address: HonHaiPr\_49:6d:bd (14:2d:27:49:6d:bd)  
     Sender IP address: 192.168.0.104  
     Target MAC address: D-Link\_3e:a6:d6 (f0:7d:68:3e:a6:d6)  
     Target IP address: 192.168.0.1

Рис. 120: ARP ответ

В ответе возвращается результирующий MAC-адрес.

Была произведена попытка произвести ping на несуществующий адрес(192.168.0.156) в текущей сети. В результате чего, роутер начал посылать широковещательные **ARP** запросы, для определения неизвестного для него адреса в сети.

```

C:\Windows\system32\cmd.exe
C:\Users\psaer>ping 192.168.0.156

Обмен пакетами с 192.168.0.156 по с 32 байтами данных:
Ответ от 192.168.0.104: Заданный узел недоступен.
Ответ от 192.168.0.1: Заданный узел недоступен.
Ответ от 192.168.0.1: Заданный узел недоступен.
Ответ от 192.168.0.1: Заданный узел недоступен.

Статистика Ping для 192.168.0.156:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0
    (0% потерь)
  
```

Рис. 121: Попытка команды ping на несуществующий адрес

12	0.994381	D-Link_3e:a6:d6	Broadcast	ARP	42	Who has 192.168.0.156? Tell 192.168.0.1
13	1.916082	D-Link_3e:a6:d6	Broadcast	ARP	42	Who has 192.168.0.156? Tell 192.168.0.1
14	2.940127	D-Link_3e:a6:d6	Broadcast	ARP	42	Who has 192.168.0.156? Tell 192.168.0.1

> Frame 12: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0  
> Ethernet II, Src: D-Link\_3e:a6:d6 (f0:7d:68:3e:a6:d6), Dst: Broadcast (ff:ff:ff:ff:ff:ff)  
▼ Address Resolution Protocol (request)  
Hardware type: Ethernet (1)  
Protocol type: IPv4 (0x0800)  
Hardware size: 6  
Protocol size: 4  
Opcode: request (1)  
Sender MAC address: D-Link\_3e:a6:d6 (f0:7d:68:3e:a6:d6)  
Sender IP address: 192.168.0.1  
Target MAC address: 00:00:00\_00:00:00 (00:00:00:00:00:00)  
Target IP address: 192.168.0.156

Рис. 1.22: Широковещательный ARP-запрос

## 1.4.4 Протокол ICMP

Для того чтобы пронаблюдать ошибку типа 3.1 (целевой узел недостижим), отправим ping-запрос на адрес(192.168.0.156), которого не существует.

```

C:\Windows\system32\cmd.exe
C:\Users\psaer>ping 192.168.0.156

Обмен пакетами с 192.168.0.156 по с 32 байтами данных:
Ответ от 192.168.0.1: Заданный узел недоступен.
Ответ от 192.168.0.1: Заданный узел недоступен.
Ответ от 192.168.0.1: Заданный узел недоступен.
Ответ от 192.168.0.1: Заданный узел недоступен.

Статистика Ping для 192.168.0.156:
    Пакетов: отправлено = 4, получено = 0, потеряно = 0
    (0% потерь)

```

Рис. 1.23: Вызов утилиты в командной строке

В пакете можно наблюдать типичный ping-запрос (ICMP-пакет типа 8.0).

11	0.912540	192.168.0.104	192.168.0.156	ICMP	74	Echo (ping) request id=0x0001, seq=1
12	0.994381	D-Link_3e:a6:d6	Broadcast	ARP	42	Who has 192.168.0.156? Tell 192.168.0.1

> Frame 11: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0  
> Ethernet II, Src: HonHaiPr\_49:6d:bd (14:2d:27:49:6d:bd), Dst: D-Link\_3e:a6:d6 (f0:7d:68:3e:a6:d6)  
> Internet Protocol Version 4, Src: 192.168.0.104, Dst: 192.168.0.156  
▼ Internet Control Message Protocol  
Type: 8 (Echo (ping) request)  
Code: 0  
Checksum: 0x4cf4 [correct]  
[Checksum Status: Good]  
Identifier (BE): 1 (0x0001)  
Identifier (LE): 256 (0x0100)  
Sequence number (BE): 103 (0x0067)  
Sequence number (LE): 26368 (0x6700)  
> [No response seen]  
▼ Data (32 bytes)  
Data: 61626364656666768696a6b6c6d6e6f707172737475767761...  
[Length: 32]

Рис. 1.24: ICMP - эхо запрос

Ответом на указанный выше запрос будет ICMP-пакет типа 3.1, свидетельствующий об ошибке «целевой узел недостижим».

No.	Time	Source	Destination	Protocol	Length	Info
17	5.644775	64.233.164.196	192.168.0.104	TCP	66	443 → 50697 [ACK] Seq=1 Ack=2 Win=392 Len=0
18	5.665357	192.168.0.1	192.168.0.104	ICMP	102	Destination unreachable (Host unreachable)
19	5.670311	192.168.0.104	192.168.0.156	ICMP	74	Echo (ping) request id=0x0001, seq=197/5043

>	Frame 18: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface 0
>	Ethernet II, Src: D-Link_3e:a6:d6 (f0:7d:68:3e:a6:d6), Dst: HonHaiPr_49:6d:bd (14:2d:27:49:6d:bd)
>	Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.104
▼	Internet Control Message Protocol
	Type: 3 (Destination unreachable)
	Code: 1 (Host unreachable)
	Checksum: 0xfcfe [correct]
	[Checksum Status: Good]
	Unused: 00000000
▼	Internet Protocol Version 4, Src: 192.168.0.104, Dst: 192.168.0.156
	0100 .... = Version: 4
	.... 0101 = Header Length: 20 bytes (5)
>	Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
	Total Length: 60
	Identification: 0x6bd6 (27606)
>	Flags: 0x00
	Fragment offset: 0
	Time to live: 128
	Protocol: ICMP (1)
	Header checksum: 0x4c96 [validation disabled]
	[Header checksum status: Unverified]
	Source: 192.168.0.104
	Destination: 192.168.0.156
	[Source GeoIP: Unknown]
	[Destination GeoIP: Unknown]
▼	Internet Control Message Protocol
	Type: 8 (Echo (ping) request)
	Code: 0
	Checksum: 0x4c97 [unverified] [in ICMP error packet]
	[Checksum Status: Unverified]
	Identifier (BE): 1 (0x0001)
	Identifier (LE): 256 (0x0100)
	Sequence number (BE): 196 (0x00c4)
	Sequence number (LE): 50176 (0xc400)
▼	Data (32 bytes)
	Data: 6162636465666768696a6b6c6d6e6f707172737475767761...
	[Length: 32]

0010	00 58 92 35 00 00 40 01 65 f6 c0 a8 00 01 c0 a8	.X.5..@. e.....
0020	00 68 03 01 fc fe 00 00 00 00 45 00 00 3c 6b d6	.h..... ..E..<k.
0030	00 00 80 01 4c 96 c0 a8 00 68 c0 a8 00 9c 08 00	....L.... .h.....
0040	4c 97 00 01 00 c4 61 62 63 64 65 66 67 68 69 6a	L....ab cdefghij
0050	6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 61 62 63	klmnopqr stuvwabc
0060	64 65 66 67 68 69	defghi

Рис. 1.25: ICMP-ответ

Ответ также содержит данные из запроса(пристыкованный IP заголовок).

17	4.477851	192.168.0.1	192.168.0.104	ICMP	102	Redirect (Redirect for host)
----	----------	-------------	---------------	------	-----	------------------------------

>	Frame 17: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface 0
>	Ethernet II, Src: D-Link_3e:a6:d6 (f0:7d:68:3e:a6:d6), Dst: HonHaiPr_49:6d:bd (14:2d:27:49:6d:bd)
>	Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.104
▼	Internet Control Message Protocol
	Type: 5 (Redirect)
	Code: 1 (Redirect for host)
	Checksum: 0x39ba [correct]
	[Checksum Status: Good]
	Gateway address: 192.168.0.156
>	Internet Protocol Version 4, Src: 192.168.0.104, Dst: 192.168.0.156

Рис. 1.26: Пакет перенаправления



Следом за ICMP-пакетом типа 3.1, следует ICMP-пакет типа 5.1 (Redirect datagrams for the Host). Данный пакет информирует хост о необходимости создания нового маршрута к указанному в сообщении хосту и внесения его в таблицу маршрутизации. Для этого в сообщении указывается IP адрес хоста, где необходима смена маршрута (адрес занесен в поле Destination в пристыкованном IP заголовке), и новый IP адрес маршрутизатора, на который необходимо направлять пакеты, адресованные данному хосту (этот адрес заносится в поле Gateway).

Gateway адрес по прежнему остался недостижимым узлом **192.168.0.156**.

## 14.5 Протокол UDP

В ознакомительных целях, проведем отправку udp-пакета на несуществующий адрес. Для этого была написана соответствующая программа, код которой приведен в приложении 1.

No.	Time	Source	Destination	Protocol	Length	Info
16	0.650855	64.233.164.188	192.168.0.104	TCP	66	5228 → 58968 [ACK] Seq=
17	1.563911	192.168.0.104	192.168.56.2	UDP	50	52913 → 8005 Len=8
18	2.062237	192.168.0.104	74.125.205.200	SSL	55	Continuation Data

>	Frame 17: 50 bytes on wire (400 bits), 50 bytes captured (400 bits) on interface 0
>	Ethernet II, Src: HonHaiPr_49:6d:bd (14:2d:27:49:6d:bd), Dst: D-Link_3e:a6:d6 (f0:7d:11:3e:a6:d6)
>	Internet Protocol Version 4, Src: 192.168.0.104, Dst: 192.168.56.2
▼	User Datagram Protocol, Src Port: 52913, Dst Port: 8005
	Source Port: 52913
	Destination Port: 8005
	Length: 16
	Checksum: 0xb77f [unverified]
	[Checksum Status: Unverified]
	[Stream index: 2]
▼	Data (8 bytes)
	Data: 7465737444617461
	[Length: 8]

0000	f0 7d 68 3e a6 d6 14 2d 27 49 6d bd 08 00 45 00	.}h>...- 'Im...E.
0010	00 24 37 eb 00 00 80 11 49 23 c0 a8 00 68 c0 a8	.\$7..... I#...h..
0020	38 02 ce b1 1f 45 00 10 b7 7f 74 65 73 74 44 61	8....E.. ..testDa
0030	74 61	ta

Рис. 1.27: UDP - пакет

Чего и требовалось ожидать, UDP - пакет был отправлен на несуществующий адрес **192.168.56.2**, с портом **8005**, ответа на данный пакет не последовало. В качестве пересылаемых данных выступали 8 байт текста - **testData**

## 14.6 Протокол TCP

Все последующие опыты будут выполнены при использовании двух ПК, находящихся в одной сети. Их сетевые параметры представлены в пункте **Конфигурация сети**. Были написаны программы tcp сервера(приложение 2, 3, 4) и клиента(приложение 5). На одном из ПК(192.168.0.102) будет запущен TCP - сервер, а на другом(192.168.0.104) TCP - клиент.

### Установка соединения

При установке соединения между клиентом и сервером, происходит передача трех пакетов. Клиент, посылает серверу сегмент с номером последовательности и флагом SYN.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.104	192.168.0.102	TCP	66	51898 → 8090 [SYN] Seq=1310762943 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	0.011447	192.168.0.102	192.168.0.104	TCP	66	8090 → 51898 [SYN, ACK] Seq=1845932618 Ack=1310762944 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
3	0.011530	192.168.0.104	192.168.0.102	TCP	54	51898 → 8090 [ACK] Seq=1310762944 Ack=1845932619 Win=16384 Len=0

> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0

> Ethernet II, Src: HonHaiPr\_49:6d:bd (14:2d:27:49:6d:bd), Dst: Giga-Byt\_24:4a:24 (94:de:80:24:4a:24)

> Internet Protocol Version 4, Src: 192.168.0.104, Dst: 192.168.0.102

▼ Transmission Control Protocol, Src Port: 51898, Dst Port: 8090, Seq: 1310762943, Len: 0

Source Port: 51898

Destination Port: 8090

[Stream index: 0]

[TCP Segment Len: 0]

Sequence number: 1310762943

Acknowledgment number: 0

Header Length: 32 bytes

> Flags: 0x002 (SYN)

Window size value: 8192

[Calculated window size: 8192]

Checksum: 0xecbc [unverified]

[Checksum Status: Unverified]

Urgent pointer: 0

> Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted

Рис. 1.28: Первый пакет при установке TCP-соединения

В случае успеха сервер посылает клиенту сегмент с номером последовательности и флагами SYN и ACK, и переходит в состояние SYN-RECEIVED. В случае неудачи сервер посылает клиенту сегмент с флагом RST.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.104	192.168.0.102	TCP	66	51898 → 8090 [SYN] Seq=1310762943 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	0.011447	192.168.0.102	192.168.0.104	TCP	66	8090 → 51898 [SYN, ACK] Seq=1845932618 Ack=1310762944 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
3	0.011530	192.168.0.104	192.168.0.102	TCP	54	51898 → 8090 [ACK] Seq=1310762944 Ack=1845932619 Win=16384 Len=0

> Frame 2: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0

> Ethernet II, Src: Giga-Byt\_24:4a:24 (94:de:80:24:4a:24), Dst: HonHaiPr\_49:6d:bd (14:2d:27:49:6d:bd)

> Internet Protocol Version 4, Src: 192.168.0.102, Dst: 192.168.0.104

▼ Transmission Control Protocol, Src Port: 8090, Dst Port: 51898, Seq: 1845932618, Ack: 1310762944, Len: 0

Source Port: 8090

Destination Port: 51898

[Stream index: 0]

[TCP Segment Len: 0]

Sequence number: 1845932618

Acknowledgment number: 1310762944

Header Length: 32 bytes

> Flags: 0x012 (SYN, ACK)

Window size value: 8192

[Calculated window size: 8192]

Checksum: 0xcc5a [unverified]

[Checksum Status: Unverified]

Urgent pointer: 0

> Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP)

> [SEQ/ACK analysis]

Рис. 1.29: Второй пакет при установке TCP-соединения

В заголовке tcp-пакета также имеются:

- **Sequence Number** - порядковый номер: 32 бита
- **Acknowledgment Number** - номер подтверждения: 32 бита

В первом пакете seq. number: 1310762943, ack. number: 0. Так как это первый пакет для инициализации соединения, seq. number был сгенерирован случайным образом, а ack. number равен 0 так как ответов от сервера еще поступало.

Во втором пакете, сервер сгенерировал свой seq. number: 1845932618, в поле же ack. number записано 1310762944, то есть сервер инкриминировал полученный от клиента seq. number, и теперь ожидает от него пакет с соответствующим seq. number.



Последний этап это отправка на сервер пакета с установленным флагом ACK, после чего соединение переходит в состояние ESTABLISHED.

No.	Time	Source	Destination	Protocol	Length	Info
2	0.011447	192.168.0.102	192.168.0.104	TCP	66	8090 → 51898 [SYN, ACK] Seq=1845932618 Ack=1310762944 Win=8192 Len=0
3	0.011530	192.168.0.104	192.168.0.102	TCP	54	51898 → 8090 [ACK] Seq=1310762944 Ack=1845932619 Win=16384 Len=0
4	3.130648	192.168.0.104	192.168.0.102	TCP	61	51898 → 8090 [PSH, ACK] Seq=1310762944 Ack=1845932619 Win=16384 Len=7

> Frame 3: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0  
> Ethernet II, Src: HonHaiPr\_49:6d:bd (14:2d:27:49:6d:bd), Dst: Giga-Byt\_24:4a:24 (94:de:80:24:4a:24)  
> Internet Protocol Version 4, Src: 192.168.0.104, Dst: 192.168.0.102  
▼ Transmission Control Protocol, Src Port: 51898, Dst Port: 8090, Seq: 1310762944, Ack: 1845932619, Len: 0  
Source Port: 51898  
Destination Port: 8090  
[Stream index: 0]  
[TCP Segment Len: 0]  
Sequence number: 1310762944  
Acknowledgment number: 1845932619  
Header Length: 20 bytes  
> Flags: 0x010 (ACK)  
Window size value: 64  
[Calculated window size: 16384]  
[Window size scaling factor: 256]  
Checksum: 0x2cee [unverified]  
[Checksum Status: Unverified]  
Urgent pointer: 0  
> [SEQ/ACK analysis]

Рис. 1.30: Третий пакет при установке TCP-соединения

В третьем пакете, как и ожидалось seq. number равен 1310762944, а ack. number равен 1845932619, теперь клиент ожидает от сервера пакет с seq. number равным 1845932619.

```
C:\Windows\system32\cmd.exe - java client.Client
C:\study\s08\Серв\task_1\simple\client\src>java client.Client
hello
echo: hello
```

Рис. 1.31: Консоль клиента

```
C:\Windows\system32\cmd.exe - java server.Server
K:\>java server.Server
Server started.
New connection accepted from: /192.168.0.104:51898
```

Рис. 1.32: Консоль сервера

## Разрыв соединения

Будет рассмотрен случай при разрыве соединения со стороны сервера.

1	0.000000	192.168.0.102	192.168.0.104	TCP	60	8090 → 52078 [FIN, ACK] Seq=150525472 Ack=1443653468 Win=256 Len=0
2	0.000002	192.168.0.102	192.168.0.104	TCP	60	[TCP Out-Of-Order] 8090 → 52078 [FIN, ACK] Seq=150525472 Ack=1443653468 Win=256 Len=0
3	0.000098	192.168.0.104	192.168.0.102	TCP	54	52078 → 8090 [ACK] Seq=1443653468 Ack=150525473 Win=64 Len=0
4	0.001959	192.168.0.104	192.168.0.102	TCP	54	52078 → 8090 [FIN, ACK] Seq=1443653468 Ack=150525473 Win=64 Len=0
5	0.004684	192.168.0.102	192.168.0.104	TCP	60	8090 → 52078 [ACK] Seq=150525473 Ack=1443653469 Win=256 Len=0

Рис. 1.33: Пакеты между сервером и клиентом

Возможно это связано с особенностями спроектированной программы, но 1 и 2 пакеты из рисунка 1.33 идентичны, за исключением контрольных сумм в заголовке пакета IPv4.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.102	192.168.0.104	TCP	60	8090 → 52078 [FIN, ACK] Seq=150525472 Ack=1443653468 Win=256 Len=0
2	0.000002	192.168.0.102	192.168.0.104	TCP	60	[TCP Out-Of-Order] 8090 → 52078 [FIN, ACK] Seq=150525472 Ack=1443653468
3	0.000098	192.168.0.104	192.168.0.102	TCP	54	52078 → 8090 [ACK] Seq=1443653468 Ack=150525473 Win=64 Len=0

> Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0  
 > Ethernet II, Src: Giga-Byt\_24:4a:24 (94:de:80:24:4a:24), Dst: HonHaiPr\_49:6d:bd (14:2d:27:49:6d:bd)  
 > Internet Protocol Version 4, Src: 192.168.0.102, Dst: 192.168.0.104  
 > Transmission Control Protocol, Src Port: 8090, Dst Port: 52078, Seq: 150525472, Ack: 1443653468, Len: 0
 

Source Port: 8090  
 Destination Port: 52078  
 [Stream index: 0]  
 [TCP Segment Len: 0]  
 Sequence number: 150525472  
 Acknowledgment number: 1443653468  
 Header Length: 20 bytes  
 > Flags: 0x011 (FIN, ACK)  
 Window size value: 256  
 [Calculated window size: 256]  
 [Window size scaling factor: -1 (unknown)]  
 Checksum: 0xa52a [unverified]  
 [Checksum Status: Unverified]  
 Urgent pointer: 0

Рис. 1.34: Первый пакет от сервера

Сервер посылает клиенту пакет с установленными флагами ACK, FIN. Сервер переходит из состояния ESTABLISHED в состояние FIN-WAIT-1.

No.	Time	Source	Destination	Protocol	Length	Info
3	0.000098	192.168.0.104	192.168.0.102	TCP	54	52078 → 8090 [ACK] Seq=1443653468 Ack=150525473 Win=64 Len=0
4	0.001959	192.168.0.104	192.168.0.102	TCP	54	52078 → 8090 [FIN, ACK] Seq=1443653468 Ack=150525473 Win=64 Len=0
5	0.004684	192.168.0.102	192.168.0.104	TCP	60	8090 → 52078 [ACK] Seq=150525473 Ack=1443653469 Win=256 Len=0

> Frame 3: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0  
 > Ethernet II, Src: HonHaiPr\_49:6d:bd (14:2d:27:49:6d:bd), Dst: Giga-Byt\_24:4a:24 (94:de:80:24:4a:24)  
 > Internet Protocol Version 4, Src: 192.168.0.104, Dst: 192.168.0.102  
 > Transmission Control Protocol, Src Port: 52078, Dst Port: 8090, Seq: 1443653468, Ack: 150525473, Len: 0
 

Source Port: 52078  
 Destination Port: 8090  
 [Stream index: 0]  
 [TCP Segment Len: 0]  
 Sequence number: 1443653468  
 Acknowledgment number: 150525473  
 Header Length: 20 bytes  
 > Flags: 0x010 (ACK)  
 Window size value: 64  
 [Calculated window size: 64]  
 [Window size scaling factor: -1 (unknown)]  
 Checksum: 0xa5ea [unverified]  
 [Checksum Status: Unverified]  
 Urgent pointer: 0  
 > [SEQ/ACK analysis]

Рис. 1.35: Первый пакет от клиента

Клиент при получении пакета переходит из состояния ESTABLISHED в состояние CLOSE-WAIT. И посылает в ответ два пакета. Первый пакет содержит флаг ACK, после принятия пакета сервером, он переходит в состояние FIN-WAIT-2.

No.	Time	Source	Destination	Protocol	Length	Info
3	0.000098	192.168.0.104	192.168.0.102	TCP	54	52078 → 8090 [ACK] Seq=1443653468 Ack=150525473 Win=64 Len=0
4	0.001959	192.168.0.104	192.168.0.102	TCP	54	52078 → 8090 [FIN, ACK] Seq=1443653468 Ack=150525473 Win=64 Len=0
5	0.004684	192.168.0.102	192.168.0.104	TCP	60	8090 → 52078 [ACK] Seq=150525473 Ack=1443653469 Win=256 Len=0

```

> Frame 4: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
> Ethernet II, Src: HonHaiPr_49:6d:bd (14:2d:27:49:6d:bd), Dst: Giga-Byt_24:4a:24 (94:de:80:24:4a:24)
> Internet Protocol Version 4, Src: 192.168.0.104, Dst: 192.168.0.102
> Transmission Control Protocol, Src Port: 52078, Dst Port: 8090, Seq: 1443653468, Ack: 150525473, Len: 0
    Source Port: 52078
    Destination Port: 8090
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 1443653468
    Acknowledgment number: 150525473
    Header Length: 20 bytes
> Flags: 0x011 (FIN, ACK)
    Window size value: 64
    [Calculated window size: 64]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0xa5e9 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0

```

Рис. 1.36: Второй пакет от клиента

Клиент посылает второй пакет с флагами FIN, ACK, после его отсылки клиент переходит в состояние LAST-ACK, а сервер в состояние TIME-WAIT.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.001959	192.168.0.104	192.168.0.102	TCP	54	52078 → 8090 [FIN, ACK] Seq=1443653468 Ack=150525473 Win=64 Len=0
5	0.004684	192.168.0.102	192.168.0.104	TCP	60	8090 → 52078 [ACK] Seq=150525473 Ack=1443653469 Win=256 Len=0

```

> Frame 5: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
> Ethernet II, Src: Giga-Byt_24:4a:24 (94:de:80:24:4a:24), Dst: HonHaiPr_49:6d:bd (14:2d:27:49:6d:bd)
> Internet Protocol Version 4, Src: 192.168.0.102, Dst: 192.168.0.104
> Transmission Control Protocol, Src Port: 8090, Dst Port: 52078, Seq: 150525473, Ack: 1443653469, Len: 0
    Source Port: 8090
    Destination Port: 52078
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 150525473
    Acknowledgment number: 1443653469
    Header Length: 20 bytes
> Flags: 0x010 (ACK)
    Window size value: 256
    [Calculated window size: 256]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0xa529 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
> [SEQ/ACK analysis]

```

Рис. 1.37: Второй пакет от сервера

Сервер посылает клиенту пакет с флагом ACK, и переходит в состояние CLOSED для данного соединения. Клиент так-же при принятии пакета переходит в состояние CLOSED.

### Попытка соединения на отсутствующий порт

При попытке подключения к отсутствующему порту, от адреса, к которому происходит подключение, приходят пакеты с флагами ACK и RST. После трех попыток соединения, написанная программа сообщает об ошибке.

No.	Time	Source	Destination	Protocol	Length	Info
3	0.377574	192.168.0.104	192.168.0.102	TCP	66	52200 → 8090 [SYN] Seq=1781921335 Win=8192 Len=0 MSS=1460 WS=256 SACK_PE
4	0.379098	192.168.0.102	192.168.0.104	TCP	60	8090 → 52200 [RST, ACK] Seq=0 Ack=1781921336 Win=0 Len=0
5	0.878909	192.168.0.104	192.168.0.102	TCP	66	[TCP Spurious Retransmission] 52200 → 8090 [SYN] Seq=1781921335 Win=8192
6	0.883076	192.168.0.102	192.168.0.104	TCP	60	8090 → 52200 [RST, ACK] Seq=0 Ack=1781921336 Win=0 Len=0
7	1.385534	192.168.0.104	192.168.0.102	TCP	62	[TCP Spurious Retransmission] 52200 → 8090 [SYN] Seq=1781921335 Win=8192
8	1.386997	192.168.0.102	192.168.0.104	TCP	60	8090 → 52200 [RST, ACK] Seq=0 Ack=1781921336 Win=0 Len=0

> Frame 4: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0  
 > Ethernet II, Src: Giga-Byt\_24:4a:24 (94:de:80:24:4a:24), Dst: HonHaiPr\_49:6d:bd (14:2d:27:49:6d:bd)  
 > Internet Protocol Version 4, Src: 192.168.0.102, Dst: 192.168.0.104  
 > Transmission Control Protocol, Src Port: 8090, Dst Port: 52200, Seq: 0, Ack: 1781921336, Len: 0  
     Source Port: 8090  
     Destination Port: 52200  
     [Stream index: 0]  
     [TCP Segment Len: 0]  
     Sequence number: 0  
     Acknowledgment number: 1781921336  
     Header Length: 20 bytes  
     > Flags: 0x014 (RST, ACK)  
         Window size value: 0  
         [Calculated window size: 0]  
         [Window size scaling factor: -1 (unknown)]  
         Checksum: 0xe1c0 [unverified]  
         [Checksum Status: Unverified]  
         Urgent pointer: 0  
     > [SEQ/ACK analysis]

Рис. 1.38: Попытка tcp - соединения на 192.168.0.102:8090

```

C:\Windows\system32\cmd.exe
C:\study\s08\Сети\task_1\simple\client\src>java client.Client
Couldn't get I/O for the connection to 192.168.0.102

```

Рис. 1.39: Окно консоли при попытке соединения

## 1.5 Вывод

В ходе работы был исследован сетевой трафик утилит ping и tracert а также протоколов ICMP, ARP, TCP и UDP.

При выполнении работы были рассмотрены различные ситуации, возникающие во время функционирования сети, такие как:

- Работа ICMP-протокола при:
  - отправке фрагментированного ping'a,
  - возникновение ошибки 3.1 (Destination host unreachable),
  - трассировка маршрута.
- работа протокола TCP при:
  - установке соединения,
  - разрыве соединения,
  - попытке соединения на отсутствующий порт.

Для вышеизложенных ситуаций, использование какой-либо программы по анализу сетевого трафика, позволяет более подробно рассмотреть происходящие при этом события.

## Приложение 1

```
1 #include <winsock2.h>
2 #include <fstream>
3
4 #define SERVER "192.168.56.2"
5 #define PORT 8005
6
7 using namespace std;
8
9 int clientSocket;
10 sockaddr_in si_server;
11 int slen=sizeof(si_server);
12
13 int main(int argc, char** argv) {
14     WSADATA wsa;
15
16     if(WSAStartup(MAKEWORD(2,2),&wsa)!=0){
17         printf("Failed WSAS initialize. Error code :%d",WSAGetLastError
↵  ());
18         exit(EXIT_FAILURE);
19     }
20
21     if((clientSocket=socket(AF_INET,SOCK_DGRAM,IPPROTO_UDP))==
↵  SOCKET_ERROR){
22         printf("Socket() failed with error code: %d",WSAGetLastError());
23         exit(EXIT_FAILURE);
24     }
25
26     memset((char *)&si_server,0,sizeof(si_server));
27     si_server.sin_family=AF_INET;
28     si_server.sin_port=htons(PORT);
29     si_server.sin_addr.S_un.S_addr=inet_addr(SERVER);
30
31     string rez="testData";
32     sendto(clientSocket,rez.c_str(),strlen(rez.c_str()), 0,(struct
↵  sockaddr *)&si_server,slen);
33
34     return 0;
35 }
```

Листинг 11: udp.cpp

## Приложение 2

```
1 package server;
2
3 public class Server{
4     public static void main(String[] args) {
5         ServerThread st=new ServerThread();
6         new Thread(st).start();
7     }
8 }
```

```
7     }
8 }
```

Листинг 1.2: Server.java

## Приложение 3

```
1 package server;
2
3 import java.io.BufferedReader;
4 import java.net.ServerSocket;
5 import java.net.Socket;
6 import java.io.IOException;
7 import java.io.InputStreamReader;
8 import java.net.InetAddress;
9 import java.util.ArrayList;
10 import java.util.Map;
11 import java.util.Scanner;
12 import java.util.concurrent.ConcurrentHashMap;
13 import java.util.logging.Level;
14 import java.util.logging.Logger;
15
16 public class ServerThread implements Runnable{
17     protected int          SERVER_PORT = 8090;
18     protected String       SERVER_IP   = "127.0.0.1";
19
20     protected ServerSocket serverSocket = null;
21     protected boolean      isStopped   = false;
22
23     private ConcurrentHashMap<clientThread, String> mClients;
24
25     Scanner s = new Scanner(System.in);
26
27     public void run(){
28         Runnable server = new Runnable() {
29             @Override
30             public void run() {
31                 while(!isStopped){
32                     String input = s.next();
33                     switch (input) {
34                         case "show":
35                             showClients();
36                             break;
37                         case "kill":
38                             int id=Integer.parseInt(s.next());
39                             deleteClientById(id);
40                             break;
41                         case "stop":
42                             stop();
43                             break;
44                     }
45                 }
46             }
47         };
48         Thread t = new Thread(server);
49         t.start();
50     }
51 }
```

```

46         }
47     };
48     new Thread(server).start();
49     mClients = new ConcurrentHashMap<>();
50     openServerSocket();
51
52     while(!isStopped){
53         Socket clientSocket = null;
54         try {
55             clientSocket = this.serverSocket.accept();
56         } catch (IOException e) {
57             System.out.println("Can't accept socket.");
58             return;
59         }
60         addClient(clientSocket);
61     }
62 }
63
64 private void addClient(Socket clientSocket){
65     clientThread CT=new clientThread(clientSocket, this);
66     CT.start();
67     synchronized (mClients) {
68         mClients.put(CT, "");
69     }
70 }
71 private synchronized void deleteClientById(int id){
72     int i=1;
73     for (Map.Entry<clientThread,String> entry : mClients.entrySet())
↪ {
74         clientThread key = entry.getKey();
75         if(i==id){
76             key.finish();
77             mClients.remove(key);
78             return;
79         }
80         i++;
81     }
82     System.out.println("No clients with this id");
83 }
84 public synchronized void removeClient(clientThread CT){
85     synchronized (mClients) {
86         mClients.remove(CT);
87     }
88 }
89 private synchronized void showClients(){
90     int i=1;
91     for (Map.Entry<clientThread,String> entry : mClients.entrySet())
↪ {
92         clientThread key = entry.getKey();
93         System.out.println(i+"|"+key.getAddr());
94         i++;
95     }
96 }

```

```

97
98     public synchronized void stop(){
99         for (Map.Entry<clientThread,String> entry : mClients.entrySet())
↪     {
100             clientThread key = entry.getKey();
101             key.finish();
102             mClients.remove(key);
103         }
104         this.isStopped = true;
105         try {
106             this.serverSocket.close();
107             System.out.println("Server stopped.");
108         } catch (IOException e) {
109             System.out.println("Error closing server. Error: "+e);
110         }
111     }
112
113     private void openServerSocket() {
114         try {
115             this.serverSocket = new ServerSocket(this.SERVER_PORT,0,
↪     InetAddress.getBy_name(SERVER_IP));
116         } catch (IOException e) {
117             throw new RuntimeException("Cannot open port 8080", e);
118         }
119         System.out.println("Server started.") ;
120     }
121 }

```

Листинг 1.3: ServerThread.java

## Приложение 4

```

1 package server;
2
3 import java.io.*;
4 import java.net.Socket;
5 import java.net.SocketAddress;
6 import java.util.HashMap;
7 import java.util.Map;
8 import java.util.concurrent.ConcurrentHashMap;
9 import java.util.logging.Level;
10 import java.util.logging.Logger;
11
12 public class clientThread extends Thread {
13     protected Socket clientSocket = null;
14     private SocketAddress remoteAddr = null;
15     private boolean finish=false;
16     private ServerThread ST;
17
18     public clientThread(Socket clientSocket, ServerThread ST) {
19         this.clientSocket = clientSocket;
20         this.ST=ST;

```



```

21         remoteAddr = this.clientSocket.getRemoteSocketAddress();
22     }
23
24     public void run() {
25         try{
26             System.out.println("New connection accepted from: " +
↪ remoteAddr);
27             PrintWriter out = new PrintWriter(clientSocket.
↪ getOutputStream(), true);
28             BufferedReader in = new BufferedReader(new InputStreamReader
↪ (clientSocket.getInputStream()));
29
30
31             String inputLine;
32             while (!finish && (inputLine = in.readLine()) != null) {
33                 out.println(inputLine);
34             }
35             }catch (IOException e) {
36                 ST.removeClient(this);
37                 System.out.println("Disconnected: " + remoteAddr +" with "+e
↪ .getMessage());
38                 try {
39                     clientSocket.close();
40                     // System.out.println("Socket closed.");
41                 } catch (IOException ex) {
42                     Logger.getLogger(clientThread.class.getName()).log(Level
↪ .SEVERE, null, ex);
43                 }
44             }
45         }
46
47         public void finish(){
48             try{
49                 clientSocket.close();
50             }catch(final IOException e){
51                 System.out.println("Can't close socket. Error: "+e.
↪ getMessage());
52             }
53         }
54
55         public String getAddr(){
56             return remoteAddr.toString();
57         }
58     }

```

Листинг 14: clientThread.java

## Приложение 5

```

1 package client;
2
3 import java.io.*;

```

```

4 import java.net.*;
5 import java.util.logging.Level;
6 import java.util.logging.Logger;
7
8 public class Client {
9
10     private static int SERVER_PORT=8090;
11     private static String SERVER_IP="192.168.0.102";
12     private static boolean work=true;
13
14     public static void main(String[] args) {
15         try{
16             Socket echoSocket = new Socket(SERVER_IP, SERVER_PORT);
17
18             PrintWriter out = new PrintWriter(echoSocket.getOutputStream
↵ (), true);
19             BufferedReader in = new BufferedReader(new InputStreamReader
↵ (echoSocket.getInputStream()));
20             BufferedReader stdIn = new BufferedReader(new
↵ InputStreamReader(System.in));
21
22
23
24             Runnable consoleReader = new Runnable() {
25                 @Override
26                 public void run() {
27                     while(work){
28                         try {
29                             String userInput = stdIn.readLine();
30                             out.println(userInput);
31                         } catch (IOException ex) {
32                             ↵ Logger.getLogger(Client.class.getName()).log
↵ (Level.SEVERE, null, ex);
33                         }
34                     }
35                 }
36             };
37             new Thread(consoleReader).start();
38
39             while (work) {
40                 String echo=in.readLine();
41                 if(echo==null){
42                     work=false;
43                     System.out.println("You are disconnected.");
44                 }else
45                     System.out.println("echo: " + echo);
46             }
47             echoSocket.close();
48             } catch (UnknownHostException e) {
49                 System.err.println("Don't know about " + SERVER_IP);
50                 System.exit(1);
51             } catch (IOException e) {

```

```
52         System.err.println("Couldn't get I/O for the connection to "
    ↪      + SERVER_IP);
53         System.exit(1);
54     }
55 }
56 }
```

Листинг 1.5: Client.java