

Федеральное государственное автономное образовательное
учреждение высшего образования
Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники
Направление подготовки 09.03.04 Программная инженерия
Дисциплина «Компьютерные сети»

Отчет по лабораторной работе №4
«Анализ трафика компьютерных сетей
с помощью утилиты Wireshark»

Студент:
Барсуков Максим Андреевич,
группа Р3315

Преподаватель:
Тропченко Андрей Александрович

Оглавление

Задание.....	2
Цель работы.....	2
Вариант.....	2
Анализ трафика утилиты ping.....	3
Выполнение.....	3
Ответы на вопросы:.....	5
Анализа трафика утилиты tracert (traceroute).....	7
Выполнение.....	7
Ответы на вопросы:.....	8
Анализ HTTP-трафика.....	11
Выполнение.....	11
Анализ ARP-трафика.....	13
Выполнение.....	13
Ответы на вопросы:.....	14
Вывод.....	16

Задание

Цель работы

Целью данной лабораторной работы является изучение структуры протокольных блоков данных, путем анализа реального трафика на компьютере студента с помощью бесплатно распространяемой утилиты Wireshark.

В процессе выполнения домашнего задания выполняются наблюдения за передаваемым трафиком с компьютера пользователя в Интернет и в обратном направлении. Применение специализированной утилиты Wireshark позволяет наблюдать структуру передаваемых кадров, пакетов и сегментов данных различных сетевых протоколов. При выполнении УИР рекомендуется выполнить анализ последовательности команд и определить назначение служебных данных, используемых для организации обмена данными в протоколах: ARP, DNS, FTP, HTTP, DHCP.

Вариант

Для выполнения лабораторной работы будут представлены пункты 4.1, 4.2, **4.3**, **4.5**.

Сайт для анализа трафика – www.bma.org.uk.

Анализ трафика утилиты ping

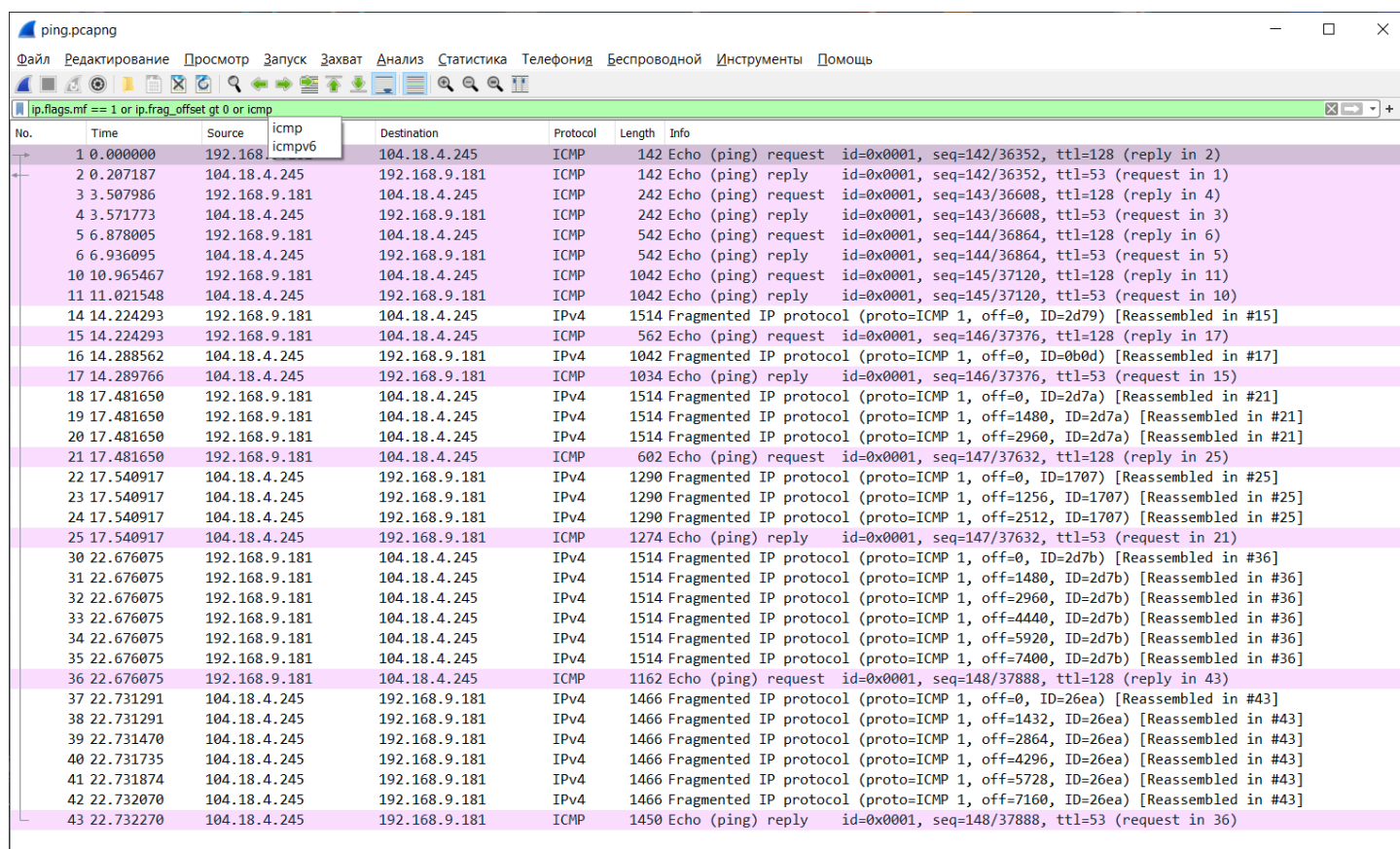
Выполнение

В командной строке поочередно с увеличением размера будем отправлять пакеты через утилиту ping на сайт www.bma.org.uk. Формат команды:

ping -l <размер пакета> -n <кол-во пакетов> www.bma.org.uk

Опция **-n** нужна для того, чтобы отправлять один пакет, так как сама по себе утилита по умолчанию отправляет 4 пакета.

Применим данную команду несколько раз для случаев, когда размер у пакета будет 100, 200, 500, 1000, 2000, 5000, 10000 байт.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.9.181	104.18.4.245	ICMP	142	Echo (ping) request id=0x0001, seq=142/36352, ttl=128 (request in 2)
2	0.207187	104.18.4.245	192.168.9.181	ICMP	142	Echo (ping) reply id=0x0001, seq=142/36352, ttl=53 (request in 1)
3	3.507986	192.168.9.181	104.18.4.245	ICMP	242	Echo (ping) request id=0x0001, seq=143/36608, ttl=128 (request in 4)
4	3.571773	104.18.4.245	192.168.9.181	ICMP	242	Echo (ping) reply id=0x0001, seq=143/36608, ttl=53 (request in 3)
5	6.878005	192.168.9.181	104.18.4.245	ICMP	542	Echo (ping) request id=0x0001, seq=144/36864, ttl=128 (request in 6)
6	6.936095	104.18.4.245	192.168.9.181	ICMP	542	Echo (ping) reply id=0x0001, seq=144/36864, ttl=53 (request in 5)
10	10.965467	192.168.9.181	104.18.4.245	ICMP	1042	Echo (ping) request id=0x0001, seq=145/37120, ttl=128 (request in 11)
11	11.021548	104.18.4.245	192.168.9.181	ICMP	1042	Echo (ping) reply id=0x0001, seq=145/37120, ttl=53 (request in 10)
14	14.224293	192.168.9.181	104.18.4.245	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=2d79) [Reassembled in #15]
15	14.224293	192.168.9.181	104.18.4.245	ICMP	562	Echo (ping) request id=0x0001, seq=146/37376, ttl=128 (request in 17)
16	14.288562	104.18.4.245	192.168.9.181	IPv4	1042	Fragmented IP protocol (proto=ICMP 1, off=0, ID=0b0d) [Reassembled in #17]
17	14.289766	104.18.4.245	192.168.9.181	ICMP	1034	Echo (ping) reply id=0x0001, seq=146/37376, ttl=53 (request in 15)
18	17.481650	192.168.9.181	104.18.4.245	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=2d7a) [Reassembled in #21]
19	17.481650	192.168.9.181	104.18.4.245	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=2d7a) [Reassembled in #21]
20	17.481650	192.168.9.181	104.18.4.245	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=2d7a) [Reassembled in #21]
21	17.481650	192.168.9.181	104.18.4.245	ICMP	602	Echo (ping) request id=0x0001, seq=147/37632, ttl=128 (request in 25)
22	17.540917	104.18.4.245	192.168.9.181	IPv4	1290	Fragmented IP protocol (proto=ICMP 1, off=0, ID=1707) [Reassembled in #25]
23	17.540917	104.18.4.245	192.168.9.181	IPv4	1290	Fragmented IP protocol (proto=ICMP 1, off=1256, ID=1707) [Reassembled in #25]
24	17.540917	104.18.4.245	192.168.9.181	IPv4	1290	Fragmented IP protocol (proto=ICMP 1, off=2512, ID=1707) [Reassembled in #25]
25	17.540917	104.18.4.245	192.168.9.181	ICMP	1274	Echo (ping) reply id=0x0001, seq=147/37632, ttl=53 (request in 21)
30	22.676075	192.168.9.181	104.18.4.245	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=2d7b) [Reassembled in #36]
31	22.676075	192.168.9.181	104.18.4.245	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=2d7b) [Reassembled in #36]
32	22.676075	192.168.9.181	104.18.4.245	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=2d7b) [Reassembled in #36]
33	22.676075	192.168.9.181	104.18.4.245	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=4440, ID=2d7b) [Reassembled in #36]
34	22.676075	192.168.9.181	104.18.4.245	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=5920, ID=2d7b) [Reassembled in #36]
35	22.676075	192.168.9.181	104.18.4.245	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=7400, ID=2d7b) [Reassembled in #36]
36	22.676075	192.168.9.181	104.18.4.245	ICMP	1162	Echo (ping) request id=0x0001, seq=148/37888, ttl=128 (request in 43)
37	22.731291	104.18.4.245	192.168.9.181	IPv4	1466	Fragmented IP protocol (proto=ICMP 1, off=0, ID=26ea) [Reassembled in #43]
38	22.731291	104.18.4.245	192.168.9.181	IPv4	1466	Fragmented IP protocol (proto=ICMP 1, off=1432, ID=26ea) [Reassembled in #43]
39	22.731470	104.18.4.245	192.168.9.181	IPv4	1466	Fragmented IP protocol (proto=ICMP 1, off=2864, ID=26ea) [Reassembled in #43]
40	22.731735	104.18.4.245	192.168.9.181	IPv4	1466	Fragmented IP protocol (proto=ICMP 1, off=4296, ID=26ea) [Reassembled in #43]
41	22.731874	104.18.4.245	192.168.9.181	IPv4	1466	Fragmented IP protocol (proto=ICMP 1, off=5728, ID=26ea) [Reassembled in #43]
42	22.732070	104.18.4.245	192.168.9.181	IPv4	1466	Fragmented IP protocol (proto=ICMP 1, off=7160, ID=26ea) [Reassembled in #43]
43	22.732270	104.18.4.245	192.168.9.181	ICMP	1450	Echo (ping) reply id=0x0001, seq=148/37888, ttl=53 (request in 36)

Для начала опишем структуру пакета. Утилита управляет ICMP запросами и ICMP ответами. Структура:

1. Канальный уровень – Ethernet 2

Заголовок содержит:

- Destination MAC address – MAC адрес получателя.
- Source MAC address – MAC-адрес отправителя.
- Type – поле типа протокола.

2. Сетевой уровень – IP-заголовок

Заголовок содержит:

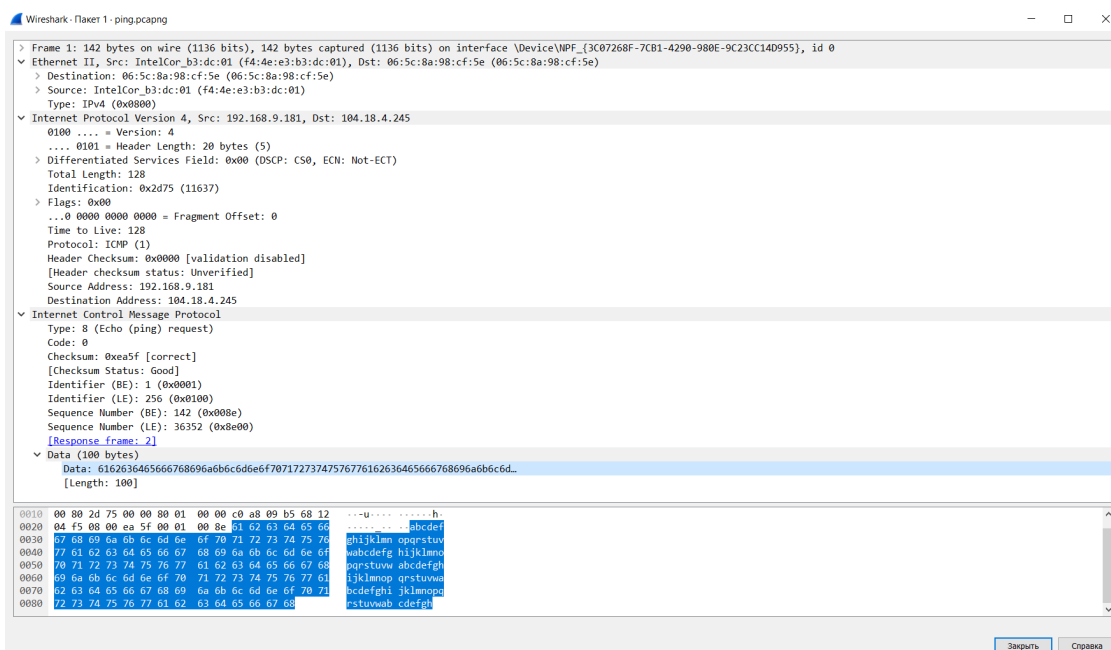
- Version
- Header Length
- Identification – идентификатор фрагмента
- Protocol – тип вложенного протокола
- Flags – указывается DF и MF
- TTL – ограничение на кол-во хопов
- Fragment offset – смещение фрагмента (если пакет был фрагментирован)
- Header Checksum – контрольная сумма заголовка
- Source IP address
- Destination IP address

3. Сетевой протокол ISMP

- Type – request или reply
- Checksum – контрольная сумма ICMP-пакета
- Identifier – уникальный ID запроса
- Seq number – номер последовательности запроса

4. Поле данных (Payload)

Примерно так выглядит структура пакета ISMP:



Ответы на вопросы:

1. Имеет ли место фрагментация исходного пакета, какое поле на это указывает?

Фрагментация происходит, когда размер IP-пакета превышает MTU (maximum transmission unit) (обычно 1480 байт для Ethernet). Признаком фрагментации служат:

- Флаг MF (More Fragments) в IP-заголовке
- Поле Fragment Offset (смещение фрагмента)

```
[Stream Index: 0]
▼ Internet Protocol Version 4, Src: 192.168.31.108, Dst: 172.67.73.26
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 548
  Identification: 0xb65a (46682)
  ▶ 000. .... = Flags: 0x0
  ...0 0000 1011 1001 = Fragment Offset: 1480
```

```
▼ Internet Protocol Version 4, Src: 192.168.31.108, Dst: 172.67.73.26
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 1500
  Identification: 0xb65a (46682)
  ▼ 001. .... = Flags: 0x1, More fragments
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
    ...0 0000 0000 0000 = Fragment Offset: 0
```

Заметим, что когда пакет фрагментируется, то часть данных отправляется вместе с ICMP заголовком, а остальные фрагменты чисто по протоколу IP.

2. Какая информация указывает, является ли фрагмент пакета последним или промежуточным?

- MF = 1 – промежуточный фрагмент.
- MF = 0 – последний фрагмент.

```
.0.. .... = Don't fragment: Not set
..1. .... = More fragments: Set
```

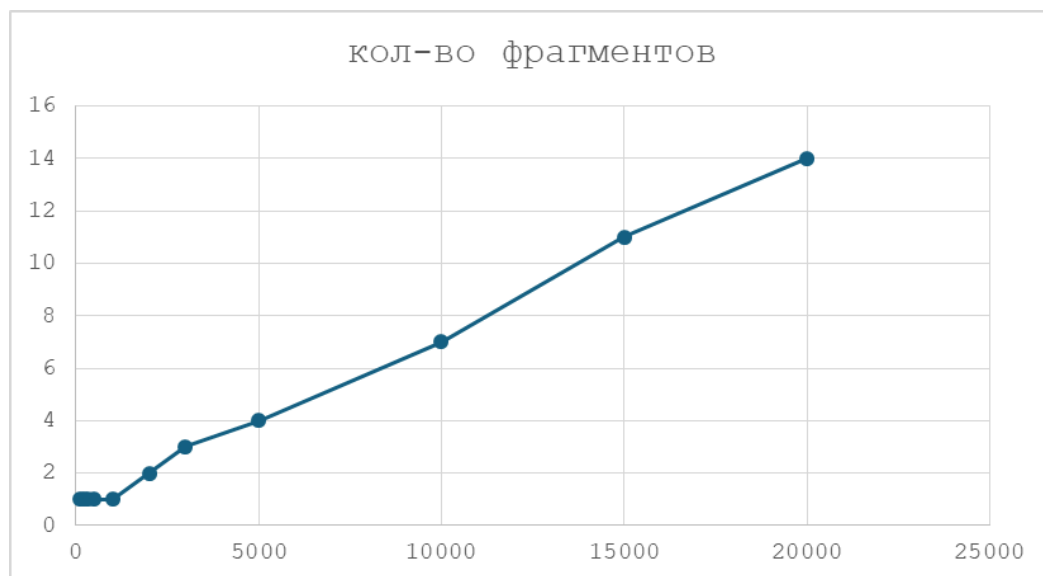
```
.0.. .... = Don't fragment: Not set
..0. .... = More fragments: Not set
```

3. Чему равно количество фрагментов при передаче ping-пакетов?

Учитывая, что один фрагмент по MTU равен примерно 1480 байт, кол-во фрагментов будет равно размеру пакета / 1480 и округлить до верхнего целого числа.

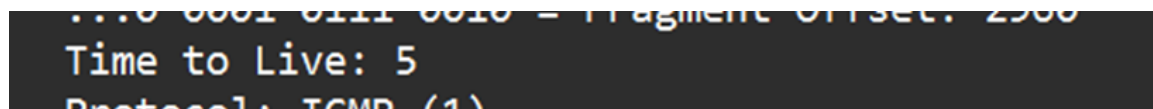
No.	Time	Source	Destination	Protocol	Length	Info
72	3.133666	192.168.31.108	172.67.73.26	ICMP	142	Echo (ping) request id=0x0001, seq=87/22272, ttl=128 (reply in 73)
73	3.148458	172.67.73.26	192.168.31.108	ICMP	142	Echo (ping) reply id=0x0001, seq=87/22272, ttl=55 (request in 72)
137	12.379758	192.168.31.108	172.67.73.26	ICMP	242	Echo (ping) request id=0x0001, seq=88/22528, ttl=128 (reply in 140)
140	12.398336	172.67.73.26	192.168.31.108	ICMP	242	Echo (ping) reply id=0x0001, seq=88/22528, ttl=55 (request in 137)
170	16.805644	192.168.31.108	172.67.73.26	ICMP	542	Echo (ping) request id=0x0001, seq=89/22784, ttl=128 (reply in 171)
171	16.820949	172.67.73.26	192.168.31.108	ICMP	542	Echo (ping) reply id=0x0001, seq=89/22784, ttl=55 (request in 170)
206	21.312734	192.168.31.108	172.67.73.26	ICMP	1042	Echo (ping) request id=0x0001, seq=90/23040, ttl=128 (reply in 207)
207	21.332380	172.67.73.26	192.168.31.108	ICMP	1042	Echo (ping) reply id=0x0001, seq=90/23040, ttl=55 (request in 206)
274	24.791832	192.168.31.108	172.67.73.26	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=b65a) [Reassembled in #275]
275	24.791832	192.168.31.108	172.67.73.26	ICMP	562	Echo (ping) request id=0x0001, seq=91/23296, ttl=128 (reply in 278)
277	24.810418	172.67.73.26	192.168.31.108	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=b65a) [Reassembled in #278]
278	24.810418	172.67.73.26	192.168.31.108	ICMP	562	Echo (ping) reply id=0x0001, seq=91/23296, ttl=55 (request in 275)
396	27.572940	192.168.31.108	172.67.73.26	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=b65b) [Reassembled in #399]
397	27.572940	192.168.31.108	172.67.73.26	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=b65b) [Reassembled in #399]
398	27.572940	192.168.31.108	172.67.73.26	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=b65b) [Reassembled in #399]
399	27.572940	192.168.31.108	172.67.73.26	ICMP	602	Echo (ping) request id=0x0001, seq=92/23552, ttl=128 (no response found!)
1234	37.321956	192.168.31.108	172.67.73.26	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=b65c) [Reassembled in #1240]
1235	37.321956	192.168.31.108	172.67.73.26	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=b65c) [Reassembled in #1240]
1236	37.321956	192.168.31.108	172.67.73.26	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=b65c) [Reassembled in #1240]
1237	37.321956	192.168.31.108	172.67.73.26	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=4440, ID=b65c) [Reassembled in #1240]
1238	37.321956	192.168.31.108	172.67.73.26	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=5920, ID=b65c) [Reassembled in #1240]
1239	37.321956	192.168.31.108	172.67.73.26	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=7400, ID=b65c) [Reassembled in #1240]
1240	37.321956	192.168.31.108	172.67.73.26	ICMP	1162	Echo (ping) request id=0x0001, seq=93/23808, ttl=128 (no response found!)

4. График: размер пакета – кол-во фрагментов.



5. Как изменить поле TTL с помощью утилиты ping?

Изменить это поле можно командой: `ping -l 3000 -n 1 -i 5 www.bma.org.uk`



6. Что содержится в поле данных ping-пакета

- Заголовок ICMP;
- Идентификатор;
- Номер последовательности;
- Содержимое.

Анализа трафика утилиты `tracert` (`traceroute`)

Выполнение

Вводим в командную строку команду:

`tracert www.bma.org.uk`

```
max@laptop ~ → tracert www.bma.org.uk
Трассировка маршрута к www.bma.org.uk.cdn.cloudflare.net [104.18.5.245]
с максимальным числом прыжков 30:
 1  1 ms  <1 ms  <1 ms  192.168.1.1
 2  2 ms  1 ms  1 ms  100.83.168.1
 3  4 ms  2 ms  2 ms  AGG-R.GW2.sknt.ru [93.100.0.81]
 4  2 ms  2 ms  2 ms  185.37.128.61
 5  *      *      *      Превышен интервал ожидания для запроса.
 6  *      *      *      Превышен интервал ожидания для запроса.
 7  2 ms  2 ms  3 ms  spb-r2-cr1.ae54-2153.rascom.as20764.net [80.64.103.130]
 8  *      *      *      Превышен интервал ожидания для запроса.
 9  *      16 ms  *      80.64.108.35.rascom.as20764.net [80.64.108.35]
10 22 ms  12 ms  22 ms  172.68.180.37
11 12 ms  12 ms  12 ms  104.18.5.245
Трассировка завершена.
```

```
max@laptop ~ → tracert -d www.bma.org.uk
Sword Art Online
Трассировка маршрута к www.bma.org.uk.cdn.cloudflare.net [104.18.5.245]
с максимальным числом прыжков 30:
 1  1 ms  1 ms  1 ms  192.168.1.1
 2  1 ms  1 ms  1 ms  100.83.168.1
 3  1 ms  1 ms  1 ms  93.100.0.81
 4  1 ms  1 ms  1 ms  185.37.128.61
 5  *      *      *      Превышен интервал ожидания для запроса.
 6  *      *      *      Превышен интервал ожидания для запроса.
 7  2 ms  2 ms  3 ms  80.64.103.130
 8  *      *      *      Превышен интервал ожидания для запроса.
 9  12 ms  12 ms  12 ms  80.64.108.35
10 12 ms  85 ms  12 ms  172.68.180.37
11 13 ms  12 ms  12 ms  104.18.5.245
Трассировка завершена.
```

Данная утилита также пользуется протоколом ICMP, поэтому разбирать его структуру мы не будем. Но, помимо этого, утилита `tracert` отправляет DNS-пакеты. DNS – это протокол, который переводит доменные имена в IP-адреса, которые понятны компьютерам. С помощью ключа `-d` можно сделать так, чтобы DNS пакеты отправлялись уже после построения маршрута, так как они не несут в себе важный функционал.

Вот структура DNS пакета:

```
▼ Domain Name System (query)
  Transaction ID: 0x2719
  ▼ Flags: 0x0100 Standard query
    0... .. = Response: Message is a query
    .000 0... .. = Opcode: Standard query (0)
    ....0... .. = Truncated: Message is not truncated
    ....1... .. = Recursion desired: Do query recursively
    ....0... .. = Z: reserved (0)
    ....0... .. = Non-authenticated data: Unacceptable
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 0
  ▼ Queries
    ▼ www.bma.org.uk: type A, class IN
      Name: www.bma.org.uk
      [Name Length: 14]
      [Label Count: 4]
      Type: A (1) (Host Address)
      Class: IN (0x0001)
      [Response In: 13]
```


Заголовок размером 12 байт содержит:

- ID – уникальный ID-запроса;
- Flags – ошибки, авторитетность, тип запроса/ответа;
- QDCOUNT – кол-во запросов;
- ANCOUNT – кол-во ответов;
- INSCOUNT – кол-во записей авторитетных серверов;
- ARCOUNT – кол-во дополнительных записей.

Раздел вопросов, который содержит доменное имя, которое мы запрашиваем.

Раздел ответов, который содержит IP-адрес в ответ на запрос.

Ответы на вопросы:

1. Сколько байт содержится в заголовке IP? Сколько содержится в поле данных?

Заголовок IP обычно составляет 20 байт для IPv4.

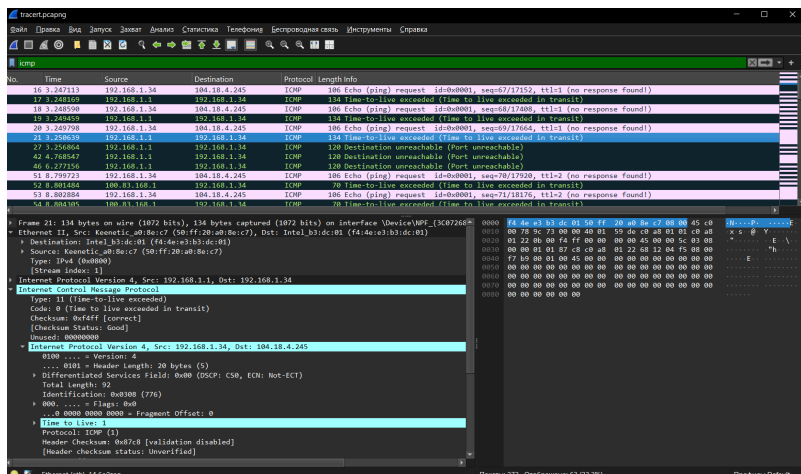
```
▼ Internet Protocol Version 4, Src: 192.168.1.34, Dst: 104.166.182.205
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 40
  Identification: 0x4af5 (19189)
```

Поле данных – это содержимое, инкапсулированное в IP-пакете, ICMP пакета.

У ICMP заголовок равен 8 байт, а сами данные 64 байта. Следовательно, данные 72 байта. IP-заголовок 20 байт.

2. Как и почему изменяется поле TTL в следующих ICMP-пакетах tracer?

Утилита traceroute посылает ICMP-пакеты с увеличивающимся TTL, начиная с 1. Каждый маршрутизатор уменьшает TTL на 1. Когда TTL становится 0 – маршрутизатор отбрасывает пакет и отправляет обратно ICMP Time Exceeded. Это позволяет traceroute определить каждый узел на пути. TTL изменяется поэтапно, чтобы каждый узел по очереди откликнулся, и таким образом строится маршрут.



3. Чем отличаются ICMP-пакеты, генерируемые tracer, от ICMP-пакетов ping?

Ping всегда шлёт ICMP Echo Request и ждёт Echo Reply. Tracer использует ICMP Echo Request с разным TTL и анализирует:

- ICMP Time Exceeded от промежуточных маршрутизаторов.
- ICMP Echo Reply от конечного узла.

То есть ping проверяет доступность узла, а tracer строит маршрут до него.

4. Чем отличаются ICMP reply от ICMP error и зачем нужны оба?

ICMP reply – отклик от целевого хоста, подтверждающий, что он доступен.

ICMP error – приходит от маршрутизаторов, когда TTL истекает. Эти пакеты нужны для определения маршрута.

Оба типа позволяют tracer:

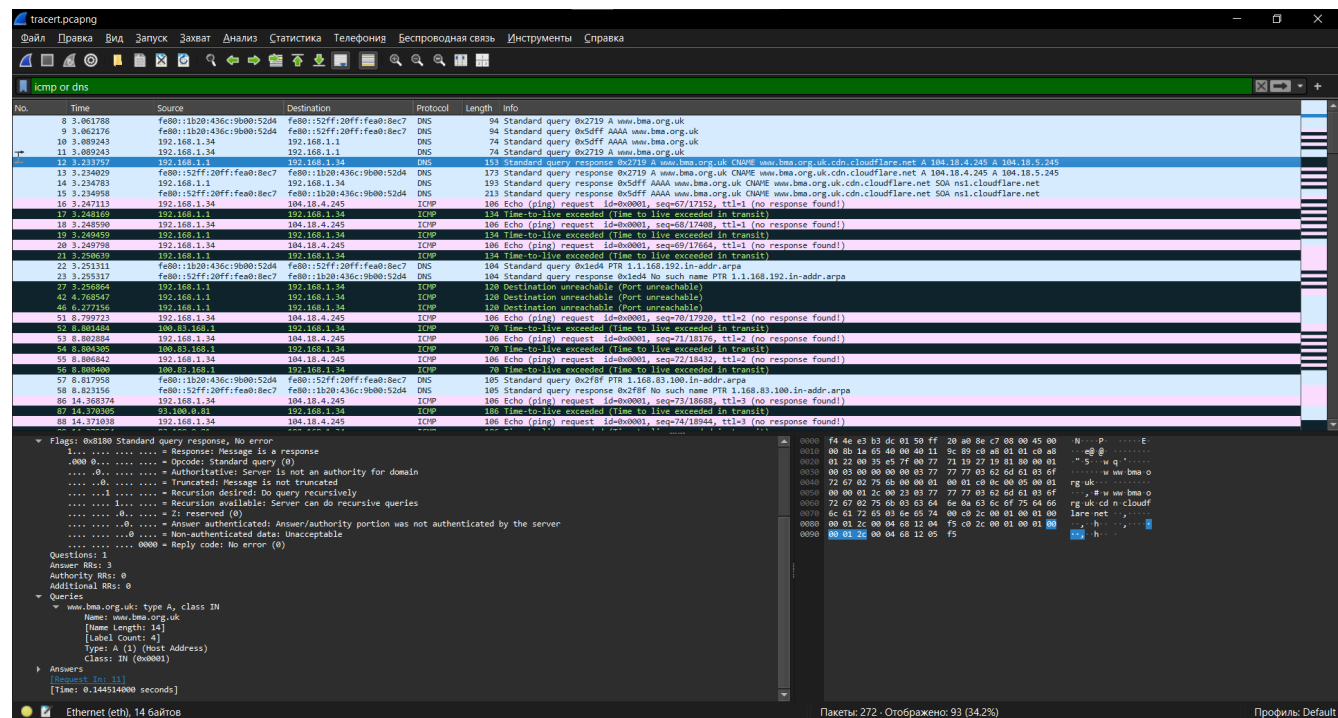
- Узнать IP каждого промежуточного маршрутизатора (через error).
- Подтвердить достижение конечного узла (через reply).

```
Internet Control Message Protocol
Type: 8 (Echo (ping) request)
Code: 0
Checksum: 0xf7b9 [correct]
[Checksum Status: Good]
Identifier (BE): 1 (0x0001)
Identifier (LE): 256 (0x0100)
Sequence Number (BE): 69 (0x0045)
Sequence Number (LE): 17664 (0x4500)
```

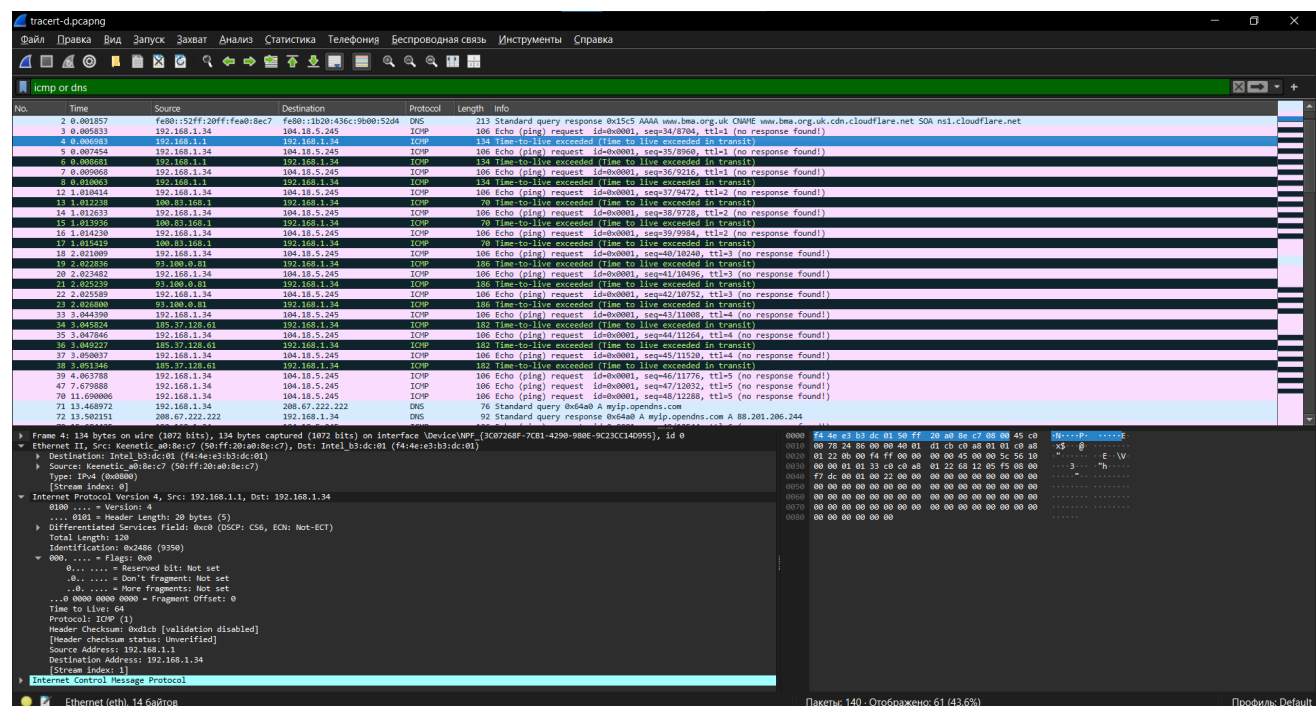
5. Что изменится в работе tracer, если убрать ключ -d? Какой трафик будет генерироваться дополнительно?

Ключ -d отключает обратное разрешение IP-адресов в доменные имена. Без -d tracer будет пытаться разрешить IP-адреса в имена хостов (через DNS). Это приведёт к дополнительному DNS-трафику, так как каждый IP будет запрашиваться у DNS-сервера.

Вот окно wireshark без ключа -d:



Вот окно wireshark с ключом -d:

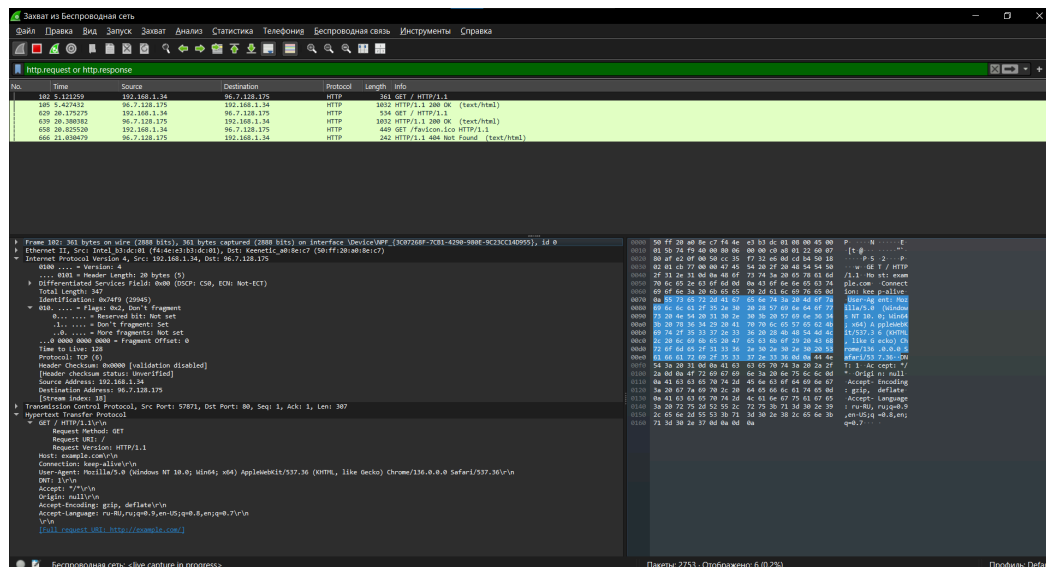


Анализ HTTP-трафика

Выполнение

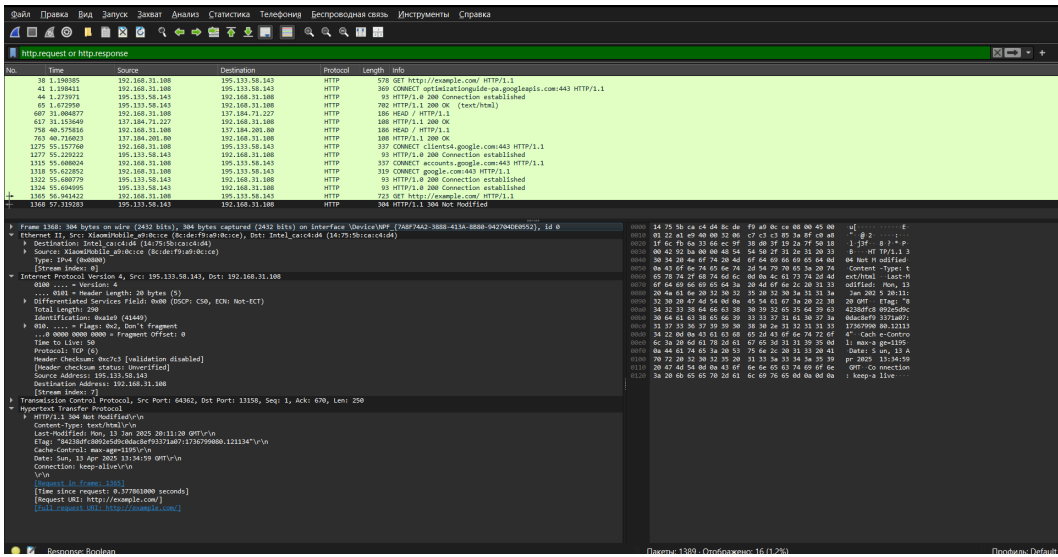
Запустим анализ в Wireshark и перейдём на сайт www.bma.org.uk. К сожалению, сайт, который подходит нам по варианту, мало того, что не обладает возможностью принимать условные GET-запросы, так еще и запрещает обращения по HTTP вместо HTTPS. Сколько раз не обновляй мы не можем получить ответ 304. Поэтому воспользуемся сайтом, который точно обладает такой возможностью, а именно сайтом example.com.

Сначала просто зайдём на сайт example.com.



Заметим, что все отработало как надо, мы получаем ответ 200.

Теперь попробуем обновить страницу и посмотрим, что будет.



Content-Type: text/html\r\n
Last-Modified: Mon, 13 Jan 2025 20:11:20 GMT\r\n

If-None-Match: "84238dfc8092e5d9c0dac8ef93371a07:1736799080.121134"\r\n
If-Modified-Since: Mon, 13 Jan 2025 20:11:20 GMT\r\n
\r\n

Заметим, что мы получаем совсем другую ситуацию. Здесь у нас получилось отправить условный GET-запрос. И мы получаем ответ 304 от сервера. Это можно понять по появившимся полям Last-Modified и If-Modified-Since.

Анализ ARP-трафика

Выполнение

Для начала очистим ARP-таблицу с помощью команды:

netsh interface ip delete arpcache

```
max@laptop ~ → netsh interface ip delete arpcache
OK.
```

Получим вот такую ARP-таблицу:

```
max@laptop ~ → arp -a

Интерфейс: 192.168.1.34 --- 0x6
  адрес в Интернете      Физический адрес      Тип
  224.0.0.2              01-00-5e-00-00-02      статический
  224.0.0.22             01-00-5e-00-00-16      статический

Интерфейс: 192.168.137.1 --- 0xb
  адрес в Интернете      Физический адрес      Тип
  224.0.0.22             01-00-5e-00-00-16      статический

Интерфейс: 25.74.63.167 --- 0x15
  адрес в Интернете      Физический адрес      Тип
  224.0.0.22             01-00-5e-00-00-16      статический
max@laptop ~ →
```

После удаления кэша браузера отправимся на сайт www.bma.org.uk и увидим новую запись в ARP-таблице.

```
max@laptop ~ → arp -a

Интерфейс: 192.168.1.34 --- 0x6
  адрес в Интернете      Физический адрес      Тип
  192.168.1.1           50-ff-20-a0-8e-c7      динамический
  224.0.0.2              01-00-5e-00-00-02      статический
  224.0.0.22             01-00-5e-00-00-16      статический
  224.0.0.251           01-00-5e-00-00-fb      статический

Интерфейс: 192.168.137.1 --- 0xb
  адрес в Интернете      Физический адрес      Тип
  224.0.0.22             01-00-5e-00-00-16      статический
  224.0.0.251           01-00-5e-00-00-fb      статический

Интерфейс: 25.74.63.167 --- 0x15
  адрес в Интернете      Физический адрес      Тип
  224.0.0.22             01-00-5e-00-00-16      статический
  224.0.0.251           01-00-5e-00-00-fb      статический
```

Заметим, что это вообще не похоже на IP адрес сайта, на который мы перешли. А всё, потому что MAC-адреса используются только в локальной сети. Мы не сможем увидеть ARP-запрос, который узнаёт MAC-адрес нашего сайта, так как его и вовсе нет. Но мы видим IP-адрес 192.168.1.1. Вероятнее всего это IP нашего маршрутизатора, который как раз таки и взялся в дальнейшем уже за поиск того сайта, на который мы перешли.

Файл Правка Вид Запуск Захват Анализ Статистика Телефония Беспроводная связь Инструменты Справка						
arp						
No.	Time	Source	Destination	Protocol	Length	Info
25	8.359594	Intel_b3:dc:01	Broadcast	ARP	42	Who has 192.168.1.1? Tell 192.168.1.34
26	8.360532	Keenetic_a0:8e:c7	Intel_b3:dc:01	ARP	42	192.168.1.1 is at 50:ff:20:a0:8e:c7
4130	13.496848	Keenetic_a0:8e:c7	Intel_b3:dc:01	ARP	42	Who has 192.168.1.34? Tell 192.168.1.1
4131	13.496867	Intel_b3:dc:01	Keenetic_a0:8e:c7	ARP	42	192.168.1.34 is at f4:4e:e3:b3:dc:01
10602	17.101971	Keenetic_a0:8e:c7	Broadcast	ARP	42	Who has 192.168.1.142? Tell 192.168.1.1

<p>Frame 25: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF_{3C07...}</p> <p>Ethernet II, Src: Intel_b3:dc:01 (f4:4e:e3:b3:dc:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)</p> <ul style="list-style-type: none"> Destination: Broadcast (ff:ff:ff:ff:ff:ff) Source: Intel_b3:dc:01 (f4:4e:e3:b3:dc:01) Type: ARP (0x0806) [Stream index: 1] <p>Address Resolution Protocol (request)</p> <ul style="list-style-type: none"> Hardware type: Ethernet (1) Protocol type: IPv4 (0x0800) Hardware size: 6 Protocol size: 4 Opcode: request (1) Sender MAC address: Intel_b3:dc:01 (f4:4e:e3:b3:dc:01) Sender IP address: 192.168.1.34 Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00) Target IP address: 192.168.1.1 	<pre> 0000 ff ff ff ff ff f4 4e e3 b3 dc 01 08 06 00 01N 0010 08 00 06 04 00 01 f4 4e e3 b3 dc 01 c0 a8 01 22N 0020 00 00 00 00 00 00 c0 a8 01 01 </pre>
---	---

Ответы на вопросы:

1. Какие MAC-адреса присутствуют в захваченных пакетах ARP-протокола? Что означают эти адреса? Какие устройства они идентифицируют?

В ARP-пакетах мы увидим два типа MAC-адресов:

- MAC-адрес отправителя запроса – адрес нашего компьютера. Он используется в поле Sender MAC-address
- MAC-адрес искомого устройства:
 - В ARP-запросе (who-has) поле Target MAC Address будет заполнено нулями, потому что он ещё известен.
 - В ARP-ответе (is-at) это будет MAC-адрес шлюза/маршрутизатора, провайдера или другого узла локальной сети, связанного с IP, на который отправляется запрос.

2. Какие MAC-адреса присутствуют в захваченных HTTP-пакетах и что означают эти адреса? Что означают эти адреса? Какие устройства они идентифицируют?

No.	Time	Source	Destination	Protocol	Length	Info
25	8.359594	Intel_b3:dc:01	Broadcast	ARP	42	Who has 192.168.1.1? Tell 192.168.1.34
26	8.360532	Keenetic_a0:8e:c7	Intel_b3:dc:01	ARP	42	192.168.1.1 is at 50:ff:20:a0:8e:c7
4130	13.496848	Keenetic_a0:8e:c7	Intel_b3:dc:01	ARP	42	Who has 192.168.1.34? Tell 192.168.1.1
4131	13.496867	Intel_b3:dc:01	Keenetic_a0:8e:c7	ARP	42	192.168.1.34 is at f4:4e:e3:b3:dc:01
10602	17.101971	Keenetic_a0:8e:c7	Broadcast	ARP	42	Who has 192.168.1.142? Tell 192.168.1.1

▶ Frame 26: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF_{3C07...}	0000	f4 4e e3 b3 dc 01 50 ff 20 a0 8e c7 08 06 00 01	·N···P···
▼ Ethernet II, Src: Keenetic_a0:8e:c7 (50:ff:20:a0:8e:c7), Dst: Intel_b3:dc:01 (f4:4e:e3:b3:dc:01)	0010	08 00 06 04 00 02 50 ff 20 a0 8e c7 c0 a8 01 01	·····P···
▶ Destination: Intel_b3:dc:01 (f4:4e:e3:b3:dc:01)	0020	f4 4e e3 b3 dc 01 c0 a8 01 22	·N·····"
▶ Source: Keenetic_a0:8e:c7 (50:ff:20:a0:8e:c7)			
Type: ARP (0x0806)			
[Stream index: 0]			
▼ Address Resolution Protocol (reply)			
Hardware type: Ethernet (1)			
Protocol type: IPv4 (0x0800)			
Hardware size: 6			
Protocol size: 4			
Opcode: reply (2)			
Sender MAC address: Keenetic_a0:8e:c7 (50:ff:20:a0:8e:c7)			
Sender IP address: 192.168.1.1			
Target MAC address: Intel_b3:dc:01 (f4:4e:e3:b3:dc:01)			
Target IP address: 192.168.1.34			

HTTP работает поверх TCP/IP и Ethernet. В Ethernet-заголовке каждого HTTP-пакета указывается:

- MAC-адрес источника – это MAC-адрес нашего компьютера
- MAC-адрес назначения – это обычно MAC-адрес ближайшего маршрутизатора/шлюза, через который трафик пойдёт в Интернет.

MAC-адреса веб-сайта, на который мы заходим, мы не увидим, потому что MAC-адреса используются только внутри локальной сети.

3. Для чего ARP-запрос содержит IP-адрес источника?

ARP-запроса содержит IP-адрес источника, чтобы:

- Получатель запроса (тот, чей IP адрес запрашивается) мог записать в свою ARP-таблицу соответствие, и тем самым сократить количество ARP-запросов в будущем.
- Получатель понимал, кто запрашивает – это нужно для формирования ARP-запроса-ответа.

IP-адрес источника нужен для обратной связи и корректного построения локальной маршрутизации.

Вывод

В ходе лабораторной работы с помощью программы Wireshark был проведен анализ передачи пакетов по сети, а также была описана структура DNS, ICMP, IP, ARP и HTTP протоколов. Я изучил, какие пакеты передаются при работе утилит ping, и traceroute и какую информацию они содержат. Также был проведен анализ трафика HTTP-запросов и влияние на него кэширования данных. Кэширование также влияет на работу DNS, во время выполнения работы нам необходимо было очистить кэши и посмотреть на работу DNS-запросов. Далее был рассмотрен трафик при выполнении ARP-запросов, для этого нужно было очистить ARP-таблицу. В результате я выяснил, что передача по сети это сложный механизм, который включает в себя взаимодействие большого количества протоколов и интерфейсов.