

Национальный исследовательский университет ИТМО
Факультет программной инженерии и компьютерной техники

Учебно-исследовательская работа №3
по дисциплине Сети ЭВМ и телекоммуникации
Анализ трафика компьютерных сетей утилитой Wireshark

Студент: Саржевский Иван
Группа: Р3302

г. Санкт-Петербург
2020 г.

Содержание

1	Цель	2
2	Анализ трафика утилиты ping	2
2.1	Фрейм	2
2.2	Ethernet II	2
2.3	IPv4	3
2.4	Internet Control Message Protocol (ICMP)	4
2.5	Ответы на вопросы	5
3	Анализ трафика утилиты traceroute	6
3.1	User Datagram Protocol	6
3.2	ICMP в ответах на запросы	7
3.3	Ответы на вопросы	8
4	Анализ HTTP-трафика	8
4.1	Transmission Control Protocol	9
4.2	Hypertext Transfer Protocol	9

1 Цель

Цель работы – изучить структуру протокольных блоков данных, анализируя реальный трафик на компьютере студента с помощью бесплатно распространяемой утилиты Wireshark.

2 Анализ трафика утилиты ping

Для анализа трафика, создаваемого утилитой ping был выбран сайт **www.ias.ru**.

```
Frame 5: 1042 bytes on wire (8336 bits), 1042 bytes captured (8336 bits) on interface wlp4s0, id 0
Ethernet II, Src: Chongqin_64:e6:c5 (c0:b5:d7:64:e6:c5), Dst: Tp-LinkT_3d:06:ae (cc:32:e5:3d:06:ae)
Internet Protocol Version 4, Src: 192.168.0.105, Dst: 81.195.71.197
Internet Control Message Protocol
```

Рис. 1: Заголовки протоколов для команды ping.

На рисунке 1 изображены заголовки различных протоколов, используемых при передаче запроса.

2.1 Фрейм

```
Interface id: 0 (wlp4s0)
Encapsulation type: Ethernet (1)
Arrival Time: Apr 12, 2020 22:23:51.094101026 MSK
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1586719431.094101026 seconds
[Time delta from previous captured frame: 0.000253948 seconds]
[Time delta from previous displayed frame: 0.000000000 seconds]
[Time since reference or first frame: 0.036999160 seconds]
Frame Number: 5
Frame Length: 1042 bytes (8336 bits)
Capture Length: 1042 bytes (8336 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ethertype:ip:icmp:data]
[Coloring Rule Name: ICMP]
[Coloring Rule String: icmp || icmpv6]
```

Рис. 2: Информация о фрейме команды ping.

Структура, представленная на рисунке 2, описывает метаданные Wireshark для этого запроса - его порядковый номер среди всех записанных, время прибытия, размер, протокол и цвет выделения в интерфейсе.

2.2 Ethernet II

Ethernet II - протокол канального уровня, т.е. описывает передачу данных в рамках локальной сети. Типичная структура кадра Ethernet II представлена в таблице 1.

Таблица 1: Структура кадра Ethernet II.

Кадр Ethernet II (от 64-х до 1528-ти байт)				
MAC-заголовок (14 байт)			Данные (от 46-ти до 1500 байт)	–
MAC получателя (6 байт)	MAC отправителя (6 байт)	Тип протокола (2 байта)	Данные	CRC (4 байта)

В данном случае получателем выступает роутер, а отправителем - рабочая машина, их MAC-адреса записаны в кадр, тип протокола - IPv4, что можно увидеть на рисунке 3.

```

Destination: Tp-LinkT_3d:06:ae (cc:32:e5:3d:06:ae)
Address: Tp-LinkT_3d:06:ae (cc:32:e5:3d:06:ae)
....0. .... = LG bit: Globally unique address (factory default)
....0. .... = IG bit: Individual address (unicast)
Source: Chongqin_64:e6:c5 (c0:b5:d7:64:e6:c5)
Address: Chongqin_64:e6:c5 (c0:b5:d7:64:e6:c5)
....0. .... = LG bit: Globally unique address (factory default)
....0. .... = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)

```

Рис. 3: Кадр Ethernet II для ping.

2.3 IPv4

IPv4 - протокол сетевого уровня. Подробные сведения полей, которые включены в заголовок протокола, приведены на рисунке 4. Туда включены IP-адреса отправителя и получателя, длина заголовка и сообщения, флаги указывающие на наличие фрагментации данных, промежуточности данного пакета и т. д.

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification															Flags			Fragment Offset													
8	64	Time To Live								Protocol							Header Checksum																
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															
24	192																																
28	224																																
32	256																																

Рис. 4: Структура заголовка IPv4.

Данные, переданные с использованием протокола IPv4 для команды ping можно увидеть на рисунке 5.

```

0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  0000 00.. = Differentiated Services Codepoint: Default (0)
  .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
Total Length: 1028
Identification: 0x8a07 (35335)
Flags: 0x4000, Don't fragment
  0... .... = Reserved bit: Not set
  .1... .... = Don't fragment: Set
  ..0. .... = More fragments: Not set
...0 0000 0000 0000 = Fragment offset: 0
Time to live: 64
Protocol: ICMP (1)
Header checksum: 0x5258 [validation disabled]
[Header checksum status: Unverified]
Source: 192.168.0.105
Destination: 81.195.71.197
[Destination GeoIP: RU]

```

Рис. 5: Данные пакета IPv4 для команды ping.

2.4 Internet Control Message Protocol (ICMP)

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Type								Code								Checksum															
4	32	Rest of Header																															

Рис. 6: Структура заголовка ICMP.

Данный протокол сетевого уровня используется для передачи служебных сообщений - кода ошибки в случае исключительной ситуации, кода запрашиваемой операции и кода подтверждения в случае удачной передачи. Подробная структура заголовка ICMP приведена на рисунке 6.

```

Type: 8 (Echo (ping) request)
Code: 0
Checksum: 0xb113 [correct]
[Checksum Status: Good]
Identifier (BE): 4975 (0x136f)
Identifier (LE): 28435 (0x6f13)
Sequence number (BE): 1 (0x0001)
Sequence number (LE): 256 (0x0100)
[Response frame: 6]
Timestamp from icmp data: Apr 12, 2020 22:23:51.000000000 MSK
[Timestamp from icmp data (relative): 0.094101026 seconds]
Data (992 bytes)

```

Рис. 7: Данные ICMP для команды ping.

Для команды ping структура ICMP представлена на рисунке 7.

Структура ответов имеет схожую структуру, отличаться они будут типом ICMP, сменой адресов получателя и отправителя, timestamp'ами.

2.5 Ответы на вопросы

1. Имеет ли место фрагментация исходного пакета, какое поле на это указывает?
– Да. Но только в том случае, если размер пакета превышает **Maximum Transmission Unit (MTU)**, равный для протокола **Ethernet II** 1500 байт. Информация о наличии фрагментации содержится во флаге в заголовке **IPv4**.
2. Какая информация указывает, является ли фрагмент пакета последним или промежуточным?
– Флаг **More Fragments** в заголовке **IPv4**.
3. Чему равно количество фрагментов при передаче ping-пакетов?
– MTU равен 1500 байт, пакет включает в себя **IPv4**-заголовок (20 байт), **ICMP**-заголовок (8 байт), и, непосредственно, данные. Это означает, что количество фрагментов равно $\lceil (s + 20 + 8) / 1500 \rceil$, где s - аргумент **-s** команды **ping**. Зависимость количества фрагментов от размера пакета приведена в таблице 2.

Таблица 2: Количество фрагментов при разных размерах пакета

Размер пакета	100	500	1000	1500	2000	3000	5000	10000
Кол-во фраг.	1	1	1	2	2	3	4	7

4. Построить график, в котором на оси абсцисс находится размер_пакета, а по оси ординат – количество фрагментов, на которое был разделён каждый **ping**-пакет.
– см. рисунок 8.

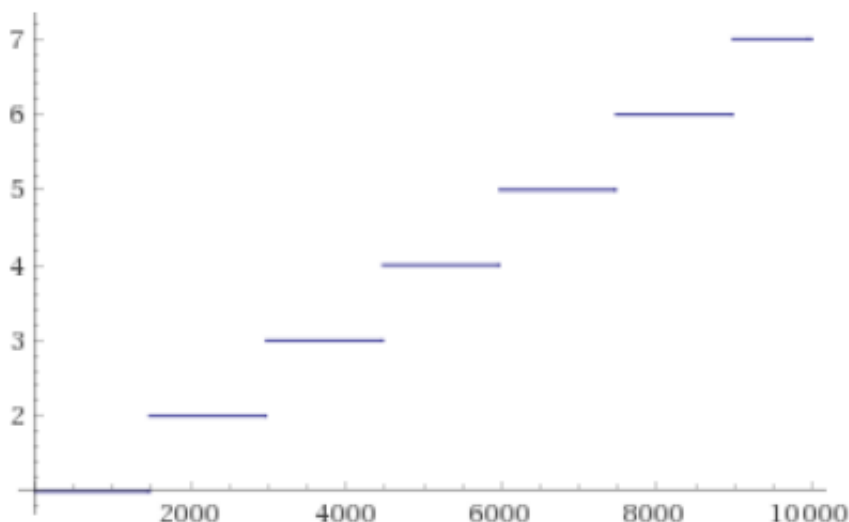


Рис. 8: Зависимость количества фрагментов от размера пакета

5. Как изменить поле TTL с помощью утилиты **ping**?

- Linux : **ping -t ttl_value**

- Windows : `ping -i ttl_value`

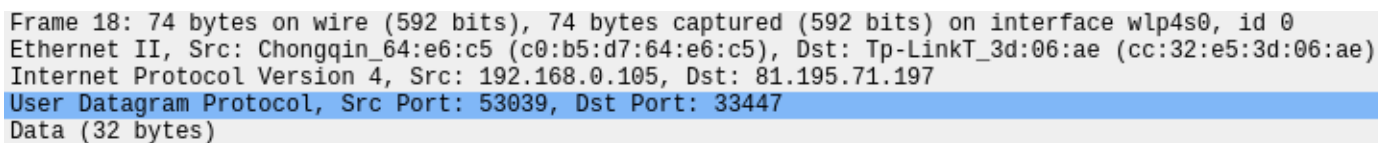
6. Что содержится в поле данных `ping`-пакета?

– В поле данных содержится текущий `timestamp`, а затем циклически повторяющиеся биты от `00` до `FF`.

```
0030 00 00 5d 12 04 00 00 00 00 00 10 11 12 13 14 15
0040 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25
0050 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35
0060 36 37 38 39 3a 3b 3c 3d 3e 3f 40 41 42 43 44 45
. . .
0110 e6 e7 e8 e9 ea eb ec ed ee ef f0 f1 f2 f3 f4 f5
0120 f6 f7 f8 f9 fa fb fc fd fe FF 00 01 02 03 04 05
```

3 Анализ трафика утилиты `tracert`

Утилита `tracert` отправляет UDP-запросы, постепенно увеличивая `ttl`, на 1 каждые 3 запроса.

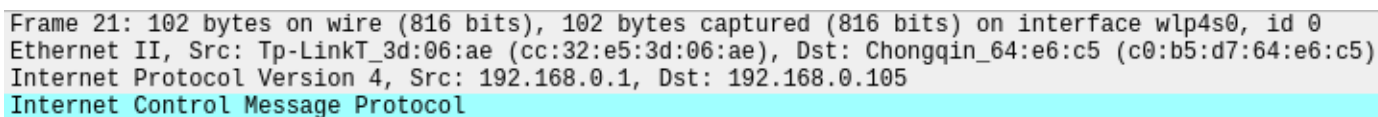


```
Frame 18: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlp4s0, id 0
Ethernet II, Src: Chongqin_64:e6:c5 (c0:b5:d7:64:e6:c5), Dst: Tp-LinkT_3d:06:ae (cc:32:e5:3d:06:ae)
Internet Protocol Version 4, Src: 192.168.0.105, Dst: 81.195.71.197
User Datagram Protocol, Src Port: 53039, Dst Port: 33447
Data (32 bytes)
```

Рис. 9: Заголовки протоколов для команды `tracert`.

На рисунке 9 изображены заголовки различных протоколов, используемых при передаче запроса.

Все протоколы, кроме UDP, были описаны для команды `ping`.



```
Frame 21: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface wlp4s0, id 0
Ethernet II, Src: Tp-LinkT_3d:06:ae (cc:32:e5:3d:06:ae), Dst: Chongqin_64:e6:c5 (c0:b5:d7:64:e6:c5)
Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.105
Internet Control Message Protocol
```

Рис. 10: Заголовки протоколов для ответа на команду `tracert`.

На рисунке 10 изображены заголовки протоколов, используемых при передаче ответа на запросы команды `tracert`.

3.1 User Datagram Protocol

UDP - один из протоколов транспортного уровня, который предполагает отсутствие механизмов установления и поддержки соединения между отправителем и получателем. Структура UDP-датаграммы представлена в таблице 3.

На рисунке 11 можно увидеть содержимое датаграммы для запроса `tracert`.

Таблица 3: Структура UDP-датаграммы.

Порт отправителя	Порт получателя	Длина датаграммы	Данные
------------------	-----------------	------------------	--------

```
Source Port: 53039
Destination Port: 33447
Length: 40
Checksum: 0x5ad3 [unverified]
[Checksum Status: Unverified]
[Stream index: 14]
[Timestamps]
```

Рис. 11: Содержимое датаграммы.

3.2 ICMP в ответах на запросы

Как уже говорилось ранее, ICMP передает служебную информацию, такую как сообщения об ошибках. Именно эту его особенность использует утилита **traceroute**. Так как мы постепенно увеличиваем **ttl** начиная с единицы, каждый маршрутизатор в сети между отправителем и получателем будет возвращать ошибку **Time-to-live exceeded** (код 11), и по отправителям этих сообщений можно судить о маршрутизаторах в сети на пути пакетов. Пример такого ICMP можно увидеть на рисунке 12.

```
Type: 11 (Time-to-live exceeded)
Code: 0 (Time to live exceeded in transit)
Checksum: 0x4fd3 [correct]
[Checksum Status: Good]
Unused: 00000000
Internet Protocol Version 4, Src: 192.168.0.105, Dst: 81.195.71.197
User Datagram Protocol, Src Port: 46331, Dst Port: 33434
Data (32 bytes)
```

Рис. 12: ICMP с сообщением **ttl-exceeded**

Если же мы установили достаточно большой **ttl** для того, чтобы добраться до получателя, в ICMP с большой вероятностью будет другая ошибка – **Destination unreachable** (код 3) с расширением **Port unreachable** (код 3), так как **traceroute** отправляет запросы на случайные порты получателя. Пример такого ICMP можно увидеть на рисунке 13.

В обоих случаях в ответном сообщении содержится копия исходного.


```
Type: 3 (Destination unreachable)
Code: 3 (Port unreachable)
Checksum: 0x57d0 [correct]
[Checksum Status: Good]
Unused: 00000000
Internet Protocol Version 4, Src: 192.168.0.105, Dst: 81.195.71.197
User Datagram Protocol, Src Port: 33829, Dst Port: 33488
Data (32 bytes)
```

Рис. 13: ICMP с сообщением **Destination unreachable**

3.3 Ответы на вопросы

1. Сколько байт содержится в заголовке IP? Сколько байт содержится в поле данных?
 - IPv4 - 20 байт, данных отправляется 32 байта. Если прибавить к этому 8 байт заголовка UDP и 14 байт заголовка Ethernet II, то получим 74 байта, это размер передаваемого сообщения.
2. Как и почему изменяется поле TTL в следующих друг за другом ICMP-пакетах traceroute?
 - Как уже было упомянуто выше, это позволяет по сообщениям с ошибкой **Time-to-live exceeded** определить все маршрутизаторы, которые работали с пакетом по пути до получателя. **ttl** увеличивается на 1 каждые 3 отправленных пакета.
3. Чем отличаются ICMP-пакеты, генерируемые утилитой **tracert**, от ICMP-пакетов, генерируемых утилитой **ping**?
 - Использовалась утилита **tracert**, которая в отличие от **tracert** отправляет UDP запросы. Однако при использовании последней ICMP пакеты от **ping** отличаются только значением **ttl**.
4. Чем отличаются полученные пакеты ICMP **reply** от ICMP **error** и зачем нужны оба этих типа ответов?
 - В **tracert** в качестве ICMP **reply** используется ICMP **error**, но с другим кодом ошибки - **Destination unreachable (Port unreachable)**. Если говорить о **tracert**, то ICMP **reply**, получаемые при достижении пакетов получателя, ничем не отличаются от аналогичных в утилите **ping**. ICMP **error** используются для идентификации всех маршрутизаторов на пути пакета, это описано в предыдущем вопросе.
5. Что изменится в работе **tracert**, если убрать ключ **-d**? Какой дополнительный трафик при этом будет генерироваться?
 - **tracert** станет преобразовывать IP адреса маршрутизаторов в их строковые адреса, а для этого потребуются дополнительные DNS запросы. **ИСПРАВИТЬ**

4 Анализ HTTP-трафика

Для выполнения этого задания был выбран сайт **www.ias.ru**.

На рисунке 14 изображена структура запроса, все протоколы, кроме TCP и HTTP были рассмотрены ранее.

```

Frame 3: 525 bytes on wire (4200 bits), 525 bytes captured (4200 bits) on interface wlp4s0, id 0
Ethernet II, Src: Chongqin_64:e6:c5 (c0:b5:d7:64:e6:c5), Dst: Tp-LinkT_3d:06:ae (cc:32:e5:3d:06:ae)
Internet Protocol Version 4, Src: 192.168.0.105, Dst: 81.195.71.197
Transmission Control Protocol, Src Port: 42758, Dst Port: 80, Seq: 1, Ack: 1, Len: 459
Hypertext Transfer Protocol

```

Рис. 14: Структура запроса

4.1 Transmission Control Protocol

Это протокол транспортного уровня, один из основных протоколов передачи данных в интернете. TCP предполагает надежную передачу потока данных, включает управление перегрузкой, рукопожатие, передачу данных. Структура TCP-протокола приведена на рисунке 15. Она включает в себя информацию о портах отправителя и получателя, размер заголовка, а также контрольную сумму и флаги:

- URG : указатель важности
- ACK : номер подтверждения
- PSH : протолкнуть данные, накопившиеся в буфере, в приложение пользователя
- RST : оборвать соединение, очистить буфер
- SYN : синхронизация объектов последовательности
- FIN : завершение соединения

Offsets	Octet	0								1								2								3							
Octet	Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0			N S	C W R	E C E	U R G	A C K	P S H	R S T	S Y N	F I N	Window Size															
16	128	Checksum																Urgent pointer (if URG set)															
20	160	Options (if <i>data offset</i> > 5. Padded at the end with "0" bytes if necessary.)																															
...																															

Рис. 15: Структура TCP-сегмента.

Структура переданного сегмента приведена на рисунке 16.

4.2 Hypertext Transfer Protocol

Это протокол прикладного уровня для передачи данных. Состоит из стартовой строки, заголовков и тела сообщения.

```

Source Port: 42758
Destination Port: 80
[Stream index: 2]
[TCP Segment Len: 459]
Sequence number: 1      (relative sequence number)
Sequence number (raw): 3258221442
[Next sequence number: 460      (relative sequence number)]
Acknowledgment number: 1      (relative ack number)
Acknowledgment number (raw): 2564089494
1000 .... = Header Length: 32 bytes (8)
Flags: 0x018 (PSH, ACK)
  000. .... = Reserved: Not set
  ...0 .... = Nonce: Not set
  .... 0... = Congestion Window Reduced (CWR): Not set
  .... .0.. = ECN-Echo: Not set
  .... ..0. = Urgent: Not set
  .... ...1 = Acknowledgment: Set
  .... .... 1... = Push: Set
  .... .... .0.. = Reset: Not set
  .... .... ..0. = Syn: Not set
  .... .... ...0 = Fin: Not set
  [TCP Flags: .....AP...]
Window size value: 501
[Calculated window size: 501]
[Window size scaling factor: -1 (unknown)]
Checksum: 0x5c8b [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[SEQ/ACK analysis]
[Timestamps]
TCP payload (459 bytes)

```

Рис. 16: TCP-сегмент переданного запроса.

В стартовой строке указывается метод запроса, версия запроса, а также путь к документу. В заголовках передается метаданная о клиенте или сервере, тело сообщения содержит непосредственно полезные данные.

Структура первичного HTTP-запроса типа GET показана на рисунке 17.

В ответе на первичный запрос приходит HTTP-ответ с кодом 200, сигнализирующем об успешной обработке запроса, временем последней модификации запрашиваемого html-документа и самим содержимым требуемого документа. Это можно увидеть на рисунке 18.

```
GET / HTTP/1.1\r\n
  [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
    Request Method: GET
    Request URI: /
    Request Version: HTTP/1.1
Host: www.ias.ru\r\n
Connection: keep-alive\r\n
Pragma: no-cache\r\n
Cache-Control: no-cache\r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (X11; Fedora; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.1...
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/sig...
Accept-Encoding: gzip, deflate\r\n
Accept-Language: en-US,en;q=0.9\r\n
\r\n
[Full request URI: http://www.ias.ru/]
[HTTP request 1/3]
[Response in frame: 8]
[Next request in frame: 10]
```

Рис. 17: Первичный HTTP-запрос.

Повторный запрос похож на первичный, за исключением того, что добавляется поле **If-Modified-Since** со значением даты, которое мы получили в первичном ответе. Таким образом, сервер вернет 200 только в случае, если запрашиваемая html-страница изменялась после переданного времени. Этот запрос можно увидеть на рисунке 19.

В ответе на повторный запрос можно увидеть код 304, что означает **Not Modified** и сигнализирует о том, что запрашиваемая страница не изменялась с момента времени, указанного в поле **If-Modified-Since** запроса, таким образом можно не передавать её содержимое снова. Это можно увидеть на рисунке 20.

```

Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
      Response Version: HTTP/1.1
      Status Code: 200
      [Status Code Description: OK]
      Response Phrase: OK
      Server: nginx/1.6.2\r\n
      Date: Wed, 15 Apr 2020 11:52:15 GMT\r\n
      Content-Type: text/html\r\n
      Last-Modified: Mon, 24 Apr 2017 11:34:28 GMT\r\n
      Transfer-Encoding: chunked\r\n
      Connection: keep-alive\r\n
      Vary: Accept-Encoding\r\n
      Content-Encoding: gzip\r\n
      \r\n
      [HTTP response 1/3]
      [Time since request: 0.032573610 seconds]
      [Request in frame: 3]
      [Next request in frame: 10]
      [Next response in frame: 11]
      [Request URI: http://www.ias.ru/]
    HTTP chunked response
      Content-encoded entity body (gzip): 80 bytes -> 97 bytes
      File Data: 97 bytes
  Line-based text data: text/html (9 lines)
    <!DOCTYPE html>\n
    <html>\n
    <head>\n
    <title>hello</title>\n
    </head>\n
    <body>\n
    <h1>hello</h1>\n
    </body>\n
    </html>\n

```

Рис. 18: Первичный HTTP-ответ.

```

GET / HTTP/1.1\r\n
  [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
    Request Method: GET
    Request URI: /
    Request Version: HTTP/1.1
  Host: www.ias.ru\r\n
  Connection: keep-alive\r\n
  Cache-Control: max-age=0\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (X11; Fedora; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.106 S
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-
  Accept-Encoding: gzip, deflate\r\n
  Accept-Language: en-US,en;q=0.9\r\n
  If-Modified-Since: Mon, 24 Apr 2017 11:34:28 GMT\r\n
  \r\n
  [Full request URI: http://www.ias.ru/]
  [HTTP request 3/3]
  [Prev request in frame: 10]
  [Response in frame: 26]

```

Рис. 19: Повторный HTTP-запрос.

```
HTTP/1.1 304 Not Modified\r\n
  ▶ [Expert Info (Chat/Sequence): HTTP/1.1 304 Not Modified\r\n]
    Response Version: HTTP/1.1
    Status Code: 304
    [Status Code Description: Not Modified]
    Response Phrase: Not Modified
  Server: nginx/1.6.2\r\n
  Date: Wed, 15 Apr 2020 11:52:18 GMT\r\n
  Last-Modified: Mon, 24 Apr 2017 11:34:28 GMT\r\n
  Connection: keep-alive\r\n
  ETag: "58fde2c4-61"\r\n
  \r\n
  [HTTP response 3/3]
  [Time since request: 0.031237148 seconds]
  \[Prev request in frame: 10\]
  \[Prev response in frame: 11\]
  \[Request in frame: 25\]
  [Request URI: http://www.ias.ru/]
```

Рис. 20: Повторный HTTP-ответ.