ПРАКТИЧЕСКАЯ РАБОТА 2. АНАЛИЗ ПРОТОКОЛА HTTP

Цели работы

• Изучение основ работы протокола HTTP: понять основные принципы функционирования HTTP, его методы (GET, POST и др.) и структуру запросов и ответов.

• Овладение навыками работы с Wireshark: научиться использовать Wireshark для захвата и анализа сетевого трафика.

• Анализ HTTP-запросов и ответов: научиться идентифици

ровать и интерпретировать HTTP-запросы и ответы в захваченном трафике.

• Изучение кодов состояния HTTP: ознакомиться с различными кодами состояния ответов HTTP и их значениями.

• Исследование заголовков HTTP: понять роль заголовков в HTTP-запросах и ответах, изучить, как они влияют на поведение клиента и сервера.

Задание 1. Загрузка файла трассировки http

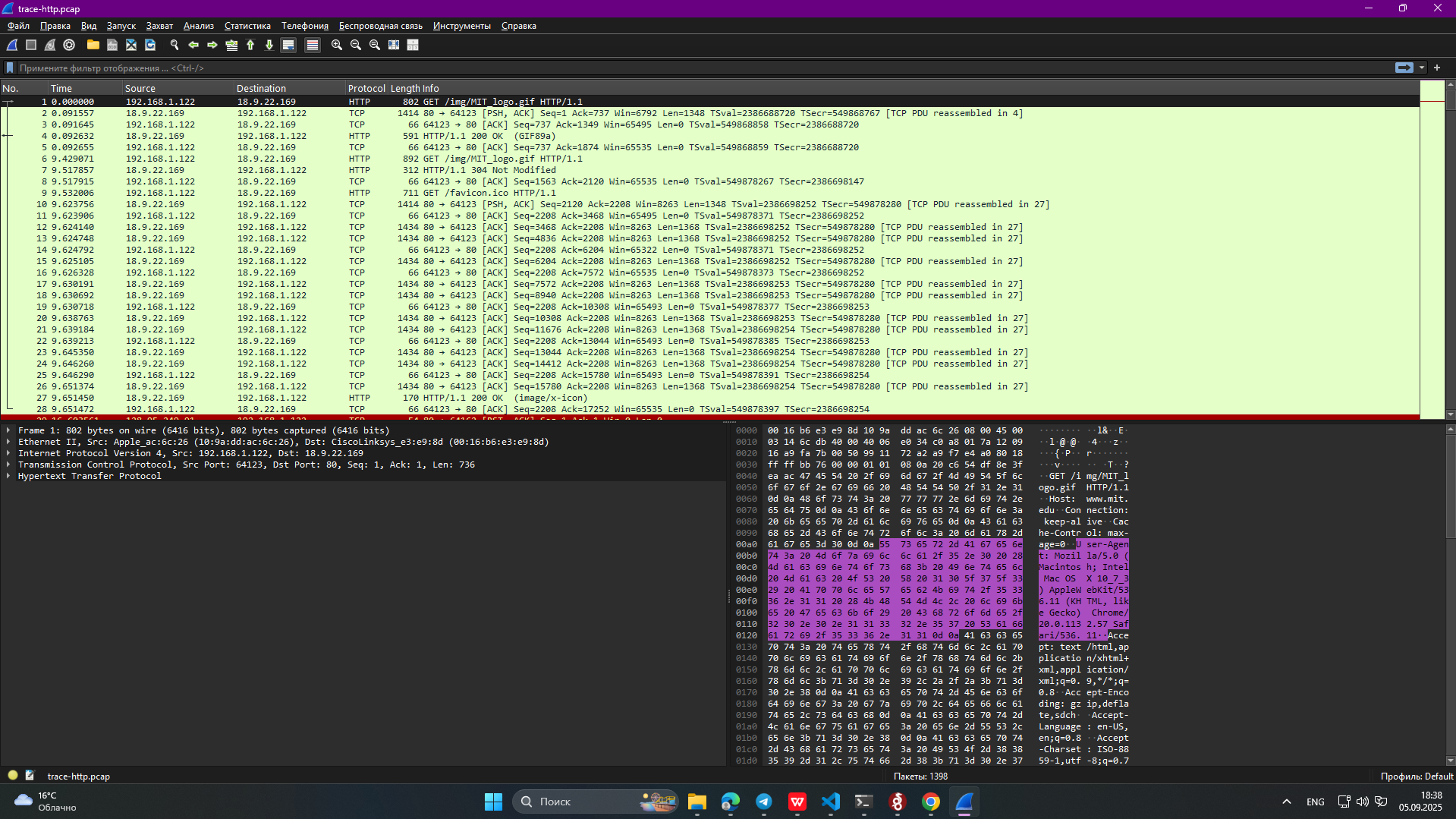


Рис. 1 открытие файла трассировки trace-http.pcap

• Количество пакетов, который были перехвачены Wireshark и присутствуют в этом файле? Ответ : 1398

Каждый пакет имеет временную метку и данные о кадре, кото рые говорят нам о времени создания пакета и о том, в каком именно кадре он передается. Другие данные включают адрес ис точника и назначения.

• Какой общий объем перехваченных пакетов (в байтах)? Ответ : 1223530

• Количество пакетов HTTP Request? Ответ : 98

• Количество пакетов HTTP Response? Ответ : 98

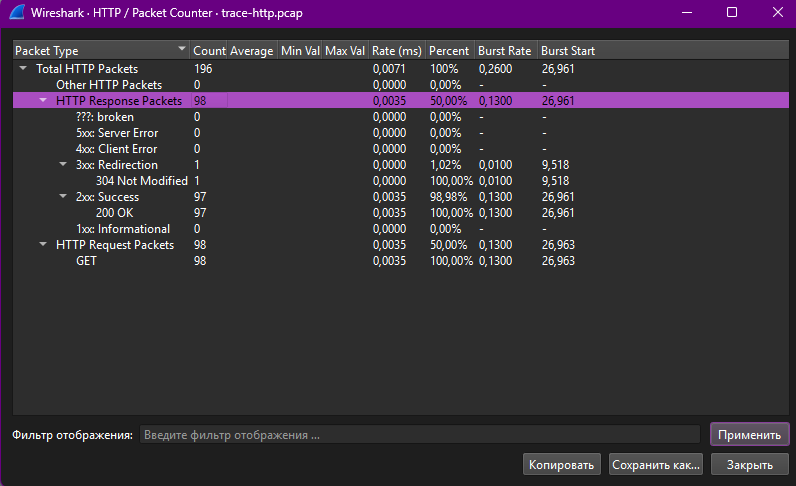


Рис. 2 Проверка статистики пакетов

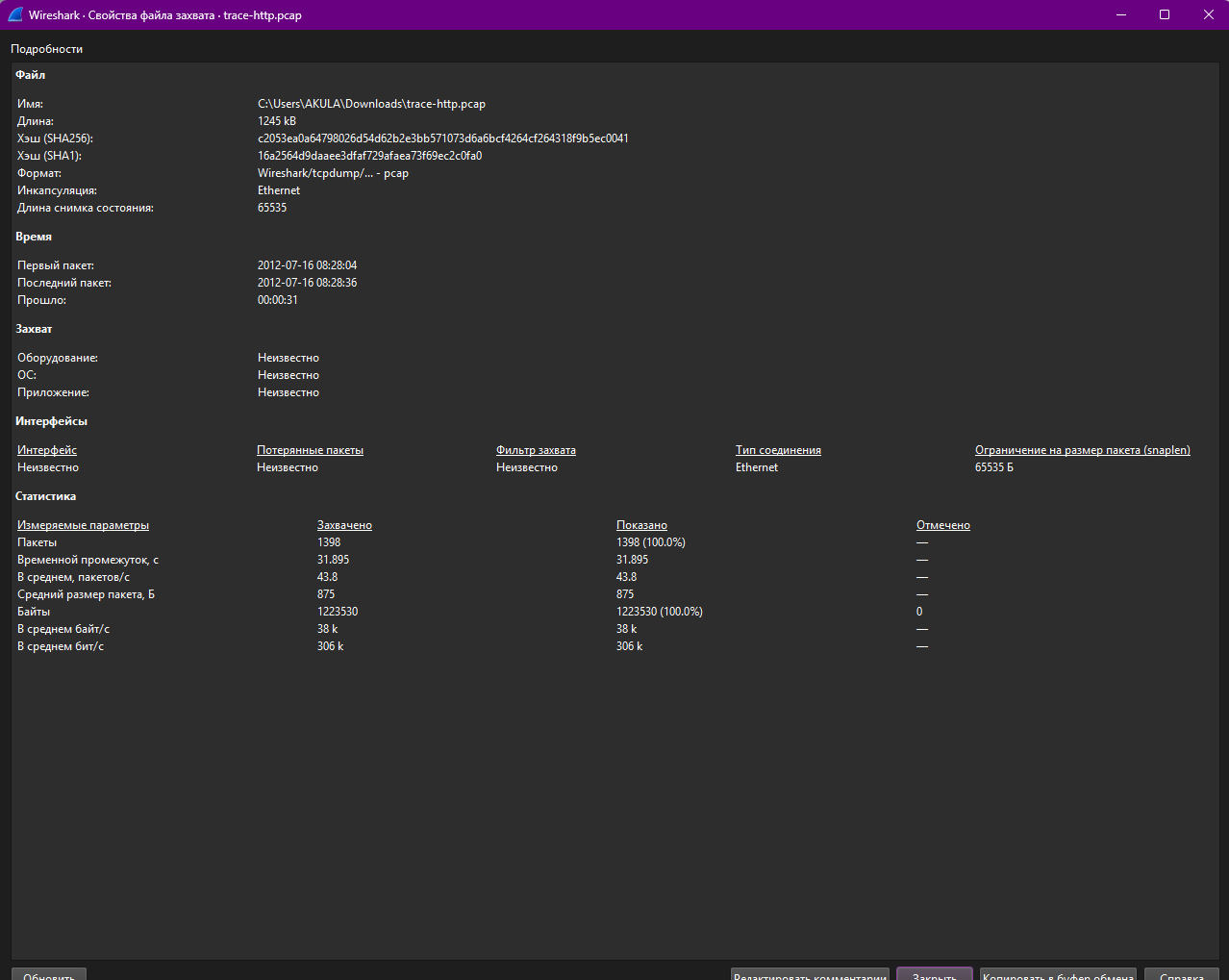
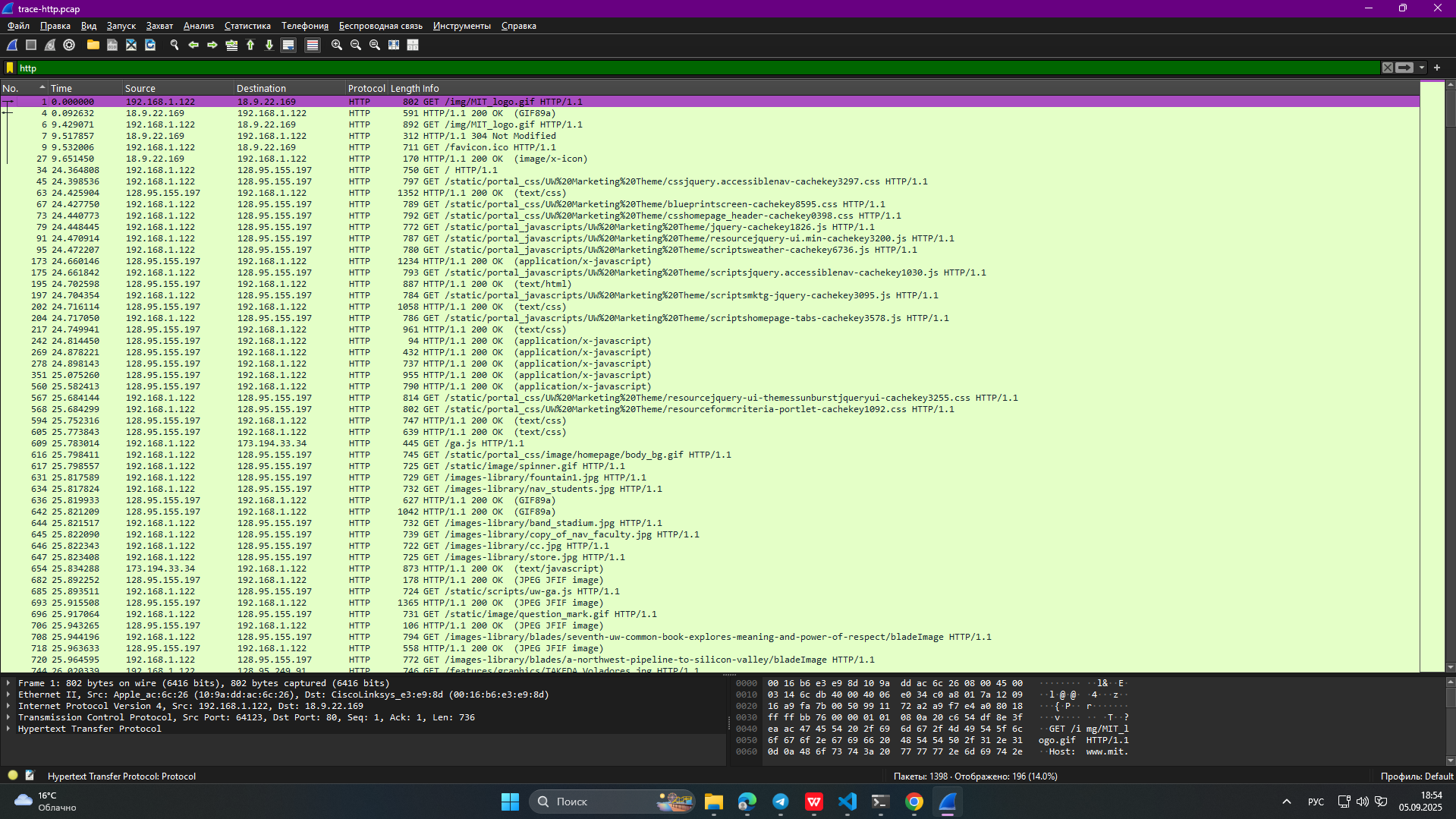
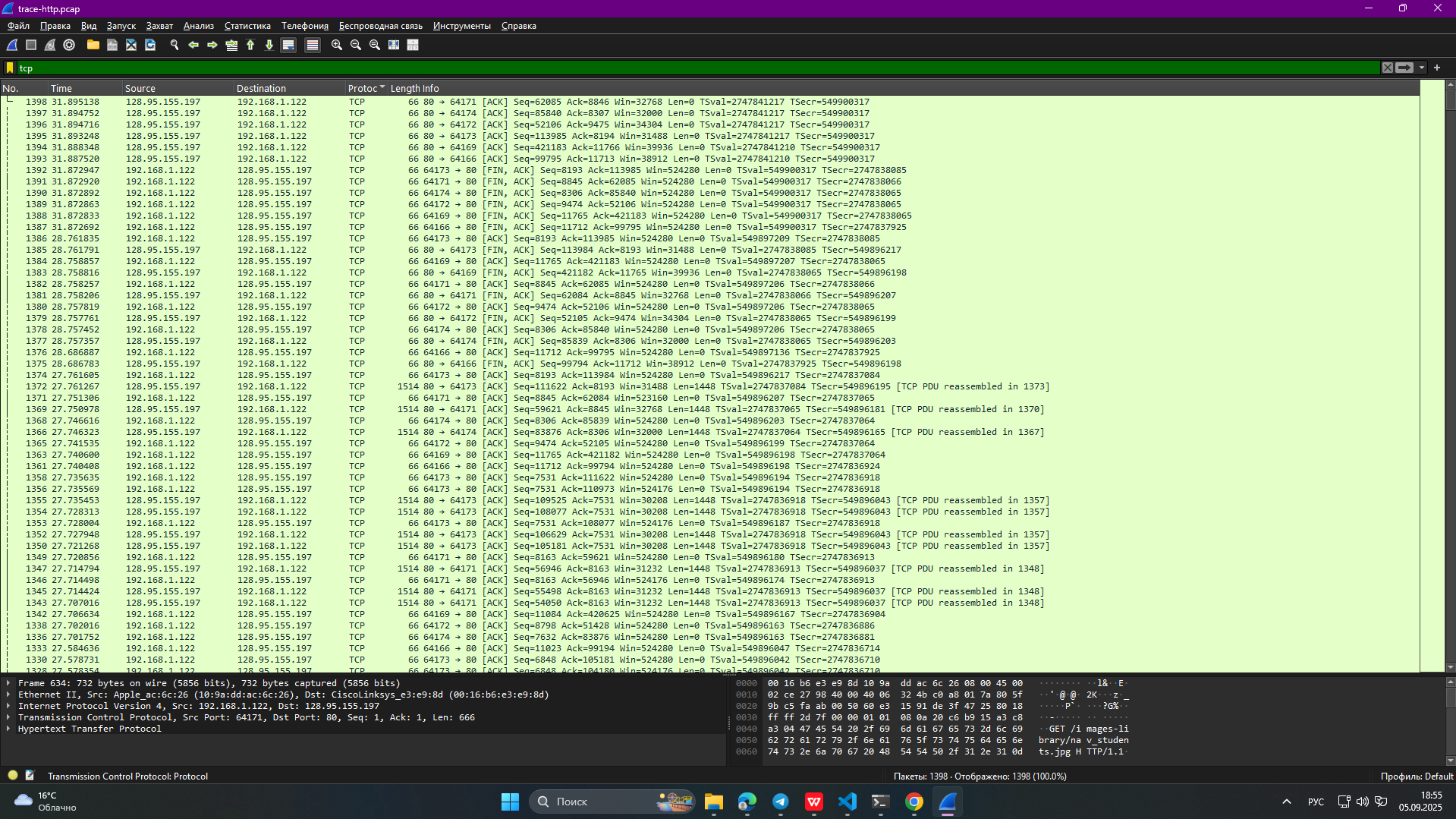


Рис. 3 Общая статистика сохранённого файла

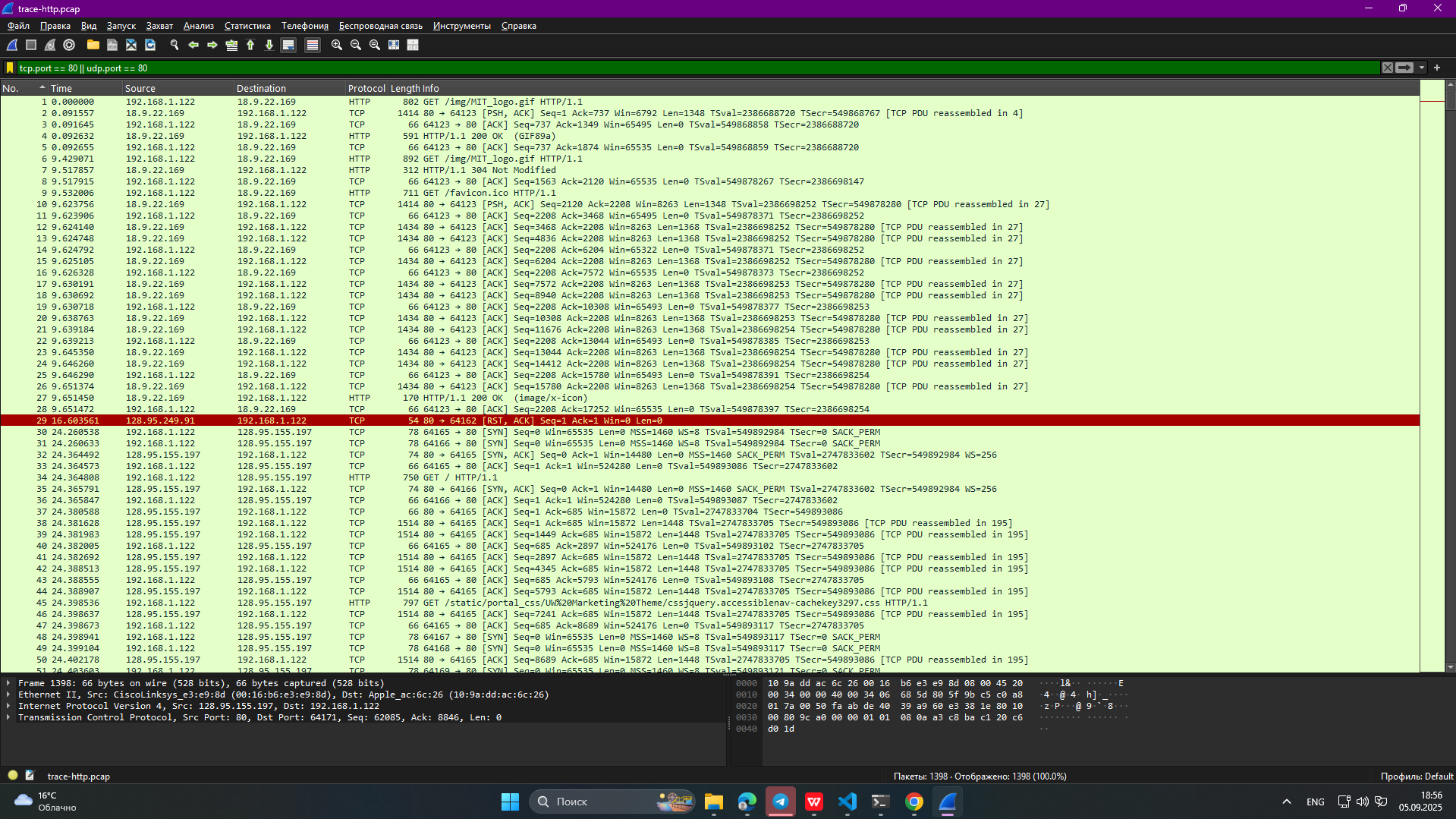
Задание 2. Изучение трассировки запроса и ответа



Применение фильтра для HTTP

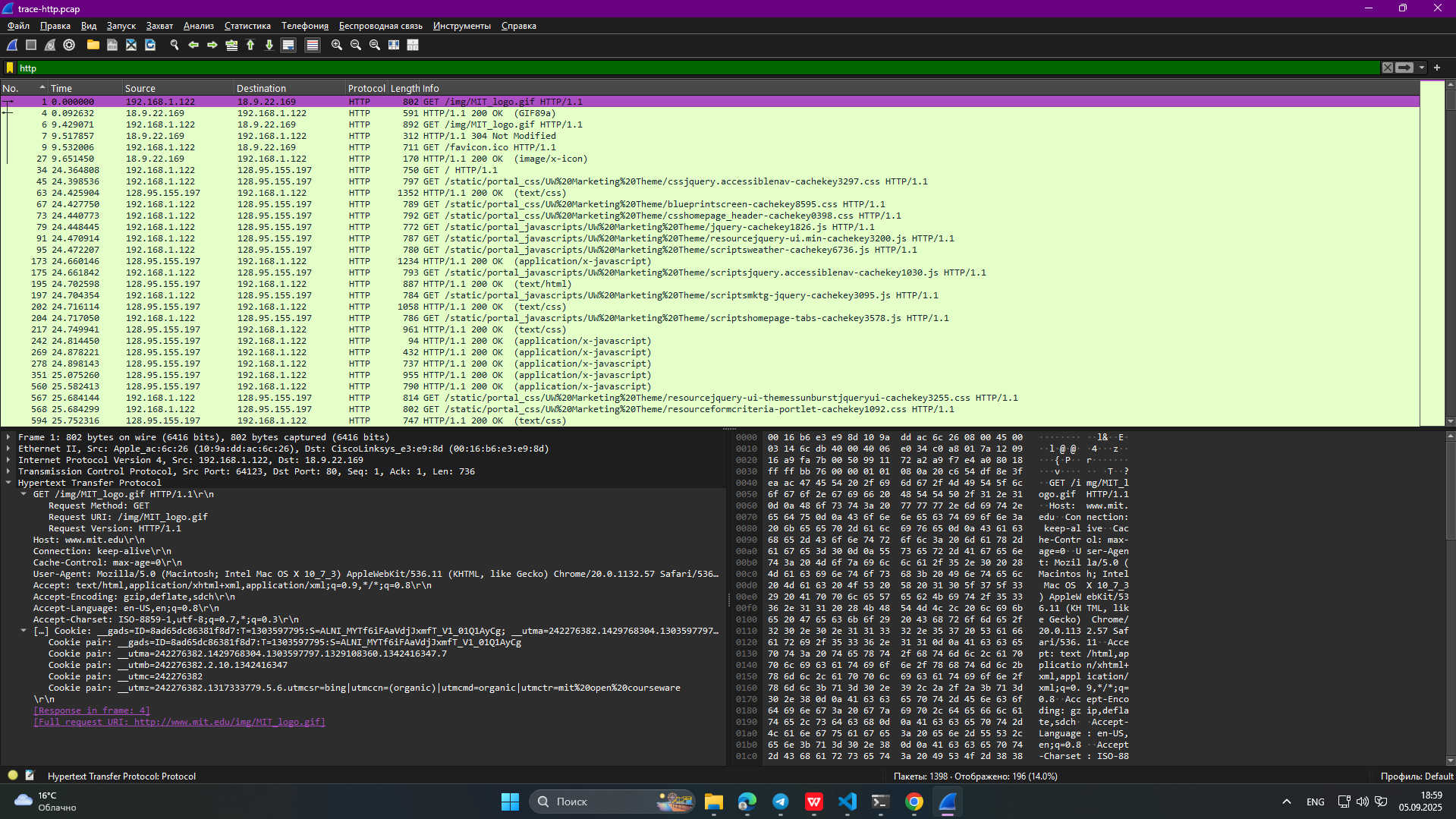


TCP



TCP или UDP с использованием порта 80

Просмотр GET запроса и изучение HTTP-заголовка



первый GET в трассировке и разверните его HTTP блок (это будет 802 GET /img/MIT\_logo.gif HTTP/1.1), как показано ниже).

Host: [www.mit.edu\r\n](http://www.mit.edu/r/n)

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_7\_3) AppleWebKit/536.11 (KHTML, like Gecko) Chrome/20.0.1132.57 Safari/536.11\r\n

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8\r\n

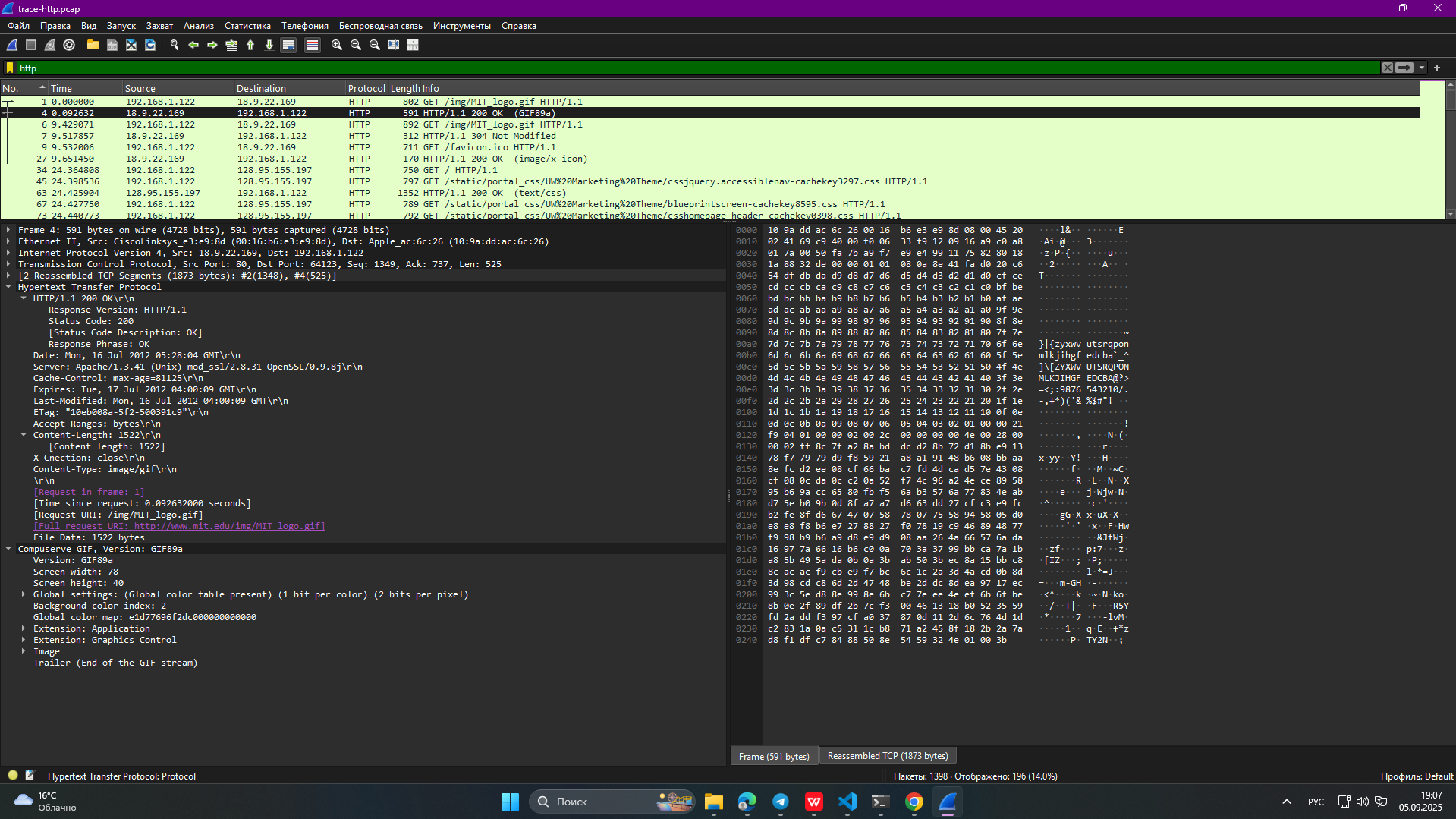
Accept-Encoding: gzip,deflate,sdch\r\n

Accept-Language: en-US,en;q=0.8\r\n

[…] Cookie: \_\_gads=ID=8ad65dc86381f8d7:T=1303597795:S=ALNI\_MYTf6iFAaVdjJxmfT\_V1\_01Q1AyCg; \_\_utma=242276382.1429768304.1303597797.1329108360.1342416347.7; \_\_utmb=242276382.2.10.1342416347; \_\_utmc=242276382; \_\_utmz=242276382.1317333779.5.6

Cache-Control: max-age=0\r\n

Заголовковки имеются



Изучение GET-запроса

Server: Apache/1.3.41 (Unix) mod\_ssl/2.8.31 OpenSSL/0.9.8j\r\n

Date: Mon, 16 Jul 2012 05:28:04 GMT\r\n

Last-Modified: Mon, 16 Jul 2012 04:00:09 GMT\r\n

Cache-Control: max-age=81125\r\n

Expires: Tue, 17 Jul 2012 04:00:09 GMT\r\n

ETag: "10eb008a-5f2-500391c9"\r\n

Вот подробный анализ HTTP-заголовков из вашего примера, а также объяснение, какую полезную информацию они содержат и что она означает.

1. Server

Пример: Server: Apache/1.3.41 (Unix) mod\_ssl/2.8.31 OpenSSL/0.9.8j

Описание: Этот заголовок указывает, какой веб-сервер обработал запрос. В данном случае:

Apache/1.3.41 — версия веб-сервера Apache.

Unix — операционная система, на которой работает сервер.

mod\_ssl/2.8.31 — модуль для поддержки SSL/TLS.

OpenSSL/0.9.8j — версия библиотеки OpenSSL, используемой для шифрования.

Полезная информация:

Позволяет определить, какое программное обеспечение используется на сервере.

Может быть полезно для диагностики совместимости или уязвимостей (например, если версия Apache или OpenSSL устарела).

2. Date

Пример: Date: Mon, 16 Jul 2012 05:28:04 GMT

Описание: Указывает дату и время, когда сервер сформировал и отправил ответ на запрос.

Полезная информация:

Помогает клиенту понять, насколько актуален ответ.

Используется для синхронизации времени между клиентом и сервером.

3. Last-Modified

Пример: Last-Modified: Mon, 16 Jul 2012 04:00:09 GMT

Описание: Указывает дату и время последнего изменения запрошенного ресурса (например, HTML-страницы, изображения).

Полезная информация:

Клиент может использовать этот заголовок для кэширования: если ресурс не изменился, сервер может вернуть ответ 304 Not Modified, экономя трафик.

Помогает определить, нужно ли загружать ресурс заново.

4. Cache-Control

Пример: Cache-Control: max-age=81125

Описание: Управляет кэшированием ответа. В данном случае:

max-age=81125 — ресурс можно кэшировать на 81125 секунд (примерно 22.5 часа).

Полезная информация:

Определяет, как долго клиент или промежуточные прокси могут хранить ответ в кэше.

Помогает снизить нагрузку на сервер и ускорить загрузку страниц для пользователей.

5. Expires

Пример: Expires: Tue, 17 Jul 2012 04:00:09 GMT

Описание: Указывает дату и время, после которого кэшированный ответ считается устаревшим.

Полезная информация:

Альтернативный способ управления кэшированием (часто используется вместе с Cache-Control).

Клиент не будет использовать кэшированный ответ после указанной даты.

6. ETag

Пример: ETag: "10eb008a-5f2-500391c9"

Описание: Уникальный идентификатор версии ресурса. Используется для проверки, изменился ли ресурс с момента последнего запроса.

Полезная информация:

Клиент может отправить этот идентификатор в следующем запросе (If-None-Match), чтобы сервер вернул ресурс только в случае его изменения.

Помогает избежать повторной передачи неизменённых данных.

Пример использования заголовков в GET-запросе

Представим, что клиент отправляет GET-запрос на страницу example.com/page.html.

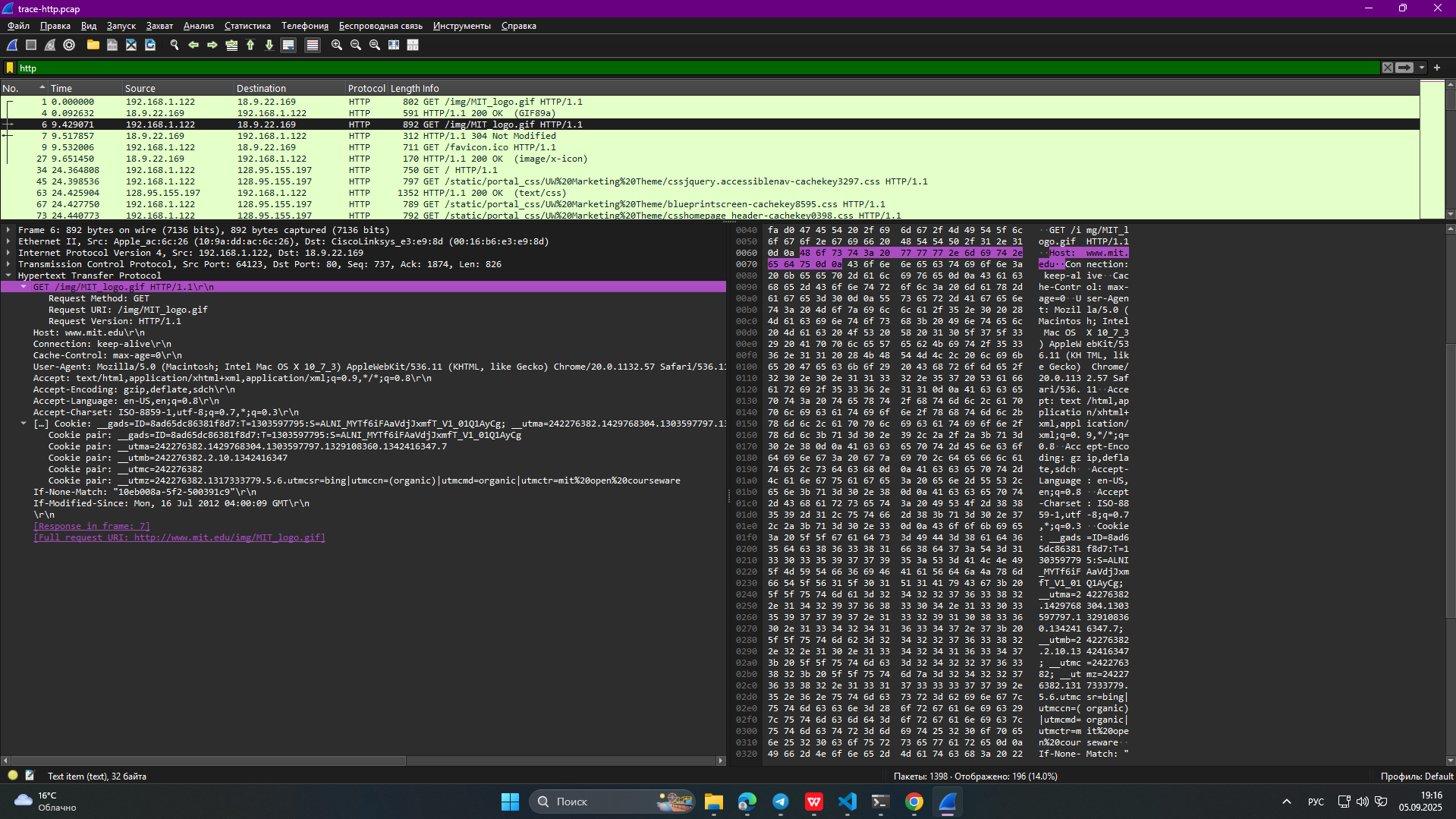
Сервер возвращает ответ с указанными заголовками.

Клиент сохраняет ресурс в кэш и запоминает ETag и Last-Modified.

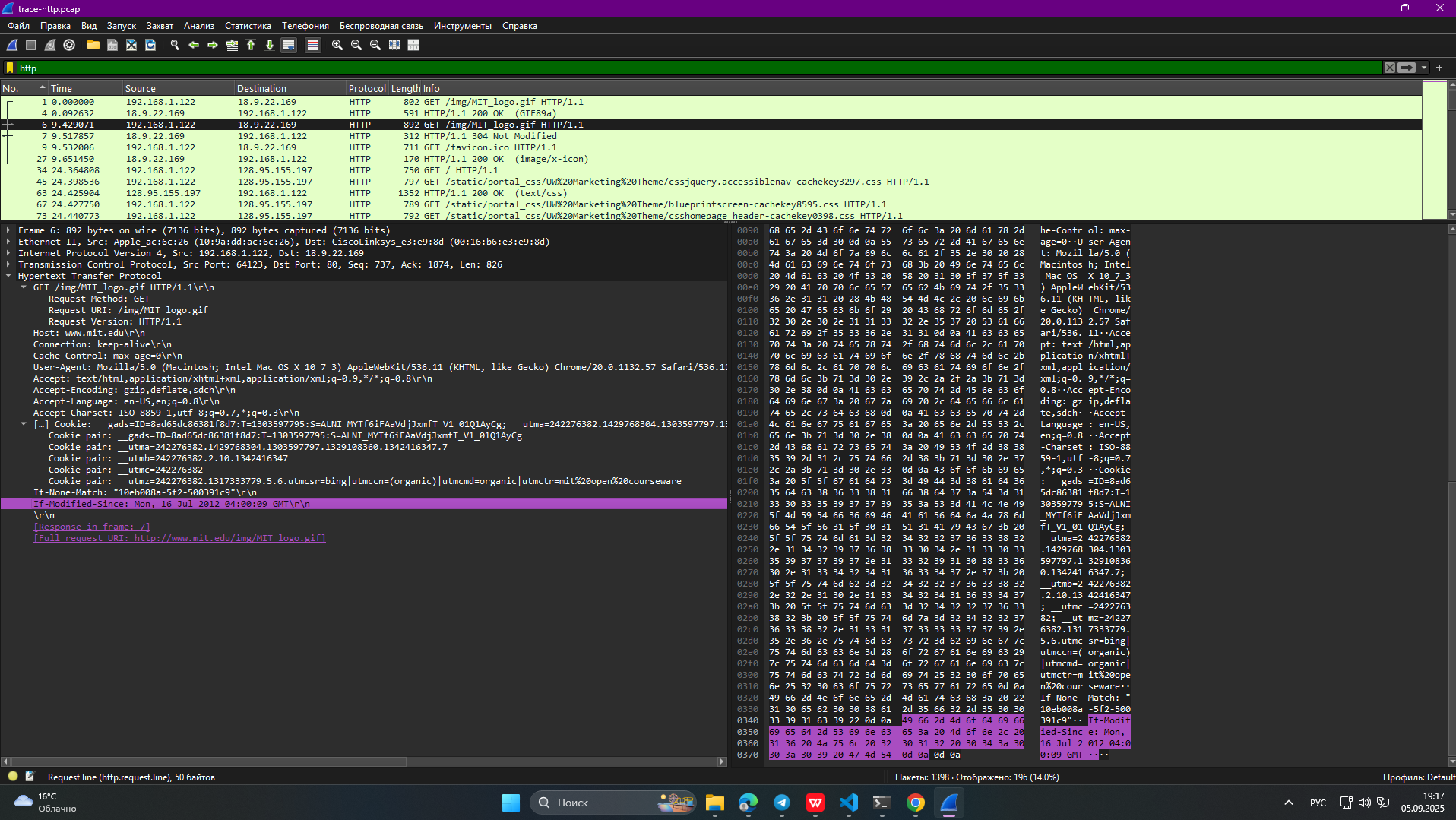
При следующем запросе клиент может отправить заголовки If-None-Match и If-Modified-Since.

Если ресурс не изменился, сервер вернёт 304 Not Modified, и клиент использует кэшированную версию, экономя трафик и время.

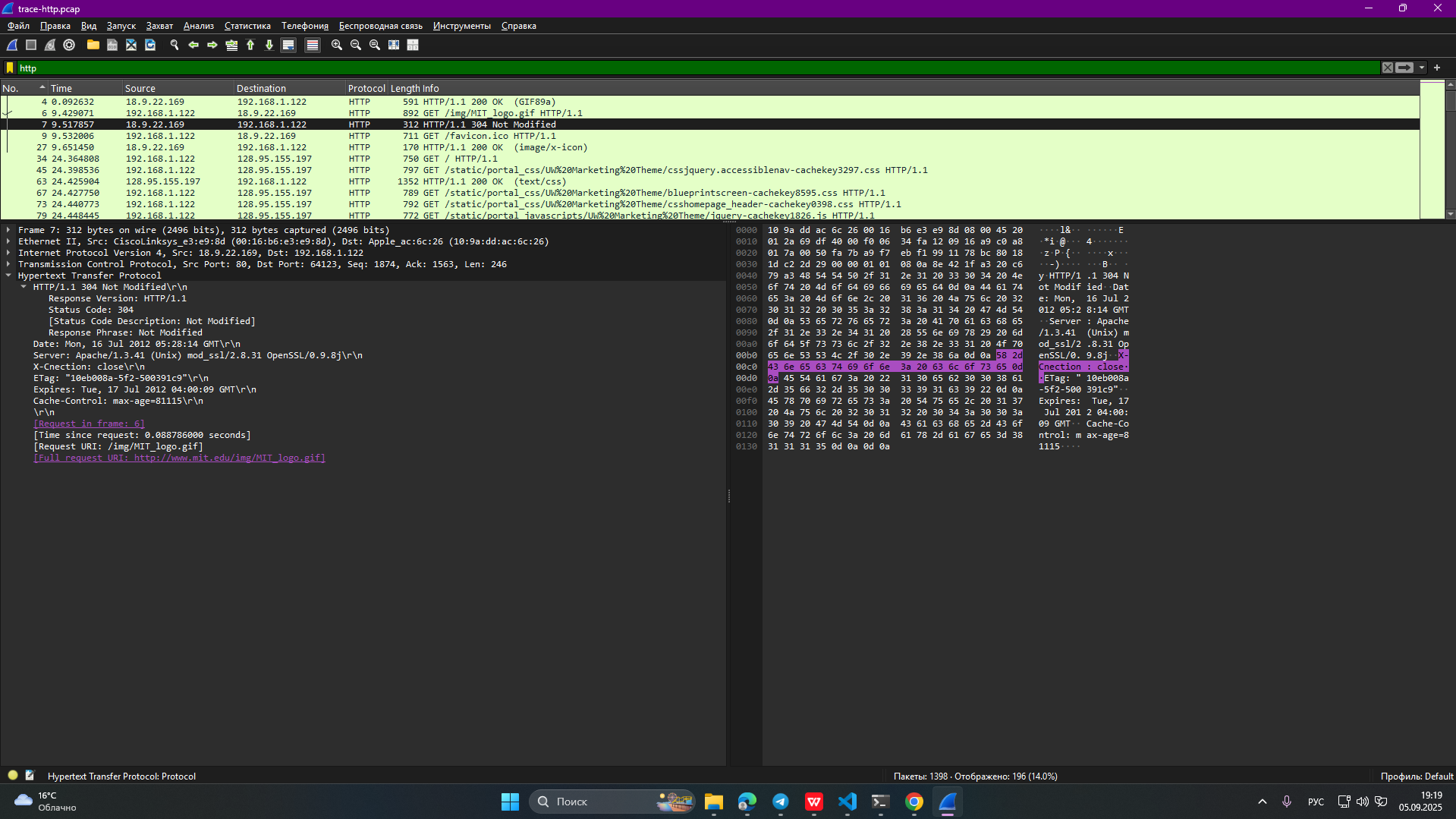
Задание 3. Кэширование контента



892 GET /img/MIT\_logo.gif HTTP 1.1



«If Modified-Since»



«304 Not Modified»

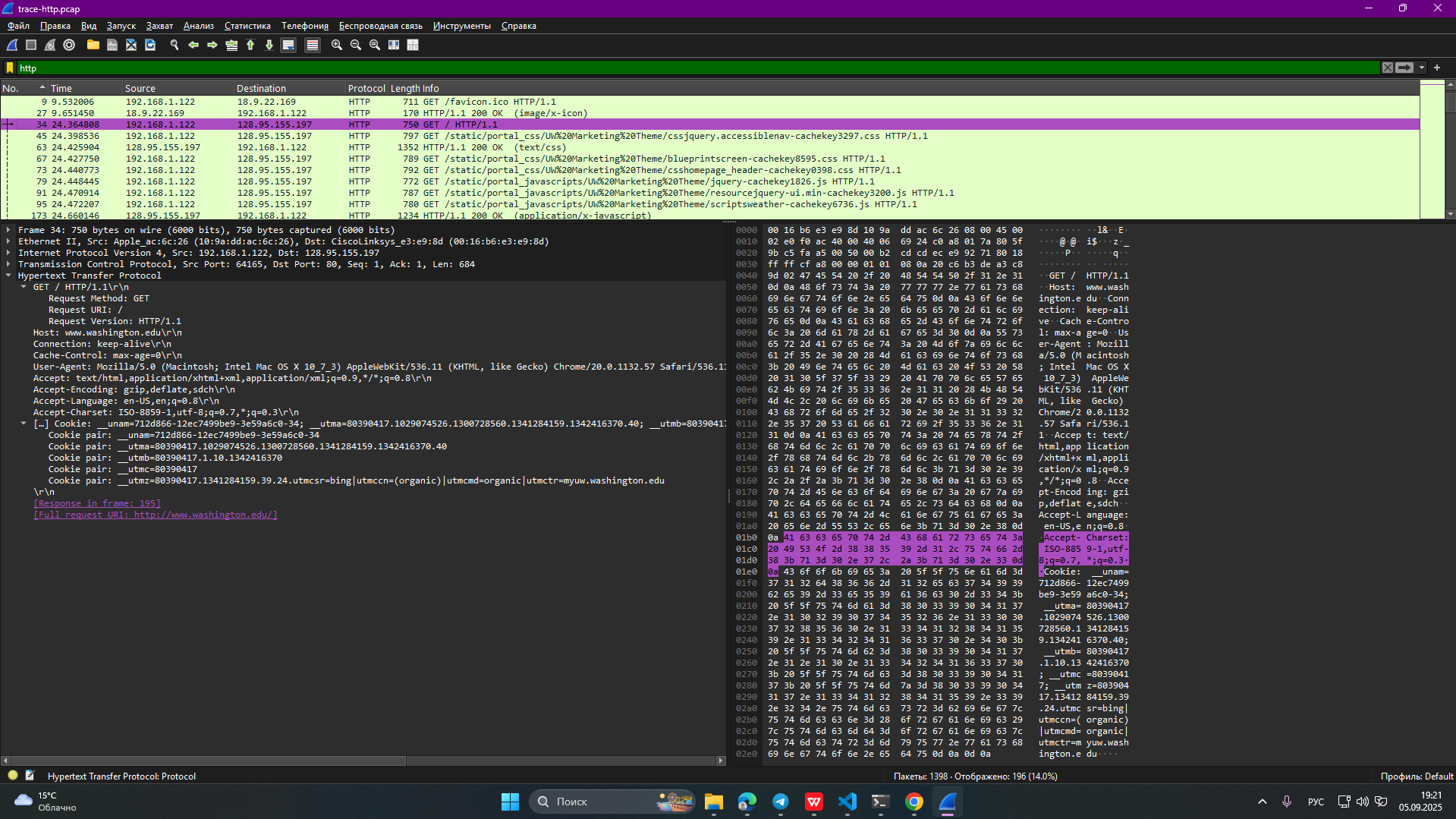


Сайт с Gif изображенем

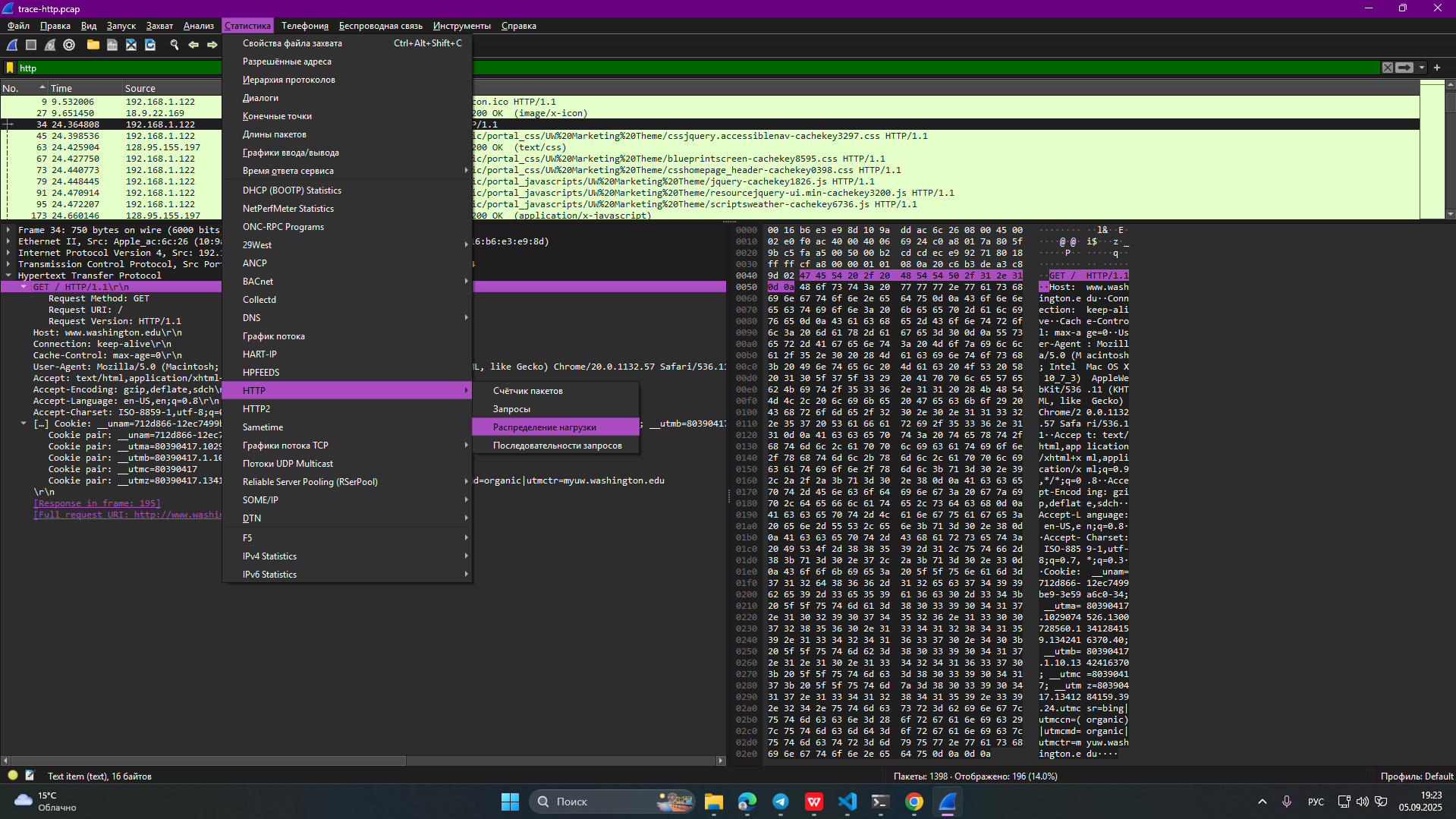
IMG_256

Gif изображения

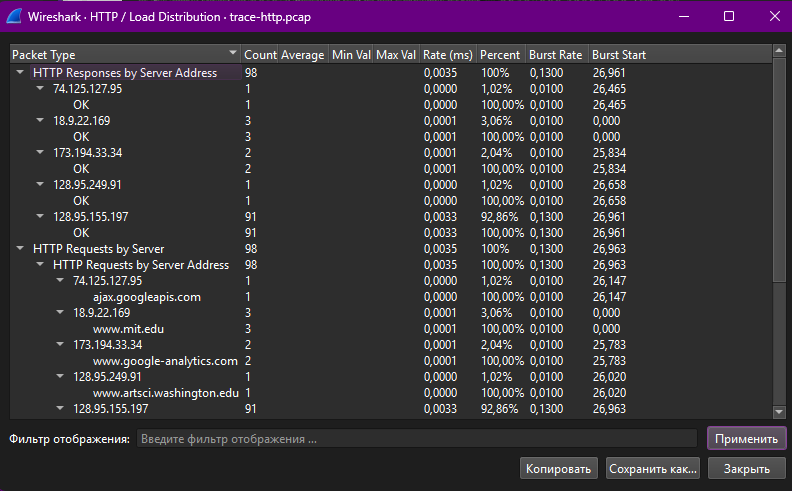
Задание 4. Сложные страницы



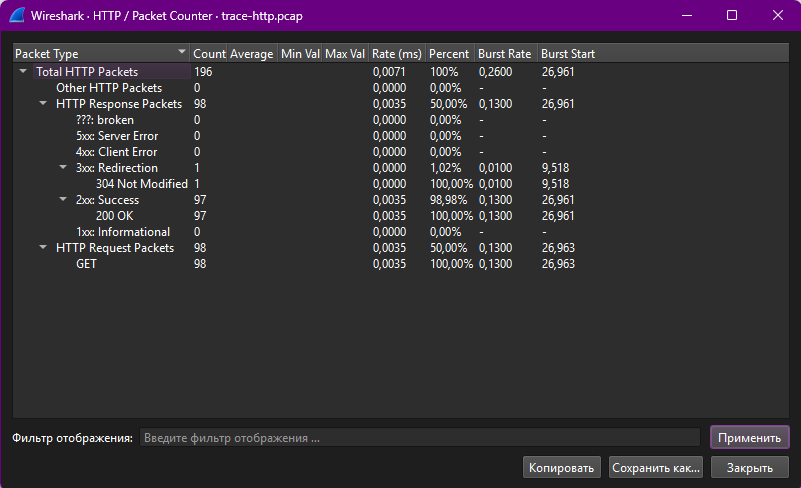
750 GET / HTTP/1.1



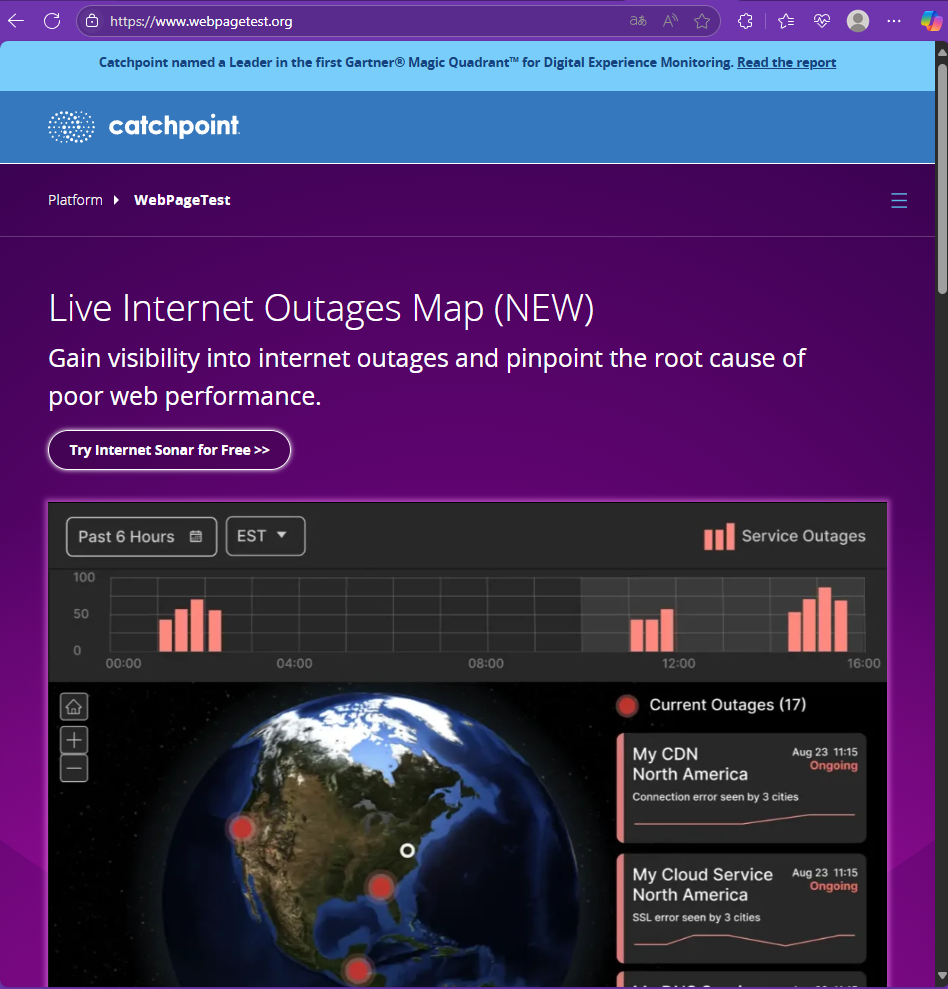
разделах Стати стика и HTTP



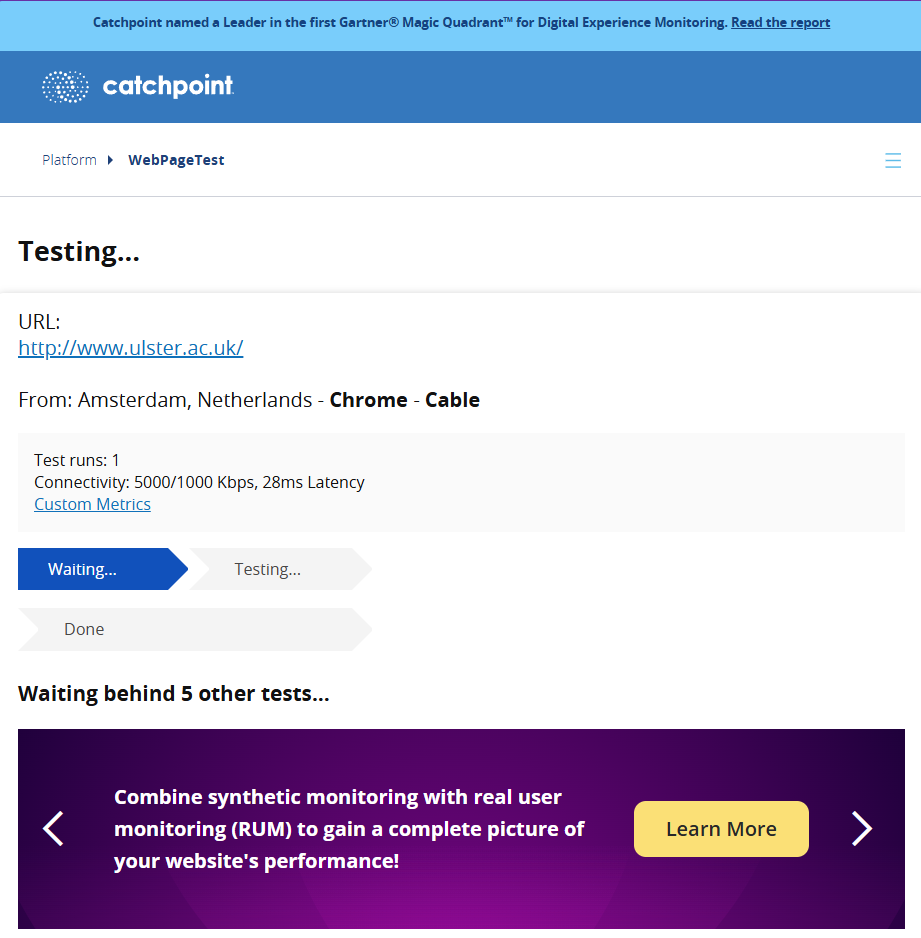
Отображение статистики по HTTP



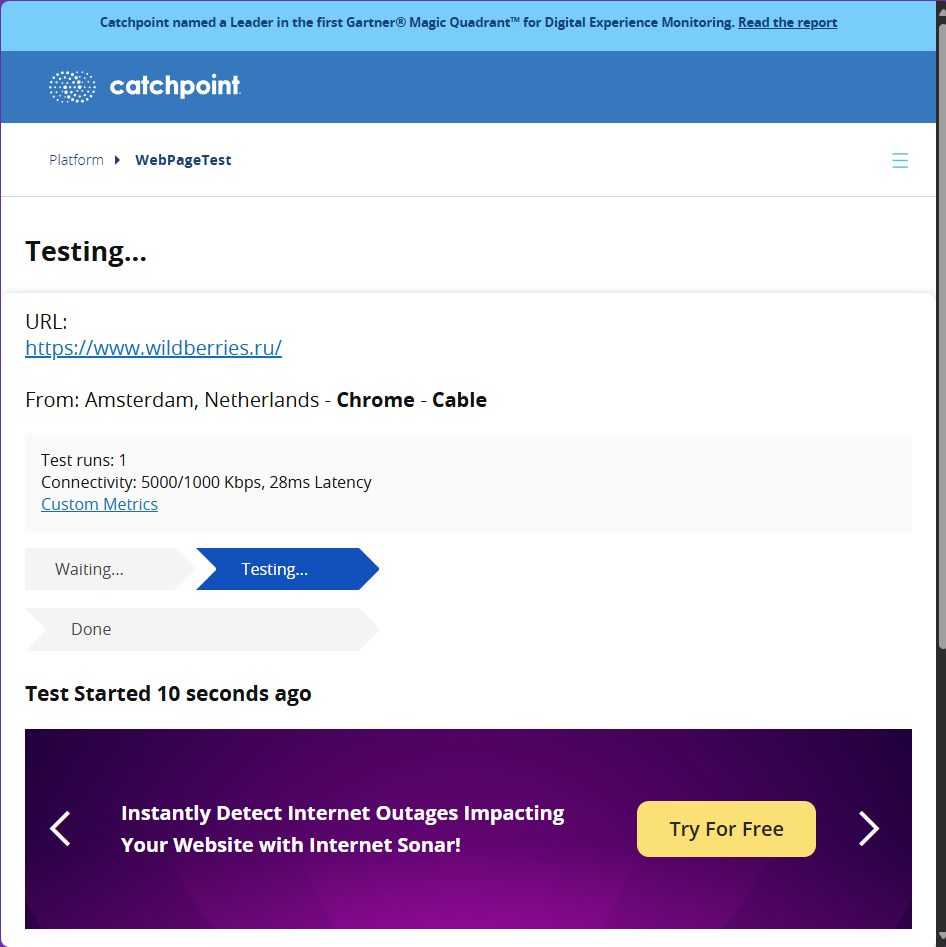
счетчика пакетов HTTP

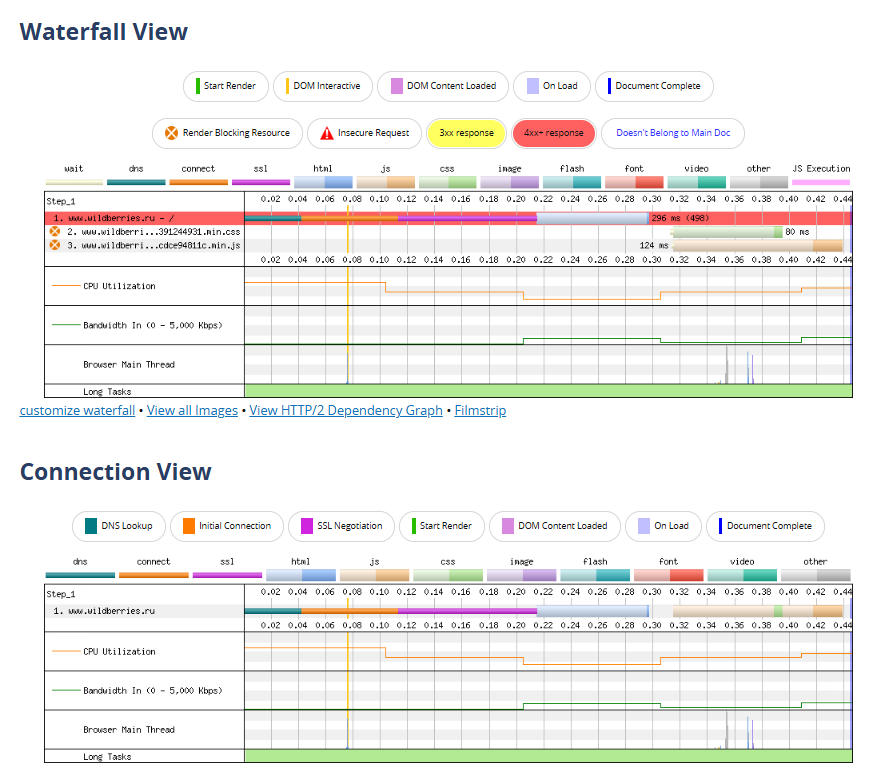


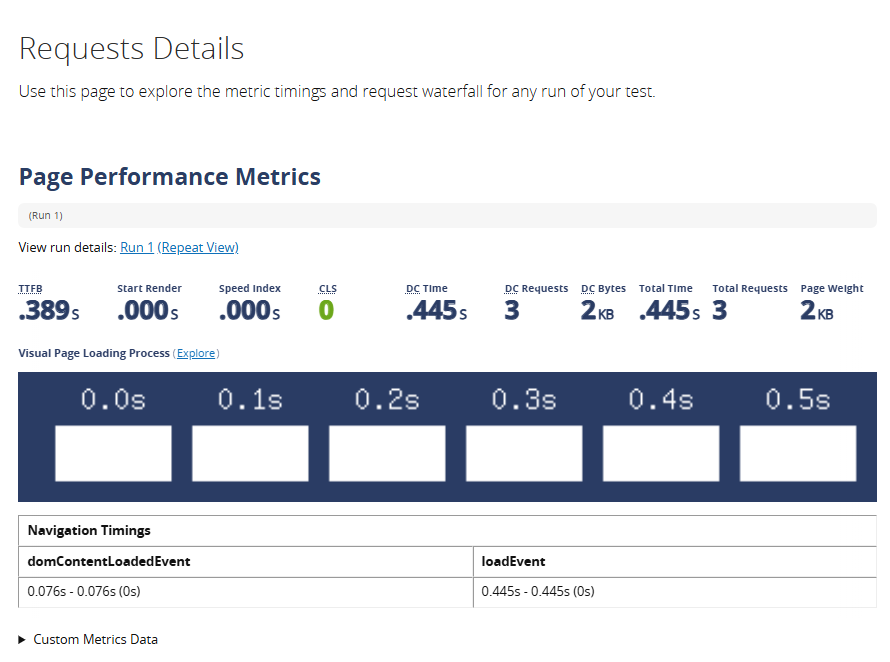
<https://www.webpagetest.org/.>



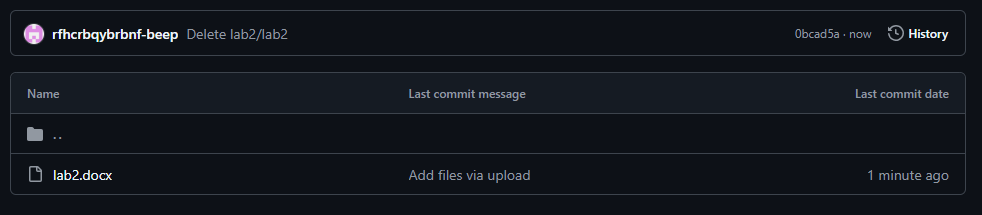
[www.ulster.ac.uk](http://www.ulster.ac.uk)







Задание 5. Отправка отчета на проверку



Контрольные вопросы

Вот краткие и структурированные ответы на контрольные вопросы по протоколу HTTP и смежным темам:

1. Что такое протокол HTTP и для чего он используется?

HTTP (HyperText Transfer Protocol) — протокол прикладного уровня для передачи данных в сети, преимущественно в вебе. Используется для обмена информацией между клиентом (браузером) и сервером (веб-сайтом). Основная задача — обеспечить передачу гипертекстовых документов (HTML, CSS, JS, медиафайлов).

2. Каковы основные методы HTTP и какова их функция?

Основные методы:

GET — запрос данных с сервера (например, загрузка страницы).

POST — отправка данных на сервер (например, отправка формы).

PUT — обновление существующего ресурса.

DELETE — удаление ресурса.

HEAD — запрос заголовков без тела ответа.

OPTIONS — запрос поддерживаемых методов для ресурса.

3. Объясните разницу между методами GET и POST.

| GET | POST |
| --- | --- |
| Данные передаются в URL (видно в адресной строке). | Данные передаются в теле запроса (не видно в URL). |
| Используется для получения данных. | Используется для отправки данных на сервер. |
| Ограничен длиной URL (~2048 символов). | Нет ограничений по объёму данных. |
| Кэшируется браузерами. | Не кэшируется. |

4. Что такое статус-коды HTTP и какова их роль?

Статус-коды — трёхзначные числа, которые сервер возвращает клиенту для информирования о результате обработки запроса. Они делятся на 5 классов:

1xx — информационные.

2xx — успешные.

3xx — перенаправления.

4xx — ошибки клиента.

5xx — ошибки сервера.

1. Перечислите основные статус-коды HTTP и объясните их значение.

| Код | Значение |
| --- | --- |
| 200 OK | Запрос успешно обработан. |
| 301 Moved Permanently | Ресурс постоянно перемещён на новый URL. |
| 302 Found | Ресурс временно доступен по другому URL. |
| 400 Bad Request | Некорректный запрос. |
| 401 Unauthorized | Требуется аутентификация. |
| 403 Forbidden | Доступ запрещён. |
| 404 Not Found | Ресурс не найден. |
| 500 Internal Server Error | Ошибка на стороне сервера. |

6. Что такое заголовки HTTP и какую информацию они содержат?

Заголовки HTTP — метаданные, передаваемые в запросах и ответах. Они содержат информацию о:

Типе контента (`Content-Type`).

Кэшировании (`Cache-Control`, `Expires`).

Аутентификации (`Authorization`).

Клиенте (`User-Agent`).

Сервере (`Server`).

Состоянии соединения (`Connection`).

7. Какова роль заголовка User-Agent в HTTP-запросах?

`User-Agent` идентифицирует клиентское приложение (браузер, ОС, версию), отправляющее запрос. Сервер может использовать эту информацию для оптимизации контента под конкретное устройство или браузер.

8. Что такое HTTPS и как он отличается от HTTP?

HTTPS (HTTP Secure) — защищённая версия HTTP, использующая шифрование (SSL/TLS) для обеспечения конфиденциальности и целостности данных. Отличия:

Данные передаются в зашифрованном виде.

Использует порт 443 (вместо 80 для HTTP).

Требует сертификат для подтверждения подлинности сервера.

9. Как работает механизм кэширования в HTTP?

Кэширование позволяет клиенту сохранять копии ресурсов (HTML, изображения) для повторного использования. Управляется заголовками:

`Cache-Control` — указывает, как долго ресурс можно кэшировать.

`Expires` — дата истечения срока кэша.

`ETag` — уникальный идентификатор версии ресурса.

`Last-Modified` — время последнего изменения.

10. Что такое сессия в контексте HTTP и как она реализуется?

Сессия — временное взаимодействие между клиентом и сервером. HTTP — протокол без состояния, поэтому сессии реализуются через:

Куки (Cookies) — хранят идентификатор сессии на стороне клиента.

Сессионные переменные — хранятся на сервере и привязываются к идентификатору сессии.

11. Как использовать Wireshark для захвата HTTP-трафика?

1. Установите и запустите Wireshark.

2. Выберите сетевой интерфейс для захвата.

3. Запустите захват пакетов.

4. Примените фильтр `http` для отображения только HTTP-трафика.

5. Анализируйте пакеты (запросы, ответы, заголовки).

12. Что такое фильтры в Wireshark и как они помогают в анализе трафика?

Фильтры позволяют отображать только нужные пакеты. Примеры:

`http.request.method == "GET"` — показывает только GET-запросы.

`ip.src == 192.168.1.1` — пакеты с определённого IP.

`tcp.port == 80` — трафик на порту 80 (HTTP).

13. Как можно просмотреть содержимое HTTP-запроса и ответа в Wireshark?

1. Найдите пакет с HTTP-запросом/ответом.

2. Разверните секцию Hypertext Transfer Protocol .

3. Просмотрите заголовки и тело (если есть) в текстовом виде.

14. Что такое TCP и как он взаимодействует с HTTP?

TCP (Transmission Control Protocol) — транспортный протокол, обеспечивающий надёжную передачу данных. HTTP использует TCP для установления соединения (порт 80 или 443) и передачи пакетов.

15. Каковы особенности работы HTTP/2 по сравнению с HTTP/1.1?

Мультиплексирование — несколько запросов/ответов в одном соединении.

Сжатие заголовков — уменьшает объём передаваемых данных.

Приоритизация — важные ресурсы загружаются быстрее.

Бинарный протокол — вместо текстового формата.

16. Что такое HTTP-заголовок Referer и для чего он используется?

`Referer` указывает URL страницы, с которой был выполнен переход. Используется для:

Аналитики трафика.

Защиты от CSRF-атак.

Логирования переходов.

17. Каковы основные принципы работы с куками в HTTP?

Куки устанавливаются сервером через заголовок `Set-Cookie`.

Клиент отправляет их обратно в заголовке `Cookie`.

Используются для хранения идентификаторов сессий, настроек пользователя, трекинга.

18. Что такое Content-Type и как он влияет на обработку данных?

`Content-Type` указывает тип данных в теле запроса/ответа (например, `text/html`, `application/json`). Браузер использует этот заголовок для корректного отображения контента.

19. Как можно использовать Wireshark для анализа проблем с производительностью HTTP?

Проверьте время ответа сервера (`Time` в Wireshark).

Анализируйте задержки между пакетами.

Ищите повторные передачи (TCP Retransmission).

Проверьте размер передаваемых данных.

20. Что такое REST и как он связан с протоколом HTTP?

REST (Representational State Transfer) — архитектурный стиль для создания веб-сервисов, использующий HTTP-методы (GET, POST, PUT, DELETE) для работы с ресурсами. REST-сервисы используют HTTP для передачи данных в формате JSON/XML и следуют принципам:

Клиент-серверная архитектура.

Без состояния (stateless).

Кэшируемость.

Единообразие интерфейса.