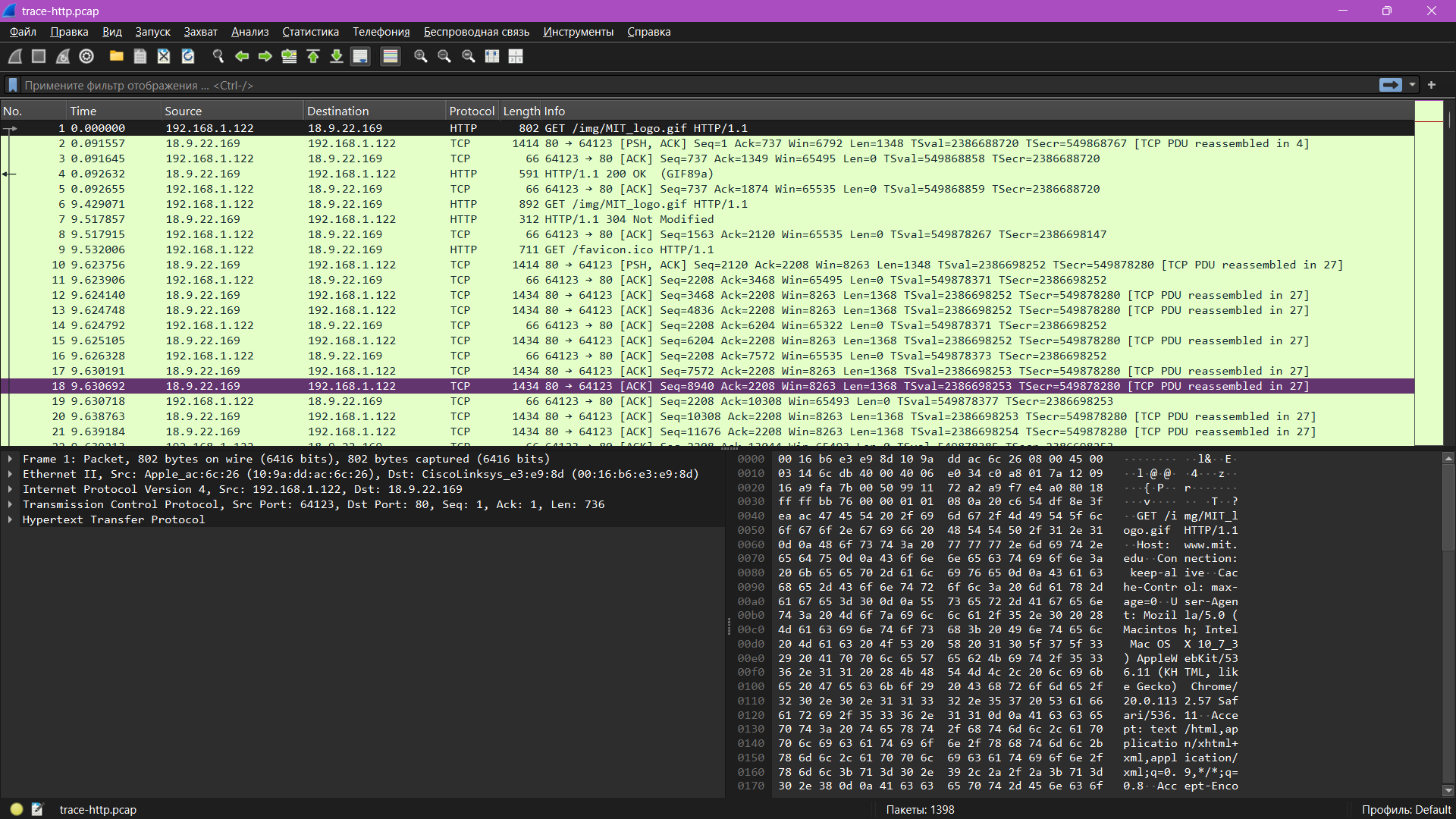
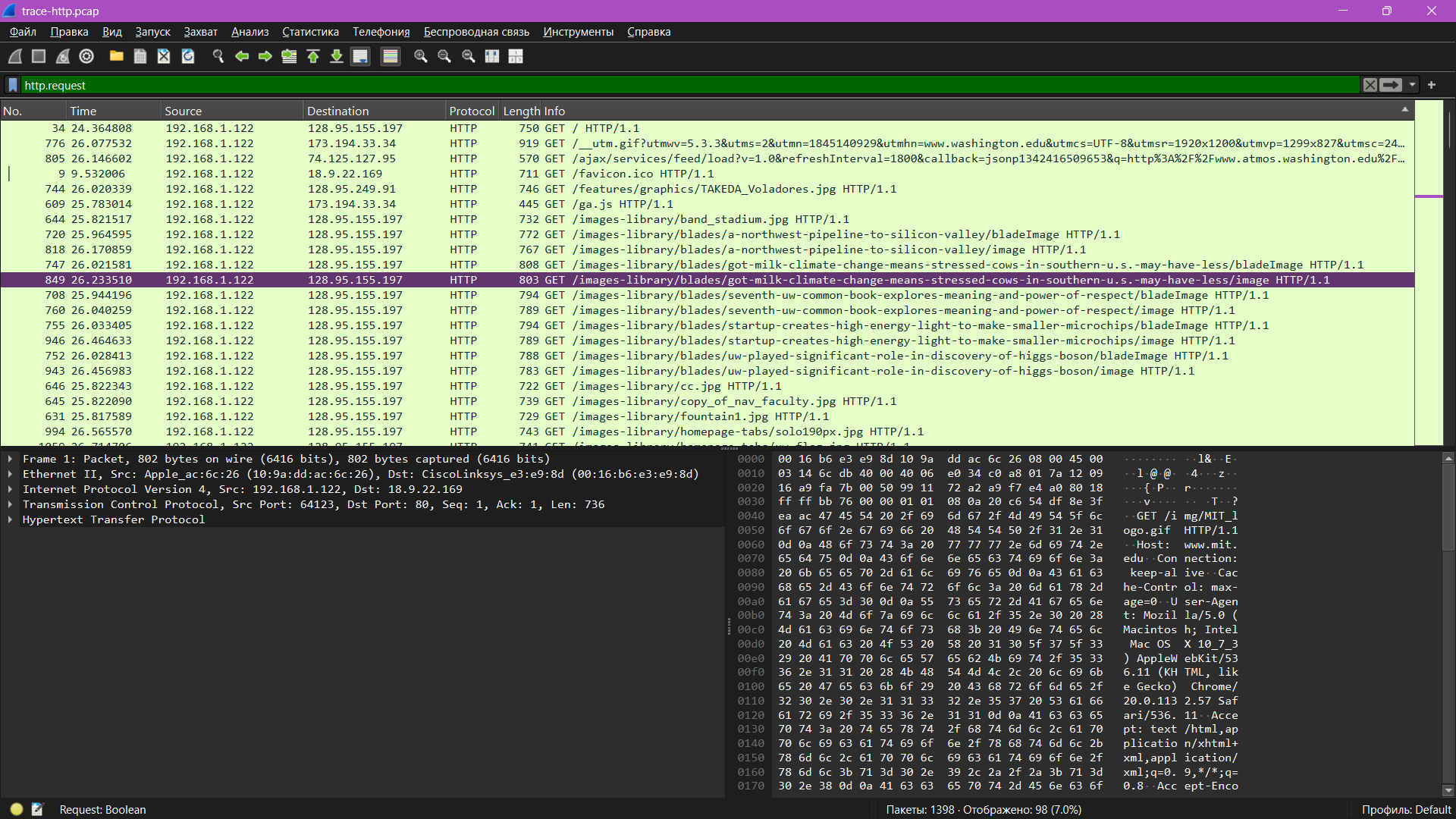
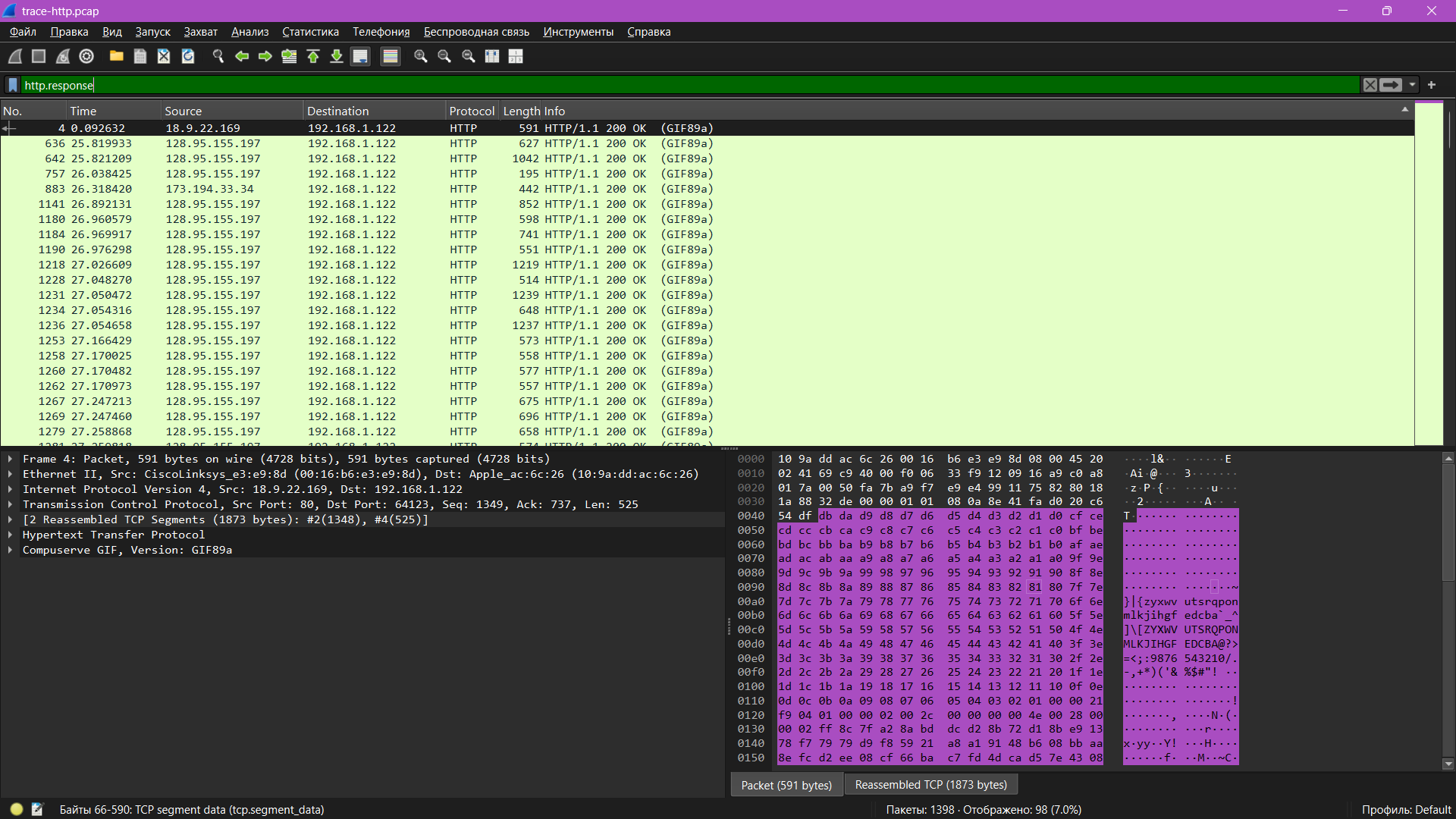
**Задание 1. Загрузка файла трассировки http**

Поведение браузера может быть достаточно сложным, так как там используя больше функций HTTP, чем просто базовый обмен данными. Поэтому в этой работе мы будем использовать предохранённый файл трассировки, который можно скачать отсюда: [trace-http.pcap](https://github.com/SerjiEvg/web-development-part1/raw/main/Practice/trace-http.pcap" \t "_blank) . Откройте этот файл в Wireshark, должен появится экран, как показано ниже.

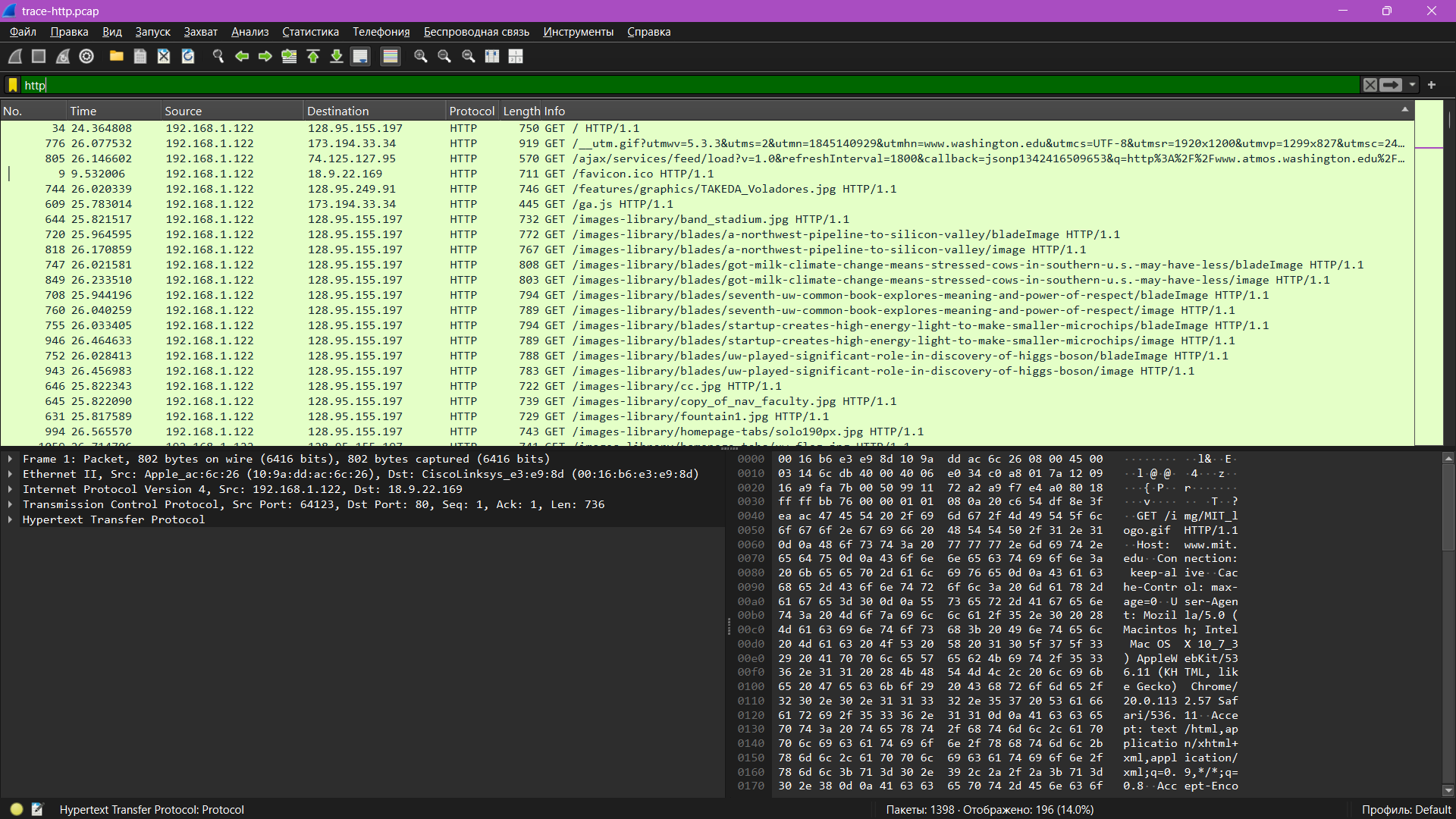
 

http.request

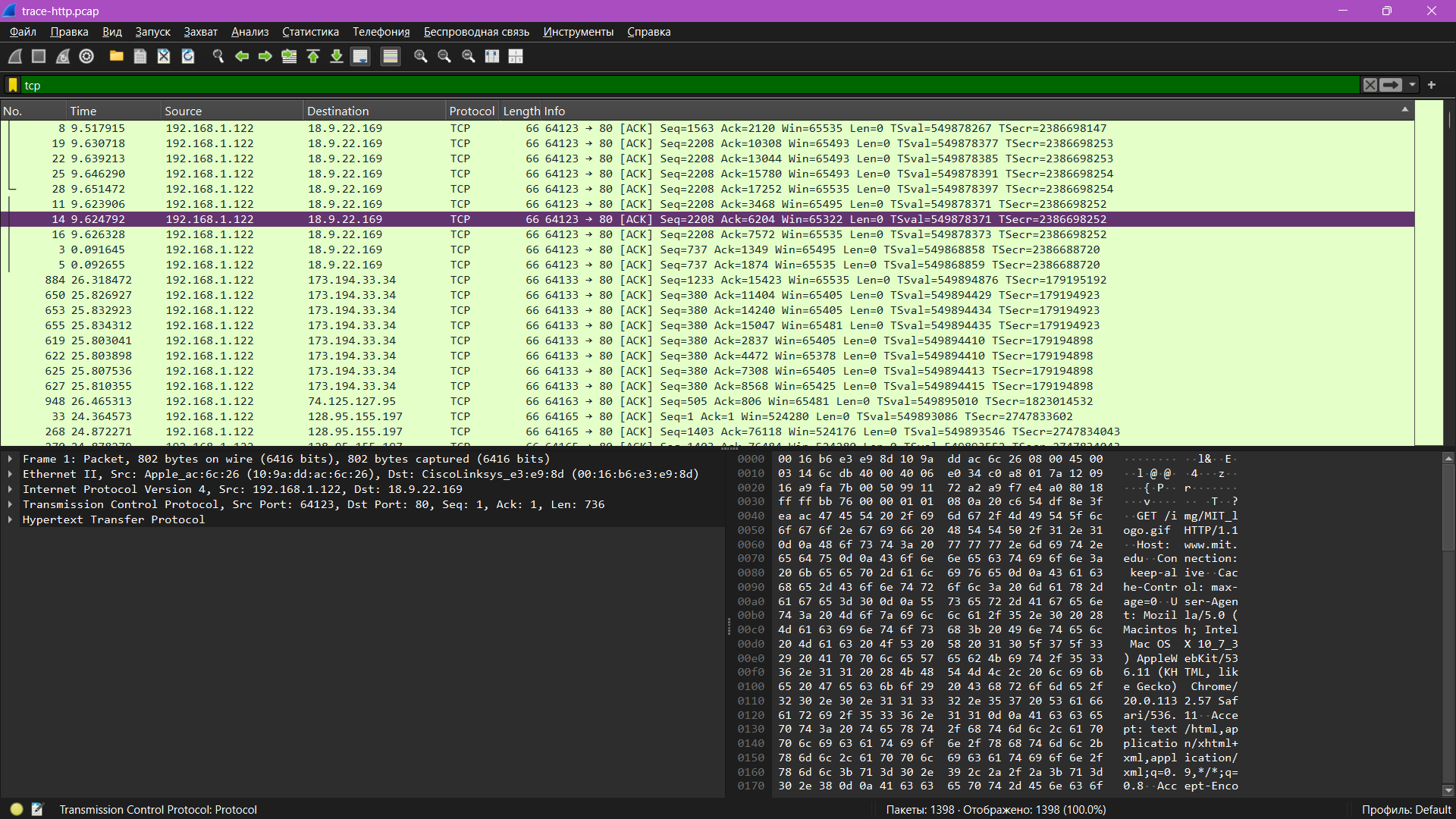
 

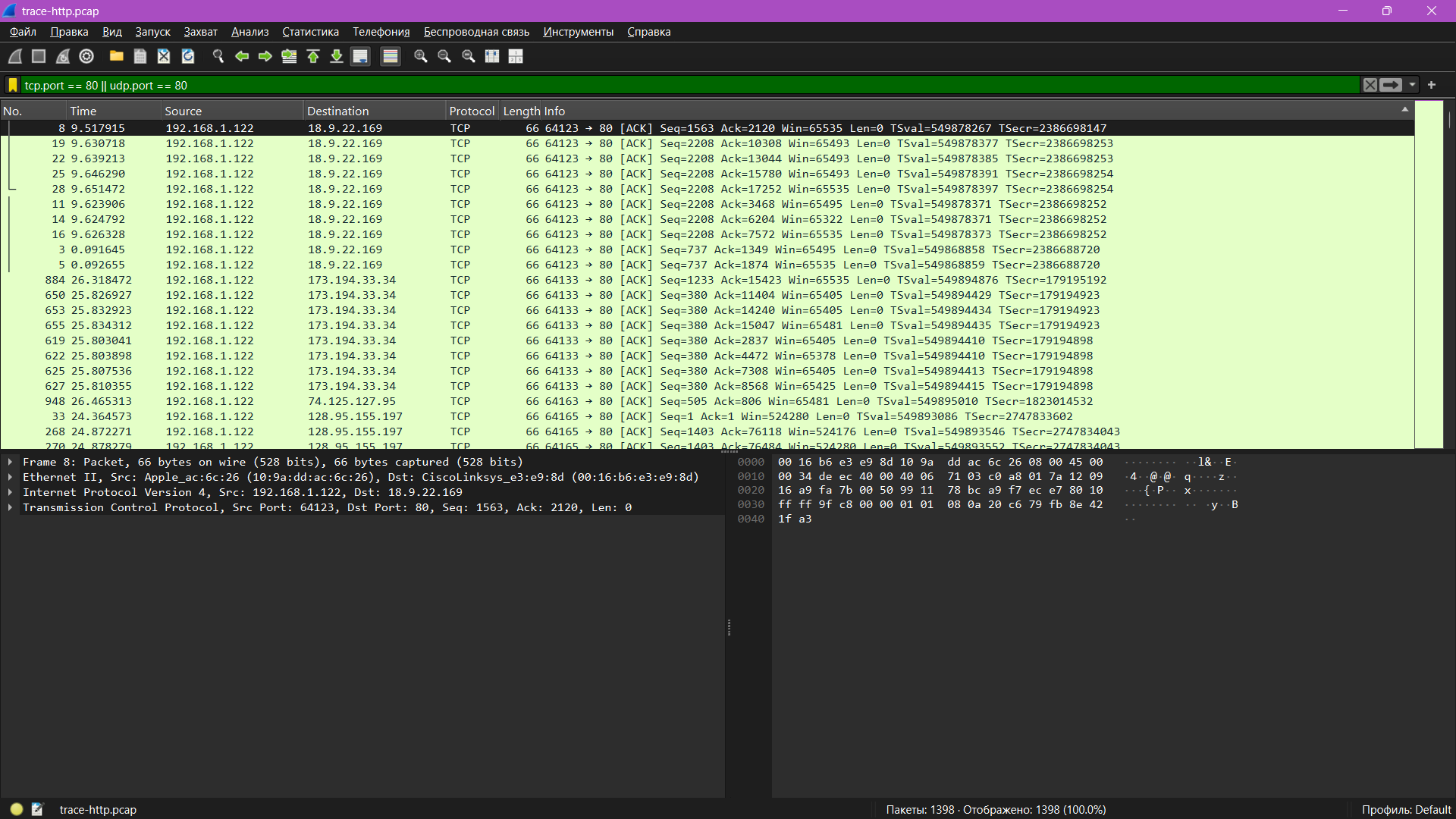
http.response

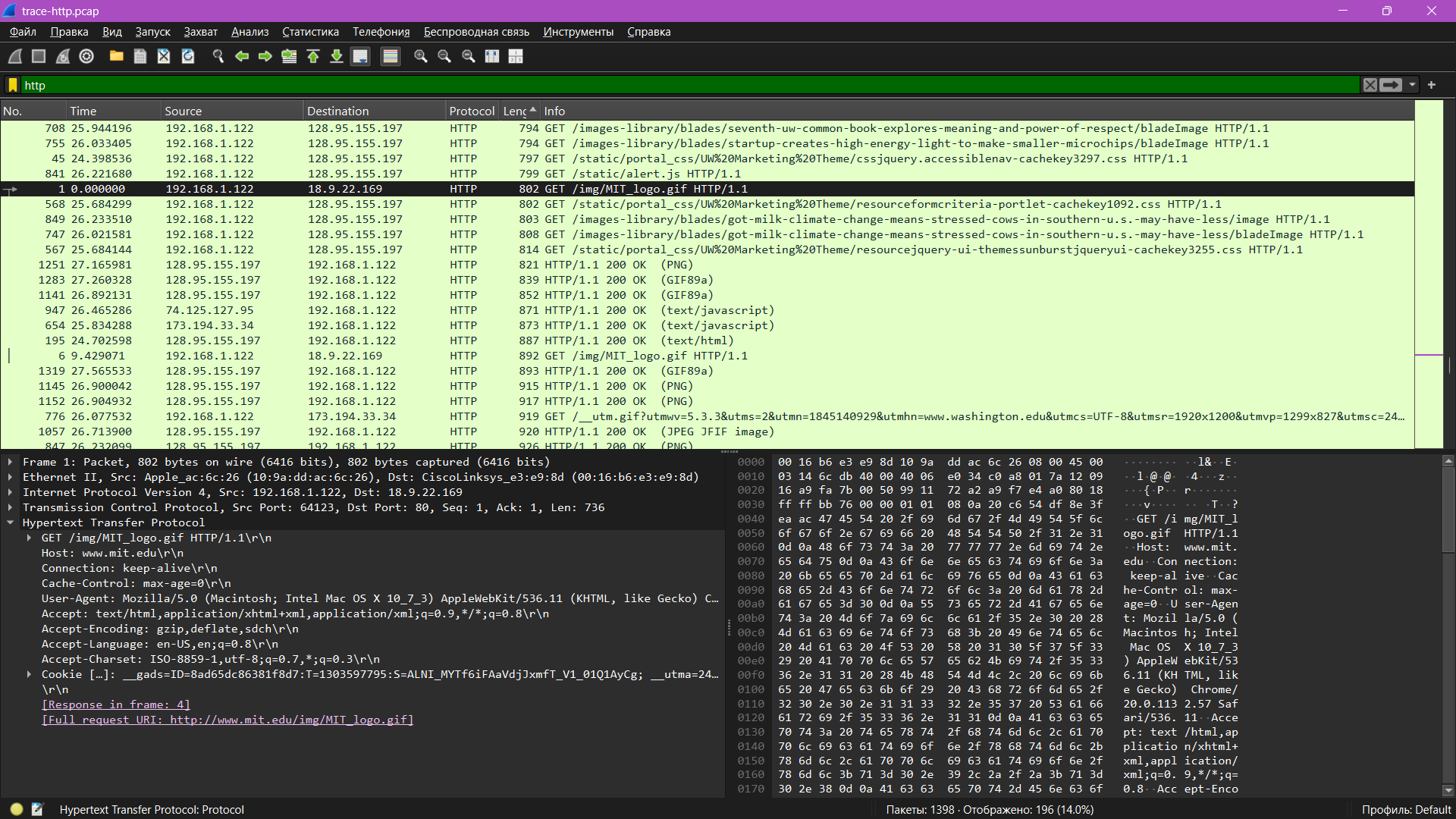
**Задание 2. Изучение трассировки запроса и ответа**

Чтобы сосредоточиться на трафике HTTP, введем и применим выражение фильтра «http». Этот фильтр отображает запросы и ответы HTTP, но не отдельные пакеты, которые в них участвуют, а весь список. Мы должны увидеть экран, как показано ниже. 

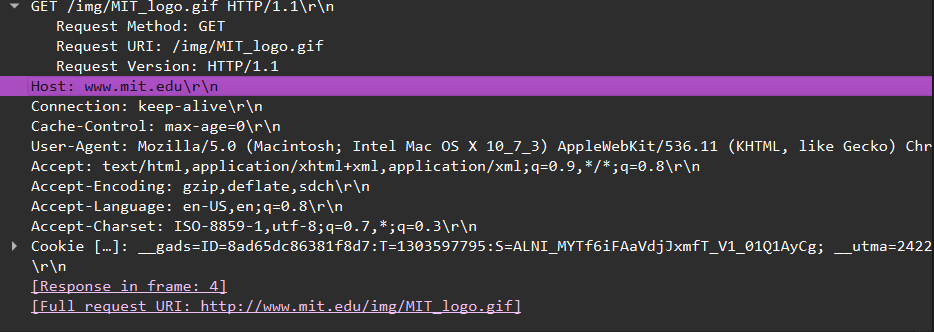
tcp

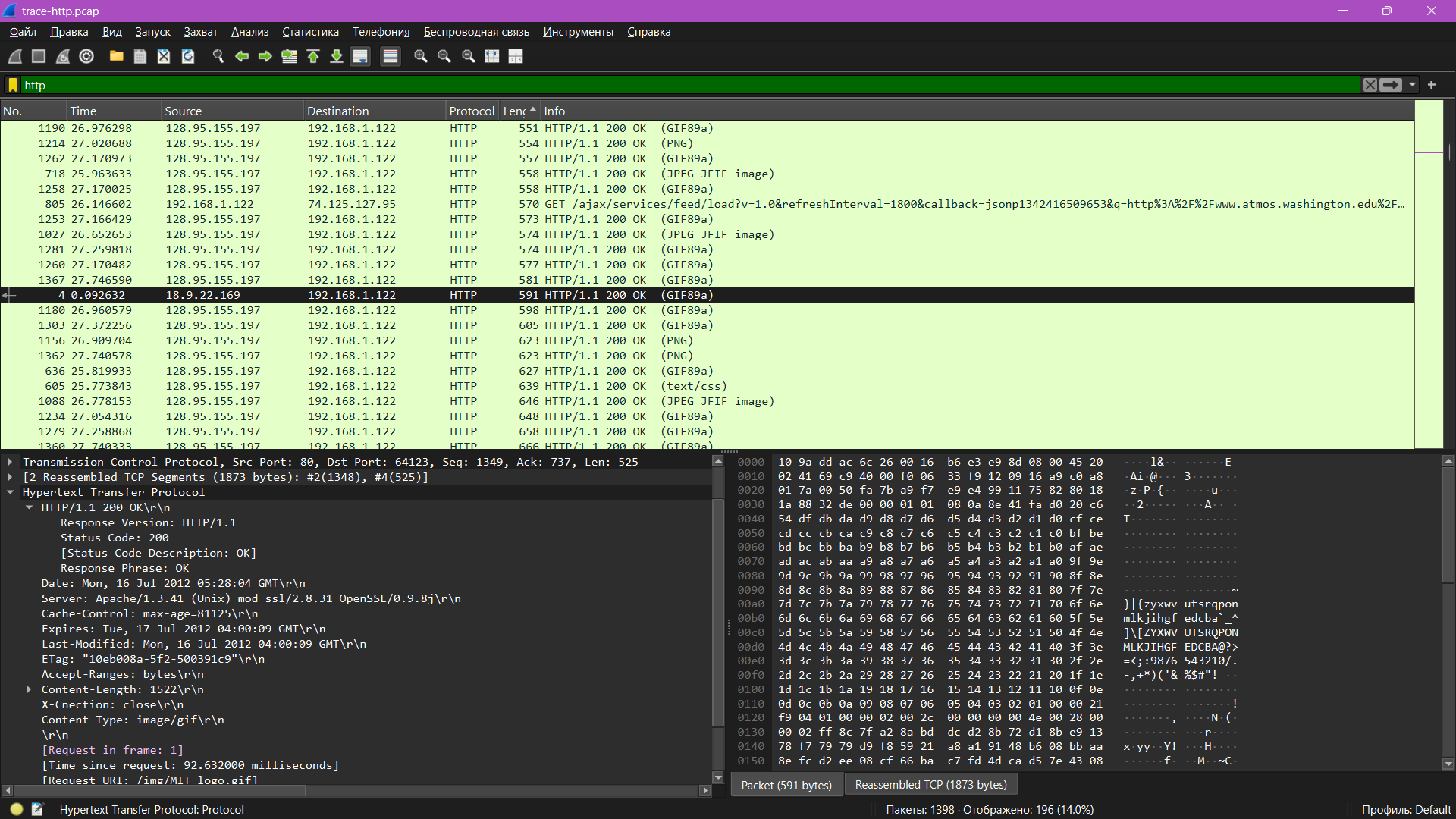
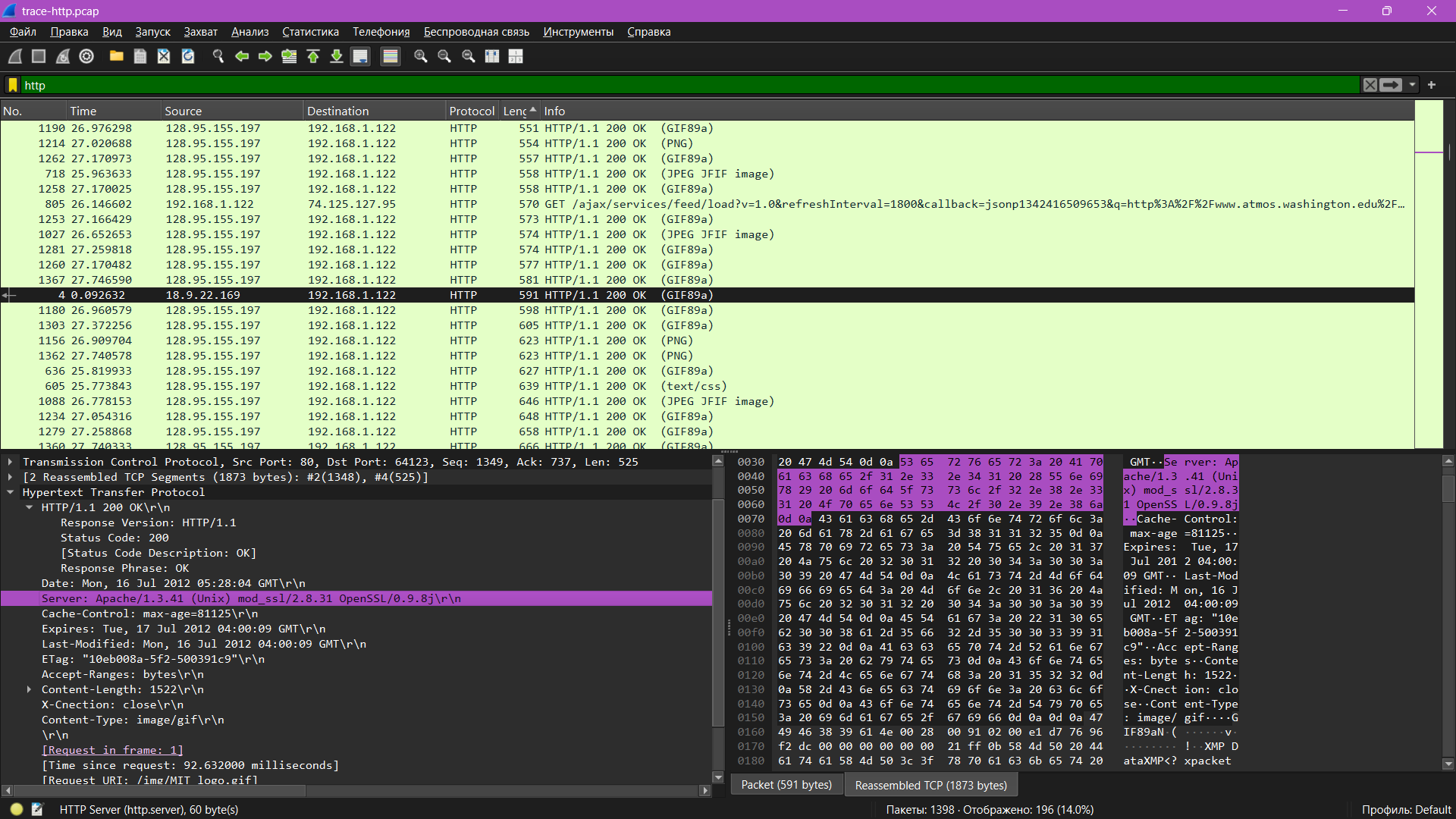
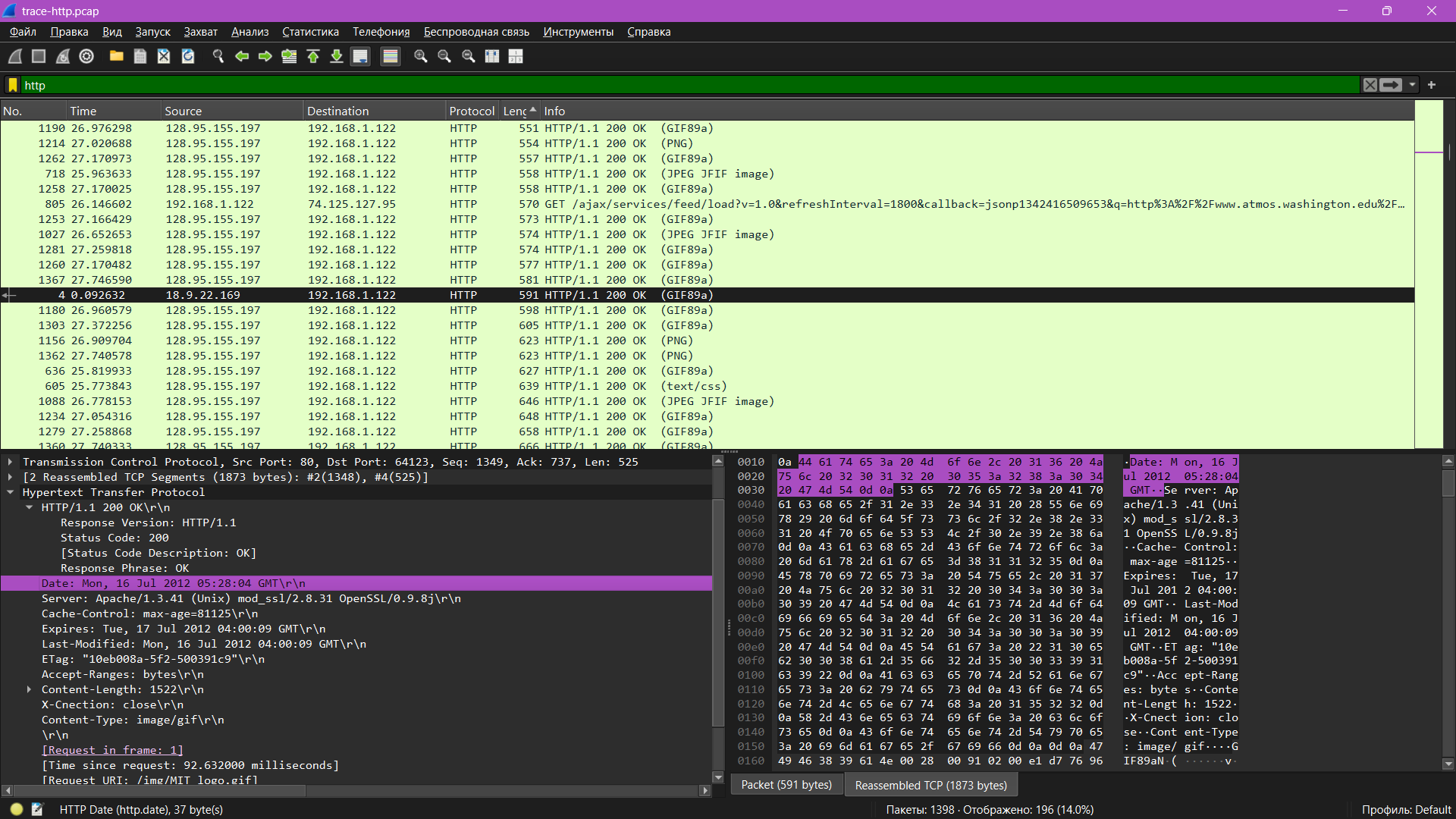
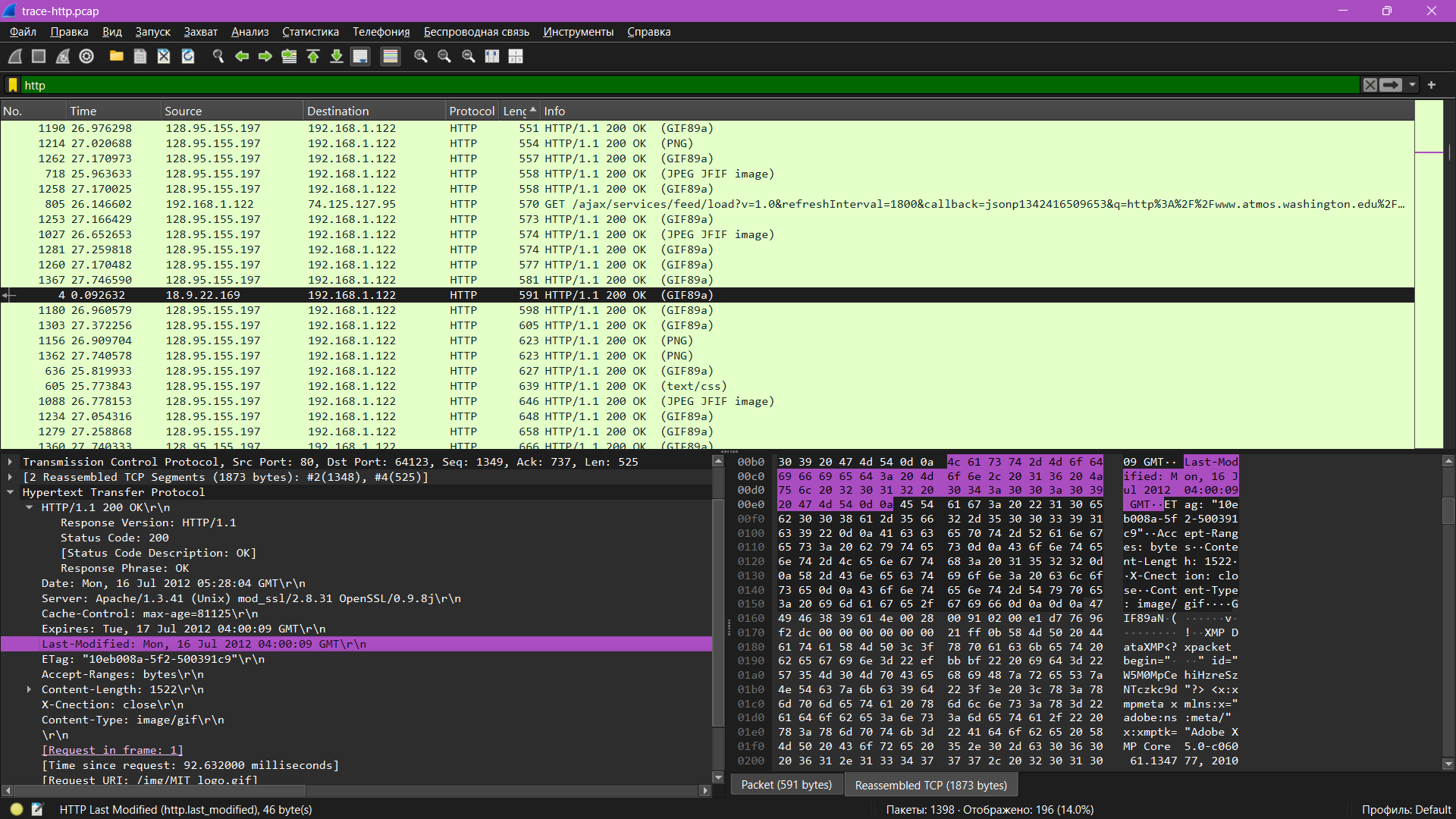
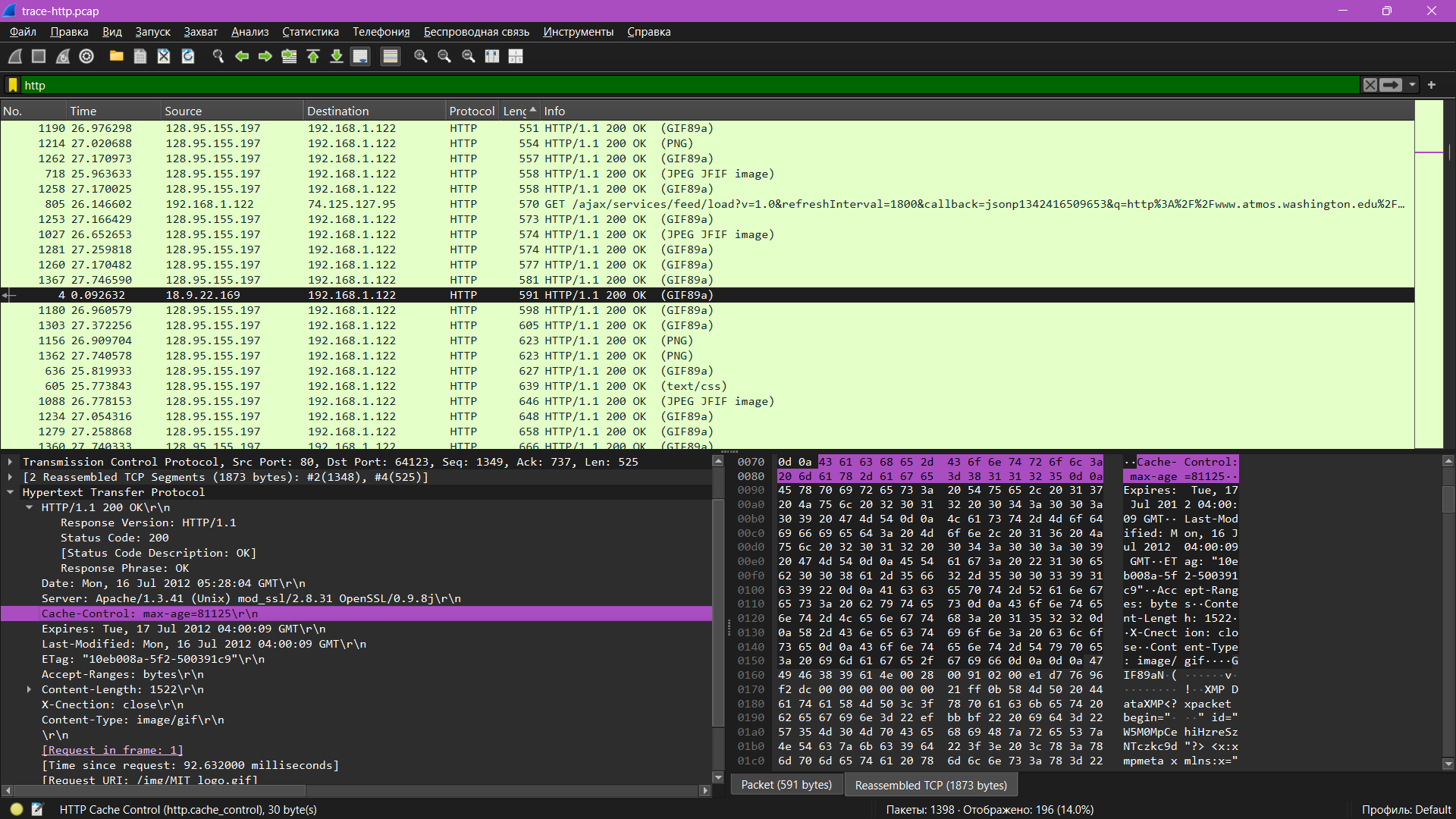
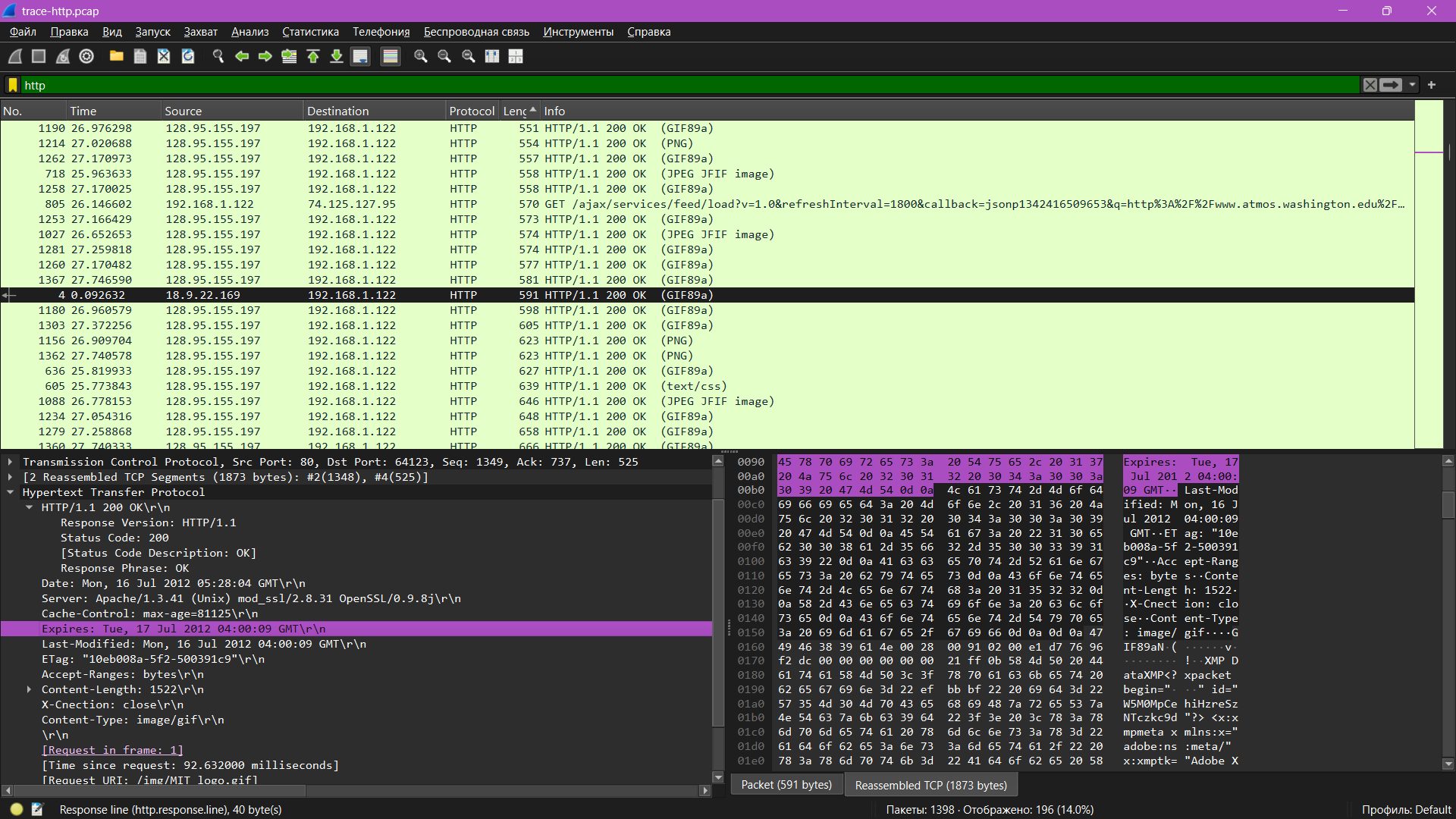
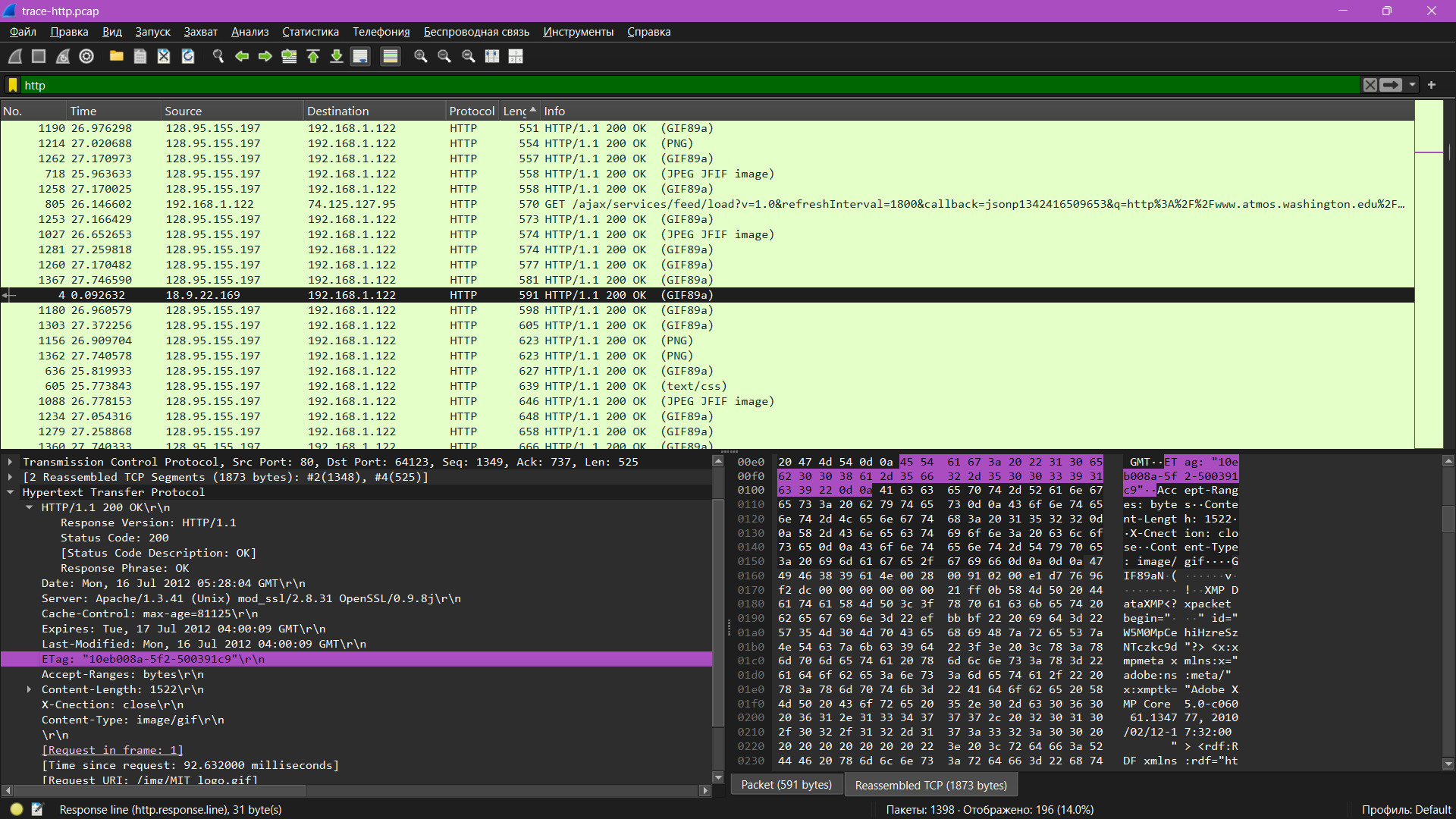
 tcp.port == 80 || udp.port == 80

 **Просмотр GET запроса и изучение HTTP-заголовка**



* **Host.** Обязательный заголовок, он идентифицирует имя (и порт) сервера.
* **User-Agent.** Вид браузера и его возможности.
* **Accept, Accept-Encoding, Accept-Charset, Accept-Language.**Описания форматов, которые будут приняты в ответе, например, text/html, включая его кодировку, например, gzip, и язык.
* **Cookie.**Имя и значение cookie-файлов, которые браузер хранит для веб-сайта.
* **Cache-Control.**Информация о том, как ответ может быть кэширован.

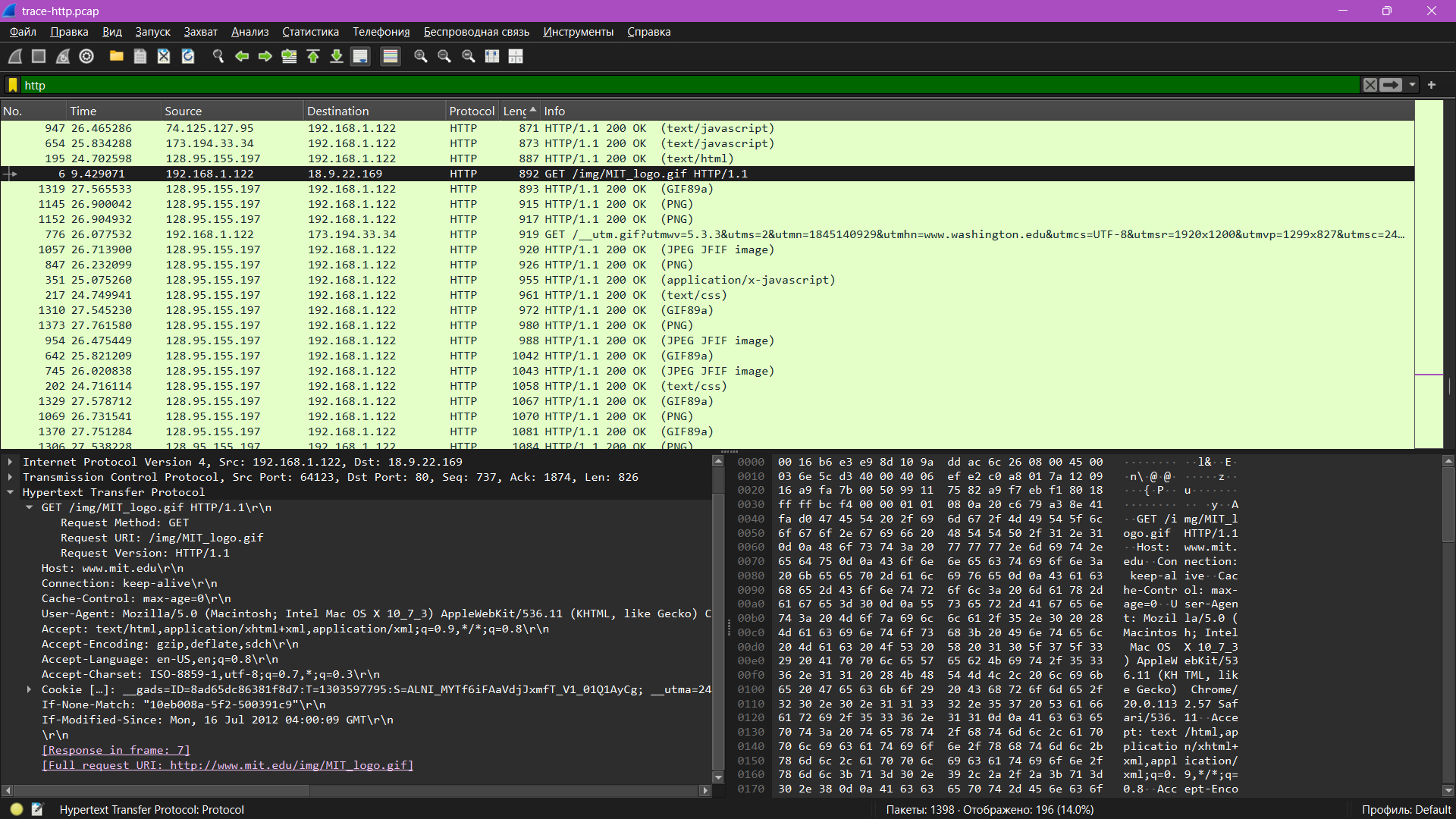


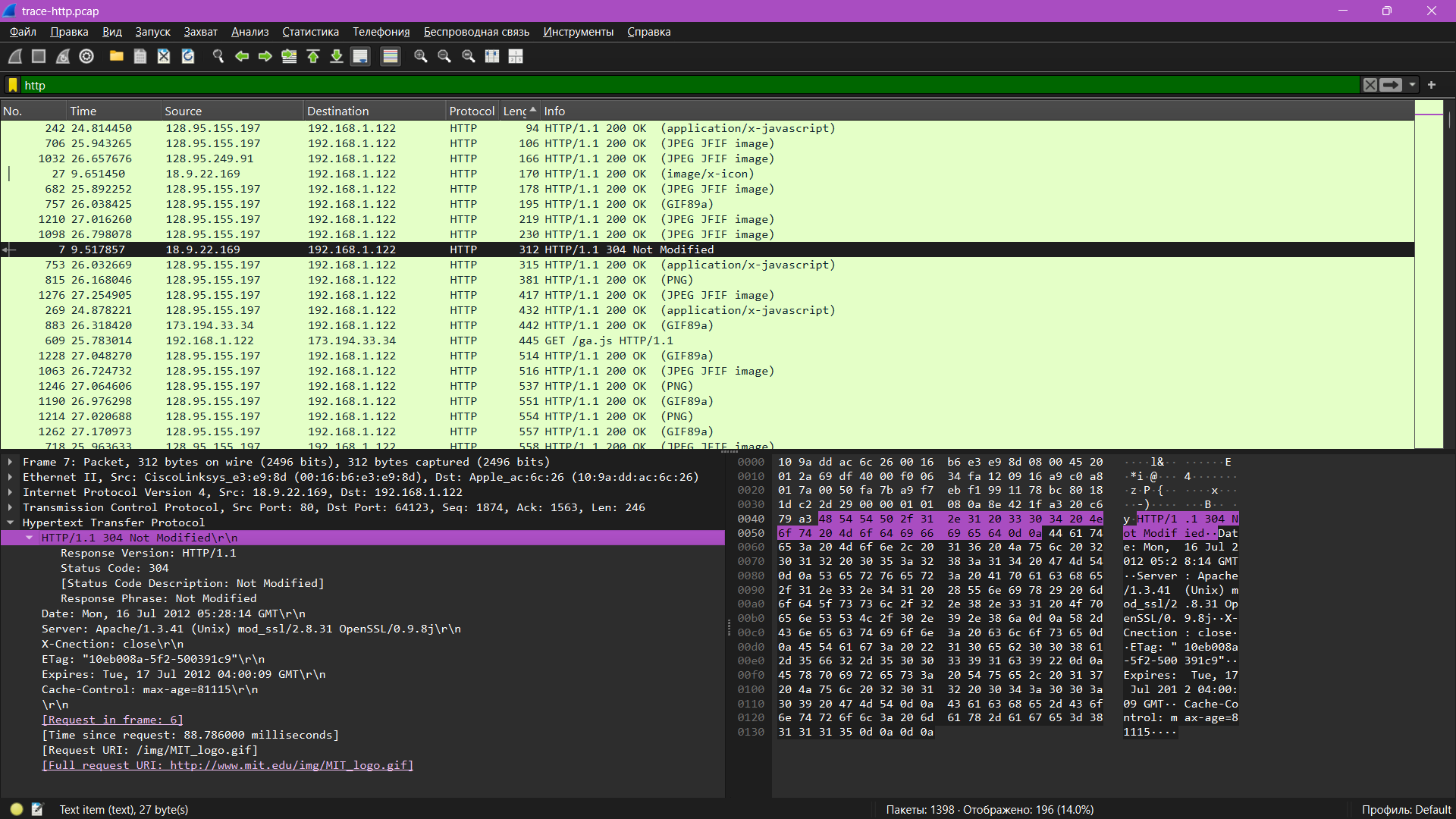
      

* **Server.** Тип сервера и его возможности.
* **Date, Last-Modified.** Время ответа и время последнего изменения контента.
* **Cache-Control, Expires, Etag.**Информация о том, как можно кэшировать ответ.

**Задание 3. Кэширование контента**

Второй блок записей в трассировке — это повторная выборка по первому URL. Этот блок дает нам возможность рассмотреть кэширование в действии, поскольку весьма вероятно, что изображение или документ не изменились и, следовательно, не нуждаются в повторной загрузке. Механизмы кэширования HTTP должны предоставить такую возможность.

Щелкните на третью строку в файле трассировки, т.е. на 892 GET /img/MIT\_logo.gif HTTP 1.1 

 Мы видим, что кэширование сработало, как и ожидалось, этот ответ не будет содержать контент. Вместо этого  
код статуса ответа — «**304 Not Modified**». Это сообщает браузеру, что контент не изменился  
по сравнению с предыдущей копией, и кэшированный контент затем может быть отображен.

**Задание 4. Сложные страницы**

Теперь мы рассмотрим четвертую выборку в трассировке. Эта выборка была для более сложной веб-страницы, которая, скорее всего, будет иметь встроенные ресурсы. Поэтому браузер загрузит исходный HTML плюс все встроенные ресурсы, необходимые для отображения страницы, плюс другие ресурсы, которые запрашиваются во время выполнения  
сценариев страницы. В этой выборке мы увидим, что одна страница может включать много GET. Щелкните по номеру 34, который является **750 GET / HTTP/1.1** (как показано ниже). 