**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ**

**ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ**

**ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»**

**ОТЧЕТ**

**по лабораторной работе №1**

«Анализ сетевого трафика»

**Выполнил: Брильянтов В. А., M3439**

**Проверил: Карапетян Г. А.**

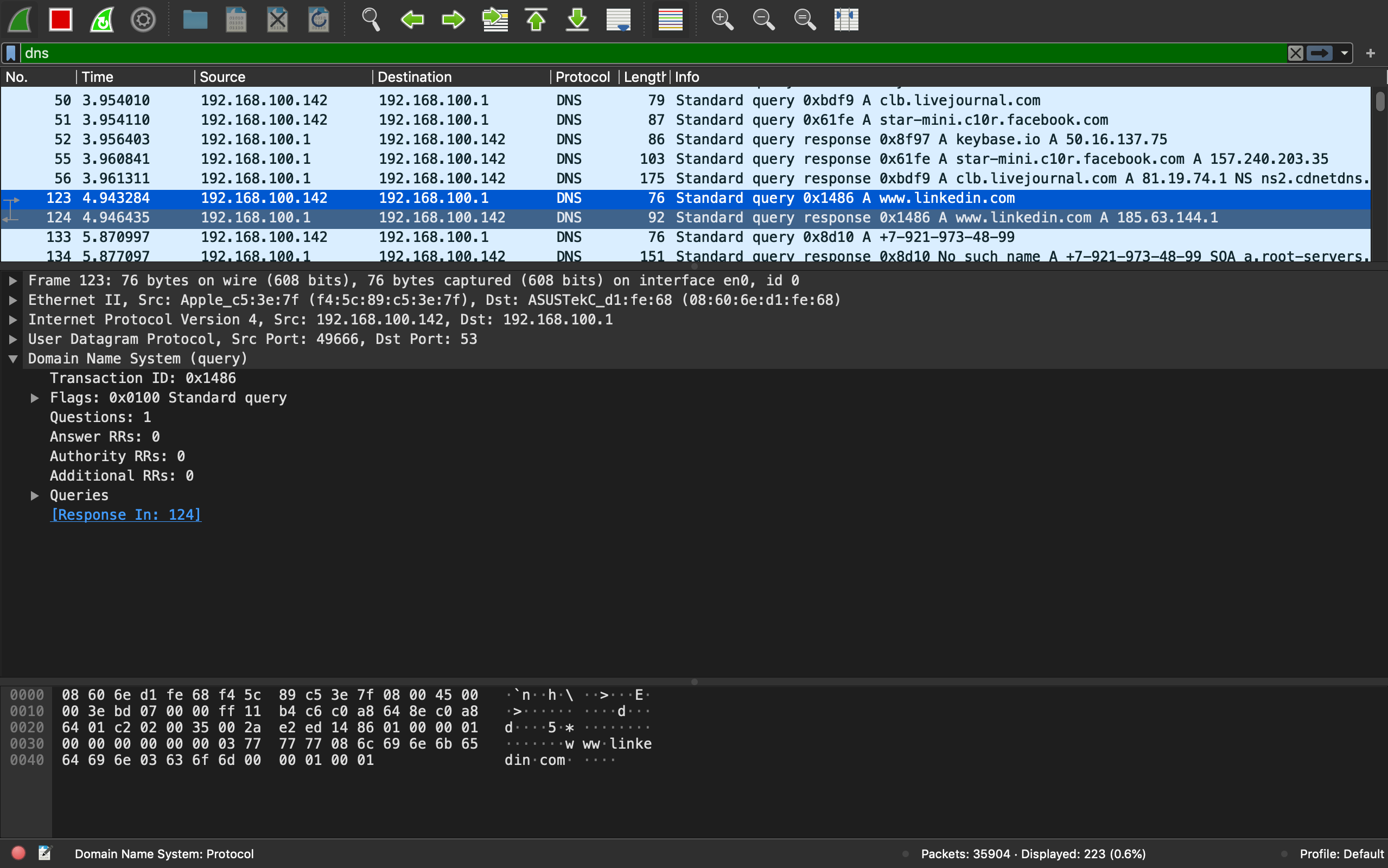
**Cанкт-Петербург, 2020**

**Цель работы**

Изучить структуру Ethernet-фрейма, IP, TCP-пакета. Получить навыки анализа сетевого трафика с помощью утилиты (Wireshark).  Трафик должен представлять из себя обмен данными по какому-нибудь протоколу прикладного уровня (DNS). Сделать сопоставление теоретических знаний и фактически полученных результатов (проанализировать структуры захваченных пакетов по сетевой модели TCP/IP).

**Ход работы**

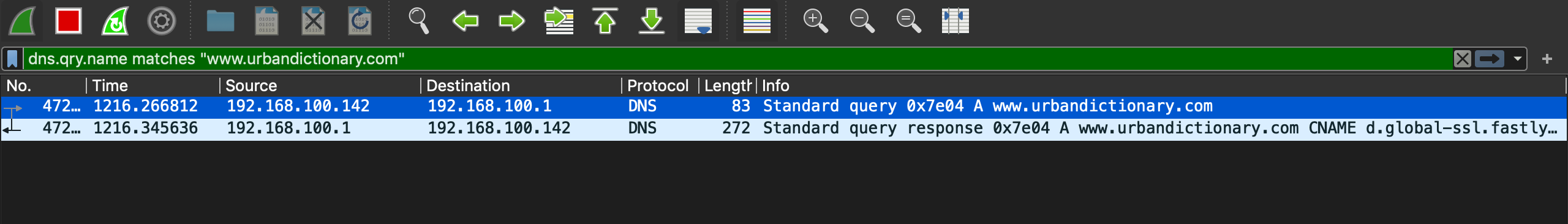
Проанализируем DNS трафик. Для этого наберем **dns** в строке фильтра программы Wireshark. Получим большое количество пойманных пакетов:



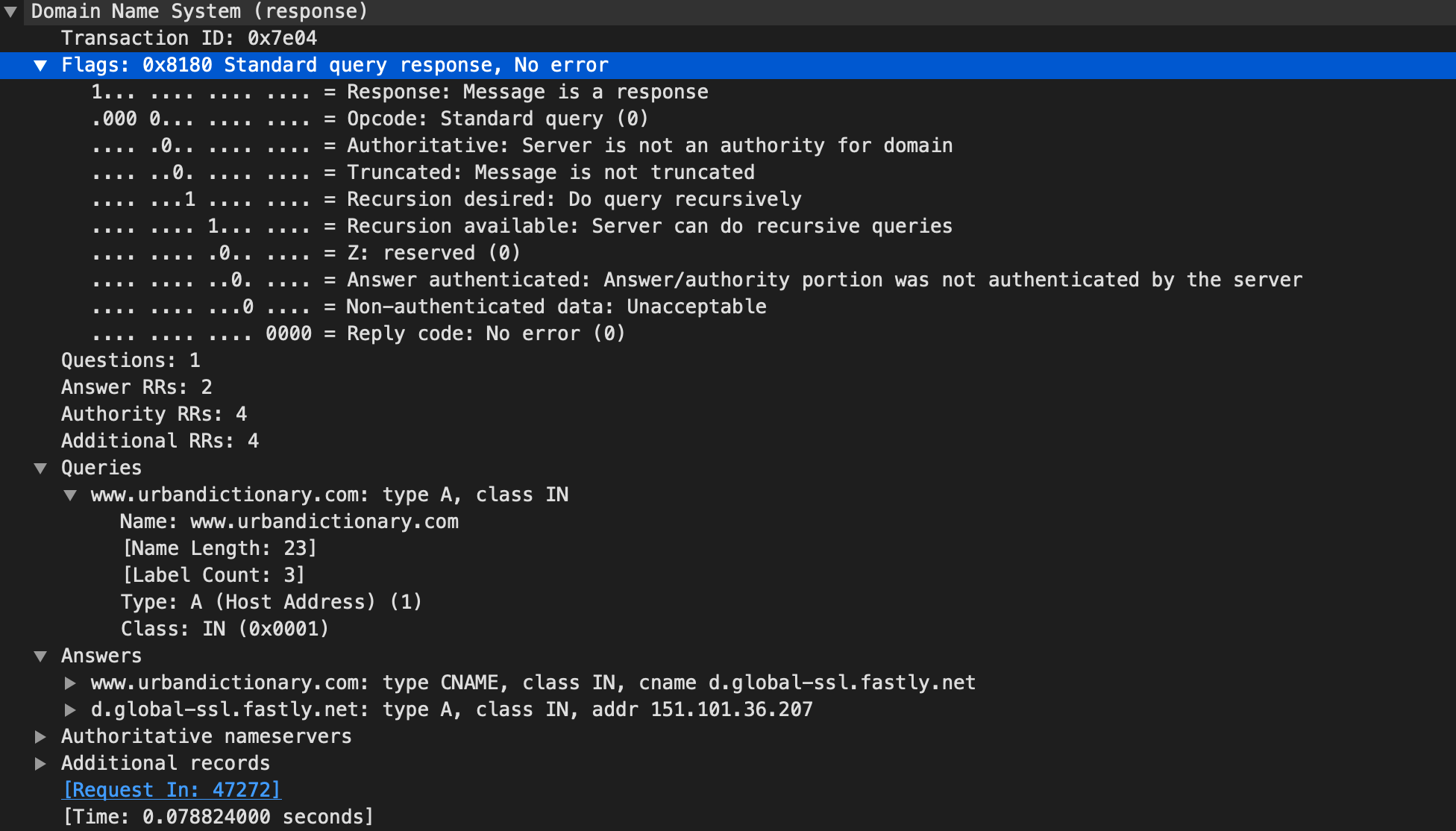
Теперь выберем для примера DNS-запрос, отправленный с целью получить информацию о домене https://www.urbandictionary.com. Для этого выставим фильтр **dns.qry.name matches "www.urbandictionary.com"**

1. **Прикладной уровень (DNS запрос):**

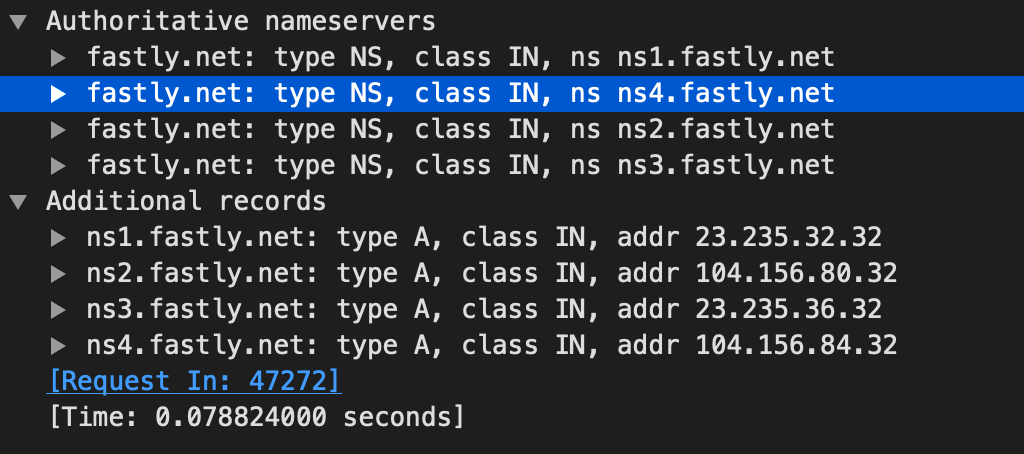
Логично, мы получим два пойманных пакета: request и response:



Т.к. оба пакета обязаны состоять из одинаковых полей (что действительно видно в Wireshark), для примера возьмем response:

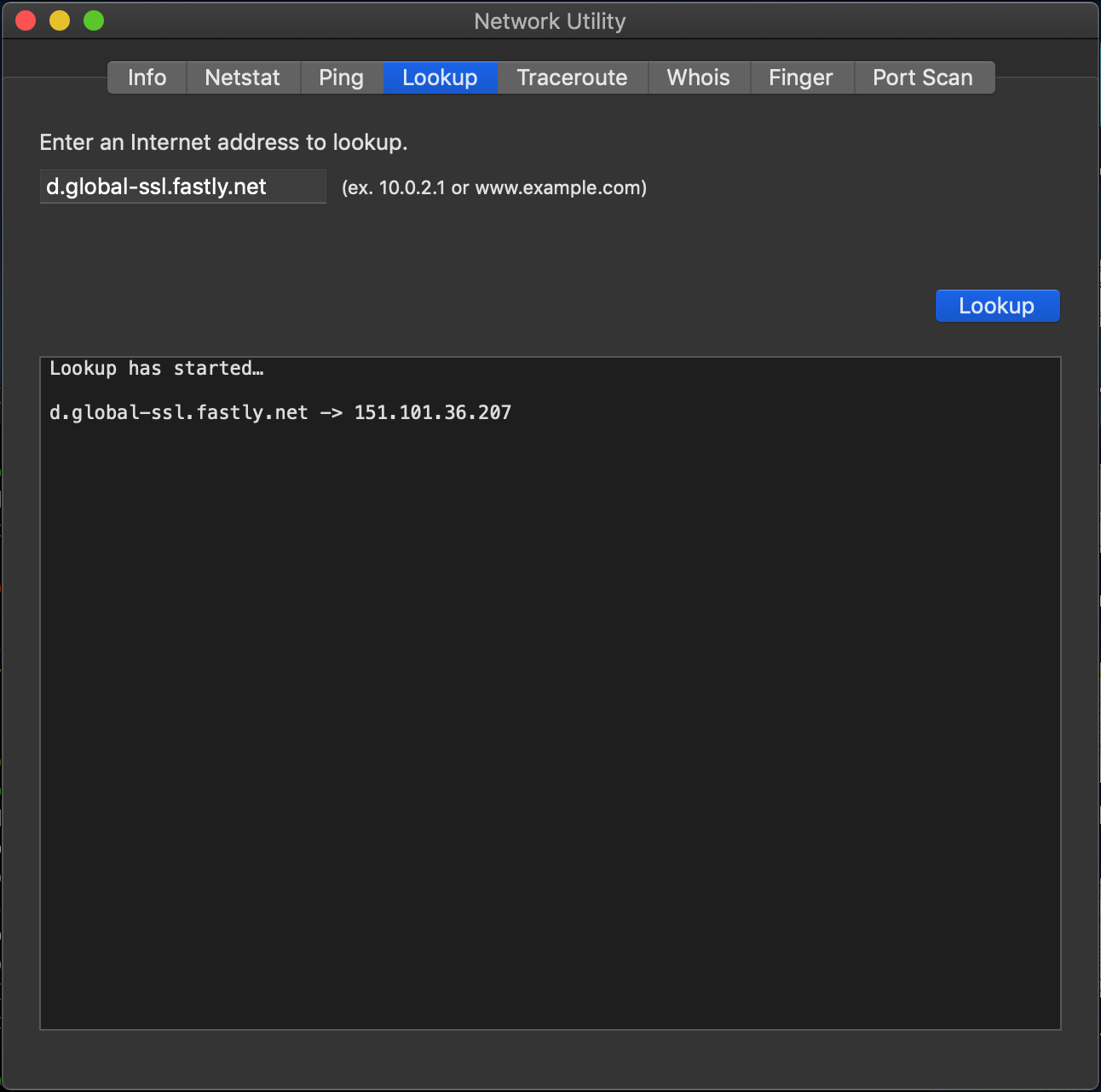
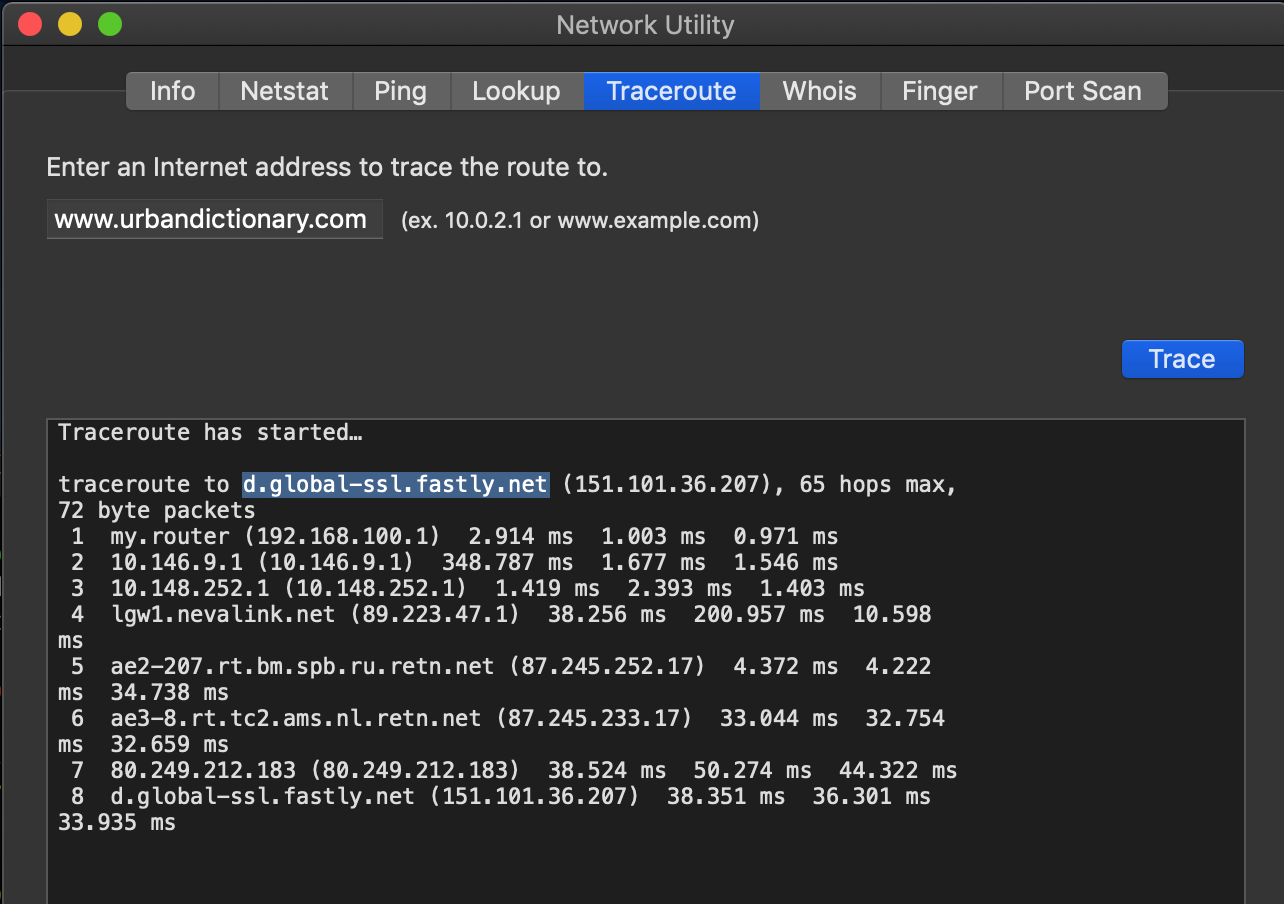


Т.к. содержимое последних 2 полей не попало в поле нашего обозрения, прикреплю их отдельно:



Итак, что мы видим?

Мы видим, что реальный DNS-пакет, пойманный и распознанный программой Wireshark, содержит поля (**Дисклеймер:** **описание полей взяты исключительно из прочтения описаний, увиденных Wireshark и преломленных под взглядом, воображением автора и его знанием английского. Нормальное описание полей, строго соответствующее теории, будет приведено позже**) :

* Flags
  + 0, если request, 1 если response
  + opcode, 0 означает Standard query
  + Authoritative – правда ли что сервер надежный для данного имени хоста
  + Truncated – правда ли что сообщение сокращенное
  + Recursive desired – правда ли что надо делать запрос рекурсивно
  + Recursive available – правда ли что сервер может запрос рекурсивно
  + Z – зарезервированное поле
  + Authenticated -- правда ли, что ответ был аутентифицирован сервером
  + Non-authenticated-data – принимаются ли неавторизированные данные
  + Reply code – код возврата (0 означает отсутствие ошибки)
* Questions (количество вопросов)
* Answer RRs (количество ответов)
* Athority RRs (количество Athoritative nameservers)
* Additional RRs (количество Additional records)
* Queries
  + Name – доменное имя, которое нам интересно ([**www.urbandictionary.com**](http://www.urbandictionary.com)**)**
  + Type – тип записи. В нашем случае тип A, что означает Address Record (то есть IPv4-адрес)
  + Class – класс запроса. В нашем случае это IN, что является сокращением от “Internet”.
* Answers
  + Name – имя запрошенного хоста. В нашем случае случился рекурсивный запрос, и в первом случае это [**www.urbandictionary.com**](http://www.urbandictionary.com), которыймы и спрашивали, а во втором случае – найденный первым запросом промежуточный hostname.
  + Type – тип ответа. Мы получили ответы двух типов: CNAME и A.
    - A означает “дай мне IPv4 адрес”
    - CNAME означает “дай мне canonical name”, по идее canonical name должен означать, что по выданному имени (hostname) возможно получить доступ к тому же самому IP адресу, однако по адресу <http://d.global-ssl.fastly.net> мне не удалось попасть на сайт urbandictionary. Оказалось, что fastly – это сервис, позволяющий обеспечить безопасность и load-balancing. Попробовав утилиту Network Utility в Mac OS (аналог trace route и dnslookup) я получил следующий результат:  
      

Также попробовав в терминале:

PING d.global-ssl.fastly.net (151.101.36.207): 56 data bytes

64 bytes from 151.101.36.207: icmp\_seq=0 ttl=56 time=34.977 ms

1. ytes from 151.101.36.207: icmp\_seq=1 ttl=56 time=34.819 ms

было проверено, что адрес d.global-ssl.fastly.net действительно пнгуется и ведет на запрашиваемый ресурс.

* + Class. В обоих случаях IN(ternet)
  + Time to live – TTL
  + Data length – длина ответа
  + Что-то соответствующее классу:
    - В случае CNAME это поле CNAME
    - В случае A это поле Address
* Athoritative nameservers
  + Type – аналогично тому, что описано выше
  + Class – аналогично тому, что описано выше
  + ns = name server: В нашем случае это [ns1—ns4].fastly.net
* Additional records
  + Type – аналогично тому, что описано выше
  + Class – аналогично тому, что описано выше
  + Address – по всей видимости IP адрес того name server, который соответствует записи. Проверим это. Для первых двух в списке указаны адреса 23.235.32.32 и 104.156.80.32:

ping ns1.fastly.net

PING ns1.fastly.net (23.235.32.32): 56 data bytes

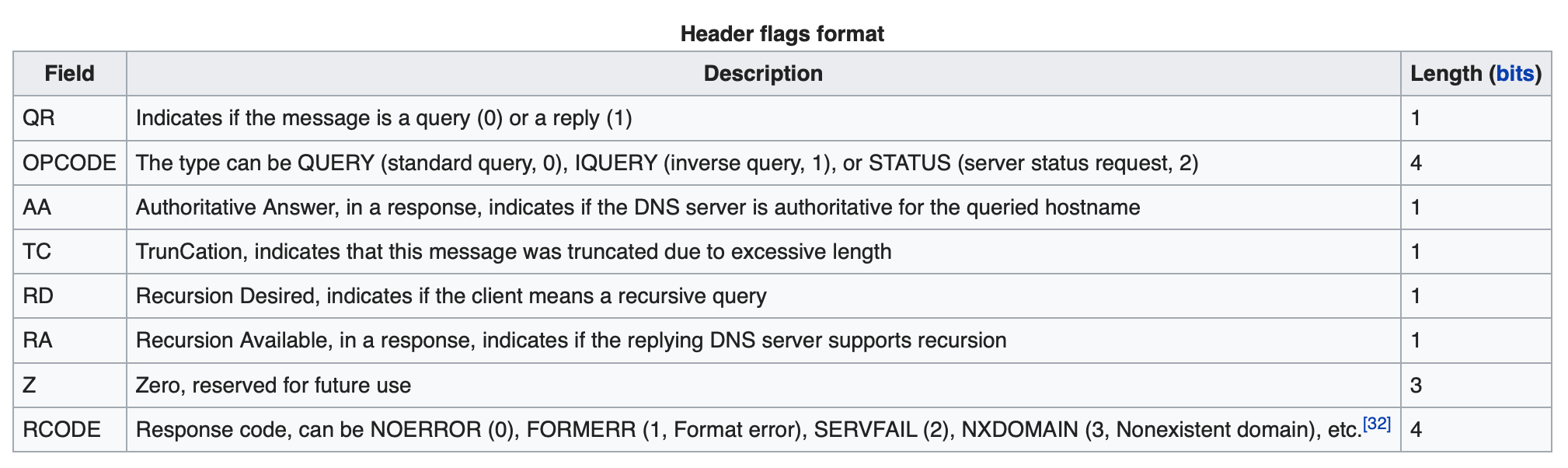
Адрес совпал

ping ns2.fastly.net

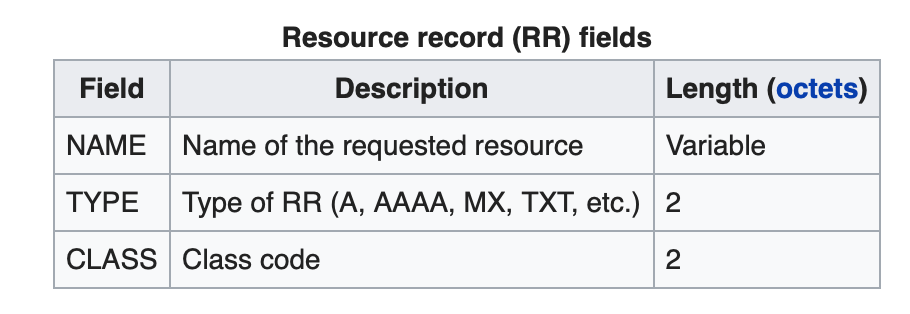
PING ns2.fastly.net (104.156.80.32): 56 data bytes

Снова адрес совпал! Значит предположение о назначении этих Additional Records рискует быть верным…

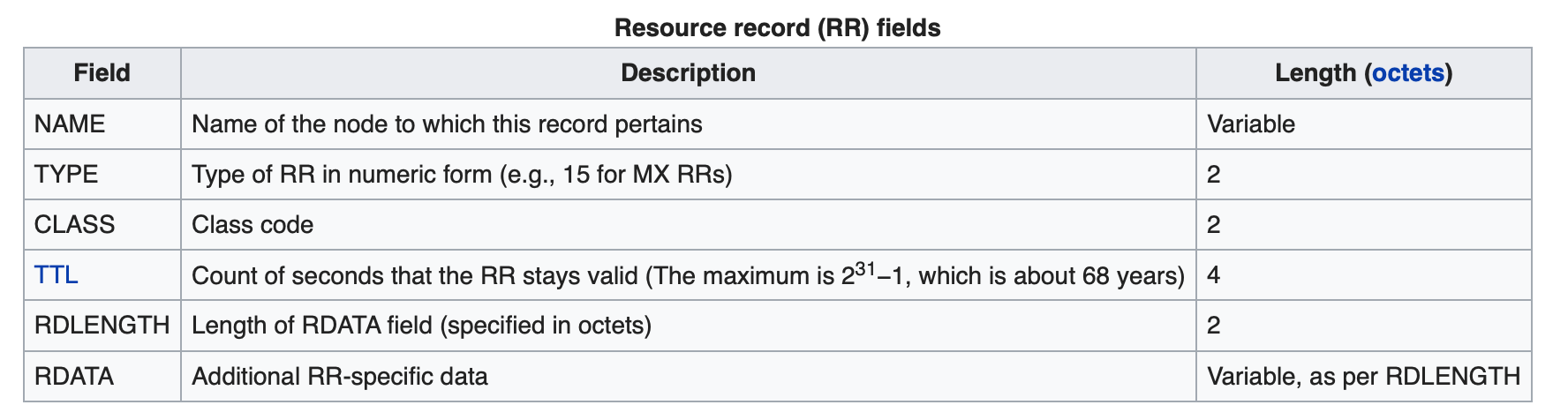
Теперь теория:



Флаги, действительно такие, и их назначение удалось достаточно точно угадать.

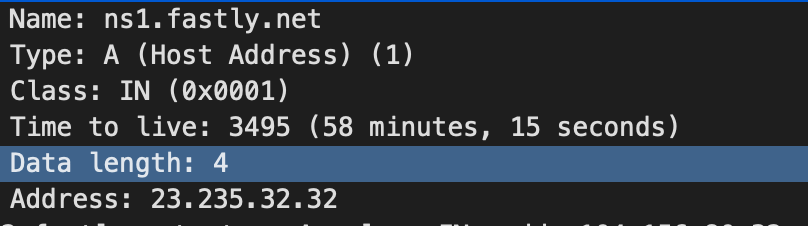


Поля секции Question совпали^

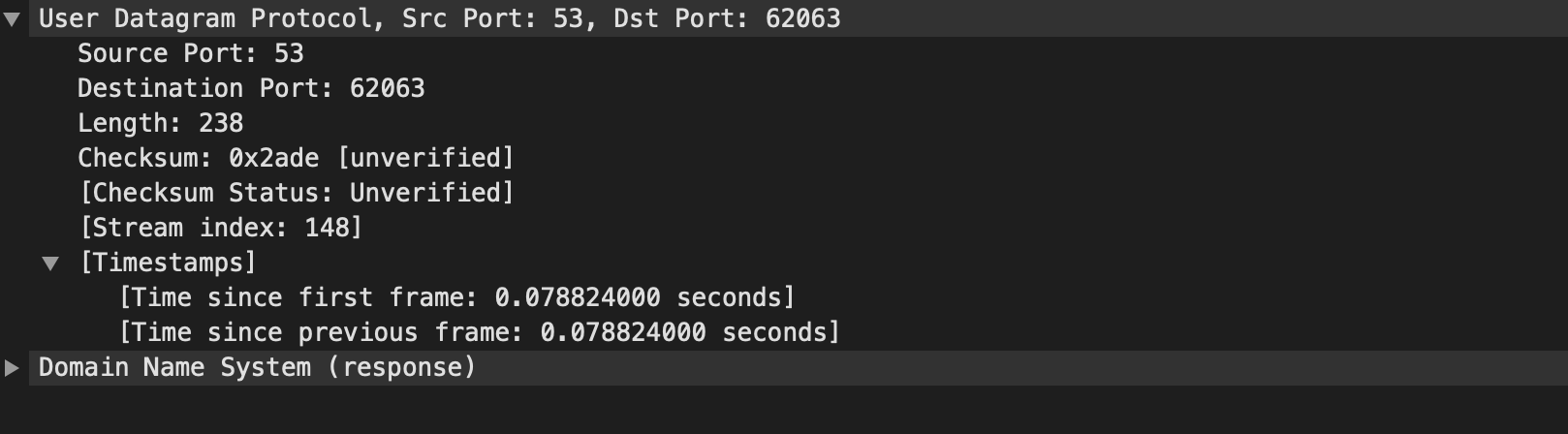


Поля ответов также совпали, правда Wireshark немного видоизменил их названия для лучшей читаемости, а также вместо RDATA написал релевантное для данной записи поле (Address, Cname).

**Дополнительное замечание: я не заметил поля NAME, TTL и RDLENGTH в Additional records, что заставило меня использовать другие утилиты для проверки своей гипотезы о том, что каждый адрес соответствует своему NS-серверу. На самом деле это действительно так, и видно в Wireshark, для того чтобы это заметить необходимо было развернуть соответствующие вкладки:**

****

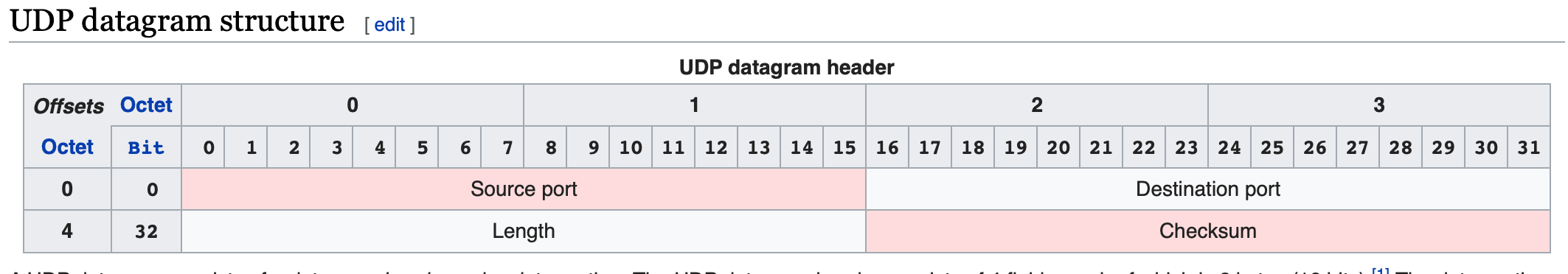
1. **Транспортный уровень (UDP):**



Мы обнаружили следующие поля:

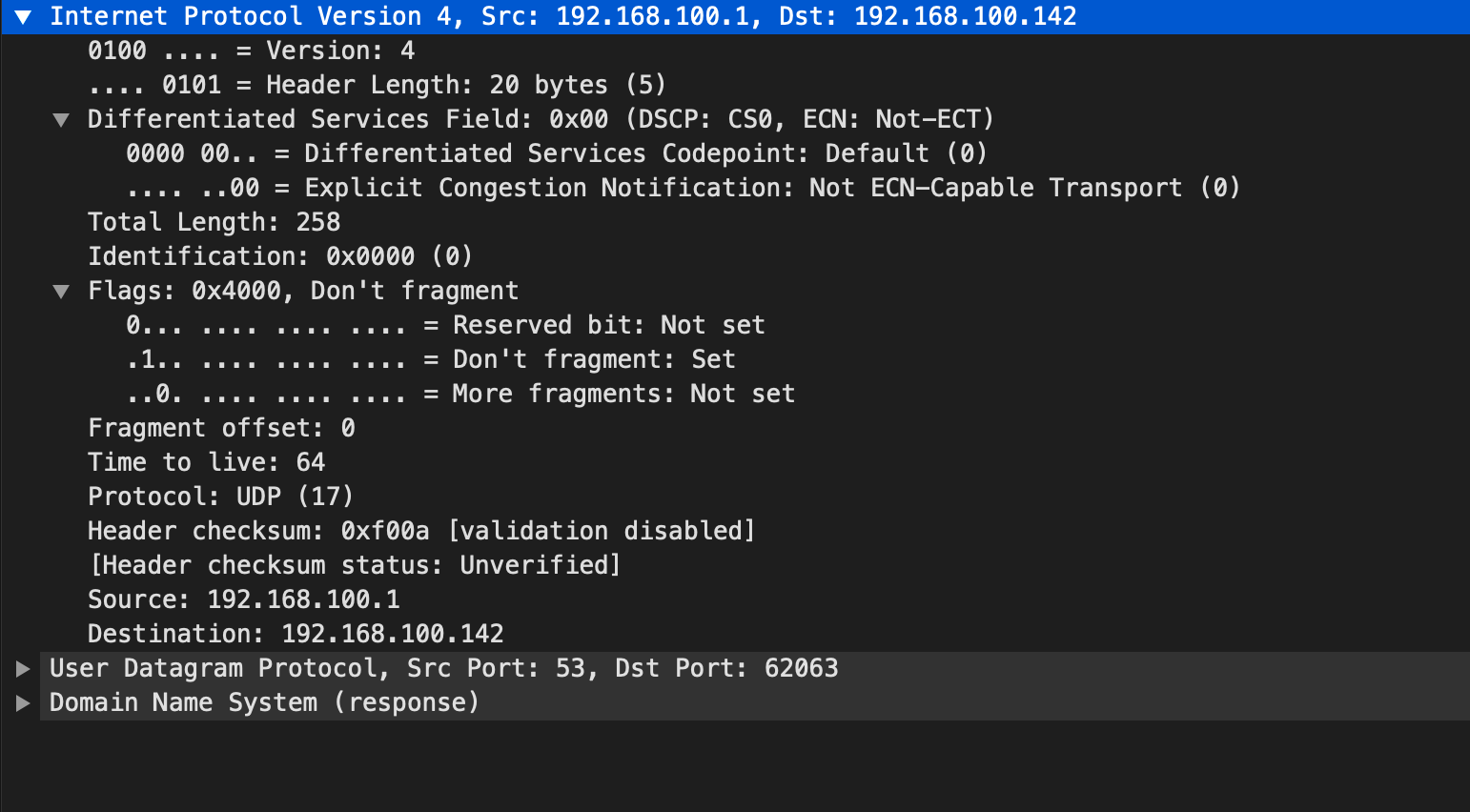
* Source port: 53 – порт отправителя
* Destination port: 62063 – порт получателя
* Length: 238 – длина пакета
* Checksum: 0x2ade – контрольная сумма

Сравним с теорией:



Действительно, именнл это мы и получили!

1. **Межсетевой уровень (IPv4):**



Мы видим поля:

* Version – версия протокола (4 означает IPv4)
* Header Length: 20 bytes – длина заголовка
* Differentiated Services Field
  + DSCP: 0 (Default). Это приоритет пакета. Поле используется в роутерах в основном для контроля трафика. Например, трафик с рекламой не будет отбрасываться, тк приносит прибыль, а трафик youtube будет, тк потеря одного пакета не сильно скажется на качестве контента
  + ESN: 0 (Non ECN-Capable Transport) – нет поддержки ECN
* Total length: 258 – длина пакета
* Identification: 0 – идентификатор пакета
* Flags:
  + Reserved – бесполезный для нас флаг
  + Don't fragment: выставлен
  + More fragments: не выставлен
* Fragment offset: 0 -- смещение фрагмента
* TTL: 64
* Protocol: UDP – протокол верхнего уровня
* Header checksum -- контрольная сумма заголовка
* Source – адрес отправителя. 192.168.100.1 – это адрес моего роутера и по совместительству это адрес DNS-сервера, используемого моим ноутбуком.

scutil --dns | grep 'nameserver\[[0-9]\*\]'

**nameserver[0]** : 192.168.100.1

**nameserver[0]** : 192.168.100.1

Для эксперимента выставим DNS-сервер в 8.8.8.8 (Google’s DNS):

╰─➤  networksetup -setdnsservers Wi-fi 8.8.8.8

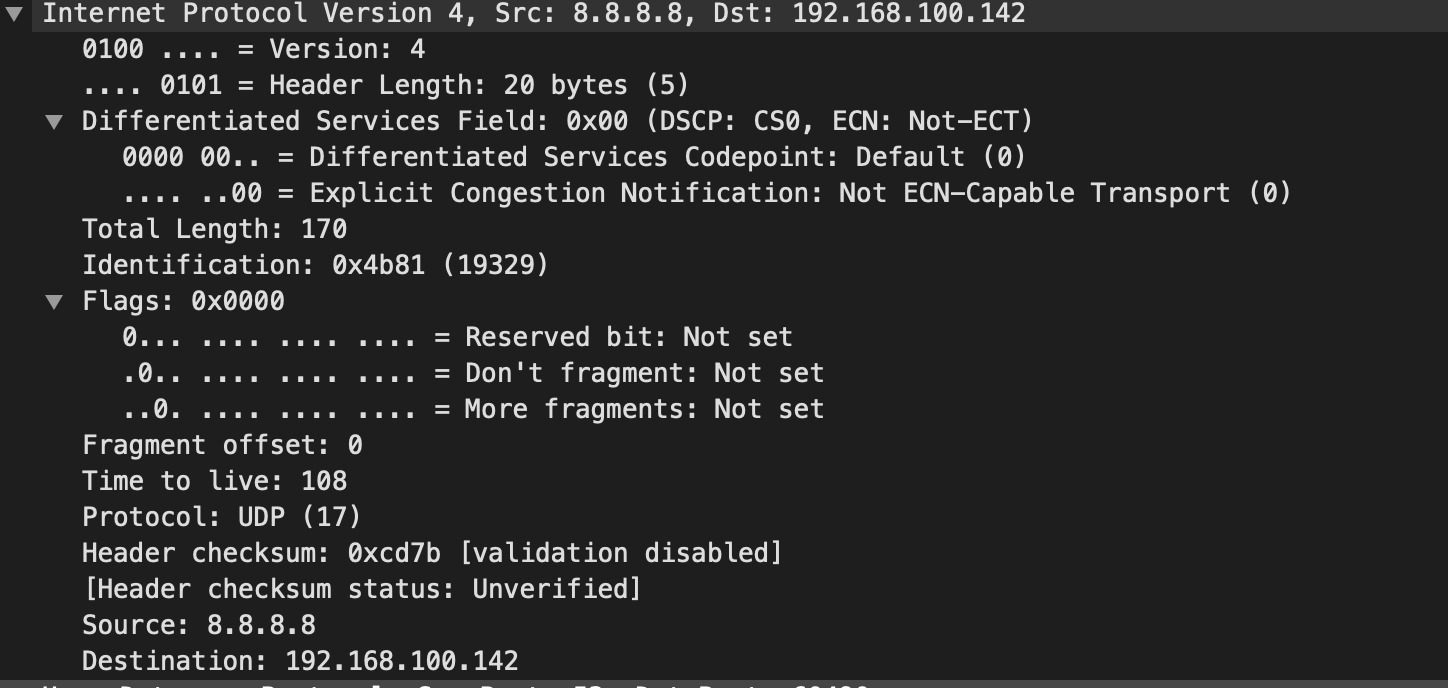
╭─Vadim@MacBook-Pro-Vadim.local **~/Documents/networks**  ‹master\*›

╰─➤  scutil --dns | grep 'nameserver\[[0-9]\*\]'

**nameserver[0]** : 8.8.8.8

**nameserver[0]** : 8.8.8.8

После этого откроем заново urbandictionary.com и обновим поиск в Wireshark:

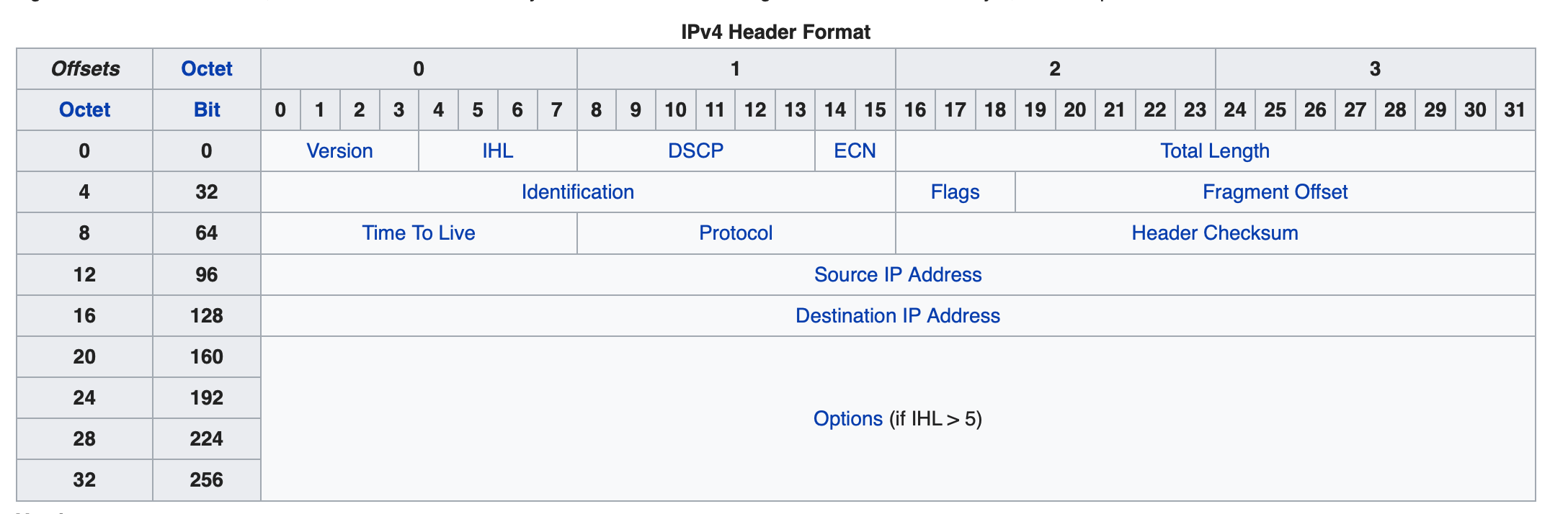


Действительно, Source в IP заголовке поменялся, следовательно он соответствует IP-адресу DNS-сервера!

* Destination – адрес получателя. 192.168.100.142 – это мой адрес:

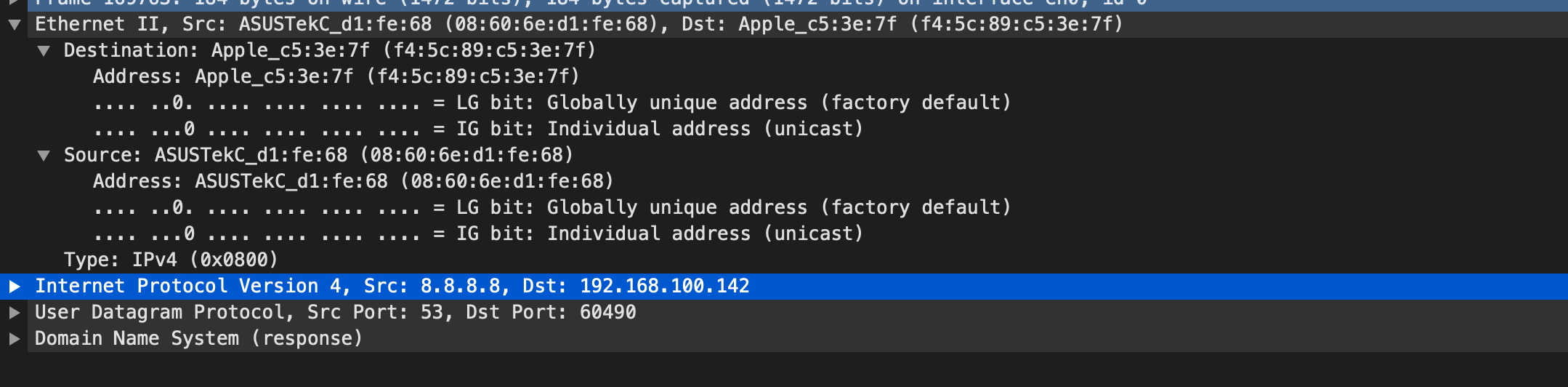
ifconfig

inet 192.168.100.142

Теперь сравним с теорией:  


Действительно, все кроме опций мы увидели. Однако они должны присутствовать только если IHL > 5, а у нас он равен 5 (см. Header Length).

1. **Сетевой уровень (Ethernet):**



Поля:

* Destination:
  + Address: f4:5c:89:c5:3e:7f – MAC-адрес моего компьютера.

Проверим. Мы знаем, что 192.168.100.142 – мой IP адрес.

╰─➤  arp 192.168.100.142

? (192.168.100.142) at f4:5c:89:c5:3e:7f on en0 ifscope permanent [ethernet]

Да, совпало!

* + LG bit
  + IG bit
* Source:
  + Address: 08:60:6e:d1:fe:68 – MAC-адрес моего роутера.

Проверим. Мы знаем, что 192.168.100.1 – его IP адрес.

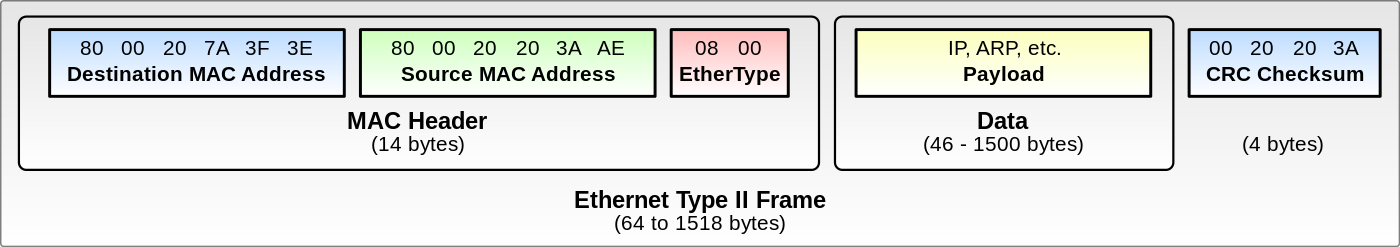
╰─➤  arp 192.168.100.1

? (192.168.100.1) at 8:60:6e:d1:fe:68 on en0 ifscope [ethernet]

Да, совпало!

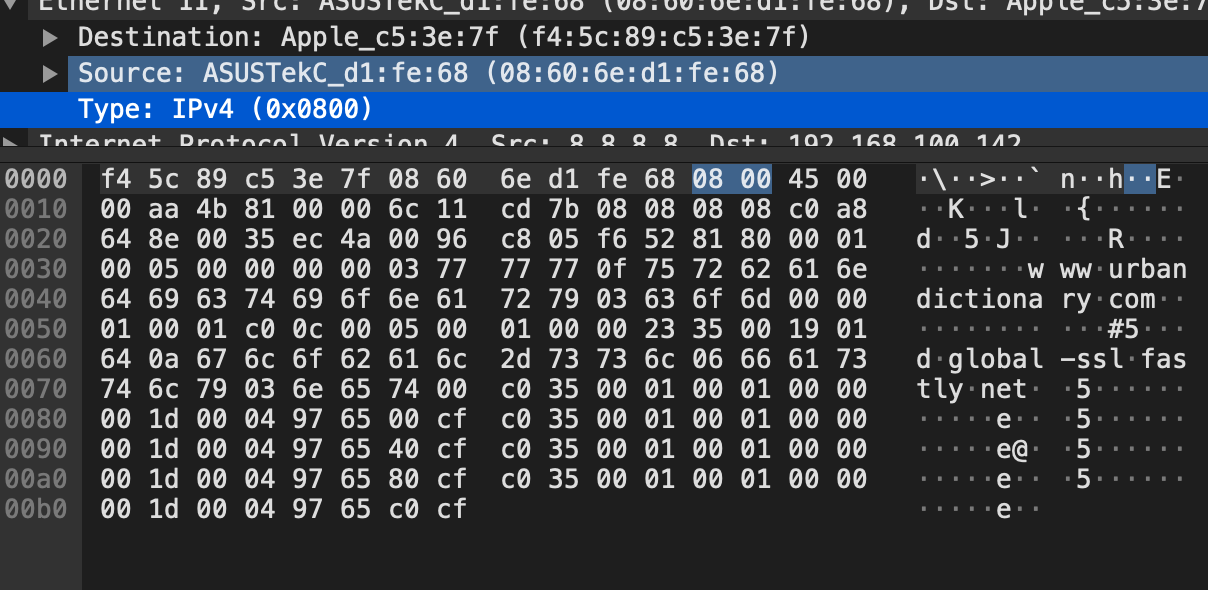
* + LG bit
  + IG bit
* Type: IPv4 – тип протокола верхнего уровня

Сравним с теорией:



Destination MAC, Source MAC– совпали, Ethertype : Ethernet-II видно по названию.

Type



Если подсветить в Wireshark необходимые байты, то мы увидим желанные 0800, что в теории должно означать как раз IPv4 (For example, an EtherType value of 0x0800 signals that the frame contains an [IPv4](https://en.wikipedia.org/wiki/IPv4) datagram).

**Вывод**

В ходе лабораторной работы был проведен анализ сетевого трафика с помощью утилиты Wireshark. Структура пакетов ожидаемо соответствует теоретическим знаниям.