

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт цифровых технологий, электроники и физики (ИЦТЭФ)
Кафедра вычислительной техники и электроники (ВТиЭ)

Лабораторная работа № 02

**Знакомство с анализаторами трафика на примере Wireshark. Захват и
анализ сетевых пакетов с помощью Wireshark.**

Выполнили: студенты 595 гр.

_____ А.В. Осипов

_____ А.В. Лаптев

_____ А.Е. Половинкин

Проверил: доцент кафедры ВТиЭ

_____ А.В. Калачёв

Лабораторная работа защищена

«___» _____ сентября _____ 2022 г.

Оценка _____

Барнаул 2022

Цель работы: Ознакомиться с программным обеспечением Wireshark, научиться захватывать и анализировать пакеты с использованием сниффера, выявлять возможные проблемы в сети.

Задачи работы:

1. Теоретическая часть

Основной инструмент для наблюдения за сообщениями, которыми обмениваются элементы исполняемого протокола, называется анализатор пакетов (или сниффер).

Как следует из названия, он анализирует (перехватывает) сообщения, которые отправляются или получаются вашим компьютером; он также обычно сохраняет и/или отображает содержимое различных полей протокола этих перехваченных сообщений.

Анализатор пакетов является пассивной программой. Он только следит за сообщениями, отправленными и полученными приложениями и протоколами, запущенными на вашем компьютере, но сам никогда не отправляет пакеты.

Полученные пакеты тоже никогда явно не адресуются анализатору. Он просто получает копию этих пакетов.

На рис. 1 показана структура анализатора пакетов. В правой части рис.1 находятся протоколы (в данном случае, Интернет-протоколы) и приложения (например, веб-браузер или FTP-клиент), которые обычно работают на вашем компьютере.

Анализатор пакетов (в пунктирном прямоугольнике) является дополнением к обычному программному обеспечению вашего компьютера и состоит из двух частей.

Библиотека захвата пакетов получает копию каждого кадра канального уровня, который отправляется или получается компьютером. Сообщения, которыми обмениваются протоколы более высокого уровня, такие как HTTP, FTP, TCP, UDP, DNS или IP, в конечном счете, заключены в кадры канального уровня, которые передаются через физический носитель, такой, как кабель Ethernet.

На рис. 1 показано предположение, что физическим носителем является Ethernet, и поэтому все протоколы верхних уровней, в конечном счете, инкапсулируются в кадр Ethernet. Захват всех кадров канального уровня, таким образом, дает все сообщения, отправленные/полученные всеми протоколами и приложениями, выполняющимися на вашем компьютере.

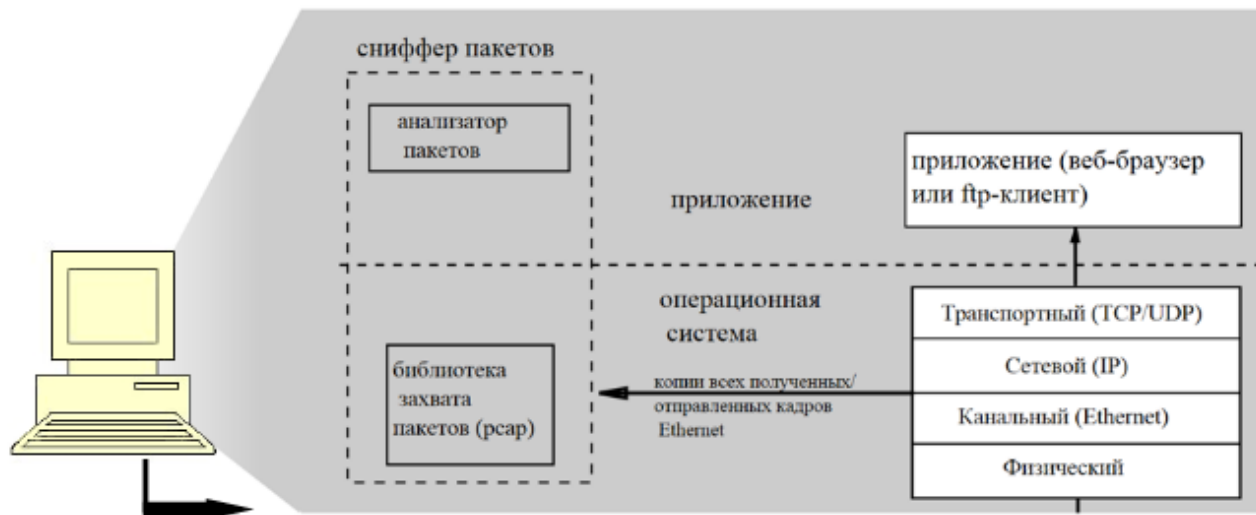


Рис. 1. Структура анализатора пакетов.

Вторым компонентом является анализатор пакетов, который отображает содержимое всех полей в протокольном сообщении. Чтобы сделать это, анализатор пакетов должен «понимать» структуру всех сообщений, которыми обмениваются протоколы.

2. Выполнение работы

Начало работы

Настройка Wireshark.

1. После запуска приложения переходим на вкладку Capture и выбрать пункт Interfaces.
2. В открывшемся окне ставим галочку напротив пункта «Подключение по локальной сети»
3. Кликаем Start для начала процесса захвата пакетов.

Пример выполнения данных действий представлен на рис. 2.

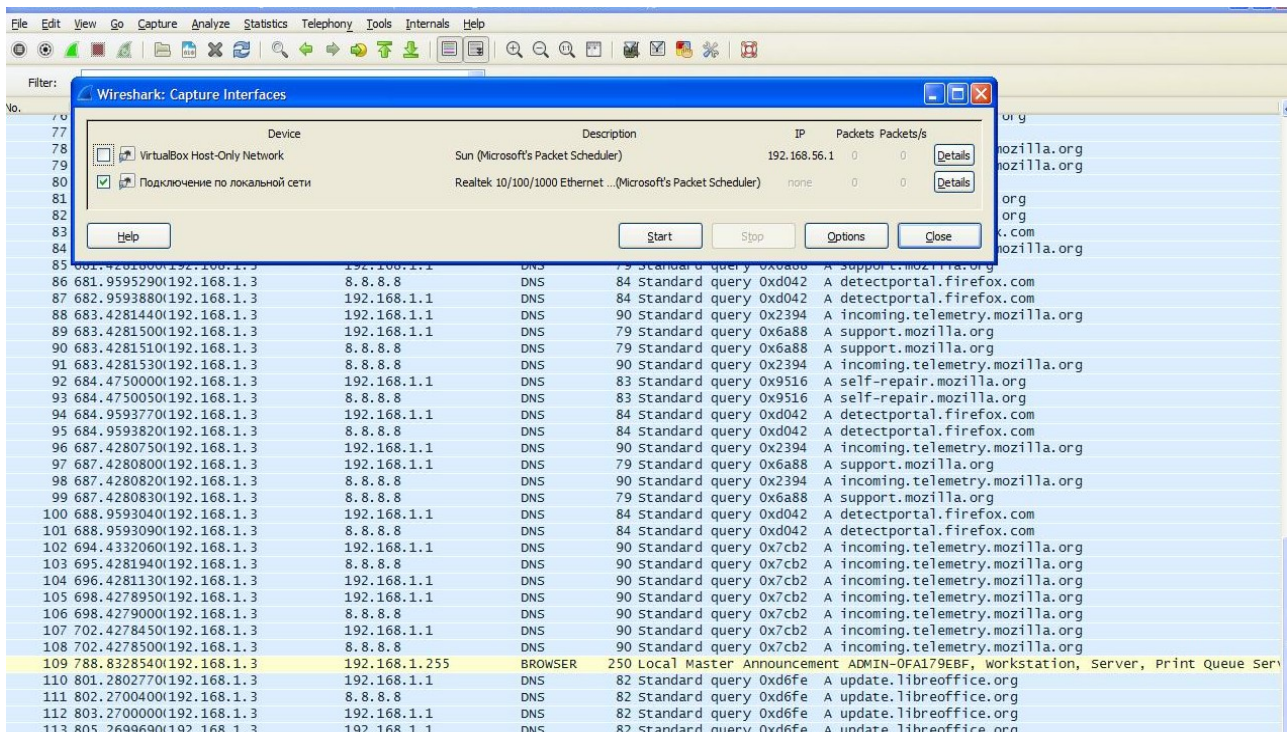


Рис. 2. Начало процесса по захвату пакетов.

Захват тестовых пакетов

После начала процесса захвата пакетов знакомимся с функционалом программы путем отправки тестовых пакетов на данный IP-адрес. Пример выполнения данной операции представлен на рис. 3.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.1	192.168.1.2	ICMP	74	Echo (ping) request id=0x0200, seq=10496/41, ttl=128 (reply in 2)
2	0.000033000	192.168.1.2	192.168.1.1	ICMP	74	Echo (ping) reply id=0x0200, seq=10496/41, ttl=128 (request in 1)
3	0.988474000	192.168.1.1	192.168.1.2	ICMP	74	Echo (ping) request id=0x0200, seq=10752/42, ttl=128
4	0.988506000	192.168.1.2	192.168.1.1	ICMP	74	Echo (ping) reply id=0x0200, seq=10752/42, ttl=128 (request in 3)
5	1.988436000	192.168.1.1	192.168.1.2	ICMP	74	Echo (ping) request id=0x0200, seq=11008/43, ttl=128 (reply in 6)
6	1.988468000	192.168.1.2	192.168.1.1	ICMP	74	Echo (ping) reply id=0x0200, seq=11008/43, ttl=128 (request in 5)
7	2.988397000	192.168.1.1	192.168.1.2	ICMP	74	Echo (ping) request id=0x0200, seq=11264/44, ttl=128 (reply in 8)
8	2.988430000	192.168.1.2	192.168.1.1	ICMP	74	Echo (ping) reply id=0x0200, seq=11264/44, ttl=128 (request in 7)
9	3.988379000	192.168.1.1	192.168.1.2	ICMP	74	Echo (ping) request id=0x0200, seq=11520/45, ttl=128
10	3.988411000	192.168.1.2	192.168.1.1	ICMP	74	Echo (ping) reply id=0x0200, seq=11520/45, ttl=128 (request in 9)
11	4.988352000	192.168.1.1	192.168.1.2	ICMP	74	Echo (ping) request id=0x0200, seq=11776/46, ttl=128 (reply in 12)
12	4.988401000	192.168.1.2	192.168.1.1	ICMP	74	Echo (ping) reply id=0x0200, seq=11776/46, ttl=128 (request in 11)
13	5.988302000	192.168.1.1	192.168.1.2	ICMP	74	Echo (ping) request id=0x0200, seq=12032/47, ttl=128 (reply in 14)
14	5.988334000	192.168.1.2	192.168.1.1	ICMP	74	Echo (ping) reply id=0x0200, seq=12032/47, ttl=128 (request in 13)
15	6.988285000	192.168.1.1	192.168.1.2	ICMP	74	Echo (ping) request id=0x0200, seq=12288/48, ttl=128
16	6.988318000	192.168.1.2	192.168.1.1	ICMP	74	Echo (ping) reply id=0x0200, seq=12288/48, ttl=128 (request in 15)
17	7.988246000	192.168.1.1	192.168.1.2	ICMP	74	Echo (ping) request id=0x0200, seq=12544/49, ttl=128 (reply in 18)
18	7.988277000	192.168.1.2	192.168.1.1	ICMP	74	Echo (ping) reply id=0x0200, seq=12544/49, ttl=128 (request in 17)
19	8.988229000	192.168.1.1	192.168.1.2	ICMP	74	Echo (ping) request id=0x0200, seq=12800/50, ttl=128 (reply in 20)
20	8.988263000	192.168.1.2	192.168.1.1	ICMP	74	Echo (ping) reply id=0x0200, seq=12800/50, ttl=128 (request in 19)
21	9.988190000	192.168.1.1	192.168.1.2	ICMP	74	Echo (ping) request id=0x0200, seq=13056/51, ttl=128
22	9.988225000	192.168.1.2	192.168.1.1	ICMP	74	Echo (ping) reply id=0x0200, seq=13056/51, ttl=128 (request in 21)
23	10.988171000	192.168.1.1	192.168.1.2	ICMP	74	Echo (ping) request id=0x0200, seq=13312/52, ttl=128 (reply in 24)
24	10.988206000	192.168.1.2	192.168.1.1	ICMP	74	Echo (ping) reply id=0x0200, seq=13312/52, ttl=128 (request in 23)
25	11.988154000	192.168.1.1	192.168.1.2	ICMP	74	Echo (ping) request id=0x0200, seq=13568/53, ttl=128 (reply in 26)
26	11.988188000	192.168.1.2	192.168.1.1	ICMP	74	Echo (ping) reply id=0x0200, seq=13568/53, ttl=128 (request in 25)
27	12.988115000	192.168.1.1	192.168.1.2	ICMP	74	Echo (ping) request id=0x0200, seq=13824/54, ttl=128
28	12.988147000	192.168.1.2	192.168.1.1	ICMP	74	Echo (ping) reply id=0x0200, seq=13824/54, ttl=128 (request in 27)
29	13.988077000	192.168.1.1	192.168.1.2	ICMP	74	Echo (ping) request id=0x0200, seq=14080/55, ttl=128 (reply in 30)
30	13.988110000	192.168.1.2	192.168.1.1	ICMP	74	Echo (ping) reply id=0x0200, seq=14080/55, ttl=128 (request in 29)

Рис. 3. Захват тестовых пакетов.

Было отправлено 15 тестовых пакетов, каждый из которых был захвачен.

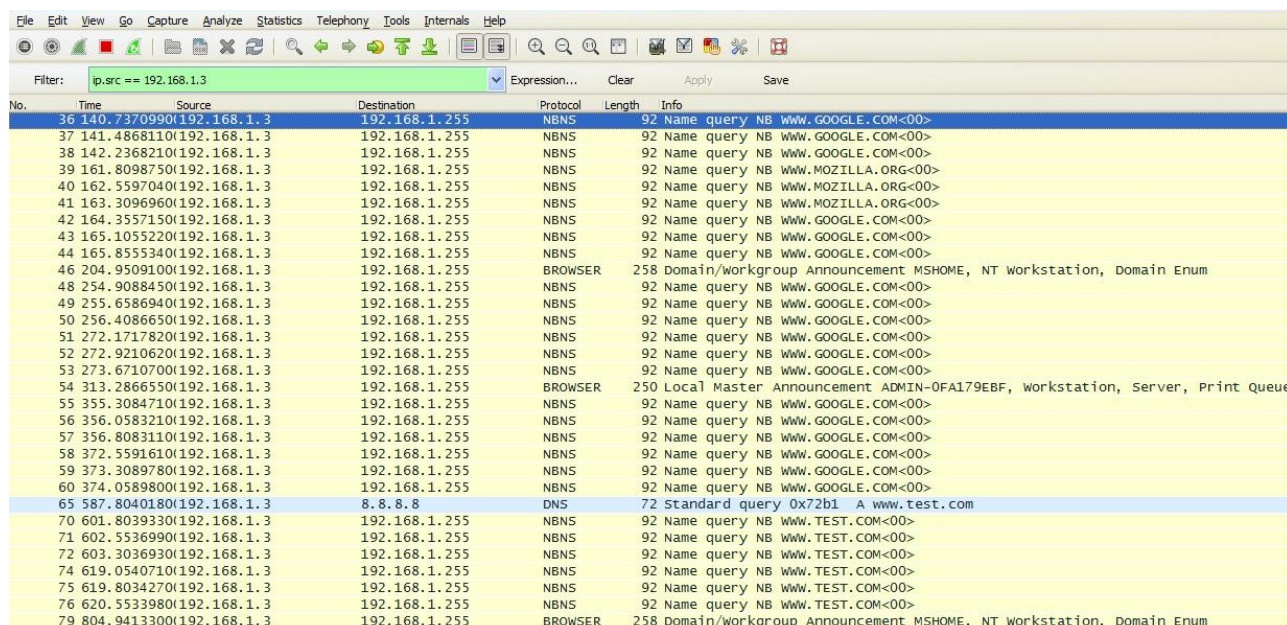
Фильтрация

Wireshark предоставляет ряд возможностей по настройке содержимого, которое будет отображаться на экране. Одна из таких возможностей — использование фильтров.

Фильтры делятся на 2 типа:

1. Фильтры захвата – фильтры, которые используются для указания того, какие данные должны записываться в журнал данных. Такие фильтры должны задаваться до начала захвата данных.
2. Фильтры отображения – фильтры, которые используются для поиска внутри журнала данных. Эти фильтры могут задаваться и изменяться в процессе захвата данных.

На рис. 4 проиллюстрирована возможность использования фильтров отображения на примере сообщений, захваченных от ПК с IP-адресом 192.168.1.3.



No.	Time	Source	Destination	Protocol	Length	Info
36	140.7370990	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.GOOGLE.COM<00>
37	141.4868110	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.GOOGLE.COM<00>
38	142.2368210	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.GOOGLE.COM<00>
39	161.8098750	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.MOZILLA.ORG<00>
40	162.5597040	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.MOZILLA.ORG<00>
41	163.3096960	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.MOZILLA.ORG<00>
42	164.3557150	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.GOOGLE.COM<00>
43	165.1055220	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.GOOGLE.COM<00>
44	165.8553400	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.GOOGLE.COM<00>
46	204.9509100	192.168.1.3	192.168.1.255	BROWSER	258	Domain/workgroup Announcement MSHOME, NT Workstation, Domain Enum
48	254.9088450	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.GOOGLE.COM<00>
49	255.6586940	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.GOOGLE.COM<00>
50	256.4086650	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.GOOGLE.COM<00>
51	272.1717820	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.GOOGLE.COM<00>
52	272.9210620	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.GOOGLE.COM<00>
53	273.6710700	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.GOOGLE.COM<00>
54	313.2866550	192.168.1.3	192.168.1.255	BROWSER	250	Local Master Announcement ADMIN-0FA179EBF, Workstation, Server, Print Queue
55	355.3084710	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.GOOGLE.COM<00>
56	356.0583210	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.GOOGLE.COM<00>
57	356.8083110	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.GOOGLE.COM<00>
58	372.5591610	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.GOOGLE.COM<00>
59	373.3089780	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.GOOGLE.COM<00>
60	374.0589800	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.GOOGLE.COM<00>
65	587.8040180	192.168.1.3	8.8.8.8	DNS	72	Standard query 0x72b1 A www.test.com
70	601.8039330	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.TEST.COM<00>
71	602.5536990	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.TEST.COM<00>
72	603.3036930	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.TEST.COM<00>
74	619.0540710	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.TEST.COM<00>
75	619.8034270	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.TEST.COM<00>
76	620.5533980	192.168.1.3	192.168.1.255	NBNS	92	Name query NB www.TEST.COM<00>
79	804.9413300	192.168.1.3	192.168.1.255	BROWSER	258	Domain/workgroup Announcement MSHOME, NT Workstation, Domain Enum

Рис. 4. Пример использования одного из фильтров, для отображения пакетов.

Вывод статистической информации

Данная программа обладает большим набором функций для вывода статистических данных о захваченных пакетах. Таким образом, полученные данные можно вывести, например, в табличном виде или же в графическом, для более наглядного представления результатов о выполненных захватах пакетов

и сборки кадров. Пример такого графика приведен на рис. 5. На этом графике отображается информация о передаче пакетов в единицу времени.

Чтобы вывести такой график достаточно выбрать пункт IO Graphs в меню Statistics.

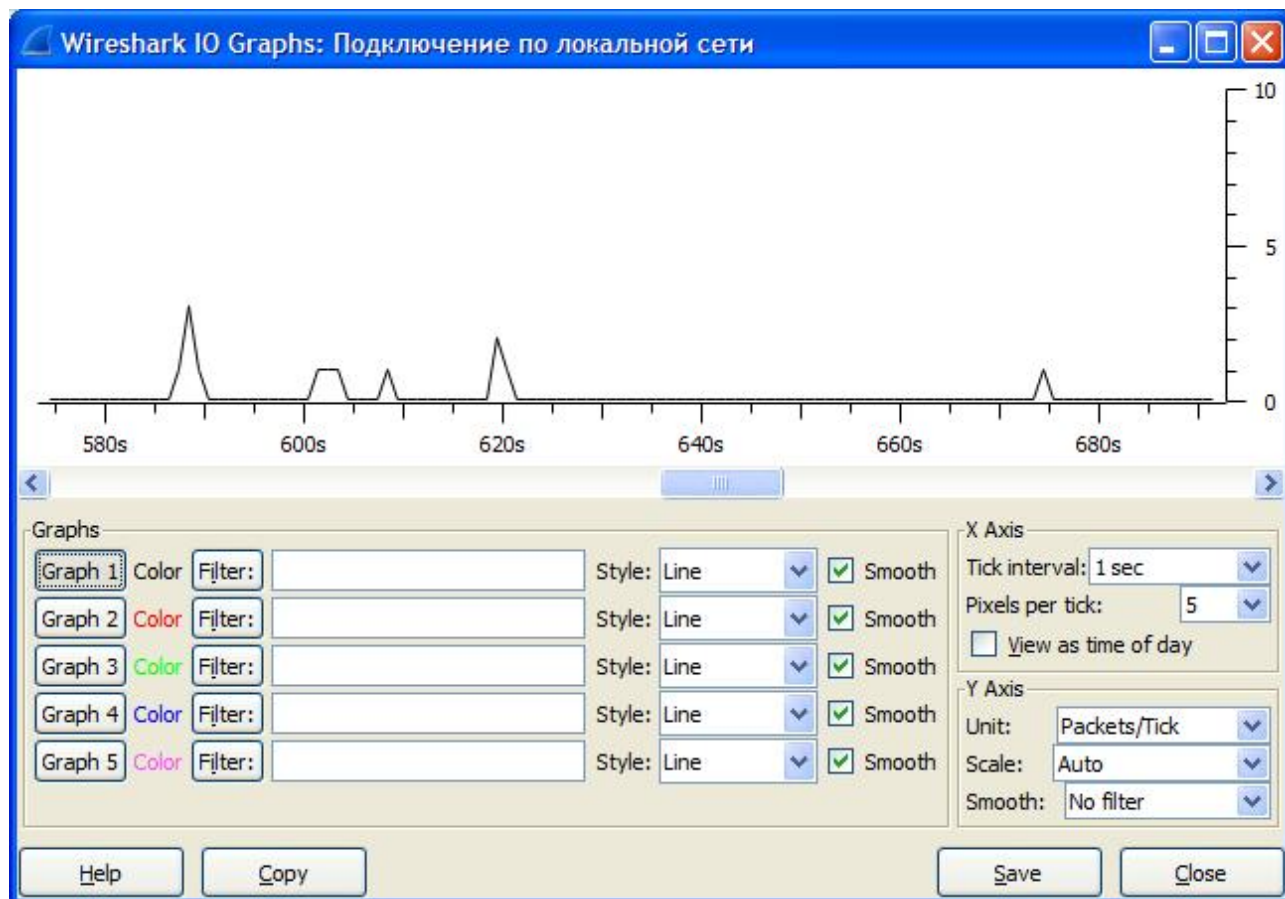


Рис. 5. Статистикой захваченных пакетов (графическое отображение).

Просмотр журнала данных

Окно Wireshark включает в себя 3 области для просмотра журнала данных с различными уровнями детализации.

В верхнем окне содержится список захваченных пакетов с краткой информацией по каждому из них. Фильтрация отображения, допустим, сводится к тому, чтобы отфильтровать пакеты именно в этом окне.

В среднем окне находится дерево протоколов для выбранного отдельного пакета. Для повышения уровня детализации выбранного протокола ветви дерева могут быть раскрыты. Самое нижнее окно содержит дамп пакета в шестнадцатеричном или же текстовом представлении.

С помощью среднего окна можно без труда определить MAC и IP-адреса отправителя и получателя пакетов, что представлено на рис. 6.

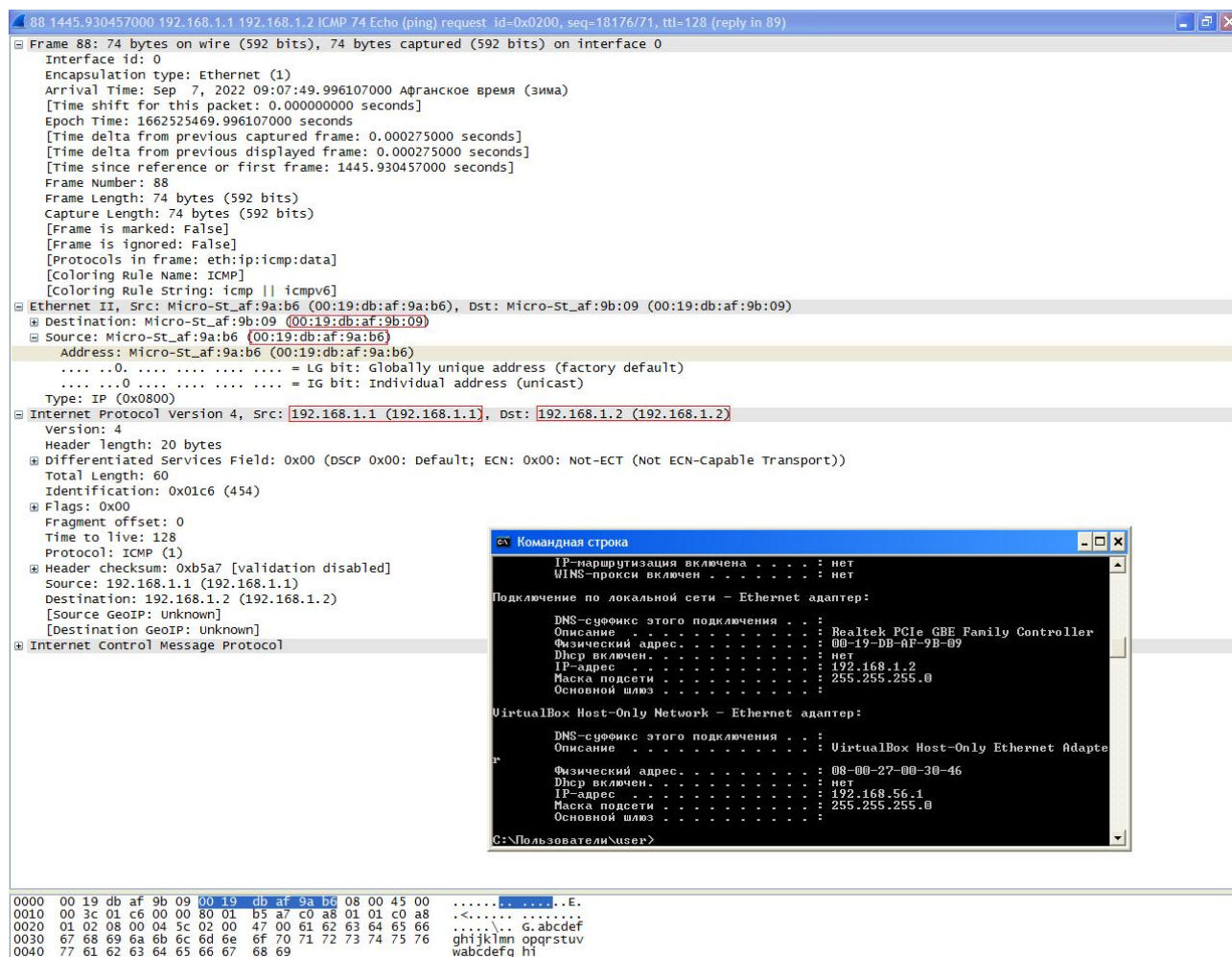


Рис. 6. Определение MAC и IP-адресов отправителя и получателя пакетов.

Захват текстовых сообщений

Также Wireshark позволяет захватывать сообщения от других IP-адресов, которые проходят по маршруту, который включает в себя текущий IP-адрес. Такая возможность продемонстрирована на рис. 7. В данном примере было передано тестовое сообщение (выделено синим).

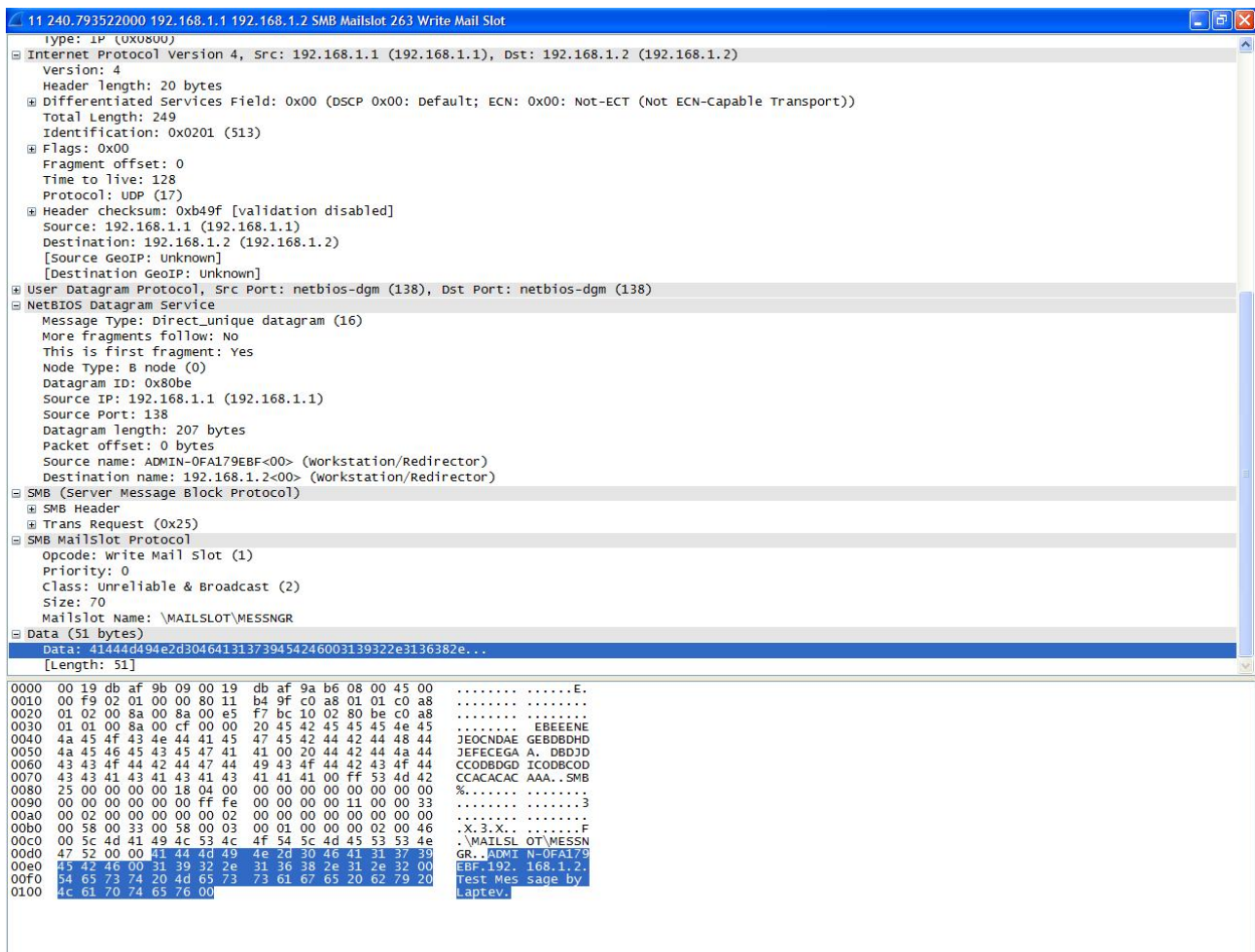


Рис. 7. Перехват тестового сообщения, проходящего через данный IP-адрес.

Работа с HTTP-протоколом

Все предыдущие примеры представляли собой работу с пакетами, захваченными в локальной сети. Но в Wireshark также есть возможность для работы с внешней сетью. Ниже представлен пример захвата пакетов по беспроводной сети, с использованием фильтра захвата, который, в данном примере, был настроен на захват данных, которые используют протокол HTTP.

На рис. 8 приведен пример для фильтра захвата, с помощью которого можно захватывать информацию только по HTTP-протоколу.

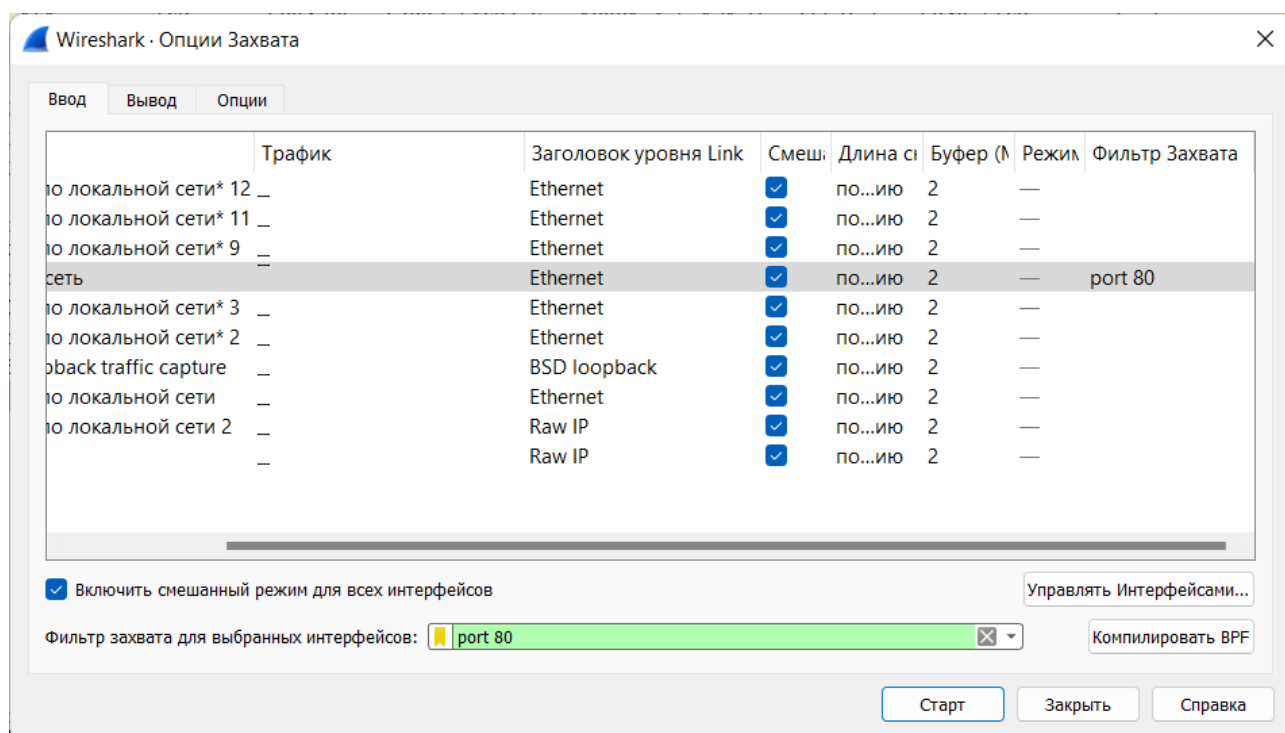


Рис. 8. Фильтр захвата – используется для захвата пакетов по внешней сети.

На рис. 9 представлена информация об одном из захваченных пакетов. В дереве протоколов отображена вся информация о пакете, касающаяся HTTP-протокола.

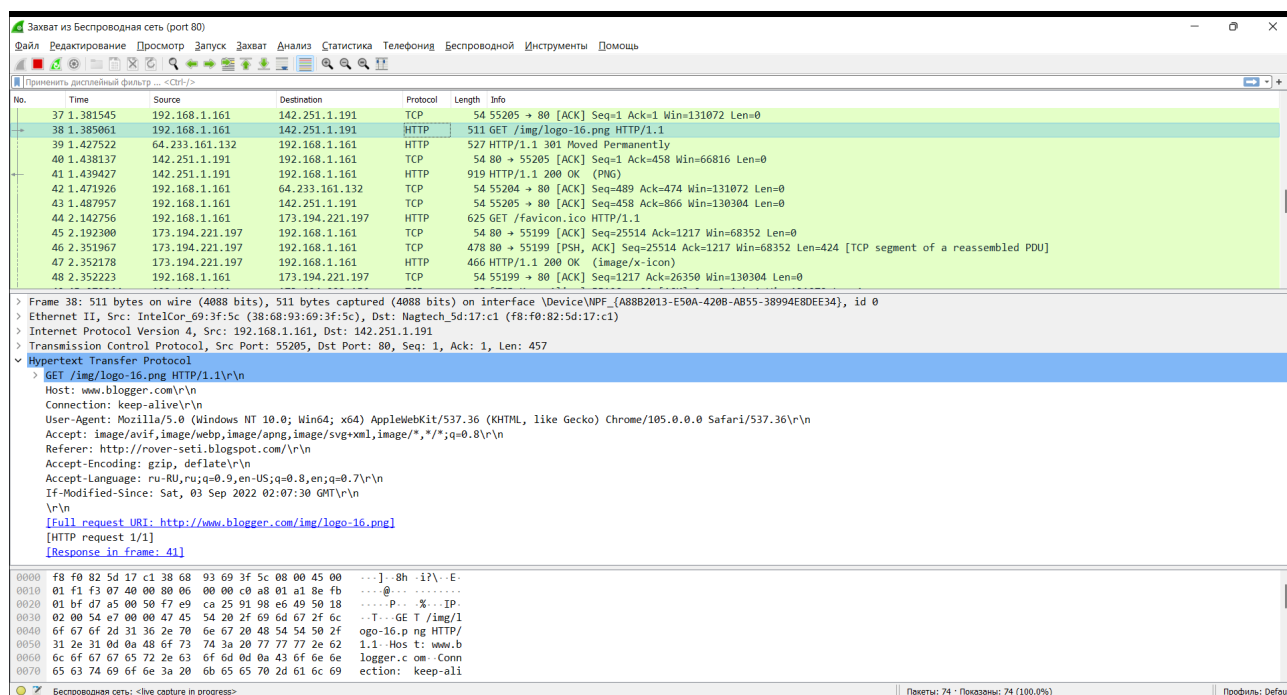


Рис. 9. Ветвь HTTP-протокола для захваченного пакета.

Помимо вышеописанного, Wireshark позволяет пользователю сохранять различные файлы данных (например, изображения, CSS и многое другое) из

просмотренных ранее страниц в Интернете. Подобная возможность проиллюстрирована на рис. 10.

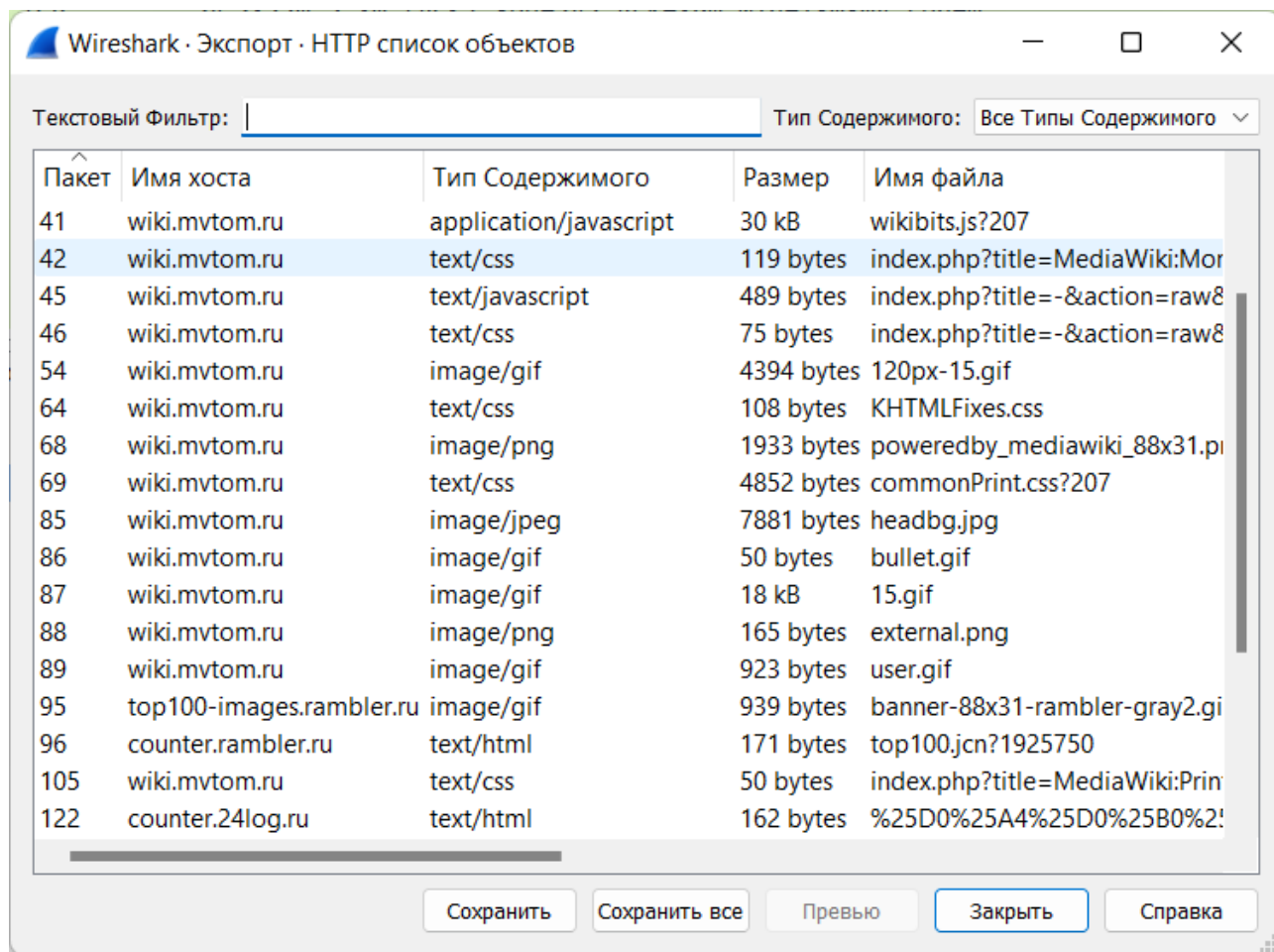


Рис. 10. Список HTTP объектов, доступных для экспорта.

3. Вывод

В результате выполнения лабораторной работы было проведено знакомство с программным обеспечением для перехвата и анализа сетевых пакетов Wireshark. Получены соответствующие навыки по захвату тестовых пакетов и последующему их анализу при помощи встроенных средств Wireshark в том числе, на наличие неполадок сети.