## Лабораторная работа

**Тема:** Создание сетевого приложения на основе TCP сокетов и перехват сетевого трафика

# Теоретическая часть

Сокет — конечная точка связи двустороннего канала между 2 компьютерами.

Если мы соединим 2 сокета, то получим канал, через который можно передавать данные в обе стороны. Одна сторона канала называется сервером, другая — клиентом.

Для передачи/приема данных нужно открыть канал. В конце всех операций — закрыть.

#### Типы сокетов

Существует 2 вида сокетов: потоковые, дейтаграммные.

Потоковый сокет — это сокет, который состоит из потока байтов, который может быть двунапрямленным (в обе стороны). Он берет на себя данных ответственность доставке И исправлении ошибок. всю o Особенностью есть возможность передачи больших объемов Использует протокол TCP (Transmission Control Protocol), именно который обеспечивает поступление данных на другую сторону нужной последовательности и без ошибок.

Дейтаграммный сокет — в отличие от потокового, имеет ограничения по размеру. Реализован через протокол UDP (User Datagram Protocol), который не отвечает за приход в конечную точку всех данных. Одним из плюсов — не нужно создавать соединения между 2 сторонами. Это очень важно, когда затраты времени недопустимы.

# Более подробная информация

http://msdn.microsoft.com/ru-ru/library/system.net.sockets.socket(v=vs.110).aspx

# Практическая часть

Для примера создадим приложение-чат. Приложение будет состоять из двух частей: серверное приложение и клиентское приложение.

## Реализация серверной части

Создадим новый проект Приложение Windows Forms на языке С#.

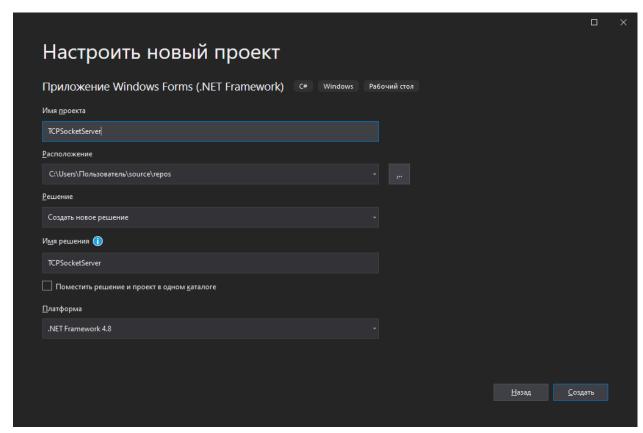


Рис.1. Создание проекта приложения

Добавим в проект пространства имен, методы которых будут использованы в проекте:

```
// Добавляем пространства имен для работы сокетов
using System.Net.Sockets;
using System.Net;
using System.Threading;
```

### Объявим необходимые глобальные переменные:

```
// Раздел глобальных переменных private static Socket Server; // Создаем объект сокета-сервера private static Socket Handler; // Создаем объект вспомогательного сокета private static IPHostEntry ipHost; // Класс для сведений об адресе веб-узла private static IPAddress ipAddr; // Предоставляет IP-адрес private static IPEndPoint ipEndPoint; // Локальная конечная точка private static Thread socketThread; // Создаем поток для поддержки потока private static Thread WaitingForMessage; // Создаем поток для приёма сообщений
```

Реализуем функцию startSocket для запуска сокета с целью прослушивания и ожидания подключения клиента:

```
// Функция запуска сокета
private void startSocket()
{
    // IP-aдрес сервера, для подключения
    string HostName = "0.0.0.0";
    // Порт подключения
    string Port = tbPort.Text;
```

```
// Разрешает DNS-имя узла или IP-адрес в экземпляр IPHostEntry.
ipHost = Dns.Resolve(HostName);
// Получаем из списка адресов первый (адресов может быть много)
ipAddr = ipHost.AddressList[0];
// Создаем конечную локальную точку подключения на каком-то порту
ipEndPoint = new IPEndPoint(ipAddr, int.Parse(Port));
try
{
    // Создаем сокет сервера на текущей машине
    Server = new Socket(AddressFamily.InterNetwork,
    SocketType.Stream, ProtocolType.Tcp);
catch (SocketException error)
    // 10061 — порт подключения занят/закрыт
    if (error.ErrorCode == 10061)
    {
        MessageBox.Show("Порт подключения закрыт!");
        Application.Exit();
    }
}
// Ждем подключений
try
{
    // Связываем удаленную точку с сокетом
   Server.Bind(ipEndPoint);
    // Не более 10 подключения на сокет
    Server.Listen(10);
    // Начинаем "прослушивать" подключения
   while (true)
    {
        Handler = Server.Accept();
        if (Handler.Connected)
            // Позеленим кнопочку для красоты, чтобы пользователь знал, что
            соединение установлено
            bConnect.Invoke(new Action(() => bConnect.Text = "Клиент
            подключен"));
            bConnect.Invoke(new Action(() => bConnect.BackColor =
            Color.Green));
            // Создаем новый поток, указываем на ф-цию получения сообщений в
            классе Worker
            WaitingForMessage = new System.Threading.Thread(new
            System.Threading.ParameterizedThreadStart(GetMessages));
            // Запускаем, в параметрах передаем листбокс (история чата)
            WaitingForMessage.Start(new Object[] { lbHistory });
        break;
    }
catch (Exception e)
{
    throw new Exception("Проблемы с подключением");
}
```

}

Запуск прослушивающего сокета производится по нажатию кнопки в отдельном потоке. Это необходимо для сохранения функциональности формы приложения:

```
socketThread = new Thread(new ThreadStart(startSocket));
socketThread.IsBackground = true;
socketThread.Start();
bConnect.Enabled = false;
bConnect.Text = "Ожидание подключения";
bConnect.BackColor = Color.Yellow;
```

В рамках метода startSocket при подключении клиента происходит запуск потока WaitingForMessage для организации принятия сообщений от клиента. В рамках потока исполняется метод GetMessages:

```
// Ф-ция, работающая в новом потоке: получение новых сообщений —
        public static void GetMessages(Object obj)
            // Получаем объект истории чата (лист бокс)
            Object[] Temp = (Object[])obj;
            System.Windows.Forms.ListBox ChatListBox =
(System.Windows.Forms.ListBox)Temp[0];
            // В бесконечном цикле получаем сообщения
            while (true)
                // Ставим паузу, чтобы на время освобождать порт для отправки сообщений
                Thread.Sleep(50);
                    try
                    {
                        // Получаем сообщение от клиента
                        string Message = GetDataFromClient();
                    // Добавляем в историю + текущее время
                    ChatListBox.Invoke(new Action(() =>
ChatListBox.Items.Add(DateTime.Now.ToShortTimeString() + " Client: " + Message)));
                    catch { }
            }
       }
```

В свою очередь метод GetMessages использует метод GetDataFromClient для принятия и обработки сообщений от клиента:

```
// Получение информации от клиента
public static string GetDataFromClient()
{
    string GetInformation = "";

    byte[] GetBytes = new byte[1024];
    int BytesRec = Handler.Receive(GetBytes);

    GetInformation += Encoding.Unicode.GetString(GetBytes, 0, BytesRec);
    return GetInformation;
}
```

Отправка сообщений от сервера к клиенту производится с использованием кода:

```
// Посылаем клиенту новое сообщение
SendDataToClient(tbMessage.Text);
// Добавляем в историю свое же сообщение + ник + время написания
lbHistory.Items.Add(DateTime.Now.ToShortTimeString() + " Server: " +
tbMessage.Text.ToString());
// Автопрокрутка истории чата
lbHistory.TopIndex = lbHistory.Items.Count - 1;
// Убираем текст из поля ввода
tbMessage.Text = "";
```

При этом используется метод SendDataToClient:

```
// Отправка информации на клиент
public static void SendDataToClient(string Data)
{
    byte[] SendMsg = Encoding.Unicode.GetBytes(Data);
    Handler.Send(SendMsg);
}
```

Внешний вид приложения приведен на рисунке 2.

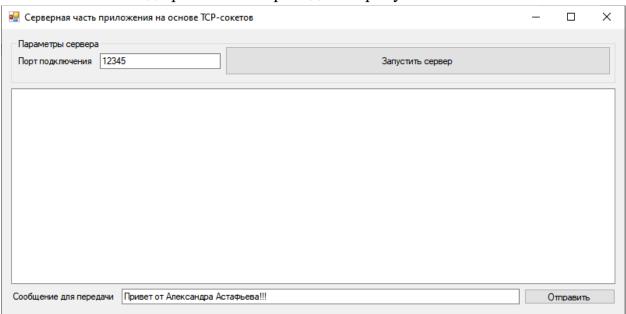


Рис. 2. Внешний вид приложения

#### Реализация клиентской части

Программная реализация клиентской части очень схожа с серверной. На первом этапе подключаем пространства имен:

```
// Добавляем пространства имен для работы сокетов
using System.Net.Sockets;
using System.Net;
using System.Threading;
```

Объявим необходимые глобальные переменные:

```
private static Socket Client; // Создаем объект сокета-сервера

private static IPHostEntry ipHost; // Класс для сведений об адресе веб-узла
private static IPAddress ipAddr; // Предоставляет IP-адрес
private static IPEndPoint ipEndPoint; // Локальная конечная точка

private static Thread socketThread; // Создаем поток для поддержки потока
private static Thread WaitingForMessage; // Создаем поток для приёма сообщений
```

#### Вид метода startSocket выглядит следующим образом:

```
private void startSocket()
 {
     // IP-адрес сервера, для подключения
     string HostName = tbAddress.Text;
     // Порт подключения
     string Port = tbPort.Text;
     // Разрешает DNS-имя узла или IP-адрес в экземпляр IPHostEntry.
     ipHost = Dns.Resolve(HostName);
     // Получаем из списка адресов первый (адресов может быть много)
     ipAddr = ipHost.AddressList[0];
     // Создаем конечную локальную точку подключения на каком-то порту
     ipEndPoint = new IPEndPoint(ipAddr, int.Parse(Port));
     try
     {
         // Создаем сокет на текущей машине
         Client = new Socket(AddressFamily.InterNetwork,
         SocketType.Stream, ProtocolType.Tcp);
         while (true)
         {
             // Пытаемся подключиться к удаленной точке
             Client.Connect(ipEndPoint);
             if (Client.Connected) // Если клиент подключился
             {
                 // Позеленим кнопочку для красоты, чтобы пользователь знал, что
                  соединение установлено
                 bConnect.Invoke(new Action(() => bConnect.Text = "Подключение
                  установлено"));
                 bConnect.Invoke(new Action(() => bConnect.BackColor =
                  Color.Green));
                 // Создаем новый поток, указываем на ф-цию получения сообщений в
                  классе Worker
                 WaitingForMessage = new System.Threading.Thread(new
                  System.Threading.ParameterizedThreadStart(GetMessages));
                 // Запускаем, в параметрах передаем листбокс (история чата)
                 WaitingForMessage.Start(new Object[] { lbHistory });
             break;
         }
     catch (SocketException error)
         // 10061 — порт подключения занят/закрыт
         if (error.ErrorCode == 10061)
         {
             MessageBox.Show("Порт подключения закрыт!");
             Application.Exit();
         }
     }
 }
```

# Метод GetMessages:

```
// Ф-ция, работающая в новом потоке: получение новых сообщенй —
       public static void GetMessages(Object obj)
            // Получаем объект истории чата (лист бокс)
            Object[] Temp = (Object[])obj;
            System.Windows.Forms.ListBox ChatListBox =
(System.Windows.Forms.ListBox)Temp[0];
            // В бесконечном цикле получаем сообщения
            while (true)
            {
                // Ставим паузу, чтобы на время освобождать порт для отправки сообщений
                Thread.Sleep(50);
                   try
                    {
                        string Message = GetDataFromServer();
                        ChatListBox.Invoke(new Action(() =>
ChatListBox.Items.Add(DateTime.Now.ToShortTimeString() + " Server: " + Message)));
                    catch { }
            }
       }
      Метод GetDataFromServer:
       // Получение данных от сервера
       public static string GetDataFromServer()
            string GetInformation = "";
            // Создаем пустое "хранилище" байтов, куда будем получать информацию
            byte[] GetBytes = new byte[1024];
            int BytesRec = Client.Receive(GetBytes);
            // Переводим из массива битов в строку
            GetInformation += Encoding.Unicode.GetString(GetBytes, 0, BytesRec);
            return GetInformation;
      }
      Метод SendDataToServer:
      // Отправка информации на сервер
       public static void SendDataToServer(string Data)
            // Используем unicode, чтобы не было проблем с кодировкой, при приеме
информации
            byte[] SendMsg = Encoding.Unicode.GetBytes(Data);
            Client.Send(SendMsg);
       }
      Код запуска сокета:
      socketThread = new Thread(new ThreadStart(startSocket));
      socketThread.IsBackground = true;
      socketThread.Start();
      bConnect.Enabled = false;
```

# Код отправки сообщения на сервер:

## Внешний вид клиентской части приведен на рисунке 3.

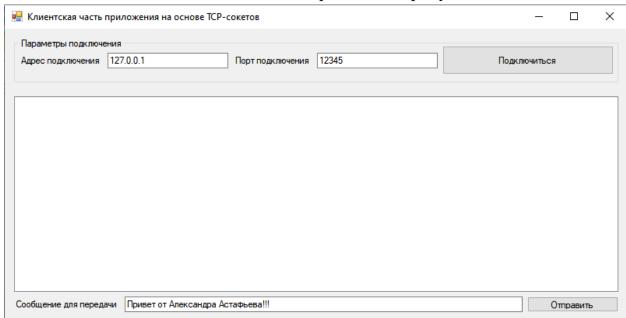


Рис. 3. Внешний вид клиентской формы

Процесс работы приложения:

- 1. Запускаем приложение-сервер.
- 2. Запускаем приложение-клиент.
- 3. На серверной части запускаем сокет с использованием кнопки «Запустить сервер». При этом необходимо указать порт подключения.
- 4. На клиентской части необходимо запустить сокет путем нажатия кнопки «Подключиться». При этом необходимо указать адрес подключения и порт подключения.
- 5. После успешной связи кнопки приложений окрасятся в зеленый цвет.
- 6. Для дальнейшей работы необходимо передавать сообщения с использованием кнопки «Отправить».

Пример работы клиент-серверного приложения приведен на рисунке 4.

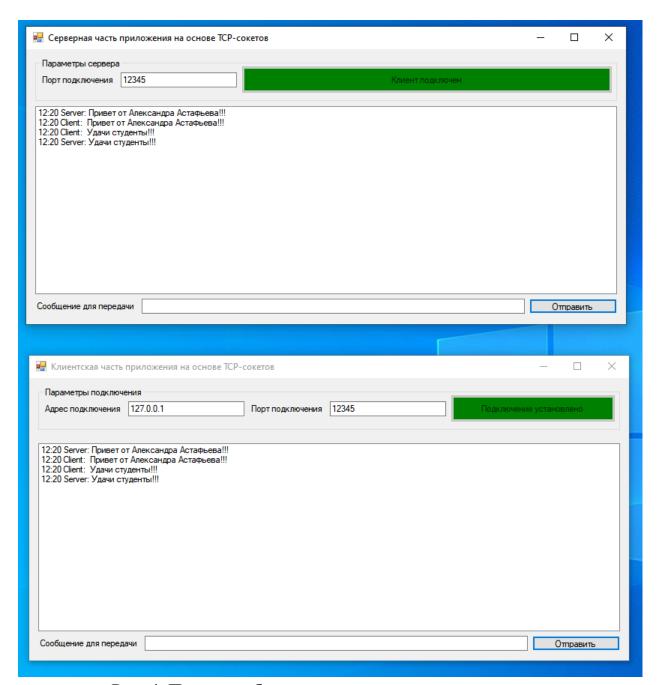


Рис. 4. Пример работы клиент-серверного приложения

# Перехват сетевого трафика

WireShark - программа-анализатор трафика для компьютерных сетей Ethernet и некоторых других.

Для начала получения сетевого трафика будем использовать программу WireShark. Интерфейс программы приведен на рисунке 5.

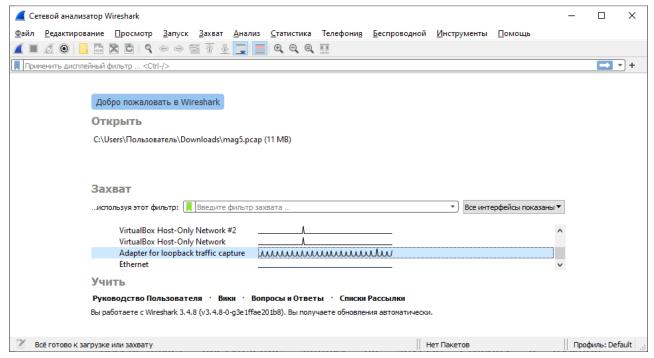


Рис.5. Интерфейс программы WireShark

В части интерфейса «Захват» в программе WireShark выводится список сетевых интерфейсов, с помощью которых возможно произведение захвата сетевого трафика. Список интерфейсов должен совпадать со списком сетевых подключений операционной системы и специализированными интерфейсами. Список сетевых подключений ОС приведен на рисунке 6.

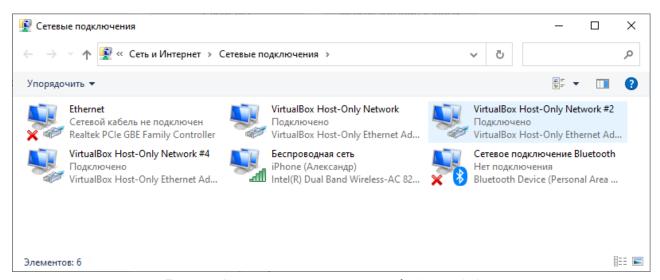


Рис. 6. Список сетевых интерфейсов ОС

#### Захват

используя этот фильтр: 📙 Введите фильтр	захвата
VirtualBox Host-Only Network #4	
Подключение по локальной сети* 10	
Подключение по локальной сети* 9	
Подключение по локальной сети* 2	
Подключение по локальной сети* 1	
Беспроводная сеть	
Сетевое подключение Bluetooth	
Подключение по локальной сети* 8	
VirtualBox Host-Only Network #2	A
VirtualBox Host-Only Network	Λ
Adapter for loopback traffic capture	
Ethernet	

Рис.7. Список интерфейсов захвата

Помимо интерфейсов ОС в списке присутствует такой интерфейс, как «Аdapter for loopback traffic capture». Если вы пытаетесь перехватить трафик с компьютера на себя, этот трафик не будет передаваться по реальному сетевому интерфейсу, даже если он отправляется по адресу на одном из сетевых адаптеров компьютера. Это означает, что вы не увидите его, если попытаетесь захватить, например, интерфейсное устройство адаптера, которому назначен адрес назначения. Вы увидите его только в том случае, если сделаете захват трафика на "Adapter for loopback traffic capture", если такой интерфейс существует и на нем можно осуществить захват. Проще говоря, весь трафик, направленный на адреса 127.0.0.1 и localhost будут видны именно на этом интерфейсе.

Для того, чтобы начать захват сетевого трафика необходимо выбрать интерфейс и кликнуть на него два раза. После этого программа переключится в режим захвата трафика. Пример начала захвата трафика приведен на рисунке 8.

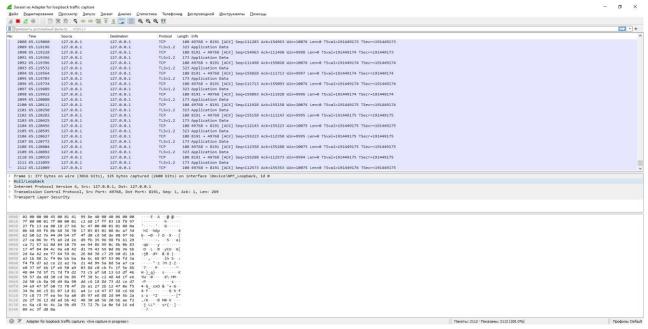


Рис. 8. Пример начала захвата трафика

Окно программы разделяется на 3 части. Верхняя часть визуализируется список захваченных пакетов. Средняя часть выводит выбранный захваченный пакет согласно его структуре. Нижняя часть показывает состав передаваемых данных.

Для первого эксперимента запустим приложение, разработанное на предыдущем занятии два раза: в режиме клиента и сервера. Создадим соединение по адресу 127.0.0.1. Таким образом, весь трафик, передаваемый между этими приложениями, будет захвачен на интерфейсе «Adapter for loopback traffic capture». Пример захвата трафика приведен на рисунке 9.

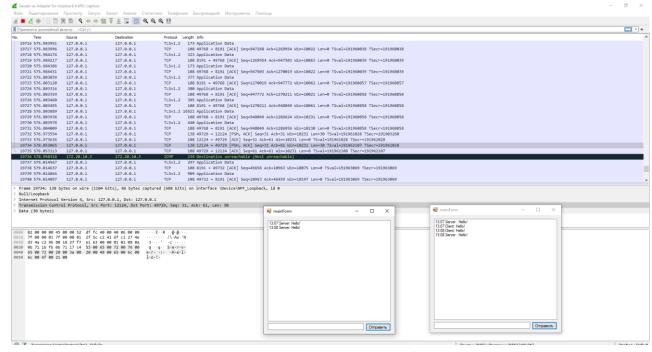


Рис.9. Пример захвата трафика

Таким образом, после передачи сообщений с сервера на клиент и в

обратную сторону в программе WireShark появляются соответствующие записи, как это показано на рисунке 10.

	19735 578.053113	127.0.0.1	127.0.0.1	TCP	108 49729 → 12124 [ACK] Seq=61 Ack=61 Win=10231 Len=0 TSval=191962108 TSecr=191962107
	19734 578.053065	127.0.0.1	127.0.0.1	TCP	138 12124 → 49729 [PSH, ACK] Seq=31 Ack=61 Win=10231 Len=30 TSval=191962107 TSecr=191961028
	19733 576.973636	127.0.0.1	127.0.0.1	TCP	108 12124 → 49729 [ACK] Seq=31 Ack=61 Win=10231 Len=0 TSval=191961028 TSecr=191961028
	19732 576.973554	127.0.0.1	127.0.0.1	TCP	138 49729 → 12124 [PSH, ACK] Seq=31 Ack=31 Win=10231 Len=30 TSval=191961028 TSecr=191903260
- 1					[]]

Рис.10. Пакеты трафика

Верхние 2 строки соответствуют передаче сообщения Hello! От клиента к серверу. Состав пакета приведен на рисунке 15.

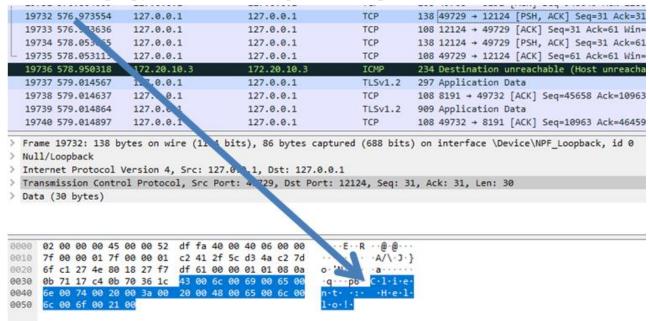


Рис. 11. Сообщение от клиента к серверу

Нижние 2 строки соответствуют сообщению Hello! От сервера к клиенту. Состав пакета приведен на рисунке 12.

```
19731 576.804009
                      127.0.0.1
                                                                          108 49768 → 8191 [ACK] Seq=948049 Ack=128
                                            127.0.0.1
                                                                 TCP 138 49729 → 12124 [PSH, ACK] Seq=31 Ack=3
TCP 108 12124 → 49729 [ACK] Seq=31 Ack=61 Win
  19732 576.973554 127.0.0.1
                                            127.0.0.1
                                            127.0.0.1
  19733 576.973636 127.0.0.1
                                                                TCP 138 [12124 → 49729 [PSH, ACK] Seq=31 Ack=61 Win
TCP 108 49729 → 12124 [ACK] Seq=31 Ack=6
  19734 578.053065 127.0.0.1
                                             127.0.0.1
  19735 578. 3113 127.0.0.1
19736 578.950. 8 172.20.10.3
                                             127.0.0.1
                                                                 ICMP 234 Destination unreachable (Host unreach
                                           172.20.10.3
                   127.0.0.1
  19737 579.014567
                                         127.0.0.1
                                                                TLSv1.2 297 Application Data
                       27.0.0.1
  19738 579.014637
                                                                  TCP
                                                                             108 8191 + 49732 [ACK] Seq=45658 Ack=1096
                                             127.0.0.1
                      12. 0.0.1
                                             127.0.0.1
   19739 579.014864
                                                                   TLSv1.2
                                                                             909 Application Data
  19740 579.014897
                      127.0.1
                                             127.0.0.1
                                                                             108 49732 → 8191 [ACK] Seq=10963 Ack=4645
                                                                  TCP
> Frame 19734: 138 bytes on wire 1104 bits), 86 bytes captured (688 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
 Internet Protocol Version 4, Src: 12, 0.0.1, Dst: 127.0.0.1
 Transmission Control Protocol, Src Port. 12124, Dst Port: 49729, Seq: 31, Ack: 61, Len: 30
> Data (30 bytes)
```

Рис. 12. Сообщение от сервера к клиенту

Так как в ходе работы приложений на базе сокетов происходит постоянный обмен информацией, то записей при захвате трафика, появляется

огромное количество. Для того, чтобы производить поиск необходимых пакетов, их необходимо проанализировать.

Исходя из того, что приложение для передачи данных было создано в рамках предыдущей работы можно описать характер передаваемой информации. Передаваемая информация включает в себя текстовую информацию. Исходя из того, что при захвате трафика были определены записи, содержащие данную информацию можно рассмотреть их более детально. На первый взгляд, записи отличаются наличием записи [PSH, ACK] (рисунок 13).

127.9 519.205588 127.0.0.1 127.0.0.1 TCP 108 12124 + 49778 [ACK] Sensal Ack=31 Min=10231 Len=0 TSval=191903260 TSccr=191903260 127.0.0.1 127.0.0.1 TCP 108 12124 + 497 [PSH, ACK] Sensal Ack=31 Min=10231 Len=0 TSval=191961028 TSccr=191961028 17472 517.654776 127.0.0.1 127.0.0.1 TCP 138 12124 + 497 [PSH, ACK] Sensal Ack=31 Min=10231 Len=30 TSval=191961028 TSccr=191961028 127.0.0.1 TCP 138 12124 + 497 [PSH, ACK] Sensal Ack=31 Min=10231 Len=30 TSval=191961028 TSccr=191961028 127.0.0.1 TCP 138 12124 + 497 [PSH, ACK] Sensal Ack=31 Min=10231 Len=30 TSval=191961028 127.0.0.1 TCP 138 12124 + 497 [PSH, ACK] Sensal Ack=31 Min=10231 Len=30 TSval=191961028 127.0.0.1 TCP 138 12124 + 497 [PSH, ACK] Sensal Ack=31 Min=10231 Len=30 TSval=191961028 127.0.0.1 TCP 138 12124 + 497 [PSH, ACK] Sensal Ack=31 Min=10231 Len=30 TSval=191961028 127.0.0.1 TCP 1320 3702 + 57164 Len=1220 127.0.0.1 TCP 1320 3702 + 57164 Len=1220 127.0.0.1 TCP 1321 3702 + 57164 Len=1220 127.0.0.1 TCP 1321 3702 + 57223 Len=1221 127.0.0.1 TCP 1321 3702 + 63223 Len=1221 127.0.0.1 TCP 1321 3702 3702 3702 3702 3702 3702 3702 3702	- (1	-	,		
17472 517.654776 127.0.0.1 127.0.0.1 TCP 138 12124 + 497 PSH, ACK] So 1 Ack=1 Win=10231 Len=30 TSval=191901709 TSecr=190490840 19734 578.053065 127.0.0.1 127.0.0.1 TCP 138 12124 + 497 PSH, ACK] So 1 Ack=1 Win=10231 Len=30 TSval=191962107 TSecr=191961028 1326 2414.675365 ::1	17479 519.205588	8 127.0.0.1	127.0.0.1	TCP	108 12124 → 497 <mark>29 [ACK] Seg=31 A</mark> ck=31 Win=10231 Len=0 TSval=191903260 TSecr=191903260
19734 578.053065 127.0.0.1 127.0.0.1 TCP 138 12124 + 497 PSH, ACK] Sc -31 Ack=61 Win=10231 Len=30 TSval=191962107 TSecr=191961028 1262 414.675365 ::1 ::1 UDP 1320 3702 + 57164 Len=1220 1381 414.831826 ::1 ::1 UDP 1320 3702 + 57164 Len=1220 13403 415.129821 ::1 ::1 UDP 1312 3702 + 57164 Len=1220 13403 415.129821 ::1 ::1 UDP 1312 3702 + 63223 Len=1221 135762 + 63223 Len=323	19733 576.973630	6 127.0.0.1	127.0.0.1	TCP	108 12124 → 497 ₽ [ACK] Seq=31 ck=61 Win=10231 Len=0 TSval=191961028 TSecr=191961028
13362 414.675365 ::1	17472 517.654770	6 127.0.0.1	127.0.0.1	TCP	138 12124 → 497 P[PSH, ACK] Se 1 Ack=1 Win=10231 Len=30 TSval=191901709 TSecr=190490840
13311 414.831826 ::1 ::1 UDP 1320 3702 + 57164 Len=1220 13403 415.129821 ::1 ::1 UDP 1321 3702 + 57164 Len=1220 85824 2706.090829 ::1 ::1 UDP 1321 3702 + 63223 Len=1221 85888 2706.472172 ::1 ::1 UDP 1321 3702 + 63223 Len=1221 85888 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221 85888 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221 85888 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221 85888 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221 85888 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221 85888 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221 85888 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221 8588 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221 8588 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221 8588 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221 8588 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221 8588 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221 858 270 270.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=218 Ack=2153 Win=26193533335 TSecr=193533334 TSecr=193533335 TSecr=193533336 TSecr=193533336 TSecr=193533336 TSecr=19353336 TSecr=1935333	19734 578.053069	5 127.0.0.1	127.0.0.1	TCP	138 12124 → 497 PSH, ACK] St =31 Ack=61 Win=10231 Len=30 TSval=191962107 TSecr=191961028
13403 415.129821 ::1 ::1 UDP 1321 3702 → 57164 Len=1220 85824 2706.090829 ::1 ::1 UDP 1321 3702 → 63223 Len=1221 85880 2706.412172 ::1 UDP 1321 3702 → 63223 Len=1221 85880 2706.870927 ::1 ::1 UDP 1321 3702 → 63223 Len=1221 85880 2706.870927 ::1 ::1 UDP 1321 3702 → 63223 Len=1221 85880 2706.870927 ::1 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 UDP 1321 3702 → 63223 Len=1221 85800 2706.870927 ::1 UDP 1321 3702 → 63223 Len=1221 2702 → 63223 Len=1221 85800 2706.870927 ::1 UDP 1321 3702 → 63	13262 414.675369	5 ::1	::1	UDP	1320 3702 → 5716
85824 2706.090829 ::1 ::1 UDP 1321 3702 + 63223 Len=1221 85875 2706.412172 ::1 ::1 UDP 1321 3702 + 63223 Len=1221 85880 2706.6870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221 68269 2149.278029 127.0.0.1 127.0.0.1 TCP 188 49255 + 8090 [ACK] Seq=1 Ack=1 Win=2619136 Len=0 TSval=193533332 TSecr=193533334 68273 2149.278029 127.0.0.1 127.0.0.1 TCP 188 49255 + 8090 [ACK] Seq=218 Ack=2153 Win=2616832 Len=0 TSval=193533335 TSecr=193533335 68273 2149.280737 127.0.0.1 127.0.0.1 TCP 188 49255 + 8090 [ACK] Seq=344 Ack=2411 Win=2616832 Len=0 TSval=193533335 TSecr=193533335 68281 2149.281769 127.0.0.1 127.0.0.1 TCP 188 49255 + 8090 [ACK] Seq=39 Ack=2716 Win=2616576 Len=0 TSval=193533336 TSecr=193533336 68282 2149.281815 127.0.0.1 127.0.0.1 TCP 188 49255 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSval=193533336 TSecr=19353336 68282 2149.281919 127.0.0.1 127.0.0.1 TCP 188 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=193533336 68282 2149.281919 127.0.0.1 127.0.0.1 TCP 188 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=19353336 68282 2149.282202 127.0.0.1 127.0.0.1 TCP 188 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=193533336 68282 149.282202 127.0.0.1 127.0.0.1 TCP 188 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=19353336 68285 2149.282202 127.0.0.1 127.0.0.1 TCP 188 49409 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=193563366 69155 2179.313465 127.0.0.1 127.0.0.1 TCP 188 49409 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193563366 TSecr=193563368 69159 2179.313462 127.0.0.1 127.0.0.1 TCP 188 49409 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193563368 TSecr=193563370 69165 2179.315783 127.0.0.1 127.0.0.1 TCP 188 49409 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193563370 TSecr=193563370 69165 2179.315783 127.0.0.1 127.0.0.1 TCP 188 49409 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193563370 TSecr=193563370	13311 414.831826	6 ::1	::1	UDP	1320 3702 → 57164 Len=1220
85875 2706.412172 ::1 ::1 UDP 1321 3702 + 63223 Len=1221  85880 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221  85880 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221  85880 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221  85880 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221  85880 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221  85880 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221  85880 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221  85880 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221  85880 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221  85880 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221  85880 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221  85880 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221  85880 2706.870927 ::1 ::1 UDP 1321 3702 + 63223 Len=1221  85880 2706.870927 ::1	13403 415.12982	1 ::1	::1	UDP	1312 3702 → 57164 Len=1220
85880 2706.870927 ::1 ::1 UDP 1321 3702 → 63223 Len=1221 68269 2149.278029 127.0.0.1 127.0.0.1 TCP 188 49255 + 8090 [ACK] Seq=1 Ack=1 Win=2619136 Len=0 TSval=193533332 TSecr=193533334 68273 2149.280737 127.0.0.1 127.0.0.1 TCP 188 49255 + 8090 [ACK] Seq=344 Ack=2411 Win=2616832 Len=0 TSval=193533335 TSecr=19353334 68277 2149.280737 127.0.0.1 127.0.0.1 TCP 188 49255 + 8090 [ACK] Seq=344 Ack=2411 Win=2616832 Len=0 TSval=193533335 TSecr=19353335 68283 2149.2818769 127.0.0.1 127.0.0.1 TCP 188 49255 + 8090 [ACK] Seq=739 Ack=2716 Win=2616576 Len=0 TSval=193533336 TSecr=19353336 68283 2149.281815 127.0.0.1 127.0.0.1 TCP 188 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=19353336 68285 2149.281919 127.0.0.1 127.0.0.1 TCP 188 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=193533336 68265 2149.281919 127.0.0.1 127.0.0.1 TCP 188 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=193533336 68267 2149.282202 127.0.0.1 127.0.0.1 TCP 184 49255 + 8090 [SYN] Seq=0 Win=65535 Len=0 MSs=65495 Ws=256 SACK_PERM=1 TSval=193533336 TSecr=0 69151 2179.311985 127.0.0.1 127.0.0.1 TCP 188 49409 + 8090 [ACK] Seq=18 Ack=2411 Win=2616320 Len=0 TSval=193563368 TSecr=193563368 69159 2179.314520 127.0.0.1 127.0.0.1 TCP 188 49409 + 8090 [ACK] Seq=218 Ack=2153 Win=2616302 Len=0 TSval=193563368 TSecr=193563369 69165 2179.315783 127.0.0.1 127.0.0.1 TCP 188 49409 + 8090 [ACK] Seq=218 Ack=2411 Win=2616320 Len=0 TSval=193563367 TSecr=193563369 69165 2179.315783 127.0.0.1 127.0.0.1 TCP 188 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616532 Len=0 TSval=193563370 TSecr=193563370 69165 2179.315783 127.0.0.1 127.0.0.1 TCP 188 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616532 Len=0 TSval=193563370 TSecr=193563370 69165 2179.315783 127.0.0.1 127.0.0.1 TCP 188 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616532 Len=0 TSval=193563370 TSecr=193563370	85824 2706.09082	29 ::1	::1	UDP	1321 3702 → 63223 Len=1221
68269 2149.278029 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=1 Ack=1 Win=2619136 Len=0 TSval=193533332 TSecr=193533332 (68273 2149.278972 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=218 Ack=2153 Win=2616832 Len=0 TSval=193533334 TSecr=193533335 (68273 2149.2881769 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=344 Ack=2411 Win=2616832 Len=0 TSval=193533335 TSecr=193533335 (68283 2149.2881769 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=739 Ack=2816 Win=2616576 Len=0 TSval=193533336 TSecr=193533336 (68283 2149.288195 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSval=193533336 TSecr=193533336 (68285 2149.288191 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=193533336 (68285 2149.288202 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=193533336 (6826 2149.288202 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=193533336 (6826 2149.27883 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=193563366 (6826 2149.27883 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193563368 TSecr=193563368 (69159 2179.313452 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=13 Ack=215 Win=2616320 Len=0 TSval=193563368 TSecr=193563368 (69159 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193563369 TSecr=193563369 (69165 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193563370 TSecr=193563370 (69165 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2840 Win=2616320 Len=0 TSval=193563370 TSecr=193563370 (69165 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193563370 TSecr=193563370 (69165 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=73	85875 2706.41217	72 ::1	::1	UDP	1321 3702 → 63223 Len=1221
68273 2149.279572 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=218 Ack=2153 Win=2617088 Len=0 TSval=193533334 TSecr=193533334 (68277 2149.280737 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=344 Ack=2411 Win=2616632 Len=0 TSval=193533335 TSecr=193553335 (68283 2149.281815 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=739 Ack=2816 Win=2616576 Len=0 TSval=193533336 TSecr=19353336 (68283 2149.281815 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSval=193533336 TSecr=193533336 (68285 2149.281919 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=19353336 (68285 2149.281919 127.0.0.1 127.0.0.1 TCP 83 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=19353336 (68285 2149.287883 127.0.0.1 127.0.0.1 TCP 83 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=19353336 (68265 2149.287883 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=1 Ack=1 Win=2616320 Len=0 TSval=193563360 TSecr=193563366 (68265 2149.287883 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=1 Ack=2411 Win=2616320 Len=0 TSval=193563360 TSecr=193563368 (68265 2179.3134620 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=1 Ack=2411 Win=2616832 Len=0 TSval=193563360 TSecr=193563360 (69163 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSval=193563370 TSecr=193563370 (69165 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSval=193563370 TSecr=193563370 (69165 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSval=193563370 TSecr=193563370 (69165 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSval=193563370 TSecr=193563370 (69165 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSval=193563370 TSecr=193563370 (69165 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK]	85880 2706.87092	27 ::1	::1	UDP	1321 3702 → 63223 Len=1221
68277 2149.280737 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=344 Ack=2411 Win=2616832 Len=0 TSval=193533335 TSecr=19353335 (68283 2149.2818769 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=739 Ack=2716 Win=2616532 Len=0 TSval=193533335 TSecr=19353336 (68283 2149.281919 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=19353336 (68285 2149.281919 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=193533336 (68285 2149.281919 127.0.0.1 TCP 108 49255 + 8090 [SN] ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=193533336 (68285 2149.281919 127.0.0.1 TCP 108 49255 + 8090 [SN] ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=193533336 (68285 2149.281919 127.0.0.1 TCP 124 49255 + 8090 [SN] Seq=0 Win=65535 Len=0 MSs=65495 Ws=256 SACK_PERM=1 TSval=193533332 TSecr=0 (6915 1279.311985 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=1 Ack=1 Win=2619136 Len=0 TSval=193563366 TSecr=193563366 (69159 2179.314620 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=218 Ack=2153 Win=2617088 Len=0 TSval=193563369 (59163 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=218 Ack=2411 Win=2616320 Len=0 TSval=193563370 TSecr=193563370 (69165 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=216 Win=2616576 Len=0 TSval=193563370 TSecr=193563370 (69165 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=216 Win=2616520 Len=0 TSval=193563370 TSecr=193563370 (69165 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616520 Len=0 TSval=193563370 TSecr=193563370 (69165 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616520 Len=0 TSval=193563370 TSecr=193563370 (69165 2179.315882 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616520 Len=0 TSval=193563370 TSecr=193563370 (69165 2179.315882 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=261652	68269 2149.27802	29 127.0.0.1	127.0.0.1	TCP	108 49255 → 8090 [ACK] Seq=1 Ack=1 Win=2619136 Len=0 TSval=193533332 TSecr=193533332
68281 2149.281769 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=739 Ack=2716 Win=2616576 Len=0 TSval=193533336 TSecr=193533336 (68283 2149.281815 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSval=193533336 TSecr=193533336 (68285 2149.281919 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=193533336 (68285 2149.288202 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=193533336 (68285 2149.288202 127.0.0.1 127.0.0.1 TCP 124 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=193533336 (68285 2149.27883 127.0.0.1 127.0.0.1 TCP 124 49255 + 8090 [SN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERN=1 TSVal=193533332 TSecr=0 (69151 2179.311985 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=1 Ack=1 Win=2619136 Len=0 TSVal=193563366 TSecr=193563368 (69159 2179.314620 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=128 Ack=2153 Win=2616382 Len=0 TSVal=193563369 TSecr=193563369 (69163 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=394 Ack=2411 Win=2616382 Len=0 TSVal=193563370 TSecr=193563370 (69165 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2847 Win=2616330 Len=0 TSVal=193563370 TSecr=193563370 (69165 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSVal=193563370 TSecr=193563370 (69167 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSVal=193563370 TSecr=193563370 (60167 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSVal=193563370 TSecr=193563370 (60167 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSVal=193563370 TSecr=193563370 (60167 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSVal=193563370 TSecr=193563370 (60167 2179.315783 127.0.0.1 127.0.0.1 TCP 108 4940	68273 2149.2795	72 127.0.0.1	127.0.0.1	TCP	
68283 2149.281815 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSval=193533336 TSecr=193533336 TSecr=19353336 TSecr=193533336 TSecr=193533336 TSecr=19353336 TSecr=19353368 TSecr=1935336 TSecr=19353368 TSecr=19353368 TSecr=19353368 TSecr=19353336 TSecr=19353338 TSecr=19353338 TSecr=19353336 TSecr=19353338 TSecr=19353338 TSecr=19353338 TSecr=19353338 TSecr=19353338 TSecr=1935338 TSecr=1935338 TSecr=1935338 TSecr=1935338 TSecr=1935338 TSecr=1935338 TSecr=1935338 TSecr=19353338 TSecr=19353338 TSecr=19353338 TSecr=1935338 TSecr=1935339 TSecr=19355339 TSecr=193553	68277 2149.2807	37 127.0.0.1	127.0.0.1	TCP	108 49255 + 8090 [ACK] Seq=344 Ack=2411 Win=2616832 Len=0 TSval=193533335 TSecr=193533335
68285 2149.281919 127.0.0.1 127.0.0.1 TCP 108 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=193533336 TSecr=19353336 TSecr=19353366 TSecr=19353336 TSecr=19353370 TSecr=19353370 TSecr=19353370 TSecr=19353370 TSecr=19353370 TSecr=19353370 TSecr=19353370 TSecr=19353370 TSecr=193533370 TSecr=19353370 TSecr=193533370 TSecr=193553370 TSecr=193533370 TSecr=193533370 TSecr=193553370 TSecr=193553370 TSecr=1935	68281 2149.2817	69 127.0.0.1	127.0.0.1	TCP	108 49255 + 8090 [ACK] Seq=739 Ack=2716 Win=2616576 Len=0 TSval=193533336 TSecr=193533336
68286 2149.282202 127.0.0.1 127.0.0.1 TCP 84 49255 + 8090 [RST, ACK] Seq=739 Ack=2847 Win=0 Len=0 68267 2149.277883 127.0.0.1 127.0.0.1 TCP 124 49255 + 8090 [SW] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1 TSval=193533332 TSecr=0 69151 2179.311985 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=1 Ack=1 Win=2619136 Len=0 TSval=193563366 TSecr=193563368 69155 2179.313415 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=218 Ack=2153 Win=26167088 Len=0 TSval=193563368 TSecr=193563368 69159 2179.314620 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=218 Ack=2411 Win=2616832 Len=0 TSval=193563369 TSecr=193563369 69163 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2716 Win=2616576 Len=0 TSval=193563370 TSecr=193563370 69165 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSval=193563370 TSecr=193563370 69167 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSval=193563370 TSecr=193563370 69167 2179.315882 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193563370 TSecr=193563370	68283 2149.28183	15 127.0.0.1	127.0.0.1	TCP	108 49255 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSval=193533336 TSecr=193533336
68267 2149.277883 127.0.0.1 127.0.0.1 TCP 124 49255 + 8090 [SYN] Seq-0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERN=1 TSVal=193533332 TSecr=0 69151 2179.313495 127.0.0.1 127.0.0.1 TCP 188 49409 + 8090 [ACK] Seq=1 Ack=1 Win=261938 Len=0 TSVal=193563365 TSecr=193563368 69159 2179.313415 127.0.0.1 127.0.0.1 TCP 188 49409 + 8090 [ACK] Seq=218 Ack=2138 Win=261088 Len=0 TSVal=193563368 TSecr=193563368 69159 2179.314620 127.0.0.1 TCP 188 49409 + 8090 [ACK] Seq=344 Ack=2411 Win=2616832 Len=0 TSVal=193563369 TSecr=193563369 69163 2179.315783 127.0.0.1 TCP 188 49409 + 8090 [ACK] Seq=739 Ack=2716 Win=2616320 Len=0 TSVal=193563370 TSecr=193563370 69165 2179.315783 127.0.0.1 127.0.0.1 TCP 188 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSVal=193563370 TSecr=193563370 69167 2179.315882 127.0.0.1 127.0.0.1 TCP 188 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSVal=193563370 TSecr=193563370	68285 2149.28193	19 127.0.0.1	127.0.0.1	TCP	108 49255 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193533336 TSecr=193533336
69151 2179.311985 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=1 Ack=1 Win=2619136 Len=0 TSval=193563366 TSecr=193563366 69155 2179.313415 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=218 Ack=2153 Win=2617088 Len=0 TSval=193563368 TSecr=193563368 69159 2179.314620 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=344 Ack=2411 Win=2616832 Len=0 TSval=193563369 TSecr=193563369 69163 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2716 Win=2616576 Len=0 TSval=193563370 TSecr=193563370 69165 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSval=193563370 TSecr=193563370 69167 2179.315882 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193563370 TSecr=193563370	68286 2149.28220	02 127.0.0.1	127.0.0.1	TCP	
69155 2179.313415 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=218 Ack=2153 Win=2617088 Len=0 TSval=193563368 TSecr=193563368 69159 2179.314620 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=344 Ack=2411 Win=2616832 Len=0 TSval=193563369 TSecr=193563369 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=379 Ack=2716 Win=2616576 Len=0 TSval=193563370 TSecr=193563370 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSval=193563370 TSecr=193563370 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSval=193563370 TSecr=193563370 TSecr=19	68267 2149.27788	83 127.0.0.1	127.0.0.1	TCP	
69159 2179.314620 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=344 Ack=2411 Win=2616832 Len=0 TSval=193563369 TSecr=193563369 69163 2179.315733 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2716 Win=2616576 Len=0 TSval=193563370 TSecr=193563370 69165 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSval=193563370 TSecr=193563370 69167 2179.315882 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193563370 TSecr=193563370	69151 2179.31198	85 127.0.0.1	127.0.0.1	TCP	108 49409 → 8090 [ACK] Seq=1 Ack=1 Win=2619136 Len=0 TSval=193563366 TSecr=193563366
69163 2179.315733 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2716 Win=2616576 Len=0 TSval=193563370 TSecr=193563370 TSecr=1935	69155 2179.3134	15 127.0.0.1	127.0.0.1	TCP	
69165 2179.315783 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2816 Win=2616320 Len=0 TSval=193563370 TSecr=193563370 TSecr=1935	69159 2179.31462	20 127.0.0.1	127.0.0.1	TCP	108 49409 → 8090 [ACK] Seq=344 Ack=2411 Win=2616832 Len=0 TSval=193563369 TSecr=193563369
69167 2179.315882 127.0.0.1 127.0.0.1 TCP 108 49409 + 8090 [ACK] Seq=739 Ack=2847 Win=2616320 Len=0 TSval=193563370 TSecr=193563370					
69168 2179.316530 127.0.0.1 127.0.0.1 TCP 84 49409 → 8090 [RST, ACK] Seq=739 Ack=2847 Win=0 Len=0					
	69168 2179.3165	30 127.0.0.1	127.0.0.1	TCP	84 49409 → 8090 [RST, ACK] Seq=739 Ack=2847 Win=0 Len=0

Рис. 13. Флаги PSH, АСК в записях трафика

Если обратиться к документации WireShark, то можно найти информацию, что

Флаг АСК означает, что машина, отправляющая пакет с АСК, подтверждает данные, которые она получила от другой машины. В ТСР, как только соединение установлено, все пакеты, отправленные любой стороной, будут содержать АСК, даже если это просто повторное подтверждение данных, которые уже были подтверждены.

Флаг PSH - это указание отправителя на то, что, если реализация TCP принимающей машины еще не предоставила полученные данные коду, считывающему данные (программе или библиотеке, используемой программой), он должен сделать это в этот момент. Процитируем RFC 793, официальную спецификацию для TCP:

Данные, которые передаются по соединению, можно рассматривать как поток октетов. Отправляющий пользователь указывает в каждом вызове ОТПРАВКИ, следует ли немедленно передавать данные этого вызова (и любых предшествующих вызовов) получающему пользователю с помощью установки флага PUSH.

Для анализа данных сетевого трафика в программе WireShark используются «Дисплейные фильтры». Работа с дисплейными фильтрами производится либо через специальный интерфейс, который находится Панель

управления – Анализ – Дисплейные фильтры (Рисунок 14). Либо применять их напрямую из строки поиска, как показано на рисунке 15.

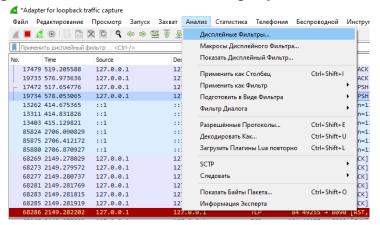


Рис. 14. Дисплейные фильтры

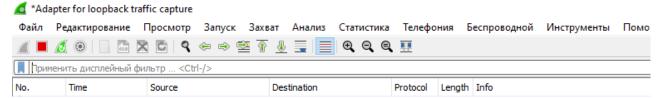


Рис. 15. Строка поиска для работы с дисплейными фильтрами Программа WireShark уже имеет дисплейные фильтры по умолчанию. Их список приведен на рисунке 16.

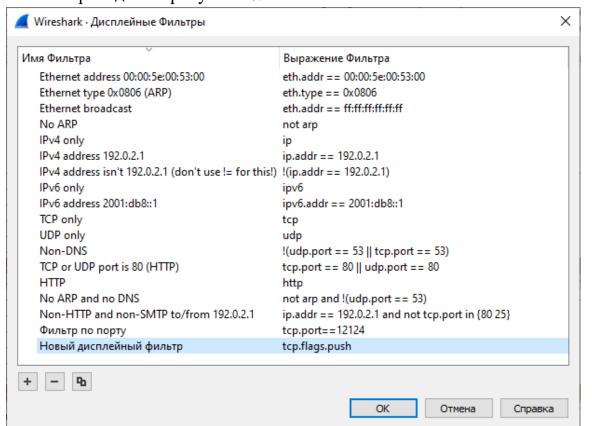


Рис. 16. Дисплейные фильтры по умолчанию

Помимо фильтров по умолчанию можно добавлять собственные фильтры, для их дальнейшего использования. Для создания фильтров можно

использовать команды, приведенные в таблице 1.

Таблица 1. Команды для создания дисплейных фильтров.

Команда	Значение	Пример использования
==	равенство	ip.dst == 193.168.3.10
!=	Не равно	udp.dst != 53
<	меньше чем	ip.ttl < 24
>	больше чем	frame.len > 10
<=	меньше или равно	frame.len <= 0x20
>=	больше или равно	tcp.analysis.bytes_in_flight >= 1000
matches	регулярные выражения	frame matches "[Pp][Aa][Ss][Ss]"
contains	содержит	dns.resp.name contains google

В нашем случае, для поиска записей трафика с сообщениями чата достаточно двух фильтров:

- 1. tcp.flags.push наличие флага PSH.
- 2. tcp.port номер порта сокета, в примере это 12345.

Таким образом, результирующий фильтр будет иметь следующий вид:

tcp.flags.push and tcp.port == 12345

Результат фильтрации по этому фильтру приведен на рисунке 17.

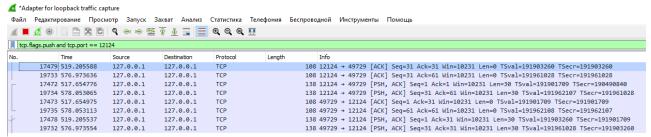


Рис. 17. Результат фильтрации

# Задание на лабораторную работу

Используя в качестве механизма сетевого взаимодействия приложение, реализованное в прошлой работе, произвести сбор и анализ сетевого трафика:

- 1. Реализовать формат передачи данных, по следующему шаблону:  $<\Phi MO\_cmy \partial e hma> \Pi e pe \partial a bae a e hube </\Phi MO\_cmy \partial e hma>.$  Например: <Asta fie v AV> Hello World! </Asta fie v AV>.
- 2. Произвести захват сетевого трафика с использованием программы WireShark по локальному интерфейсу «Adapter for loopback traffic capture».
- 3. Создать дисплейный фильтр для фильтрации сообщений, передаваемых приложениями на базе сокетов.
- 4. Оформить отчёт по проделанной работе с фиксацией основных моментов работы. В отчёте должны быть представлены сообщения и первого и второго абонентов.