##### Министерство образования и науки Российской Федерации

###### Новосибирский государственный технический университет

В.Г. КОБЫЛЯНСКИЙ

СЕТЕВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Методические указания к выполнению лабораторных работ для магистрантов, обучающихся по направлениям 01.04.02 «Прикладная математика и информатика» и 02.04.03 «Математическое обеспечение и администрирование информационных систем»

Новосибирск

2020

**СОДЕРЖАНИЕ**

[Организация лабораторного практикума 3](#_Toc56254165)

[Лабораторная работа № 1. Моделирование компьютерных сетей 4](#_Toc56254166)

[Лабораторная работа № 2. Анализ трафика компьютерной сети 17](#_Toc56254167)

[Лабораторная работа № 3. Протоколы стека TCP/IP 31](#_Toc56254168)

[Лабораторная работа № 4. Протокол ПЕРЕДАЧИ ФАЙЛОВ ftp 54](#_Toc56254169)

[Лабораторная работа № 5. Протоколы электронной почты 65](#_Toc56254170)

[Лабораторная работа № 6. Технология WEB 91](#_Toc56254171)

[Список источников 111](#_Toc56254172)

[Приложение 1 113](#_Toc56254173)

[Приложение 2 114](#_Toc56254174)

[Приложение 3 117](#_Toc56254175)

[Приложение 4 119](#_Toc56254176)

[Приложение 5 120](#_Toc56254177)

# Организация лабораторного практикума

Целью лабораторного практикума по курсу «Сетевые информационные технологии» является получение практических навыков моделирования и анализа трафика локальных сетей Ethernet, а также изучение прикладных сетевых протоколов и разработка программ, реализующих эти протоколы. Для выполнения лабораторных работ необходимо установить специальное программное обеспечение – сетевой эмулятор NetEmul и анализатор трафика WireShark. Обе программы являются свободно распространяемыми по лицензии GPL.

# Лабораторная работа № 1. Моделирование компьютерных сетей

1. Цель работы

Целью работы является изучение технологии моделирования компьютерных сетей и получение практических навыков работы с сетевыми эмуляторами.

2. Методические указания

Эмулятор – программный продукт, с помощью которого имитируется работа реальной системы. Эмулятор компьютерной сети позволяет спроектировать сеть, настроить сетевые характеристики узлов, смоделировать и продемонстрировать процессы, происходящие в построенной сети. Для этого имеет смысл кратко напомнить основные сведения о принципах построения и работы компьютерной сети.

2.1 Общие сведения

2.1.1 Сетевое оборудование

Сетевое оборудование – это набор устройств, из которых состоит компьютерная сеть (компьютеры, коммутаторы, маршрутизаторы, точки доступа и т.д.). Физическая среда передачи данных между этими устройствами может быть проводной или беспроводной. Проводная среда также относится к сетевому оборудованию и часто называется кабельной системой, представленной одним из трех видов проводников – витой парой, коаксиальным кабелем или оптоволоконным кабелем. В качестве беспроводной среды обычно используется радиоканал.

Коаксиальный кабель состоит из центрального медного проводника, слоя изолятора в алюминиевой или медной оплетке и внешней изоляции. Преимущество данного кабеля - помехоустойчивость, что достигается за счет оплетки кабеля. Недостатками являются невозможность одновременной передачи нескольких сигналов из-за использования всего двух проводников, а также сложность монтажа из-за повышенной жесткости. Коаксиальный кабель в основном использовался на ранних этапах развития компьютерных сетей, обеспечивая скорость передачи до 10 Мбит/сек.

Самый распространенный проводник для построения локальных сетей – это витая пара, которая состоит из нескольких скрученных между собой пар изолированных проводников. Передача данных происходит по методу точка-точка (один приемник и один передатчик). Задержка сигнала в среднем составляет 4-5 наносекунд на метр кабеля. Достоинствами данного кабеля являются простой монтаж и низкая стоимость по сравнению с другими кабелями, недостатками - подверженность помехам и низкая секретность передачи информации. Скорость передачи информации по витой паре зависит от её категории и может достигать 40 Гбит/сек для категории 8.

Оптоволоконный кабель представляет собой несколько стеклянных световодов, которые защищены изоляцией. а источником сигнала при передаче данных служит мощный светодиод (лазер). При этом сигнал передается без затухания на расстояние, измеряемое в километрах. Достоинствами оптоволоконного кабеля являются помехоустойчивость и высокая информационная безопасность. Скорость передачи информации в таком кабеле составляет 10 Гбит/сек.

**Сетевая карта** (сетевой адаптер) служит для объединения компьютеров в локальную сеть и для подключения компьютера к сети Интернет. Каждая сетевая карта имеет свой индивидуальный номер – MAC-адрес, который предоставляется производителем. MAC-адрес позволяет однозначно определить каждый узел сети и доставить данные именно этому узлу.

В кабельной системе данные передаются в виде световых или электрических импульсов, которые называются сигналами. По мере прохождения сигналы становятся слабыми и менее различимыми, поэтому для передачи на большие расстояния применяют специальные устройства –**повторители** (repeater), которые усиливают эти сигналы. Повторитель имеет два порта – входной и выходной.

**Концентратор** (hub) – это сетевое устройство, имеющее несколько портов (от 4 до 32) и служащее для объединения компьютеров в сеть Ethernet. Концентратор часто называют многопортовым повторителем (multiport repeater), так как он не может определить, какому узлу сети предназначен передаваемый кадр, и поэтому ретранслирует входящие кадры с одного порта на все подключенные порты, что снижает уровень безопасности. В настоящее время концентраторы сняты с производства и встречаются редко.

**Коммутатор** (switch) – устройство, предназначенное для объединения узлов в локальную сеть, но в отличие от концентратора он передает данные конкретному узлу. Коммутатор находит в принятом кадре MAC-адреса отправителя и получателя, соединяет порты, к которым подключены эти узлы и передает данные, оставляя остальные порты свободными. В сетях с большим количеством узлов коммутатор позволяет значительно увеличить производительность сети за счет увеличения фактической скорости передачи кадров и уменьшения количества паразитного трафика.

Коммутатор хранит в памяти таблицу, которая определяет связь MAC-адреса узла с соответствующим портом коммутатора. При включении коммутатора эта таблица пуста и коммутатор начинает работать в режиме обучения. В этом режиме при поступлении на один из портов кадров они дублируются на все остальные подключенные порты. При этом коммутатор проводит анализ кадра, определяет MAC-адрес отправителя и заносит в таблицу этот адрес и номер порта, на который поступил кадр. Таким образом в процессе работы коммутатор строит таблицу для всех своих портов.

Если поступает кадр для узла, MAC-адрес которого уже есть в таблице, то кадр будет передан только через порт, номер которого указан в соответствующей строке таблицы.

**Маршрутизатор** (router) – сетевое устройство, которое на основании информации о структуре сети по определенному алгоритму выбирает маршрут для пересылки пакетов между различными сетями, в том числе несовместимыми по архитектуре и протоколам. Маршрутизация проводится на основе анализа сетевых IP-адресов, которые содержатся в каждом принятом пакете или дейтаграмме. Также маршрутизатор используется для обеспечения доступа из локальной сети в глобальную сеть Интернет, осуществляя при этом функции межсетевого экрана.

2.1.2 Топология сети

Топология сети определяет физическое размещение компьютеров сети по отношению друг к другу. Выделяют три основных топологии: шина, кольцо, звезда:

* при построении сети по шинной схеме все компьютеры присоединяется к одному общему кабелю;
* при кольцевой топологии все узлы соединяются каналами связи в неразрывное кольцо, выход одного компьютера соединяется с входом другого, а данные в кольце всегда движутся в одном и том же направлении, причем начинают и заканчивают движение в одной точке;
* при построении сети по топологии «звезда» каждый компьютер сети подсоединяется кабелем к портам одного коммутатора.

2.1.3 Типы IP-адресов

Все IP-адреса протокола IPv4 делятся на публичные (глобальные, внешние, белые) и частные (локальные, внутренние, серые). Публичным IP-адресом называется IP-адрес, который используется для выхода в Интернет. Доступ к устройству с публичным IP-адресом можно получить из любой точки глобальной сети, т.к. эти IP-адреса маршрутизируются в Интернете в отличие от частных IP-адресов. Количество публичных адресов протокола IPv4 ограничено, поэтому в локальных сетях используются частные IP-адреса.

Частный IP-адрес принадлежит к специальному диапазону, не используемому в сети Интернет. Следующие диапазоны определены как адреса, выделенные для локальных сетей:

10.0.0.0 – 10.255.255.255 (маска подсети 255.0.0.0);

172.16.0.0 — 172.31.255.255 (маска подсети 255.240.0.0);

192.168.0.0 — 192.168.255.255 (маска подсети 255.255.0.0)

2.2 Эмуляторы компьютерной сети

В настоящее время наиболее распространенными эмуляторами являются программы NetEmul, Cisco Packet Tracer, GNS3.

Программа **NetEmul** предназначена в основном для обучения сетевым технологиям разного уровня – от базовых понятий до особенностей обработки отдельных полей сетевых пакетов, а также основным принципам администрирования IP-сетей. Сайт программы - http://netemul.sourceforge.net.

NetEmul включает в себя следующие возможности и технологии:

* маршрутизация, система моделирования каналов, IP-фильтрация;
* типы пакетов: ICMP, UDP, TCP, а так же низкоуровневые ARP-запросы;
* концепция интерфейсов и сокетов (простой, дейтаграммный и потоковый);
* эмуляция хостов, коммутаторов второго уровня и концентраторов;
* установка уровня помех на канале;
* связывание нескольких Network Emulator через реальную сеть TCP/IP.

Эмулятор **Cisco Packet Tracer** разработан компанией Cisco и рекомендован для изучения телекоммуникационных сетей и сетевого оборудования. Программа имеет следующие особенности:

* моделирование логической топологии;
* моделирование в режиме реального времени;
* моделирование физической топологии;
* улучшенный GUI, необходимый для более качественного понимания организации сети и принципов работы устройств;
* усовершенствованное изображение сетевого оборудования со способностью добавлять / удалять различные компоненты;
* возможность создания шаблонов сетей и их последующее использование.

Packet Tracer способен моделировать большое количество устройств различного назначения, а также немало различных типов связей, что позволяет проектировать сети любого размера на высоком уровне сложности. Моделируемые устройства:

* коммутаторы третьего уровня (Router 2620XM, Router 2621 XM);
* коммутаторы второго уровня (Switch 2950-24);
* сетевые концентраторы (HubPT);
* оконечные устройства (рабочая станция PC\_PT);
* беспроводные устройства (точка доступа AccessPoint-PT);
* глобальная сеть WAN.

Cisco Packet Tracer не является свободно распространяемым программным обеспечением, доступен бесплатно только для участников Программы сетевой академии Cisco. Сайт разработчика: https://www.netacad.com/ru/courses/packet-tracer.

**GNS3** - это графический симулятор сети, который позволяет смоделировать виртуальную сеть из маршрутизаторов и виртуальных машин, работает практически на всех платформах. Одной из особенностей GNS3 является возможность соединения проектируемой топологии с реальной сетью. Это дает возможность проверить на практике какой-либо проект без использования реального оборудования.

Программа распространяется по лицензии GPL, сайт программы: http://www.gns3.com.

2.3Интерфейс эмулятора NetEmul

При выполнении данной лабораторной работы для имитации работы компьютерной сети будет использоваться эмулятор NetEmul. Главное окно программы представляет собой панель устройств, панель параметров и рабочую область. На панели устройств расположены сетевые устройства и кнопки запуска и остановки симуляции в соответствии с рисунком 2.1. На панели параметров отображаются только те параметры, которые доступны для выбранного устройства.

Для того чтобы добавить устройство на рабочую область необходимо выбрать устройство на панели устройств и поместить его в свободную клетку рабочей области. У каждого используемого устройства есть индикатор, который показывает его состояние. Красный индикатор означает, что устройство не подключено, желтый – подключено, но не настроено, зелёный – устройство подключено и настроено. Удаление устройства из рабочей области выполняется через контекстное меню.

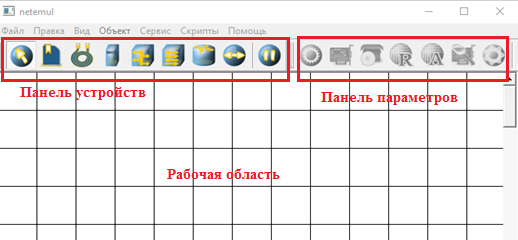


Рис. 2.1 Окно программы NetEmul

Создадим простую сеть, состоящую из двух компьютеров и коммутатора. Для этого на рабочую область поместим два компьютера и концентратор (рисунок 2.2) и соединим компьютеры с концентратором, используя порт сетевой карты eth0. Жёлтый цвет индикатора означает, что компьютеры подключены.

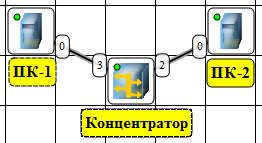


Рис. 2.2 Простая сеть

Настроим сетевую карту компьютеров. Вызовем контекстное меню, нажав правой кнопкой мыши на компьютере, и выберем пункт «интерфейсы». В открывшемся окне необходимо ввести IP-адрес компьютера. Зададим IP-адреса 192.168.10.1 и 192.168.10.2 для первого и второго компьютера соответственно, маска 255.255.255.0 появится автоматически, при необходимости её можно изменить. Также в окне «Интерфейс» можно увидеть информацию о получаемых и передаваемых кадрах и пакетах, а также MAC-адрес устройства, который выдается автоматически. После настройки у компьютеров должен появиться зеленый индикатор и начнется движение кадра ARP-протокола, что необходимо для поиска совпадающих IP-адресов. Если таковые найдутся, то высветится сообщение об ошибке.

Отправим данные с одного компьютера на другой. Для этого нажмите кнопку «Отправить данные», находящуюся на панели устройств, а затем на компьютер, с которого будем отправлять данные. В открывшемся окне выберите протокол и размер данных. Далее нажмите на компьютер-получатель и выберите «Интерфейс приёмника». Затем нажмите кнопку «Запустить» и пакеты данных начнут двигаться по смоделированной сети. Одновременно с этим будет проводиться запись в журнал всех сведений о прохождении пакетов в текстовом виде.

2.4 Сетевой калькулятор LanCalculator

Программа LanCalculator предназначена для расчёта IP-адресов при проектировании сети, что дает возможность назначить адреса с учетом маски подсети, рассчитать адрес сети, широковещательный IP-адрес, общее число IP-адресов в подсети и т.д. Для выполнения расчета необходимо ввести IP-адрес и маску подсети (или её префикс) и нажать кнопку «Рассчитать» (рисунок 2.3).

Для наглядности результаты расчета могут выводиться в двоичной, десятичной или шестнадцатеричной нотации.

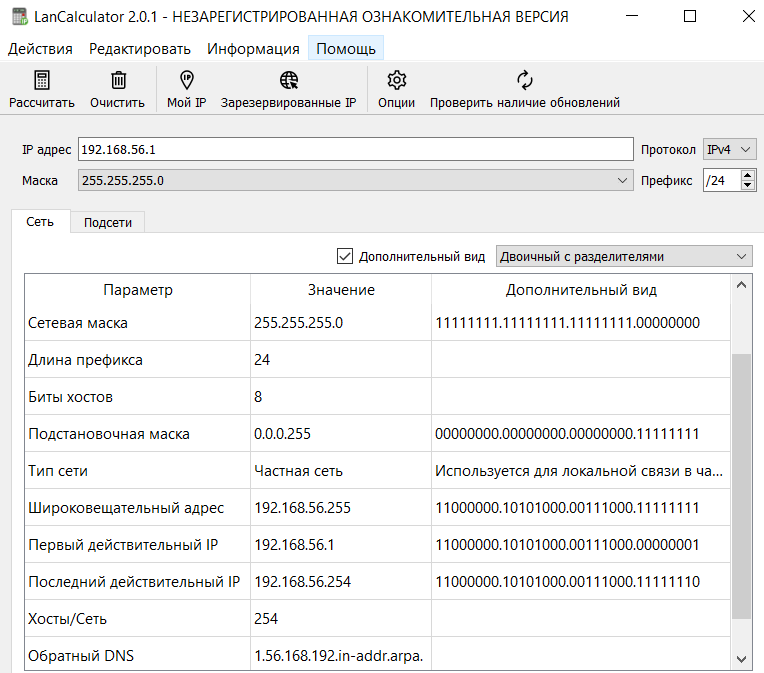


Рис. 2.3 Интерфейс LanCalculator

3. Примеры моделирования сетей с различными топологиями

*Задание № 1.**Смоделировать одноранговую локальную сеть, состоящую из одного концентратора и двух компьютеров. IP-адреса компьютерам выдать статически, начальный IP-адрес – 172.27.34.0/10.*

Поместим на рабочую область два компьютера и концентратор. Соединим порт сетевых карт компьютеров с портами концентратора с помощью кабеля. Произведём настройку сетевых карт следующим образом: назначим первому компьютеру начальный ip-адрес 172.27.34.0, второму компьютеру – адрес 172.27.34.1. Маску сети согласно задания будет равна 255.192.0.0. Продемонстрируем работу построенной сети, отправив данные с одного компьютера на другой.

*Задание № 2.**Вариант: 172.27.34.0/10*

*Поместим на рабочую область 3 компьютера и коммутатор. Соединим все компьютеры с коммутатором. Настроим сетевые карты компьютера в соответствии с вариантом. Первому компьютеру выдадим ip-адрес 172.27.34.0, второму - 172.27.34.2 и третьему- 172.27.34.2, а также маску для всех компьютеров 255.192.0.0.*

**Задание № 3.** Разместим пять компьютеров и коммутатор на рабочей области. Увеличим количество портов коммутатора до 5, сделав следующие действия: в контекстном меню коммутатора выберем пункт «Свойства», в открывшемся окне выберем количество, равное 5. Соединим все компьютеры с коммутатором и настроим сетевую карту компьютера-сервера: IP-адрес 192.168.1.1 и маска 255.255.255.0.

Установим на этот компьютер DHCP-сервер. В контекстном меню компьютера выберем пункт «Программы». В окне «Программы» показаны программы, установленные на компьютер, в данный момент список пуст. Нажмем кнопку «Добавить» и выберем из списка доступных программ DHCP-сервер. В списке установленных программ появится DHCP сервер. Отметив эту программу галочкой и нажав кнопку «Настройки», перейдем к настройкам DHCP сервера, выбрав динамическое распределение адресов. Укажем диапазон адресов, который сервер будет распределять между компьютерами от 172.27.34.1 до 172.27.34.4 и маску 255.192.0.0. Так как в динамическом распределении адрес выдается на временное пользование, укажем срок аренды, например, 500 сек.

На другие компьютеры необходимо установить программу DHCP-клиент. И при установке программы в настройках выбрать eth0 в качестве интерфейса, контролируемого DHCP. Так же этим четырем компьютерам необходимо в интерфейсе поставить галочку напротив «Получать настройки автоматически», после чего компьютеры-клиенты получат от DHCP-сервера нужные параметры.

**Задание № 4:** Вариант: 192.168.15.1/18

Помещаем на рабочую область необходимое оборудование. Соединим коммутаторы между собой, два компьютера подключим к одному коммутатору, остальные – к другому. Начиная с 192.168.15.1 и увеличивая на единицу, раздадим компьютерам ip-адреса и выдадим маску 255.255.192.0.

Протестируем сеть, создав и отправив пользовательский пакет. В контекстном меню компьютера выберем пункт «Конструктор пакетов». Откроем вкладку «Кадр» и отметим галочкой IP. Далее перейдем во вкладку «IP», выберем протокол TCP и зададим IP-адреса отправителя и получателя. После нажатия кнопки «Ок» начнется анимация передачи данных.

**Задание № 5:** Вариант: Начальный ip-адрес и маска для первой подсети 192.168.78.1/24, для второй – 10.81.0.1/16, для третьей – 172.17.98.0/17.

Расположим маршрутизатор, два коммутатора и не менее шести компьютеров на рабочей области. Соединим сеть кабелем. Настроим маршрутизатор. Настроим сетевые карты для подсетей, в соответствии с вариантом:

LAN1: начальный ip-адрес 192.168.78.1, маска 255.255.255.0;

LAN2: начальный ip-адрес 10.81.0.1, маска 255.255.0.0;

LAN3: начальный ip-адрес 172.17.98.0, маска 255.255.128.0.

В свойствах маршрутизатора необходимо выставить флаг напротив «Включить маршрутизацию». Далее настроим компьютеры подсети LAN1, выдав IP-адреса начиная с 192.168.78.1 и задав маску 255.255.255.0, также нужно установить шлюз по умолчанию, через контекстное пункт «Свойства». В окне свойства присвоим шлюз по умолчанию 192.168.78.0 и поставим галочку напротив пункта «Включить маршрутизацию». Аналогично настроим подсеть LAN2 и LAN3. Для подсети LAN2 – IP-адреса начиная с 10.81.0.1, маска 255.255.0.0 и шлюз по умолчанию 10.81.0.0, для подсети LAN3 – начиная с 172.17.98.0, маска 255.255.128.0 и шлюз по умолчанию 172.17.0.0.

Установим на маршрутизатор программу RIP через контекстное меню маршрутизатора пункт «Программы».

3. Порядок выполнения работы

1. Смоделировать одноранговую локальную сеть, состоящую из одного концентратора и двух компьютеров. IP-адреса компьютерам выдать статически в соответствии с заданным вариантом из таблицы 2.1. Протестируйте сеть, отправив пользовательские пакеты.

Таблица 2.1 - Диапазон адресов для 1 и 2 задания

|  |  |  |
| --- | --- | --- |
| Номер бригады | Начальный IP-адрес для 1 и 2 задания | Начальный IP-адрес для 4 задания |
| 1 | 172.27.34.0/10 | 192.168.15.1/18 |
| 2 | 10.72.0.1/19 | 10.10.14.1/23 |
| 3 | 192.168.11.1/24 | 172.17.98.0/17 |
| 4 | 10.10.84.0/20 | 192.168.12.1/10 |
| 5 | 192.168.15.1/12 | 10.10.11.0/15 |
| 6 | 172.30.6.0/21 | 172.18.73.0/20 |
| 7 | 10.10.56.0/13 | 192.168.9.1/14 |
| 8 | 172.98.8.1/22 | 172.98.8.1/22 |
| 9 | 192.168.9.1/14 | 10.10.56.0/13 |
| 10 | 172.18.73.0/20 | 172.30.6.0/21 |
| 11 | 10.10.11.0/15 | 192.168.15.1/12 |
| 12 | 192.168.12.1/10 | 10.10.84.0/20 |
| 13 | 172.17.98.0/17 | 192.168.11.1/24 |
| 14 | 10.10.14.1/23 | 10.72.0.1/19 |
| 15 | 192.168.15.1/18 | 172.27.34.0/10 |

1. Смоделировать локальную сеть топологии «Звезда», состоящую из трёх компьютеров и одного коммутатора. IP-адреса компьютерам выдать статически в соответствии с заданным вариантом из таблицы 2.1.
2. Смоделировать локальную сеть топологии «Звезда», состоящую из пяти компьютеров и одного коммутатора. IP-адреса компьютерам выдать динамически, используя DHCP протокол.
3. Смоделировать локальную сеть топологии «Дерево», состоящую из пяти компьютеров и двух коммутаторов. IP-адреса компьютерам выдать статически в соответствии с заданным вариантом из таблицы 2.1.
4. Смоделировать локальную сеть, разделенную на три подсети. Используйте протокол маршрутизации RIP. В соответствии с заданным вариантом статически раздать IP-адреса компьютерам из таблицы 2.2.

Таблица 2.2 - Диапазон адресов для 5 задания

|  |  |  |  |
| --- | --- | --- | --- |
| Номер бригады | Начальный ip-адрес для первой подсети | Начальный ip-адрес для второй подсети | Начальный ip-адрес для третьей подсети |
| 1 | 192.168.78.1/24 | 10.81.0.1/16 | 172.17.98.0/17 |
| 2 | 10.79.1.1/24 | 192.168.82.0/17 | 172.18.73.0/20 |
| 3 | 10.10.80.1/24 | 192.168.84.1/24 | 172.27.34.0/10 |
| 4 | 10.10.56.0/13 | 192.168.82.0/17 | 172.30.6.0/21 |
| 5 | 10.82.0.1/16 | 172.26.86.0/10 | 192.168.14.1/23 |
| 6 | 192.168.8.1/24 | 172.28.83.1/24 | 10.10.11.0/15 |
| 7 | 172.24.48.1/24 | 192.168.9.0/17 | 10.72.0.1/19 |
| 8 | 10.10.85.1/24 | 192.168.19.1/24 | 172.98.8.1/22 |
| 9 | 192.168.6.1/24 | 10.10.11.0/16 | 192.168.9.0/14 |
| 10 | 192.168.0.1/16 | 192.168.85.1/24 | 10.10.84.0/20 |
| 11 | 10.88.1.1/24 | 192.168.87.0/10 | 10.10.56.0/13 |
| 12 | 192.168.89.0/22 | 172.29.89.1/24 | 192.168.11.0/11 |
| 13 | 10.90.0.1/16 | 172.16.80.1/24 | 192.168.12.0/10 |
| 14 | 10.91.1.1/24 | 172.24.82.0/15 | 192.168.15.0/18 |
| 15 | 192.168.29.1/24 | 172.17.90.0/21 | 192.168.15.0/12 |

Каждый вариант сети необходимо протестировать, отправив пользовательские пакеты, результаты моделирования занести в отчет.

В таблицах вариантов IP-адрес представлен в префиксной форме (CIDR-нотации). Маска в такой записи указывается в виде числа после символа «слэш». Например, маска 255.255.255.128 (в двоичном виде будет выглядеть 11111111.11111111.11111111.10000000. Здесь количество единиц равняется 25, поэтому маска будет записываться как /25.

4. Контрольные вопросы

1. Назовите основные виды сетевых проводников. Перечислите их достоинства и недостатки.

2. В чем заключаются функции маршрутизатора?

3. Назовите достоинства и недостатки одноранговой сети.

4. Как строится и функционирует сеть топологии звезда? Назовите достоинства и недостатки.

5. Каково назначение маски подсети?

6. Опишите принцип работы концентратора.

7. Опишите принцип работы коммутатора.

8. Назовите назначение концентратора и коммутатора. В чем заключается отличие концентратора и коммутатора?

9. Что такое DHCP ? Необходим ли подобный сервис в локальных сетях ?

# Лабораторная работа № 2. Анализ трафика компьютерной сети

1. Цель работы

Целями работы является изучение программного обеспечения, предназначенного для контроля и анализа сетевого трафика, а также получение практических навыков работы с программой WireShark.

2. Методические указания

2.1 Общие сведения

Мониторинг трафика жизненно важен для эффективного управления сетью. Он является источником информации о функционировании корпоративных приложений, которая учитывается при распределении средств, планировании вычислительных мощностей, определении и локализации отказов, решении вопросов безопасности.

Одной из актуальных задач в настоящее время является анализ и дальнейшее прогнозирование структуры трафика в современных мультисервисных сетях. Для решения этой задачи необходимо проводить сбор и анализ разнообразной статистики в действующих сетях - скорость, объемы переданных и принятых данных и т.д. Сбор такой статистики в том или ином виде возможен различными программными средствами.

В настоящее время разработано достаточно много программных продуктов, предназначенных для анализа и мониторинга сетевого трафика, каждый из которых обладает своими достоинствами и недостатками. Анализатор трафика или сниффер (от англ. to sniff - нюхать) – это программа или программно-аппаратный комплекс, предназначенный для перехвата и последующего анализа сетевого трафика для всех узлов сети. Во время работы сниффера сетевой интерфейс переключается в режим прослушивания, что и позволяет ему получать пакеты, адресованные другим интерфейсам в сети.

Перехват трафика может осуществляться:

* обычным "прослушиванием" сети, что эффективно при использовании в сегменте концентраторов (хабов), но малоэффективно при использовании коммутаторов (свитчей), поскольку на сниффер попадают лишь отдельные фреймы;
* подключением сниффера в разрыв канала;
* ответвлением (программным или аппаратным) трафика и направлением его копии на сниффер;
* через анализ побочных электромагнитных излучений с последующим восстановлением трафика;
* через атаку на канальном (МАС) или сетевом (IP) уровне, приводящую к перенаправлению трафика на сниффер с последующим возвращением трафика в надлежащий адрес.

В 1990-х снифферы широко применялись хакерами для захвата пользовательских логинов и паролей, которые в ряде протоколов передаются в незашифрованном или слабо зашифрованном виде, а использование концентраторов позволяло захватывать трафик в больших сегментах сети практически без риска быть обнаруженным. В наcтоящее время снифферы находят применение в сетевом администрировании для следующих целей:

* обнаружение паразитного, вирусного и закольцованного трафика, наличие которого увеличивает загрузку сетевого оборудования и каналов связи. Однако снифферы для этих целей не очень эффективны, т.к. более удобным является использование специализированных программ для сбора разнообразной статистики по активности серверов и другого сетевого оборудования;
* выявление в сети вредоносного и несанкционированного ПО, например, сетевых сканеров, флудеров, троянов, клиентов пиринговых сетей и других (для этого используют специализированные снифферы – мониторы сетевой активности);
* локализация неисправностей сети или ошибок конфигурирования оборудования.

Одним из простых и свободно распространяемых программных продуктов этого класса является программа WireShark, распространяемая под свободной лицензией GNU GPL для многих операционных систем (GNU/Linux, FreeBSD, OpenBSD, Mac OS X, Windows).

2.2 Анализатор трафика WireShark

WireShark – это анализатор сетевого трафика, основная задача которого состоит в том, чтобы перехватить сетевой трафик и отобразить его в детальном виде. Он работает на основе библиотеки Pcap (packet capture), позволяющей разрабатывать программы для анализа данных, поступающих на сетевую карту компьютера.

Возможности WireShark:

* перехват трафика сетевого интерфейса в режиме реального времени для различных сетевых устройств, включая беспроводные устройства;
* наличие множества протокольных декодировщиков (TELNET, FTP, POP, RLOGIN, ICQ, SMB, MySQL, HTTP, NNTP, X11, NAPSTER, IRC, RIP, BGP, IMAP 4, VNC, LDAP, NFS, SNMP, MSN, YMSG и другие);
* сохранение и просмотр ранее сохраненного сетевого трафика;
* импорт и экспорт файлов из других пакетных анализаторов;
* фильтрация и поиск пакетов по множеству критериев, а также создание разнообразной статистики.

Рассмотрение возможностей программы будем проводить на примере локализованной версии WireShark 3.0.6.

2.2.1 Перехват трафика

Для анализа трафика нужно сначала указать сетевой интерфейс, с которого будет выполняться перехват. Это делается через пункт меню Захват/Опции (рис. 2.1**Рис.** ) путем выбора из списка доступных интерфейсов в окне «Интерфейсы захвата» (рис.2.2).

Раскрыв каждый из пунктов списка сетевых интерфейсов, можно получить информацию об IP-адресе интерфейса, его сетевой активности, посмотреть настройки перехвата. В настройках перехвата можно изменять такие параметры, как фильтрация пакетов, запись дампа в несколько файлов, прекращение перехвата по разным критериям (количество пакетов, количество мегабайт, количество минут), опции отображения пакетов. Интерфейс по умолчанию можно указать в настройках программы через меню Редактирование/Параметры/Capture.

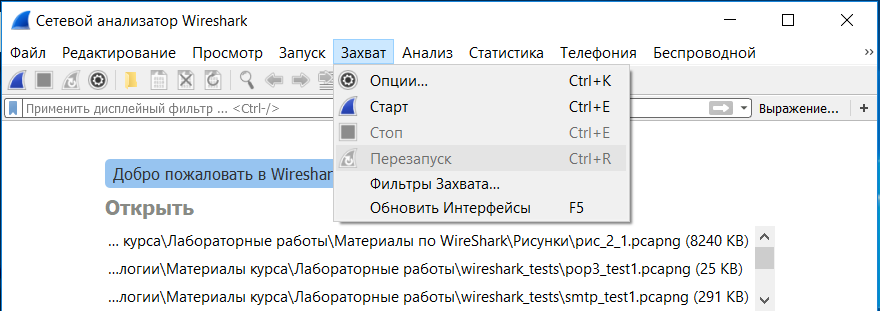


Рис. 2.1 Выбор интерфейса для перехвата

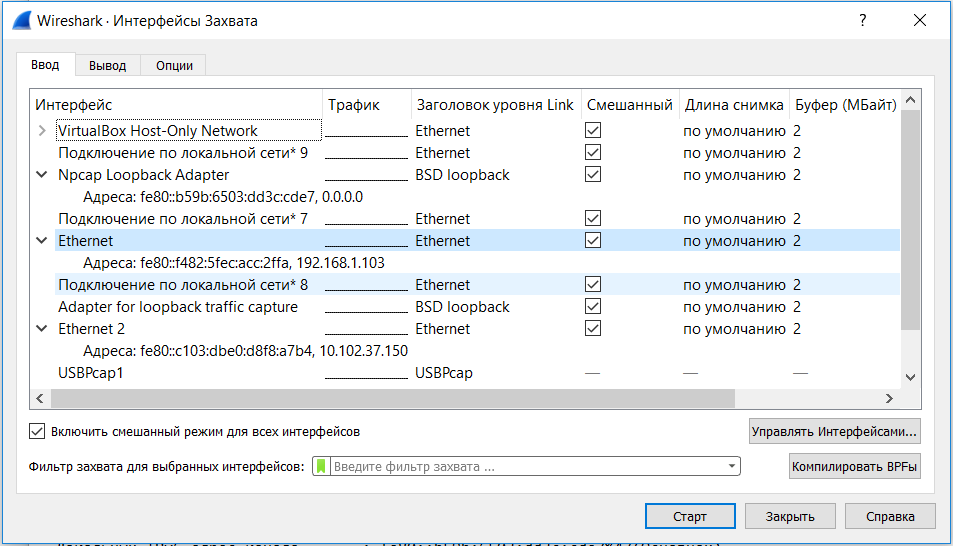


Рис. 2.2 Список сетевых интерфейсов

После нажатия кнопки Start запускается перехват и, если сетевая активность высокая, можно сразу увидеть входящие и исходящие пакеты (рисунок 2.3). Основное окно WireShark содержит три панели просмотра с различными уровнями детализации:

* верхняя панель содержит список всех собранных пакетов с их кратким описанием;
* в средней панели показывается дерево протоколов, инкапсулированных в выбранный пакет, при этом ветви дерева могут быть раскрыты для повышения уровня детализации выбранного протокола;
* нижняя панель содержит дамп выбранного пакета в шестнадцатеричном и текстовом представлении.

В верхней панели по умолчанию отображается 6 колонок: номер пакета в списке, его временная метка, адреса отправителя и получателя, название протокола и краткое описание пакета. Выбрав в списке конкретный пакет, его содержимое можно просмотреть в средней панели, где отображается дерево всех протоколов, инкапсулированных в выбранный пакет. Дерево отображает каждое поле и его значение для заголовков всех протоколов стека.

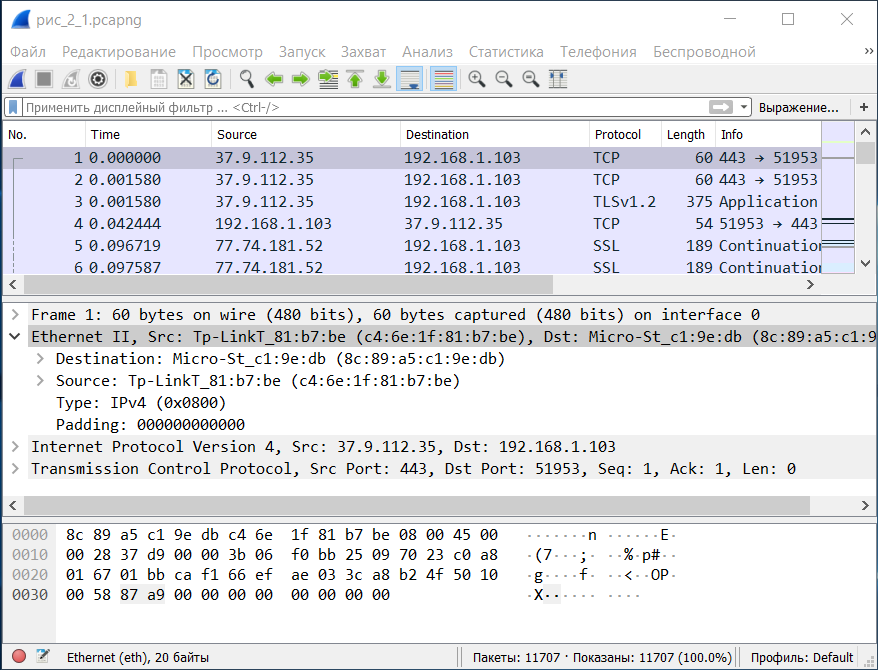


Рис. 2.3 Процесс перехвата пакетов

Если в средней панели кликнуть мышкой на пиктограмме со стрелкой у заголовка какого-либо протокола, то можно просмотреть состав и содержимое этого протокола. При этом выбранная область протокола выделяется подсветкой в дампе пакета нижнего окна.

Настройка общих параметров WireShark `проводится в меню Редактирование/Параметры, параметров просмотра – в меню Просмотр (количество панелей для просмотра пакетов, тип временных меток, выделение цветом и т.д.).

Поиск пакета по заданному признаку можно проводить через меню Редактирование/Найти пакет или нажатием Ctrl+F, после чего в строке поиска надо задать область поиска (список пакетов, информация о пакете или байты пакета), тип данных (строка, шестнадцатиричное число, регулярное выражение) и значение для поиска.

2.2.2 Фильтрация пакетов

В верхней части основного окна имеется панель для управления фильтрацией пакетов. Фильтрация возможна как по протоколам, так и по отдельным полям заголовков этих протоколов.

Значение фильтра задается с помощью зарезервированных слов, обычно представляющих собой названия протокола или его полей заголовков, операции отношения и конкретного значения в шестнадцатеричном или десятичном виде. Например, выражение с операцией отношения «равно» записывается с помощью двойного знака равенства «==» (допустимо использование «eq»). Другие операции отношения записываются с помощью операторов, представленных в таблице 2.1.

Таблица 2.1

|  |  |
| --- | --- |
| Оператор отношения | Пример |
| a.!= (ne)— не равно, | eth.type!= 0x0800; |
| b. > (gt)— больше, | tcp.srcport > 1023 |
| c. < (lt)— меньше, | frame.pkt\_len lt 60; |
| d. >= (ge)— больше или равно, | frame.pkt\_len ge 60; |
| <= (le)— меньше или равно, | tcp.dstport <=1023. |

Условие для фильтрации можно вводить в ручном режиме или с помощью построителя выражений, значение условия возвращает переменную булевского типа. Например, выражение «udp» означает присутствие в кадре заголовка протокола UDP, по аналогии с этим выражение «tcp.flags.syn» означает присутствие в заголовке протокола TCP бита синхронизации сессии в установленном состоянии (значение бита равно 1). К любому из выражений можно применить операцию логического отрицания, заключив его в скобки и поставив перед ним знак отрицания «NOT» или «!». Например, выражение «!(ip.addr == 10.0.0.1)» означает, что из буфера отображения необходимо убрать все пакеты, в которых встречается IP-адрес 10.0.0.1. Включение и отключение режима фильтрации проводятся кнопками, расположенными в правой части панели фильтрации.

Перехват пакетов протокола FTP

Протокол FTP был одним из первых протоколов, разработанных в 70-х годах для обмена файлами между удаленными друг от друга компьютерами. Основные свойства протокола:

* обеспечивает надежную передачу с применением TCP-соединения и использованием технологии «клиент-сервер»;
* передает данные открытым текстом, не обеспечивая их защиту от несанкционированного доступа;
* позволяет передавать текстовые и двоичные файлы между различными файловыми системами;
* команды и данные передаются по разным портам.

В настоящее время в сети Интернет имеется множество FTP-серверов с открытым доступом. Список таких серверов можно найти на сайте **mmnt.net**. Большинство FTP-серверов имеют функцию аутентификации клиента. В этом случае в качестве имени пользователя необходимо использовать имя “anonymous”, в качестве пароля обычно указывается адрес своей электронной почты.

Для выполнения лабораторной работы можно использовать любой доступный сервер, например, FTP-сервер НГТУ ([ftp.nstu.ru](ftp://ftp.nstu.ru)). Подключение можно проводить через командную строку (например, командой **ftp ftp.nstu.ru**) или с помощью любого браузера, предварительно включив в WireShark режим захвата пакетов. С сервера необходимо скачать два файла из корневого каталога: INDEX и NEW-THIS-WEEK.

*Обратите внимание: сервер работает под управлением ОС Linux, поэтому имена файлов регистрозависимы.*

На рисунке 2.4 приведены результаты перехвата, из которых видно, что клиент и сервер работают в режиме запрос-ответ (Request, Response) и команды, включая логин и пароль, передаются в открытом виде без шифрования. Например, для чтения файла сервер выполняет команду RETR.

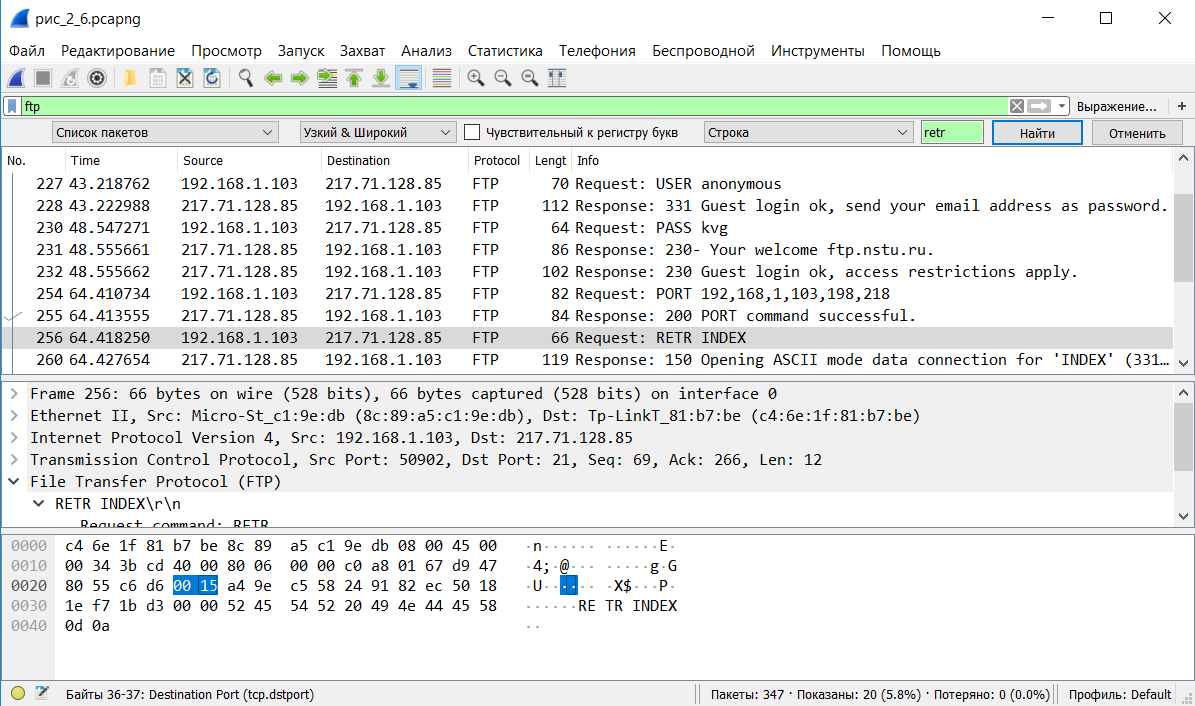


Рис. 2.4 Перехват логина и пароля

Для просмотра FTP-команд необходимо задать значение фильтра “ftp”, а для просмотра данных – “ftp-data”.

Перехват пакетов протокола ICMP

Для перехвата пакетов ICMP после включения режима захвата введем в консоли команду **ping yandex.ru**, как показано на рисунке 2.5, а после ее выполнения в поле ввода фильтра Wireshark укажем значение «icmp» и нажмем кнопку «Применить фильтр». Результат поиска приведен на рис. 2.6, где мы можем наблюдать, как происходит обмен запросами и ответами echo Request и echo Reply в протоколе ICMP изнутри: какие тестовые данные посылаются, какие флаги символизируют о том, что это именно Echo Request, и другую не менее важную информацию.

Обратите внимание: выражение для фильтра вводится строчными символами без кавычек !

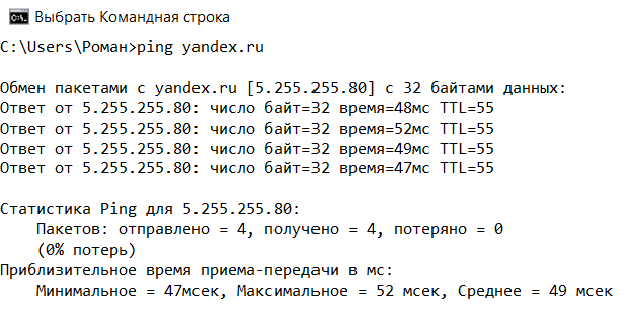


Рис. 2.5 – Выполнение команды ping

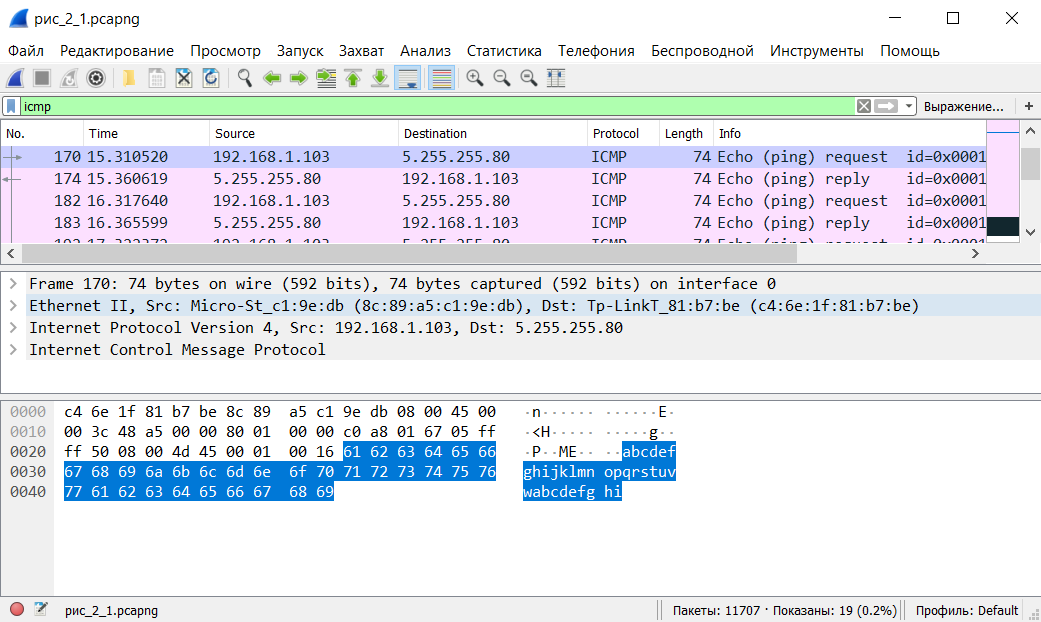


Рис. 2.6 Фильтрация по протоколу ICMP

2.2.3 Анализ перехваченного трафика

После отключения режима захвата все результаты можно записать в файл с расширением .pcapng для последующего анализа. Рассмотрим некоторые возможности Wireshark по анализу трафика.

Общие сведения можно получить в меню Статистика/Свойства файла захвата (рис. 2.7). В окно выводятся имя и размер файла захвата, сведения о процессоре и операционной системе, длительность сеанса, общее число пакетов, объем трафика и т.д.

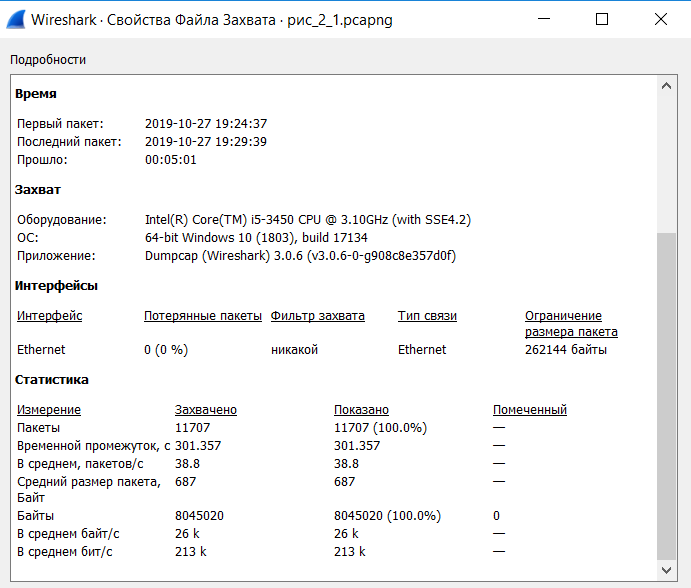


Рис. 2.7 Общие сведения о сеансе

Меню Анализ/Следовать/Поток TCP|UDP|TLS|HTTP позволяет собрать сеанс обмена воедино и посмотреть его содержимое в целом, предварительно отфильтровав пакеты по заданному протоколу. На рис. 2.8 приведен результат сборки сеанса FTP-обмена, представленного на рис.2.6.

Через меню Статистика/График Потока можно получить сведения об обмене данными по установленным фильтрам или по определенным протоколам. На рис. 2.9 приведена диаграмма потоков для протокола FTP. Пункт меню Статистика/Иерархия Протоколов выводит статистику по используемым протоколам с учетом их инкапсуляции, в том числе в процентном соотношении (рисунок 2.10).

Меню Статистика/Конечные точки выводит сведения по количеству переданных и принятых пакетов и байтов для конечных узлов сети, принимавших участие в обменах по разным протоколам (рис.2.11). Аналогичную информацию в виде диалогов можно получить из меню Статистика/Диалоги (рис. 2.12).

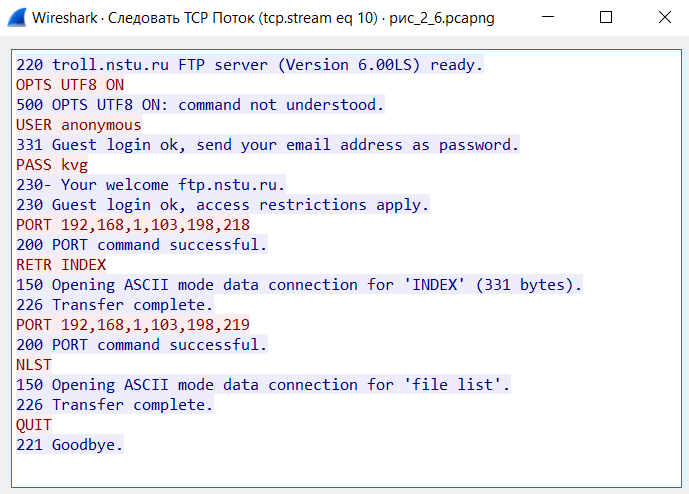


Рис. 2.8 Восстановление сеанса FTP-обмена

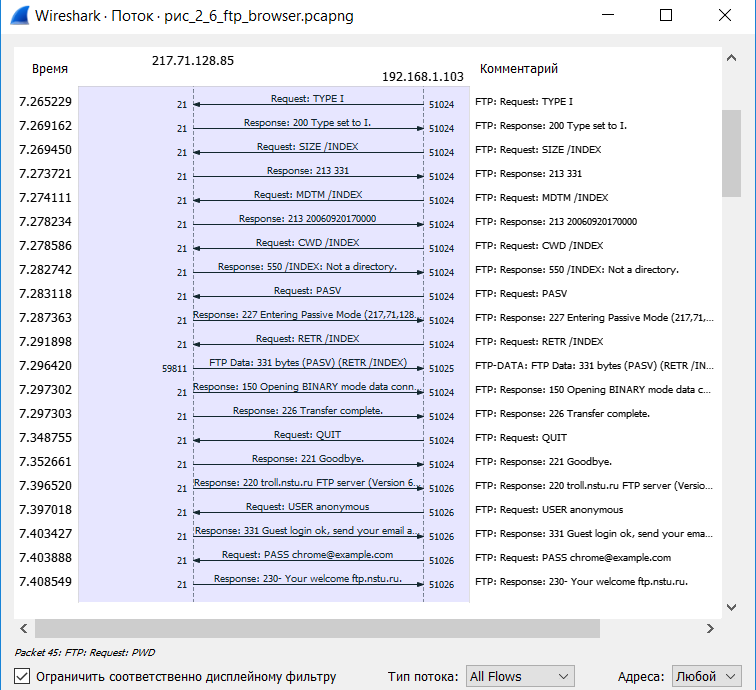


Рис. 2.9 Окно диаграммы потоков

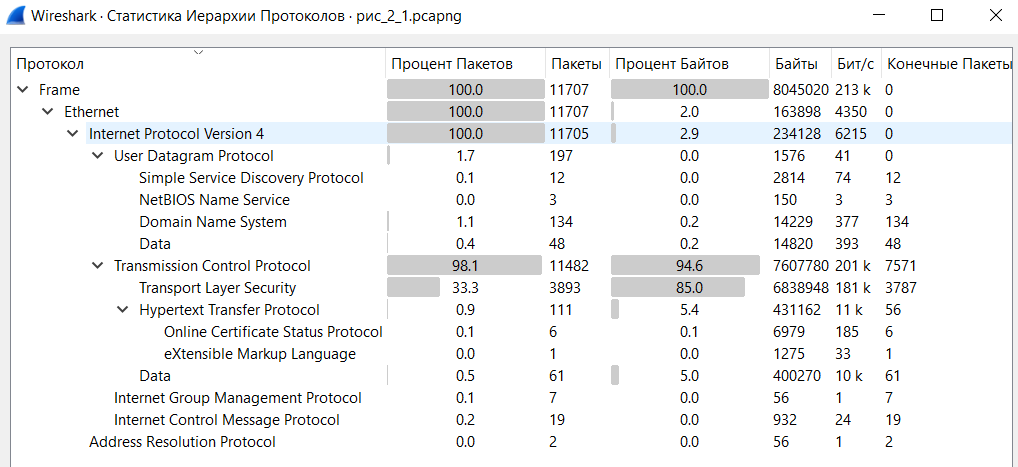


Рис. 2.10 Иерархия протоколов

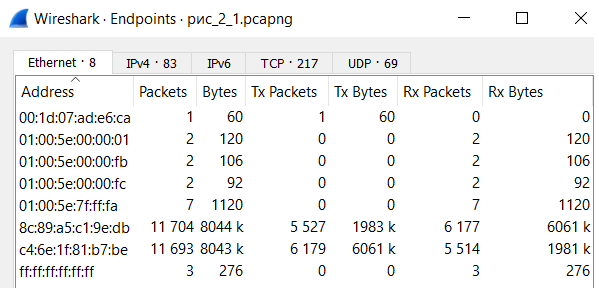


Рис. 2.11 Данные по конечным узлам

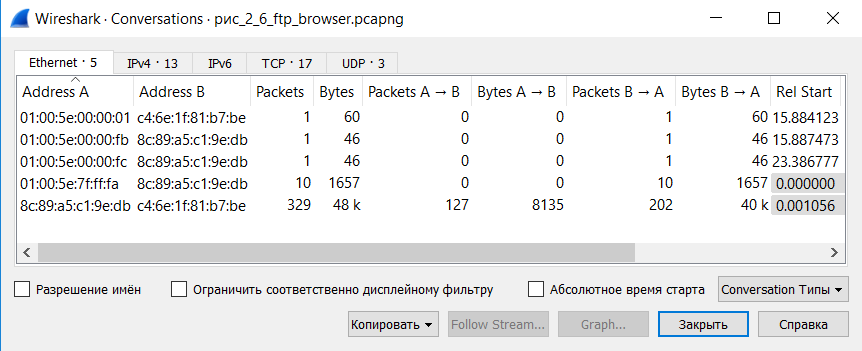


Рис. 2.12 Данные по диалогам

3. Порядок выполнения работы

1. Запустить захват сетевого трафика в WireShark, проходящего через интерфейс, подключенный к локальной или внешней сети. Эмулировать сетевую активность в течение 10 минут выполнением указанных действий:
   * посетить различные сайты, просмотреть текстовый и видеоконтент;
   * выполнить пинг и трассировку любых узлов сети Интернет;
   * с помощью браузера подключиться к серверу ftp.nstu.ru и скачать из корневого каталога файлы INDEX и NEW-THIS-WEEK;
   * отключить перехват и сохранить сеанс в файле с расширением .pcapng.
2. Выполнить фильтрацию трафика по протоколам HTTP, ICMP, ARP, FTP. Для FTP выполнить перехват команд и данных.
3. Заполнить таблицу 2.2, используя данные из отчета Статистика/Свойства файла. При заполнении таблицы обратите внимание на соблюдение размерности величин (Кбайт, Мбайт, Мбит).

Таблица 2.2

|  |  |
| --- | --- |
| Параметр | Значение |
| Время захвата, мин |  |
| К-во захваченных пакетов |  |
| Объем трафика, Мбайт |  |
| Средний размер пакета, Кбайт |  |
| Средняя скорость, пакетов/сек |  |
| Средняя скорость, Мбит/сек |  |

1. По данным отчета Статистика/Иерархия Протоколов заполнить таблицу 2.3 распределения трафика по протоколам и сделать выводы о соотношении прикладных и служебных протоколов.

Таблица 2.3

|  |  |  |
| --- | --- | --- |
| Протокол | Трафик, Мбайт | Трафик, % |
| HTTP |  |  |
| FTP |  |  |
| .. |  |  |
| Итого |  |  |

1. Заполнить таблицу 2.4 распределения Ethernet-трафика по узлам сети. Исходные данные для заполнения таблицы получить из отчета Статистика/Конечные точки. Определить, какие из узлов наиболее загружены с учетом направления трафика (исходящий, входящий, общий).

Таблица 2.4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| MAC-адрес | Трафик | | | | | |
| *входящий* | | *исходящий* | | *общий* | |
| Мбайт | % | Мбайт | % | Мбайт | % |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. По данным табл. 2.2 определить относительную загрузку сети (в %) за контрольный период времени по формуле:

http://www.4stud.info/networking/img/f1.png

Пропускную способность Интернет-соединения можно определить с помощью online-сервисов https://2ip.ru/speed/ и SpeedTest.net, а пропускную способность локальной сети – с помощью программ, информацию по которым можно найти по адресу http://wiki.compowiki.info/IzmerenieSkorostiLAN.

4. Контрольные вопросы

1. Назовите основные функции снифферов.
2. Фильтрация пакетов, способы задания значения фильтра.
3. Основные возможности WireShark по анализу пакетов.
4. Основные возможности WireShark по сбору статистики.
5. На примере полученных результатов поясните понятие инкапсуляции.
6. Поясните принцип работы программы tracert, используя результаты перехвата соответствующих пакетов.

# ЛАБОРАТОРНАЯ РАБОТА № 3. ПРОТОКОЛы стека TCP/IP

1. Цель работы

Целью работы является изучение структуры передаваемых по сети кадров и пакетов, работающих на канальном и сетевом уровне.

2. Методические указания

2.1 Общие сведения

Современные коммуникационные сети представляют собой сложные технические объекты, представленные очень большим числом разнообразного оборудования и программного обеспечения. Разработкой сетевых компонентов занимается множество различных и никак не связанных между собой организаций. Поэтому основной проблемой надежного функционирования сетей является обеспечение совместимости всех сетевых компонентов.

Основной путь решения проблемы совместимости – это наличие соглашений между разработчиками, начиная с самых общих принципов функционирования сети и заканчивая функционированием конкретных типов устройств. Только благодаря наличию таких соглашений стало возможным быстрое развитие сетей.

Соглашения оформляются в виде формализованных описаний процедур взаимодействия двух объектов, называемых протоколом. Набор протоколов, достаточный для организации взаимодействия узлов в сети, называется стеком протоколов. Как правило, протоколы разрабатываются отдельными разработчиками или группами разработчиков и затем реализуются в виде стандартов.

Существуют стандарты отдельных разработчиков, стандарты группы разработчиков (комитетов), национальные стандарты и международные стандарты. Любая новая технология внедряется повсеместно только тогда, когда её содержание закрепляется в соответствующем стандарте. Примерами международных стандартов являются стандарты Internet, которые оформляются в виде нумерованных документов RFC, или многоуровневая модель открытых систем.

2.2 Модель открытых систем

Модель открытых систем (Open System Interconnection, OSI) является обобщенным представлением сетевой архитектуры и описывает, каким образом информация проделывает путьчерез среду сети от одной прикладной программы к другой прикладной программе, выполняющейся на другом компьютере.

Модель определяет уровни взаимодействия систем в сетях с коммутацией пакетов, названия уровней и функции, которые должен выполнять каждый уровень. Она дает описание только системных средств взаимодействия, которые реализуются операционной системой, системными утилитами, системным аппаратным обеспечением и коммутационными устройствами, но не включает средства взаимодействия приложений конечных пользователей, которые могут реализовывать собственные протоколы.

Модель OSI содержит семь уровней – физический, канальный, сетевой, транспортный, сеансовый, представительный и прикладной. На рис. 2.1 показана схема взаимодействия сетевых приложений, запущенных на двух удаленных компьютерах. Здесь П1, П2, П3 и т.д. – протоколы взаимодействия соответствующих уровней.

Прикладной

Представительный

Сеансовый

Транспортный

Сетевой

Канальный

Физический

Прикладной

Представительный

Сеансовый

Транспортный

Сетевой

Канальный

Физический

Физическая среда передачи данных

Приложение А

Приложение В

П1

П2

П3

П4

П5

П6

П7

Компьютер А

Компьютер А

Рис.2.1 Схема взаимодействия узлов сети

Средства протоколов одного уровня, установленные на разных узлах сети, не общаются между собой непосредственно. Их общение проходит через посредников, в качестве которых выступают средства протоколов нижележащих уровней. Например, средства сетевого уровня общаются через канальный и физический уровни.

**Физическая среда передачи данных** – это среда, по которой передается поток битов. Она может быть проводной и беспроводной. Примерами проводной среды являются витая пара или оптоволоконный кабель, беспроводные среды – радиоканал и инфракрасный световой поток.

**Физический уровень** формирует поток битов и передает его в среду передачи данных. На физическом уровне определяются характеристикиэлектрических сигналов (требования к фронтам импульсов, уровням напряжения или тока, тип кодирования и другие), а также стандартизуются типы разъёмов и назначение их каждого контакта (например, RJ-45). Функции физического уровня реализуются во всех устройствах, подключённых к сети. В компьютерах функции физического уровня выполняются сетевым адаптером.

**Канальный уровень** выполняет следующие функции:

* проверка доступности среды передачи данных;
* установление логического соединения между взаимодействующими узлами;
* согласование скоростей передатчика и приемника данных;
* формирование из потока битов собственных протокольных единиц данных – кадров (фреймов);
* преобразование адресов сетевого уровня в адреса канального уровня;
* продвижение кадров в пределах локальной сети;
* обнаружение и коррекция ошибок.

Стандарты IEEE 802.2 и IEEE 802.3 предусматривают разделение канального уровня на два подуровня – логический (LLC) и управление доступом к среде передачи данных (MAC). Через подуровень LLC сетевой протокол запрашивает у канального уровня нужную ему транспортную операцию с нужным качеством. По протоколу LLC кадры передаются либо дейтаграммным способом, либо с помощью процедур с установлением соединения между взаимодействующими узлами сети и восстановлением кадров путем их повторной передачи при наличии в них искажений.

Протоколы канального уровня обеспечивают доставку данных между двумя любыми узлами только в сетях с определенной типовойтопологией (общая шина, звезда, кольцо) с использованием MAC-адресации. Функции канального уровня реализуются сетевыми адаптерами компьютеров, а также коммутационным оборудованием – мостами и коммутаторами.

**Сетевой уровень** обеспечивает доставку данных узлами, которые находятся в разных сетях, каждая из которых может использовать собственную технологию канального уровня и способ адресации. На этом уровне используется IP-адресация, в которой каждый адрес состоит из двух частей – номера сети и номера узла в пределах этой сети. Функции сетевого уровня реализуются группой сетевых протоколов и специальными устройствами – маршрутизаторами, которые физически соединяют разные сети. Протокольными единицами данных сетевого уровня являются пакеты.

Основные функции сетевого уровня:

* сбор информации о топологии межсетевых соединений;
* определение маршрута доставки пакетов;
* фрагментация и дефрагментация пакетов;
* продвижение пакетов по определенному маршруту;
* формирование и передача сообщений об ошибках, возникающих при продвижении пакетов
* обеспечение возможности обмена данными между узлами, имеющими белые и серые IP-адреса.

**Транспортный уровень** обеспечивает верхним уровням стека TCP/IP передачу данных с той степенью надежности, которая им требуется. Основные функции транспортного уровня:

* мультиплексирование и демультиплексирование потоков данных и управление ими;
* разбивка сообщения сеансового уровня на пакеты (TCP –сегменты или UDP –дейтаграммы) и их нумерация;
* адресация прикладных процессов;
* контроль доставки пакетов;
* буферизация принимаемых пакетов;
* упорядочивание прибывающих пакетов;

Мультиплексирование – это процедура слияния нескольких потоков данных в один поток. Потоки данных из приложений в виде сообщений поступают на транспортный уровень и объединяются в единый поток дейтаграмм и сегментов, передаваемый на сетевой уровень. Обратная процедура называется демультиплексированием.

**Сеансовый уровень** обеспечивает управление диалогом объектов прикладного уровня:

* установление способа обмена сообщениями (дуплексный или полудуплексный);
* идентификации объектов диалога;
* поддержка адресов процессов и преобразование адресов процесса в адреса транспортного уровня;
* синхронизация обмена сообщениями;
* организация "контрольных точек" диалога**.**

**Представительный уровень** имеет дело с формой представления передаваемых по сети данных, не меняя при этом их содержания. При этом информация, пере-даваемая прикладным уровнем одного узла сети, всегда понятна прикладному уровню другого узла. Например, протоколы прикладных уровней могут преодолеть синтаксические различия в представлении данных или различия в кодах символов, например в кодах ASCIIи UNICODE.

На этом уровне может выполняться шифрование и дешифрование данных, благодаря которому секретность обмена данными обеспечивается сразу для всех прикладных служб. Примером такого протокола является протокол Secure Socket Layer (SSL), который обеспечивает защищенный обмен сообщениями для протоколов прикладного уровня стека TCP/IP.

**Прикладной уровень** осуществляет связь пользовательских приложений с сетью и представляет собой набор разнообразных протоколов, с помощью которых пользователи сети получают доступ к разделяемым ресурсам (файлы, гипертекстовые Web-страницы, принтеры) или организуют совместную работу, например с помощью протоколов электронной почты. Единица данных, с которой оперирует прикладной уровень, называется сообщением.

Три нижних уровня (физический, канальный и сетевой) являются cетезависимыми, то есть протоколы этих уровней тесно связаны с технической реализацией сети и используемым коммуникационным оборудованием. Например, переход на оборудование FDDI (волоконно-оптический интерфейс передачи данных) означает полную смену протоколов физического и канального уровней во всех узлах сети.

Три верхних уровня (прикладной, представительный и сеансовый) не зависят от технической реализации сети и ориентированы на приложения. На протоколы этих уровней не влияют изменения в топологии сети, замена оборудования или переход на другую сетевую технологию.

Транспортный уровень является промежуточным, он скрывает все детали функционирования нижних уровней от верхних. Это позволяет разрабатывать приложения, не зависящие от технических средств непосредственной транспортировки сообщений.

В ***модели сетевого программного обеспечения сети Интернет*** все верхние уровни обычно объединяют в один уровень – прикладной, поэтому такая модель включает пять уровней – физический, канальный, сетевой, транспортный и прикладной. Это связано со следующими причинами:

* используемая в Интернете модель взаимодействия «клиент-сервер» вытесняет сеансовый уровень, так как теперь явно определена сторона, которая является активной и управляет сеансом;
* форма представления данных в Интернете стандартизирована, поэтому отпадает необходимость использования функций представительного уровня.

2.3 Протоколы канального уровня

В современных сетях основным протоколом канального уровня является протокол Ethernet, поддерживающий четыре формата кадров, приведенных на рис.2.2, где все размеры полей приведены в байтах. В зависимости от способа указания типа инкапсулированных данных кадры делятся на две группы – с явным и неявным указанием типа данных. Тип инкапсулированных данных указывается в виде условного кода протокола, вложившего эти данные в кадр.

Исторически первым появился формат **Ethernet DIX (Ethernet II)**, использующий явное указание типа инкапсулированных данных.



Рис. 2.2

Назначение полей кадра Ethernet DIX:

* DA (destination address) – MAC адрес узла назначения;
* SA (source address) – MAC адрес узла отправителя;
* T (type) – условный код протокола верхнего уровня;
* **Data – передаваемые данные**;
* FSC – контрольная сумма.

Размер передаваемых данных в кадре Ethernet DIX – от 46 до 1500 байтов, если длина данных меньше 46 байт, то поле дополняется до минимального размера байтами заполнения.

Формат Ethernet DIX не учитывает наличие подуровня LLC в канальном уровне модели OSI, поэтому после появления стандартов IEEE 802.2 и IEEE 802.3, которые предусматривали разделение канального уровня на два подуровня (LLC и MAC), были разработаны другие форматы кадров. Такие форматы указывают тип передаваемых данных неявно, используя для этого начальные байты поля данных, а в поле T вместо кода протокола верхнего уровня хранят длину передаваемых данных.

В соответствии с протоколом LLC в кадр формата **802.3/LLC** включается заголовок подуровня LLC, содержащий поля DSAP и SSAP, в которые записываются однобайтовые коды протоколов верхнего уровня, получающего и отправляющего данные с помощью этого кадра. Если получателем и отправителем является один протокол, то эти коды совпадают. Например, согласно стандарта IEEE 802.2 для протокола IP поля DSAP и SSAP имеют значение 0х06, для STP – 0х42, для SNAP (Subnetwork Access Protocol, протокол доступа к подсети) – 0хАА. Отметим, что протокол SNAP используется для инкапсуляции дейтаграмм [IP](https://ru.wikipedia.org/wiki/IP) и запросов ARP в сетях IEEE 802.

Управляющее поле Control имеет размер 1 байт для ненумерованных кадров или 2 байта для информационных и управляющих кадров. Тип кадров определяется режимом, в котором работает подуровень LLC. Возможны три режима:

LLC1 – сервис без установления соединения и без подтверждения;

LLC2 – сервис c установлением соединения и с подтверждением;

LLC3 – сервис без установления соединения, но с подтверждением.

В режиме LLC2 в поле Control также указываются порядковые номера отправленных и успешно принятых кадров.

В кадр **Ethernet SNAP** по сравнению с форматом **802.3/LLC** дополнительно введены поля заголовка протокола SNAP - OUI и Type, а поля DSAP и SSAP имеют значение 0xAA. Поле Type состоит из 2-х байт и повторяет по формату и назначению поле Type кадра Ethernet DIX (т.е. в нем используются те же значения кодов протоколов). Поле OUI размером 3 байта определяет идентификатор организации, контролирующей коды протоколов в поле Type.

Формат **Raw 802.3/Novell 802.3** был разработан фирмой Novell для своего оборудования. Такой кадр не содержит информации о протоколах верхнего уровня, вложивших в него данные, т.к. здесь всегда используется только один собственный протокол IPX. Поле данных всегда начинается с двух байтов, содержащих 0хFFFF.

*Обратите внимание: при увеличении количества заголовков размер поля данных пропорционально уменьшается.*

Современное коммутационное оборудование может распознавать все форматы кадров, но основным форматом кадров в настоящее время является Ethernet DIX, использующий явное указание типа инкапсулированных данных.

К протоколам канального уровня относятся также протоколы **ARP** и **RARP**, определяющие соответствие IP- и MAC-адресов. Первый имеет Ethernet-код 0x0806 и определяет MAC-адрес по известному IP-адресу, а второй имеет Ethernet-код 0x8035 и определяет IP-адрес по известному MAC –адресу. На рис. 2.3. представлена структура пакета ARP, имеющего размер 28 байтов. Здесь обозначено:

* тип оборудования – номер канального протокола передачи данных (для Ethernet равен 0x0001);
* тип протокола – код сетевого протокола, для IPv4 будет иметь значение 0x0800;
* HA-Len – длина физического адреса в байтах, для Ethernet равен 0x06;
* PA-Len – длина логического адреса в байтах, для IPv4 равен 0x04;
* код операции – для запроса имеет значение 0x0001, для ответа – 0x0002;
* MAC-адрес и IP-адрес отправителя;
* MAC-адрес (при запросе заполняется нулями) и IP-адрес адресата.



Рис. 2.3

На рис.2.4 показана инкапсуляция протоколов канального уровня.

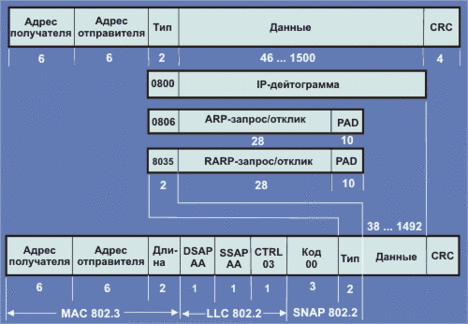


Рис.2.4

2.4 Протоколы сетевого уровня

Основным протоколом сетевого уровня в стеке TCP/IP является протокол IP, основным назначением которого является продвижение пакетов между сетями. Такими сетями могут быть подсети, образующие составную локальную сеть, или сети, подключенные к Интернет. Кроме того, этот протокол обеспечивает интерфейсы с транспортным и канальным уровнями. Продвижение пакетов IP-протоколом осуществляется дейтаграммным способом, т.е. без установления предварительного соединения. Структура IP-пакета приведена на рис.2.5.



Рис. 2.5.

Здесь обозначено:

* номер версии – номер версии протокола IP;
* длина заголовка, содержит количество 32 – битных слов;
* тип сервиса – используется для дифференцированного обслуживания пакетов, изменением значения этого поля можно управлять приоритетом обработки пакетов в маршрутизаторах;
* общая длина пакета с учетом заголовка и полей данных;
* идентификатор IP пакета – уникальный номер пакета, используемый для распознавания пакетов при сборке в случае их фрагментации; все фрагменты одного пакета имеют одинаковое значение этого поля;
* флаги содержат признаки, связанные с фрагментацией;
* смещение фрагмента – задает смещение поля данных этого фрагмента относительно начала поля данных исходного пакета;
* время жизни (TTL) – максимальное число транзитных узлов, которое разрешено пройти пакету; при прохождении каждого узла значение TTL уменьшается на единицу и при достижении нулевого значения пакет уничтожается;
* протокол верхнего уровня – указывает какому протоколу принадлежат данные, переносимые пакетом (6 – TCP, 17 – UDP, 1 – ICMP);
* контрольная сумма заголовка – контрольная сумма, рассчитанная только по заголовку;
* IP адрес источника;
* IP адрес назначения.

Протокол IP может использоваться совместно с разными технологиями канального уровня, каждая из которых характеризуется собственным значением максимального размера передаваемого кадра MTU ( Ethernet – 1500 байтов, Frame Relay – 4080 байтов, X.25 – 128 байтов и другими). Для этого предусмотрена возможность разбиения пакета на фрагменты определенного размера. Такая процедура называется фрагментацией IP-пакета. Для обеспечения возможности сборки фрагментов на узле-получателе параметры фрагментации передаются в полях заголовка пакета «Идентификатор IP пакета», «Флаги» и «Смещение фрагмента».

При этом пакету присваивается уникальный идентификатор, значение которого для каждого фрагмента записывается в поле «Идентификатор IP пакета». Флаг MF является признаком фрагментации пакета: значение 1 соответствует фрагментированному пакету, значение 0 – нефрагментированному пакету. Флаг DF используется для запрета фрагментации (значение 1 запрещает маршрутизатору фрагментацию, значение 0 – разрешает). Поле «Смещение фрагмента» задает смещение фрагмента в байтах относительно начала исходного нефрагментированного фрагмента и должно быть кратно 8 байт.

Если пакет имеет значение DF=1, но не может быть доставлен адресату без фрагментации, то он уничтожается, а отправителю направляется соответствующее диагностическое сообщение.

К протоколам сетевого уровня относятся также протокол межсетевых управляющих сообщений ICMP, который предназначен для передачи сообщений о состоянии узла назначения IP-пакета, а также ошибках, возникающих во время прохождения IP-пакета по сети. Маршрутизатор, который обнаруживает ошибку, выполняет следующие действия:

* формирует ICMP-пакет о причине ошибки;
* формирует новый IP-пакет и вкладывает в него свой ICMP-пакет;
* направляет сформированный IP-пакет отправителю пакета, в котором произошла ошибка;
* ошибочный IP-пакет отбрасывается.

Структура ICMP-пакета приведена на рис.2.6 и содержит заголовок и поля данных. Заголовок включает обязательную и необязательную части по 4 байта каждая. Обязательная часть имеет три поля:

* тип – указывает основную причину формирования сообщения;
* код – уточняет причину формирования сообщения;
* контрольная сумма сообщения.

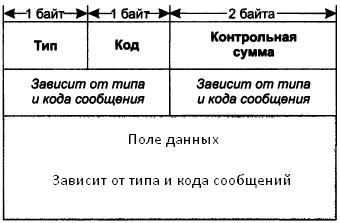


Рис. 2.6

Интерпретация необязательной части заголовка и поля данных зависят от типа сообщения. Например, для сообщений типа 8 (эхо-запрос) и 0 (эхо-ответ) необязательная часть заголовка может содержать поля «Идентификатор» и «Номер», которые используются отправителем для проверки соответствия между запросом и ответом, а поле данных для сообщений типа 3 (адресат недоступен), 5 (перенаправление маршрута) и 11 (истечение TTL) содержит заголовок и первые 8 байтов поля данных IP-пакета, вызвавшего ошибку, чтобы отправитель мог идентифицировать этот пакет. Поле данных имеет переменную длину и содержит данные, которые необходимо вернуть отправителю.

На рис. 2.7 приведены основные типы ICMP-сообщений:

* *Запрос/ответ эхо - повтора*. Запрос эхо - повтора может быть передан программному обеспечению протокола ICMP любого компьютера. В ответ на запрос эхо – повтора программное обеспечение ICMP должно послать ответ эхо – повтора ICMP, который содержит те же данные, что и запрос. Этот тип сообщений используется программами **ping** и **tracert.**
* *Узел назначения недостижим*. Обнаружив, что дейтаграмма не может быть доставлена в место назначения, маршрутизатор отправляет узлу, на котором она была создана, сообщение о том, что получатель недоступен. В сообщении указано, является ли недоступным конкретный узел назначения или недоступна сеть, к которой подключен этот узел.
* *Истечение времени дейтаграмм*. Сообщение формируется, если TTL=0.



Рис. 2.7

2.5 Протоколы транспортного уровня

Основное отличие протоколов транспортного уровня от протоколов сетевого и канального уровней состоит в том, что они организуют взаимодействие между приложениями, а не между узлами сети. Узлы идентифицируются по их адресам, а приложения – по номеру порта.

Для организации удаленного взаимодействия каждое сетевое приложение создает специальный объект – сокет, основными параметрами которого являются IP-адрес узла и номер порта. Строго говоря, номер порта сопоставляется не приложению, а процессу, который создается операционной системой при запуске этого приложения. Если процесс имеет несколько потоков, то для каждого потока выделяется собственный порт. Например, в браузере для каждой открытой страницы создается отдельный порт, поэтому каждая страница может установить соединение с любым web-сервером.

Номера портов в диапазоне от 0 до 1023 являются уникальными и задаются международной организацией IANA (Internet Assigned Numbers Authority), распределяющей адреса в Интернете. Такие номера называются *хорошо* *известными* и выделяются стандартным сетевым протоколам и службам. Номера в диапазоне от 1024 до 49150 доступны для всех разработчиков приложений и, при необходимости, могут быть закреплены за приложением после регистрации в IANA. Номера в диапазоне от 49151 до 65535 обычно выделяются операционной системой динамически для клиентских приложений.

Основными протоколами транспортного уровня являются UDP и TCP. Каждый из протоколов имеет собственное пространство номеров портов. Примерами TCP-портов являются порты протоколов FTP (20, 21), HTTP (80), HTTPS (443), SMTP (25), UDP-портов – протоколы RIP (520), TFTP (69). Некоторые приложения могут взаимодействовать по выбору как с протоколом UDP, так и с TCP (например, служба DNS использует TCP-порт 53 и UDP-порт 53).

**Протокол UDP** обеспечивает доставку дейтаграмм “по возможности”, не используя средства контроля их доставки и предварительного установления соединения. Структура дейтаграммы приведена на рис. 2.8 и содержит заголовок и поле данных.

Заголовок включает следующие поля:

* порт источника – номер UDP-порта отправителя, на этот порт будет посылаться ответ;
* порт назначения – номер UDP-порта получателя; например, при обращении к веб-серверу по умолчанию будет иметь значение 80 (протокол HTTP);
* длина UDP – длина дейтаграммы; значение может находиться в диапазоне от 8 до 65535 байтов;
* контрольная сумма UDP – контрольная сумма поля данных.

Порт источника (2байта)

Порт назначения (2байта)

Длина UDP (2байта)

Контрольная сумма UDP

Данные

Рис. 2.8

Поле данных содержит сообщение, передаваемое с верхнего уровня. Например, при ответе веб-сервера здесь будет находиться запрошенная браузером веб-страница. При прохождении пакета каждый маршрутизатор проводит расчет контрольной суммы поля данных и при обнаружении ошибки этот пакет отбрасывается без запроса о повторной отправке.

**Протокол TCP** обеспечивает надежный логический канал доставки пакетов между прикладными процессами. Сообщение, поступающее на вход модуля TCP с прикладного уровня, рассматривается как неструктурированный поток байтов. Этот поток буферизуется и делится на сегменты определенного размера, каждый из которых снабжается заголовком и передается на сетевой уровень.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Бит** | **0 — 3** | **4 — 9** | **10 — 15** | **16 — 31** |
| **0** | Порт источника, **Source Port** | | | Порт назначения, **Destination Port** |
| **32** | Порядковый номер, **Sequence Number (SN)** | | | |
| **64** | Номер подтверждения, **Acknowledgment Number (ACK\_SN)** | | | |
| **96** | Длина заголовка | Зарезервировано | Флаги | Размер окна |
| **128** | Контрольная сумма | | | Указатель важности |
| **160** | **Опции (необязательное)** | | |  |
|  | Данные | | | |

Рис. 2.9

На рис. 2.9 приведена структура TCP-сегмента, который имеет следующие поля заголовка:

* номер порта источника (Source Port, 2 байта) – номер порта, который идентифицирует приложение клиента, отправляющего пакеты; ответные данные передаются клиенту на основании этого номера;
* номер порта назначения (Destination Port, 2 байта) – номер порта, который принимает данные;
* порядковый номер (Sequence Number, 4 байта) – счетчик переданных байтов, каждый переданный байт полезных данных увеличивает это значение на 1;
* номер подтверждения (Acknowledgment Number, 4 байта) – начальный номер байта следующей ожидаемой порции данных;
* длина заголовка – число 32-разрядных слов в заголовке;
* резерв – разряды поля, установленные в ноль;
* флаги – 6 разрядов, определяющих тип сегмента, например, сегмент установки соединения (SYN) и завершения сеанса (FIN), сегмент подтверждения принятых данных (ACK), срочного сообщения (URG), PSH - протолкнуть данные и RST - оборвать соединение;
* размер окна (Window Size) – число байтов, передаваемых отправителем, начиная с байта, номер которого указан в поле «Номер подтверждения»;
* контрольная сумма (Checksum) – значение контрольной суммы заголовка и поля данных;
* указатель важности (Urgent pointer) – указывает на конец данных, которые необходимо срочно принять, несмотря на переполнение буфера;
* опции (Option) – поле не обязательное, используется для согласования максимального размера TCP сегмента;
* данные (Data) – сообщение протокола верхнего уровня.

Установление логического соединения осуществляется путем обмена служебными пакетами, которые отличаются от обычных пакетов данных наличием специальных флагов, и проводится в три этапа:

* клиент посылает на сервер запрос синхронизации с флагом SYN;
* сервер получает от операционной системы необходимые ресурсы (буферы, таймеры, счетчики), подтверждает запрос синхронизации клиента флагом ACK и задает свои параметры синхронизации флагом SYN; в случае неудачи сервер посылает клиенту сегмент с флагом RST;
* клиент посылает на сервер подтверждение об установлении соединения флагом ACK.

Процедуру установления логического соединения часто называют тройным рукопожатием. TCP-соединение обеспечивает дуплексный обмен данными, т.е. каждая сторона обмена может передавать и принимать данные одновременно. Системные ресурсы закрепляются за соединением с момента установления и до момента его разрыва.

Завершение соединения может инициировать любая из сторон. Эта процесс реализуется служебными пакетами и имеет четыре этапа:

* узел-отправитель, инициализирующий завершение, посылает сегмент с флагом FIN;
* узел-получатель подтверждает конец передачи флагом ACK;
* узел-получатель также посылает сигнал конца передачи FIN;
* узел-отправитель подтверждает получение информации флагом ACK.

*Обратите внимание: все служебные пакеты имеют незаполненное поле данных !*

При инициализации логического соединения (если установлен флаг SYN) поле SN содержит начальный порядковый номер (ISN, Initial Sequence Number), значение которого генерируется случайным образом в диапазоне от 0 до 232-1. Первый байт полезных данных в установленном соединении будет иметь номер ISN+1. В противном случае (если SYN не установлен) первый байт данных, передаваемый в пакете, имеет номер ISN.

Если установлен флаг ACK, то поле ACK\_SN содержит порядковый номер байта, который отправитель данного сегмента желает получить. Это значит, что все предыдущие байты с номерами от ISN+1 до ACK\_SN-1 включительно были успешно получены.

Каждая сторона имеет собственные счетчики переданных и полученных данных, поэтому значение SN одной из сторон всегда соответствует значению ACK\_SN другой стороны.

Ниже приведен пример процесса установления соединения по протоколу TCP. На рис. 2.10 приведена структура запроса TCP-соединения между клиентом (IP-адрес –192.168.31.143, порт 52323) и FTP-сервером (IP-адрес –192.168.31.139, порт 21). Поля SNк и ACK\_SNк имеют нулевые значения. Номер порта сервера является стандартным и определяется протоколом FTP, номер порта клиента выбирается случайно из диапазона 1024-65535. Номера стандартных портов можно посмотреть в файле *%Windir%\System32\drivers\etc\services*.

В ответ на этот пакет сервер, если он готов ответить, посылает пакет с установленными битами SYN, ACK, собственным значением SNс=0 и подтверждением получения пакета от клиента, увеличивая значение ACK\_ SNк на единицу, т.е. ACK\_ SNс = ACK\_ SNк + 1 = 1, как показано на рис.2.11.

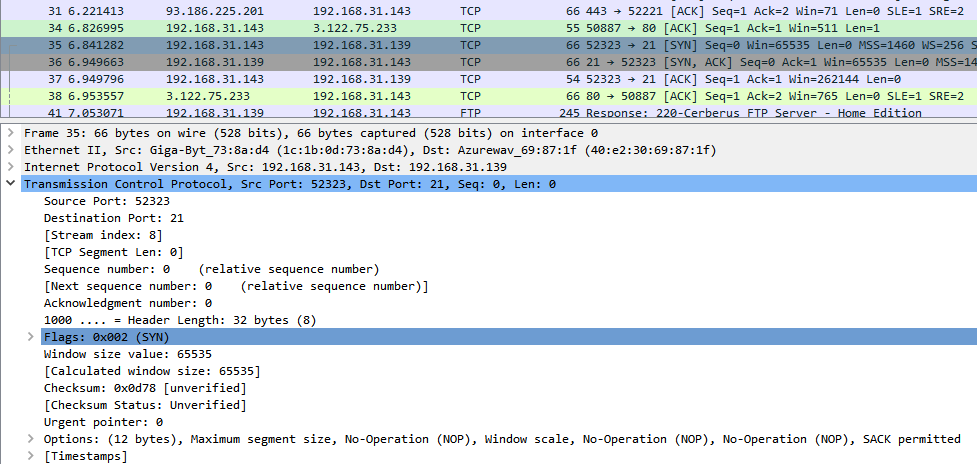


Рис. 2.10.

Далее клиент высылает пакет с установленным битом ACK и значениями SNк = ACK\_SNс =1, ACK\_SNк = SNс+1 = 1, что подтверждается рис.2.12. С этого момента соединение считается установленным и дальнейшие пакеты будут передавать уже полезную нагрузку – данные протокола FTP.

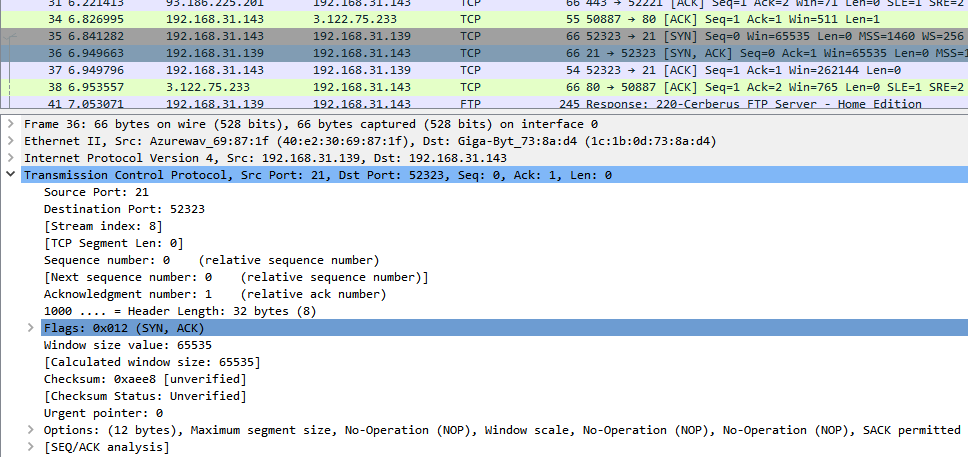


Рис. 2.11.

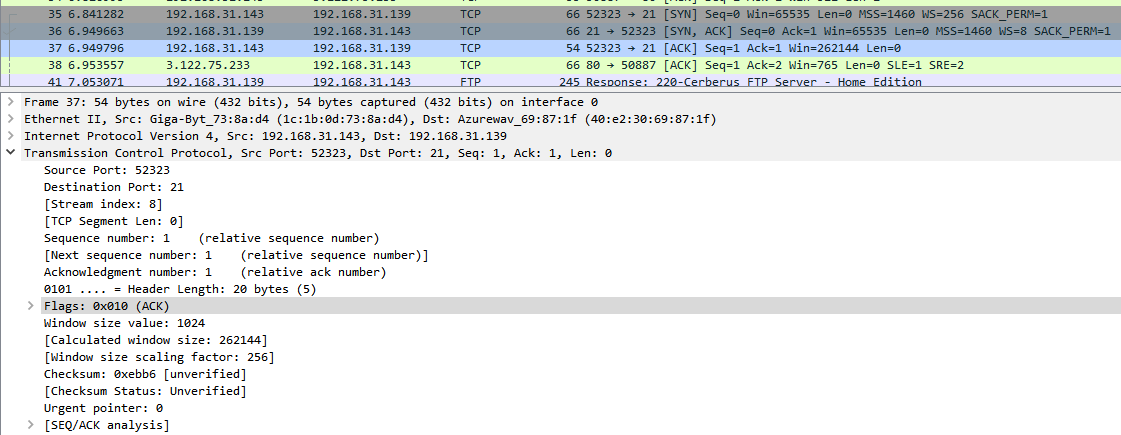


Рис. 2.12.

1. Задание на лабораторную работу
2. Запустить перехват пакетов в WireShark.
3. Определить с помощью утилиты **ping** доступность заданных узлов в соответствии с вариантом задания (таблица 3.1), выполнить трассировку к одному из узлов.
4. С помощью браузера просмотреть несколько страниц на сайте **nstu.ru**; подключиться к системе Moodle и просмотреть файлы с календарным планом выполнения лабораторных работ и рейтинговой системой по курсу «Сетевые информационные технологии».
5. С помощью клиента WinSCP подключиться по протоколу FTP к серверу fpm2.ami.nstu.ru и выполнить копирование в Ваш домашний каталог текстового файла согласно варианта из таблицы 3.1. Архив с файлами можно скачать из системы Moodle.
6. Остановить перехват пакетов и сохранить результаты в файл с расширением .pcapng.
7. С помощью WireShark определить внутреннюю структуру кадров и пакетов, передаваемых по сети; сравнить ее со структурами, описанными в протоколах Ethernet, IP и TCP.
8. Определить последовательность прохождения запросов, реализующих алгоритм трассировки одного из заданных в таблице 3.1 узлов.
9. Восстановить сеанс обмена данными по протоколу HTTP между браузером и сервером при выполнении п.3.
10. Восстановить сеанс обмена данными по протоколу FTP при выполнении п.4, найти перехваченные логин и пароль, а также восстановить содержимое переданного файла.
11. Определить последовательность прохождения запросов, реализующих один из протоколов в соответствии с вариантом из таблицы 3.1 (ICMP, DNS или ARP). Построить схему работы протокола и формат пакетов.
12. Найти в перехваченном трафике пакеты, передаваемые по протоколу в соответствии с вариантом задания (см. таблицу 3.1), определить назначение данного протокола.
13. Найти в перехваченном трафике широковещательные запросы по протоколам DHCP, ARP и ответы на них. Определить структуру передаваемых по этим протоколам кадров.
14. Определить значение поля «Тип данных» для кадра Ethernet при передаче пакетов IP, ARP, ICMP, DNS, DHCP.
15. Построить статистику по используемым за время сеанса протоколам.
16. Изучить процесс установления соединения по протоколу TCP.

Таблица 3.1

|  |  |
| --- | --- |
| Вариант | Номер пункта и задание |
| 1 | 2. Утилита ping: opennet.ru, kremlin.ru, yandex.ru, google.com, rambler.ru, pochta.ru |
| 4. test.txt |
| 10. ICMP |
| 11. HTTP |
| 2 | 2. Утилита ping: iforum.pro, meteoinfo.ru, mail.ru, drom.ru, ngs.ru, habr.com |
| 4. test1.txt |
| 10. DNS |
| 11. TCP |
| 3 | 2. Утилита ping: sklad-service.ru, eye.moof.ru, gmail.com, wiw.ru, luminator.ru,  hotlog.ru. |
| 4. test2.txt |
| 10. ARP |
| 11. STP |
| 4 | 2. Утилита ping: sportmaster.ru, nem.ru, nightwish.ru ,email.com, akkords.ru , mtw.ru |
| 4. test3.txt |
| 10. DNS |
| 11. SSH |
| 5 | 2. Утилита ping: mts.ru, pandora.com, www.intuit.ru , formoza.ru, iz.ru, cars.ru |
| 4. test4.txt |
| 10. ICMP |
| 11. TLS |
| 6 | 2. Утилита ping: moyproekt.ru, stockmap.ru, flagma.ru, zurbazar.ru, itcsme.ru,  www.ofnet.ru |
| 4. test5.txt |
| 10. ARP |
| 11. HTTPS |
| 7 | 2. Утилита ping: asutp.ru, ohranatruda.ru,volgaweb.ru, toolsmart.ru, sviazist.nnov.ru, dogma.su , sec.ru |
| 4. test6.txt |
| 10. ICMP |
| 11. SSH |
| 8 | 2. Утилита ping: eprussia.ru, youtube.com, wikipedia.org, rutracker.org , ucoz.ru,  twitter.com |
| 4. test7.txt |
| 10. DNS |
| 11. STP |
| 9 | 2. Утилита ping: advmaker.ru, zr.ru, hh.ru, ozon.ru, rbc.ru, ifolder.ru, photofile.ru |
| 4. test8.txt |
| 10. ARP4 |
| 11. DNS |
| 10 | 2. Утилита ping: weather.com, topwar.ru , novosib.express-shina.ru, webhamster.ru, taiga.info, sec.ru |
| 4. test9.txt |
| 10. DNS |
| 11. ICMP |

4. Контрольные вопросы

1. Cтруктура кадра Ethernet DIX.
2. Типы кадров Ethernet.
3. Алгоритм распознавания типа кадра Ethernet.
4. Cтруктура кадра ARP.
5. Алгоритм работы протокола ARP.
6. Назначение и основные свойства протокола IP.
7. Что характеризует понятие TTL для IP-пакета.
8. Почему возникает необходимость фрагментации IP-пакета
9. Назначение и основные свойства протокола ICMP.
10. Этапы установления соединения в протоколе TCP.

# Лабораторная работа № 4. Протокол ПЕРЕДАЧИ ФАЙЛОВ ftp

1. Цель работы

Целью работы является изучение принципов организации взаимодействия прикладных программ с помощью протокола передачи данных FTP и приобретение практических навыков создания клиентских приложений, использующих протокол FTP.

2. Методические указания

2.1 Общие сведения

FTP (File Transfer Protocol) является базовым протоколом передачи данных в сети Internet (первая версия протокола появилась 1971 г., последняя – в 1985 г.) и обеспечивает надежный доступ к файловым системам удаленных узлов сети. Протокол позволяет передавать текстовые и двоичные файлы между различными файловыми системами. Надежность обеспечивается использованием TCP-соединения, обмен основан на технологии «клиент-сервер».

FTP передает данные открытым текстом, не обеспечивая их защиту от несанкционированного доступа. Большинство FTP-серверов имеют функцию аутентификации клиента, но при этом пароль и идентификатор пользователя передаются клиентом также открытым текстом.

На рис.2.1 показана модель работы протокола FTP. Этот протокол, в отличие от протоколов передачи электронной почты, использует два параллельных TCP-соединения: канал управления и канал данных. Канал управления служит для передачи команд клиента серверу (имя пользователя, пароль, команды управления удаленной и локальной файловыми системами и т.д.) и ответов сервера клиенту.  Команды FTP вводятся через интерфейс пользователя, генерируются интерпретатором протокола пользователя и передаются на сервер.

Канал данных предназначен для обеспечения двухстороннего обмена данными между сервером и клиентом и может быть использован как для приема, так и для передачи данных. При передаче данных между компьютерами канал управления должен быть открыт, в случае его закрытия передача данных прекращается.

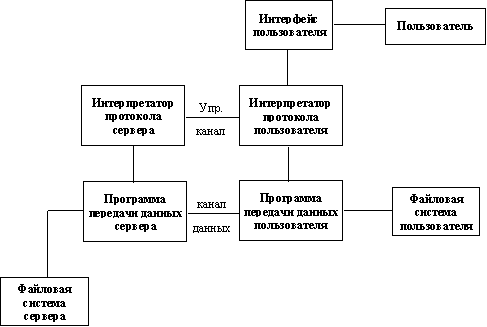


Рис. 2.1 Модель работы протокола FTP

Алгоритм работы протокола FTP состоит в следующем:

1. FTP-сервер использует для канала управления TCP-порт 21.

2. После того, как устанавливается управляющее соединение модуля «Интерпретатор протокола пользователя» с модулем сервера «Интерпретатор протокола сервера», клиент может отправлять на сервер команды. FTP-команды определяют параметры соединения передачи данных: роль участников соединения (активный или пассивный), порты соединения для модулей «Программа передачи данных пользователя» и «Программа передачи данных сервера», тип передачи, тип передаваемых данных, структуру данных и управляющие директивы, обозначающие действия, которые пользователь хочет выполнить (например, получить файл с сервера, загрузить файл на сервер и т.д.).

3. После того, как установлены все параметры канала передачи данных, один из участников соединения (пассивный модуль) становится в режим ожидания открытия соединения на определенный порт. После этого другая сторона (активный модуль) открывает соединение на указанный порт и начинается обмен данными.

4. После окончания передачи данных соединение между «Программой передачи данных сервера» и «Программой передачи данных пользователя» закрывается, но управляющее соединение «Интерпретатора протокола сервера» и «Интерпретатора протокола пользователя» остается открытым. Пользователь, не закрывая сессии FTP, может еще раз открыть канал передачи данных.

Современные FTP-серверы могут иметь функцию FXP (File eXchange Protocol — протокол обмена файлами), которая обеспечивает возможность передачи файлов между двумя FTP-серверами напрямую без передачи их на локальный компьютер. При этом время передачи файлов будет зависеть только от скорости соединения между двумя удаленными FTP-серверами. Режим FXP предполагает другую модель работы протокола FTP, когда каналы управления организуются для каждого сервера, а канал данных работает напрямую между серверами (рис.2.2).

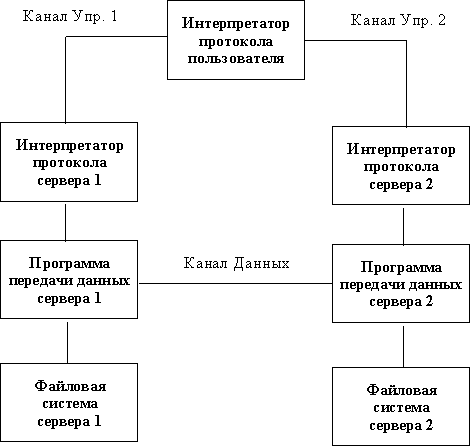


Рис. 2.2 Модель работы протокола FTP в режиме FXP

2.2 Управление процессом обмена данными

Основу передачи данных FTP составляет механизм установления соединения между сервером т клиентом. Каждый участник FTP-соединения должен поддерживать порт передачи данных по умолчанию, когда «Программа передачи данных пользователя» использует тот же порт, что и для передачи команд, а «Программа передачи данных сервера» использует порт (U-1), где U – управляющий порт. Однако для ускорения процессов обмена данными участники соединения часто используют другие порты передачи данных.

Передача данных может вестись в активном или пассивном режиме. Установление соединений при активном режиме происходит следующим образом (рис.2.3):

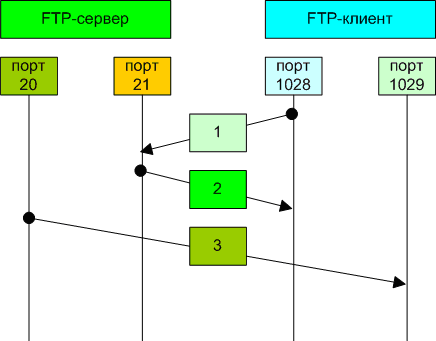


Рис.2.3 Схема установления соединений в активном режиме

1. Клиент создает сокет для канала управления, выделяя для него любой свободный порт K (K >1024), и направляет серверу на стандартный порт 21 команду PORT, в которой указывает свой IP-адрес и выбранный для передачи данных второй порт L из динамического диапазона 1024 – 65535.
2. В случае успешного получения и обработки команды PORT сервер по управляющему каналу отправляет клиенту положительный отклик на порт K и подключается к порту клиента L, используя со своей стороны номер TCP-порта 20 для передачи данных.

В приведенном на рис. 2.3 примере K=1028, L=1029.

Алгоритм установления соединений при пассивном режиме (рис. 2.4):

1. Клиент создает сокет для канала управления, выделяя для него любой свободный порт K (K >1024) и отправляет по нему на стандартный порт 21 команду PASV, указывающую серверу о намерении клиента работать в пассивном режиме.
2. 2. В случае успешного получения и обработки команды PASV сервер создает сокет для канала передачи данных, выделяя для него порт L из динамического диапазона 1024 – 65535, переводит этот сокет в режим ожидания соединения и отправляет клиенту ответ, в котором указываются параметры соединения (IP-адрес сервера и номер порта L).
3. Клиент, получив положительный отклик с параметрами соединения, создает сокет с использованием порта M из динамического диапазона 1024 – 65535 и пытается соединиться с портом L сервера.
4. В случае успешного установления соединения клиент начинает процесс передачи данных, используя при этом управляющий канал для отправки FTP-команд.

В приведенном на рис. 2.4 примере K=1028, L=2026, M=1029.

В обоих режимах одновременно с передачей данных по управляющему каналу могут передаваться уведомления о получении данных между интерпретаторами протокола сервера и клиента. Управляющее соединение должно быть открыто до тех пор, пока по каналу обмена данными идет передача.

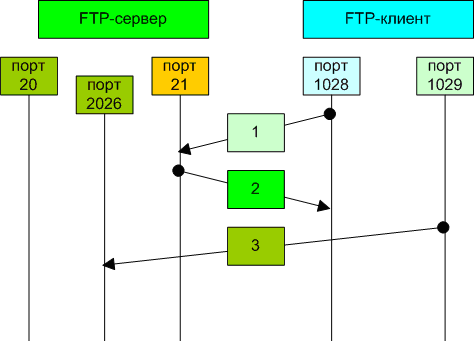


Рис. 2.4 Схема установления соединений в пассивном режиме

Сервер самостоятельно закрывает канал передачи данных в следующих случаях:

* сервер закончил передачу данных в формате, который требует закрытия соединения;
* сервер получил от пользователя команду «прервать соединение»;
* пользователь изменил параметры порта передачи данных;
* было закрыто управляющее соединение;
* возникли ошибки, при которых невозможно возобновить передачу данных.

Главное отличие между активным и пассивным режимами FTP  определяется тем, какая сторона открывает соединение для передачи данных. В активном режиме это соединение открывает сервер, в пассивном – клиент. Если клиент не может принять входящее TCP-соединение (например, он находится в локальной сети или его доступ к FTP -серверу осуществляется через proxy-сервер), то обмен данными возможен только в пассивном режиме.

2.3 Команды протокола FTP

Команды FTP можно разделить на три группы.

1. *Команды управления доступом* к системеобеспечивают аутентификацию пользователя в системе, выход из нее и настройку некоторых текущих параметров соединения.

2. *Команды управления потоком данных*устанавливают параметры передачи данных. Все параметры, описываемые этими командами, имеют значение по умолчанию, поэтому команды управления потоком используются только тогда, когда необходимо изменить значение параметров передачи, используемых по умолчанию. Команды управления потоком могут подаваться в любом порядке, но все они должны предшествовать командам FTP-сервиса.

3. *Команды FTP-сервиса*определяют действия, которые необходимо произвести с указанными файлами. Как правило, аргументами команд этой группы являются имена файлов.

Полный перечень команд протокола FTP содержится в спецификациях RFC765 и RFC959. Существует много реализаций FTP-клиентов для разных операционных систем, отличающихся друг от друга набором поддерживаемых команд, который можно посмотреть командами **help** или “**?**”.

Все команды отправляются интерпретатором протокола пользователя в текстовом виде построчно. Каждая командная строка заканчивается символами CRLF (перевод строки и возврат каретки). Имя команды всегда отделяется от аргумента символом пробела. Интерпретатор протокола сервера возвращает код обработки каждой команды, состоящий из трех цифр, за которым следует символ пробела и текст пояснения.

Ниже будут рассмотрены наиболее часто применяемые команды.

2.3.1. Команды управления сеансом

|  |  |  |
| --- | --- | --- |
| **RFC** | **FTP-клиент Windows** | **Назначение** |
| user | user | Открывает сессию между клиентом и сервером. Аргументом команды является имя (идентификатор) пользователя для работы с файловой системой. Эта команда может подаваться не только в начале, но и в середине сессии, если пользователь желает изменить идентификатор, от имени которого будут проводиться действия. При этом все переменные, относящиеся к старому идентификатору, освобождаются. Если во время изменения идентификатора происходит обмен данными, обмен завершается со старым идентификатором пользователя. |
| pass | отсутствует | Ввод пароля пользователя, который передается в качестве аргумента. FTP-клиент Windows делает запрос пароль самостоятельно. |
| pasv | отсутствует | Переводит сервер в пассивный режим работы; ответом на данную команду является строка, содержащая адрес и порт узла, находящегося в режиме ожидания соединения. |
| open | open | Подключает клиента к указанному серверу; аргументами являются доменное имя или IP-адрес, а также номер порта (по умолчанию используется стандартный порт 21). |
| close | close | Завершает сеанс, открытый командой OPEN |
| quit | quit или bye | Закрывает управляющий канал. Если в момент подачи команды происходит передача данных, канал закрывается после окончания передачи данных. |
| abor | disconnect | Прерывает выполнение предшествующей команды (например, передачу файла) и закрывает канал передачи данных. |
| quote | quote | Отправка произвольной команды серверу FTP. |
| ! | ! | Временный переход из среды FTP в командную строку Windows (cmd.exe), для возврата в FTP используется команда EXIT |
|  | status | Вывод текущего состояния сеанса - к какому серверу выполнено подключение, тип передаваемых данных, состояние режимов, изменяемых командами FTP |

2.3.2. Команды управления потоком данных

|  |  |  |
| --- | --- | --- |
| **RFC** | **FTP-клиент Windows** | **Назначение** |
| port | отсутствует | Команда указывает серверу адрес и порт, которые будут использоваться клиентом для прослушивания соединения. Синтаксис команды:  PORT h1,h2,h3,h4,p1,p2  Аргументами команды являются 32-битный IP адрес и 16-битный номер порта соединения. Эти значения разбиты на шесть 8-битных полей и представлены в десятичном виде, где hN – байты адреса (от старшего к младшему), а pN – байты порта (от старшего к младшему). |
| type | ascii, binary | Определяет тип передаваемых данных (текстовый или двоичный). |
| mode | отсутствует | Определяет режим передаваемых данных (потоковый, блочный). |

2.3.3. Команды FTP-сервиса

|  |  |  |
| --- | --- | --- |
| **RFC** | **FTP-клиент Windows** | **Назначение** |
| retr | get, mget | Получает с сервера копию файла, заданного параметром этой команды; командой mget можно получить несколько файлов. |
| stor | put, send, mput | Передает на сервер один (команды put и send) или несколько файлов (команда mput). Если такой файл уже существует, то он будет перезаписан. |
| rnfr, rnto | rename | Переименование файла на сервере |
| dele | delete | Удаление указанного файла на сервере |
| mkd | mkdir | Создание указанного в аргументе каталога |
| rmd | rmdir | Удаление указанного в аргументе каталога |
| list | Ls, dir, mls, mdir | Получение списка файлов в указанном каталоге, mls и mdir выводят содержимое из нескольких каталогов |
| cwd | cd | Изменение текущего каталога в файловой системе сервера |
|  | lcd | Изменение текущего каталога в локальной файловой системе. |
| pwd | pwd | Вывод имени текущего каталога на сервере |

Подсказку по конкретной команде можно получить, введя знак вопроса или **help** и имя команды, например: **? open** или **help open**

Некоторые FTP–серверы разрешают подключение незарегистрированных пользователей. В этом случае в качестве имени пользователя необходимо использовать имя “anonymous”, в качестве пароля – адрес своей электронной почты.

Из списка перечисленных команд видно, что стандартный FTP–клиент в ОС Windows не поддерживает команду переключения в пассивный режим обмена с сервером FTP (**pasv**), что делает его пригодным только для использования в локальных сетях, где возможно прямое TCP-соединение между клиентом и сервером. Несмотря на это, некоторые FTP–клиенты могут проводить обмен данными в этом режиме, для чего можно применить команду **quote pasv**

Каждая полученная от клиента команда сопровождается ответом сервера. В таблице 1 приведены основные коды ответов на команды.

Таблица 1

|  |  |
| --- | --- |
| **Код** | **Описание** |
| 110 | Комментарий |
| 125 | Канал открыт, обмен данными начат |
| 150 | Статус файла правилен, подготавливается открытие канала |
| 200 | Команда корректна |
| 220 | Слишком много подключений к FTP-серверу |
| 221 | Успешное завершение по команде quit |
| 225 | Канал сформирован, но информационный обмен отсутствует |
| 226 | Закрытие канала, обмен завершен успешно |
| 230 | Пользователь идентифицирован, продолжайте |
| 250 | Запрос прошел успешно |
| 331 | Имя пользователя корректно, нужен пароль |
| 332 | Для входа в систему необходима аутентификация |
| 421 | Процедура невозможна, канал закрывается |
| 425 | Открытие информационного канала невозможно |
| 426 | Канал закрыт, обмен прерван |
| 450 | Запрошенная функция не реализована, файл недоступен |
| 451 | Локальная ошибка, операция прервана |
| 452 | Ошибка при записи файла (недостаточно места) |
| 500 | Синтаксическая ошибка, команда не может быть интерпретирована |
| 501 | Синтаксическая ошибка (неверный параметр или аргумент) |
| 503 | Неудачная последовательность команд |
| 504 | Команда не применима для такого параметра |

Примеры работы FTP-клиента в командном режиме приведены в таблице 2:

Таблица 2

|  |  |
| --- | --- |
| **Команда** | **Назначение** |
| open fpm2.ami.nstu.ru | Подключение к серверу fpm2.ami.nstu.ru по протоколу FTP |
| pwd | Вывод имени текущего каталога на сервере |
| get myfile.txt | Копировать файл myfile.txt из текущего каталога сервера в текущий каталог локального компьютера |
| put myfile.txt | Копировать файл myfile.txt из текущего каталога локального компьютера в текущий каталог сервера |
| mkdir mydir | Создать каталог mydir в текущем каталоге сервера |
| quote pasv | Перевести сервер в пассивный режим работы |

1. Задание на лабораторную работу

1. Ознакомиться с теоретическим материалом. Результаты выполнения всех последующих заданий занести в отчет.

2. Создать на локальном компьютере каталог Lab3.

3. Запустите анализатор Wireshark для перехвата FTP–трафика.

4. С помощью браузера подключиться по протоколу FTP к серверам fpm2.ami.nstu.ru и ftp.nstu.ru. Изучить способы навигации по файловой системе сервера, скачать из домашнего каталога на сервере fpm2.ami.nstu.ru в файл, имя которого задано в таблице 2.5 лабораторной работы № 2. По захваченным данным построить диаграмму потоков.

5. Запустить стандартного ftp-клиента Windows и с его помощью выполнить действия:

* вывести список поддерживаемых команд;
* подключиться к серверу fpm2.ami.nstu.ru;
* создать на локальном компьютере в каталоге Lab3 подкаталоги Lab3\_1 и Lab3\_2;
* скачать в подкаталог Lab3\_1 несколько файлов из Вашего домашнего каталога на сервере;
* просмотреть содержимое каталога Lab3\_1 путем временного перехода в командную строку Windows;
* изучить способы навигации по файловой системе сервера;
* создать в домашнем каталоге сервера подкаталог FTP и скопировать в него с локального компьютера файл, содержащий отчет по лабораторной работе № 2;
* по захваченным данным построить диаграмму потоков, сравнить ее с диаграммой из п.3.

1. Разработать программу, реализующую следующие функции ftp–клиента:

* создание и удаление каталогов;
* перемещение по каталогам;
* вывод содержимого каталога;
* получение, отправка, удаление и переименование файлов;

1. С помощью разработанной программы выполнить указанные действия, используя для этого локальный каталог Lab3\_2:

* провести сеанс обмена файлами с сервером fpm2.ami.nstu.ru (см. п.5);
* провести аналогичный сеанс обмена файлами с любым доступным ftp-сервером;
* по захваченным данным построить диаграмму потоков, сравнить ее с ранее построенными диаграммами.

4. Контрольные вопросы

1. Сколько логических соединений необходимо для работы протокола FTP, каким образом они используются?

2. Какие стадии можно выделить в FTP-сессии?

3. Работа протокола FTP в активном режиме.

4. Работа протокола FTP в пассивном режиме.

5. Какая сторона соединения отвечает за закрытие канала передачи данных, в каких случаях это происходит ?

6. Структура команды и ответа протокола FTP.

7. Основные команды протокола FTP.

# Лабораторная работа № 5. ПротоколЫ электронной почты

1. Цель работы

Целью работы является изучение основных принципов работы электронной почты и почтового протокола SMTP, а также разработка программы, реализующей этот протокол.

2. Методические указания

2.1 Общие сведения об электронной почте

Электронная почта – это сервис, предназначенный для сетевого обмена сообщениями в режиме офлайн. Отправитель направляет сообщение с указанием адреса назначения, а получатель его получает, просматривая свой почтовый ящик. Такое сообщение часто называется письмом аналогично обычной почте.

Электронная почта использует технологию «клиент-сервер». Почтовый сервер организует хранение, сортировку и структуризацию писем, а клиенты отвечают за отправку и получение писем. Примерами наиболее популярных почтовых клиентов являются Microsoft Outlook, Thunderbird, Mailbird, The Bat! и др.

Отправка и получение почты проводится на основе разных протоколов: для отправки используется протокол SMTP, для получения – протоколы POP3 или IMAP. Программные модули, реализующие эти протоколы, имеются как на почтовом сервере (SMTP- и POP3-серверы), так и на каждом почтовом клиенте (SMTP- и POP3-клиенты).

Возможны три варианта работы электронной почты: а) непосредственное взаимодействие клиента и сервера, б) схема с выделенным почтовым сервером, в) схема с двумя почтовыми серверами.

Первый вариант требует, чтобы на компьютерах каждого пользователя были установлены почтовый сервер и клиент. Отправитель вызывает функцию отправки сообщения и встроенный SMTP-клиент посылает запрос на установление связи SMTP-серверу на компьютере получателя и передает письмо через сеть. Это письмо сохраняется в памяти компьютера, а POP-клиент по команде получателя выводит его на экран, при необходимости выполняя преобразование формата.

Недостатком данного варианта является то, что компьютеры пользователей должны быть постоянно включены. Этот вариант в настоящее время не используется.

Второй вариант предусматривает размещение почтового сервера на отдельном мощном компьютере, который в круглосуточном режиме может передавать почтовые сообщения от многих отправителей ко многим получателям. На компьютерах пользователей устанавливаются только почтовые клиенты. Система DNS для каждого домена имен создает специальные записи, в которых хранятся DNS-имена почтовых серверов, обслуживающих пользователей, относящихся к этому домену.

В этом случае почтовая служба реализует следующий алгоритм:

* отправитель вызывает функцию отправки сообщения, указывая адрес получателя, например *user2@firm.ru;*
* SMTP-клиент отправителя обращается к системе DNS для определения имени почтового сервера в домене *firm.ru*. После получения от DNS в качестве ответа имени *mail.firm.ru*, SMTP-клиент еще раз обращается к DNS, чтобы узнать IP-адрес этого сервера;
* SMTP-клиент отправителя посылает по данному IP-адресу запрос на установление соединения с SMTP-сервером через порт 25 и передает письмо;
* письмо сохраняется в почтовом ящике получателя *user2@firm.ru*;
* получатель запускает свою почтовую программу, с помощью встроенного POP3-клиента выполняет подключение к почтовому POP3-серверу по порту 110 и копирует письмо на свой локальный компьютер.

Третий вариант предполагает использование двух почтовых серверов, каждый из которых обслуживает домен своего клиента. На каждом из этих серверов устанавливаются также клиентские части протокола SMTP. При отправке письма почтовый клиент отправителя передает сообщение по протоколу SMTP почтовому серверу своего домена. Это сообщение на сервере буферизуется, а затем по протоколу SMTP передается дальше на почтовый сервер домена получателя, откуда попадает на компьютер получателя.

Протокол SMTP не обеспечивает немедленную доставку, а задерживает ее на стороне отправителя, стороне получателя или на почтовых шлюзах. Для хранения писем на почтовом сервере отправителя организуется очередь, в которую поступают все сообщения, отправленные его клиентами, а на почтовом сервере получателя создаются почтовые ящики, в которых хранятся принятые сообщения для всех клиентов этого сервера.

Двухступенчатая передача через два почтовых сервера повышает надежность и гибкость процедуры доставки сообщения, что позволяет реализовать различные реакции на отказы на стороне сервера. Например, при невозможности передать письмо почтовому серверу получателя сервер отправителя может пытаться снова послать письмо, повторяя эти попытки в течение определенного времени (обычно несколько дней), после чего клиенту-отправителю направляется соответствующее сообщение.

Таким образом, SMTP-сервер может быть как конечным пунктом назначения, так и промежуточным ретранслятором почтовых сообщений.

2.2 Структура электронного письма

Письмо, передаваемое по электронной почте, содержит набор служебных заголовков и передаваемые пользователем данные (тело письма). Почтовые клиенты по умолчанию не отображают заголовки, но пользователь при необходимости всегда может их прочитать. Порядок доступа к заголовкам определяется разработчиками почтового клиента и описывается в его технической документации. Для наиболее популярных почтовых сервисов такие сведения можно найти в [13].

Заголовки содержат:

* информацию о типе передаваемых данных и способе их кодирования;
* результаты проверки сообщения антивирусными программами на наличие спама и вредоносных программ;
* результаты аутентификации, которая используется для борьбы с фишинговыми рассылками и почтовым спамом;
* идентификатор сообщения;
* дату и время отправки и получения сообщения;
* доменные имена и IP-адреса серверов отправителя и получателя.

Аутентификация сообщения применяется для проверки того, что письмо действительно было отправлено с заявленного домена. Для этого обычно используется технология DKIM (DomainKeys Identified Mail), основанная на добавлении к передаваемому сообщению цифровой подписи, связанной с именем домена отправителя.

Передаваемое сообщение может содержать различные типы данных и проходить через почтовые серверы, работающие с различными символьными кодировками. Это могут быть тексты, рисунки, фотографии, видеофайлы, а также различные виды электронных документов (Word, Excel, AutoCad и т.д). Основной формат электронных сообщений определен спецификацией RFC 5322, в соответствии с которой электронная почта является **символьно-ориентированной** системой передачи данных, ограничивающей почтовые сообщения 7-битовым символьным набором US-ASCII со строками размером не более 1000 символов, включая строчные разделители CRLF. Это означает, что любая передаваемая информация преобразуется в последовательность байтов, каждый из которых интерпретируется как символ в 7-битной кодировке ASCII (коды 0…127).

Для реализации данного преобразования при отправке сообщений применяется стандарт MIME, предусматривающий различные кодирующие механизмы для разных типов данных. Этот стандарт описывает, как пересылать по электронной почте текстовые, графические, мультимедийные, бинарные и исполняемые файлы. MIME позволяет размечать письмо на части, содержащие разные типы передаваемых данных так, чтобы получатель мог определить, что делать с каждой из частей сообщения.

Стандарт MIME предусматривает включение в письмо специального заголовка, включающего следующие основные поля:

* Mime-Version – содержит номер версии используемого протокола;
* Content-type – указывает тип и подтип содержимого:

1. text – текстовая информация. Основой подтип – ***plain***, соответствует обычному неформатированному тексту и не требует специального программного обеспечения для отображения этого текста за исключением поддержки национальных кодировок;
2. image – графические данные. Графика требует соответствующего устройства вывода (графический дисплей, принтер) для отображения информации. Изначально определены два подтипа для наиболее распространенных графических форматов - *jpeg* и *gif*;
3. audio – звуковая информация. Требует звуковое устройство (динамик или наушники) для вывода информации, основной подтип – *basic*;
4. video – видеоданные. Требует специальных аппаратных и программных возможностей для отображения видеоинформации. Стандартом определен единственный подтип – *mpeg*;
5. application – неинтерпретируемый двоичный код или информация, предназначенная для обработки почтовым клиентом. Основным подтипом является octet-stream, описывающий неинтерпретируемые двоичные данные, рекомендуемым действием для которых является предложение пользователю сохранить файл на диске.
6. **message** – письмо в письме;
7. multipart – указывает, что письмо состоит из нескольких частей, содержащих данные различных независимых типов. Изначально определено четыре подтипа: *mixed* – основной, *alternative* – для представления одних и тех же данных в разных форматах, *parallel*– если разные части документа должны просматриваться одновременно, *digest* – если каждая из частей тела письма имеет тип "message".

* Content-transfer-encoding – задает способ преобразования последовательности байтов передаваемых данных к стандартному 7-битному коду ASCII. Стандарт предлагает пять алгоритмов преобразований передаваемых данных:

1. 7bit – тело письма является текстом в кодировке US-ASCII (коды 0 -127);
2. 8bit – тело письма является текстом, в котором могут содержаться не-ASCII символы (коды 128 -255);
3. Binary – тело письма может содержать не-ASCII символы и строки могут быть произвольной длины (иметь размер более 1000 символов) и может не соблюдаться соглашение по признаку конца строки CRLF, принятое в RFC 5322;
4. quoted-printable – используется для представления данных, в основном состоящих из байтов, соответствующим печатным символам ASCII. Каждый байт, кроме обозначающих конец строки, представляется в виде двух 16-ричных цифр, перед которыми записывается символ “=”, а вся строка заключается в кавычки. При этом длина строки не может быть более 76 символов, включая строчный разделитель CRLF. Если нужно закодировать более длинные строки, то вводятся мягкие разрывы строк, которые обозначаются символом “=” в конце строки;
5. base64 – передаваемые данные преобразуются на основе позиционной системы счисления с основанием 64, в качестве цифр которой используются символы латинского алфавита (A-Z, a-z) и цифры (0-9), а также символы “+” и “/” (всего 64 символа).

*Обратите внимание: алгоритмы "7bit", "8bit" и "binary" не производят преобразования входных данных!*

Например, MIME-заголовок может выглядеть следующим образом:

*Mime-Version: 1.0*

*Content-Type: text/plain; charset=ISO-8859-1*

*Content-transfer-encoding: base64*

Видно, что письмо передается с применением протокола MIME версии 1.0 и содержит текстовые данные в представлении ISO-8859-1, а тело письма использует кодировку base64.

Кроме указанных основных полей заголовок MIME может включать в себя дополнительные поля, описывающие содержимое письма. Также MIME дает возможность применения нестандартных алгоритмов преобразования двоичных данных в текстовый вид, в этом случае перед именем алгоритма указывается префикс “х-”. Например, некоторые почтовые программы используют алгоритм *uuencode*, который является предшественником более современного алгоритма base64 [4]. В этом случае соответствующее поле MIME-заголовка будет таким:

*Content-transfer-encoding: х-uuencode*

Алгоритм base64 основан на замене 8-битного представления входных данных 6-битным представлением с последующим их кодированием на основе 64-символьного алфавита, представленного в таблице 5.1. На вход алгоритма подается последовательность байтов, в качестве которых могут выступать текст, графические изображения, программные файлы или любая другая информация, представленная набором байтов. Далее выполняются следующие действия:

* входной поток байтов разбивается на группы, состоящие из трех байтов (24 бита);
* каждая группа разбивается на четыре 6-битовых байта и определяются их числовые представления;
* каждый из 6-битовых байтов записывается символом, код которого берется из алфавита Base64;
* если при разбиении входного потока какая-то группа имеет размер менее трех байтов, то недостающие биты заполняются нулями, а в выходной поток вместо каждого нулевого 6-битового байта записывается символ “=”.

Таблица 2.1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Значение | Код | Значение | Код | Значение | Код | Значение | Код |
| 0 | A | 16 | Q | 32 | g | 48 | w |
| 1 | B | 17 | R | 33 | h | 49 | x |
| 2 | C | 18 | S | 34 | i | 50 | y |
| 3 | D | 19 | T | 35 | j | 51 | z |
| 4 | E | 20 | U | 36 | k | 52 | 0 |
| 5 | F | 21 | V | 37 | l | 53 | 1 |
| 6 | G | 22 | W | 38 | m | 54 | 2 |
| 7 | H | 23 | X | 39 | n | 55 | 3 |
| 8 | I | 24 | Y | 40 | o | 56 | 4 |
| 9 | J | 25 | Z | 41 | p | 57 | 5 |
| 10 | K | 26 | a | 42 | q | 58 | 6 |
| 11 | L | 27 | b | 43 | r | 59 | 7 |
| 12 | M | 28 | c | 44 | s | 60 | 8 |
| 13 | N | 29 | d | 45 | t | 61 | 9 |
| 14 | O | 30 | e | 46 | u | 62 | + |
| 15 | P | 31 | f | 47 | v | 63 | / |

Таким образом, алгоритм проводит обратимое кодирование с возможностью восстановления и переводит любую последовательность байтов в символы, гарантированно сохраняющиеся при передаче данных в любых сетях и между любыми устройствами. Недостатком является увеличение размера выходных данных в соотношении 4:3.

Определим, например, содержимое электронного письма при передаче строки «Студент», записанной в кодировке Windows 1251:

* находим 16-ричные и двоичные значения байтов исходной строки, дополнив двоичное представление двумя нулевыми байтами:

D1 F2 F3 E4 E5 ED F2

11010001 11110010 11110011 11100100 11100101 11101101 11110010 00000000 00000000

* делим входной поток на 6-битовые байты, определяем коды base64 и соответствующие им символы ASCII (табл. 5.2);
* результатом обработки будет строка “0fLz5OXt8g==”.

Таблица 2.2

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 6-битовый байт | 110100 | 011111 | 001011 | 110011 | 111001 | 001110 | 010111 | 101101 | 111100 | 100000 | 000000 | 000000 |
| Код base64 | 52 | 31 | 11 | 51 | 57 | 14 | 23 | 45 | 60 | 32 |  |  |
| Символ base64 | 0 | f | L | z | 5 | O | X | t | 8 | g | = | = |

2.3 Протокол SMTP

SMTP (Simple Mail Transfer Protocol) – это сетевой протокол, широко используемый в сетях TCP/IP для передачи электронной почты. Последняя версия протокола, включающая расширение ESMTP, описана в спецификации RFC 5321. Обмен данными по протоколу SMTP организуется по надёжному каналу, в роли которого обычно выступает TCP-соединение.

Протокол обеспечивает двухсторонний обмен сообщениями между клиентом и сервером в соответствии с рисунком 5.1. Почтовые сообщения и команды передаются по направлению от клиента к серверу, а от сервера к клиенту возвращаются ответы. SMTP-сервер должен постоянно находиться в режиме ожидания запросов со стороны SMTP-клиентов.

Алгоритм работы протокола:

* после получения команды на отправку сообщения SMTP-клиент посылает запрос серверу на установление соединения на порт 25, являющийся стандартным портом для протокола SMTP;
* если сервер готов, он возвращает свои идентификационные данные, в том числе свое DNS-имя. Иначе посылается соответствующее сообщение клиенту, в ответ тот снова посылает запрос, пытаясь заново установить соединение;
* клиент передает серверу адреса отправителя и получателя;
* если имя получателя соответствует ожидаемому, то сервер дает согласие на установление TCP-соединения и происходит передача сообщения. В рамках одного TCP-соединения клиент может передать несколько сообщений, предваряя каждое из них указанием адресов отправителя и получателя. Переданные сообщения сохраняются в буфере на сервере. Если сервер не может доставить сообщение, то он передает отправителю сообщение об ошибке и разрывает соединение;
* после завершения передачи TCP- и SMTP-соединения разрываются.

SMTP-клиент

SMTP-сервер

(порт 25)

Файловая

система

Команды и ответы

сообщения

Файловая система

Пользователь

Рис. 2.1. Модель SMTP протокола

После того, как канал передачи устанавливается и получено начальное подтверждение связи, SMTP-клиент инициализирует почтовую транзакцию, которая состоит из серии команд, задающих отправителя и адресата, и передаваемого сообщения.

Начало транзакции начинается командой MAIL, которая назначает идентификатор отправителю. Далее следует команда RCPT, которая указывает получателя (таких команд может быть несколько, если сообщение должно быть отправлено по нескольким адресам). Затем передается команда DATA, которая инициирует передачу почтовых данных и прекращается индикатором «end mail», завершающим транзакцию.

Диалог между клиентом и сервером осуществляется по принципу «команда клиента – ответ сервера». На каждую команду сервер всегда отвечает ответом. Ответы могут указывать на то, что команда была исполнена или содержать сообщение об ошибке. В случае отправки сообщения нескольким получателям протокол SMTP поддерживает передачу только одной копии данных для всех получателей, причём доставка в этом случае осуществляется через один хост или через один промежуточный ретранслятор.

После того, как почтовое сообщение было передано, клиент может потребовать закрыть соединение или может инициировать другую почтовую транзакцию. Кроме того, SMTP-клиент может использовать соединение с SMTP-сервером для дополнительных услуг, например, для проверки адреса электронной почты.

Команды и ответы протокола SMTP передаются в виде текстовых строк, которые состоят из нуля или более символов данных и завершаются последовательностью ASCII-символов CR (возврат каретки, код символа 0x0D) и LF (перевод строки, код символа 0x0A). Эта последовательность обозначается как <CRLF>. Внутри передаваемых строк применение символов CR и LF не допускается. Формат команды SMTP протокола: *команда [параметр] CRLF*

Здесь необязательный параметр определяет тип данных в команде. Ниже представлены команды протокола SMTP и их описание, а также возможные ответы сервера.

2.3.1 Приветствие (EHLO)

Сеанс связи клиента с сервером начинается с приветствия клиента. Синтаксис команды: *EHLO argument CRLF*,

где argument – полное доменное имя хоста клиента или адрес отправителя.

Команда используется для идентификации SMTP-клиента, при этом SMTP-сервер также идентифицирует себя в ответе на команду EHLO. Если SMTP-сервер поддерживает расширения службы SMTP, то ответ на данную команду будет «положительный», иначе будет отправлено сообщение об ошибке.

Ранние версии протокола SMTP предусматривали использование команды HELO вместо EHLO, поэтому современные SMTP-серверы обычно поддерживают команду HELO, чтобы давать на нее правильный ответ.

2.3.2 Начало транзакции (MAIL)

Синтаксис команды: *MAIL FROM:Reverse\_path CRLF*,

где Reverse\_path – почтовый ящик отправителя.

Эта команда используется для инициирования почтовой транзакции, в которой почтовые данные передаются на SMTP-сервер. Команда MAIL может передаваться только тогда, когда отсутствуют незавершенные почтовые транзакции.

*Обратите внимание: в команде MAIL не допускаются пробелы с любой стороны от знака «:» после FROM.*

2.3.3 Указание получателя (RCPT)

Синтаксис команды: *RCPT TO:Forward\_path CRLF*,

где Forward\_path – адрес получателя.

*Обратите внимание: не допускаются пробелы с любой стороны от знака «:».*

Адрес получателя обычно указывает почтовый ящик получателя. Если необходимо отправить почтовые данные нескольким получателям, то команду необходимо повторить для каждого получателя в отдельности. Для одного письма SMTP-сервер может принять до 100 команд RCPT. SMTP-сервер анализирует каждый адрес и выдает ответ, в котором указывается возможно или невозможно отправить сообщение данному адресату. Команда RCPT может выполниться успешно только после выполнения команды MAIL.

2.3.4 Текст сообщения (DATA)

Синтаксис команды: *DATA CRLF*

Команда DATA не требует никаких параметров и завершается последовательностью CRLF. Эта команда может быть выполнена только после успешного выполнения хотя бы одной команды RCPT.

Данная команда начинает передачу серверу сообщения. Отклик сервера на команду DATA оповещает клиента о готовности или неготовности сервера принять сообщение. Если сервер вернул положительный ответ, то клиент передает сообщение. Передаваемое сообщение состоит из заголовка и тела сообщения, которые разделены пустой строкой. Окончание сообщения обозначается строкой, состоящей из одной точки.

Если в теле сообщения содержится строка, состоящая из одной точки, то может произойти несвоевременное окончание сообщения. Для того, чтобы данная ситуация не возникла, на передающей стороне добавляется еще одна точка к началу каждой строки, которая начинается с точки. При приеме сообщения эти точки удаляются. Получив конец сообщения, сервер высылает ответ о возможности или невозможности отправить сообщение.

2.3.5 Выход (QUIT)

Синтаксис команды: *QUIT CRLF*

Данная команда завершает диалог между SMTP-сервером и SMTP-клиентом. В ответ на QUIT сервер высылает подтверждение и прекращает связь, после чего клиент тоже закрывает соединение.

2.3.6 Помощь (HELP)

Синтаксис команды: *HELP [command] CRLF*,

где command – название команды;

Команда может принимать в качестве аргумента любую команду протокола, в ответ сервер передаст её описание. Также сервер поддерживает команду HELP без аргументов, в этом случае сервер выдает список всех доступных команд.

2.3.7 Сброс (RSET)

Синтаксис команды: *RSET CRLF*

Эта команда прерывает текущую транзакцию. При этом все сохраненная информация об отправителе, получателе и передаваемые данные отбрасываются, а все буферы и таблицы состояния очищаются.

2.3.8 Пустая команда (NOOP)

Синтаксис команды: *NOOP CRLF*

Эта команда не влияет на какие-либо параметры или ранее введенные команды. Она не указывает никаких действий, кроме того, что сервер посылает подтверждение выполнения "250 OK".

Ответ сервера на команду состоит из кода завершения и текстовой строки. Числовой код необходим для распознавания ответа клиентом, а строка предназначена для пользователя. Исключением является отклик на команду EHLO и некоторые команды расширенной версии ESMTP. В ответ на команду EHLO сервер посылает список расширений ESMTP, которые поддерживаются этим сервером.

В таблице 2.3 приведены ответы, предусмотренные для команд SMTP.

Таблица 2.3

|  |  |  |
| --- | --- | --- |
| Код | Расшифровка | Команды |
| 211 | Состояние системы | HELP |
| 214 | Информация об использовании команд | HELP |
| 220 | Готовность к работе | Установление соединения |
| 221 | Канал передачи закрыт | QUIT |
| 250 | Команда выполнена успешно | EHLO, HELO, MAIL, RCPT, DATA, RSET, VRFY,EXPN, NOOP |
| 251 | Почта для получателя переадресована и будет доставлена на новый адрес | RCPT, VRFY |
| 354 | Команда DATA принята, ожидается текст сообщения, заканчивающийся строкой, сосотоящей из одной точки | DATA |
| 421 | Служба недоступна, связь прекращается. Ответ выдается при отключении сервера во время сеанса связи. | Любая команда |
| 450 | Доставка сообщения невозможна, почтовый ящик недоступен | RCPT |
| 451 | Выполнение команды прервано, ошибка сервера | MAIL, RCPT, DATA |
| 452 | Выполнение команды прервано, недостаточно памяти | MAIL, RCPT, DATA |
| 500 | Синтаксическая ошибка, команда не распознана | Несуществующая команда |
| 501 | Синтаксическая ошибка в параметрах или аргументах | Любая команда |
| 502 | Команда не поддерживается (отключена администратором) | VRFY, EXPN, HELP |
| 503 | Неправильный порядок команд | MAIL, RCPT, DATA |
| 504 | Параметр команды не поддерживается | EHLO, HELO, VRFY,EXPN, HELP |
| 550 | Команда не выполнена, почтовый ящик недоступен (не найден, запрещен доступ и т.д.) | EHLO, HELO, MAIL, RCPT, VRFY,EXPN |
| 551 | Адрес пользователя изменился | RCPT, VRFY |
| 552 | Выполнение команды прервано, превышен объем памяти | MAIL, RCPT, DATA |
| 553 | Неправильный синтаксис адреса | MAIL, RCPT, VRFY |
| 554 | Служба SMTP на вызываемой машине не запущена | Установление соединения |

2.4 Протокол POP3

POP3 (Post Office Protocol version 3) – протокол, определенный в спецификации RFC 1939 и предназначенный для доставки сообщений электронной почты конечному пользователю.

Протокол обеспечивает двухсторонний обмен сообщениями между клиентом и сервером на основе надежного TCP-соединения. Стандартными портами являются порт 110 для незащищенного соединения и порт 995 для соединения, защищенного протоколами SSL или TLS. В связи с многопользовательским режимом работы почтового сервера протокол POP3 поддерживает аутентификацию пользователей на основе идентификаторов и паролей.

Протокол POP3 оптимизирован для коротких подключений к почтовому серверу в условиях дорогой и медленной связи. При этом клиент подключается к серверу, считывает новую (или всю, в зависимости от настройки) почту на локальный компьютер и отключается от сервера. Начальная версия протокола предполагала считывание всех сообщений и их удаление с сервера. Обработка почты (чтение, сортировка по адресатам, темам и важности) проводится уже на локальном компьютере.

Достоинства POP3 – кратковременное соединение с сервером, т.е. минимальное потребление трафика, и минимальное использование ресурсов сервера.

В зависимоcти от настроек клиентской программы письма на сервере могут быть удалены сразу после скачивания или по заданному в клиенте алгоритму (например, после отключения клиента от сервера). При подключении по протоколу POP3 в почтовом ящике можно просматривать только папку «Входящие» (Inbox), скачивание с сервера структуры папок недоступно. Для контроля за приходом новых почтовых сообщений современные почтовые клиенты могут подключаться к серверу через заданные промежутки времени.

Сессия протокола POP3 может находиться в трёх состояниях:

* AUTHORIZATION – аутентификация клиента;
* TRANSACTION – транзакция, основной режим работы клиента с почтовым ящиком (вывод списка полученных писем, передача выбранных писем клиенту, пометка указанных писем для удаления);
* UPDATE – обновление, сервер удаляет помеченные письма и закрывает соединение.

Для получения почты клиент устанавливает TCP соединение с сервером, затем сервер высылает клиенту приветственное сообщение и сессия переходит в состояние AUTHORIZATION. После отправки клиентом своего имени и пароля и его успешной аутентификации сессия переходит в состоянии TRANSACTION и клиент имеет монопольный доступ к почтовому ящику. Далее клиент и сервер обмениваются командами и ответами до тех пор, пока соединение не будет закрыто или прервано. При этом сессия переводится в состояние UPDATE.

POP3-сервер может иметь таймер автоматического завершения сессии при бездействии клиента в течение определенного промежутка времени (inactivity autologout timer). Если клиент не передает серверу команду в этом интервале времени, то сервер разрывает TCP соединение без перехода в состояние UPDATE, т.е. без удаления сообщений и ответов клиенту.

Ответ сервера состоит из индикатора состояния и ключевого слова. Индикатор может быть положительный – «+OK» или отрицательный – «-ERR».

Для установления соединения клиент должен направить серверу запрос на подключение путем указания его доменного имени и номера порта. Если сервер доступен, то он посылает клиенту приветствие:

+OK POP3 server ready

Далее клиент должен аутентифицироваться одним из двух способов. Первый, наиболее часто используемый способ, это комбинация команд USER и PASS, второй способ – команда APOP.

2.4.1 Ввод имени пользователя (USER)

Синтаксис: *USER arg,*

где arg – имя почтового ящика.

Если POP3-сервер возвращает положительный ответ, то клиент может продолжить аутентификацию путем ввода пароля с помощью команды PASS или закрыть сессию командой QUIT. Если POP3-сервер возвращает индикатор состояния "-ERR", то клиенту необходимо повторить команду USER или завершить сессию.

2.4.2 Ввод пароля (PASS**)**

Синтаксис: *PASS arg,*

где arg – пароль от электронного ящика.

Эта команда может быть выполнена только после успешного выполнения команды USER. После ввода команды PASS сервер использует аргументы команд USER и PASS для проверки доступа к соответствующему почтовому ящику.

Если аутентификация клиента выполнена успешно, то POP3-сервер вернет положительный ответ, после этого клиент может работать со своим почтовым ящиком. Сервер может вернуть отрицательный ответ в случае, если не найден электронный ящик или пароль введен неверно.

2.4.3 Ввод имени и пароля пользователя (APOP)

Синтаксис: *APOP arg1 arg2,*

где arg1 – имя электронного ящика (аналогично аргументу команды USER), arg2 – пароль, зашифрованный по алгоритму MD5.

Аутентификации пользователя командой APOP не является обязательно реализуемой командой POP3-сервера. Она представляет более высокий уровень защиты, чем комбинация команд USER и PASS. Алгоритм шифрования пароля составляет секретный ключ, используя открытый ключ и временную метку. Сервер передает клиенту временную метку вместе с приветствием.

2.4.4 Завершение работы (QUIT)

Синтаксис: *QUIT*

Если сессия находится в состоянии TRANSACTION и клиент вводит команду QUIT, то POP3-сессия переходит в состояние UPDATE. В состоянии UPDATE сервер производит удаление сообщений, помеченных на удаление. Но если на момент ввода команды QUIT сессия находилась в состоянии AUTHORIZATION, то сессия завершится.

2.4.5 Проверка текущего состояния почтового ящика **(**STAT)

Синтаксис: *STAT*

Данная команда проверяет текущее состояние почтового ящика. В ответ на команду POP3-сервер возвращает количество сообщений, которые может получить клиент с сервера, а также их общий размер в байтах. При этом сообщения, помеченные на удаление, не учитываются.

После открытия почтового ящика сервер устанавливает каждому сообщению уникальный номер (нумерация проводится, начиная с единицы) и определяет его размер в байтах. Обращение к сообщениям происходит по их номеру.

2.4.6 Вывод списка сообщенийв почтовом ящике **(**LIST)

Синтаксис: *LIST [arg],*

где arg – необязательный аргумент, который указывает номер сообщения.

Если команда LIST введена с аргументом, и сообщение с указанным номером существует, то сервер вернет его номер и размер в байтах. Если команда введена без аргументов, то POP3-сервер вернет информацию о всех сообщениях в почтовом ящике. Информация о сообщениях, которые были помечены на удаление, не выводится.

2.4.7 Передача сообщенияклиенту **(**RETR)

Синтаксис: *RETR arg,*

где arg – номер сообщения.

Команда предназначена для передачи сообщения клиенту. Если сообщение с указанным номером существует, то сервер передает клиенту текст этого сообщения, иначе вернет сообщение об ошибке.

2.4.8 Пометка сообщения на удаление (DELE)

Синтаксис: *DELE arg,*

где arg – номер удаляемого сообщения.

Командой DELE помечаются сообщения на удаление. При этом сообщения удаляются, после завершения транзакции, когда сессия находится в состоянии UPDATE.

2.4.9 Проверка состояния соединения (NOOP)

Синтаксис: *NOOP*

Команда NOOP используется для проверки состояния соединения POP3-сервера с клиентом. Если соединение активно, то сервер вернет положительный ответ «+OK».

2.4.10 Отмена удаления помеченных сообщений (RSET**)**

Синтаксис: *RSET*

Эта команда используется для отката транзакции внутри сессии. Клиент может отменить удаление помеченных сообщений, отправив команду RSET.

2.4.11 Вывод заданного числа строк сообщения (TOP**)**

Синтаксис: *TOP arg1 arg2,*

где arg1– номер сообщения, arg2 – количество строк.

С помощью этой командой пользователь может получить несколько строк сообщения. В ответ на команду сервер отправляет тему сообщения с указанным номером и строки сообщения, количество которых указанно вторым аргументом. Если значение второго аргумента больше, чем количество строк, запрашиваемого сообщения, то сервер выведет всё сообщение.

Различные расширения протокола POP3 могут включать и другие команды.

*Обратите внимание: команды 2.4.5 – 2.4.11 могут использоваться только в режиме TRANSACTION.*

2.5 Примеры сеансов почтовых протоколов

2.5.1 Протокол SMTP

Ниже приведены результаты журналирования двух сеансов передачи почтовых сообщений по протоколу SMTP. Первый сеанс проводится по незащищенному соединению с использованием порта 25 почтового сервера *cn.ami.nstu.ru*, а второй – по защищенному соединению с использованием порта 587 и протокола TLS почтового сервера *mail.ru*.

Сеанс по незащищенному соединению

Сообщение отправляется с адреса *b4@cn.ami.nstu.ru* получателю с адресом *b10@cn.ami.nstu.ru* с помощью программы, приведенной в приложении 1. Результат перехвата команд этого сеанса приведен на рис.2.2, исходный код программы – в приложении 1.

**Client**: EHLO localhost CRLF //Идентификация клиента

**Server**: 220 cn.ami.nstu.ru ESMTP //Ответ – идентификация сервера

//Клиент начинает почтовую транзакцию, b4@cn.ami.nstu.ru – адрес отправителя

**Client**: MAIL FROM:b4@cn.ami.nstu.ru CRLF

**Server**: 250-cn.ami.nstu.ru //Сервер сообщает список поддерживаемых расширений

250-SIZE 20480000

250-AUTH LOGIN

250 HELP

**Client**: RCPT TO:b10@cn.ami.nstu.ru CRLF // b10@cn.ami.nstu.ru – адрес получателя

**Server**: 250 OK

**Client**: DATA CRLF //Клиент готов отправить сообщение

**Server**: 250 OK //Сервер готов к приему сообщения

**Client**: FROM: b4@cn.ami.nstu.ru

TO: b10@cn.ami.nstu.ru

SUBJECT: Test message

CRLF //пустая строка для распознавания сервером начала сообщения

**Client**:

**Client**: This is a test message! CRLF

**Client**: . CRLF //обозначение конца сообщения

**Server**: 354 OK, send.

**Client**: QUIT CRLF //Завершение диалога сервера и клиента

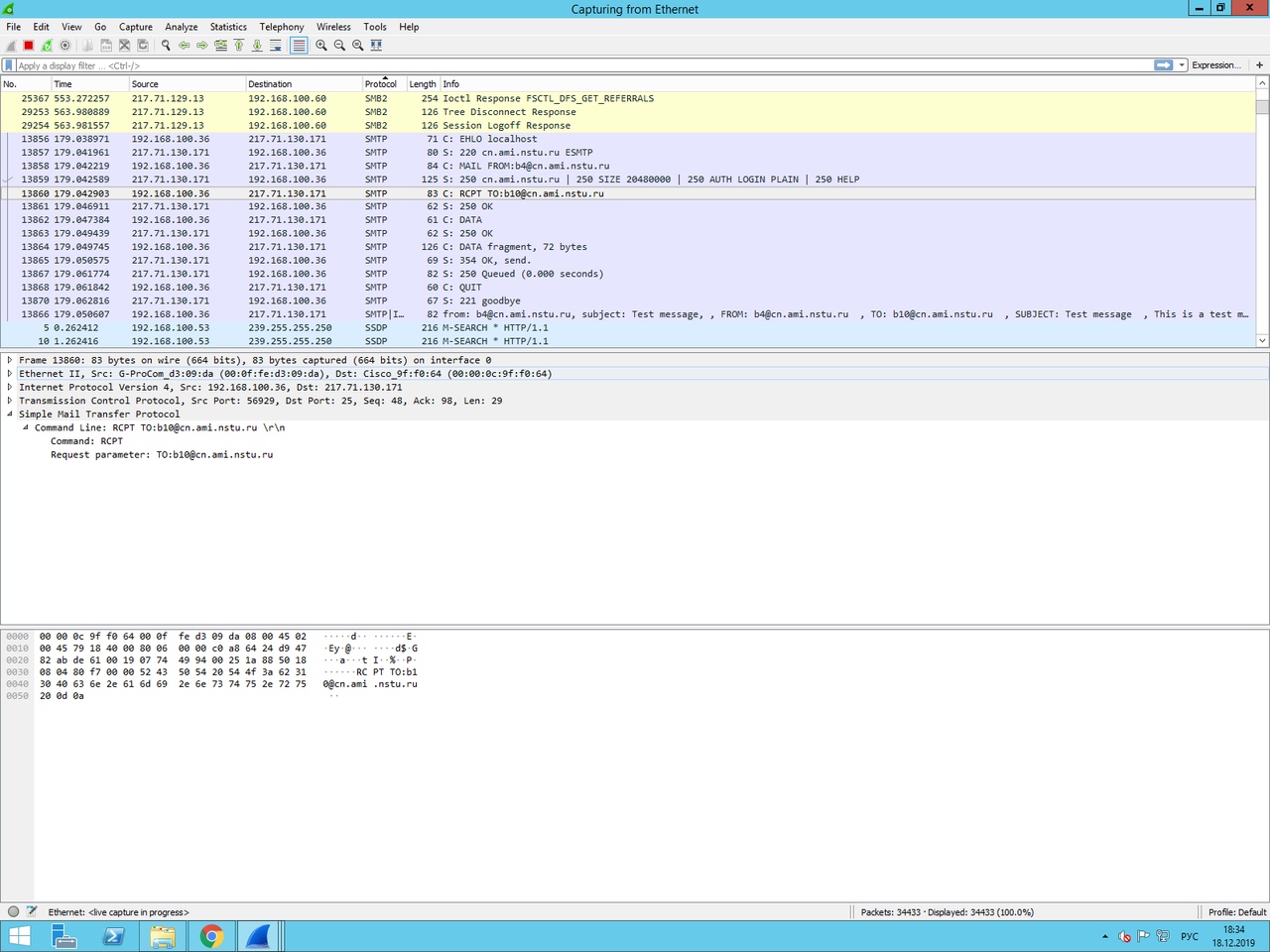
**Server**: 250 Queued (0.000 seconds)

Рис.2.2

Сеанс по защищенному соединению

Сообщение отправляется с адреса *ivanov@gmail.com* получателю на этот же адрес с помощью программы, приведенной в приложении 2.

*Обратите внимание: работа с протоколом TLS возможна только после ввода команды STARTTLS.*

**Server**: 220 smtp.gmail.com ESMTP a18sm9102014ljp.33 - gsmtp

**Client**: EHLO 127.0.0.1

**Server**: 250-smtp.gmail.com at your service, [80.89.157.74]

250-SIZE 35882577

250-8BITMIME

250-STARTTLS

250-ENHANCEDSTATUSCODES

250-PIPELINING

250-CHUNKING

250 SMTPUTF8

**Client**: STARTTLS

**Server**:220 2.0.0 Ready to start TLS

**Client**: AUTH LOGIN

**Server**:334 VXNlcm5hbWU6

**Client**: a3VsYWJ1aG92YS5zQGdtYWlsLmNvbQ==

**Server**:334 UGFzc3dvcmQ6

**Client**:

**Server**:235 2.7.0 Accepted

**Client**: MAIL FROM: <ivanov@gmail.com>

**Server**:250 2.1.0 OK a18sm9102014ljp.33 - gsmtp

**Client**: RCPT TO: < ivanov@gmail.com>

**Server**:250 2.1.5 OK a18sm9102014ljp.33 - gsmtp

**Client**: DATA

**Server**:354 Go ahead a18sm9102014ljp.33 - gsmtp

**Client**: FROM: ivanov@gmail.com

**Client**: TO: ivanov@gmail.com

**Client**: SUBJECT: header

**Client**:

**Client**: .

**Server**:250 2.0.0 OK 1574140653 a18sm9102014ljp.33 - gsmtp

**Client**: QUIT

**Server**:221 2.0.0 closing connection a18sm9102014ljp.33 – gsmtp

Результат перехвата в виде диаграммы потоков приведен на рис.2.3, откуда видно, что после поступления от клиента команды STARTTLS и взаимного приветствия сервер и клиент начинают работу по защищенному протоколу TLS.

2.5.2 Протокол POP3

Ниже приведен один из вариантов сеанса получения почты (рис.2.4). Здесь пользователь подключается к POP3-серверу, расположенному на компьютере с адресом 192.168.31.13 и загружает с него сообщение № 4 размером 252 байта.

**Server:** +OK Greetings!

**Client:** USER user@192.168.31.13

**Server:** +OK Send your password

**Client:** PASS 123456

**Server:** +OK Mailbox locked and ready

**Client:** STAT

**Server:** +OK 6 1506

**Client:** RETR 4

**Server:** +OK 252 octets

Return-Path: user@192.168.31.139

Received: from 192.168.31.139 (BadPC [192.168.31.143])

by DESKTOP-0UJTSSF with ESMTPA

; Sat, 23 Nov 2019 17:51:03 +0700

Message-ID: <CE1D8DA4-226F-4853-8C09-41EB7D290343@DESKTOP-0UJTSSF>

SUBJECT:TEST!

Это сообщение для лабораторной работы!

**Client:** QUIT

**Server:** +OK server saying goodbye…

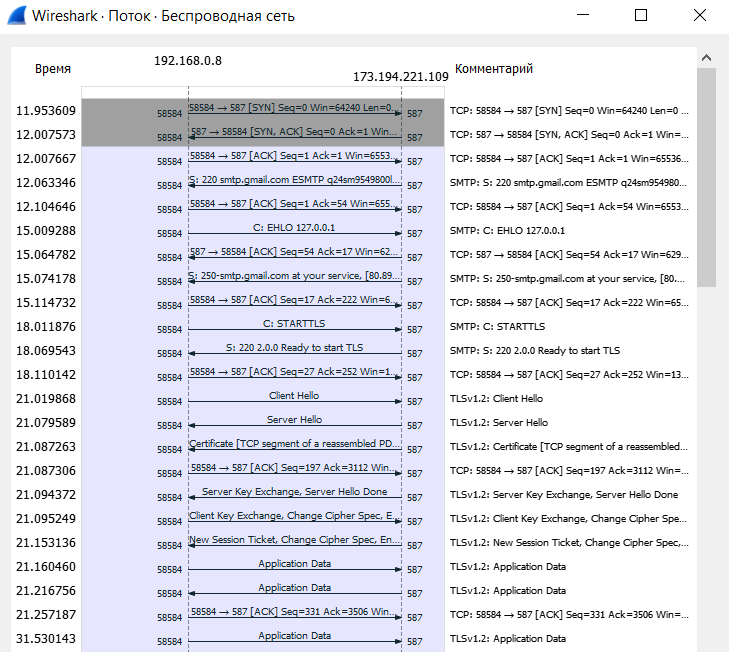


Рис.2.3

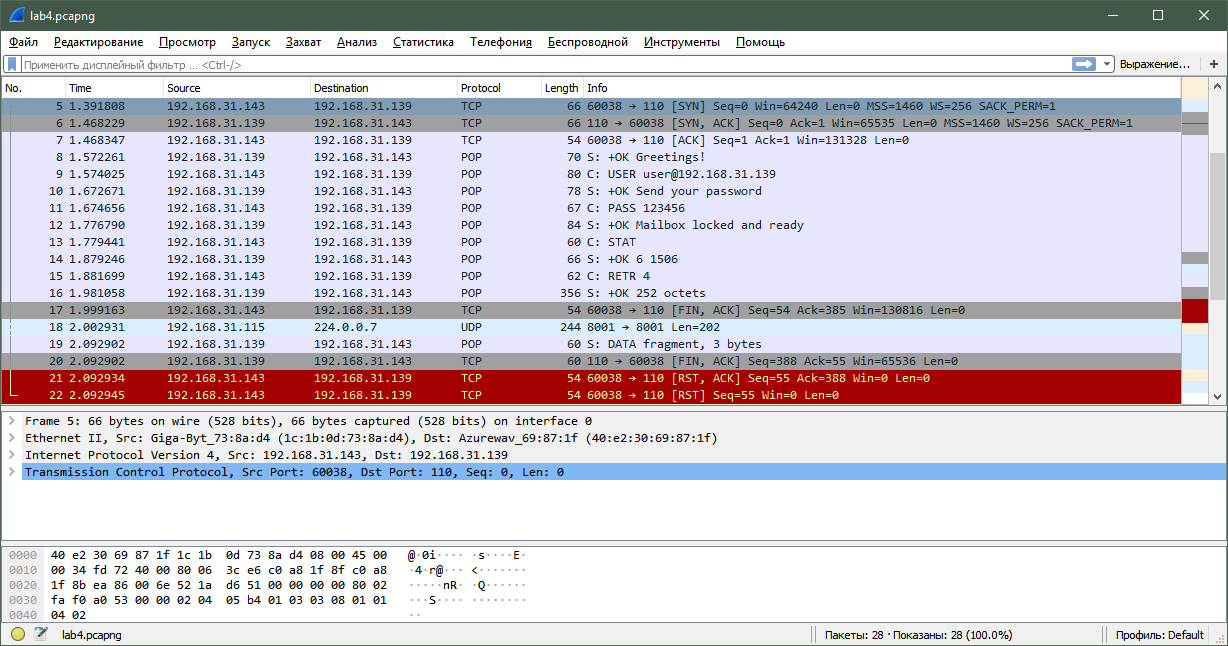


Рис. 2.4

Исходный текст программы POP3-клиента приведен в приложении 3.

3. Особенности выполнения лабораторной работы

3.1 Общие сведения

Лабораторная работа выполняется с любым доступным почтовым сервером. Можно использовать учебный почтовый сервер ФПМИ *cn.ami.nstu.ru*, учетную запись для которого необходимо получить у преподавателя. Сервер доступен только в пределах локальной сети ФПМИ по стандартному порту 25.

Возможно подключение к корпоративным или общедоступным почтовым службам – *mail.nstu.ru, mail.ngs.ru, gmail.com, mail.ru, mail.yandex.ru* и т.д. Для этого необходимо иметь зарегистрированный почтовый ящик и ознакомиться с его требованиями по подключению со стороны внешних приложений. Например, для *smtp.gmail.com* необходимо в настройках почты разрешить доступ на подключение внешних небезопасных приложений (рис.2.5)

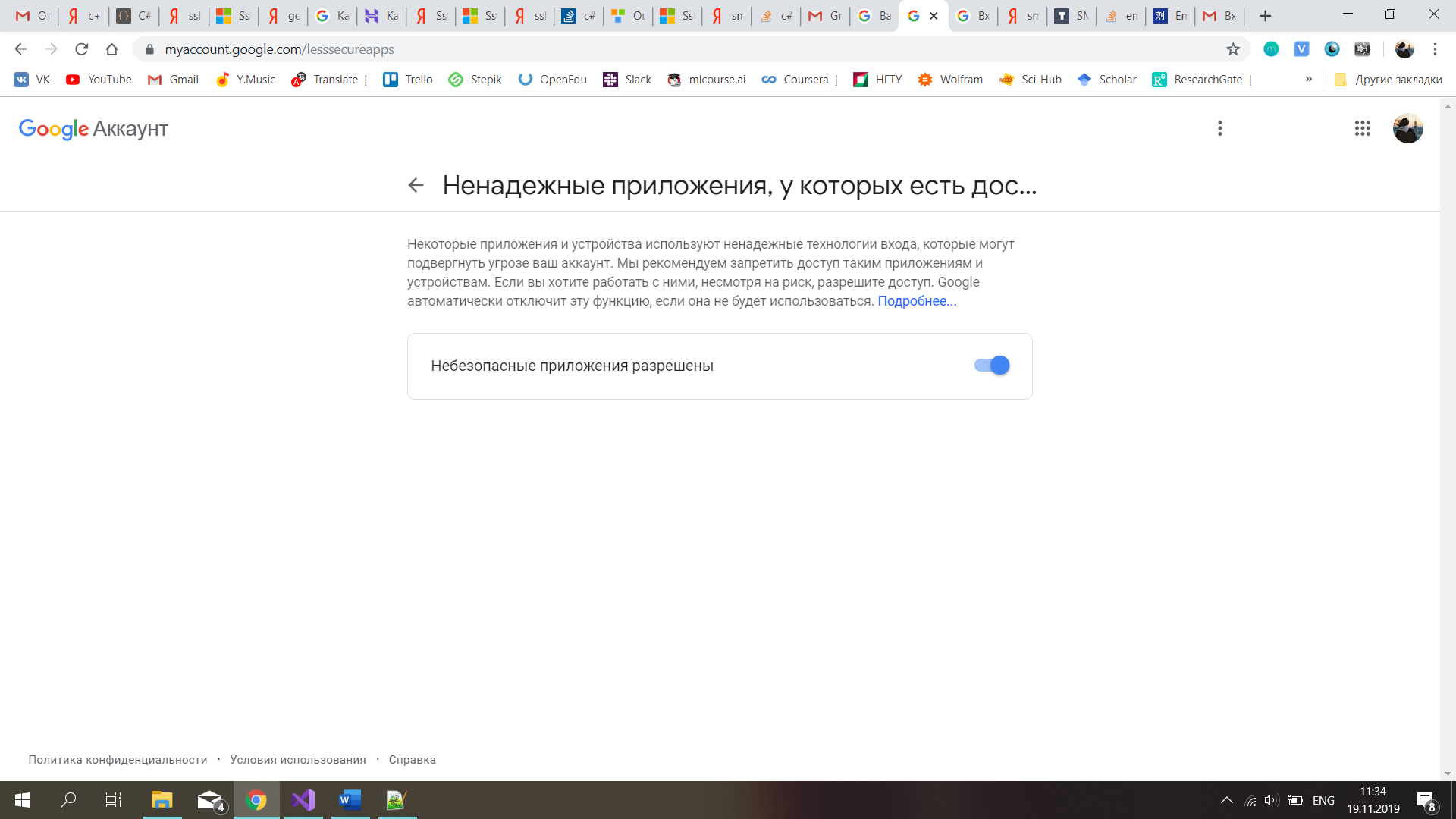


Рис. 2.5

Для корректной работы с протоколом POP3 в настройках сервера pop.gmail.com надо включить поддержку этого протокола (рис. 2.6):

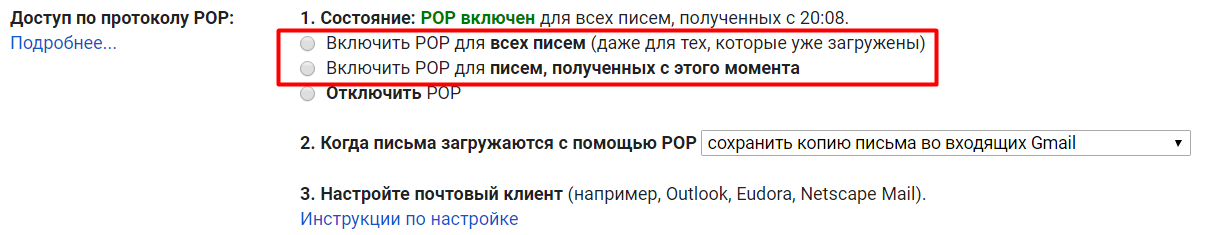


Рис. 2.6

Некоторые SMTP-серверы используют шифрование данных по протоколам SSL или TLS. В этом случае доступ должен проводиться по портам 465 или 587, а перед авторизацией необходимо передать на сервер команду STARTTLS. Например, при работе с SMTP-сервером НГТУ (mail.nstu.ru) в почтовой программе необходимо установить следующие настройки: протокол – TLS(STARTTLS), номер порта – 587, режим аутентификации должен быть включен, в качестве логина используется Ваша учетная запись от личного кабинета в информационной системе НГТУ. На диаграмме рис. 2.3 видно, что после поступления от клиента команды STARTTLS и взаимного приветствия сервер и клиент начинают работу по защищенному протоколу TLS.

Для POP3-серверов также возможно использование защищенных протоколов, использующих порт 995.

В таблице 2.3 приведены настройки некоторых общедоступных почтовых серверов.

Таблица 2.3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | gmail.com | yandex.ru | mail.ru | ngs.ru |
| Протокол SMTP | Хост | smtp.gmail.com | smtp.yandex.ru | smtp.mail.ru | smtp.ngs.ru |
| Порт | 587 | 587 | 465 | 25/587 |
| SSL/TLS | да | да | да | нет/да |
| Протокол POP3 | Хост | pop.gmail.com | pop.yandex. ru | pop.mail.ru | mail.ngs.ru |
| Порт | 995 | 995 | 995 | 110/995 |
| SSL/TLS | да | да | да | нет/да |

Разработка клиентских приложений для доступа к SMTP- и POP3-серверам может проводиться на любом языке программирования, поэтому средства для организации обмена данными будут определяться возможностями этого языка. Например, в C# для обмена данными с защищенным сервером можно использовать класс

*System.Net.Security.SslStream.*

3.2 Порядок выполнения работы

1. Ознакомьтесь с типовой сессией SMTP (п.2.5.1).
2. Запустите анализатор Wireshark и загрузите в него файл *smtp\_testX.pcapng*, где X – номер бригады. В файле находятся результаты обмена данными по протоколу SMTP с почтовым сервером. Отфильтруйте из общего потока пакетов сеанс связи с SMTP-сервером и выполните его анализ. Все файлы можно скачать с сайта *moodle.ami.nstu.ru*.
3. Получите логин и пароль для доступа к почтовому серверу. Разработайте клиентское приложение для отправки текстовых сообщений по протоколу SMTP с учетом следующих требований:

* все команды и данные должен вводить пользователь (адреса получателя и отправителя, текст сообщения);
* подключение к почтовому серверу реализовать на основе сокетов;
* приложение должно формировать строки команд в соответствии с протоколом SMTP, выводить их на экран и отправлять на сервер. Ответы сервера также должны выводиться на экран.
* весь процесс почтовой сессии должен сохраняться в файле журнала smtp\_Х.log;

1. С помощью разработанного приложения отправьте сообщение получателю, указанному в таблице 2.4 и с помощью Wireshark выполните анализ переданных пакетов путем фильтрации исходящего трафика по IP-адресам источника и получателя, названию протокола (SMTP) или номеру порта SMTP-сервера. Сравните результаты анализа с данными из файла журнала.

Таблица 2.4

|  |  |
| --- | --- |
| Номер бригады отправителя | Номер бригады получателя |
| 1 | 5 |
| 2 | 9 |
| 3 | 6 |
| 4 | 10 |
| 5 | 7 |
| 6 | 11 |
| 7 | 8 |
| 8 | 13 |
| 9 | 1 |
| 10 | 14 |
| 11 | 2 |
| 12 | 15 |
| 13 | 3 |
| 14 | 12 |
| 15 | 4 |

1. По захваченным данным постройте диаграмму потоков.
2. Сохраните перехваченный трафик в файле *smtp\_X.pcapng*.
3. Ознакомьтесь с типовой сессией POP3 (п.2.5.2).
4. Запустите анализатор Wireshark и загрузите в него файл *pop3\_testX.pcapng*, где X – номер Вашей бригады. В файле находятся результаты обмена данными по протоколу POP3 с почтовым сервером. Отфильтруйте из общего потока пакетов сеанс связи с POP3-сервером и выполните его анализ.
5. Разработайте клиентское приложение для получения почтовых сообщений по протоколу POP3 с учетом требований п.3 задания для SMTP-клиента.
6. С помощью разработанного приложения получите из почтового ящика сообщения, которые были направлены при выполнении п.4 задания и выполните анализ трафика путем фильтрации по IP-адресам источника и получателя, названию протокола и номеру порта сервера. Сравните результаты анализа с данными из файла журнала.
7. По захваченным данным постройте диаграмму потоков.
8. С помощью почтового сервиса, установленного на Вашем компьютере (Microsoft Outlook, mail.ru, gmail.com и т.д.) откройте почтовый ящик с сообщениями, которые были направлены при выполнении п.4 задания. Выполните анализ служебных заголовков одного из писем, занесите заголовки в отчет.

Отчет по работе должен содержать описание алгоритмов, исходные тексты программ, полную сессию обмена данными с SMTP- и POP3-серверами, а также анализ перехваченных пакетов SMTP и POP3.

1. Разработайте клиентское почтовое приложение, реализующее протоколы SMTP и POP3. Основу приложения должны составить программы, разработанные в предыдущих пунктах задания и объединенные общим графическим интерфейсом. Требования к программе:

* обеспечить выполнение следующих функций: просмотр содержимого почтового ящика; отправка, прием и удаление сообщений электронной почты.
* имена отправителей и получателей должны задаваться пользователем;
* сеансы отправки и получения сообщений должны выводится в отдельные текстовые поля формы.

4. Контрольные вопросы

* 1. Варианты организации электронной почты.
  2. Структура электронного письма, стандарт MIME.
  3. Типы содержимого письма.
  4. Алгоритмы кодирования содержимого письма.
  5. Поясните на конкретном примере основные этапы кодирования base64.
  6. Назначение протокола SMTP, параметры SMTP-серверов.
  7. Модель SMTP протокола.
  8. Опишите алгоритм установления соединения клиента с SMTP-сервером, подтвердите ответ результатами перехвата пакетов с помощью WireShark.
  9. Команды SMTP протокола и их порядок в почтовой сессии.
  10. Классификация ответов SMTP-сервера.
  11. Можно ли передать одно сообщение нескольким получателям ?
  12. Можно ли передать в одной сессии несколько сообщений ?
  13. Назначение и модель протокола POP3.
  14. Алгоритм установления соединения клиента с POP3-сервером, подтвердите ответ результатами перехвата пакетов с помощью WireShark.
  15. Команды протокола POP3, порядок команд в почтовой сессии.
  16. В каких состояниях может находиться POP3-сессия.
  17. Варианты ответов POP3-сервера.

# 

# Лабораторная работа № 6. технология WEB

1. Цель работы

Изучить основы Web-технологии, типы Web-документов и архитектуру программного обеспечения Web-браузеров.

1. Методические указания

2.1 Общие сведения

В настоящее время в сети Интернет используется множество различных сервисов, каждый из которых реализуется на основе собственной технологии. Напомним, что под термином «технология» понимается совокупность методов и инструментов для достижения желаемого результата. Некоторые виды сервисов представлены на рис. 2.1.

Интернет-технологии

Мессенджеры и социальные сети

Web-технологии

Электронная коммерция

Электронная почта

IP-телефония

Телеконференции

JavaScript

CSS

Браузеры

Поисковые системы

Языки Web-про-граммирования

HTML, XML

Рис.2.1 Сервисы сети Интернет

Технология World Wide Web (WWW, Всемирная паутина), часто называемая для краткости Web-технологией, в настоящее время является одним из наиболее распространенных сервисов Интернет, представляющим собой распределенную информационную систему. Здесь, как в любой информационной системе, основными функциями являются хранение и поиск информации.

Идея создания Всемирной паутины принадлежит Тиму Бернерс-Ли и Роберту Кайо (1989 г) и основана на объединении двух существующих технологий - технологии применения IP-протоколов для передачи данных и технологии гипертекста, предназначенной для реализации быстрого перехода от одного фрагмента текста к другому по выделенным ссылкам. Web-технология, в отличие от существующих в то время технологий обмена данными (telnet, ftp, электронная почта), должна обеспечить интерактивный режим работы пользователя. Информационная система, построенная на этих принципах, могла объединить множество информационных ресурсов, называемых сайтами и разбросанных по многочисленным компьютерам.

Сервис WWW, как и все сетевые приложения, работает на основе архитектуры «клиент-сервер». Web-серверы являются поставщиками информационных ресурсов, а клиенты – потребителями этих ресурсов. В качестве клиентов используются специальные программы, называемые браузерами, которые устанавливаются на компьютерах конечных пользователей.

Основной логической единицей информации, поставляемой клиенту сервером, является страница, на которой могут отображаться любые виды контента (текст, графика, аудио, видео, ссылки на другие страницы и т.д.). Описание страницы проводится средствами языка гипертекстовой разметкиHTML, который позволяет разработчику страницы указать, что на ней находится и в каком порядке должны быть представлены элементы страницы на экране. В настоящее время, учитывая возрастающую сложность организации страниц, вместо термина «страница» часто используется термин «документ».

Физически описание каждой страницы хранится на сервере в виде отдельного файла. При обращении браузера к странице сервер передает ему соответствующий файл, после чего браузер обрабатывает HTML-код страницы, формируя её визуальное представление на мониторе.

Для однозначной идентификации и доступа к web-страницам был разработан стандартный формат. Этот формат представляется символьной строкой, называется унифицированным локатором ресурсов(Uniform Resourse Locator,URL) и имеет следующий вид:

*метод\_доступа://имя\_хоста[:порт]/имя\_файла*

Здесь *метод\_доступа* определяет способ доступа к странице, *имя\_хоста* – доменное имя компьютера, на котором находится страница, *порт* – необязательный номер порта протокола, *имя\_файла* – полное имя файла, в котором содержится требуемая страница.

Метод доступа может быть локальным в случае доступа к ресурсу, расположенному на локальном компьютере, или сетевым для доступа к удаленному компьютеру. В первом случае доступ проводится через файловую систему и в URL указывается “file”, во втором случае в качестве метода доступа в URL указывается имя соответствующего сетевого протокола (http, ftp и т.д.). При локальном доступе имя хоста можно не указывать.

Разработка стандарта URL вызвана следующими причинами:

* система WWW содержит множество компьютеров (хостов) и страница может находиться на любом из них;
* каждый компьютер может содержать множество страниц, поэтому каждой из них должно быть присвоено уникальное имя;
* web-технология должна обеспечивать отображение страниц, содержащих элементы, представленные различными типами данных (например, текстовые или двоичные);
* система WWW интегрирована с другими приложениями, поэтому браузер должен знать, какой прикладной протокол должен применяться для доступа к странице.

2.2 Основные типы WEB-документов

В целом Web-документы могут быть разбиты на три категории.

*Статический* документ создается его разработчиком, имеет фиксированное содержимое и находится в файле, который хранится на сервере. Конечный пользователь не может изменить содержимое документа и при обращении к нему получает только его копию.

Основными преимуществами статического документа являются простота, надежность и скорость воспроизведения. Для его создания достаточно знания языка гипертекстовой разметки HTML, с помощью которого можно не только создавать тексты, но включать в них команды форматирования и ссылки на другие элементы данных. Созданный статический документ всегда останется действительным, а браузер всегда может его сохранить на локальном диске для ускоренного выполнения последующих запросов на получение этого документа.

Недостатком статического документа является отсутствие гибкости: документ должен пересматриваться при каждом изменении содержащейся в нем информации, а внесение изменений требует участия человека. Поэтому в статическом документе не может быть представлена информация, которая часто меняется.

*Динамический* документ создается внешней прикладной программой, которая запускается web-сервером при поступлении соответствующего запроса от браузера. Такая программа может храниться как на самом web-сервере, так и на удаленном компьютере. Содержимое динамического документа может изменяться от одного запроса к другому.

Для создания и обработки динамических документов могут применяться разные технологии. Одной из первых появилась технология CGI, которая определяет интерфейс для связи внешней программы с web-сервером, в том числе структуры входных и выходных данных внешней программы. Такая программа часто называется CGI-скриптом и может быть написана на любом языке программирования, поддерживающем консольный ввод- вывод. По своей сути CGI-скрипт является шлюзом, который применяется для организации доступа web-сервера к другим ресурсам, например, к базам данных. Все скрипты на сервере обычно помещают в каталог *cgi* или *cgi-bin*.

Взаимодействие браузера и web-сервера организуется следующим образом:

* входные данные вводятся пользователем в поля экранной формы (например, для регистрации на сайте) и посылаются от браузера к web-серверу в виде набора параметров, добавляемых к URL; в качестве стартового символа строки параметров применяется символ “?”, параметры в списке разделяются символом “&”;
* web-сервер принимает запрос, записывает набор параметров в свою область окружения;
* web-сервер запускает CGI-скрипт, передавая ему набор параметров как глобальную переменную;
* CGI-скрипт формирует результат обработки в виде HTML-кода, который возвращается через web-сервер браузеру.

Ниже приведен пример запроса динамической страницы:

*https://yandex.ru/search/?text=нгту&lr=65*

Здесь браузер обращается к поисковой системе yandex.ru и передает ей значения двух параметров – *text* и *lr*.

Основные недостатки динамических документов:

* значительные затраты на их создание, т.к. требуется разработка специального программного обеспечения;
* необходимость использования мощного серверного компьютера;
* увеличение времени формирования документа;
* результат, возвращаемый браузеру, не изменяется во времени, т.е. браузер не имеет информации о том, является страница статической или получена динамически с помощью компьютерной программы.

*Активный* документ создается разработчиком и хранится на web-сервере, но, в отличие от статического документа, содержит исполняемый код, который должен выполняться на стороне клиента и реализовать определенные алгоритмы вычисления и отображения результатов. В этом случае по запросу браузера web-сервер возвращает ему активный документ в виде потока байтов. Когда клиент получает этот документ, он может сохранить его в собственной памяти и в дальнейшем работать с ним без применения повторного запроса к web-серверу. В качестве примеров активного документа можно привести применение бизнес-мессенджера для общения с клиентами на сайтах интернет-магазинов или формирование анимированного изображения, когда web-сервер передает браузеру статическое изображение в виде графического файла и программу анимации.

Основным преимуществом активного документа перед динамическим является его способность непрерывно обновлять информацию. Например, активный документ, который показывает цены на акции, может осуществлять выборку биржевой информацию, не требуя каких-либо действий со стороны пользователя.

Недостатками активных документов являются дополнительные затраты на создание и выполнение таких документов, а также повышенная уязвимость к вредоносным программам. Увеличение затрат связано с требованиями высокой квалификации разработчиков, возможности исполнения программного кода на любых аппаратных платформах и наличием мощного клиентского компьютера.

2.3 Структура и основы функционирования браузера

2.3.1 Архитектура браузера

Браузер – это клиентская программа, которая применяется для отображения web-ресурсов и содержит следующие компоненты (рис. 2.2):

* интерфейс пользователя (user interface), включающий все визуальные элементы окна за исключением области, в которой отображается запрашиваемая страница;

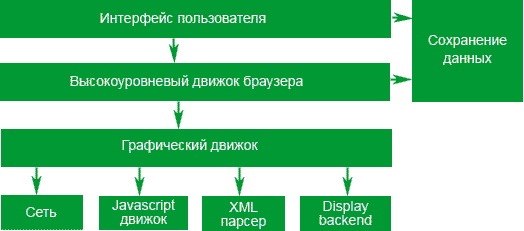


Рис. 2.2

* высокоуровневый движок (browser engine) – контроллер, управляющий взаимодействием интерфейса пользователя и модулем отображения. Он интерпретирует щелчки мыши и нажатия клавиш на клавиатуре, а также вызывает другие компоненты для выполнения операций, указанных пользователем. Например, при вводе в адресную строку или щелчке на гипертекстовой ссылке он вызывает клиентскую программу для выборки затребованного документа с удаленного сервера, а затем вызывает интерпретатор для отображения документа в окне браузера;
* модуль отображения (rendering engine) – является основным компонентом браузера, отвечающим за формирование визуального представления запрошенной страницы. Он анализирует полученный код HTML или XML, учитывает влияние CSS, Javascript и других объектов, расположенных в web-документе (например, изображения или flash-анимации). На основе всех этих данных модуль создает макет документа, который пользователь видит на экране. По этой причине модуль отображения часто называют графическим движком браузера, ключевыми компонентами которого являются HTML и CSS парсеры, позволяющие отобразить документ даже при наличии ошибок в HTML и CSS.
* сетевой механизм (networking) – предоставляет функциональные возможности для получения и обработки URL-адресов, управления всеми аспектами удаленного доступа к серверам, использования различных интернет-протоколов (HTTP, FTP и др.), обеспечения безопасности и кэширования полученных данных для минимизации сетевого трафика. В нем предусмотрен кроссплатформенный интерфейс и низкоуровневые реализации для каждой из поддерживаемых платформ.
* внутренний интерфейс пользователя (display backend) – кроссплатформенный интерфейс, применяемый для отрисовки основных графических элементов (окна, выпадающие списки и т.д.). На низком уровне использует методы, предусмотренные операционной системой;
* JavaScript интерпретатор (JavaScript Interpreter) – необходим для выполнения кода JavaScript;
* хранилище данных (Data Persistence) – используется для хранения различных данных браузера на жестком диске локального компьютера (закладки, пароли, настройки и т.д.). В спецификации HTML5 вводится новая технология Web Database, которая представляет собой полнофункциональный вариант базы данных в самом браузере.

Основным требованием, предъявляемым к браузеру, является обеспечение быстрой загрузки web-документов. Время, затрачиваемое на получение от сервера требуемого документа, складывается из времени выполнения DNS-запроса (определение IP-адреса по доменному имени сервера), времени подключения к серверу по IP-адресу (установление TCP-соединения) и времени ожидания ответа сервера (время прохождения пакетов данных к серверу и обратно, а также время обработки данных сервером). На рис. 2.3 показано распределение этих временных интервалов для одного из запросов.

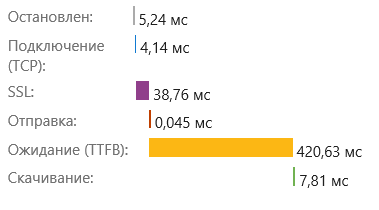


Рис. 2.3

Первые спецификации протокола HTTP (версии 0.9 и 1.0), с помощью которого организуется взаимодействие браузера с web-сервером, предусматривали кратковременное соединение браузера с web-сервером. Причины этого были связаны со следующими факторами:

* страницы содержали в основном гипертекстовую информацию;
* компьютеры имели низкую производительность;
* каналы передачи данных были низкоскоростными.

В этом случае браузер устанавливал соединение, отправлял запрос, получал затребованный элемент данных или сообщение об отсутствии такого элемента и соединение закрывалось. Если возникала необходимость перехода по гиперссылке, то браузер открывал новое соединение.

По мере усложнения передаваемых документов кратковременное соединение стало неэффективным. Например, на одной странице могут содержаться не только текстовые данные с гиперссылками, но и фотографии, видеоролики, анимированные изображения и т.д. Каждый из этих мультимедийных элементов данных хранится в отдельном файле, поэтому для отображения страницы браузер должен выполнить несколько запросов к серверу, каждый раз открывая новое соединение, что приводит к существенному увеличению общего времени загрузки документа.

Для сокращения затрат времени на установление соединений протокол HTTP версии 1.1 предусматривает продолжительное подключение браузера к web-серверу (keep-alive). Соединение не разрывается, если все запросы направлены на один хост. При этом все запросы могут отправляться асинхронно, а ответы от сервера должны приходить последовательно. Если какой-либо элемент web-документа размещен на другом хосте, то для запроса этого элемента устанавливается новое соединение, начиная с DNS-запроса.

Еще большее быстродействие имеет протокол HTTP/2, который, в отличие от предыдущих версий, является бинарным и использует сжатие заголовков, параллельную загрузку нескольких элементов web-документа в одном TCP-соединении и мультиплексирование запросов.

2.3.2 Кэширование данных

Количественный анализ затрат времени на получение web-документов от сервера показывает, что основным фактором задержки является не установление соединения, а ожидание ответа от сервера, что наглядно видно на рис. 6.3. Для сокращения времени ожидания необходимо сократить число обращений к серверу, для чего современные браузеры используют механизм *кэширования*, который позволяет сохранять на локальном диске загружаемые пользователем страницы целиком или частично. Это позволяет не только ускорить загрузку страниц, но и сократить сетевой трафик.

Идея кэширования основана на том, что даже в динамических и активных web-документах всегда есть статические элементы данных, например логотип, CSS-файлы, фрагменты текста или элементы интерфейса (кнопки, меню и т.д.). При первой загрузке web-документа браузер сохраняет такие элементы в кэше и каждый последующий раз он уже не будет запрашивать эти данные из интернета, а извлечет их из кэша. Кэширование является особенно важным при просмотре больших документов или при использовании низкоскоростных сетевых соединений.

Механизм кэширования реализует определенный набор правил, которые устанавливаются протоколом HTTP, администраторами сайтов или пользователем браузера. Кэширование не проводится в следующих случаях:

* в заголовке ответа от сервера явно указывается запрет кэширования;
* запрос к серверу является авторизованным (authorized) или безопасным (HTTPS)
* кэшированный контент считается свежим (freshness), то есть его установленное время жизни не завершилось.

Если время жизни контента в кэше истекло, то на исходный сервер направляется запрос на валидацию (validation) для проверки этого контента на актуальность. Если контент был изменен с момента истечения времени жизни, то сервер возвращает браузеру его свежую версию, в противном случае направляет подтверждение о валидации.

Недостатком кэширования является доступность содержимого кэша для всех пользователей, имеющих учетные записи на локальном компьютере, а также для вредоносного программного обеспечения. Это связано с тем, что кэш хранится в определенных скрытых папках. Ниже приведены имена таких папок для некоторых браузеров:

*Google Chrome:*

c:\users\Пользователь\AppData\Local\Google\Chrome\User Data\Default\Cache;

c:\users\Пользователь\AppData\Local\Google\Chrome\User Data\Profile 1\Cache;

*Internet Explorer 11:*

c:\users\Пользователь\AppData\Local\Microsoft\Windows\INetCache;

c:\users\Пользователь\AppData\Local\Microsoft\Windows\Temporary Internet Files;

2.3.3 Файлы Cookie

Кроме основной информации, возвращаемой браузеру, многие сайты передают вспомогательные служебные файлы, которые называются куки (cookie). Изначально куки предназначались для сохранения некоторых параметров сессии с целью повышения комфортности работы пользователя. Например, можно сохранить регистрационные данные пользователя (логин, пароль, номер телефона, адрес электронной почты), регион, номер банковской карты, список выбранных к покупке в интернет-магазине товаров для того, чтобы при повторном обращении к этому сайту повторно не вводить эти данные.

Создание куки в браузере может проводится двумя способами:

* куки порождается на web-сервере и передается в браузер совместно с запрошенным документом;
* куки порождается в браузере с помощью программы, встроенной в активный документ.

С развитием рекламных и статистических сайтов основной функцией куки стало накопление максимального количества сведений о пользователе – статистика посещений сайтов, область предпочтительных интересов, часто используемые объекты поиска. Собранная информация используется для показа контекстной рекламы и других целей, известных заинтересованным лицам.

Использование куки всегда несет потенциальную опасность попадания конфиденциальной информации в чужие руки, т.к. её можно перехватить при обмене данными браузера с web-сервером или путем считывания с диска компьютера с помощью различных вредоносных программ.  Поэтому все браузеры имеют функцию удаления куки, которую рекомендуется регулярно использовать.

Основными параметрами куки являются:

* name – имя куки-файла;
* value –значение куки, которое определяется содержимым файла;
* expires, max-age – определяют срок жизни куки, по истечению которого она будет автоматически удалена. Если не указать этот параметр, или установить его значение в ноль, то куки будут автоматически удалены при закрытии браузера.
* path – указывает путь к каталогу сервера, для которого будут доступны куки. Если указать корневой каталог “/”, то куки будут доступны всему домену. Значением по умолчанию является текущий каталог, из которого куки устанавливается в браузер.

2.4 Основные сведения о протоколе HTTP

Взаимодействие браузера с Web-сервером проводится с помощью протокола HTTP. Рассмотрим его основные характеристики на примере HTTP/1.1.

HTTP реализует технологию “клиент-сервер” и, подобно ранее рассмотренным прикладным протоколам (FTP, SMTP и POP3), является текстовым протоколом, использующим кодировку ASCII. Сервером является программа web-сервера, установленная на удаленном хосте, а в качестве клиента выступает браузер, установленный на локальном компьютере конечного пользователя. Протокол позволяет браузеру запросить конкретный элемент данных у сервера. причем вместе с каждой передачей данных сервер отправляет дополнительную информацию. Браузер может не только запрашивать, но и передавать данные на сервер.

Структура HTTP-сообщений включает стартовую строку, набор заголовков (метаданные) и тело сообщения. Стартовая строка определяет тип сообщения, заголовки характеризуют тело сообщения, параметры передачи и прочие сведения, а тело сообщения содержит передаваемые данные и обязательно отделяется от заголовков пустой строкой.

Стартовые строки запроса и ответа имеют разный формат:

* для запроса – *операция URI HTTP/версия*
* для ответа – *HTTP/версия КодСостояния [Пояснение]*

Здесь *операция* – это регистрозависимая символьная строка, определяющая основную операцию над ресурсом; *URI* – унифицированный идентификатор ресурса; *версия* – версия протокола HTTP; *КодСостояния –* код, указывающий браузеру наличие или отсутствие ошибок при выполнении запроса; *Пояснение* – необязательное короткое текстовое пояснение к коду ответа.

Идентификатор ресурса URI (Uniform Resource Identifier) — это символьная строка, которая идентифицирует какой-либо web-ресурс: документ, изображение, файл, службу, ящик электронной почты. В отличие от него, унифицированный локатор ресурса (URL) не только идентифицирует web-ресурс, но и указывает его местонахождение в следующем формате: *имя\_схемы://имя\_домена/путь*

Например, строка *mailto:myname@yandex.ru* является идентификатором почтового ящика, а строка *http://moodle.ami.nstu.ru/login/index.php* является локатором ресурса.

Основные операции протокола HTTP представлены в таблице 2.1.

Таблица 2.1

|  |  |
| --- | --- |
| Операция | Описание |
| GET | Запрос конкретного ресурса с сервера. Сервер возвращает заголовок, за которым следует код состояния, пустая строка и затребованный ресурс. Ответы на запрос GET могут кэшироваться. |
| HEAD | Запрос метаданных ресурса. Сервер возвращает только заголовки, не передавая тело сообщения. Заголовки могут кэшироваться. Применяется для валидации URL, чтобы узнать, не изменился ли ресурс с момента последнего обращения. При несовпадении полученных метаданных с соответствующей информацией в кэше копия ресурса помечается как устаревшая. |
| POST | Применяется для передачи пользовательских данных заданному ресурсу; сервер проводит обработку и добавляет переданные в теле запроса данные к указанному элементу (например, добавляет переданное сообщение к существующему списку сообщений). Ответы сервера на запрос POST не кэшируются. |
| PUT | Применяется для загрузки содержимого запроса на указанный в запросе ресурс. Если по указанному URI такой ресурс не существует, то он создается, иначе его содержимое заменяется новым содержимым. Ответы сервера на запрос PUT не кэшируются. |
| DELETE | Применяется для удаления указанного элемента. |

При вводе пользователем в адресной строке URL или выборе ссылки браузер формирует запрос GET, в котором указан определенный элемент данных, а сервер возвращает запрашиваемый элемент. Запрос GET имеет следующую форму:

*GET URL версия CRLF*

Здесь *URL –* указатель требуемого элемента данных, *версия* – указывает версию протокола HTTP, *CRLF* – символы CR и LF кодировки ASCII.

Формат заголовков **HTTP**  совпадает с форматами, которые используются в почтовых протоколах, и представляет собой регистронезависимую разделённую двоеточием пару «параметр-значение». Перед заголовками всегда выводится стартовая строка, сообщающая браузеру с помощью кода состояния о статусе выполненного запроса.

Коды состояния сгруппированы в пять групп: 1хх – информационные коды, 2хх – коды успешного завершения, 3хх – коды перенаправления запроса, 4хх – коды, связанные с ошибками в запросе, 5хх – коды, связанные с ошибками сервера.

В таблице 2.2 приведены некоторые коды ответов сервера, полный список можно найти в [21].

Таблица 2.2

|  |  |  |
| --- | --- | --- |
| Код | Ошибка | Описание |
| 200 | Запрос выполнен успешно | Затребованный ресурс был отправлен клиенту. |
| 201 | Запрос выполнен успешно | В результате успешного выполнения запроса был создан новый ресурс. |
| 300 | Множественный выбор (Multiple Choices) | Затребованный URL обозначает более одного ресурса, сервер не смог однозначно определить, к какой странице относится URL. |
| 303 | Смотрите другой ресурс (See Other) | Запрошенный ресурс находится под другим адресом и его следует запрашивать, используя метод GET. |
| 307 | Временное перенаправление Temporary Redirect | Затребованный ресурс был временно переведен на другой адрес, который необходимо прописать в Location. |
| 400 | Неверный запрос (Bad Request) | В запросе клиента обнаружена синтаксическая ошибка. |
| 401 | Неавторизов(Unauthorized) | Для доступа к документу необходимо вводить пароль или быть зарегистрированным пользователем. |
| 403 | Доступ к документу запрещен (Forbidden) | Если вы хотите, чтобы страница индексировалась, необходимо разрешить доступ к ней. |
| 404 | Ресурс не найден (Not Found) | Документ не существует |
| 500 | Внутренняя ошибка сервера (Internal Server Error) | Сервер столкнулся с непредвиденным условием, которое не позволяет ему выполнить запрос. |
| 501 | Метод не поддерживается (Not Implemented) | Сервер не поддерживает функциональные возможности, требуемые для выполнения запроса. Этот ответ соответствует состоянию, когда сервер не распознает метод запроса и не способен обеспечить его для любого ресурса. |
| 504 | Время прохождения через межсетевой шлюз истекло (Gateway Timeout) | Сервер, при работе в качестве внешнего шлюза или прокси-сервера, своевременно не получил отклик от вышестоящего сервера, к которому он обратился, пытаясь выполнить запрос. |

Если запрос был сформирован неправильно или затребованный в нем элемент недоступен, код состояния позволяет найти эту проблему.

Дополнительные строки заголовка предоставляют более полную информацию об элементе данных (размер, время последнего изменения, тип содержимого и т.д.). Ответы сервера могут содержать несколько независимых частей (multipart), которые называются сущностями (entity) и могут иметь различные параметры. Например, при передаче активного документа в ответе могут находиться html-текст и программный код и каждый из этих фрагментов должен кодироваться собственным способом.

Протокол HTTP предусматривает четыре группы заголовков:

* основные заголовки (General) – включаются в любое сообщение клиента и сервера.
* заголовки запроса (Request) – используются только в запросах клиента.
* заголовки ответа (Response) – присутствуют только в ответах сервера.
* заголовки сущности (Entity) – сопровождают каждую сущность сообщения.

В таблице 6.3 приведены некоторые часто используемые заголовки.

Таблица 6.3

| **Группа** | **Заголовок** | **Краткое описание** |
| --- | --- | --- |
| Entity | Content-Encoding | Применяется при необходимости перекодировки содержимого (например, gzip/deflated). |
| Content-Language | Локализация содержимого (язык(и)) |
| Content-Length | Размер тела сообщения (в октетах) |
| Content-Type | Указывает тип содержимого (mime-type, например text/html), часто включает указание на таблицу символов (charset) |
| Last-Modified | Дата/время последней модификации сущности |
| Expires | Дата предполагаемого истечения срока актуальности сущности. |
| General | Cache-Control | Определяет директивы управления механизмами кэширования для прокси-серверов. |
| Connection | Задает параметры, требуемые для конкретного соединения. |
| Date | Дата и время формирования сообщения |
| Transfer-Encoding | Задает тип преобразования, применимого к телу сообщения. В отличие от Content-Encoding этот заголовок распространяется на все сообщение, а не только на сущность. |
| Request | Accept | Определяет применимые типы данных, ожидаемых в ответе. |
| Accept-Encoding | Определяет применимые форматы кодирования/декодирования содержимого (напр, gzip) |
| Accept-Language | Применимые языки. Используется для согласования передачи. |
| Authorization | Учетные данные клиента, запрашивающего ресурс. |
| From | Электронный адрес отправителя |
| Host | Имя/сетевой адрес [и порт] сервера. Если порт не указан, используется 80. |
| User-Agent | Информация о пользовательском агенте (клиенте) |
| Response | Server | Информация о программном обеспечении сервера, отвечающего на запрос (это может быть как веб- так и прокси-сервер). |
| ETag | Идентификатор, значение которого зависит от состояния загружаемого клиентом ресурса, используется для проверки актуальности |
| Accept-Ranges | Перечень единиц измерения |

На рис.2.4 приведен пример заголовков HTTP-ответа сервера.

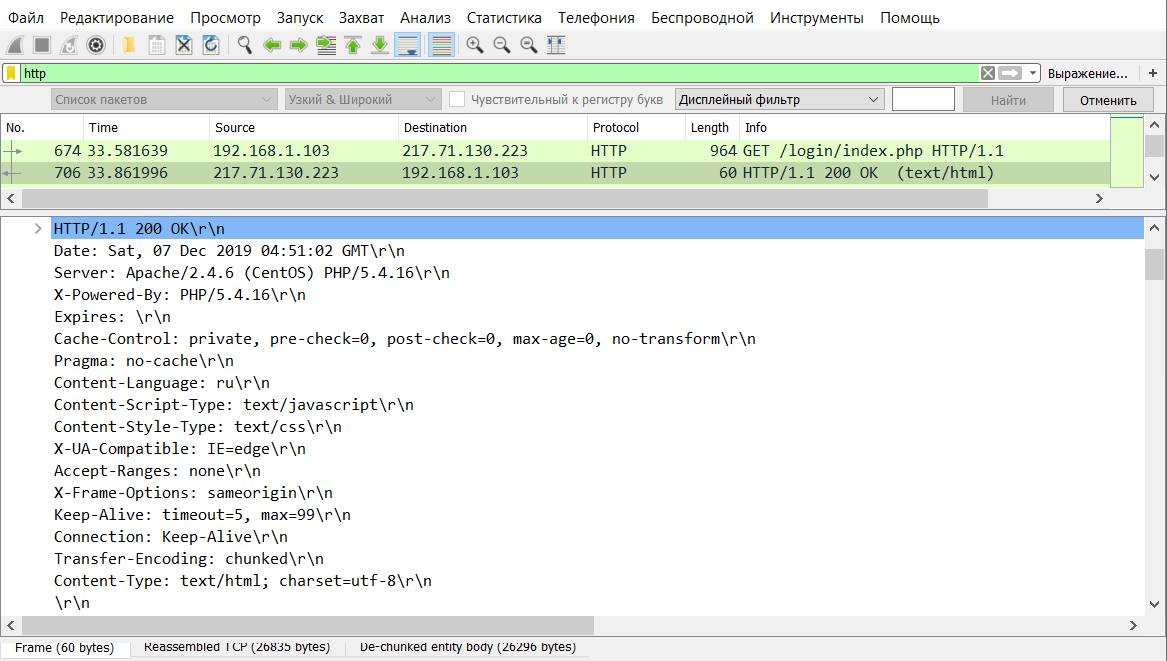


Рис. 2.4

Код состояния 200 в первой строке указывает, что сервер успешно выполнил запрос; остальные строки содержат дополнительную информацию о затребованном элементе данных.

Просмотр всех запросов и ответов, включая служебные заголовки, можно проводить в любом браузере путем включения режима разработчика клавишей F12 и перехода на вкладку «Сеть» основного окна. На рис. 2.5 показано окно разработчика на примере Internet Explorer.

На этой вкладке в левом окне выводится перечень всех запросов, отправляемых браузером на сервер для отображения главной страницы сайта *moodle.ami.nstu.ru*. В правом окне выводятся характеристики выбранного запроса: заголовки, тело, параметры, куки и затраты времени на выполнение (рис.2.6). Список запросов может быть можно фильтровать по операции (GET, POST и т.д.) и по типу содержимого (документ, таблица стилей, графика и т.д.).

В протоколе HTTP предусмотрены определенные средства, которые обеспечивают кэширование. При формировании заголовка HTTP сервером может быть указан тайм-аут кэша для страницы. Если страница содержит информацию, которая постоянно изменяется, то при формировании ее сервером может быть указано, что страница не предназначена для кэширования. Кроме того, браузер может отправить запрос с HTTP-заголовком, где указано, что для ответа на этот запрос не должны использоваться данные из кэша.

3. Порядок выполнения лабораторной работы

1. Каждая бригада должна разработать простой сайт для доступа к отчетам по выполнению лабораторных работ в виде простой HTML- страницы, которая будет выводить на экран список имен файлов. Желательно использовать файлы в формате .pdf, так как содержимое таких файлов отображается большинством браузеров. Каждый элемент списка должен являться ссылкой на соответствующий файл, т.е. при щелчке мыши по имени файла содержимое этого файла должно быть выведено на экран. В заголовке страницы укажите Вашу группу и фамилии членов бригады. HTML-код страницы сохраните в файле *index.html*, пример кода приведен в приложении 4.

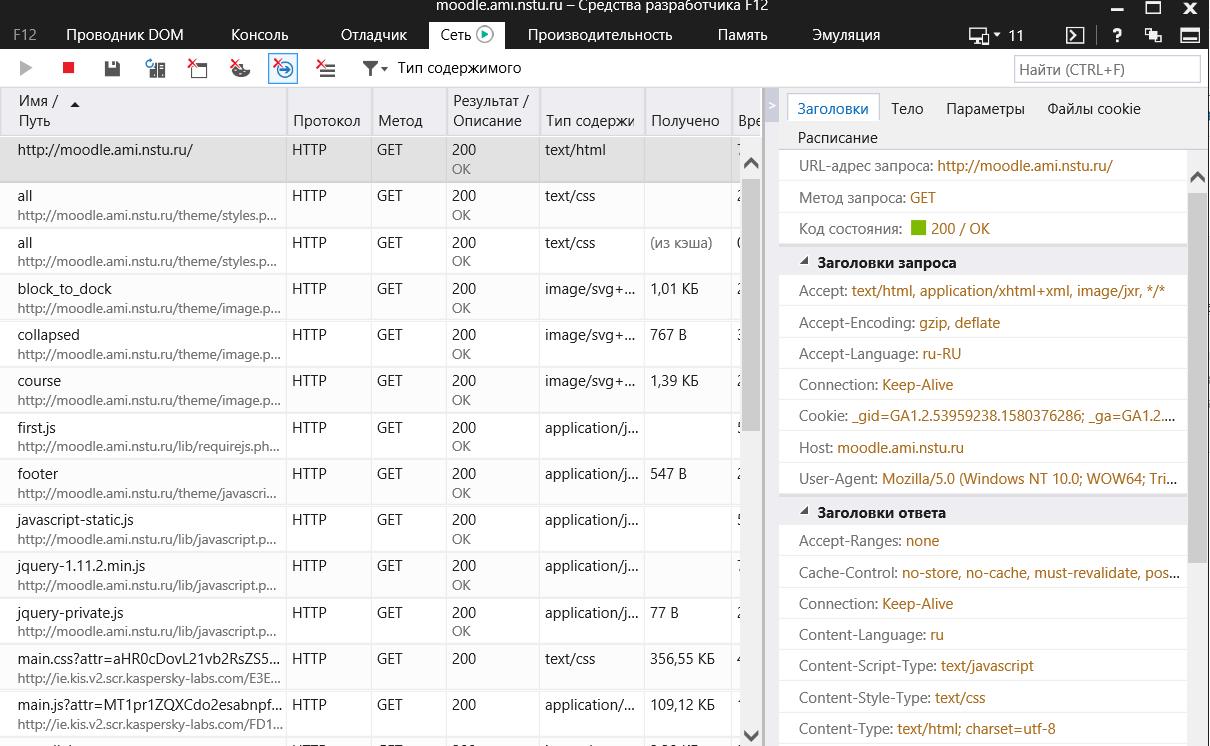


Рис. 2.5

Работоспособность страницы можно проверить, сделав щелчок мыши на имени файла в проводнике. Для корректного отображения символов кириллицы желательно использовать кодировку utf-8. Все файлы, имеющие отношение к странице, должны быть расположены в одном каталоге.

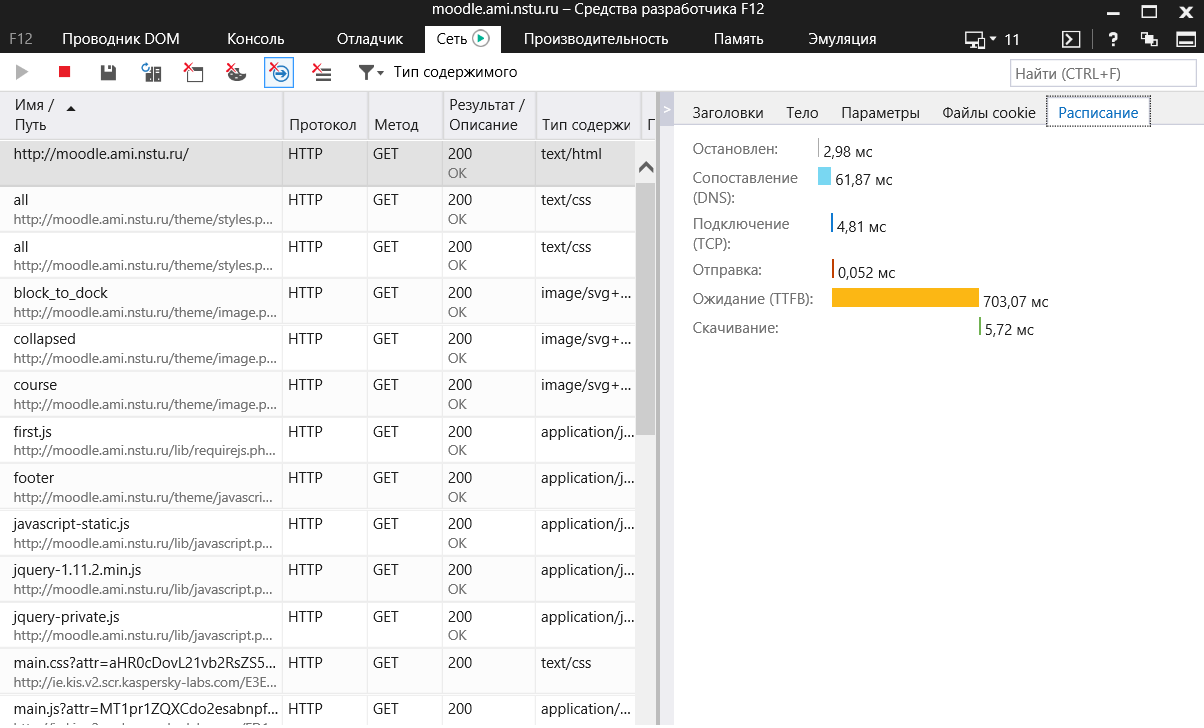


Рис. 2.6

1. Подключитесь к серверу *fpm2.ami.nstu.ru*, разрешите полный доступ к домашнему каталогу для всех пользователей командой **chmod -R 755 ~**
2. Создайте в домашнем каталоге сервера подкаталог с именем *public\_html* и разместите в нем файл *index.html*, после чего содержимое этого файла будет доступно по адресу следующего вида: *http://fpm2.ami.nstu.ru/~ваш логин*. Например, если Ваш логин «pm8103», то по адресу *http://fpm2.ami.nstu.ru/~pm8103* браузер откроет Вашу начальную страницу *index.html*. Если Вы создали HTML- страницу в Windows, то для копирования на сервер можно использовать утилиту WinSCP.

При наличии возможности можно использовать ранее созданные на сервере html-файлы.

1. Разработайте Web-сервер, который должен удовлетворять следующим требованиям: работать в локальной сети факультета по протоколу HTTP/1.1; отображать только Вашу собственную страницу; выводить на экран тексты полученных запросов и формируемых ответов; обеспечить многопользовательский режим доступа к странице, для чего функцию обработки данных надо оформить в виде отдельного потока. Алгоритм работы сервера следующий:
2. инициализируем, создаем и настраиваем основной сокет;
3. связываем сокет с IP-адресом и портом, переводим его в состояние прослушивания и организуем бесконечный цикл с ожиданием запроса на подключение;
4. в случае поступления запроса от браузера создаем для принятого запроса новый клиентский сокет, делаем лексический разбор полученного запроса, определяем наличие (или отсутствие) требуемого ресурса и формируем ответ в соответствии с протоколом HTTP;
5. отправляем ответ через клиентский сокет;
6. закрываем клиентский сокет;
7. возвращаемся в начало цикла на пункт 4.

Сообщения, передаваемые клиенту, должны иметь строгий формат, определенный протоколом HTTP.

Например, браузер может передать на сервер следующие строки:

// запрос данных с сервера (/–путь к файлу, по умолчанию корневой каталог)

GET / HTTP/1.1

Accept: \*/\*

Accept-Language: ru

Accept-Encoding: gzip, deflate X4

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

Host: ninja:3000

Connection: Keep-Alive

Значащей строкой здесь является GET / HTTP/1.1. В случае успешного выполнения запроса сервер посылает ответ в виде стартовой строки, набора заголовков и содержимого требуемого файла:

HTTP/1.0 200 OK // протокол, код ошибки, результат

Server: CNAI Demo Web Server // имя web сервера

Content-Length: 235 // длина html файла

Content-Type: text/html // тип файла – текстовый html

Об остальных типах сообщений можно узнать, например, из RFC2616.

1. Просмотрите web-страницу, разработанную в п.1, с помощью браузера. Web-сервер и браузер должны быть загружены на разных компьютерах. Обращение к Web -серверу проводится путем ввода в адресную строку браузера URL по следующему шаблону: *http://имя\_хоста:номер\_порта*, например, [*http://pc-208-05:2008*](http://pc-208-05:2008). Пример программы web-сервера приведен в приложении 5.
2. Включите в браузере режим разработчика, подключитесь к серверу *moodle.ami.nstu.ru* и скачайте файл с календарным планом. Определите текст этого запроса и затраты времени на выполнение всех его этапов.

4. Контрольные вопросы

1. Интерфейс Web-браузера.
2. Основные типы документов Web, преимущества и недостатки документов каждого типа
3. Идентификация Web-страниц.
4. Поясните различие терминов URL и URI*.*
5. Как происходит взаимодействие между Web-браузером и сервером?
6. В чем назначение протокола HTTP, приведите его основные операции?
7. Как формируется запрос Web-браузера?
8. Приведите пример заголовка HTTP, возвращаемого сервером.
9. Структура Web – браузера.
10. Кэширование в Web-браузерах.

# Список источников

1. Олифер В.Г., Олифер Н. А. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 4-е изд. – СПб.: Питер, 2010
2. Васин Н. Основы построения сетей пакетной коммутации. Транспортный уровень моделей OSI, TCP/IP [https://www.intuit.ru/studies/courses/3645/887/info](https://www.intuit.ru/studies/courses/3645/887/info%20) , дата обращения 15.01.2020
3. Берлин А. Н. Основные протоколы Интернет: Учебное пособие / А.Н. Берлин — М.: Национальный открытый университет «ИНТУИТ», 2016. – 603 с. – <https://www.intuit.ru/studies/courses/2/2/info>
4. Донин С.Н. Электронная почта: руководство пользователя **. –**[http://citforum.ru/pp/email/index.shtml](http://citforum.ru/pp/email/index.shtml%20)[Электронный ресурс], дата обращения 15.01.2020
5. Леонов А. SGVsbG8gd29ybGQh или история base64. –https://habr.com/ru/post/88077/**.** [Электронный ресурс], дата обращения 15.01.2020
6. Некрасов С. И., Некрасова Н. А. Философия науки и техники: тематический словарь. – Орёл: ОГУ. 2010. URL http://library.miit.ru/methodics/16012012/10%20-%202266.pdf
7. Просмотр служебных заголовков писем. – RU Center. – <https://www.nic.ru/help/prosmotr-sluzhebnyh-zagolovkov-pisem_6806.html>. –[Электронный ресурс], дата обращения 15.01.2020
8. Структура пакета TCP. – <https://webhamster.ru/mytetrashare/index/mtb0/1501768410v2kmovcru6>. – [Электронный ресурс], дата обращения 15.01.2020
9. Федорук В. Г. Протоколы сетевого взаимодействия TCP/IP. – <https://www.opennet.ru/docs/RUS/tcpip/#c4_5_tcp_scenario> [Электронный ресурс] , дата обращения 15.01.2020
10. Храмцов П. Организация и администрирование почтовых и файловых серверов Internet. – <http://citforum.ru/internet/servers/index.shtml>. – [Электронный ресурс], дата обращения 15.01.2020
11. NetEmul [Электронный ресурс] : рук.разработчика / П. Семёнов, А. Омилаева. – Режим доступа: <http://netemul.sourceforge.net/ruindex.html>. дата обращения 15.01.2020
12. Cisco Systems Cisco Packet Tracer [Электронный ресурс] : рук.разработчика / Cisco Systems. – Режим доступа: <https://www.netacad.com/about-networking-academy/packet-tracer/>. дата обращения 15.01.2020
13. GNS3 [Электронный ресурс] : рук.разработчика / OpenSource. – Режим доступа: <https://www.gns3.com/software>. дата обращения 15.01.2020
14. Wireshark. http://www.wireshark.org/, дата обращения 15.01.2020.
15. Wireshark Display Filter Reference. http://www.wireshark.org/docs/dfref/, дата обращения 13.04.2016.
16. Wireshark Trace Files. <http://www.wiresharkbook.com> , дата обращения 15.01.2020
17. Tali Garsiel, Paul Irish Как работает браузер или что происходит за кулисами современных браузеров.– <https://webknowledge.ru/kak-rabotaet-brauzer-chto-proishodit-za-kulisami-sovremennyh-brauzerov/> дата обращения 25.01.2020
18. Рыжков Е. Как работает браузер: архитектура.– <http://xiper.net/learn/also-need-to-know/how-does-a-browser-architecture> дата обращения 25.01.2020
19. Рыжков Е. Как работает браузер: обмен данными с сервером.– <http://xiper.net/learn/also-need-to-know/how-does-a-browser-communicate-with-the-server> дата обращения 25.01.2020
20. Протокол HTTP. – Уральский федеральный университет.– <https://iit-web-lectures.readthedocs.io/ru/latest/www/http.html#id9> дата обращения 25.01.2020
21. Коды статусов ответа по протоколу HTTP. – <https://htmlweb.ru/other/http_code.php> дата обращения 26.10.2020.
22. Основы организации сетей Cisco, том 1. – М.: Издательский дом «Вильямс, 2002 Режим доступа: <http://sokolovdenis.ru/Portals/0/Documents/Cisco--Основы-организации-сетей-1.pdf>. дата обращения 12.02.2021

# Приложение 1

Пример SMTP-клиента для работы по незащищенному соединению

import socket

host = 'cn.ami.nstu.ru'

port = 25

client = socket.socket()

client.connect((host, port))

print("connect to %s through %s port" % (host, port))

print("CLIENT: EHLO localhost CRLF")

client.send('EHLO localhost \r\n'.encode())

data = client.recv(1024)

print("Server: %s" % data.decode('utf-8'))

print("CLIENT: MAIL FROM:b4@cn.ami.nstu.ru CRLF")

client.send('MAIL FROM:b4@cn.ami.nstu.ru \r\n'.encode())

data = client.recv(1024)

print("Server: %s" % data.decode('utf-8'))

print("CLIENT: RCPT TO:b10@cn.ami.nstu.ru CRLF")

client.send('RCPT TO:b10@cn.ami.nstu.ru \r\n'.encode())

data = client.recv(1024)

print("Server: %s" % data.decode('utf-8'))

print("CLIENT: DATA CRLF")

client.send('DATA \r\n'.encode())

data = client.recv(1024)^

print("Server: %s" % data.decode('utf-8'))

message = 'FROM: b4@cn.ami.nstu.ru\r\n' + \

'TO: b10@cn.ami.nstu.ru\r\n' + \

'SUBJECT: Test message\r\n'

print("CLIENT: %s CRLF" % message)

client.send(message.encode())

print("CLIENT: ")

client.send(''.encode())

print("CLIENT: This is a test message! CRLF")

client.send('This is a test message!\r\n'.encode())

print("CLIENT: . CRLF")

client.send('.\r\n'.encode())

data = client.recv(1024)

print("Server: %s" % data.decode('utf-8'))

print("CLIENT: QUIT CRLF")

client.send('QUIT\r\n'.encode())

data = client.recv(1024)

print("Server: %s" % data.decode('utf-8'))

client.close()

# ПРИЛОЖЕНИЕ 2

Пример реализации SMTP-клиента с защищенным соединением

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Net.Mail;

using System.Net;

using System.Net.Sockets;

using System.Net.Security;

using System.IO;

namespace SMTPClient

{

class Program

{

static FileStream fstream;

static NetworkStream stream;

static SslStream sslstream;

static void Main(string[] args)

{

StartSTMPSession();

}

public static void WriteToConsole(string type, string message)

{

Console.WriteLine(type + message);

byte[] array = System.Text.Encoding.Default.GetBytes(type + message);

fstream.Write(array, 0, array.Length);

}

public static void WriteToServerSSL(string message)

{

byte[] data;

string response = message;

data = System.Text.Encoding.UTF8.GetBytes(response);

sslstream.Write(data, 0, data.Length);

WriteToConsole("Client:\t", message);

}

public static void WriteToServer(string message)

{

byte[] data;

string response = message;

data = System.Text.Encoding.UTF8.GetBytes(response);

stream.Write(data, 0, data.Length);

WriteToConsole("Client:\t", message);

}

public static void ReadFromServerSSL()

{

System.Threading.Thread.Sleep(3000);

byte[] data = new byte[256];

int bytes = sslstream.Read(data, 0, data.Length); // получаем кол-во считанных байтов

string message = Encoding.UTF8.GetString(data, 0, bytes);

WriteToConsole("Server:\t", message);

}

public static void ReadFromServer()

{

System.Threading.Thread.Sleep(3000);

byte[] data = new byte[256];

int bytes = stream.Read(data, 0, data.Length); // получаем количество считанных байтов

string message = Encoding.UTF8.GetString(data, 0, bytes);

WriteToConsole("Server:\t", message);

}

public static void StartSTMPSession()

{

fstream = new FileStream("log.txt", FileMode.OpenOrCreate);

TcpClient tcpClient = new TcpClient();

Console.Write(">> Enter SMTP server: "); string server = Console.ReadLine(); Console.WriteLine();

Console.Write(">> Enter port: "); int port = Int32.Parse(Console.ReadLine()); Console.WriteLine();

tcpClient.Connect(server, port);

stream = tcpClient.GetStream();

ReadFromServer();

// приветствуем

WriteToServer("EHLO 127.0.0.1\r\n");

ReadFromServer();

// хотим авторизоваться

WriteToServer("STARTTLS\r\n");

ReadFromServer();

sslstream = new SslStream(tcpClient.GetStream());

sslstream.AuthenticateAsClient("smtp.gmail.com");

WriteToServerSSL("AUTH LOGIN\r\n");

ReadFromServerSSL();

// ввод логина

Console.Write(">> Enter login: "); string login = Console.ReadLine(); Console.WriteLine();

string encodeLogin = System.Convert.ToBase64String(System.Text.Encoding.ASCII.GetBytes(login));

WriteToServerSSL(encodeLogin+"\r\n");

ReadFromServerSSL();

// ввод пароля

Console.Write(">> Enter password: "); string password = Console.ReadLine(); Console.WriteLine();

string encodePassword = System.Convert.ToBase64String(System.Text.Encoding.ASCII.GetBytes(password));

WriteToServerSSL(encodePassword + "\r\n");

ReadFromServerSSL();

// начало транзакции

WriteToServerSSL("MAIL FROM: <"+login+">\r\n");

ReadFromServerSSL();

// указание получателя

Console.Write(">> Enter for whom the message: "); string to = Console.ReadLine(); Console.WriteLine();

WriteToServerSSL("RCPT TO: <"+to+">\r\n");

ReadFromServerSSL();

// текст сообщения

WriteToServerSSL("DATA\r\n");

ReadFromServerSSL();

// заголовок

WriteToServerSSL("FROM: " + login + "\r\n");

WriteToServerSSL("TO: " + to + "\r\n");

Console.Write(">> Enter the header of the message: "); string header = Console.ReadLine(); Console.WriteLine();

WriteToServerSSL("SUBJECT: "+header +"\r\n");

// пустая строка

WriteToServerSSL("\r\n");

// текст сообщения

Console.WriteLine(">> Enter the message (end with '.'): "); Console.WriteLine();

Console.Write("message: "); string message = Console.ReadLine(); Console.WriteLine();

while (message != ".")

{

WriteToServerSSL(message+"\r\n");

Console.Write("message: "); message = Console.ReadLine(); Console.WriteLine();

}

WriteToServerSSL(".\r\n");

ReadFromServerSSL();

WriteToServerSSL("QUIT\r\n");

ReadFromServerSSL();

stream.Close();

sslstream.Close();

tcpClient.Close();

Console.Write("Enter Escape to exit...");

while (true)

{

if (Console.ReadKey().Key == ConsoleKey.Escape)

{

Environment.Exit(0);

}

}

}

}

}

# Приложение 3

Пример реализации POP3-клиента

Здесь клиент подключается к почтовому серверу, установленному на компьютере в локальной сети с IP – адресом 192.168.31.139.

#include "stdafx.h"

#include <windows.h>

#include <winsock.h>

#pragma comment (lib,"Ws2\_32.lib")

#include <atlenc.h>

#include <iostream>

#include <string>

WSADATA WsaData;

SOCKET s;

struct sockaddr\_in addr;

hostent \*d\_addr;

char text[1024];

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

s = socket(AF\_INET, SOCK\_STREAM, 0);

//Подключаемся к серверу

d\_addr = gethostbyname("192.168.31.139");

addr.sin\_family = AF\_INET;

addr.sin\_addr.s\_addr = \*((unsigned long \*)d\_addr->h\_addr);

addr.sin\_port = htons(110);

recv(s, text, sizeof(text), 0);

cout << "recv - " << text << endl;

//Передаем имя пользователя

strcpy(text, "USER user@192.168.31.139\r\n");

send(s, text, strlen(text), 0);

cout << "send - " << text << endl;

recv(s, text, sizeof(text), 0);

cout << "recv - " << text << endl;

//Передаем пароль

strcpy(text, "PASS 123456\r\n");

send(s, text, strlen(text), 0);

cout << "send - " << text << endl;

recv(s, text, sizeof(text), 0);

cout << "recv - " << text << endl;

if (text[0] == '+')

{

//Запрашиваем список писем

strcpy(text, "STAT\r\n");

send(s, text, strlen(text), 0);

cout << "send - " << text << endl;

recv(s, text, sizeof(text), 0);

cout << "recv - " << text << endl;

//Запрашиваем четвертое письмо из списка

strcpy(text, "RETR 4\r\n");

send(s, text, strlen(text), 0);

cout << "send - " << text << endl;

recv(s, text, sizeof(text), 0);

cout << "recv - " << text << endl;

}

//Закрываем сессию

strcpy(text, "QUIT\r\n ");

send(s, text, strlen(text), 0);

cout << "send - " << text << endl;

recv(s, text, sizeof(text), 0);

cout << "recv - " << text << endl;

}

closesocket(s);

system("PAUSE");

return 0;

}

# ПРИЛОЖЕНИЕ 4

Пример простой HTML-страницы

На этой странице выводится заголовок и таблица, в которой содержится одна запись с именем файла, оформленным в виде ссылки, и дата загрузки этого файла в виде константы.

<!DOCTYPE html>

<html>

<head>

<meta charset="RU">

<title>Сетевые технологии</title>

</head>

<body>

<table>

<caption>Пример простой HTML-страницы</caption>

<tr>

<th>Имя файла</th>

<th>Дата загрузки</th>

<tr>

<td><a href=".\post\_server.pdf"> Документ типа .pdf </a><td>

<td>20.10.2020<td>

</tr>

</table>

</body>

# ПРИЛОЖЕНИЕ 5

Пример реализации WEB-сервера

#include "Server.h"

void createResponsible(char \*msg, int client\_socket)

{

int cur;

string path = "";

for (cur = 5; msg[cur] != ' '; ++cur)

{

path += msg[cur];

}

if (path == "")

{

path = "index.html";

}

stringstream response;

stringstream response\_body;

ifstream web(path);

if (!web.is\_open())

{

cerr << "Error open file" << endl;

return;

}

response\_body << web.rdbuf();

response << "HTTP/1.1 200 OK\r\n"

<< "Version: HTTP/1.1\r\n"

<< "Content-Type: text/html; charset=utf-8\r\n"

<< "Content-Length: " << response\_body.str().length()

<< "\r\n\r\n"

<< response\_body.str();

if (send(client\_socket, response.str().c\_str(), response.str().length(), 0) = = SOCKET\_ERROR) {

cerr << "Send failed: " << WSAGetLastError() << endl;

}

}

MyWebServer::MyWebServer(const char\* port)

{

WSADATA wsaData;

if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {

throw exception("WSAStartup cant started");

}

addr = NULL;

struct addrinfo hints;

ZeroMemory(&hints, sizeof(hints));

hints.ai\_family = AF\_INET;

hints.ai\_socktype = SOCK\_STREAM;

hints.ai\_protocol = IPPROTO\_TCP;

hints.ai\_flags = AI\_PASSIVE;

if (getaddrinfo("0.0.0.0", port, &hints, &addr) != 0) {

WSACleanup();

throw exception("Getaddrinfo failed");

}

listen\_socket = socket(addr->ai\_family, addr->ai\_socktype, addr->ai\_protocol);

if (listen\_socket == INVALID\_SOCKET) {

freeaddrinfo(addr);

WSACleanup();

throw exception("Error listen socket");

}

if (bind(listen\_socket, addr->ai\_addr, (int)addr->ai\_addrlen) == SOCKET\_ERROR)

{

freeaddrinfo(addr);

closesocket(listen\_socket);

WSACleanup();

throw exception("Error binded");

}

if (listen(listen\_socket, SOMAXCONN) == SOCKET\_ERROR)

{

closesocket(listen\_socket);

WSACleanup();

throw exception("Error listened");

}

}

DWORD WINAPI process(LPVOID p)

{

char msg[MyWebServer::MAX\_LENGTH];

int client\_socket = \*((int\*)p);

delete p;

int res = recv(client\_socket, msg, MyWebServer::MAX\_LENGTH, 0);

if (res == SOCKET\_ERROR)

{

cerr << "Recv failed: " << WSAGetLastError() << endl;

return 1;

}

else if (res == 0)

{

cerr << "Connection closed...\n";

return 1;

}

msg[res] = '\0';

createResponsible(msg, client\_socket);

closesocket(client\_socket);

return 0;

}

int MyWebServer::Run()

{

int \*client\_socket = new int(accept(listen\_socket, NULL, NULL));

if (\*client\_socket == INVALID\_SOCKET) {

cerr << "Accept failed: " << WSAGetLastError() << endl;

return 1;

}

DWORD threadID;

CreateThread(NULL, NULL, process, client\_socket, NULL, &threadID);

}

void MyWebServer::Stop()

{

closesocket(listen\_socket);

freeaddrinfo(addr);

WSACleanup();

}

#pragma once

#include <iostream>

#include <sstream>

#include <fstream>

#include <string>

#include <mutex>

#define \_WIN32\_WINNT 0x501

#include <WinSock2.h>

#include <WS2tcpip.h>

#pragma comment(lib, "Ws2\_32.lib")

using namespace std;

class MyWebServer

{

private:

struct addrinfo\* addr;

int listen\_socket;

public:

static const int MAX\_LENGTH = 1024;

MyWebServer(const char\* port);

int Run();

void Stop();

};