ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ

Кафедра МСИБ

«Анализ трафика мультисервисных сетей»

Учебное пособие для проведения лабораторно-практических занятий

Составители: д.т.н., профессор Лихтциндер Б.Я.
к.т.н., доцент Поздняк И. С.

Редактор: д.т.н., профессор Карташевский В. Г.

Рецензент: д.т.н., профессор Васин Н. Н.

Самара, 2013

Содержание

1 Трафик мультисервисных сетей

Развитие сетей связи в перспективе будет происходить в рамках реализации основных положений концепции сетей следующего поколения NGN (Next Generation Network). Она формирует правила построения сетей связи, обеспечивающих предоставление неограниченного набора услуг с заданными характеристиками качества [1].

Концепция NGN родилась не на пустом месте. Ее основное положение обобщают опыт реализации наиболее успешных телекоммуникационных проектов, главным образом, сети Интернет и сетей подвижной связи.

Необходимо отметить следующие важные положения.

– Изменение схемы формирования сетей инфраструктуры. Используемая ранее монолитная сетевая инфраструктура становится многослойной. Каждый слой отвечает за решение определенного круга задач.

– Трансформация понятия услуги. Названия отдельных технологий и услуг обезличиваются – пользователю необходим один вид сервиса под названием «соединение с сетью», подразумевающий возможность получения мультимедийной информации в разнообразных сочетаниях, определяемых абонентом, в соответствии со своими индивидуальными запросами по качеству и скорости.

– Доминирующая роль протокола IP. Дешевизна решений на базе IP при интеграции услуг и пользовательских групп.

Упрощенная схема многоуровневой сетевой инфраструктуры, отражающая основные положения концепции NGN, показана на рис. 1.1.

Можно рассматривать два сценария построения сети:

– построение сети с избытком. Передаточного ресурса и минимум контроля за сетью.

– при реализации второго сценария применяются более совершенные средства контроля и управления за процессом передачи информации. Требуемые характеристики качества работы сети достигаются в результате дифференцированного обслуживания пользователей в соответствии с заявленными показателями. Канальный ресурс распределяется более эффективно.

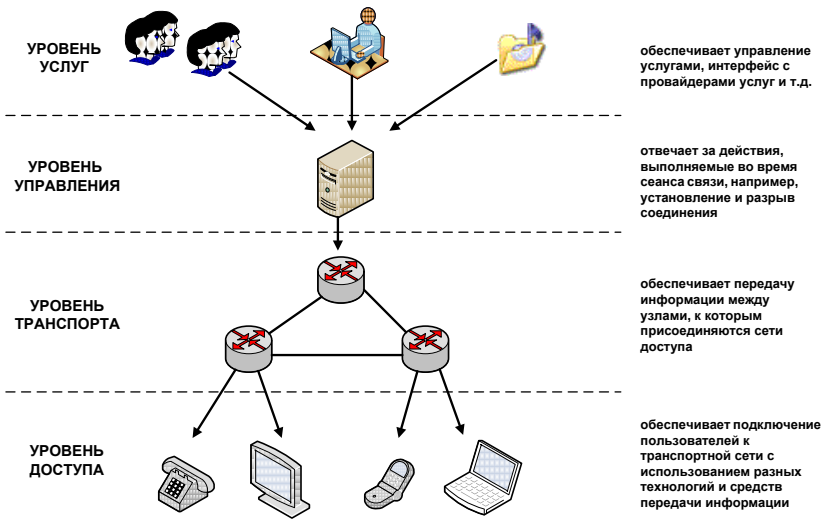


Рис. 1.1 - Схема сетевой инфраструктуры перспективных сетей связи

Расширение видов услуг, услуги машина-машина, сенсорные сети, мультимедийность трафика приведут к развитию второго сценария.

Отмеченные тенденции делают актуальными разработку средств оптимизации планирования сетевой инфраструктуры на базе внедрения более совершенных средств контроля за процессом передачи информации.

Классы сервиса и приоритеты обслуживания трафика

Требования к условиям передачи естественным образом вытекают из характера предоставляемых услуг. Соответствующий перечень классов сервисов выглядит следующим образом.

Класс 0 – потоки реального времени, отличающиеся высокой степенью интерактивности и чувствительные к вариации задержки (высококачественная пакетная телефония и видеоконференц-связь).

Класс 1 – потоки реального времени, интерактивные и чувствительные к вариации задержки (пакетная телефония, видеоконференц-связь).

Класс 2 – транзакции данных, отличающиеся высокой степенью интерактивности (сигнализация).

Класс 3 – транзакции данных, интерактивные.

Класс 4 – потоки, чувствительные к потере информации в процессе ее передачи по сети (массивные данные, потоковое видео).

Класс 5 – традиционные приложения IP – сетей с характеристиками передачи по умолчанию.

В таблице 1.1 предоставлены верхние границы характеристик доставки IP – пакетов для каждого класса сервиса [1].

Таблица 1.1

Характеристики доставки IP – пакетов	Классы качества передачи информационных потоков					
	0	1	2	3	4	5
Задержка доставки IP-пакета, IPTD	100мс	400мс	100мс	400мс	1с	Не опр.
Вариация задержки доставки IP-пакета, IPDV	50мс	50мс	Не опр.	Не опр.	Не опр.	Не опр.
Для потерянных IP- пакетов, IPLR	1*10-3	1*10-3	1*10-3	1*10-3	1*10-3	Не опр.
Доля IP-пакетов, переданных с ошибкой, IPER	1*10-4	1*10-4	1*10-4	1*10-4	1*10-4	Не опр.

Здесь сервисы делятся в зависимости от характеристик передачи пакетов. С точки зрения восприятия пользователей трафик делится на:

– трафик реального времени (голосовая связь, видеоконференция)

– трафик интерактивной передачи данных (обмен веб-страницами)

– трафик, терпимый к задержкам (передача электронной почты)

При этом используются характеристики:

– для потерянных пакетов, определяемая как доля IP-пакетов, отброшенных из-за блокировки в процессе их передачи по сети;

– доля отказов в установлении соединения, определяемая как доля заявок, составляющих рассматриваемый поток, для которых механизм управления допуском отказал в резервировании канального ресурса в количестве, необходимом для обслуживания поступившей заявки;

– скорость передачи информации пользователя, определяемая как отношение объема успешно переданной информации к периоду наблюдения и измеряемая в битах в секунду.

Для обеспечения указанных характеристик, к обслуживанию трафика различных классов предъявляются различные требования.

Для трафика, допускающего малые задержки, необходимо установить некоторые преимущества, называемые «приоритетами». Приоритеты заявок характеризуются целыми положительными числами, причем более высокому приоритету ставится в соответствие меньшее число.

Если приоритеты учитываются только в моменты выбора заявки, то их называют относительными. Если же выбранная заявка наивысшего приоритета прерывает уже начавшееся обслуживание заявки более низкого приоритета, то такая дисциплина обслуживания называется обслуживание с абсолютным приоритетом.

В телекоммуникационных сетях обычно используется обслуживание с относительным приоритетом.

Для заявок каждого приоритета образуется отдельная очередь. Заявка из очереди, соответствующей более низкому приоритету, выбирается на обслуживание лишь в случае, когда все очереди более высоких приоритетов оказываются пустыми.

Управление процессом передачи сообщений

Одной из важнейших функций сети, направленной на повышение загрузки канального ресурса и улучшение качества обслуживания абонента, является управление процессом передачи сообщений. Оно реализуется в разных формах и зависит от степени детализации, используемой при анализе информационных потоков. Перечень возможных решений показан в таблице 2 с указанием шкалы времени, на которой соответствующее решение принимается.

Передача производится пакетами (или ячейками)

- без установления соединения
- с установлением виртуального соединения.

В этом случае требуется контроль за установлением соединения.

Одной из важнейших задач, относящихся к управлению сетью, является выполнение действий, направленных на устранение блокировок.

Таблица 1.2.

Управленческие решения, принимаемые сетью при организации процесса передачи сообщений, в зависимости от шкалы времени [1]

Управляющее решение	Шкала времени
Отброс или понижение качества обслуживания для ячеек или пакетов, не удовлетворяющих принятому заранее соглашению по трафику (policing)	Время между моментами поступления ячеек, пакетов
Задержка передачи для части ячеек или пакетов, направленная на улучшение характеристик качества передачи (shaping)	
Организация и планирование очередей для ячеек или пакетов (queueing and scheduling)	
Контроль доставки пакетов или ячеек при динамическом изменении потребляемого ресурса и уменьшения блокировок (flow control)	Время распространения сигнала в прямом и обратном направлениях
Контроль приема заявок на выделение канального ресурса с целью пропуска пользовательского трафика, маршрутизация трафика (call admission control, routing)	Время между последовательными поступлениями заявок
Принятие управляющих решений на сети для улучшения характеристик обслуживания (network management)	Минуты, часы, дни
Принятие решений по изменению трафиков (pricing policy)	Месяцы

Избыток трафика, который ввел сеть в состояние перегрузки, можно:

– заблокировать, т.е. удалить соответствующие пакеты из передачи (как правило, данное действие приводит к повторной передачи заблокированных пакетов, что только усугубляет ситуацию перезагрузки);

– доставить адресату с худшими показателями качества, например, за большее время или с большей долей потерянных пакетов;

– доставить адресату за большую стоимость.

Последнее из упомянутых действий выглядит предпочтительней, поскольку оно не уменьшает доход, а также не ухудшает значения показателей обслуживания.

2 Стек протоколов TCP/IP

Для обеспечения сетевых взаимодействий между сетевыми объектами каждый из объектов должен понимать другой объект, говорить с ним на одном языке. Другими словами, сетевые объекты должны удовлетворять определенному ***протоколу, набору правил общения***. Поскольку сетевое общение между сетевыми объектами является достаточно сложным процессом, описываемым всеми уровнями модели OSI, то и набор правил для организации общения достаточно сложен. Для реализации данного набора правил разработаны наборы, или ***стеки,*** взаимосвязанных и логически сгруппированных протоколов. Существует достаточно большое количество стеков протоколов, однако наиболее широкое распространение в настоящее время получил ***стек*** **TCP/IP.**

В стеке TCP/IP определены четыре уровня. Каждый из них несет на себе некоторую долю нагрузки по решению основной задачи-организации надежной и производительной работы составной сети, части которой построены на основе разных сетевых технологий.

Основное, что отличает Интернет от других сетей - это ее протоколы — TCP/IP. Вообще, термин TCP/IP обычно означает все, что связано с протоколами взаимодействия между компьютерами в Интернете. Он окатывает целое семейство протоколов, прикладные программы и даже саму сеть. TCP/IP — это технология межсетевого взаимодействия. Если речь идет о глобальной сети, объединяющей множество сетей с технологией TCP/IP, то ее называют Интернетом.

Так как стек TCP/IP был разработан до появления модели взаимодействия открытых систем OSI, то, хотя он также имеет многоуровневую структуру, соответствие уровней стека TCP/IP уровням модели OSI достаточно условно [2].

Прикладной уровень

Прикладной уровень стека TCP/IP, соответствует представительному, прикладному и сеансовому уровням модели OSI.

Прикладной уровень объединяет все службы, представляемые системой пользовательским приложениями. Этот уровень содержит протоколы, определяющие детали взаимодействия с каждым конкретным приложением, наиболее распространённые (табл. 2.1):

SNMP (Simple Network Management Protocol) - простой протокол управления сетью;

FTP (File Transfer Protocol) - протокол передачи файлов;

Telnet-удаленный терминал;

SMTP (Simple Mail Transfer Protocol) - простой протокол передачи электронной почты.

Таблица 2.1

<table>
<tr><td>7
6</td><td colspan="2">WWW</td><td>SNMP</td><td>FTP</td><td>Telnet</td><td>TFTP</td><td>SMTP</td><td>I</td></tr>
<tr><td>5
4</td><td colspan="3">TCP</td><td colspan="3">UDP</td><td>II</td></tr>
<tr><td>3</td><td colspan="2">IP</td><td>ICMP</td><td>RIP</td><td colspan="2">OSPF</td><td>ARP</td><td>III</td></tr>
<tr><td>2
1</td><td colspan="7">Ethernet, Gigabit Ethernet, Token Ring,
PPP, FDDI, X.25, SLIP, Frame Relay...</td><td>IV</td></tr>
<tr><td>уровни OSI</td><td colspan="7">Протоколы</td><td>уровни TCP/IP</td></tr>
</table>

Основной уровень стека TCP/IP

Этот уровень уровень отвечает за транспортировку потока данных между двумя компьютерами и обеспечивает работу прикладных уровней. В стеке протоколов TCP/IP два основных транспортных протокола:

TCP (Transmission Control Protocol - протокол управления передачей). Основная задача протокола TCP - обеспечение надёжной передачи данных между двумя системами. Надёжная доставка данных на транспортном уровне гарантирует протоколам прикладного уровня целостность обрабатываемого байтового потока. TCP позволяет без ошибок доставлять сформированный на одном из компьютеров поток байт в любой другой компьютер, входящий в составную сеть. TCP делит поток байт на части - сегменты и передает их нижележащему уровню межсетевого взаимодействия. После того, как эти сегменты

будут доставлены в пункт назначения, протокол TCP снова соберет их в непрерывный поток байт.

UDP (User Datagram Protocol - протокол пользовательских дейтаграмм). Протокол UDP обеспечивает негарантированную доставку дейтаграмм от одной системы к другой. Отсутствие механизмов подтверждения и управления скоростью передачи делают протокол UDP менее надёжным, но значительно повышает скорость передачи данных, что во многих случаях является более важным моментом. За надёжность передачи данных с использованием протокола UDP отвечает протокол прикладного уровня.

Уровень межсетевого взаимодействия

Стержнем всей архитектуры является уровень межсетевого взаимодействия, или сетевой уровень, который реализует концепцию передачи пакетов в режиме без установления соединений, то есть дейтаграммным способом. рациональным. Этот уровень также называют уровнем Internet, указывая, тем самым, на основную его функцию - передачу данных через составную сеть.

Основным протоколом уровня (в терминологии модели OSI) в стеке TCP/IP является протокол IP. Этот протокол изначально проектировался как протокол передачи пакетов в составных сетях, состоящих из большого количества локальных сетей. Поэтому протокол IP хорошо работает в сетях со множеством топологий, рационально используя наличие в них подсистем и экономно расходуя пропускную способность низко скоростных линий связи. Так как протокол IP является дейтаграммным протоколом, он не гарантирует доставку пакетов до узла назначения, но старается это сделать.

. Вспомогательные протоколы межсетевого уровня стека TCP/IP:

- ICMP (Internet Control Message Protocol) - протокол контрольных сообщений;
- ARP (Address Resolution Protocol) - протокол преобразования адресов;
- RARP (Reverse Address Resolution Protocol) - протокол обратного преобразования адресов;
- Протоколы маршрутизации, например, RIP, OSPF и др.

Уровень сетевых интерфейсов

Уровень сетевого интерфейса, соответствует двум уровням модели OSI (канальному и физическому) и определяет протоколы взаимодействия с физической средой передачи данных и протоколы,

описывающие передачу данных при помощи определённых технологий (Ethernet, ATM, PPP и т.д.).

Уровень сетевых интерфейсов в протоколах TCP/IP не регламентируется, но он поддерживает все популярные стандарты физического и канального уровней: для локальных сетей - это Ethernet, TokenRing, FDDI, Fast Ethernet, Gigabit Ethernet, 100VG-AnyLAN, для глобальных сетей - протоколы соединений «точка-точка» SLIP и PPP, протоколы территориальных сетей с коммутацией пакетов X.25, Frame Relay. Разработана также специальная спецификация, определяющая использование технологии ATM в качестве транспорта канального уровня.

3 Протоколы сети Интернет

3.1 WWW(Протокол HTTP)

WWW (World Wide Web) - распределенная система, предоставляющая доступ к связанным между собой документам, расположенным на различных компьютерах, подключенных к Интернету. Протокол, принятый в WWW, называется HyperText Transfer Protocol [3]**,** сокращенно **–** HTTP**.** Это протокол высокого уровня (а именно, уровня приложений), обеспечивающий необходимую скорость передачи данных, требующуюся для распределенных информационных систем гипермедиа.

Практические информационные системы требуют большего, чем примитивный поиск, модификация и аннотация данных. HTTP/1.0 предоставляет открытое множество методов, которые могут быть использованы для указания целей запроса. Они построены на дисциплине ссылок, где для указания ресурса, к которому должен быть применен данный метод, используется Универсальный Идентификатор Ресурсов (Universal Resource Identifier - URI), в виде местонахождения (URL) или имени (URN). Формат сообщений сходен с форматом Internet Mail или Multipurpose Internet Mail Extensions (MIME -- Многоцелевое Расширение Почты Internet).

HTTP 1.0 используется также для коммуникаций между различными пользовательскими просмотрщиками и шлюзами, дающими гипермедиа доступ к существующим Internet протоколам, таким как SMTP, NNTP, FTP, Gopher и WAIS. HTTP/1.0 разработан, чтобы позволять таким шлюзам через proxy серверы, без какой-либо потери передавать данные с помощью упомянутых протоколов более ранних версий.

Общая Структура.

HTTP основывается на парадигме запросов/ответов [2]. Запрашивающая программа (обычно она называется клиент) устанавливает связь с обслуживающей программой-получателем (обычно называется сервер) и посылает запрос серверу в следующей форме:

- метод запроса,
- URI,
- версия протокола,
- за которой следует MIME-подобное сообщение, содержащее управляющую информацию запроса, информацию о клиенте и, может быть, тело сообщения.

Сервер отвечает сообщением, содержащим строку статуса (включая версию протокола и код статуса - успех или ошибка), за которой следует MIME-подобное сообщение, включающее в себя информацию о сервере, метаинформацию о содержании ответа, и, вероятно, само тело ответа. Следует отметить, что одна программа может быть одновременно и клиентом и сервером. Использование этих терминов в данном тексте относится только к роли, выполняемой программой в течение данного конкретного сеанса связи, а не к общим функциям программы.

В Internet коммуникации обычно основываются на TCP/IP протоколах. Для WWW номер порта по умолчанию -- TCP:80, но также могут быть использованы и другие номера портов -- это не исключает возможности использовать HTTP в качестве протокола верхнего уровня.

Для большинства приложений сеанс связи открывается клиентом для каждого запроса и закрывается сервером после окончания ответа на запрос. Тем не менее, это не является особенностью протокола. И клиент, и сервер должны иметь возможность закрывать сеанс связи, например, в результате какого-нибудь действия пользователя. В любом случае, разрыв связи, инициированный любой стороной, прерывает текущий запрос, независимо от его статуса.

3.2 Протокол SNMP

SNMP (Simple Network Management Protocol — простой протокол управления сетями) — это протокол управления сетями связи на основе архитектуры UDP.

На основе концепции TMN в 1980—1990 гг. различными органами стандартизации был выработан ряд протоколов управления сетями передачи данных с различным спектром реализации функций TMN. К одному из типов таких протоколов управления относится SNMP.

Также эта технология, призванная обеспечить управление и контроль за устройствами и приложениями в сети связи путём обмена управляющей информацией между агентами, располагающимися на сетевых устройствах, и менеджерами, расположенными на станциях управления. SNMP определяет сеть как совокупность сетевых управляющих станций и элементов сети (главные машины, шлюзы и маршрутизаторы, терминальные серверы), которые совместно обеспечивают административные связи между сетевыми управляющими станциями и сетевыми агентами.

Определяет всего пять типов сообщений, которыми обмениваются менеджер и клиент.

- Получить значение одной или нескольких переменных: оператор *get-request.*
- Получить следующую переменную после этой или несколько указанных переменных: оператор *get-next-request.*
- Установить значение одной или нескольких переменных: оператор *set-request.*
- Выдать значение одной или нескольких переменных: оператор *get-response*. Это сообщение возвращается агентом менеджеру в ответ на операторы get-request, get-next-request и set-request.
- Уведомить менеджера, когда что-либо произошло с агентом: оператор *trap.*

Первые три сообщения отправляются от менеджера к агенту, а последние два от агента к менеджеру. На рисунке 1 приведены все пять операторов.

Так как четыре из пяти snmp сообщений реализуются простой последовательностью запрос-отклик [4] (менеджер отправляет запрос, а агент возвращает отклик), SNMP используют UDP. Это означает, что запрос от менеджера может не прибыть к агенту, а отклик от агента может не прибыть к менеджеру. В этом случае менеджер, возможно, отработает тайм-аут и осуществит повторную передачу.

На рисунке 3.1 показан формат пяти snmp сообщений, инкапсулированных в UDP дейтаграмму.

Менеджер отправляет эти три запроса на udp порт 161. Агент отправляет ловушки (trap) на udp порт 162. Так как используются два разных порта, одна система может выступать в роли менеджера и агента одновременно.

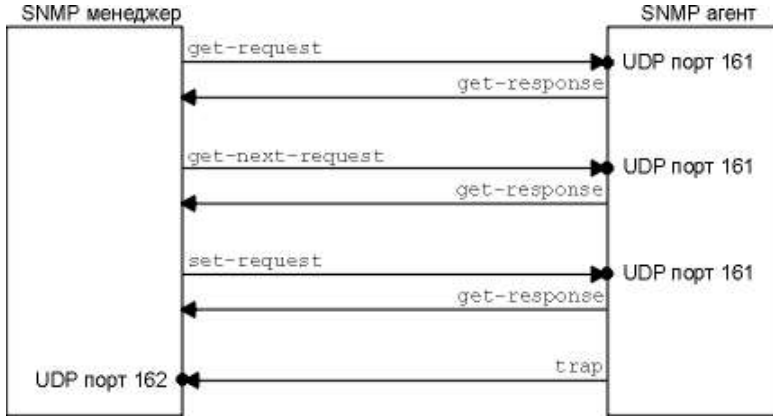


Рисунок 3.1 - Пять операторов SNMP.

3.3 Протокол FTP

FTP (File Transfer Protocol, или “Протокол передачи данных”) - один из старейших протоколов в Internet и входит в его стандарты. Первые спецификации FTP относятся к 1971 году. С тех пор FTP претерпел множество модификаций и значительно расширил свои возможности. FTP может использоваться как в программах пользователей, так и в виде специальной утилиты операционной системы.

FTP предназначен для решения задач разделения доступа к файлам на удаленных хостах, прямого или косвенного использования ресурсов удаленных компьютеров, обеспечения независимости клиента от файловых систем удаленных хостов, эффективной и надежной передачи данных.

Обмен данными в FTP происходит по TCP-каналу. Обмен построен на технологии “клиент-сервер”. FTP не может использоваться для передачи конфиденциальных данных, поскольку не обеспечивает защиты передаваемой информации и передает между сервером и клиентом открытый текст. FTP-сервер может потребовать от FTP-клиента аутентификации (т.е. при присоединении к серверу FTP-пользователь должен будет ввести свой идентификатор и пароль). Однако пароль, и идентификатор пользователя будут переданы от клиента на сервер открытым текстом.

Модели работы FTP

Простейшая модель работы протокола FTP представлена на схеме (рис. 3.2). В FTP соединение инициируется интерпретатором протокола пользователя. Управление обменом осуществляется по каналу управления в стандарте протокола TELNET. Команды FTP генерируются интерпретатором протокола пользователя и передаются на сервер. Ответы сервера отправляются пользователю также по каналу управления. В общем случае пользователь имеет возможность установить контакт с интерпретатором протокола сервера и отличными от интерпретатора протокола пользователя средствами.

Рисунок 3.2 – Модель работы FTP

Команды FTP определяют параметры канала передачи данных и самого процесса передачи. Они также определяют и характер работы с удаленной и локальной файловыми системами. Сессия управления инициализирует канал передачи данных. При организации канала передачи данных последовательность действий другая, отличная от организации канала управления. В этом случае сервер инициирует обмен данными в соответствии с согласованными в сессии управления параметрами.

Канал данных устанавливается для того же хоста, что и канал управления, через который ведется настройка канала данных. Канал данных может быть использован как для приема, так и для передачи данных.

Алгоритм работы протокола FTP состоит в следующем[5]:

1. Сервер FTP использует в качестве управляющего соединение на TCP порт 21, который всегда находится в состоянии ожидания соединения со стороны пользователя FTP.

2. После того как устанавливается управляющее соединение модуля “Интерпретатор протокола пользователя” с модулем сервера — “Интерпретатор протокола сервера”, пользователь (клиент) может отправлять на сервер команды. FTP-команды определяют параметры

соединения передачи данных: роль участников соединения (активный или пассивный), порт соединения (как для модуля “Программа передачи данных пользователя”, так и для модуля “Программа передачи данных сервера”), тип передачи, тип передаваемых данных, структуру данных и управляющие директивы, обозначающие действия, которые пользователь хочет совершить (например, сохранить, считать, добавить или удалить данные или файл и другие).

3. После того как согласованы все параметры канала передачи данных, один из участников соединения, который является пассивным (например, “Программа передачи данных пользователя”), становится в режим ожидания открытия соединения на заданный для передачи данных порт. После этого активный модуль (например, “Программа передачи данных сервера”) открывает соединение и начинает передачу данных.

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.

3.4 Протокол Telnet

Протокол **Telnet** позволяет обслуживающей машине рассматривать все удаленные терминалы как стандартные "сетевые виртуальные терминалы" строчного типа, работающие в коде ASCII, а также обеспечивает возможность согласования более сложных функций (например, локальный или удаленный эхо-контроль, страничный режим, высота и ширина экрана и т.д.) TELNET работает на базе протокола TCP. На прикладном уровне над TELNET находится либо программа поддержки реального терминала (на стороне пользователя), либо прикладной процесс в обсуживающей машине, к которому осуществляется доступ с терминала.

Работа с TELNET походит на набор телефонного номера. Пользователь набирает на клавиатуре что-то вроде:

telnet delta

и получает на экране приглашение на вход в машину delta.

Протокол TELNET существует уже давно. Он хорошо опробован и широко распространен. Создано множество реализаций для самых разных операционных систем. Вполне допустимо, чтобы процесс-

клиент работал, скажем, под управлением ОС VAX/VMS, а процесс-сервер под ОС UNIX System V.

Принципы построения

Строится на базе TCP протокола и работает по дуплексному, многопользовательскому протоколу. Это значит, что один сервер может обслуживать одновременно несколько клиентов.

Telnet построен на трех основных принципах:

1. NVT - Netvork Virtual Terminal - Принцип виртуальных терминалов. После установления соединения предполагается, что каждый участник работает как «Виртуальный сетевой терминал» - мнимое устройство, выполняющее стандартные сетевые промежуточные функции обычного терминала.
2. Принцип настраиваемых параметров. Если хост предоставляет дополнительный сервис помимо NVT, и клиент в состоянии его использовать, telnet предоставляет возможностьь сделать это.
3. Принцип симметрии терминалов и процессов. Участники соединения равноправны.

TELNET является универсальным клиентом и позволяет соединиться с большим колическтвом портов и общаться с различными приложениями.

3.5 TFTP

TFTP (Trivial File Transfer Protocol — простой протокол передачи файлов) используется главным образом для первоначальной загрузки бездисковых рабочих станций. TFTP, в отличие от FTP, не содержит возможностей аутентификации (хотя возможна фильтрация по IP-адресу) и основан на транспортном протоколе UDP.

В протоколе TFTP содержатся 5 сообщений [6]:

RRQ, которое выводится как запрос на чтение;

WRQ, которое выводится как запрос на запись;

ACK, которое выводится как сообщение подтверждения;

ERROR, которое выводится как сообщение об ошибке;

DATA, которое выводится как сообщение на чтение или запись следующей части данных.

3.6 Протокол SMTP

Основная задача протокола **SMTP** (Simple Mail Transfer Protocol – простой протокол передачи почты) заключается в том, чтобы обеспечивать передачу электронных сообщений (почту). Для работы через протокол SMTP клиент создаёт TCP соединение с сервером через порт 25. Затем клиент и SMTP сервер обмениваются информацией,

пока соединение не будет закрыто или прервано. Основной процедурой в SMTP является передача почты (Mail Procedure). Далее идут процедуры форвардинга почты (Mail Forwarding), проверка имён почтового ящика и вывод списков почтовых групп. Самой первой процедурой является открытие канала передачи, а последней - его закрытие.

Модель протокола. Взаимодействие в рамках SMTP строится по принципу двусторонней связи [7], которая устанавливается между отправителем и получателем почтового сообщения. При этом отправитель инициирует соединение и посылает запросы на обслуживание, а получатель - отвечает на эти запросы. Фактически отправитель выступает в роли клиента, а получатель - сервера.

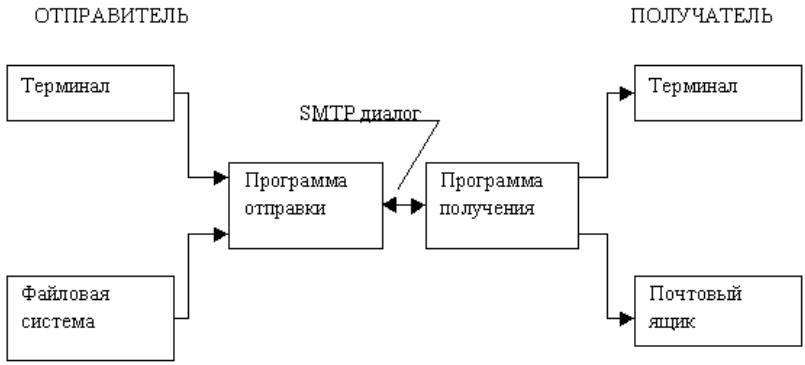


Рисунок 3.3 – Схема взаимодействия по протоколу SMTP

Канал связи устанавливается непосредственно между отправителем и получателем сообщения. При таком взаимодействии почта достигает абонента в течение нескольких секунд после отправки.

Команды SMTP. Простой протокол передачи почты обеспечивает двухсторонний обмен сообщениями между локальным клиентом и удаленным сервером MTA. MTA-клиент шлет команды MTA-серверу, а он, в свою очередь, отвечает клиенту. Другими словами, протокол SMTP требует получать ответы от приемника команд SMTP. Обмен командами и ответами на них называется почтовой транзакцией (mail transaction). Данные, передаются в формате NVT ASCII. Кроме того, команды тоже передаются в формате NVT ASCII. Команды передаются в форме ключевых слов, а не специальных символов, и указывают на необходимость совершить ту или иную операцию.

3.7 Протокол UDP

Протокол UDP (User Datagram Protocol - протокол пользовательских дейтаграмм, RFC 768) является одним из двух основных протоколов транспортного уровня, расположенных непосредственно над IP. Он предоставляет прикладным процессам транспортные услуги, которые не многим отличаются от услуг, предоставляемых протоколом IP. Протокол UDP обеспечивает ненадежную доставку дейтаграмм и не поддерживает соединений из конца в конец. Другими словами, его пакеты могут быть потеряны, продублированы или прийти не в том порядке, в котором они были отправлены. Протокол UDP предполагает, что нижестоящим протоколом является Internet (IP). Данный протокол предоставляет прикладной программе процедуру для посылки сообщений другим программам, причем механизм протокола минимален. Протокол UDP ориентирован на транзакции, получение дейтаграмм и защита от дублирования не гарантированы. Примерами сетевых приложений, использующих UDP, являются NFS (Network File System) и SNMP [8].

Протокол UDP не использует никаких механизмов подтверждения переданных данных, также протокол UDP не нумерует переданных пакетов и никак не управляет скоростью передачи данных. В результате UDP дейтаграммы могут прийти не по порядку или могу быть потеряны, также не исключено, что UDP дейтаграммы могут прийти раньше, чем получатель сможет их обработать. Основная задача протокола UDP - уменьшить время, затрачиваемое на перенос данных между двумя взаимодействующими приложениями различных открытых систем.

Заголовок UDP содержит только три поля «*Port*» (Порт), «*Checksum*» (Контрольная сумма) и «*Length*» (Длина дейтаграммы). Первое предназначено для мультиплексирования дейтаграмм по различным приложениям, второе - обеспечивает целостность каждой конкретной переданной дейтаграммы.

Формат UDP-дейтаграмм

Формат UDP дейтаграммы показан в таблице 3.1.

Таблица 3.1

Source port	**Destination Port**
Length	**Checksum**
Данные	

Поля UDP дейтаграммы:

Поле *Source port* (Порт отправителя) указывает порт приложения, отправившего дейтаграмму. Это поле необязательное, если оно используется, то содержит нуль.

Поле *Destination port* (Порт получателя) указывает порт приложении, которое должно обработать эту дейтаграмму у получателя;

Поле *Length* (Длина) определяет число октетов (байт) в UDP дейтаграмме, включая заголовок;

Поле *Checksum* (Контрольная сумма) указывает контрольную сумму UDP дейтаграммы. Поле необязательно содержит контрольную сумму. Для упрощения вычислений в поле Checksum записывается нуль, тогда получатель не проверяет контрольную сумму дейтаграммы. Однако, это, может, привести к неприятным последствиям, поскольку протокол IP не вычисляет контрольную сумму переносимых им данных (в частности, UDP дейтаграммы). Если расчетная контрольная сумма равна нулю, она передается как поле, целиком состоящее из единиц (эквивалент при дополнении до единицы). Передача поля, целиком состоящего из нулей, означает, что отправитель дейтаграммы не вычислял контрольной суммы (при отладке, а также для тех протоколов, которые не требуют точности передачи).

Инкапсуляция UDP, мультиплексирование и порты

Протокол UDP, согласно стеку протоколов TCP/IP, должен взаимодействовать с протоколами прикладного уровня и протоколами межсетевого уровня. Протоколы прикладного уровня передают битовый (байтовый) поток протоколу UDP. Этот поток протокол UDP фрагментирует и вставляет в область данных своих дейтаграмм. Далее протокол UDP вычисляет контрольную сумму, добавляет свой заголовок и передаёт свои дейтаграммы нижележащему протоколу межсетевого уровня - протоколу IP. Протокол IP вставляет в область своих данных UDP дейтаграмму целиком, не изменяя, и добавляет свой IP заголовок, таким образом, получая целостный IP пакет. После чего, IP пакет передаётся протоколу нижележащего уровня сетевого интерфейса. В случае, использования на этом уровне протокола Ethernet, IP пакет вставляется в область данных Ethernet кадра, после чего протокол Ethernet вычисляет свой заголовок и окончание и дописывает их к области данных. В результате получается полный Ethernet кадр с вложенными IP пакетом и UDP дейтаграммой, содержащей битовый поток от конкретного приложения. Процесс вложения называется инкапсуляцией пользовательских данных и показан в таблице 3.2 и рис. 3.4.

Таблица 3.2 - Инкапсуляция UDP дейтаграмм

		Заголовок UDP	Область данных UDP
	Заголовок IP	Область данных IP	
Заголовок Ethernet	Область данных кадра		Окончание

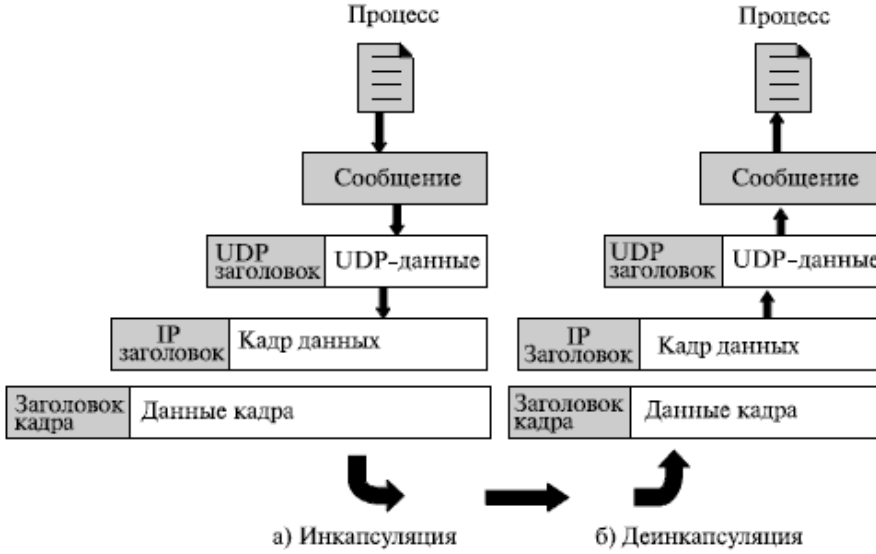


Рисунок 3.4 – Инкапсуляция и деинкапсуляция

Обратный процесс называется декапсуляцией и происходит он, по мере надобности, в промежуточных узлах и обязательно на узле назначения. Его необходимость обусловлена различными зонами ответственности конкретных протоколов, а, следовательно, и заголовков этих протоколов.

Зона ответственности протокола Ethernet - локальная сеть. Если узел находится за пределами локальной сети, кадр должен быть декапсюлирован до IP пакета, чтобы осуществить доставку по IP адресу, указанном в IP заголовке. Двух заголовков Ethernet и IP хватит, чтобы доставить UDP пакет до узла назначения, но не хватит, чтобы передать конкретному приложению переданный битовый поток.

Поэтому необходимо на узле получателя декапсулировать пакет, сначала до UDP дейтаграммы, заголовок которой поможет мультиплексировать битовый поток между приложениями, и затем отбросить и UDP заголовок, передав данные в чистом виде приложению.

Мультиплексирование данных протоколом UDP между приложения осуществляется с помощью механизма портов. Порт - это своеобразный адрес конкретного приложения на определённом хосте (Порт + IP-адрес = сокет. Сокет - уникальный идентификатор приложения конкретного узла в сети). Такой же механизм адресации используется и для протокола TCP, причём порты TCP и UDP могут иметь одно и то же значение. Следовательно, можно обобщить понятие порт до адреса транспортного уровня.

Порт в TCP или UDP – это логический канал с определенным номером (от 0 до 65536), обеспечивающий текущее взаимодействие между отправителем и получателем. Порты позволяют компьютеру с одним IP-адресом параллельно обмениваться данными с множеством других компьютеров. Некоторые номера портов (от 0 до 1024) привязаны к определенным службам и приложениям.

Пакеты, поступающие на транспортный уровень, организуются операционной системой в виде множества очередей к точкам входа различных прикладных процессов. В терминологии TCP/IP такие системные очереди называются портами. Прикладной процесс, предоставляющий некоторые услуги другим прикладным процессам (сервер), ожидает поступления сообщений в порт, специально выделенный для этих услуг. Сообщения отправляются процессами-клиентами и должны содержать запросы на предоставление услуг. Порты нумеруются, начиная с нуля. Например, сервер SNMP всегда ожидает поступлений сообщений в порт 161. В каждом узле может быть только один сервер SNMP, так как существует только один UDP-порт 161. Данный номер порта является общеизвестным, то есть фиксированным номером, официально выделенным для услуг SNMP. Таким образом, адресом назначения, который используется на транспортном уровне, является идентификатор (номер) порта прикладного сервиса. Номер порта, задаваемый транспортным уровнем, в совокупности с номером сети и номером компьютера, задаваемыми сетевым уровнем, однозначно определяют прикладной процесс в сети.

Назначение номеров портов прикладным процессам осуществляется либо централизовано, если эти процессы представляют собой популярные общедоступные сервисы, либо локально для тех

сервисов, которые еще не стали столь распространенными, чтобы за ними закреплять стандартные(зарезервированные) номера.

Централизованное присвоение сервисам номеров портов выполняется организацией Internet Assigned Numbers Authority (IANA) [2] (табл. 3.3).

В случае если приложение не имеет назначенного IANA порта, оно назначает себе порт самостоятельно из диапазона 49152- 65535, начиная с большего значения. Далее, чтобы узнать какой порт приложение назначило себе самостоятельно, любой узел может отправить запрос узлу.

Таблица 3.3. Используемые протоколами UDP и TCP порты

Десятичный номер порта	Обозначение порта	Описание
0	-	Зарезервировано
20	FTP-data	FTP (данные)
21	FTP	Протокол пересылки файлов (управление)
23	telnet	Подключение терминала
25	SMTP	Протокол передачи почтовых сообщений
49	login	Протокол входа в ЭВМ
53	Domain	Сервер имен доменов (dns)
69	TFTP	Упрощенная пересылка файлов
80	WWW-HTTP	World Wide Web HTTP
101	Hostname	Сервер имен ЭВМ для сетевого информационного центра
110	POP3	Почтовый протокол POP3
123	NTP	Сетевой протокол синхронизации
179	BGP	Динамический протокол внешней маршрутизации
5060	SIP	Протокол инициации сеансов

Использование

Недостаточная надёжность протокола может выражаться как в потере отдельных пакетов, так и в их дублировании. UDP используется при передаче потокового видео, игр реального времени, а также некоторых других типов данных.

Ненадёжность протокола UDP надо понимать в том смысле, что в случаях влияния внешних факторов, приводящих к сбоям, протокол UDP не предусматривает стандартного механизма повторения передачи потерянных пакетов. В этом смысле он настолько же надежен, как и протокол ICMP.

Если приложению требуется большая надёжность, то используется протокол TCP или SCTP, либо реализуется какой-нибудь свой нестандартный алгоритм повторения передач в зависимости от условий.

3.8 Протокол TCP

Основная задача протокола TCP (Transmission Control Protocol - протокол управления передачей, RFC 793) обеспечить сквозную гарантированную доставку данных между прикладными процессами. Другими словами, если приложение на одном узле хочет передать данные без каких-либо изменений такому же приложению на другом узле, то задачу эту выполнить приложение доверяет протоколу TCP.

В стеке протоколов TCP/IP протокол TCP работает, как и протокол UDP, на транспортном уровне. Протокол TCP предоставляет транспортные услуги, отличающиеся от услуг UDP. Вместо ненадежной доставки дейтаграмм без установления соединений, он обеспечивает гарантированную доставку с установлением соединений между прикладными процессами в виде байтовых потоков.

Протокол TCP используется в тех случаях, когда требуется надежная доставка сообщений. Он освобождает прикладные процессы от необходимости использовать таймауты и повторные передачи для обеспечения надежности. Наиболее типичными прикладными процессами, использующими TCP, являются FTP и TELNET. Кроме того, TCP используют система X-Window, rcp (remotecopy - удаленное копирование) и другие "r-команды". Большие возможности TCP даются не бесплатно. Реализация TCP требует большой производительности процессора и большой пропускной способности сети. Внутренняя структура модуля TCP гораздо сложнее структуры модуля UDP.

Единицей данных протокола TCP является сегмент. Информация, поступающая к протоколу TCP в рамках логического соединения от протоколов более высокого уровня, рассматривается протоколом TCP как неструктурированный поток байт. Поступающие данные буферизуются средствами TCP. Для передачи на сетевой уровень из буфера "вырезается" некоторая непрерывная часть данных, которая и называется сегментом. Сегменты состоят из заголовка и блока данных.

Поскольку приложение полностью полагается на протокол TCP (зачастую, не имея собственных механизмов проверки данных), протокол TCP должен реализовывать доставку собственными механизмами [2]. Инкапсуляция данных при использовании протокола TCP происходит схожим с UDP образом, только TCP заголовок значительно отличается от заголовка UDP. Заголовок TCP несет себе множество полей служебной информации, служащей ДЛЯ управления передачей данных по каналам связи.

Заголовок TCP-сегмента

Заголовок TCP сегмента состоит из 32-разрядных слов (32 бита) и имеет переменную длину, зависящую от размера поля «*Options*» и «*Padding*», но всегда кратную 32 битам. Формат заголовка TCP сегмента представлен в таблице 3.4.

Таблица 3.4. Формат заголовка TCP сегмента

<table>
<tr><td colspan="8">Source Port</td><td>Destination Port</td></tr>
<tr><td colspan="9">Sequence Number (SN)</td></tr>
<tr><td colspan="9">Acknowledgment Number (ACK)</td></tr>
<tr><td>Data Offset (0-3)</td><td>Reserved (4-9)</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>Window</td></tr>
<tr><td colspan="8">Checksum</td><td>Urgent Pointer</td></tr>
<tr><td colspan="8">Options</td><td>Padding</td></tr>
<tr><td colspan="9">Data</td></tr>
</table>

Поле «*Source Port*» и поле «*Destination Port*» (порт отправителя и получателя) указывает номера портов источника и получателя соответственно.

Поле «*Sequence Number*» (номер последовательности) определяет порядковый номер первого байта в поле данных сегмента среди всех октетов потока данных для текущего логического соединения (TCP сессия). Если в TCP сегменте передаются октеты с 5-го по 87-ой, то SN = 5. В случае начальной установки соединения (TCP сегмент с флагом «SYN») в поле SN записывается случайный номер ISN (Initial Sequence Number = начальный номер последовательности), в первом TCP сегменте без флага «SYN» поле SN = ISN + 1.

Поле «*Acknowledgment Number*» (номер подтверждения) указывает (в совокупности с флагом «ACK») порядковый номер октета, который отправитель данного сегмента желает получить подразумевая тем

самым, что все предыдущие байты (с номерами от ISN+1 до ACK-1) были успешно получены.

Поле «*Data Offset*» (смещение данных, 4 бита) указывает смещение начала данных TCP в сегменте относительно начала сегмента. Поле указывает длину TCP заголовка. Минимальная длина заголовка без параметров может быть равна 5-ти 32-х битным словам или 160 байтам. Максимальная длина заголовка - 480 байт.

Поле «*Reserved*» (зарезервировано, 6 бит) зарезервировано для дальнейшего использования, заполняется нулями.

Поле «*Flags*» (флаги, 6 бит) содержит управляющие биты, для контроля над соединением.

- Флаг *URG* (Urgent - срочно) указывает срочность доставки сегмента;
- Флаг *ACK* (Acknowledgment - подтверждение) указывает, что поле «Acknowledgment Number» указывает следующий октет, который предполагает принять получатель;
- Флаг *PSH* (Push - протолкнуть) указывает на незамедлительность отправки данных из TCP сегмента процессу прикладного уровня, даже, если приёмный TCP буфер не заполнен. Используется при передаче данных интерактивных приложений;
- Флаг *RST* (Reset - перезагрузка) указывает на необходимость перезагрузки текущего TCP соединения;
- Флаг *SYN* (Synchronization - синхронизация) указывает на то, что данный TCP сегмент является запросом на установление соединения;
- Флаг *FIN* (Finish - конец) указывает на закрытие TCP соединения;

Поле «*Window*» (окно) указывает размер TCP «окна» в октетах. Размер «окна» указывает отправителю максимально возможное число TCP сегментов, которые могут быть отправлены без подтверждения.

Поле «*Checksum*» (контрольная сумма) содержит контрольную сумму всего TCP сегмента. Обеспечивает поразрядную проверку целостности.

Поле «*Urgent Pointer*» (указатель срочности) используется для указания длины срочных данных, которые размещаются в начале поля данных сегмента. Число в этом поле указывает смещение октета срочных данных относительно первого октета данных в сегменте. Например, если в TCP сегменте передаются окном с 5-го по 87-ой и первые 15 из них срочные, то поле «Urgent Pointer» = 15. Протокол TCP не регламентирует, как должны обрабатываться срочные данные, определено только, что они должны быть максимально быстро

отправлены прикладному процессу. Поле «Urgent Pointer» действует в совокупности флагом «URG».

Поле «*Options*» (опции, поле переменной длины) поле указывает на реализацию работы дополнительных услуг (опций) протокола TCP. Поле конкретной опции в поле «Options» состоит из октета «Option Kind» (тип опции), октета «Option Length» (длина опции) и октетов «Option Octets» (октетов с данными опции). Стандарт протокола TCP определяет три опции:

«*End of Option List*» (конец списка опций) имеет тип опции, равный «О», и длину в один октет. Определяет конец списка опций. Не используется, если набор опций TCP выходит за 32-битную границу.

«*No Operation*» (бездействие) имеет тип опции, равный «1», и длину в один октет. Используется между TCP опциями для 32- битного выравнивания.

«*Maximum Segment Size*» имеет тип опции, равный «2», и длину в четыре октета. Описывает максимальный размер сегмента, который может быть передан по TCP соединению. Вычисляется на основании MTU (Maximum Transfer Unit - максимальный блок передачи), который может быть передан по соединению за вычетом IP и TCP заголовков. Данная опция включается только на стадии установки соединения, вместе с флагом «SYN».

Поле «Padding» (заполнение) заполняется нулями до выравнивания по 32-битной границе в случае, если поле «Options» не укладывается в 32-битное слово.

Установка TCP соединения

Надёжность передачи данных по протоколу TCP обеспечивается за счёт установления логического соединения. TCP соединения является двунаправленным полнодуплексным каналом между двумя прикладными процессами в общей IP сети. TCP соединение устанавливается в три фазы (3-way handshake) (рисунок 3.5).

Установление TCP соединения начинается с посылки TCP узлом 1 сегмента с установленным флагом «SYN» [9]. В качестве номера последовательности указывается случайное число «ISN», в качестве номера подтверждения указывается нулевой октет, размер «окна» выставляется по умолчанию для определённой операционной системы (для Windows 16384 байт), параметр «MSS» и параметр запроса для опции «SACK» (Selective ACK - выборочные подтверждения). Опция «SACK» позволяет установить соединение с выборочным подтверждением принятых TCP сегментов, экономя тем самым полосу пропускания и аппаратные ресурсы компьютера.

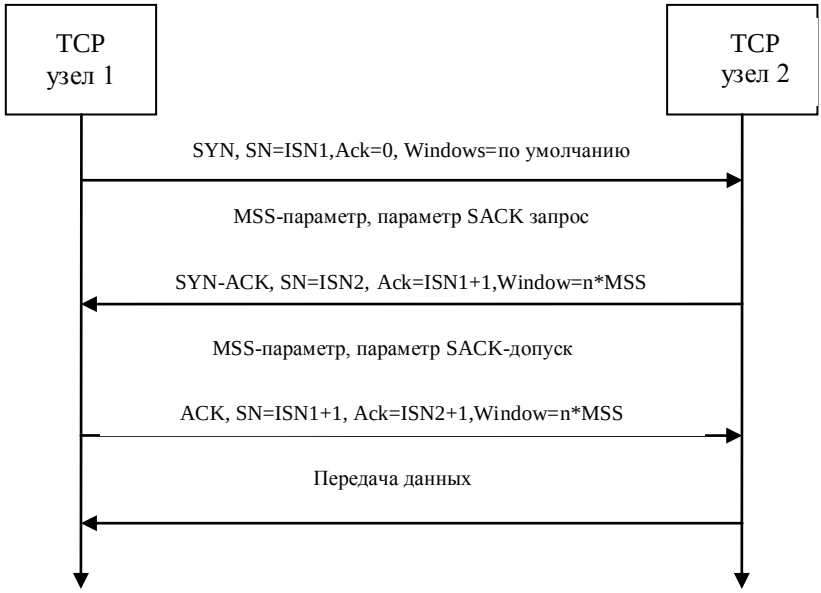


Рисунок 3.5 - Установление TCP соединения

Если TCP узел 2 готов установить TCP соединение с TCP узлом 1, то есть данный TCP порт не закрыт и ресурсов компьютера хватает для установки соединения, то узел 2 посылает ответ с установленными флагами «SYN» и «ACK». Такой ответ также содержит параметры «SN» равный ISN2, «Аск» равный ISN1+1, «Window» равный n*MSS, «MSS» и параметр «SACK-допуск», как указание на готовность использовать опцию «SACK». Перед тем, как отравить этот сегмент, TCP узел 2 должен подсчитать допустимый размер «окна». Для этого вычисляется сначала общий «MSS» минимальный «MSS» для двух узлов, затем делится размер «окна» умолчанию (16384 байта) на величину общего «MSS» и полученный результат округляется в большую сторону до целого числа «n». Допустимый размер «окна» вычисляет как произведение «n» на общий «MSS».

На завершающей стадии установления соединения TCP узел отправляет подтверждение принятых параметров от TCP узла 2.

Обеспечение достоверности и управление скоростью передачи данных

Протокол TCP обеспечивает гарантированную доставку данных приложений, что подразумевает защиту от повреждения, потери, дублирования и нарушения очередности получения данных. Все октеты, передаваемые по TCP соединению, пронумерованы (начиная со случайного числа), все TCP сегменты содержат контрольную сумму, дабы избежать повреждения целостности сегмента. Протокол TCP имеет приёмный и передающий буферы, размеры которых определяются программно на уровне свойств конкретной операционной системы. Исходя из размеров приёмного буфера и его наполнения, протокол TCP принимающего узла определяет текущий размер окна, который и передаётся в сегменте подтверждения «АСК» передающему узлу, чтобы проинформировать передающий узел о максимально возможном количестве одновременно отправляемых сегментов. Таким образом, любой принимающий узел может ограничивать количество одновременно направляемых ему данных, регулируя тем самым скорость передачи.

Пакеты подтверждения «АСК» должны отправляться в ответ на каждый принятый пакет с данными, но существует технология позволяющая подтверждать одним пакетом «АСК» сразу несколько сегментов данных - кумулятивное подтверждение. Такая технология может быть полезна в случае прихода TCP сегментов не в порядке отправления и с запаздыванием. Например, были отправлены сегменты с порядковыми номерами 4, 5, 6 и 7, и в некоторый момент времени получатель принял сегменты 4 и 7. По правилам работы протокола TCP, получатель должен подтвердить сегмент 4 и ожидать (сохранив в буфере сегмент 7) прихода сегментов 5 и 6. Но кумулятивное подтверждение позволяет выждать некоторое время, дождавшись сегментов 5 и 6, и отправить общее подтверждение сегментов 4, 5, 6, и 7 в одном сегменте «АСК».

Гарантировать доставку в негарантированной среде передачи IP означает смириться с возможной потерей некоторых данных и обеспечить их повторную передачу. Существует два разных способа обнаружить потерю TCP сегмента вызванную некорректной работой сети. Во-первых, существует так называемый «тайм-аут повторной передачи» (retransmission timeout), который выжидается после отправки очередного сегмента до решения о потери этого сегмента. После принятия решения о потери сегмента, сегмента высылается повторно. Величина этого таймаута рассчитывается, исходя из времени оборота (RTT - Round-Trip Time). Второй кри-терий обнаружения

потери сегмента - поступление трёх повтор¬ных сегментов «АСК» (DUP#ACK).

Повторные «АСК» - это сегменты подтверждения ранее подтверждённых данных, отправленные на сегменты пришедшие позднее. Возвращаясь к рассмотренному ранее примеру, если получатель не дождётся всех переданных сегментов, чтобы образовать кумулятивное подтверждение он пошлёт подтверждение получения 4-го сегмента и повторное подтверждение 4-го сегмента в ответ на получение 7-го сегмента. Посылка повторного подтверждения также производится в ответ на прием 8-го и 9-го сегментов. Только после получения потерянных 5-го и 6-го сегментов производится подтверждение всех 9-ти принятых сегментов. Таким образом, если отправитель получает три повторных подтверждения (triple duplicate acknowledgment), он принимает решение об изменении размера окна.

Действительно, потеря пакетов означает, что IP сеть не смогла по каким-то причинам доставить данные, возможно (и очень вероятно), ввиду большой загруженности. Чтобы не загружать сеть ещё больше и, соответственно, не терять ещё больше сегментов, TCP протокол уменьшает скорость отправки данных, снижая текущий 1размер «окна», тем самым, уменьшая количество одновременно отравляемых сегментов.

На данный момент существует больше десятка различных версий протокола TCP (Vegas, Reno, Newreno и т.д.) и основная разница этих версий заключается в механизмах уменьшения «окна».

Уменьшение «окна» приведёт к одностороннему снижению скорости передачи данных и, возможной, разгрузке сети. Чтобы увеличить скорость передачи данных, необходимо увеличить размер текущего окна. Существуют два последовательно работающих алгоритма для увеличения размера «окна». Первый называется медленный старт» (slow-start) и осуществляется с первого момента увеличения окна и до половинного размера «окна», при котором произошла потеря. На стадии «медленного старта» «окно» увеличивается на один MSS с каждым поступившим сегментом подтверждения, таким образом, осуществляется мультипликативный рост «окна». Как только размер текущего «окна» доходит до половины размера «окна», зафиксированного перед потерей, вступает силу алгоритм «предотвращения перегрузки» (congestion avoidance), в котором «окно» возрастает аддитивно: в ответ на каждое подтверждение «окно» увеличивается на величину 1/MSS. Следовательно, при поступлении подтверждения «полного окна» «окно» увеличиться на MSS. Эта часть алгоритма помогает избежать

перегрузки сети, подыскивая оптимальную скорость передачи сегментов. Оба этих алгоритма действуют и на начальной стадии соединения, сразу после его установления, чтобы не перегрузить сеть резким выбросом пакетов сразу для большого размер согласованного «окна». Только граница перехода из одного алгоритма в другой не находится эмпирическим путём, а указывается конкретной величиной, специфической для определенной операционной системы.

Завершение TCP соединения

Для завершения TCP соединения служат сегменты с установленным флагом «FIN», которые должны быть подтверждены сегментом с флагом «ACK». Соединение должно быть закрыто с двух сторон, иначе соединение останется в полуоткрытом состоянии. Полуоткрытые порты закрываются самостоятельно операционной системой, если это предусмотрено. Если этого не предусмотрено то в системе могут просто кончиться доступные порты для TCP соединений, что вызовет отказ службы TCP.

3.9 Протокол IP

Протокол IP является самым главным во всей иерархии протоколов семейства TCP/IP. Именно он используется для управления рассылкой TCP/IP пакетов по сети Internet. Среди различных функций, возложенных на IP обычно выделяют следующие:

-определение пакета, который является базовым понятием и единицей передачи данных в сети Internet. Многие зарубежные авторы называют такой IP-пакет дейтаграммой;

-определение адресной схемы, которая используется в сети Internet;

-передача данных между канальным уровнем (уровнем доступа к сети) и транспортным уровнем (другими словами мультиплексирование транспортных дейтаграмм во фреймы канального уровня);

-маршрутизация пакетов по сети, т.е. передача пакетов от одного шлюза к другому с целью передачи пакета машине-получателю;

-"нарезка" и сборка из фрагментов пакетов транспортного уровня.

Главными особенностями протокола IP является отсутствие ориентации на физическое или виртуальное соединение. Это значит, что прежде чем послать пакет в сеть, модуль операционной системы, реализующий IP, не проверяет возможность установки соединения, т.е. никакой управляющей информации кроме той, что содержится в самом IP-пакете, по сети не передается. Кроме этого, IP не заботится о проверке целостности информации в поле данных пакета, что заставляет отнести его к протоколам ненадежной доставки.

Целостность данных проверяется протоколами транспортного уровня (TCP) или протоколами приложений.

Таким образом, вся информация о пути, по которому должен пройти пакет берется из самой сети в момент прохождения пакета. Именно эта процедура и называется маршрутизацией в отличии от коммутации, которая используется для предварительного установления маршрута следования данных, по которому потом эти данные отправляют.

При неустойчивой работе сети пакеты могут пересылаться по различным маршрутам и затем собираться в единое сообщение. При коммутации путь придется каждый раз вычислять заново для каждого пакета, а в этом случае коммутация потребует больше накладных затрат, чем маршрутизация.

Вообще говоря, версий протокола IP существует несколько. В настоящее время используется версия IPv4 (RFC791). Формат пакета протокола представлен на рисунке 3.6 [7].

Рисунок 3.6 - Формат пакета Ipv4

Фактически, в этом заголовке определены все основные данные, необходимые для перечисленных выше функций протокола IP: *адрес отправителя* (4-ое слово заголовка), *адрес получателя* (5-ое слово заголовка), *общая длина пакета* (поле Total Lenght) *и тип пересылаемой дейтаграммы* (поле Protocol).

Используя данные заголовка, машина может определить на какой сетевой интерфейс отправлять пакет. Если IP-адрес получателя принадлежит одной из ее сетей, то на интерфейс этой сети пакет и будет отправлен, в противном случае пакет отправят на другой шлюз.

Если пакет слишком долго "бродит" по сети, то очередной шлюз может отправить ICMP-пакет на машину-отправитель для того, чтобы

уведомить о том, что надо использовать другой шлюз. При этом, сам IP-пакет будет уничтожен. На этом принципе работает программа ping, которая используется для деления маршрутов прохождения пакетов по сети.

Зная протокол транспортного уровня, IP-модуль производит раскапсулирование информации из своего пакета и ее направление на модуль обслуживания соответствующего транспорта.

Размер максимально возможного фрейма, который передается по сети, определяется величиной MTU (Maximum Transsion Unit), определенной для протокола канального уровня. Для того, чтобы потом восстановить пакет IP должен держать информацию о своем разбиении. Для этой цели используется поля "flags" и "fragmentation offset". В этих полях определяется, какая часть пакета получена в данном фрейме, если этот пакет был фрагментирован на более мелкие части.

Обсуждая протокол IP и вообще все семейство протоколов TCP/IP нельзя не упомянуть, что в настоящее время перед Internet возникло множество по-настоящему сложных проблем, которые требуют изменения базового протокола сети.

IPing - новое поколение протоколов IP

В начале 1995 года IETF, после 3-х лет консультаций и дискуссий, выпустило предложения по новому стандарту протокола IP - IPv6, который еще называют IPing. К слову, следует заметить, что сейчас Internet-сообщество живет в основном по стандарту IPv4. IPv6 призван не только решить адресную проблему, но и попутно помочь решению других задач, стоящих в настоящее время перед Internet.

Однако, не только адресная проблема определила появление нового протокола. Разработчики позаботились и о масштабируемой адресации IP-пакетов, ввели новые типы адресов, упростили заголовок пакета, ввели идентификацию типа информационных потоков для увеличения эффективности обмена данными, ввели поля идентификации и конфиденциальности информации.

Новый заголовок IP-пакета показан на рисунке 3.7.

В этом заголовке поле *версия* - номер версии IP, равное 6.

Поле *приоритет* может принимать значения от 0 до 15. Первые 8 значений закреплены за пакетами, требующими контроля переполнения. Например, 0 - несимвольная информация; 1 - информация заполнения (news), 2 - не критичная ко времени передача данных (e-mail); 4 - передача данных режима on-line (FTP, HTTP, NFS и т.п.); 6 - интерактивный обмен данными (telnet, X); 7 - системные данные или данные управления сетью (SNMP, RIP и т.п.).

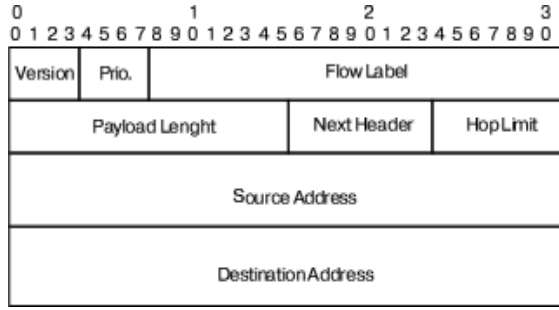


Рисунок 3.7 - Заголовок IPv6

Поле *метка потока* предполагается использовать для оптимизации маршрутизации пакетов. В IPv6 вводится понятие потока, который состоит из пакетов. Пакеты потока имеют одинаковый адрес отправителя и одинаковый адрес получателя и ряд других одинаковых опций. Подразумевается, что маршрутизаторы будут способны обрабатывать это поле и оптимизировать процедуру пересылки пакетов, принадлежащих одному потоку. В настоящее время алгоритмы и способы использования поля "метка потока" находятся на стадии обсуждения.

Поле *длины пакета* определяет длину следующей за заголовком части пакета в байтах. Поле *следующий заголовок* определяет тип следующего за заголовком IP-заголовка. Заголовок IPv6 имеет меньшее количество полей, чем заголовок IPv4. Многие необязательные поля могут быть указаны в дополнительных заголовках, если это необходимо.

Поле *ограничение переходов* определяет число промежуточных шлюзов, которые ретранслируют пакет в сети. При прохождении шлюза это число уменьшается на единицу. При достижении значения "0" пакет уничтожается. После первых 8 байтов в заголовке указываются адрес отправителя пакета и адрес получателя пакета. Каждый из этих адресов имеет длину 16 байт. Таким образом, длина заголовка IPv6 составляет 48 байтов.

После 4 байтов IP-адреса стандарта IPv4, шестнадцать байт IP-адреса для IPv6 выглядят достаточными для удовлетворения любых потребностей Internet. Не все 2128 адресов можно использовать в качестве адреса сетевого интерфейса в сети. Предполагается выделение отдельных групп адресов, согласно специальным префиксам внутри IP-адреса, подобно тому, как это делалось при определении типов сетей в IPv4. Так, двоичный префикс "0000 010" предполагается закрепить за

отображением IPX-адресов в IP-адреса. В новом стандарте выделяются несколько типов адресов: unicast addresses - адреса сетевых интерфейсов, anycast addresses - адреса не связанные с конкретным сетевым интерфейсом, но и не связанные с группой интерфейсов и multicast addresses - групповые адреса. Разница между последними двумя группами адресов в том, что anycast address это адрес конкретного получателя, но определяется адрес сетевого интерфейса только в локальной сети, где этот интерфейс подключен, а multicast-сообщение предназначено группе интерфейсов, которые имеют один multicast-адрес.

Маршрутизировать IPv6-пакеты предполагается также, как и IPv4-пакеты. Однако, в стандарт были добавлены три новых возможности маршрутизации: маршрутизация поставщика IP-услуг, маршрутизация мобильных узлов и автоматическая переадресация. Эти функции реализуются путем прямого указания промежуточных адресов шлюзов при маршрутизации пакета. Эти списки помещаются в дополнительных заголовках, которые можно вставлять вслед за заголовком IP-пакета.

Кроме перечисленных возможностей, новый протокол позволяет улучшить защиту IP-трафика. Для этой цели в протоколе предусмотрены специальные опции. Первая опция предназначена для защиты от подмены IP-адресов машин. При ее использовании нужно кроме адреса подменять и содержимое поля идентификации, что усложняет задачу злоумышленника, который маскируется под другую машину. Вторая опция связана с шифрацией трафика. 6 июня 2012 на международной конференции в Сан-Хосе вице-президент Google Винтон Серф объявил об официальном запуске применения IPv6 по всему миру.

3.10 Протокол ICMP

Протокол передачи команд и сообщений об ошибках (ICMP - internet control message protocol, RFC-792, - 1256) выполняет многие и не только диагностические функции, хотя у рядового пользователя именно этот протокол вызывает раздражение, сообщая об его ошибках или сбоях в сети. Именно этот протокол используется программным обеспечением ЭВМ при взаимодействии друг с другом в рамках идеологии TCP/IP. Осуществление повторной передачи пакета, если предшествующая попытка была неудачной, лежит на TCP или прикладной программе. При пересылке пакетов промежуточные узлы не информируются о возникших проблемах, поэтому ошибка в маршрутной таблице будет восприниматься как неисправность в узле

адресата и достоверно диагностироваться не будет. ICMP-протокол сообщает об ошибках в IP-дейтограммах, но не дает информации об ошибках в самих ICMP-сообщениях. icmp использует IP, а IP-протокол должен использовать ICMP. В случае ICMP-фрагментации сообщение об ошибке будет выдано только один раз на дейтограмму, даже если ошибки были в нескольких фрагментах.

Задачи, решаемые ICMP

Можно сказать, что ICMP-протокол осуществляет [10]:

- передачу отклика на пакет или эхо на отклик;
- контроль времени жизни дейтограмм в системе;
- реализует переадресацию пакета;
- выдает сообщения о недостижимости адресата или о некорректности параметров;
- формирует и пересылает временные метки;
- выдает запросы и отклики для адресных масок и другой информации.

ICMP-сообщения об ошибках никогда не выдаются в ответ на:

- ICMP-сообщение об ошибке.
- При мультикастинг или широковещательной адресации.
- Для фрагмента дейтограммы (кроме первого).
- Для дейтограмм, чей адрес отправителя является нулевым, широковещательным или мультикастинговым.

Эти правила призваны блокировать потоки дейтограмм, посылаемым в отклик на мультикастинг или широковещательные ICMP-сообщения. ICMP-сообщения имеют свой формат, а схема их вложения аналогична UDP или TCP и представлена на рис. 3.8.

Рис. 3.8 - Схема вложения ICMP-пакетов в Ethernet-кадр

Все ICMP пакеты начинаются с 8-битного поля типа ICMP и его кода (15 значений).

По существу, значения полей типа и кода выполняют почти ту же функцию, что и порт в TCP и UDP-протоколах.

Код уточняет функцию ICMP-сообщения.

Процедура "отключение источника" (quench, поле тип ICMP=4) позволяет управлять потоками данных в каналах Интернет. Не справляясь с обработкой дейтаграмм, адресат может послать запрос "отключить источник" отправителю, который может сократить темп посылки пакетов или вовсе прервать их посылку. Специальной команды, отменяющей прежние запреты, не существует. Если сообщения об отключении прекращаются, источник может возобновить посылку пакетов или увеличить частоту их отправки. Ниже (на рис. 3.9) представлен формат эхо-запроса (ping) и отклика для протокола ICMP.

Рис. 3.9 - Формат эхо-запроса и отклика ICMP

Поля *Идентификатор* (обычно это идентификатор процесса) и *Номер по порядку* (увеличивается на 1 при посылке каждого пакета) служат для того, чтобы отправитель мог связать в пары запросы и отклики. Поле *Тип* определяет, является ли этот пакет запросом (8) или откликом (0). Поле *Контрольная сумма* представляет собой 16-разрядное дополнение по модулю 1 контрольной суммы всего ICMP-сообщения, начиная с поля Тип. Поле *Данные* служит для записи информации, возвращаемой отправителю. При выполнении процедуры ping эхо-запрос с временной меткой в поле Данные посылается адресату. Если адресат активен, он принимает IP-пакет, меняет адрес отправителя и получателя местами и посылает его обратно. Отправитель, восприняв этот отклик, может сравнить временную метку, записанную им в пакет, с текущим показанием внутренних часов и определить время распространения пакета (RTT - round trip time). Размер поля Данные не регламентирован и определяется предельным размером IP-пакета.

Так как в пакете ICMP нет поля порт, то при запуске нескольких процессов PING одновременно может возникнуть проблема с тем какому из процессов следует передать тот или иной отклик. Для преодоления этой неопределенности следует использовать уникальные значения полей идентификатор.

Поле идентификатор бывает важно, когда ЭВМ используется как программируемый генератор трафика. В этом случае очередной ICMP-пакет посылается, не дожидаясь прихода отклика. Более того, такие пакеты могут генерироваться несколькими процессами одновременно. В этом случае поле идентификатор становится необходимым, чтобы определить, какому процессу ОС передать очередной отклик.

Время распространения ICMP-запроса, вообще говоря, не равно времени распространения отклика. Это связано не только с возможными изменениями в канале. В общем случае маршруты их движения могут быть различными.

Когда маршрутизатор не может доставить дейтаграмму по месту назначения, он посылает отправителю сообщение "адресат не достижим" (destination unreachable).

Так как собственные часы различных ЭВМ имеют свое представление о времени, протокол ICMP, среди прочего, служит для синхронизации работы различных узлов, если это требуется (запросы временных меток). Протокол синхронизации NTP (network time protocol) описан в RFC-1119.

Когда дейтограммы поступают слишком часто и принимающая сторона не справляется с этим потоком, отправителю может быть послано сообщение с требованием резко сократить информационный поток (quench-запрос), снижение потока должно продолжаться до тех пор, пока не прекратятся quench-запросы.

В глобальной сети таблицы маршрутизации остаются без изменений достаточно долгое время, но иногда они меняются. Если маршрутизатор обнаружит, что ЭВМ использует неоптимальный маршрут, он может послать ей ICMP-запрос переадресации.

Команды переадресации маршрутизатор посылает только ЭВМ и никогда другим маршрутизаторам. Рассмотрим конкретный пример. Пусть некоторая ЭВМ на основе своей маршрутной таблицы посылает пакет маршрутизатору М1. Он, просмотрев свою маршрутную таблицу, находит, что пакет следует переслать маршрутизатору М2. Причем выясняется, что пакет из М1 в М2 следует послать через тот же интерфейс, через который он попал в М1. Это означает, что М2 доступен и непосредственно для ЭВМ-отправителя. В такой ситуации М1 посылает ICMP-запрос переадресации ЭВМ-отправителю указанного пакета с тем, чтобы она внесла соответствующие коррективы в свою маршрутную таблицу.

Маршрутная таблица может загружаться из файла, формироваться менеджером сети, но может создаваться и в результате запросов и объявлений, посылаемых маршрутизаторами. После загрузки системы

маршрутизатор посылает широковещательный запрос. Один или более маршрутизаторов посылают в ответ сообщения об имеющейся маршрутной информации. Кроме того, маршрутизаторы периодически (раз в 450-600 сек.) широковещательно объявляют о своих маршрутах, что позволяет другим маршрутизаторам скорректировать свои маршрутные таблицы. В RFC-1256 описаны форматы ICMP-сообщений такого рода (см. рис. 3.10).

0	8	16	31
Тип (9)	Код (0)	Контрольная сумма	
Число адресов	Длина адреса (2)	Время жизни	
Адрес маршрутизатора (1)			
Уровень приоритета (1)			
Адрес маршрутизатора (2)			
Уровень приоритета (2)			

Рис. 3.10 - Формат ICMP-сообщений об имеющихся маршрутах

Поле *Число адресов* характеризует количество адресных записей в сообщении. Поле *Длина адреса* содержит число 32-битных слов, необходимых для описания адреса маршрутизатора. Поле *Время жизни* предназначено для записи продолжительности жизни объявленных маршрутов (адресов) в секундах. По умолчанию время жизни равно 30 мин. Поля *Уровень приоритета* представляют собой меру приоритетности маршрута по отношению к другим маршрутам данной подсети. Чем больше этот код, тем выше приоритет. Маршрут по умолчанию имеет уровень приоритета 0.

Так как следующий прогон (hop) дейтограммы определяется на основании локальной маршрутной таблицы, ошибки в последней могут привести к зацикливанию пакетов. Для подавления таких циркуляций используется контроль по времени жизни пакета (TTL). При ликвидации пакета по истечении TTL маршрутизатор посылает отправителю сообщение "время истекло.

Когда маршрутизатор выявил какую-либо ошибку, не из числа описанных выше (например, нелегальный заголовок дейтаграммы), посылается сообщение "Конфликт параметров". Это может произойти при неверных параметрах опций.

В процессе прокладывания маршрутов возникает проблема синхронизации работы часов различных рабочих станций. К счастью синхронизация их внутренних часов требуется не так часто (например, при выполнении синхронных измерений), негативную роль здесь могут

играть задержки в каналах связи. Для запроса временной метки другой используется сообщение «Запрос временной метки» (рис. 3.11).

0	8	16	31
Тип (13 или 14)	Код (0)	Контрольная сумма	
Идентификатор		Номер по порядку	
Исходная временная метка			
Временная метка на входе			
Временная метка на выходе			

Рис. 3.11 - Формат ICMP-запроса временной метки

Поле *Тип=13* указывает на то, что это запрос, а *Тип=14* - на то, что это отклик. Поле *Идентификатор* и *Номер по порядку* используются отправителем для связи запроса и отклика. Поле *Исходная временная метка* заполняется отправителем непосредственно перед отправкой пакета. Поле *Временная метка на входе* заполняется маршрутизатором при получении этого пакета, а *Временная метка на выходе* - непосредственно перед его отправкой. Именно этот формат используется в процедурах ping и traceroute. Эти процедуры позволяют не только диагностировать, но и оптимизировать маршруты.

При работе с подсетью важно знать ее маску. Как уже отмечалось выше, IP-адрес содержит секцию адреса ЭВМ и секцию адреса сети. Для получения маски подсети ЭВМ может послать "запрос маски" в маршрутизатор и получить отклик, содержащий эту маску. ЭВМ может это сделать непосредственно, если ей известен адрес маршрутизатора, либо прибегнув к широковещательному запросу.

3.11 Протокол RIP

RIP работает на основе UDP-протокола и использует порт 520 [8]. На каждом хосте, использующем RIP, должно быть установлено программное обеспечение, обрабатывающее RIP-пакеты.

Полное сообщение протокола RIP IP, включая заголовок и данные, инкапсулируется в порцию данных дейтаграммы протокола UDP, которая, в свою очередь, инкапсулируется в IP дейтаграмму (рис. 3.12).

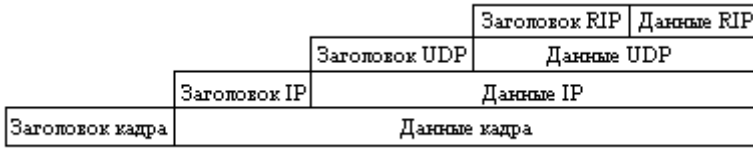


Рис. 3.12 - Инкапсуляция сообщений протокола RIP

Все сообщения протокола RIP состоят из заголовка фиксированной длины и следующего за ним списка сетей, которые могут быть достижимы с использованием данного передающего маршрутизатора (рис. 3.13).

Команда (8 бит)	Версия (8 бит)	Должно быть ноль (16 бит)
Идентификатор адресной схемы (16 бит)		Должно быть ноль (16 бит)
IP-адрес (32 бита)		
Должно быть ноль (32 бита)		
Должно быть ноль (32 бита)		
Количество переходов (32 бита)		
Идентификатор адресной схемы (16 бит)		Должно быть ноль (16 бит)
IP-адрес (32 бита)		
Должно быть ноль (32 бита)		
Должно быть ноль (32 бита)		
Количество переходов (32 бита)		
. . .		
Идентификатор адресной схемы (16 бит)		Должно быть ноль (16 бит)
IP-адрес (32 бита)		
Должно быть ноль (32 бита)		
Должно быть ноль (32 бита)		
Количество переходов (32 бита)		

Рис.3.13 - Формат сообщения протокола RIP

Команда (8 бит). Поле содержит число, обозначающее либо запрос, либо ответ. Команда -запрос запрашивает хост или маршрутизатор об отправке всей таблицы маршрутизации или ее части. Пункты назначения, для которых запрашивается ответ, перечисляются далее в данном пакете. Ответная команда представляет собой ответ на запрос

или какую-нибудь не затребованную регулярную корректировку маршрутизации. Отвечающая система включает в ответный пакет всю таблицу маршрутизации или ее часть. Регулярные сообщения о корректировке маршрутизации включают в себя всю таблицу маршрутизации.

Версия (8 бит). Поле версии определяет реализуемую версию RIP. Поскольку в сети возможны многие реализации RIP, это поле может быть использовано для сигнализации о различных потенциально несовместимых реализациях.

Должно быть ноль(16/32 бит). Поле заполнено нулями.

Идентификатор адресной схемы (16 бит). Это поле определяет конкретное семейство адресов. В сети Internet этим адресным семейством обычно является IP (значение равно 2), но могут быть также представлены другие типы сетей.

IP-адрес (32 бита). В реализациях RIP-Internet это поле обычно содержит какой-нибудь адрес IP (для RIP это может быть либо IP-адресом хоста, либо подсети, либо сети).

Количество переходов (32 бита). Этот показатель представляет собой число пересылок (hop count) или транзитных участков (маршрутизаторов) сети, прежде чем можно будет добраться до пункта назначения.

В каждом отдельном пакете RIP может быть перечислено до 25 пунктов назначения. Для передачи информации из более крупных маршрутных таблиц используется множество пакетов RIP.

Формат сообщения протокола RIP-2 IP

До этого момента рассматривался протокол маршрутизации RIP IP первой версии (RIP-1 IP). Однако существует вторая версия этого популярного протокола (RIP-2 IP), описанная в документе RFC 1388. Вторая версия протокола определяет полезные расширения первой, такие как поддержка CIDR, выполнение аутентификации, поддержка подсетей и групповой передачи. На рисунке(рис. 3.14) приведен формат сообщения второй версии протокола RIP-2 IP.

Команда(8 бит)	Версия(8 бит)	Домен маршрутизации(16 бит)
Идентификатор адресной схемы(16 бит)		Метка маршрута(16 бит)
IP-адрес(32 бита)		
Маска подсети(32 бита)		
Следующий переход(32 бита)		
Метрика(32 бита)		

Рис.3.14 - Формат сообщения протокола RIP-2 IP

Сообщение первой версии протокола содержит нулевые поля. Данные поля предостовляют место для расширений, вносимых второй версией. Протокол RIP-2 IP наследует все поля первой версии, добавляя следующие:

Поле "Домен маршрутизации" используется вместе с полем "Следующий переход" для позволения нескольким автономным системам разделять одну физическую среду передачи.

Поле "Маска подсети" позволяет выполнять маршрутизацию в сформированной структуре подсетей.

Поле "Метка маршрута" предназначено для сигнализации внешних маршрутов и используется протоколами политики маршрутизации(EGP или BGP).

Сообщения протокола RIP-2 IP посылаются с версией равной 2. Однако при получении такого сообщения маршрутизатором поддерживающим протокол RIP-1 IP будет просто игнорировать любые поля, которые длжны содержать нули в первой версии. Следовательно, он будет корректно обрабатывать все записи, которые не используют расширения протокола RIP-2 IP.

Протокол RIP-1 IP не обеспечивает функций безопасности. Любое устройство (рабочая станция или сервер), посылающие сообщения протокола UDP с номером порта, равным 520, будет рассматриваться маршрутизатором как его сосед. Отсутствие процедур проверки авторизации возлагает на администратора дополнительные задачи по управлению правилами. Например, администратор может вручную настроить список авторизованных сетей.

Протокол RIP-2 IP включает процедуру аутентификации. стандарт на протокол определяет, что первая запись в сообщении может быть замещена сегментом аутентификации.

3.12 Протокол OSPF

В OSPF используется принцип контроля состояния канала (link-state protocol). А метрика представляет собой оценку эффективности связи в этом канале: чем меньше метрика, тем эффективнее организация связи. В простейшем случае метрика маршрута может ровняться его длине в пересылках (хопы). Но в общем случае значения метрики могут определяться в гораздо более широком диапазоне. С помощью протокола OSPF маршрутизатор отображает видимый ему граф домена мультисервисной сети, где для каждой пары смежных вершин графа (маршрутизаторов) указано ребро (канал), их соединяющее, и метрика этого ребра.

Для каждой из метрик протокол OSPF строит отдельную таблицу маршрутизации [11]. Чаще всего OSPF выбирает маршрут на основании полосы пропускания канала. Еще одна возможная метрика – задержка – определяет время в микросекундах, которое требуется маршрутизатору для обработки, установки в очередь и передачи пакетов.

В случае, когда имеется несколько маршрутов с одинаковыми значениями метрик, маршрутизаторы могут использовать для передачи пакетов все эти пути, обеспечивая тем самым сбалансированную нагрузку ресурсов сети. Маршрутизатор OSPF помещает в таблицу маршрутизации все маршруты с одинаковыми значениями метрики, и балансировка нагрузки между маршрутами происходит автоматически. Стандартизированный порядок расчета метрик, оценивающих надежность, задержку и стоимость, пока не определен. Вопросы подобного рода решаются администратором сети.

Маршрутизатор, работающий по протоколу OSPF, выполняет последовательно три операции: определяет отношения соседства и смежности с другими маршрутизаторами, обменивается с ними OSPF-пакетами извещений LSA, формируя таким образом полную топологическую карту сети, а затем вычисляет дерево маршрутов, используя алгоритм «первый выбирается кратчайший путь» SPF (Shortest Path First).

3.13 Протокол ARP

Для определения локального адреса по IP-адресу используется протокол разрешения адреса *ARP* [10] *(Address Resolution Protocol).* Протокол ARP работает различным образом в зависимости от того, какой протокол канального уровня работает в данной сети - протокол локальной сети (Ethernet, Token Ring, FDDI) с возможностью широковещательного доступа одновременно ко всем узлам сети, или

же протокол глобальной сети (X.25, frame relay), как правило не поддерживающий широковещательный доступ. Существует также протокол, решающий обратную задачу - нахождение IP-адреса по известному локальному адресу. Он называется реверсивный ARP - *RARP (Reverse Address Resolution Protocol)* и используется при старте бездисковых станций, не знающих в начальный момент своего IP-адреса, но знающих адрес своего сетевого адаптера.

В локальных сетях протокол ARP использует широковещательные кадры протокола канального уровня для поиска в сети узла с заданным IP-адресом.

Узел, которому нужно выполнить отображение IP-адреса на локальный адрес, формирует ARP запрос, вкладывает его в кадр протокола канального уровня, указывая в нем известный IP-адрес, и рассылает запрос широковещательно. Все узлы локальной сети получают ARP запрос и сравнивают указанный там IP-адрес с собственным. В случае их совпадения узел формирует ARP-ответ, в котором указывает свой IP-адрес и свой локальный адрес и отправляет его уже направленно, так как в ARP запросе отправитель указывает свой локальный адрес.

Функционально, ARP делится на две части. Одна - определяет физический адрес при посылке пакета, другая отвечает на запросы других машин. ARP-таблицы имеют динамический характер, каждая запись в ней "живет" определенное время после чего удаляется. Менеджер сети может осуществить запись в ARP-таблицу, которая там будет храниться "вечно". ARP-пакеты вкладываются непосредственно в ethernet-кадры. Формат ARP-пакета показан на рис. 3.15.

0 8 16 24 31

Тип оборудования		Тип протокола
HA-Len	PA-Len	Код операции
Аппаратный адрес отправителя (октеты 0…3)		
Адрес отправителя (октеты 4,5)		IP-адрес отправителя (октеты 0,1)
IP-адрес отправителя (октеты 2,3)		Аппаратный адрес получателя (0,1)
Аппаратный адрес получателя (2,5)		
IP-адрес получателя (октеты 0-3)		

Рис. 3.15 - Формат пакета ARP

HA-Len - длина аппаратного адреса.

PA-Len - длина протокольного адреса (длина в байтах, например, для IP-адреса PA-Len=4).

Тип оборудования - это тип интерфейса, для которого отправитель ищет адрес; код содержит 1 для Ethernet.

Тип протокола – содержит код используемого протокола.

Поле *код операции* определяет, является ли данный пакет ARP-запросом (код = 1), ARP-откликом (2), RARP-запросом (3), или RARP-откликом (4). Это поле необходимо, как поле тип кадра в Ethernet пакетах, они идентичны для ARP-запроса и отклика.

ARP-таблицы строятся согласно документу RFC-1213 и для каждого IP-адреса содержит четыре кода:

Ifindex	Физический порт (интерфейс), соответствующий данному адресу;
Физический адрес	MAC-адрес, например Ethernet-адрес;
IP-адрес	IP-адрес, соответствующий физическому адресу;
Тип адресного соответствия	Это поле может принимать 4 значения: 1 - вариант не стандартный и не подходит ни к одному из описанных ниже типов; 2 - данная запись уже не соответствует действительности; 3 - постоянная привязка; 4 - динамическая привязка;

ARP запросы могут решать и другие задачи. Так при загрузке сетевого обеспечения ЭВМ такой запрос может выяснить, а не присвоен ли идентичный IP-адрес какому-то еще объекту в сети. При смене физического интерфейса такой запрос может инициировать смену записи в ARP-таблице.

В рамках протокола ARP возможны *самообращенные запросы* (**gratuitous ARP**). При таком запросе инициатор формирует пакет, где в качестве IP используется его собственный адрес. Это бывает нужно, когда осуществляется стартовая конфигурация сетевого интерфейса. В таком запросе IP-адреса отправителя и получателя совпадают.

4 Анализатор протоколов Wireshark

Wireshark - это программный анализатор трафика, который позволяет перехватывать информационные потоки, передаваемые по сети. Программа в первую очередь предназначена для сбора информации о сетевых взаимодействиях и для обнаружения и

устранения неполадок в сети. Анализаторы трафика (сниферы) так же часто применяются при разработке новых протоколов и программного обеспечения и в образовательных целях.

Установленная и запущенная на компьютере программа Wireshark позволяет обнаружить и изучить любой протокольный блок данных (Protocol Data Unit, PDU), который был отправлен или получен с помощью любого из установленных на компьютере сетевых адаптеров (Network Interface Card, NIC).

Программа позволяет отслеживать весь исходящий по сети трафик, используя для сетевой карты так называемый «широковещательный режим». Возможности программы несколько напоминают небезызвестное приложение TCP Dump, однако, по сравнению с ним, обладает более расширенной функциональностью касательно сортировки, поиска и фильтрования необходимой информации. Отслеживание информации тем более удобно, что все представление трафика показывается в графическом режиме.

Также программа имеет отличные возможности в поддержке протоколов DNS, FDDI, FTP, HTTP, ICQ, IPV6, IPX, IRC, MAPI, MOUNT, NETBIOS, NFS, NNTP, POP, PPP, TCP, TELNET, X25 и т. д. К тому же, приложение обладает и расширенными возможностями захвата, которые, прежде всего, определяются тем, что программа способна открывать файлы, захваченные с использованием других программ. Ко всему прочему, распознавание большого числа различных протоколов делает программу универсальной, поскольку в ней изначально заложена возможность отображения информации сетевого пакета с анализом значение каждого поля протокола любых уровней. И хоть для захвата данных и используется собственный протокол программы PCAP, тем не менее, она работает со многими форматами именно входных данных. Для формирования своего графического интерфейса приложение использует библиотеку GTK+, что и дает возможность работать с большим количеством входных форматов. Вообще, стоит сказать, что приложение эффективно при условии использовании в сегменте именно концентраторов (хабов), а не коммутаторов (свитчей). В противном случае метод анализа исходящего трафика малоэффективен, поскольку на снифер попадают лишь отдельные фреймы.

Программу можно скачать по адресу компании-разработчика программы [12].

4.1 Первый запуск и начало работы с программой

Для начала сбора пакетов сообщений перехваченных программой по сети, необходимо выбрать пункт главного меню Capture>Interfaces или кнопку на верхней панели инструментов Interface List (рис. 4.1).

Рисунок 4.1 – Начало работы с Wireshark

После этого на экране появится диалоговое окно (рис. 4.2)

Рисунок 4.2 - Capture Interface.

С помощью кнопки Options возможна установка желаемых параметров работы программы. Для того, чтобы начать процедуру захвата, необходимо нажать кнопку Start, после чего интерфейс программы примет вид, изображенный на рисунке 4.3.

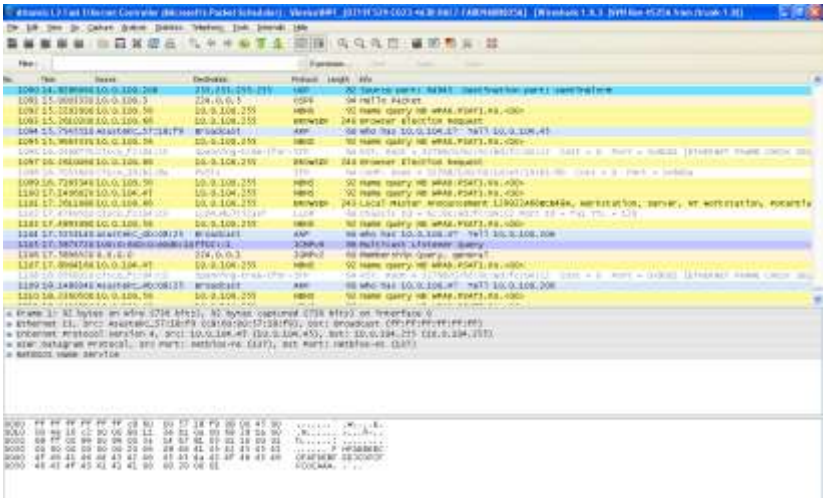


Рисунок 4.3 - Процесс, отображающий захват пакетов

Как видно из изображения, подобно другим анализаторам протоколов, окно Wireshark включает в себя 3 области просмотра с различными уровнями детализации. Верхнее окно содержит список собранных пакетов с кратким описанием, в среднем окне показывается дерево протоколов, инкапсулированных в кадр. Ветви дерева могут быть раскрыты для повышения уровня детализации выбранного протокола. Последнее окно содержит дамп пакета в шестнадцатеричном или текстовом представлении. Программа Wireshark представляет пользователю ряд уникальных возможностей, не поддерживаемых другими анализаторами протоколов. Программа обеспечивает возможность сбора всех пакетов заданного соединения TCP и представления данных в удобном для просмотра формате.

4.2 Начальная настройка программы и запуск захвата трафика

При запуске программы открывается окно со стартовым интерфейсом программы (рис. 4.4).

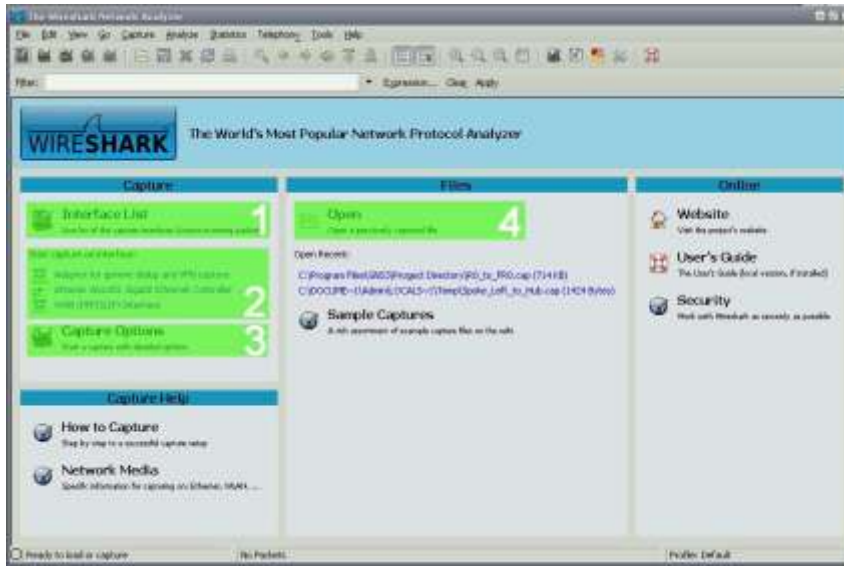


Рисунок 4.4 - Стартовый интерфейс программы.

Interface List (выделенная область 1) – кнопка, при нажатии на которую программа выведет список активных сетевых адаптеров (рис. 4.4), с которых возможен захват трафика.

Start capture on interface (выделенная область 2) – список активных сетевых интерфейсов. Нажатие на любой интерфейс из списка немедленно запустит процесс захвата трафика.

Capture Options (выделенная область 3) – кнопка, при нажатии на которую программа выведет окно настроек процесса захвата трафика (рис. 4.5).

Open (выделенная область 4) – кнопка, позволяющая загружать в программу захваченный ранее и сохраненный файл и отчет о захваченном сетевом трафике.

Раздел *Capture Help* содержит информацию справочного характера по захвату трафика.

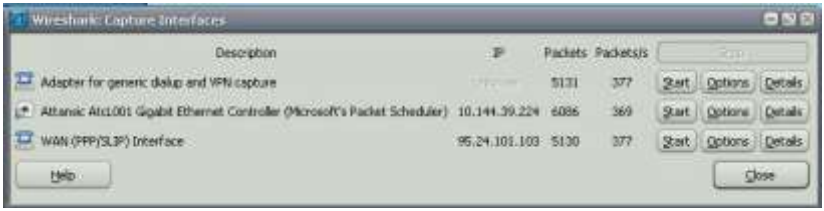


Рисунок 4.5 - Список активных сетевых адаптеров.

Список активных адаптеров имеет вид интерактивной таблицы со следующими полями (рис. 4.5):

Description – Описание адаптера.

IP – Сетевой адрес (если есть).

Packets – Количество захваченных блоков данных (PDU) с момента вызова таблицы.

Packets/s – Скорость обработки (приема и отправки пакетов).

Также напротив каждого интерфейса расположены 3 кнопки:

Start - Начать захват трафика

Options – Вызов окна настроек захвата трафика.

Details – Подробная информация о сетевом адаптере.

При нажатие на кнопку *Capture Options* появится диалоговое окно с настройками захвата (рис. 4.6).

Interface (выделенная область 1) – раздел, в котором выбирается интерфейс для захвата трафика. В этой области расположены два выпадающих меню. Первое (левое) определяет тип используемого интерфейса: Local (локальный) или Remote (удаленный). Второе (правое) меню – сам интерфейс. Очень важно выбрать соответствующий сетевой адаптер, иначе запись кадров будет производиться из другого сегмента сети. В компьютере, имеющем всего один сетевой адаптер, среди возможных сетевых интерфейсов часто присутствует контроллер удаленного доступа.

Рисунок 4.6 - Окно настроек захвата сетевого трафика.

Capture packets in promiscuous mode (выделенная область 2) – захват пакетов в режиме приема всех сетевых пакетов. Если эта опция включена, то программа будет захватывать все PDU, которые принимает сетевой адаптер. Если опция выключена – программа будет захватывать только PDU, предназначенные компьютеру, на котором она установлена.

Buffer size — размер буфера захвата (по умолчанию 1 Мб). При малом размере буфера существует опасность того, что при его заполнении запись новых кадров будет производиться поверх записанных ранее.

Limit each packet to — запись только нескольких первых байт (определяется установленным значением параметра) каждого кадра.

Capture Filter — фильтр захвата. Фильтр захвата экономит объем буфера, отбрасывая «лишний мусор», однако увеличивает нагрузку на процессор, вследствие чего некоторые кадры могут быть потеряны. Поэтому в некоторых случаях вместо фильтра записи предпочтительнее использовать фильтр отображения кадров в буфере, а запись производить без фильтрации.

Capture File(s) — файл захвата. Опция полезна при осуществлении захвата трафика в течение длительного периода времени.

Stop Capture — условия автоматического завершения захвата.

Display Options (выделенная область 3) – отображение захвата пакетов. Опции увеличивают нагрузку на процессор, вследствие чего некоторые кадры могут быть потеряны. Содержит три опции:

- *Update list of packets in real time* – обновление списка в реальном времени.

- *Automatic scrolling in live capture* – автоматическая прокрутка списка при захвате трафика. Если эта опция включена, программа будет автоматически удерживать в окне вывода захваченной информации последние захваченный PDU.

- *Hide capture info dialog* – скрыть информационно-диалоговое окно захвата.

Name Resolution (выделенная область 4) – преобразование имен. Содержит три опции:

- *Enable MAC name resolution* – включить преобразование MAC адресов. Эта опция включает автоматическое преобразование физических адресов устройств в более понятный для человека формат.

Пример: 00:09:5b:01:02:03 → **Netgear**_01:02:03. Выделенная часть сетевого адреса закреплена за производителем **Netgear,** поэтому программа преобразовала ту часть в название производителя.

Примечание: если включена эта опция, то в некоторых случаях программа выводит DNS имя вместо MAC-адреса.

Enable network name resolution - включить преобразование сетевых имен. Эта опция включает автоматическое преобразование сетевых адресов устройств в DNS имена устройств.

Enable transport name resolution - включить преобразование TCP/UDP. Эта опция включает автоматическое преобразование TCP/UDP закрепленных ха определенными протоколами портов в названиях этих протоколов.

Stop Capture (невыделенная область) – интервал, после которого остановится захват:

after X packet(s) - через *X* пакетов;

after X megabyte(s) - через *X* мегабайт;

after X minute(s)- через *X* минут.

После того, как начат захват, появляется информационно-диалоговое окно (рис. 4.7). Здесь отображается информация по захваченным пакетам (*Captured Packets*) и указано время, на протяжение которого ведется этот захват (*Running*).

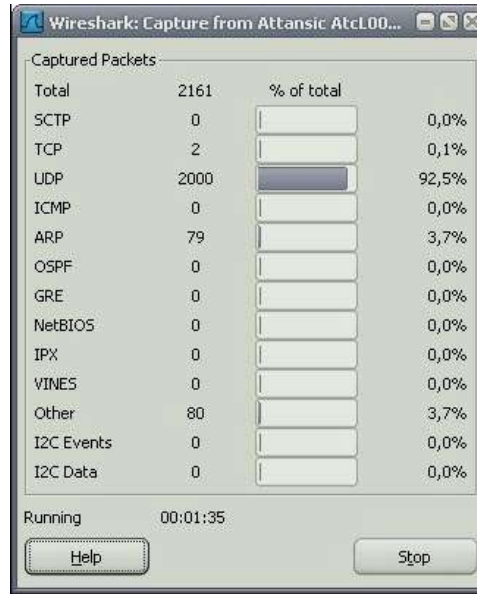


Рисунок 4.7 - Информационно-диалоговое окно захвата.

В первом столбце (Total) указываются названия протоколов. В данном примере (рис. 4.7) указаны наиболее распространенные протоколы.

Во втором столбце приводится количество захваченных PDU приведенных протоколов. Сверху - общее число захваченных пакетов.

В третьем (% of total) и четвертом столбцах показаны графическое и числовое отображение процентного отношения захваченных PDU конкретного протокола к общему числу захваченных пакетов.

4.3 Главное рабочее окно программы

После выбора интерфейса и запуска захвата PDU программа откроет окно, показанное на рисунке 4.8.

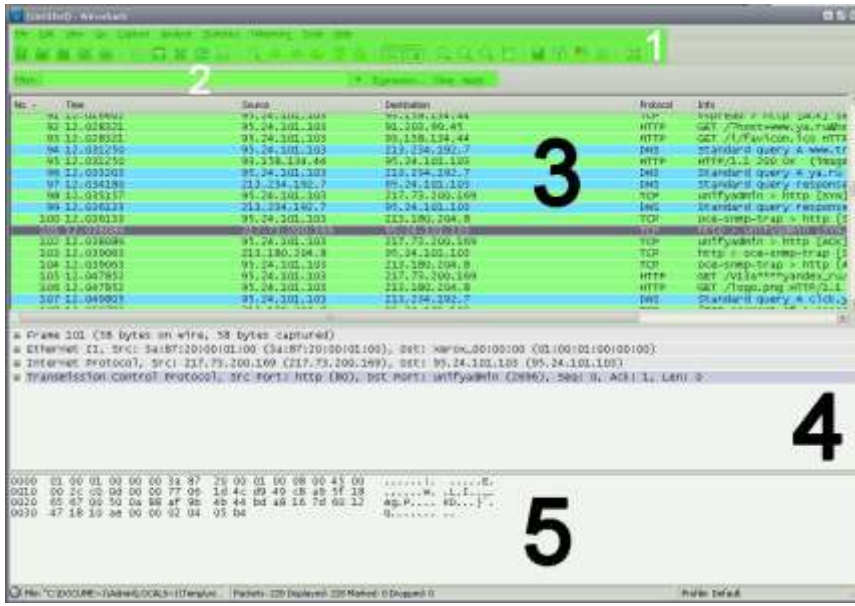


Рисунок 4.8 - Окно отображения захваченного трафика

Верхняя выделенная область содержит меню программы и панель инструментов, предоставляющую доступ к наиболее часто используемым функциям программы.

Вторая выделенная область отображает фильтр, позволяющий производить выборочный захват пакетов.

В третьей области приводится список захваченных пакетов, отображающий краткую информацию по ним.

Выделенная область номер 4 – информационное поле, в котором содержится подробная информация по выбранному PDU.

Пятая область отображает данные в шестнадцатеричной и текстовой форме то строки, которая подсвечена в информационном поле выше.

Рассмотрим более подробно каждую из областей.

4.4 Панель инструментов

Панель инструментов представлена на рисунке 4.9.

Рисунок 4.9 - Панель инструментов

Ниже в таблице 4.1 подробно описаны все кнопки.

Таблица 4.1. Описание панели инструментов

№	Кнопка	Название кнопки	Соответствующая опция в меню	Функции кнопки
1		Interfaces…	Capture / Interfaces…	Вызов списка активных сетевых адаптеров (Рисунок X).
2		Options…	Capture / Options…	Вызов окна настроек захвата сетевого трафика (Рисунок X).
3		Start…	Capture / Start…	Старт захвата трафика с текущими параметрами захвата.
4		Stop…	Capture / Stop…	Остановить захват трафика.
5		Restart…	Capture / Restart…	Перезапустить захват трафика с текущими параметрами.
6		Open…	File / Open…	Открыть файл с отчётом о захваченном трафике.
7		Save As…	File / Save As…	Сохранить текущий отчёт о захваченном трафике в файл.
8		Close…	File / Close…	Закрыть текущий отчёт о захваченном трафике.
9		Reload…	View / Reload…	Закрыть и открыть заново текущий отчёт о захваченном трафике.
10		Print…	File / Print…	Распечатать текущий отчёт о захваченном трафике.
11		Zoom In…	View / Zoom In…	Увеличить размер шрифта.
12		Zoom Out…	View / Zoom Out…	Уменьшить размер шрифта.
13		Normal Size…	View / Normal Size…	Установить размер шрифта, используемый по умолчанию.
14		Preferences…	Edit / Preferences…	Вызов меню настроек.
15		Help…	Help / Contents…	Вызов справки.

4.5 Фильтр

Фильтр позволяет настроить программу Wireshark на отображение только определённого фильтра сетевого трафика.

Фильтр может применяться как при захвате трафика в реальном времени, так и при анализе захвата, сохранённого в файле.

Панель фильтра представлена на рисунке 4.10.

Рисунок 4.10 - Панель фильтра.

В таблице 4.2 приводится описание полей фильтра.

Таблица 4.2.

№	Кнопка / Поле	Название Кнопки / поля	Функции кнопки / поля
1	Filter:	Filter:	Вызов диалогового окна для создания и сохранения пользовательских фильтров (Рисунок X).
2		Filter Input	Поле ввода фильтра.
3			Вызов списка применённых ранее фильтров.
4	Expression...	Expression...	Вызов диалогового окна, позволяющего выбирать фильтры из базы данных программы.
5	Clear	Clear	Очистить поле ввода фильтра.
6	Apply	Apply	Применить фильтр.

Для задания фильтра необходимо:

1. Ввести фильтр в поле ввода.
2. Нажать кнопку “*Apply*”.

Если фильтр введён в соответствии с правилами построения фильтров, то цвет поля ввода будет зелёным (рис. 4.11), если фильтр введён с ошибкой – красным (рис.4.12).

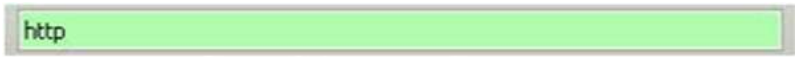


Рисунок 4.11 - Фильтр введен правильно.

Рисунок 4.12 - Фильтр введен неверно.

4.6 Построение фильтров

Фильтрацию, применяемую в программе Wireshark можно условно разделить на две категории:

• Фильтрация по определённым протоколам.

• Фильтрация по определённым значениям полей в заголовках протоколов.

Для применения фильтрации по определённому протоколу необходимо ввести имя протокола в поле ввода фильтра.

Пример выполнения фильтрации по протоколу HTTP показан на рисунках 4.13 и 4.14.

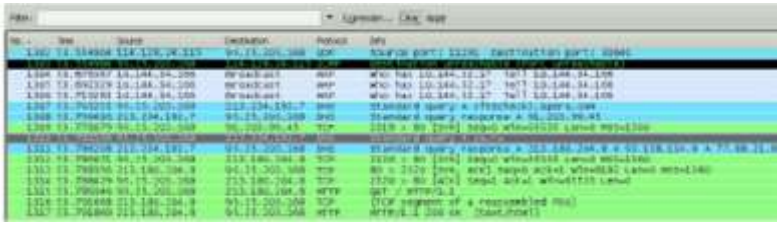


Рисунок 4.13 - Вывод программы до применения фильтра.

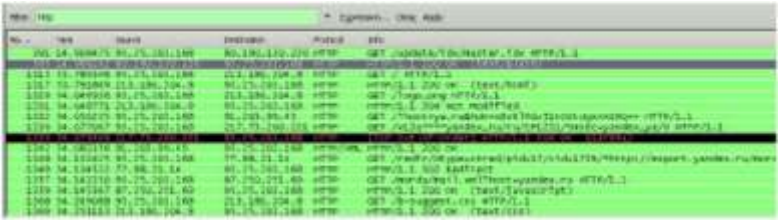


Рисунок 4.14 - Вывод программы после применения фильтра.

Фильтрация по определённому значению поля в заголовках протоколов строится по следующему синтаксису:

Поле - Оператор сравнения – Значение.

Некоторые обозначения полей и операторы сравнения, которые могут использоваться при построении фильтров, представлены в таблицах ниже.

Таблица 4.3 Краткий перечень обозначения полей

Поле	Описание
eth.addr	Физический адрес источника или получателя в кадре протокола Ethernet.
eth.dst	Физический адрес получателя в кадре протокола Ethernet.
eth.src	Физический адрес источника в кадре протокола Ethernet.
eth.len	Длина кадра протокола Ethernet.
ip.addr	Сетевой адрес источника или получателя в пакете протокола IP.
ip.dst	Сетевой адрес получателя в пакете протокола IP.
ip.src	Сетевой адрес источника в пакете протокола IP.
ip.proto	Обозначения протокола, который был инкапсулирован в пакет IP.
tcp.ack	Подтверждения (ACK) протокола TCP
tcp.port	Порт источника или получателя в сегменте протокола TCP.
tcp.dstport	Порт получателя в сегменте протокола TCP.
tcp.srcport	Порт источника в сегменте протокола TCP.
udp.port	Порт источника или получателя в сегменте протокола UCP.
udp.dstport	Порт получателя в сегменте протокола UCP.
udp.srcport	Порт источника в сегменте протокола UCP.
dns.qry.name	Имя сетевого ресурса в DNS запросе.
dns.resp.name	Имя сетевого ресурса в DNS ответе.

Таблица 4.4. Операторы сравнения

Оператор		Значение	Примеры
= =	eq	Равно	ip.addr==192.168.1.1 Отображать только те пакеты протокола IP, в которых сетевой адрес отправителя или получателя равен 192.168.1.1 eth.dst==ff:ff:ff:ff:ff:ff Отображать только широковещательный (broadcast) кадры протокола Ethernet.
!=	ne	Не равно	ip.dst!=255.255.255.255 Не отображать широковещательные (broadcast) пакеты протокола IP
>	gt	Больше	tcp.dstport□ 10000 Отображать только те дейтаграммы протокола TCP, в которых порт получателя больше 10000
<	lt	Меньше	tcp.dstport□ 1024 Отображать только те дейтаграммы протокола TCP, в которых порт получателя меньше 1024

При построении фильтра можно комбинировать два и более условия, используя логические операторы. Комбинирование условий при построении операторов производится по следующему принципу: Условие1 - Логический оператор - Условие2 - Логический оператор.

В качестве условия может использоваться как фильтрация по протоколам, так и фильтрация по значениям определенных полей в протоколах.

В таблице 4.5 представлены некоторые логические операторы для комбинированных условий.

Таблица 4.5. Комбинированные логические операторы

Оператор		Значение	Примеры
&&	and	И	**ip.src==192.168.1.1 && ip.dst==192.168.1.10** Отображать только сообщения отправленные устройством с сетевым адресом 192.168.1.1 для устройства с сетевым адресом 192.168.1.10
||	or	ИЛИ	**eth.dst==ff:ff:ff:ff:ff:ff || ip.dst==255.255.255.255** Отображать только широковещательные кадры протокола Ethernet или пакеты протокола IP.
!	not	НЕ (Отрицание)	**!arp** Не отображать PDU протокола ARP.

4.7 Поле захваченных PDU

В поле списка захваченных PDU (рисунок 4.15) выводится сводная информация по всему трафику, захваченному с помощью программы Wireshark.

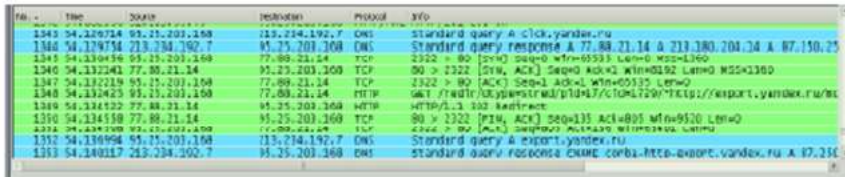


Рисунок 4.15 - После списка захваченных PDU

Сводная информация выводит в виде таблицы со следующими полями:

No – порядковый номер захваченного PDU. При использовании фильтра порядковый номер не изменяется.

Time – временная отметка, обозначающая время (в секундах), прошедшее с момента захвата PDU.

Source – сетевой адрес отправителя.

Destination - сетевой адрес получателя.

Protocol – протокол.

Info – дополнительная информация о захваченном PDU.

На рисунке 4.16 приведен пример сводной инфомации о захваченной PDU.

Рисунок 4.16 - Пример записи в списке захваченных PDU.

4.8 Информационное поле

В информационном поле (рисунок 4.17) отображается подробная информация о захваченном PDU, выделенном в поле списка захваченных PDU.

Рисунок 4.17 - Информационное поле программы.

В верхней части в списке захваченных пакетов выделена одна из строк с помощью нажатия левой кнопки мыши. Программа помечает текущую выделенную запись серым цветом.

В нижней выделенной области содержится подробная информация о выделенном PDU.

Информация о выделенном PDU выводится в виде иерархического списка. Эта иерархия соответствует порядку инкапсуляции данных, применяемой при использовании протоколов стека TCP/IP для передачи информации между устройствами. На рисунке 4.18 показан пример вывода информации о захваченном PDU протокола HTTP.

1. Верхняя выделенная область содержит список захваченных PDU. Серым цветом подсвечивается один из пунктов.

2. В этом вложенном списке содержится справочная информация (время захвата, длина пакета и т. д.) о захваченном пакете, отмеченном в верхнем разделе. В данном случае этот список носит название Frame 14.18.

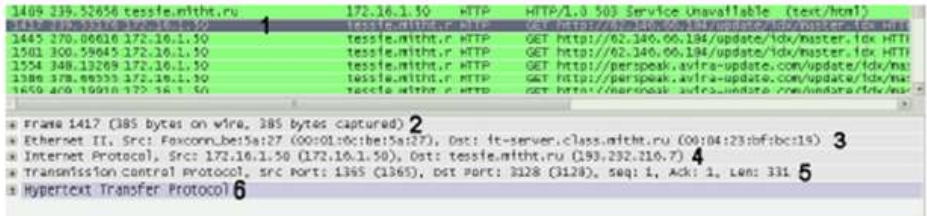


Рисунок 4.18 - Информация о захваченном PDU протокола HTTP.

3. Раскрывающийся список Ethernet II содержит информацию о заголовке протокола канального уровня. В данном случае это протокол Ethernet.

4. Раскрывающийся список Internet Protocol, содержащий информацию о заголовке протокола сетевого уровня. . В данном случае это протокол IP.

5. Раскрывающийся список Transmission Control Protocol, содержащий информацию о заголовке протокола транспортного уровня. В данном случае это протокол TCP.

6. Раскрывающийся список Hypertext Transfer Protocol, содержащий информацию о заголовке протокола прикладного уровня. В данном случае это протокол HTTP.

4.9 Интерпретация вложенных списков

Каждый вложенный список представляет собой последовательность полей (всех или основных), содержащих в заголовке протокола, используемого при инкапсуляции данных.

Порядок полей в списке соответствует порядку полей в заголовке протокола.

Протокол Ethernet

Стандарты Ethernet определяют проводные соединения и электрические сигналы на физическом уровне. Формат кадров и протоколы управления доступом к среде – на канальном уровне модели OSI.

Схематическое изображение кадры протокола Ethernet и соответствующий вывод программы Wireshark показаны на рисунке 4.19. Цветом обозначены поля, выводимые программой.

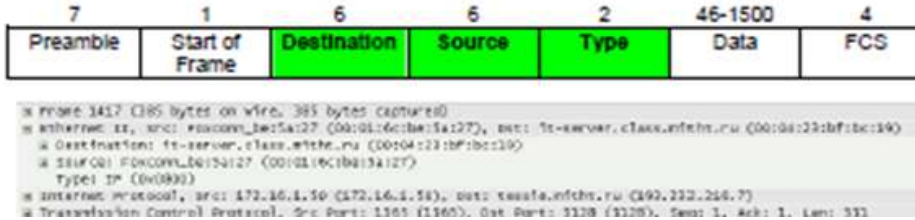


Рисунок 4.19 - Поля заголовка кадра протокола Ethernet.

Информацию в заголовке можно интерпретировать следующим образом:

Ethernet II – кадр протокола Ethernet.

Src: Foxconn_be: 6a:27 (00:01:6c:be:Sa:27) – физический адрес устройства отправителя 00:01:6c:be:Sa:27, производитель сетевой карты – компания Foxconn.

Dst: it-server.clacc.mltnt.ru (00:04:23:bf:bc:19) – физический адрес устройства получателя 00:04:23:bf:bc:19, DNS имя устройства it-server.clacc.mltnt.ru.

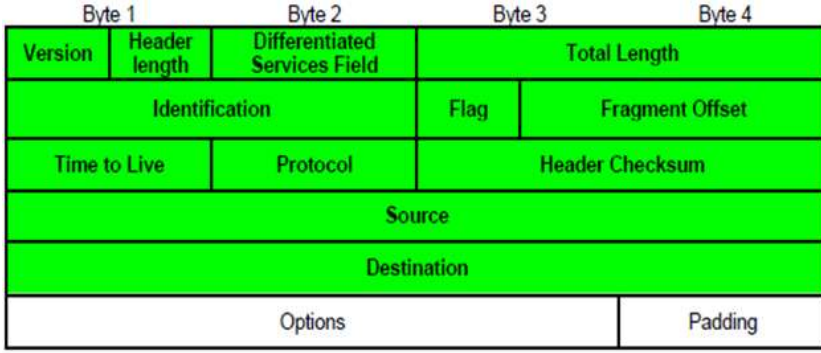
Таблица 4.6

Поле	Описание
Destination	Destination: it-server.clacc.mltnt.ru (00:04:23:bf:bc:19) Интерпретация аналогична интерпретации информации из заголовка списка
Source	Source: Foxconn_be: 6a:27 (00:01:6c:be:Sa:27) Интерпретация аналогична интерпретации информации из заголовка списка
Type	Type: IP (0x0800) – на сетевом уровне используется протокол IPv4. Значение этого пол позволяет устройству определить какому протоколу сетевого уровня следует дальше передать полученные PDU. В данном случае – это протокол IP. Другие наиболее часто встречающиеся значения поля Type: 0x0806 – ARP, 0x86DD – Ipv6.

Протокол IP

Протокол IP — протокол сетевого уровня, обеспечивающий систему глобальной логической адресации для устройств в сети.

Схематичное изображение заголовка пакета протокола IP и соответствующий вывод программы Wireshark показаны на рисунке 4.20. Цветом выделены поля, выводимые программой.




```
⊞ Frame 1417 (385 bytes on wire, 385 bytes captured)
⊞ Ethernet II, Src: Foxconn_be:5a:27 (00:01:6c:be:5a:27), Dst: ft-server.class.mitht.ru (00:04:23:bf:bc:19)
⊟ Internet Protocol, Src: 172.16.1.50 (172.16.1.50), Dst: tessie.mitht.ru (193.232.216.7)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 371
    Identification: 0xd63d (54845)
  ⊞ Flags: 0x04 (Don't Fragment)
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
  ⊞ Header checksum: 0xdc14 [correct]
    Source: 172.16.1.50 (172.16.1.50)
    Destination: tessie.mitht.ru (193.232.216.7)
⊞ Transmission Control Protocol, Src Port: 1365 (1365), Dst Port: 3128 (3128), Seq: 1, Ack: 1, Len: 331
⊞ Hypertext Transfer Protocol
```

Рисунок 4.20 - Поля заголовка пакета протокола IP

Информацию в заголовке можно интерпретировать следующим образом:

Internet protocol, – пакет протокола IP,

src: 172.16.1.50 (172.16.1.50), - сетевой адрес устройства отправителя 172.16.1.50.

Dst: tessie.mitht.ru (193.232.216.7) – сетевой адрес устройства получателя 193.232.216.7. DNS имя устройства получателя tessie.mitht.ru.

Значение наиболее важных полей приведены в таблице ниже.

Таблица 4.7

Поле	Описание
Time to Live	Time to live: 128 – Максимально возможное количество сетевых устройств, которые могут обработать и передать пакет дальше по сети равняется 128.
Protocol	Protocol: TCP (0x06) – На транспортном уровне используется протокол TCP. Значение, этого поля позволяет устройству определить, какому протоколу транспортного уровня следует дальше передать полученное PDU. В данном случае – это протокол TCP. Другие наиболее часто встречающиеся значения поля Protocol: 0x01 – ICMP, 0x11 - UDP
Source	Source: 172.16.1.50 (172.16.1.50), Интерпретация аналогична интерпретации информации из заголовка списка.
Destination	Destination: tessie.mitht.ru (193.232.216.7) Интерпретация аналогична интерпретации информации из заголовка списка.

Протокол TCP

Протокол TCP – протокол транспортного уровня, обеспечивающий надёжную передачу информации между приложениями взаимодействующих устройств.

Схематичное изображение заголовка пакета протокола IP и соответствующий вывод программы Wireshark показаны на рисунке 4.21. Цветом выделены поля, выводимые программой.

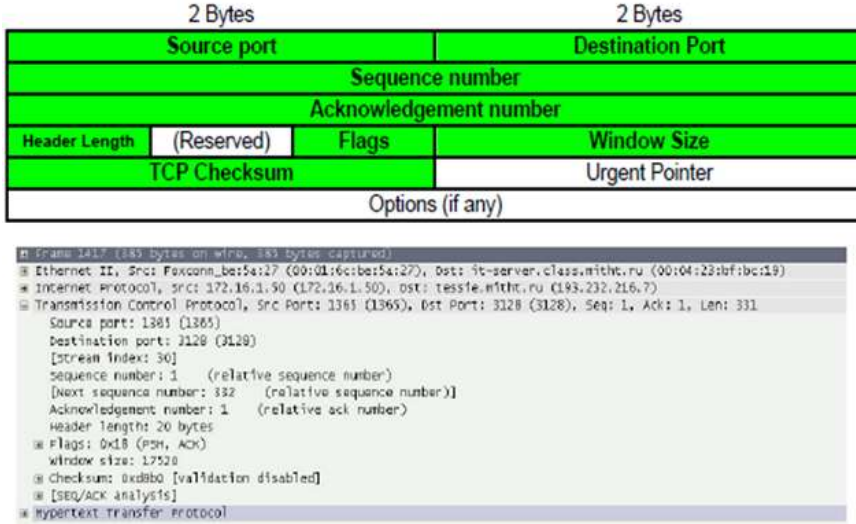


Рисунок 4.21 -. Поля заголовка семента TCP.

Transmission control Protocol, - сегмент протокола TCP.

Src Port: 1365 (1365), - Приложение устройства отправителя использует порт 1365.

Dst Port: 3128 (3128), - Приложение устройства получателя использует порт 3128

Len: 331 – Сегмент содержит 331 байт информации.

Значения наиболее важных полей приведена в таблице ниже.

Таблица 4.8

Поле	Описание
Source port	Source Port: 1365 (1365) Интерпретация аналогична интерпретации информации из заголовка списка.
Destination port	Destination Port: 3128 (3128) Интерпретация аналогична интерпретации информации из заголовка списка.
Sequence number и Acknowledgement number	Sequence number: 1 (.relative sequence number) [Next sequence number: 332 (relative sequence number)] Acknowledgement number: 1 (relative ack number) Поля, использующиеся для организации надёжной доставки информации между приложениями.
Window size	Количество байт, которые могут быть переданы без подтверждения.

4.10 Статистика в Wireshark

Wireshark обеспечивает широкий круг сетевых статистических характеристик, доступ к которым возможен из пункта меню **Statistics**.

Разнообразие статистических характеристик колеблется, начиная от общей информации о загруженном файле захвата (например, число захваченных пакетов), заканчивая статистикой по некоторым протоколам (например, статистика по числу HTTP запросов и ответов).

- Общая статистика:
  - **Summary** – Суммарная статистика по файлу захвата.
  - **Protocol Hierarchy –** Иерархия протоколов захваченных пакетов
  - **Conversations** (Обсуждение). Например, трафика между определенными IP-адресами.
  - **Endpoints (**Конечные точки)**.** Например, IP-адреса источника и получателя**.**
  - **IO Graphs (**Графики)**.** Визуальное представление числа пакетов (и др.) в единицу времени.

- Особенная статистика протоколов:
    - **Service Response Time** (Время отклика). Время между запросом и ответом некоторых протоколов.
    - **Various other** Другие характеристики протоколов.

Некоторая статистика какого-либо протокола требует знание его особых параметров. Поэтому, обладая недостаточными знаниями о протоколе, будет сложно понять определенные статистические характеристики.

Суммарная статистика (Summary)

В этом разделе представлена общая статистика о текущем файле захвата.

Рисунок 4.22 - Окно суммарной статистики

- **File** (Файл)**:** Общая информация о файле захвата.
- **Time** (Время): временные отметки о том, когда были захвачены первый и последний пакеты (и время между ними).
- **Capture** (Захват): информация, полученная с момента захвата (доступна информация, если данные были захвачены из сети, а не загружались из файла).
- **Display** (Отображение): отображение некоторой соответствующей информации.
- **Traffic** (Трафик): отображаются некоторые статистики сетевого трафика. Если установлены фильтры, то в графе *Displayed* будут показаны соответствующие значения. А если некоторые пакеты были отмечены, то значения будут показаны в графе Marked.

4.11 Графики (IO Graphs)

С помощью этого пункта меня, пользователь может строить графики зависимости захваченных сетевых пакетов. Возможен выбор до пяти различных цветов линий.

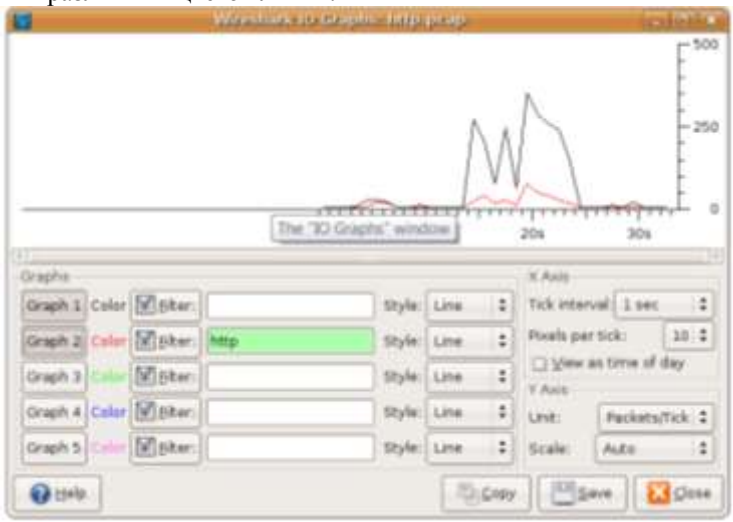


Рисунок 4.23 - Окно «IO Graphs»

Допускается изменение следующих параметров:

- **Graphs (**Графики**)**

    - **Graph 1-5** (Графики 1-5): включает определенные графики (график 1 стоит по умолчанию).
    - **Color** (Цвет): цвет графа (не может быть изменен).
    - **Filter** (Фильтр): показывает фильтр для данного графа (только пакеты, которые фильтруются, будут приняты во внимание для этого графика).
    - **Style** (Стиль): стиль начертания графика (линия/ импульс/точка).
- **X Axis** (Ось X)
    - **Tick interval** (расстояние): расстояние между отсчетами по оси *x* (10/1 минут или 10/1/0.1/0.01/0.001 секунд).
    - **Pixels per tick**: возможность использования 10/5/2/1 отсчетов на единичном интервале.
    - **View as time of day** (просмотр): возможность просматривать отсчеты оси *x* относительно времени суток
- **Y Axis** (Ось Y)
    - **Unit** (единицы измерения): единицы измерения по оси Y (пакеты на интервал, байты в единицу измерения, биты на интервал,) [XXX –описание расширенной функции
    - **Scale** (масштаб): масштаб для единиц измерения по оси Y (логарифмический, автоматический, 10, 20, 50,100,200,500,...)

При нажатии на клавишу **Save** сохраняется отображаемая на экране часть графика в одном из доступных форматов файлов. Функция сохранения доступна только для Wireshark версии выше 0.99.7.

При нажатии на кнопку **Copy** значения выделенных графов скопируются в буфер обмена в формате CSV. Функция копирования доступна только для Wireshark версии выше 0.99.8.

Внимание! При нажатии на графике выделится первый пакет в выбранном интервале.

5 Лабораторные работы

5.1 Лабораторная работа «Анализ пакетного трафика»

Цель работы:

- знакомство с принципами работы программы анализа пакетного трафика Wireshark;
- практическое освоение приемов работы сбора и анализа трафика с помощью утилиты Wireshark.

Задачи

- практическое освоение приемов работы сбора и анализа трафика с помощью утилиты Wireshark (фильтры захвата и отображения);
- получение навыков работы с настройками сохранения.

Выполнения работы

1. Запустите программу Wireshark.

2. Выполните команду меню *Capture -> Options*. В открывшемся диалоговом окне устанавливаются следующие параметры захвата кадров (рис. 5.1).

3. Уберите маркер напротив опции «Capture packets in promiscuous mode» для захвата только «своих» кадров (кадры с широковещательным адресом также будут захватываться). В таком режиме работы число захваченных пакетов будет существенно меньше, что облегчит выполнение заданий.

4. Очистите кэш протокола ARP.

Для этого выполните команду: Пуск → Выполнить. В окне введите команду *cmd* и нажмите ОК. Далее в командной строке наберите *arp – d* и нажмите Enter.

В Wireshark для запуска процесса захвата нажмите кнопку «Start».

В командной строке выполните команду *ping ya.ru* (в качестве параметра команды можно использовать IP-адрес сервера). По завершении команды *ping* остановите захват, нажав кнопку «Stop».

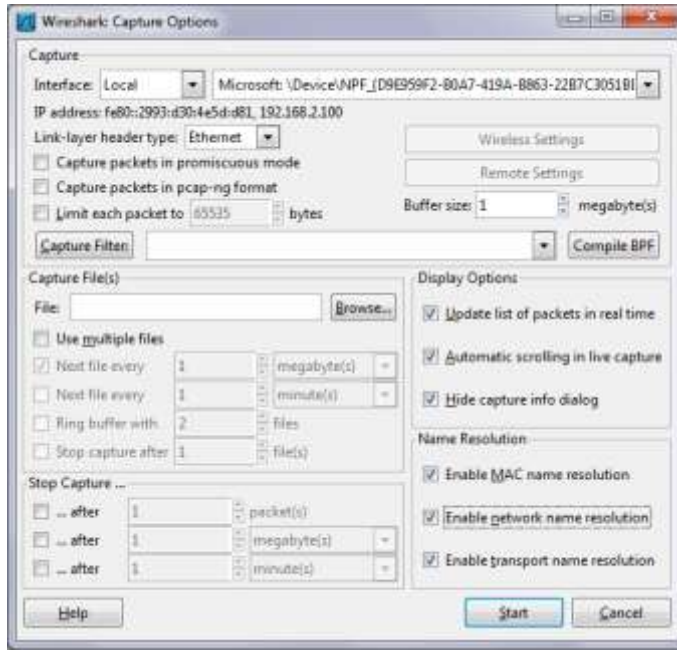


Рисунок 5.1 - Диалоговое окно настройки захвата кадров.

5. Для отображения только ICMP-сообщений в строке ввода «Filter» наберите «icmp» и нажмите кнопку «Apply».

6. Отобразите все кадры, переданные вашим узлом, исключая сообщения ICMP.

При создании выражения фильтрации имейте в виду, что в буфере могут находиться кадры других узлов.

Укажите результирующее выражение фильтрации с необходимыми пояснениями. После просмотра результата для отображения пакетов без фильтрации нажмите кнопку «Clear» в строке фильтра.

7. Пометьте первый и последний пакеты, относящиеся к функционированию команды Ping.

В списке буфера ключевые или наиболее важные для дальнейшего анализа пакеты можно помечать с помощью команды Edit → Mark Packet (toggle) основного меню или команды Mark Packet (toggle) контекстного меню. Также можно отметить все отображаемые пакеты с помощью команды Edit → Toggle Marking Off All Displayed Packets. Эта возможность полезна при дальнейшем поиске таких пакетов в

большом буфере, так как они выделяются другим цветом, а также при сохранении, экспортировании и печати пакетов.

Информация о маркированных пакетах нигде не сохраняется, поэтому все маркеры будут потеряны при выгрузке файла данных.

8. Сохраните в файле с именем «ping» только трафик команды Ping.

Сохранение данных в файле производится из меню File → Save или File → Save As.

Обратите внимание, что сохранить можно:

- все пакеты (All packets);
- только отображаемые (Displayed);
- выбранный пакет (Selected packet only);
- ранее маркированные с помощью основного или контекстного меню (Marked packet only и From first to last marked packet);
- указанный диапазон пакетов (Specify a packet range).

9. Выберите указателем мыши в списке пакетов первый ICMP-запрос и сохраните в файле «1.txt» информацию о нем с максимально возможной детализацией всех заголовков в декодере протоколов.

10. Сохраните все захваченные пакеты в файле lab1.pcap.

11. Отобразите первый и второй пакеты трафика команды Ping в отдельных окнах, сделайте скриншоты, и приведите обоснованные доводы, доказывающие их взаимосвязь.

При составлении отчетов с использованием «скриншотов», а иногда и при анализе данных для просмотра двух пакетов одновременно удобно использовать возможность отображения пакета в отдельном окне.

Это реализуется с помощью команды «Show Packet in New Window» контекстного или основного меню программы «View».

Контрольные вопросы:

1. Назначение программы Wireshark.
2. Как установить время, после которого окончится захват?
3. Как сделать так, чтобы список захваченных пакетов постоянно обновлялся?
4. Для чего применяют фильтры захвата и отображения:
5. Поиск кадров в буфере.

Приложение.

Окно настройки параметров захвата представлен на рис. 5.1

− Interface — сетевой адаптер.

− Buffer size — размер буфера захвата (по умолчанию 1 Мб).

− Capture packets in promiscuous mode — использование режима беспорядочного захвата.

− Limit each packet to — запись только нескольких первых байт (определяется установленным значением параметра) каждого кадра.

− Capture Filter — фильтр захвата.

− Capture File(s) — файл захвата.

− Stop Capture — условия автоматического завершения захвата.

− Display Options — отображение пакетов в реальном времени и автоматический скроллинг окна информации.

− Name Resolution — разрешение имен на физическом, сетевом и транспортном уровнях.

Общий вид окна приложения представлен на рисунке 4.8.

Пользовательский интерфейс программы содержит следующие компоненты:

− меню команд и панель инструментов;

− фильтр отображения пакетов;

− список пакетов в буфере;

− панель отображения декодера протоколов;

− панель отображения пакета в шестнадцатеричном коде и символах ASCII.

Панель со списком пакетов построчно отображает характеристики того или иного пакета (номер по порядку в буфере, время захвата, адреса источника и получателя, тип протокола и общая информация о нем). Перемещение по списку осуществляется с помощью мыши или клавиатуры, причем информация на двух других панелях обновляется автоматически. На панели декодера протоколов, нажимая указателем мыши на символы «+» или «–», можно отображать информацию о полях заголовков протоколов с требуемым уровнем детализации. При выборе того или иного служебного поля в заголовке оно автоматически выделяется на нижней панели, где отображается текущий пакет в шестнадцатеричном виде.

С помощью фильтра отображения можно быстро убрать «мусор». Выражение фильтрации может представлять собой просто название протокола, который присутствует в пакете на том или ином уровне вложенности. Например: arp — для отображения пакетов протокола ARP, tcp — для отображения пакетов, в которых присутствует заголовок протокола TCP.

Более сложные выражения фильтрации строятся с помощью зарезервированных слов, обычно представляющих собой названия полей заголовков того или иного протокола, знака операции сравнения и конкретного значения в шестнадцатеричном или десятичном виде.

Например, выражение с операцией сравнения «Равно» записывается с помощью двойного знака равенства «==» (допустимо использование «eq»). Другие операции сравнения записываются с помощью следующих операторов:

a. != (ne)— не равно, пример: eth.type != 0x0800;
b. > (gt)— больше, пример: tcp.srcport > 1023;
c. < (lt)— меньше, пример: frame.pkt_len lt 60;
d. >= (ge)— больше или равно, пример: frame.pkt_len ge 60;
e. <= (le)— меньше или равно, пример: tcp.dstport <=1023.

Значение любого выражения фильтрации возвращает переменную булевского типа. Таким образом, 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 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.

При необходимости создания сложного выражения фильтрации в меню «Apply as Filter» (рис.5.2) выбирайте команды, начинающиеся с многоточия, при этом новое выражение будет добавлено к результирующему выражению фильтрации.

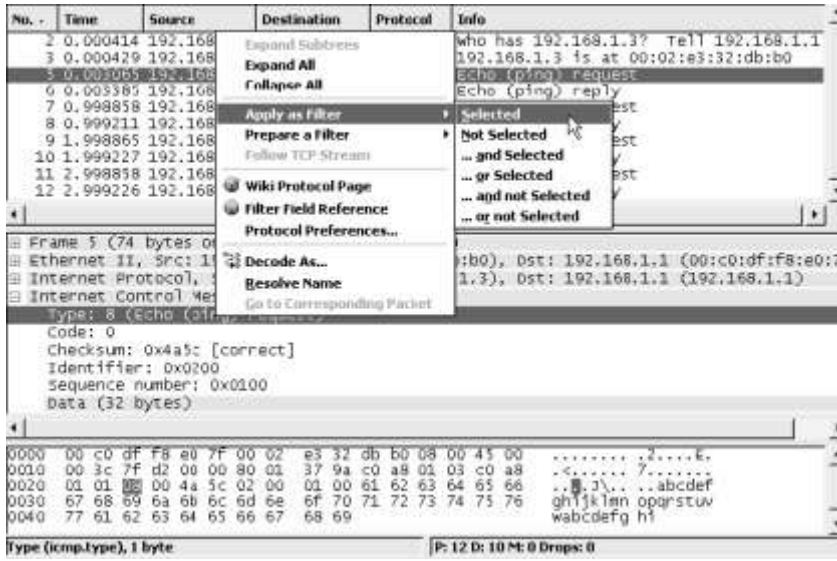


Рисунок 5.2 - Контекстное меню создания фильтра

Поиск кадров в буфере, удовлетворяющих тем или иным критериям, осуществляется с помощью команды меню ***Edit → Find Packet***. Диалоговое окно определения критериев поиска пакетов изображено на рис.5.3.

Рисунок 5.3 - Диалоговое окно поиска кадров

Критерии поиска можно определять в виде выражения фильтрации (Display filter), шаблона в шестнадцатеричном виде (Hex value) и текстовой строки (String) в кодировке ASCII и (или) Unicode. В первом случае можно использовать все допустимые выражения фильтрации и их логические комбинации. Во втором случае указывается шаблон для

поиска в шестнадцатеричном коде. Поиск в строке может осуществляться в области общей информации о пакете (Packet list), в панели декодера протоколов (Packet details) и непосредственно в самом пакете (Packet bytes). Поиск может производиться вверх или вниз по списку пакетов (Direction).

5.2 Лабораторная работа «Анализ протоколов Ethernet и ARP»

Цель работы

анализ протокола ARP.

– ознакомиться с иерархией протоколов для пакетного трафика;
– ознакомиться с протоколами Ethernet и ARP;
– получение навыков выработки правил фильтрации на основе анализа заголовков пакетов протокола ARP;
– освоение возможностей программы Wireshark.

Методика выполнения работы

1. Запустите программу Wireshark.

Выполните команду меню *Capture → Options.*

При анализе протоколов Ethernet и ARP, которые находятся в иерархии протоколов ниже IP, для выключения отображения «лишней» информации на панелях программы целесообразно отключить в программе анализ заголовка IP. Это реализуется с помощью команды «Enabled Protocols…» меню «Analyze» программы «Wireshark». В диалоговом окне данной команды необходимо найти протоколы IPv4 и IPv6, убрать соответствующие маркеры, затем последовательно нажать кнопки «Apply» и «OK» (рис.5.4).

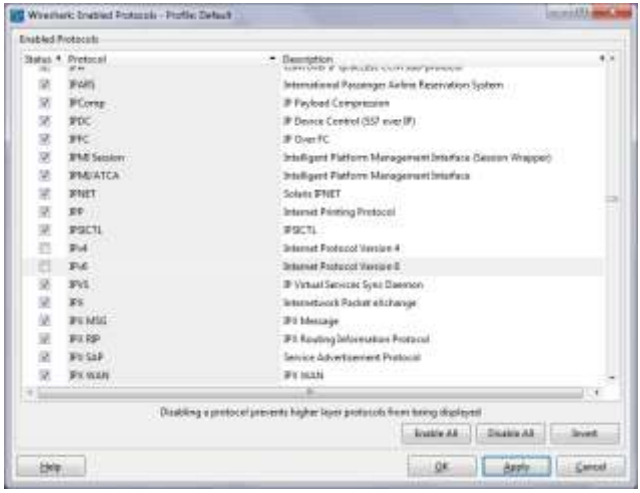


Рисунок 5.4 - Окно выбора протоколов для анализа

2. Отключите анализ заголовка IP.

3. Уберите маркер напротив опции «Capture packets in promiscuous mode» для захвата только «своих» кадров (кадры с широковещательным адресом также будут захватываться). В таком режиме работы число захваченных пакетов будет существенно меньше, что облегчит выполнение заданий.

4. Очистите кэш протокола ARP.

Для этого выполните команду: Пуск → Выполнить. В окне наберите *cmd* и нажмите ОК. В командной строк наберите *arp –d* и нажмите Enter.

В Wireshark для запуска процесса захвата нажмите кнопку «Start». В командной строке выполните команду *ping ya.ru* .

В ряде случаев при отключении анализа заголовка IP отображаемые в списке буфера IP-адреса источника и получателя могут измениться!

Примените к захваченным пакетам фильтр **«arp»** (Сбор пакетов ARP может занять некоторое время!).

Получится примерно следующее (рис. 5.5)

Рис. 5.5- Захваченные пакеты протокола ARP

5. Отобразите в отдельных окнах пакеты запроса и ответа протокола ARP.

В отчете привести:

1. Схему рабочего места с проставленными IP-адресами всех задействованных интерфейсов устройств (IP-адреса вашего компьютера, маршрутизатора, и т.д.),

2. Вид окон с пакетами запроса и ответа протокола ARP.

3. Приведите заголовок Ethernet одного из пакетов в шестнадцатеричном представлении, выделив каждое поле, опишите назначение полей.

4. Выводы.

Контрольные вопросы:

1. Какое значение поля «тип протокола» в кадре Ethernet указывает на протокол ARP?

2. По какому MAC-адресу отправлен запрос и ответ ARP?
3. Каким полем идентифицируются запрос и ответ ARP?
4. В каких полях заголовка ARP передан запрос вашего узла?
5. В каких полях заголовка ARP передан ответ вашему узлу?

Приложение.

Интерпретация вложенных списков

Каждый вложенный список представляет собой последовательность полей (всех, или основных), содержащихся в заголовке протокола, используемого при инкапсуляции данных.

Порядок полей в списке соответствует порядку полей в заголовке протокола.

Протокол Ethernet

Стандарты Ethernet определяют проводные соединения и электрические сигналы на физическом уровне, формат кадров и протоколы управления доступом к среде — на канальном уровне модели OSI.

Схематичное изображение кадра протокола Ethernet и соответствующий вывод программы Wireshark показаны на рисунке 5.6.

Цветом выделены поля, выводимые программой.

Рис. 5.6 - Поля заголовка кадра протокола Ethernet.

Destination - Destination: it-server.class.mitht.ru (00:04:23:bf:bc:19) – Физический адрес устройства получателя 00:04:23:bf:bc:19 и DNS имя устройства – it server.class.mitht.ru.

Source - Source: Foxconn_be:5a:27 (00:01:6c:be:5a:27) - Физический адрес устройства отправителя, 00:01:6c:be:5a:27, производитель сетевой карты – компания Foxconn.

Type - Type: IP (0x0800) – На сетевом уровне используется протокол IPv4. Значение, этого поля позволяет устройству определить, какому протоколу сетевого уровня следует дальше передать полученное PDU. В данном случае – это протокол IP. Другие наиболее часто встречающиеся значения поля Type: 0x0806 – ARP, 0x86DD – IPv6.

Протокол ARP

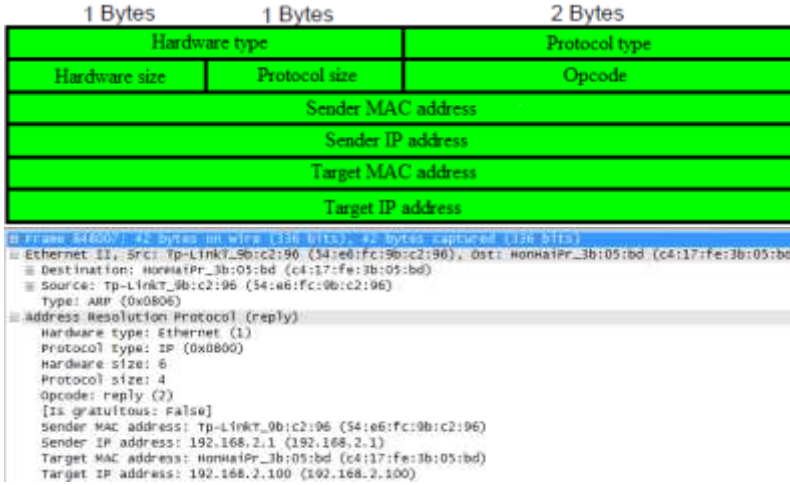


Рис. 5.7- Поля заголовка кадра протокола ARP.

Hardware type - Каждый транспортный протокол передачи данных имеет свой номер, который хранится в этом поле. Например, Ethernet имеет номер 0x0001.

Protocol type - Код протокола. Например, для IPv4 будет записано 0x0800.

Hardware size - Длина физического адреса в байтах. Ethernet адреса имеют длину 6 байт.

Protocol size - Длина логического адреса в байтах. IPv4 адреса имеют длину 4 байта.

Opcode - Код операции отправителя: 1 в случае запроса и 2 в случае ответа.

Sender MAC address - Физический адрес отправителя.

Sender IP address - Логический адрес отправителя.

Target MAC address - Физический адрес получателя. Поле пусто при запросе.

Target IP address - Логический адрес получателя.

5.3 Лабораторная работа «Анализ протокола TCP»

Цель работы

- анализ протокола TCP.
- ознакомиться с протокол TCP,
- получение навыков выработки правил фильтрации на основе анализа заголовков пакетов протокола TCP,
- освоение возможностей программы Wireshark.

Методика выполнения работы

1. Захватите сетевой трафик при обращении к стартовой странице сервера www.google.com.

Для отображения в буфере кадров с протоколом TCP примените соответствующее выражение фильтрации.

В буфере захвата у вас находятся кадры, принадлежащие обмену клиента с сервером по протоколу HTTP, но в рамках текущего упражнения прикладной протокол нас не интересует, поэтому отключите анализ протокола HTTP, ARP.

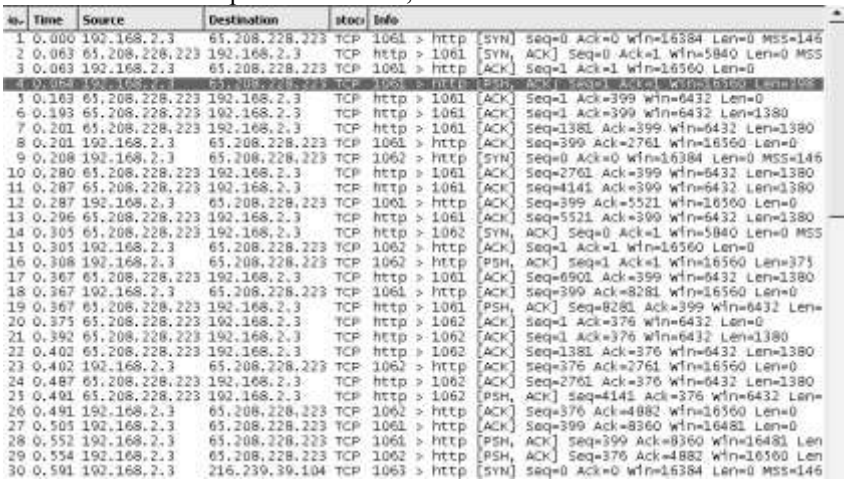


Рис. 5.8 - Отображение информации о протоколе TCP

Каждая TCP-сессия (причем при обращении к одной странице сессий может быть несколько) начинается с обмена тремя TCP-

сегментами с установленными битами SYN, SYN-ACK и ACK. На рис. 1 можно видеть открытие трех сессий TCP (кадры с номерами 1, 2, 3; 9, 14, 15; 30, 31, 32 соответственно).

2. Определите количество сеансов TCP в буфере захваченных пакетов.

На рис. 5.9 также видно, что сеансы TCP начинаются с относительных последовательных номеров, равных нулю. Для того чтобы отобразить реальные последовательные номера, выбранные узлами при взаимодействии, необходимо выполнить команду меню ***Edit -> Preferences***, в появившемся диалоговом окне (фрагмент диалогового окна см. на рис. 5.10) выбрать протокол TCP и убрать маркер в строке параметра «Relative sequence numbers and window scaling».

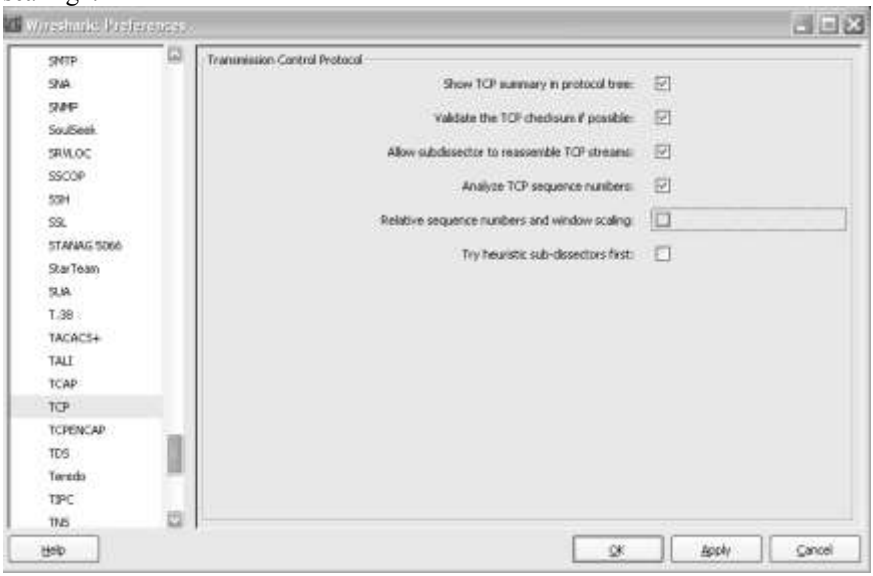


Рис.5.9 - Параметры анализа протокола TCP

3. Отобразите реальные последовательные номера в рамках сеансов TCP.

4. Проанализируйте третий кадр в рамках какого-либо сеанса TCP.

Немаловажная возможность программы Wireshark по анализу TCP трафика состоит в том, что с помощью команды меню ***Statistics -> Conversations*** можно быстро определить все сеансы, имеющиеся в буфере. В диалоговом окне для отображения сеансов TCP необходимо выбрать закладку TCP (рис.5.10) .

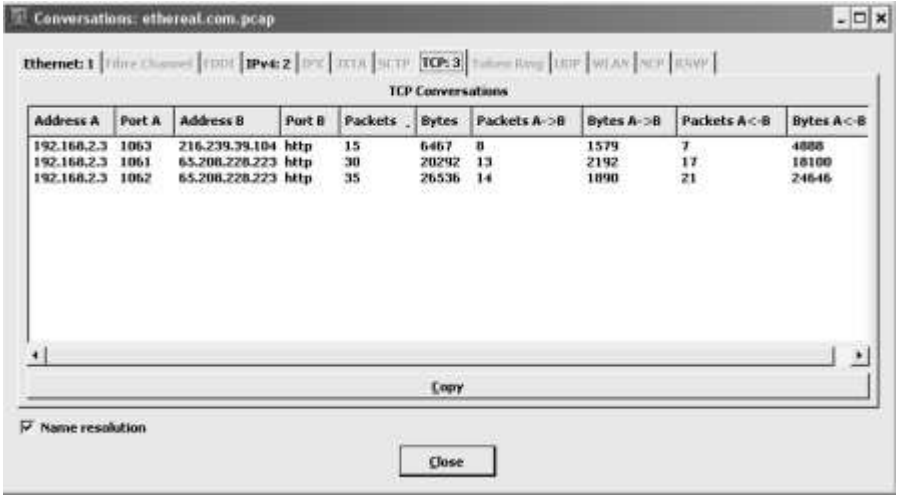


Рис. 5.10 - Статистика по сеансам TCP

5. Отобразите статистику сеансов TCP.
6. Выберите первый сеанс и с помощью контекстного меню *Apply as Filter -> Selected -> A<—>B* отобразите в буфере кадры, принадлежащие этому сеансу.

Для того чтобы быстро просмотреть передаваемые данные в рамках того или иного сеанса, используют команду меню ***Analyze -> Follow TCP Stream***. После выполнения команды на экране появится диалоговое окно, в котором разными цветами будут отображены как запросы клиента, так и ответы сервера.

Кнопка «Entire conversation» с раскрывающимся списком позволяет отобразить обе стороны, участвующие в обмене, или только одну из них.

Определите, что передавалось в рамках захваченных вами сеансов TCP.

В отчете привести:

1. Схему рабочего места с проставленными IP-адресами всех задействованных интерфейсов устройств,
2. Анализ третьего кадра сеанса TCP .
3. Окно статистики по TCP,
4. Результат выполнения пункта 6.
5. Выводы.

Контрольные вопросы:

1. Какие порты используются клиентом и сервером?
2. Какой начальный последовательный номер выбран клиентом?
3. Присутствует ли в этом кадре поле подтверждения, каково его значение?
4. Какая длина заголовка TCP, присутствуют ли данные в этом кадре?
5. Какой бит флагов установлен и для чего он служит?
6. Какие дополнительные опции TCP передаются клиентом в этом кадре?
7. Сохраните кадр и выделите различным цветом поля заголовка TCP, пояснив их назначение.

5.4 Лабораторная работа «Исследование протокола FTP»

Цель работы

- знакомство с принципами работы протокола FTP;
- практическое освоение приемов работы сбора и анализа FTP трафика с помощью утилиты Wireshark.

Задачи

- практическое освоение приемов работы сбора и анализа FTP трафика с помощью утилиты Wireshark (захват пакетов авторизации);

Методика выполнения работы

1. Запустить программу Wireshark.
2. Начать захват пакетов.
3. С помощью браузера Internet Explorer нужно обратиться на FTP сервер.
4. Провести авторизацию с логином <login> и паролем <password>.
5. Закончить захват пакетов для анализа результатов.

После запроса соединения сервер сообщает, что он активен и для входа в систему необходимо имя пользователя (выделенная строка на рис. 5.11.):

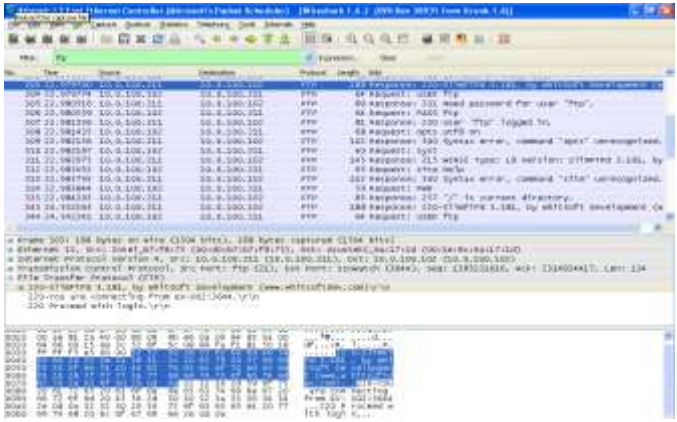


Рисунок 5.11

Пользователь посылает пакет с именем пользователя (рис. 5.13):

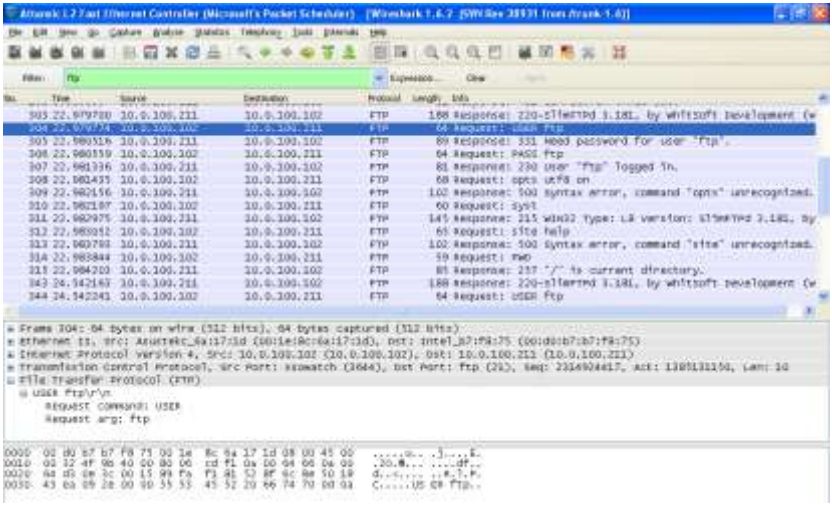


Рисунок 5.12

Сервер сообщает пользователю, что требуется пароль (рис. 5.13.):

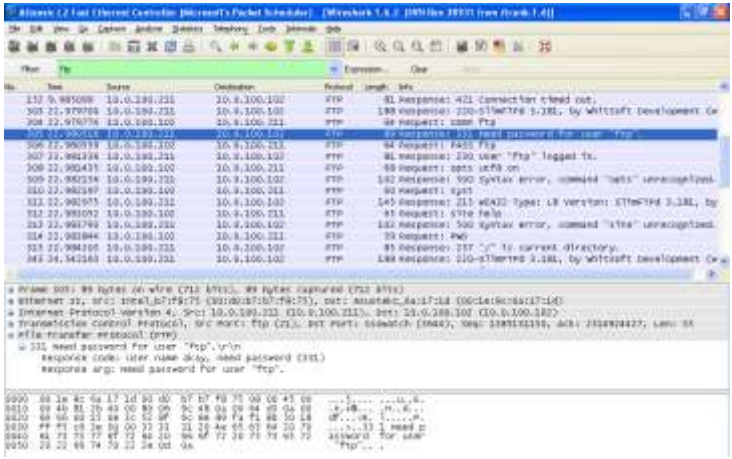


Рисунок 5.13

Пользователь посылает FTP пакет с паролем (рис. 5.14.):

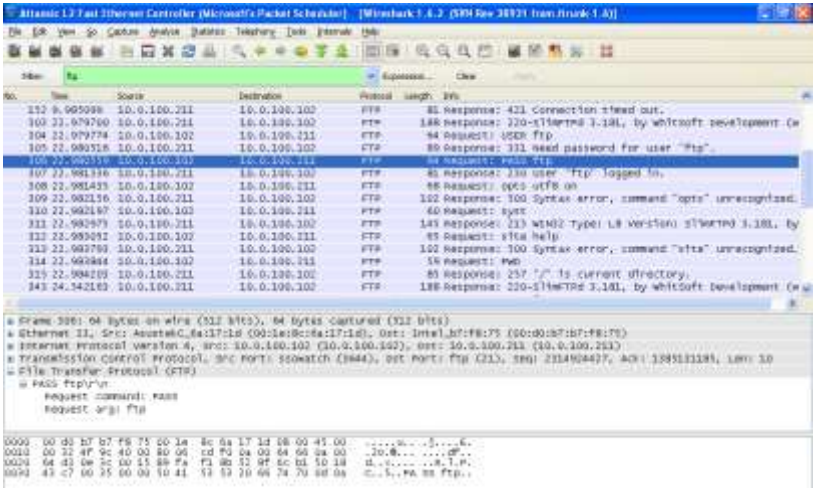


Рисунок 5.14

После проверки соответствия имени пользователя и пароля сервер открывает доступ к FTP архиву (рис. 5.15.):

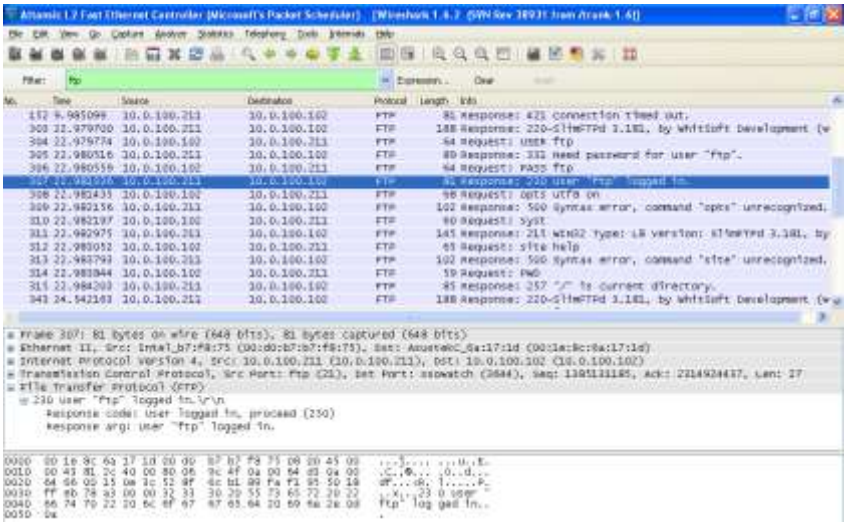


Рисунок 5.15

Контрольные вопросы:

1. Назначение протокола FTP.
2. Модель работы FTP.
3. Алгоритм работы FTP.

5.5 Лабораторная работа «Хранение полученных трафиков»

Цель работы

- знакомство с принципами работы программы анализа пакетного трафика Wireshark;
- практическое освоение приемов работы хранения трафика с помощью утилиты Wireshark ;
- получение навыков работы с графиками и сохранениями.

Методика выполнения работы

1. Запустите программу Wireshark.
2. Выполните команду меню *Capture -> Options*. В открывшемся диалоговом окне устанавливаются следующие параметры захвата кадров (рис. 5.16):

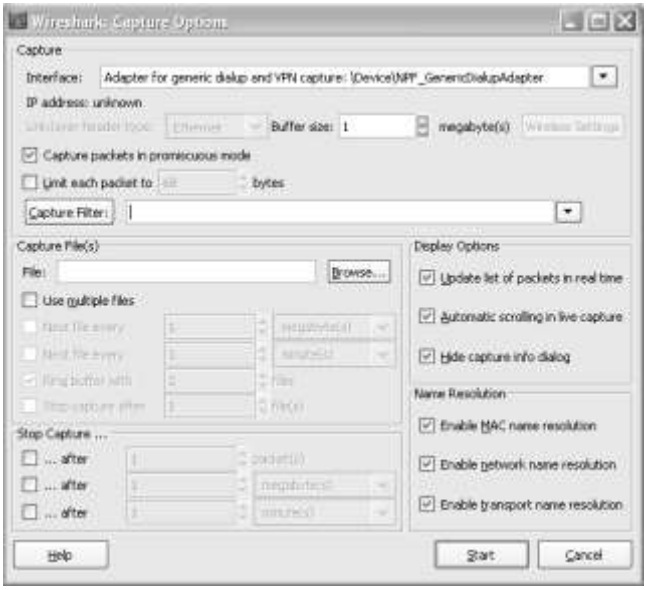


Рисунок 5.16- Окно настройки параметров захвата

3. Уберите маркер напротив опции «Capture packets in promiscuous mode» для захвата только «своих» кадров (кадры с широковещательным адресом также будут захватываться).
4. В Wireshark для запуска процесса захвата нажмите кнопку «Capture». В командной строке выполните команду ping <имя_сервера> (в качестве параметра команды можно использовать IP-адрес сервера). По завершении команды Ping остановите захват, нажав кнопку «Stop».

На экране монитора в программе Wireshark вы увидите несколько панелей с отображением сетевых пакетов, только что записанных в буфер. Общий вид окна приложения представлен на рисунке 5.17.

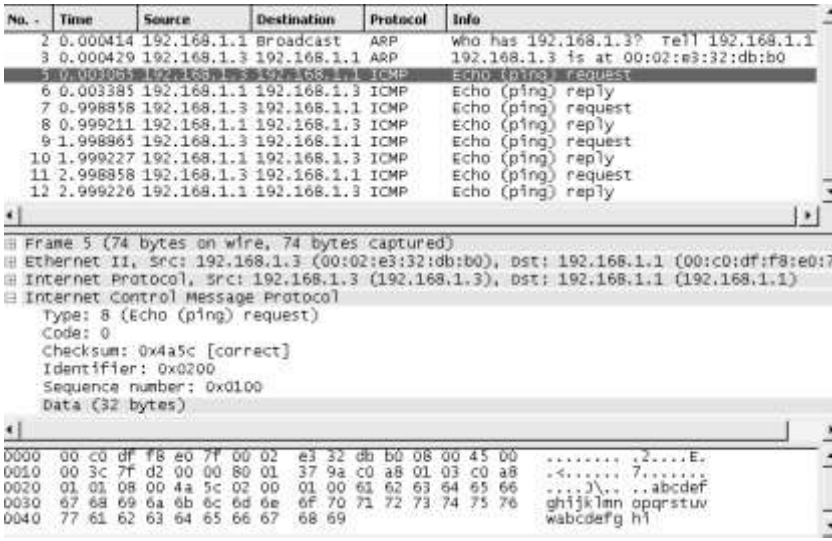


Рисунок 5.17- Общий вид приложения Wireshark

5. Сохраните в файле с именем «ping» только трафик команды Ping.

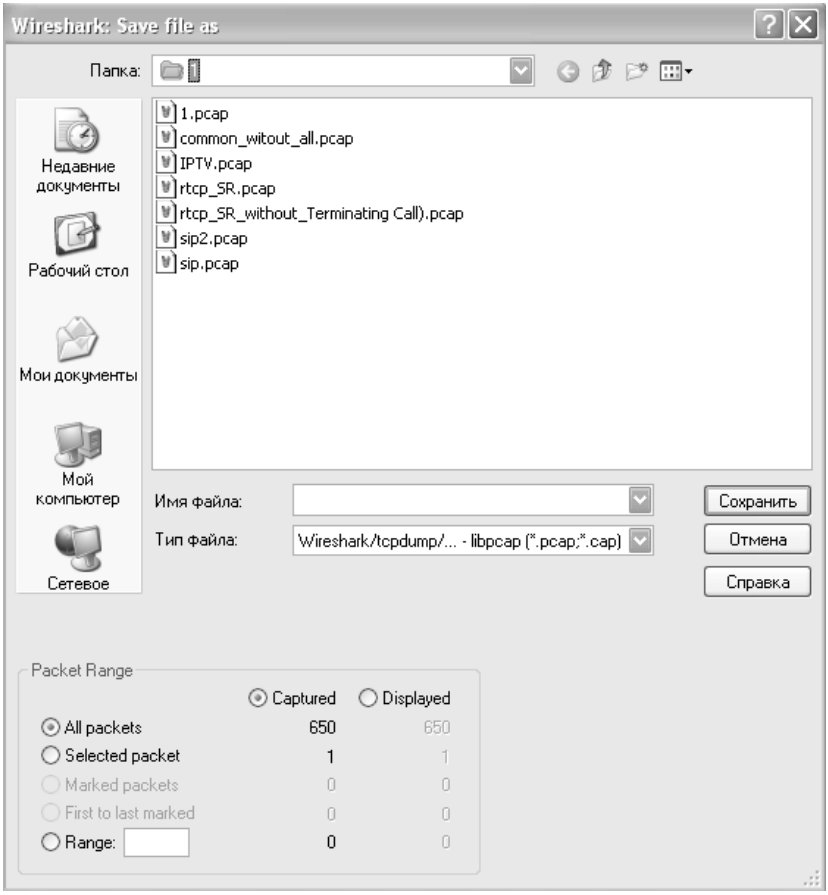


Рисунок 5.18 - Диалоговое окно сохранения данных

Распечатка информации о том или ином пакете или их множестве осуществляется посредством выполнения команды Print основного или контекстного меню.

При печати есть возможность осуществить вывод в указанный файл (Output to file) в виде простого текста (Plain text), определив диапазон распечатываемых пакетов (Packet Range) и формат вывода информации (Packet Format). Опции панели «Packet Range» полностью идентичны опциям соответствующей панели диалогового окна сохранения данных. При определении формата вывода в панели «Packet Format» есть возможность включить общую характеристику

пакета (информацию верхней панели основного окна — «Packet summary line»), информацию, отображаемую на панели декодера протоколов с той или иной степенью детализации (Packet details) и собственно сам пакет в шестнадцатеричном виде (Packet bytes).

6. Сохраните данные полученные с помощью программы Wireshark с помощью Экспорта. Для этого выберете *File -> Export -> File*. Выбирите полное сохранение в формате PDML, суммарное значение в формате SPML, и сохранение в виде текстового документа.

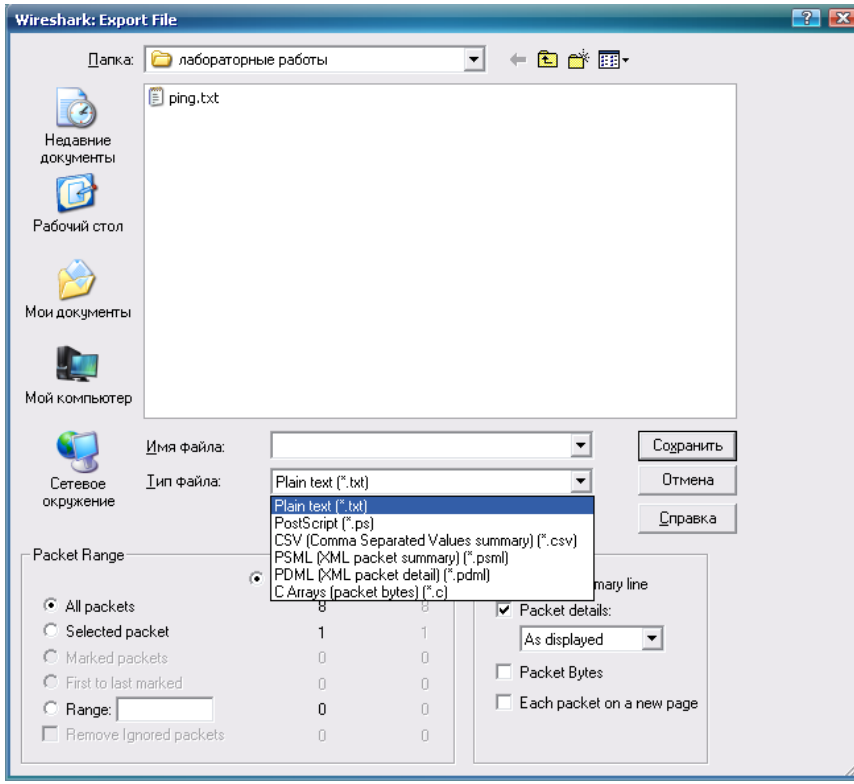


Рисунок 5.19 – Сохранение документа

7. Загрузите с сервера уже снятые трафики для дальнейшего анализа.

8. Постройте график для данных захватов. Для этого откройте меню *Statistics -> IO Graphs.* Измените интервал захвата на 0.1 секунду.

9. Откройте заранее снятые трафики с сервера с помощью сторонних программ. Для формата PDML и SDML используйте программу Excel. Для формата TXT используйте программу Блокнот. Объясните Формат пакета в каждом из форматов. Отметьте достоинства и недостатки каждого формата хранения информации.

Контрольные вопросы:

1. Назначение команды ping.
2. Формат пакета в формате PDML. Его достоинства и недостатки.
3. Формат пакета в формате SDML. Его достоинства и недостатки.
4. Формат пакета в формате TXT. Его достоинства и недостатки.

Список использованной литературы:

1. Степанов С.Н. Основы телетрафика мультисервисных сетей / С. Н. Степанов. – М.: Эко-Трендз, 2010. – 392 с.
2. Олифер, В. Транспортная подсистема неоднородных сетей [Электронный ресурс] / В. Олифер, Н. Олифер // Режим доступа: http://citforum.ru/nets/tpns/contents.shtml
3. Рассохин, Д. Н. World Wide Web - всемирная информационная паутина в сети Internet. Практическое руководство / Д. Н. Рассохин, А.И. Лебедев. – Издательство МГУ, 1997. – 208 с.
4. Case, J. RFC 1157. A Simple Network Management Protocol (SNMP) [Электронный ресурс] / J. Case, M. Fedor, M. Schoffstall, J. Davin// Режим доступа: http://www.ietf.org/rfc/rfc1157.txt.
5. Кульгин, М. Оптимизация работы протокола TCP в распределенных сетях [Электронный ресурс] / М. Кульгин // Сети. – 1999. - №10. - Режим доступа: http://citforum.ru/nets/tcp/optimize01.shtml
6. Паркер, Д. Обзор протокола TFTP [Электронный ресурс] / Д. Паркер // Режим доступа: http://www.netdocs.ru/articles/Understanding-TFTP-Protocol.html
7. Храмцов, П. Б. Администрирование сети и сервисов internet. Учебное пособие [Электронный ресурс] / Режим доступа: http://citforum.ru/nets/services/index.shtml
8. Таненбаум, Э. Компьютерные сети / Э. Таненбаум; пер. сангл. подред. В. Шрага. – 4-издание. – Спб.: Питер, 2006. – 992 с.
9. Олифер, В. Введение в IP-сети Электронный ресурс] / В. Олифер, Н. Олифер // Режим доступа: http://citforum.ru/nets/ip/contents.shtml
10. Семенов , Ю. А . Telecommunication technologies - телекоммуникационные технологии [Электронный ресурс] / Ю. А. Семенов // Режим доступа: http://book.itep.ru/4/44/rut_4411.htm.
11. Олифер Н. Протоколы маршрутизации [Электронный документ] / Н. Олифер // Журнал сетевых решений LAN. – 2001. – № 09. – Режим доступа: http://www.osp.ru/lan/2001/09/024.htm. – 22.11.2006
12. http://www.wireshark.org/download.html.