МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ РАДИОФИЗИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ
Кафедра радиофизики и цифровых медиатехнологий

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ЗАХВАТА СЕТЕВОГО ТРАФИКА

Курсовая работа

Войтика Андрея Павловича
студента 4 курса,
специальность «радиофизика»

Научный руководитель:
старший преподаватель
кафедры радиофизики и цифровых
медиатехнологий
А. В. Микулович

Минск 2018

ОГЛАВЛЕНИ

Е

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ....4
ВВЕДЕНИЕ....5
ГЛАВА 1 ОБЗОР ПРОТОКОЛОВ СЕТИ ИНТЕРНЕТ....6
1.1 Понятие протокола....6
1.2 Стек протоколов TCP/IP....6
1.3 Протокол DHCP....11
1.4 Протокол HTTP....13
1.5 Протокол FTP....13
1.6 Протокол TELNET....14
1.7 Протокол WAIS....14
1.8 Протокол Gorpher....15
1.9 Протокол WAP....15
1.10 Протокол BGP....16
1.11 Протокол ICMP....19
1.12 Протокол SLIP....22
1.14 Протокол состояния связей OSPF....23
1.15 Протокол Frame Relay....26
ГЛАВА 2 ФОРМАТЫ СТРУКТУРНЫХ ЕДИНИЦ ПРОТОКОЛОВ....28
2.1 Форматы кадров технологии Ethernet....28
2.2 Формат IP-пакета....29
2.3 Формат PPP....34
2.4 Формат HTTP-сообщений....36
2.5 Формат UDP....38
2.6 Формат стека TCP/IP....39
ГЛАВА 3 РАЗРАБОТКА ПРОГРАМНОГО ОБЕСПЕЧЕНИЯ....42
3.1 Получение списка устройств....42
3.2 Открытие адаптера и получение пакетов....43
3.3 Захват пакетов без обработчика событий....45
3.4 Запись пакетов в файл....46
3.5 Считывание информации о пакетах из файла....46

ЗАКЛЮЧЕНИЕ.....48
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.....49

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

OSI - Open System Interconnection;
ANSI - American National Standarts Institute;
CCITT - Consultative Committee for International Telegraphy;
SLIP - Serial Line Interface Protocol;
PPP - Point to Point Protocol;
IP - Internet Protocol;
ARP - Address Resolution Protocol;
TCP - Transmission Control Protocol;
UDP - User Datagram Protocol;
RIP - Routing Internet Protocol;
OSPF - Open Shortest Path First;
ICMP - Internet Control Message Protocol;

ВВЕДЕНИЕ

Перехват пакетов (или перехвата пакетов) — это процесс сбора, всех пакетов проходящих через определенный сетевой интерфейс. Захват сетевых пакетов в наших приложениях позволяет реализовывать такие приложения как: мониторинг сети, сетевые анализаторы и средства безопасности. Одним из средств реализующих перехват сетевых пакетов, является библиотека SharpPcap. Цель SharpPcap - обеспечить основу для захвата, инъекции и анализа сетевых пакетов для .NET приложений.

Для более детального анализа сетевого трафика требуется удаление служебной информации из сетевого трафика.

Целью данной курсовой работы является разработка программного обеспечения для захвата сетевого трафика.

Для достижения поставленной цели необходимо решить следующие задачи:

1. рассмотреть протоколы, используемые в сети интернет;
2. изучить форматы структурных единиц протоколов;
3. разработать программное обеспечение для сетевого трафика.

ГЛАВА 1
ОБЗОР ПРОТОКОЛОВ СЕТИ ИНТЕРНЕТ

1.1 Понятие протокола

Очевидно, что рано или поздно компьютеры, расположенные в разных точках земного шара, по мере увеличения своего количества должны были обрести некие средства общения. Такими средствами стали компьютерные сети. Сети бывают локальными и глобальными. Локальная сеть - это сеть, объединяющая компьютеры, географически расположенные на небольшом расстоянии друг от друга - например, в одном здании. Глобальные сети служат для соединения сетей и компьютеров, которых разделяют большие расстояния - в сотни и тысячи километров. Интернет относится к классу глобальных сетей. Протокол - это совокупность правил, в соответствии с которыми происходит передача информации через сеть [1].

Основные протоколы используемые в работе Интернет:

- TCP/IP
- POP3
- SMTP
- FTP
- HTTP
- IMAP4
- WAIS
- Gorpher
- WAP

1.2 Стек протоколов TCP/IP

На самом деле TCP/IP является не одним протоколом, а целым набором протоколов, работающих совместно. Он состоит из двух уровней. Протокол верхнего уровня, TCP, отвечает за правильность преобразования сообщений в пакеты информации, из которых на приемной стороне собирается исходное послание. Протокол нижнего уровня, IP, отвечает за правильность доставки сообщений по указанному адресу. Иногда пакеты одного сообщения могут доставляться разными путями.

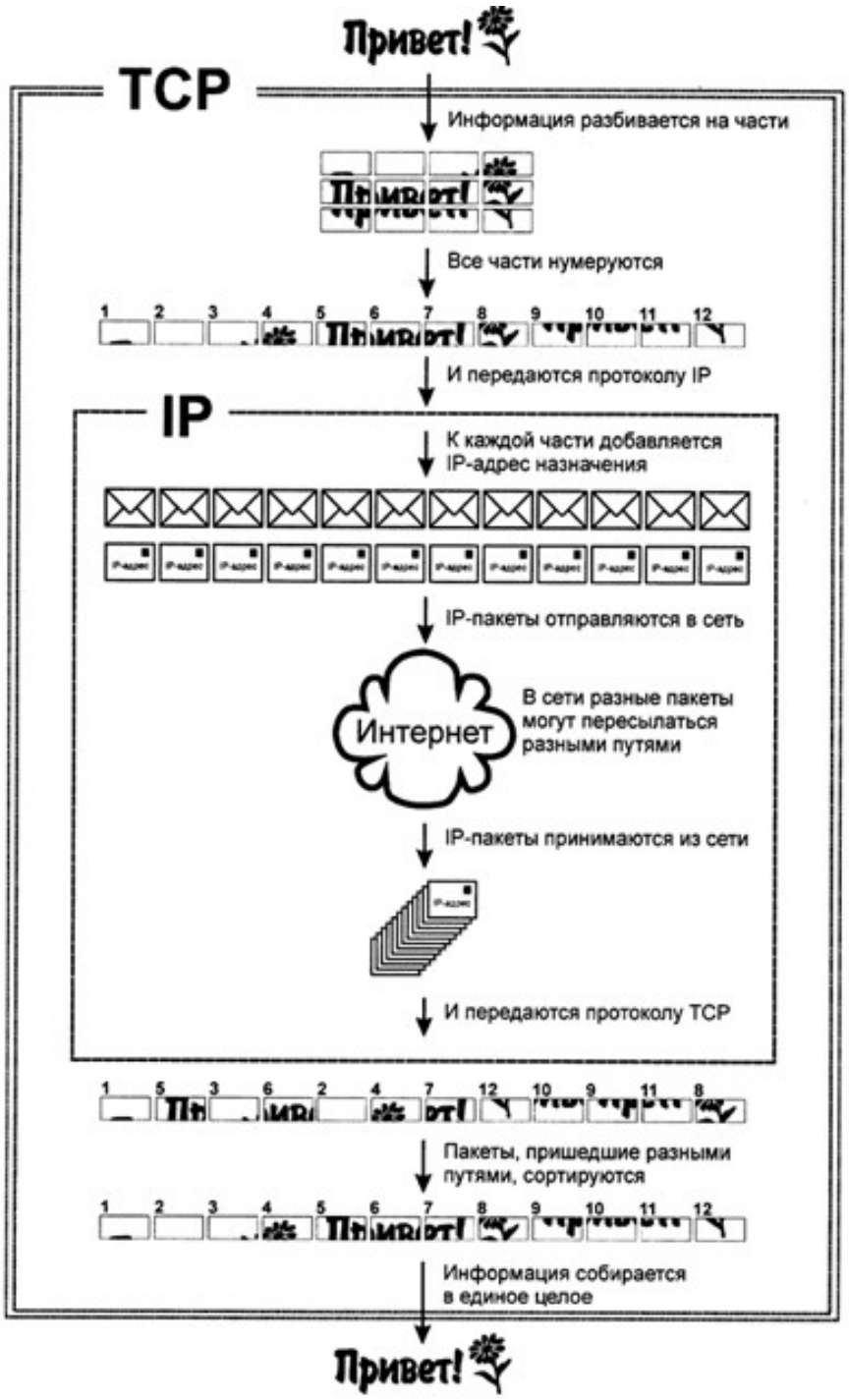

Рисунок 1 – Схема функционирования протокола TCP/IP

Сегодня стек TCP/IP широко используется как в глобальных, так и в локальных сетях. Этот стек имеет иерархическую структуру, в которой определено 4 уровня (таб. 1).

Таблица 1 – Иерархическая структура стека TCP/IP

Прикладной уровень	FTP, Telnet, HTTP, SMTP, SNMP, TFTP
Транспортный уровень	TCP, UDP
Сетевой уровень	IP, ICMP, RIP, OSPF
Уровень сетевых интерфейсов	Не регламентируется

Прикладной уровень стека TCP/IP соответствует трем верхним уровням модели OSI: прикладному, представления и сеансовому. Он объединяет сервисы, предоставляемые системой пользовательским приложениям. За долгие годы применения в сетях различных стран и организаций стек TCP/IP накопил большое количество протоколов и служб прикладного уровня. К ним относятся такие распространенные протоколы, как протокол передачи файлов (File Transfer Protocol, FTP), протокол эмуляции терминала telnet, простой протокол передачи почты (Simple Mail Transfer Protocol, SMTP), протокол передачи гипертекста (Hypertext Transfer Protocol, HTTP) и многие другие. Протоколы прикладного уровня развертываются на хостах.

Транспортный уровень стека TCP/IP может предоставлять вышележащему уровню два типа сервиса:

- гарантированную доставку обеспечивает протокол управления передачей (Transmission Control Protocol, TCP);
- оставку по возможности, или с максимальными усилиями, обеспечивает протокол пользовательских дейтаграмм (User Datagram Protocol, UDP).

Для того чтобы обеспечить надежную доставку данных, протокол TCP предусматривает установление логического соединения, что позволяет ему нумеровать пакеты, подтверждать их прием квитанциями, в случае потери организовывать повторные передачи, распознавать и уничтожать дубликаты, доставлять прикладному уровню пакеты в том порядке, в котором они были отправлены. Благодаря этому протоколу объекты на хосте-отправителе и хосте-получателе могут поддерживать обмен данными в дуплексном режиме. TCP дает возможность без ошибок доставить сформированный на одном из компьютеров поток байтов на любой другой компьютер, входящий в составную сеть [2].

Второй протокол этого уровня, UDP, является простейшим дейтаграммным протоколом, который используется тогда, когда задача надежного обмена данными либо вообще не ставится, либо решается средствами более высокого уровня — прикладным уровнем или пользовательскими приложениями.

В функции протоколов TCP и UDP входит также исполнение роли связующего звена между прилегающими к транспортному уровню прикладным и сетевым уровнями. От прикладного протокола транспортный уровень принимает задание на передачу данных с тем или иным качеством прикладному уровню-получателю. Нижележащий сетевой уровень протоколы TCP и UDP рассматривают как своего рода инструмент, не очень надежный, но способный перемещать пакет в свободном и рискованном путешествии по составной сети [3].

Программные модули, реализующие протоколы TCP и UDP, подобно модулям протоколов прикладного уровня, устанавливаются на хостах.

Сетевой уровень, называемый также уровнем Интернета, является стержнем всей архитектуры TCP/IP. Именно этот уровень, функции которого соответствуют сетевому уровню модели OSI, обеспечивает перемещение пакетов в пределах составной сети, образованной объединением нескольких подсетей. Протоколы сетевого уровня поддерживают интерфейс с вышележащим транспортным уровнем, получая от него запросы на передачу данных по составной сети, а также с нижележащим уровнем сетевых интерфейсов, о функциях которого мы расскажем далее.

Основным протоколом сетевого уровня является межсетевой протокол (Internet Protocol, IP). В его задачу входит продвижение пакета между сетями — от одного маршрутизатора к другому до тех пор, пока пакет не попадет в сеть назначения. В отличие от протоколов прикладного и транспортного уровней, протокол IP развертывается не только на хостах, но и на всех маршрутизаторах (шлюзах). Протокол IP — это дейтаграммный протокол, работающий без установления соединений по принципу доставки с максимальными усилиями. Такой тип сетевого сервиса называют также «ненадежным».

К сетевому уровню TCP/IP часто относят протоколы, выполняющие вспомогательные функции по отношению к IP. Это, прежде всего, протоколы маршрутизации RIP и OSPF, предназначенные для изучения топологии сети, определения маршрутов и составления таблиц маршрутизации, на основании которых протокол IP перемещает пакеты в нужном направлении. По этой же причине к сетевому уровню могут быть отнесены протокол межсетевых управляющих сообщений (Internet Control Message Protocol, ICMP), предназначенный для передачи маршрутизатором источнику сведений об ошибках, возникших при передаче пакета, и некоторые другие протоколы.

Идеологическим отличием архитектуры стека TCP/IP от многоуровневой архитектуры других стеков является интерпретация функций самого нижнего уровня — уровня сетевых интерфейсов.

Напомним, что нижние уровни модели OSI (канальный и физический)

реализуют множество функций доступа к среде передачи, формированию кадров, согласованию величин электрических сигналов, кодированию и синхронизации, а также некоторые другие. Все эти весьма конкретные функции составляют суть таких протоколов обмена данными, как Ethernet, PPP и многих других [4].

У нижнего уровня стека TCP/IP задача существенно проще — он отвечает только за организацию взаимодействия с подсетями разных технологий, входящими в составную сеть. TCP/IP рассматривает любую подсеть, входящую в составную сеть, как средство транспортировки пакетов между двумя соседними маршрутизаторами.

Задачу организации интерфейса между технологией TCP/IP и любой другой технологией промежуточной сети упрощенно можно свести к двум задачам:

- упаковка (инкапсуляция) IP-пакета в единицу передаваемых данных промежуточной сети;
- преобразование сетевых адресов в адреса технологии данной промежуточной сети.

Такой гибкий подход упрощает решение проблемы расширения набора поддерживаемых технологий. При появлении новой популярной технологии она быстро включается в стек TCP/IP путем разработки соответствующего стандарта, определяющего метод инкапсуляции IP-пакетов в ее кадры (например, спецификация RFC 1577, определяющая работу протокола IP через сети ATM, появилась в 1994 году вскоре после принятия основных стандартов ATM). Так как для каждой вновь появляющейся технологии разрабатываются собственные интерфейсные средства, функции этого уровня нельзя определить раз и навсегда, и именно поэтому нижний уровень стека TCP/IP не регламентируется.

Каждый коммуникационный протокол оперирует некоторой единицей передаваемых данных . Названия этих единиц иногда закрепляются стандартом, а чаще просто определяются традицией. В стеке TCP/IP за многие годы его существования образовалась устоявшаяся терминология в этой области (рис. 2).



Рисунок 2 – Названия протокольных единиц данных в TCP/IP

Потоком данных, информационным потоком, или просто потоком, называют данные, поступающие от приложений на вход протоколов транспортного уровня — TCP и UDP.

Протокол TCP «нарезает» из потока данных сегменты.

Единицу данных протокола UDP часто называют дейтаграммой, или датаграммой. Дейтаграмма — это общее название для единиц данных, которыми оперируют протоколы без установления соединений. К таким протоколам относится и протокол IP, поэтому его единицу данных иногда тоже называют дейтаграммой, хотя достаточно часто используется и другой термин — пакет.

В стеке TCP/IP единицы данных любых технологий, в которые упаковываются IP-пакеты для их последующей передачи через сети составной сети, принято называть также кадрами, или фреймам. При этом не имеет значения, какое название используется для этой единицы данных в технологии составляющей сети. Для TCP/IP фреймом является и кадр Ethernet, и ячейка ATM, и пакет X.25 в тех случаях, когда они выступают в качестве контейнера, в котором IP-пакет переносится через составную сеть.

1.3 Протокол DHCP

Для нормальной работы сети каждому сетевому интерфейсу компьютера и маршрутизатора должен быть назначен IP-адрес.

Процедура присвоения адресов происходит в ходе конфигурирования компьютеров и маршрутизаторов. Назначение IP-адресов может происходить вручную в результате выполнения процедуры конфигурирования интерфейса, для компьютера сводящейся, например, к заполнению системы экранных форм. При этом администратор должен помнить, какие адреса из имеющегося множества он уже использовал для других интерфейсов, а какие еще свободны. При конфигурировании помимо IP-адресов сетевых интерфейсов (и соответствующих масок) устройству сообщается ряд других конфигурационных параметров [5]. При конфигурировании администратор должен назначить клиенту не только IP-адрес, но и другие параметры стека TCP/IP, необходимые для его эффективной работы, например маску и IP-адрес маршрутизатора по умолчанию, IP-адрес DNS-сервера, доменное имя компьютера и т. п. Даже при не очень большом размере сети эта работа представляет для администратора утомительную процедуру.

Протокол динамического конфигурирования хостов (Dynamic Host Configuration Protocol, DHCP) автоматизирует процесс конфигурирования сетевых интерфейсов, обеспечивая отсутствие дублирования адресов за счет

централизованного управления их распределением. Работа DHCP описана в RFC 2131 и 2132.

Протокол DHCP работает в соответствии с моделью клиент-сервер. Во время старта системы компьютер, являющийся DHCP-клиентом, посылает в сеть широковещательный запрос на получение IP-адреса.

DHCP-сервер откликается и посылает сообщение-ответ, содержащее IP-адрес и некоторые другие конфигурационные параметры.

При этом сервер DHCP может работать в разных режимах, включая:

- ручное назначение статических адресов;
- автоматическое назначение статических адресов;
- автоматическое распределение динамических адресов.

Во всех режимах работы администратор при конфигурировании DHCP-сервера сообщает ему один или несколько диапазонов IP-адресов, причем все эти адреса относятся к одной сети, то есть имеют одно и то же значение в поле номера сети.

В ручном режиме администратор, помимо пула доступных адресов, снабжает DHCP-сервер информацией о жестком соответствии IP-адресов физическим адресам или другим идентификаторам клиентских узлов. DHCP-сервер, пользуясь этой информацией, всегда выдаст определенному DHCP-клиенту один и тот же назначенный ему администратором IP-адрес (а также набор других конфигурационных параметров).

В режиме автоматического назначения статических адресов DHCP-сервер самостоятельно без вмешательства администратора произвольным образом выбирает клиенту IP-адрес из пула наличных IP-адресов. Адрес дается клиенту из пула в постоянное пользование, то есть между идентифицирующей информацией клиента и его IP-адресом по-прежнему, как и при ручном назначении, существует постоянное соответствие. Оно устанавливается в момент первого назначения DHCP-сервером IP-адреса клиенту. При всех последующих запросах сервер возвращает клиенту тот же самый IP-адрес.

При динамическом распределении адресов DHCP-сервер выдает адрес клиенту на ограниченное время, называемое сроком аренды. Когда компьютер, являющийся DHCP-клиентом, удаляется из подсети, назначенный ему IP-адрес автоматически освобождается. Когда компьютер подключается к другой подсети, то ему автоматически назначается новый адрес. Ни пользователь, ни сетевой администратор не вмешиваются в этот процесс [6].

Это дает возможность впоследствии повторно использовать этот IP-адрес для назначения другому компьютеру. Таким образом, помимо основного преимущества DHCP — автоматизации рутинной работы

администратора по конфигурированию стека TCP/IP на каждом компьютере, режим динамического распределения адресов в принципе позволяет строить IP-сеть, количество узлов в которой превышает количество имеющихся в распоряжении администратора IP-адресов.

1.4 Протокол HTTP

Протокол HTTP (Hypertext Transfer Protocol - Протокол передачи гипертекста) является протоколом более высокого уровня по отношению к протоколу TCP/IP - протоколом уровня приложения. HTTP был разработан для эффективной передачи по Интернету Web-страниц. Именно благодаря HTTP мы имеем возможность созерцать страницы Сети во всем великолепии. Протокол HTTP является основой системы World Wide Web.

Вы отдаете команды HTTP, используя интерфейс браузера, который является HTTP-клиентом. При щелчке мышью на ссылке браузер запрашивает у Web-сервера данные того ресурса, на который указывает ссылка - например, очередной Web-страницы.

Чтобы текст, составляющий содержимое Web-страниц, отображался на них определенным образом - в соответствии с замыслом создателя страницы - он размечается с помощью особых текстовых меток - тегов языка разметки гипертекста (HyperText Markup Language, HTML) [7].

Адреса ресурсов Интернета, к которым вы обращаетесь по протоколу HTTP, выглядит примерно следующим образом: http://www.google.ru

1.5 Протокол FTP

Протокол FTP (File Transfer Protocol) протокол, предназначенный для передачи файлов в компьютерных сетях. FTP позволяет подключаться к серверам FTP, просматривать содержимое каталогов и загружать файлы с сервера или на сервер; кроме того, возможен режим передачи файлов между серверами.

FTP является одним из старейших прикладных протоколов, появившимся задолго до HTTP, в 1971 году. Он и сегодня широко используется для распространения ПО и доступа к удалённым хостам.

Адрес FTP-ресурса в Интернете выглядит следующим образом: ftp://ftp.netscape.com

1.6 Протокол TELNET

С помощью этого протокола вы можете подключиться к удаленному компьютеру как пользователь (если наделены соответствующими правами, то есть знаете имя пользователя и пароль) и производить действия над его файлами и приложениями точно так же, как если бы работали на своем компьютере.

Telnet является протоколом эмуляции терминала. Работа с ним ведется из командной строки. Если вам нужно воспользоваться услугами этого протокола, не стоит рыскать по дебрям Интернета в поисках подходящей программы. Telnet-клиент поставляется, например, в комплекте Windows 98.

Чтобы дать команду клиенту Telnet соединиться с удаленным компьютером, подключитесь к Интернету, выберите в меню Пуск (Start) команду Выполнить (Run) и наберите в строке ввода, например, следующее: telnet google.ru

(Вместо google.ru вы, разумеется, можете ввести другой адрес.) После этого запустится программа Telnet, и начнется сеанс связи [8].

1.7 Протокол WAIS

WAIS расшифровывается как Wide-Area Information Servers. Этот протокол был разработан для поиска информации в базах данных. Информационная система WAIS представляет собой систему распределенных баз данных, где отдельные базы данных хранятся на разных серверах. Сведения об их содержании и расположении хранятся в специальной базе данных - каталоге серверов. Просмотр информационных ресурсов осуществляется с помощью программы - клиента WAIS.

Поиск информации ведется по ключевым словам, которые задает пользователь. Эти слова вводятся для определенной базы данных, и система находит все соответствующие им фрагменты текста на всех серверах, где располагаются данные этой базы. Результат представляется в виде списка ссылок на документы с указанием того, насколько часто встречается в данном документе искомое слово и все искомые слова в совокупности.

Даже в наши дни, когда систему WAIS можно считать морально устаревшей, специалисты во многих областях при проведении научных исследований, тем не менее, обращаются к ней в поисках специфической информации, которую не могут найти традиционными средствами.

Адрес ресурса WAIS в Интернете выглядит выглядит следующим образом: wais://google.ru

1.8 Протокол Gorpher

Протокол Gopher - протокол уровня приложения, разработанный в 1991 году. До повсеместного распространения гипертекстовой системы World Wide Web Gopher использовался для извлечения информации (в основном текстовой) из иерархической файловой структуры. Gopher был провозвестником WWW, позволявшим с помощью меню передвигаться от одной страницы к другой, постепенно сужая круг отображаемой информации. Программы-клиенты Gopher имели текстовый интерфейс. Однако пункты меню Gopher могли указывать и не только на текстовые файлы, но также, например, на telnet-соединения или базы данных WAIS.

Сейчас ресурсы Gopher можно просматривать с помощью обычного Web-браузера, так как современные браузеры поддерживают этот протокол.

Адреса информационных ресурсов Gopher имеют следующий вид: gopher://gopher.tc.umn.edu

1.9 Протокол WAP

WAP (Wireless Application Protocol) был разработан в 1997 году группой компаний Ericsson, Motorola, Nokia и Phone.com (бывшей Unwired Planet) для того, чтобы предоставить доступ к службам Интернета пользователям беспроводных устройств - таких, как мобильные телефоны, пейджеры, электронные органайзеры и др., использующих различные стандарты связи.

К примеру, если ваш мобильный телефон поддерживает протокол WAP, то, набрав на его клавиатуре адрес нужной Web-страницы, вы можете увидеть ее (в упрощенном виде) на дисплее вашего телефона. В настоящее время подавляющее большинство производителей устройств уже перешли к выпуску моделей с поддержкой WAP, который также продолжает совершенствоваться [9].

Помимо этих протоколов, так же существуют и другие, которые представлены в табл. 2.

Таблица 2 – Другие протоколы OSI

Уровень OSI	Протоколы, соответствующие уровню OSI
Прикладной	BGP, DNS, FTP, HTTP, HTTPS, IMAP, LDAP, POP3, SNMP, SMTP, SSH, Telnet, XMPP (Jabber)
Сеансовый/Представления	SSL, TLS
Транспортный	TCP, UDP
Сетевой	EIGRP, ICMP, IGMP, IP, IS-IS, OSPF, RIP
Канальный	Arcnet, ATM, Ethernet, Frame relay, HDLC, PPP, L2TP, SLIP, Token ring

1.10 Протокол BGP

Пограничный (внешний) шлюзовой протокол (Border Gateway Protocol, BGP) версии 4 является сегодня основным протоколом обмена маршрутной информацией между автономными системами Интернета. Протокол BGP пришел на смену протоколу EGP1, использовавшемуся в тот начальный период, когда Интернет имел единственную магистраль. Эта магистраль являлась центральной автономной системой, к которой присоединялись в соответствии с древовидной топологией все остальные автономные системы. Так как между автономными системами при такой структуре петли исключались, протокол EGP не предпринимал никаких мер для того, чтобы исключить зацикливание маршрутов.

Поясним основные принципы работы BGP на примере (рис. 3).

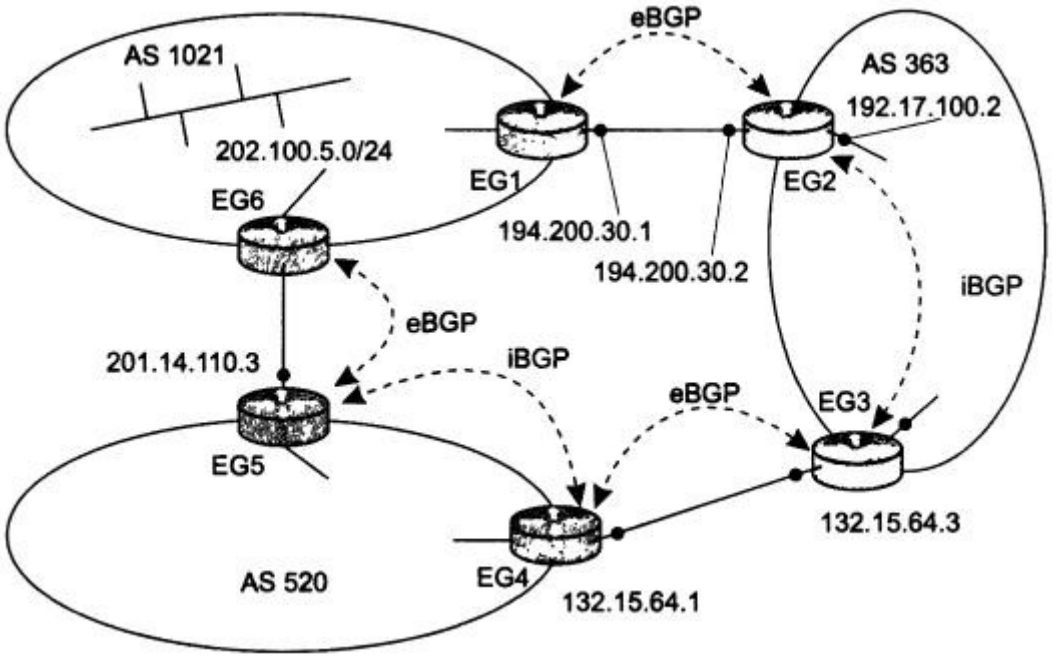

Рисунок 3 – Поиск маршрута между автономными системами

В каждой из трех автономных систем (AS 1021, AS 363 и AS 520) имеется несколько маршрутизаторов, исполняющих роль внешних шлюзов. На каждом из них работает протокол BGP, с помощью которого они общаются между собой.

Маршрутизатор взаимодействует с другими маршрутизаторами по протоколу BGP только в том случае, если администратор явно указывает при конфигурировании, что эти маршрутизаторы являются его соседями. Например, маршрутизатор EG1 в рассматриваемом примере будет взаимодействовать по протоколу BGP с маршрутизатором EG2 не потому, что эти маршрутизаторы соединены двухточечным каналом, а потому, что при конфигурировании маршрутизатора EG1 в качестве соседа ему был указан маршрутизатор EG2 (с адресом 194.200.30.2). Аналогично, при конфигурировании маршрутизатора EG2 его соседом был назначен маршрутизатор EG1 (с адресом 194.200.30.1).

Такой способ взаимодействия удобен в ситуации, когда маршрутизаторы, обменивающиеся маршрутной информацией, принадлежат разным поставщикам услуг (ISP). Администратор ISP может решать, с какими автономными системами он будет обмениваться трафиком, а с какими нет, задавая список соседей для своих внешних шлюзов. Протоколы RIP и QSPF, разработанные для применения внутри автономной системы, обмениваются маршрутной информацией со всеми маршрутизаторами, находящимися в пределах их непосредственной досягаемости (по локальной сети или через двухточечный канал). Это означает, что информация обо всех сетях появляется в таблице маршрутизации каждого маршрутизатора, так что каждая сеть оказывается достижимой для каждой. В корпоративной сети это нормальная ситуация, а в сетях ISP нет, поэтому протокол BGP и исполняет здесь особую роль.

Для установления сеанса с указанными соседями BGP-маршрутизаторы используют протокол TCP (порт 179). При установлении BGP-сеанса могут применяться разнообразные способы аутентификации маршрутизаторов, повышающие безопасность работы автономных систем.

Основным сообщением протокола BGP является сообщение UPDATE (обновить), с помощью которого маршрутизатор сообщает маршрутизатору соседней автономной системы о достижимости сетей, относящихся к его собственной автономной системе. Само название этого сообщения говорит о том, что это — триггерное объявление, которое посылается соседу только тогда, когда в автономной системе что-нибудь резко меняется: появляются новые сети или новые пути к сетям либо же, напротив, исчезают существовавшие сети или пути.

В одном сообщении UPDATE можно объявить об одном новом

маршруте или аннулировать несколько маршрутов, переставших существовать. Под маршрутом в BGP понимается последовательность автономных систем, которую нужно пройти на пути к указанной в адресе сети. Более формально информация о маршруте (BGP Route) к сети (Network/ Maskjength) выглядит так:

BGP Route = AS_Path; NextHop; Network/Maskjength;

Здесь AS_Palh — набор номеров автономных систем, NextHop — IP-адрес маршрутизатора, через который нужно передавать пакеты в сеть Network/Maskjength. Например, если маршрутизатор EG1 хочет объявить маршрутизатору EG2 о том, что в AS 1021 появилась новая сеть 202.100.5.0/24, то он формирует такое сообщение:

AS 1021; 194.200.30.1; 202.100.5.0/24,

после чего передает его маршрутизатору EG2 автономной системы AS 363 (с которым у него, конечно, должен быть установлен BGP-сеанс) [10].

Маршрутизатор EG2, получив сообщение UPDATE, запоминает в своей таблице маршрутизации информацию о сети 202.100.5.0/24 вместе с адресом следующего маршрутизатора 194.200.30.1 и отметкой о том, что эта информация была получена по протоколу BGP. Маршрутизатор EG2 обменивается маршрутной информацией с внутренними шлюзами системы AS 363 по какому-либо протоколу группы IGP, например OSPF. Если у EG2 установлен режим перераспределения маршрутов BGP в маршруты OSPF, то все внутренние шлюзы AS 363 узнают о существовании сети 202.100.5.0/24 с помощью объявления OSPF, которое будет внешним. В качестве адреса следующего маршрутизатора маршрутизатор EG2 начнет теперь объявлять адрес собственного внутреннего интерфейса, например 192.17.100.2.

Однако для распространения сообщения о сети 202.100.5.0/24 в другие автономные системы, например в AS 520, протокол OSPF использоваться не может. Маршрутизатор EG3, связанный с маршрутизатором EG4 автономной системы 520, должен пользоваться протоколом BGP, генерируя сообщение UPDATE нужного формата. Для решения этой задачи он не может задействовать информацию о сети 202.100.5.0/24, полученную от протокола OSPF через один из своих внутренних интерфейсов, так как она имеет другой формат и не содержит, например, сведений о номере автономной системы, в которой находится эта сеть.

Проблема решается за счет того, что маршрутизаторы EG2 и EG3 также устанавливают между собой BGP-сеанс, хотя они и принадлежат одной и той же автономной системе. Такая реализация протокола BGP называется внутренней (Interior BGP, iBGP), в отличие от основной, внешней (Exterior BGP, eBGP). В результате маршрутизатор EG3 получает нужную информацию от маршрутизатора EG2 и передает ее внешнему соседу —

маршрутизатору EG4. При формировании нового сообщения UPDATE маршрутизатор EG3 трансформирует сообщение, полученное от маршрутизатора EG2, добавляя в список автономных систем собственную автономную систему AS 520, а полученный адрес следующего маршрутизатора заменяет адресом собственного интерфейса:

AS 363, AS 1021; 132.15.64.3; 202.100.5.0/24.

Номера автономных систем позволяют исключать зацикливание сообщений UPDATE. Например, когда маршрутизатор EG5 передаст сообщение о сети 202.100.5.0/24 маршрутизатору EG6, то последний не будет его использовать, так как оно будет иметь вид:

AS 520, AS 363, AS 1021; 201.14.110.3; 202.100.5.0/24.

Так как в списке автономных систем уже есть номер собственной автономной системы, очевидно, что сообщение зациклилось.

Протокол BGP используется сегодня не только для обмена маршрутной информацией между автономными системами, но и внутри них.

1.11 Протокол ICMP

Можно представить ряд ситуаций, когда протокол IP не может доставить пакет адресату, например истекает время жизни пакета, в таблице маршрутизации отсутствует маршрут к заданному в пакете адресу назначения, пакет не проходит проверку по контрольной сумме, шлюз не имеет достаточно места в своем буфере для передачи какого-либо пакета и т. д., и т. п.

Как мы не раз отмечали, протокол IP доставляет данные, руководствуясь принципом «по возможности», то есть не предпринимает мер для гарантированной передачи данных адресату. Это свойство «необязательности» протокола IP компенсируется протоколами более высоких уровней стека TCP/IP, например TCP на транспортном уровне и в какой-то степени DNS на прикладном уровне. Они берут на себя обязанности по обеспечению надежности, применяя такие известные приемы, как нумерация сообщений, подтверждение доставки, повторная посылка данных [11].

Протокол ICMP также служит дополнением, компенсирующим ненадежность протокола IP, но несколько другого рода. Он не предназначен для исправления возникших при передаче пакета проблем: если пакет потерян, ICMP не может послать его заново. Задача ICMP другая — он является средством оповещения отправителя о «несчастных случаях», произошедших с его пакетами. Пусть, например, протокол IP, работающий

на каком-либо маршрутизаторе, обнаружил, что пакет для дальнейшей передачи по маршруту необходимо фрагментировать, но в пакете установлен признак DF (не фрагментировать). Протокол IP, обнаруживший, что он не может передать IP-пакет далее по сети, прежде чем отбросить пакет, должен отправить диагностическое ICMP-сообщение конечному узлу-источнику.

Для передачи по сети ICMP-сообщение инкапсулируется в поле данных IP-пакета. IP-адрес узла-источника определяется из заголовка пакета, вызвавшего инцидент.

Сообщение, прибывшее в узел-источник, может быть обработано там либо ядром операционной системы, либо протоколами транспортного и прикладного уровней, либо приложениями, либо просто проигнорированы. Важно, что обработка ICMP-сообщений не входит в обязанности протоколов IP и ICMP.

Заметим, что некоторые из пакетов могут исчезнуть в сети, не вызвав при этом никаких оповещений. В частности, протокол ICMP не предусматривает передачу сообщений о проблемах, возникающих при обработке IP-пакетов, несущих ICMP-сообщения об ошибках. Такое решение было принято разработчиками протокола, чтобы не порождать «штормы» в сетях, когда количество сообщений об ошибках лавинообразно возрастает.

Особенностью протокола ICMP является функциональное разнообразие решаемых задач, а следовательно, и связанных с этим сообщений. Все типы сообщений имеют один и тот же формат (рис. 4), однако интерпретация полей существенно зависит от того, к какому типу относится сообщение.

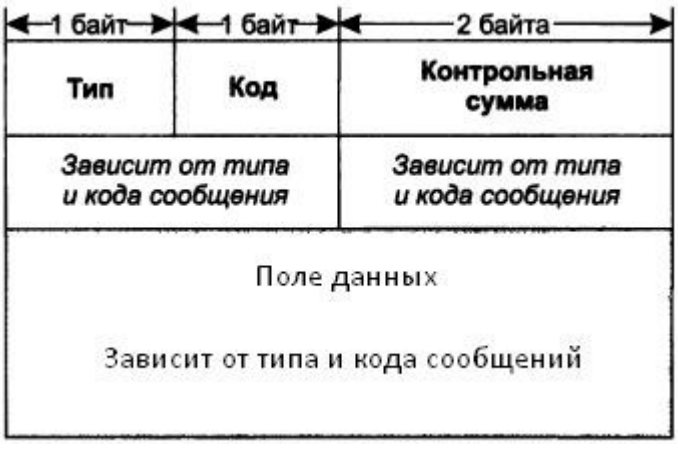

Рисунок 4 – Формат ICMP-сообщения

Заголовок ICMP-сообщения состоит из 8 байт:

- тип (1 байт) — числовой идентификатор типа сообщения;
- код (1 байт) — числовой идентификатор, более тонко

дифференцирующий тип ошибки;

- контрольная сумма (2 байта) — подсчитывается для всего ICMP-сообщения.

Содержимое оставшихся четырех байтов в заголовке и поле данных зависит от значений полей типа и кода.

На рис. 5 показана таблица основных типов ICMP-сообщений. Эти сообщения можно разделить на две группы (помеченные на рисунке условными символами):

- сообщения об ошибках;
- сообщения запрос-ответ.

Сообщения типа запрос-ответ связаны в пары: эхо-запрос — эхо-ответ, запрос маски -ответ маски, запрос времени — ответ времени. Отправитель сообщения-запроса всегда рассчитывает на получение соответствующего сообщения-ответа.



Рисунок 2 – Типы и коды ICMP-сообщений

Сообщения, относящиеся к группе сообщений об ошибках, конкретизируются уточняющим кодом. На рисунке показан фрагмент таблицы кодов для сообщения об ошибке недостижимости узла назначения, имеющей тип 3. Из таблицы мы видим, что это сообщение может быть

вызвано различными причинами, такими как неверный адрес сети или узла (код О или 1), отсутствием на конечном узле-адресате необходимого протокола прикладного уровня (код 2 — «протокол недостижим») или открытого порта UDP/TCP (код 3 — «порт недостижим»). Узел (или сеть) назначения может быть также недостижим по причине временной неработоспособности аппаратуры или из-за того, что маршрутизатор не имеет данных о пути к сети назначения. Всего таблица содержит 15 кодов. Аналогичные таблицы кодов существуют и для других типов сообщений об ошибках.

1.12 Протокол SLIP

Протоколы SLIP (Serial Line Internet Protocol) и PPP (Point to Point Protocol) обеспечивают подключение к коммутируемой линии через каналы передачи данных.

SLIP - это протокол, позволяющий в качестве линий связи использовать последовательные телефонные линии. Программное обеспечение, реализующее работу с протоколом SLIP, принимает символы, приходящие с устройства последовательной передачи данных. Рассматривает их как составляющие IP-пакета, затем упаковывает данные в IP-пакет и передает его модулю TCP. И наоборот, SLIP получает от модуля TCP IP-пакет, выделяет содержимое, форматирует, делит на символы и отправляет его через устройство последовательной передачи в сеть.

1.13 Протокол RIP

Протокол RIP (Routing Information Protocol) представляет собой один из старейших протоколов обмена маршрутной информацией, однако он до сих пор чрезвычайно распространен в вычислительных сетях. Помимо версии RIP для сетей TCP/IP, существует также версия RIP для сетей IPX/SPX компании Novell.

В этом протоколе все сети имеют номера (способ образования номера зависит от используемого в сети протокола сетевого уровня), а все маршрутизаторы - идентификаторы. Протокол RIP широко использует понятие "вектор расстояний". Вектор расстояний представляет собой набор пар чисел, являющихся номерами сетей и расстояниями до них в хопах [12].

Вектора расстояний итерационно распространяются маршрутизаторами по сети, и через несколько шагов каждый маршрутизатор имеет данные о

достижимых для него сетях и о расстояниях до них. Если связь с какой-либо сетью обрывается, то маршрутизатор отмечает этот факт тем, что присваивает элементу вектора, соответствующему расстоянию до этой сети, максимально возможное значение, которое имеет специальный смысл - "связи нет". Таким значением в протоколе RIP является число 16.

При использовании протокола RIP работает эвристический алгоритм динамического программирования Беллмана-Форда, и решение, найденное с его помощью является не оптимальным, а близким к оптимальному. Преимуществом протокола RIP является его вычислительная простота, а недостатками - увеличение трафика при периодической рассылке широковещательных пакетов и неоптимальность найденного маршрута.

1.14 Протокол состояния связей OSPF

Протокол OSPF (Open Shortest Path Firs) является достаточно современной реализацией алгоритма состояния связей (он принят в 1991 году) и обладает многими особенностями, ориентированными на применение в больших гетерогенных сетях.

Протокол OSPF вычисляет маршруты в IP-сетях, сохраняя при этом другие протоколы обмена маршрутной информацией.

Непосредственно связанные (то есть достижимые без использования промежуточных маршрутизаторов) маршрутизаторы называются "соседями". Каждый маршрутизатор хранит информацию о том, в каком состоянии по его мнению находится сосед. Маршрутизатор полагается на соседние маршрутизаторы и передает им пакеты данных только в том случае, если он уверен, что они полностью работоспособны. Для выяснения состояния связей маршрутизаторы-соседи достаточно часто обмениваются короткими сообщениями HELLO.

Для распространения по сети данных о состоянии связей маршрутизаторы обмениваются сообщениями другого типа. Эти сообщения называются router links advertisement - объявление о связях маршрутизатора (точнее, о состоянии связей). OSPF-маршрутизаторы обмениваются не только своими, но и чужими объявлениями о связях, получая в конце-концов информацию о состоянии всех связей сети. Эта информация и образует граф связей сети, который, естественно, один и тот же для всех маршрутизаторов сети.

Кроме информации о соседях, маршрутизатор в своем объявлении перечисляет IP-подсети, с которыми он связан непосредственно, поэтому после получения информации о графе связей сети, вычисление маршрута до

каждой сети производится непосредственно по этому графу по алгоритму Дэйкстры. Более точно, маршрутизатор вычисляет путь не до конкретной сети, а до маршрутизатора, к которому эта сеть подключена. Каждый маршрутизатор имеет уникальный идентификатор, который передается в объявлении о состояниях связей. Такой подход дает возможность не тратить IP-адреса на связи типа "точка-точка" между маршрутизаторами, к которым не подключены рабочие станции.

Маршрутизатор вычисляет оптимальный маршрут до каждой адресуемой сети, но запоминает только первый промежуточный маршрутизатор из каждого маршрута. Таким образом, результатом вычислений оптимальных маршрутов является список строк, в которых указывается номер сети и идентификатор маршрутизатора, которому нужно переслать пакет для этой сети. Указанный список маршрутов и является маршрутной таблицей, но вычислен он на основании полной информации о графе связей сети, а не частичной информации, как в протоколе RIP.

Описанный подход приводит к результату, который не может быть достигнут при использовании протокола RIP или других дистанционно-векторных алгоритмов. RIP предполагает, что все подсети определенной IP-сети имеют один и тот же размер, то есть, что все они могут потенциально иметь одинаковое число IP-узлов, адреса которых не перекрываются. Более того, классическая реализация RIP требует, чтобы выделенные линии "точка-точка" имели IP-адрес, что приводит к дополнительным затратам IP-адресов [13].

В OSPF такие требования отсутствуют: сети могут иметь различное число хостов и могут перекрываться. Под перекрытием понимается наличие нескольких маршрутов к одной и той же сети. В этом случае адрес сети в пришедшем пакете может совпасть с адресом сети, присвоенным нескольким портам.

Если адрес принадлежит нескольким подсетям в базе данных маршрутов, то продвигающий пакет маршрутизатор использует наиболее специфический маршрут, то есть адрес подсети, имеющей более длинную маску.

Например, если рабочая группа ответвляется от главной сети, то она имеет адрес главной сети наряду с более специфическим адресом, определяемым маской подсети. При выборе маршрута к хосту в подсети этой рабочей группы маршрутизатор найдет два пути, один для главной сети и один для рабочей группы. Так как последний более специфичен, то он и будет выбран. Этот механизм является обобщением понятия "маршрут по умолчанию", используемого во многих сетях.

Использование подсетей с различным количеством хостов является вполне естественным. Например, если в здании или кампусе на каждом этаже имеются локальные сети, и на некоторых этажах компьютеров больше, чем на других, то администратор может выбрать размеры подсетей, отражающие ожидаемые требования каждого этажа, а не соответствующие размеру наибольшей подсети.

В протоколе OSPF подсети делятся на три категории:

- "хост-сеть", представляющая собой подсеть из одного адреса;
- "тупиковая сеть", которая представляет собой подсеть, подключенную только к одному маршрутизатору;
- "транзитная сеть", которая представляет собой подсеть, подключенную к более чем одному маршрутизатору.

Транзитная сеть является для протокола OSPF особым случаем. В транзитной сети несколько маршрутизаторов являются взаимно и одновременно достижимыми. В широковещательных локальных сетях, таких как Ethernet или Token Ring, маршрутизатор может послать одно сообщение, которое получат все его соседи. Это уменьшает нагрузку на маршрутизатор, когда он посылает сообщения для определения существования связи или обновленные объявления о соседях. Однако, если каждый маршрутизатор будет перечислять всех своих соседей в своих объявлениях о соседях, то объявления займут много места в памяти маршрутизатора. При определении пути по адресам транзитной подсети может обнаружиться много избыточных маршрутов к различным маршрутизаторам. На вычисление, проверку и отбраковку этих маршрутов уйдет много времени.

Когда маршрутизатор начинает работать в первый раз (то есть инсталлируется), он пытается синхронизировать свою базу данных со всеми маршрутизаторами транзитной локальной сети, которые по определению имеют идентичные базы данных. Для упрощения и оптимизации этого процесса в протоколе OSPF используется понятие "выделенного" маршрутизатора, который выполняет две функции.

Во-первых, выделенный маршрутизатор и его резервный "напарник" являются единственными маршрутизаторами, с которыми новый маршрутизатор будет синхронизировать свою базу. Синхронизировав базу с выделенным маршрутизатором, новый маршрутизатор будет синхронизирован со всеми маршрутизаторами данной локальной сети.

Во-вторых, выделенный маршрутизатор делает объявление о сетевых связях, перечисляя своих соседей по подсети. Другие маршрутизаторы просто объявляют о своей связи с выделенным маршрутизатором. Это делает объявления о связях (которых много) более краткими, размером с объявление о связях отдельной сети.

Для начала работы маршрутизатора OSPF нужен минимум информации - IP-конфигурация (IP-адреса и маски подсетей), некоторая информация по умолчанию (default) и команда на включение. Для многих сетей информация по умолчанию весьма похожа. В то же время протокол OSPF предусматривает высокую степень программируемости.

1.15 Протокол Frame Relay

С возрастанием потребностей в передаче больших объемов информации и ростом скоростей каналов передачи данных, используемых в глобальных сетях, возникла необходимость в новых телекоммуникационных технологиях. Frame Relay был разработан как протокол, обеспечивающий большую производительность и пропускную способность, реализующий преимущества новейших цифровых и оптоволоконных технологий.

Frame Relay первоначально замышлялся как протокол для использования в интерфейсах ISDN, и исходные предложения, представленные в CCITT в 1984 г., преследовали именно эту цель. Работали также над Frame Relay в аккредитованном ANSI комитете по стандартам T1S1 в США. Сеть Frame Relay позволяет передавать по одному физическому каналу разнообразный трафик - данные, голос, видео. Интеграция услуг существенно сокращает затраты клиентов на аренду линии связи.

Frame Relay предоставляет новое поколение услуг пакетных сетей и позволяет осуществлять взаимодействие локальных сетей, выполнять распределенные вычисления, реализовывать распределенную архитектуру клиент-сервер в глобальных сетях.

Структура протокола Frame Relay.

Frame Relay - протокол передачи данных, охватывающий два нижних уровня иерархии модели OSI: канальный и физический. Он прозрачен для вышележащих протоколов. Этот протокол комбинирует свойства технологий разделения времени и статистического мультиплексирования.

В отличие от технологии X.25 протокол Frame Relay использует только часть функций второго уровня, которые включают проверку на наличие ошибок, но исключают требования повторной передачи в случае их обнаружения. Это позволяет существенно сократить время на обработку кадров в узлах сети. В сети Frame Relay работа по восстановлению потерянных данных и коррекцию ошибок возложена на протоколы верхних уровней.

Услуги Frame Relay реализуются в рамках процедур уровня пользователя и процедур уровня управления. Такое разделение дает возможность пользователям выбрать требуемый набор услуг. На уровне пользователя реализуются базовые функции Frame Relay - передача данных по физической линии по логическим соединениям. Уровень управления предназначен для контроля состояния постоянных виртуальных соединений (PVC), а также организации коммутируемых виртуальных соединений (SVC). Определены два типа интерфейсов: UNI (user-to-network) для взаимодействия пользователя с сетью и NNI (network-to-network) для взаимодействия между подсетями Frame Relay.

Важнейшей задачей сети Frame Relay является обеспечение пользователя гарантированным сервисом, если скорость работы ниже уровня CIR. Существенным преимуществом протокола является возможность передачи трафика пользователя, превышающего CIR. Для этого служит параметр Be, задающий дополнительный объем трафика, который может быть передан по данному PVC, например, в том случае, если другие PVC не активны. Т.е. при наличии свободных ресурсов за период Tc по PVC будет передано Bc+Be бит данных. Трафик, превышающий Bc+Be, стирается.

По параметру CIR в сети Frame Relay обычно определяются цены на услуги сети. Возможность передачи по сети Frame Relay трафика, превышающего CIR, выгодна для пользователя.

К дополнительным возможностям Frame Relay относятся следующие. Групповая передача, позволяющая отправить кадр нескольким абонентам. Этот режим работы существенно сокращает трафик по сети за счет исключения дублирующихся кадров в магистральных каналах. Мультипротокольная инкапсуляция, при которой по одному виртуальному каналу можно передавать различные виды трафика (инкапсулировать в кадры Frame Relay протоколы TCP/IP, X.25, SNA, NETBIOS, IPX и т.п.)

ГЛАВА 2
ФОРМАТЫ СТРУКТУРНЫХ ЕДИНИЦ ПРОТОКОЛОВ

2.1 Форматы кадров технологии Ethernet

Существует несколько стандартов формата кадра Ethernet. На практике в оборудовании Ethernet используется только один формат кадра, а именно кадр Ethernet DIX, который иногда называют кадром Ethernet II по номеру последнего стандарта DIX.

Таблица 3 – Формат кадра Ethernet DIX (II)

6 байт	6 байт	2 байта	46-1500 байт	4 байта
DA	SA	T	Данные	FCS

Первые два поля заголовка отведены под адреса:

DA (Destination Address) — MAC-адрес узла назначения;

SA (Source Address) — MAC-адрес узла отправителя. Для доставки кадра достаточно одного адреса — адреса назначения; адрес источника помещается в кадр для того, чтобы узел, получивший кадр, знал, от кого пришел кадр и кому нужно на него ответить. Принятие решения об ответе не входит в компетенцию протокола Ethernet, это дело протоколов верхних уровней. Ethernet же только выполнит такое действие, если с сетевого уровня поступит соответствующее указание.

Поле T (Type, или EtherType) содержит условный код протокола верхнего уровня, данные которого находятся в поле данных кадра, например шестнадцатеричное значение 08-00 соответствует протоколу IP. Это поле требуется для поддержки интерфейсных функций мультиплексирования и демультиплексирования кадров при взаимодействии с протоколами верхних уровней.

Поле данных может содержать от 46 до 1500 байт. Если длина пользовательских данных меньше 46 байт, то это поле дополняется до минимального размера байтами заполнения. Эта операция требуется для корректной работы метода доступа Ethernet (он рассматривается в следующем разделе).

Поле контрольной последовательности кадра (Frame Check Sequence, FCS) состоит из 4 байт контрольной суммы. Это значение вычисляется по алгоритму CRC-32.

Кадр Ethernet DIX (II) не отражает разделения канального уровня Ethernet на уровень MAC и уровень LLC: его поля поддерживают функции

обоих уровней, например интерфейсные функции поля Г относятся к функциям уровня LLC, в то время как все остальные поля поддерживают функции уровня MAC.

Существуют еще три стандартных формата кадра Ethernet:

- кадр 802.3/LLC является стандартом комитета IEEE 802 и построен в соответствии с принятым разбиением функций канального уровня на уровень MAC и уровень LLC. Поэтому результирующий кадр является вложением кадра LLC, определяемого стандартом 802.2, в кадр MAC, определяемого стандартом 802.3;
- кадр Raw 802.3, или Novell 802.3, появился в результате усилий компании Novell по ускорению разработки своего стека протоколов в сетях Ethernet;
- кадр Ethernet SNAP стал результатом деятельности комитета 802.2 по приведениюпредыдущих форматов кадров к некоторому общему стандарту и приданию кадру необходимой гибкости для учета в будущем возможностей добавления полей или изменения их назначения.

2.2 Формат IP-пакета

Имеется прямая связь между количеством полей заголовка пакета и функциональной сложностью протокола, который работает с этим заголовком. Чем проще заголовок — тем проще соответствующий протокол. Большая часть действий протокола связана с обработкой той служебной информации, которая переносится в полях заголовка пакета. Изучая назначение каждого поля заголовка IP-пакета, мы получаем не только формальные знания о структуре пакета, но и знакомимся с основными функциями протокола IP.

Таблица 4 – Структура заголовка IP-пакета

<table>
<tr><td colspan="2">4 бита
Номер версии</td><td colspan="2">4 бита
Номер версии</td><td colspan="5">8 бит
Тип сервиса</td><td colspan="2">16 бит
Общая длина</td></tr>
<tr><td colspan="2" rowspan="2">16 бит
Индетификатор пакета</td><td colspan="5">3 бита
Флаги</td><td colspan="2" rowspan="2">13 бит
Смещение фрагмента</td></tr>
<tr><td></td><td colspan="2">D</td><td colspan="2">M</td></tr>
<tr><td colspan="2">8 бит
Время жизни</td><td colspan="2">8 бит
Протокол верхнего уровня</td><td colspan="5">16 бит
Контрольная сумма</td></tr>
<tr><td colspan="9">32 бита
IP-адрес источника</td></tr>
<tr><td colspan="9">32 бита
IP-адрес назначения</td></tr>
<tr><td colspan="9">Параметры и выравнивание</td></tr>
</table>

Поле номера версии занимает 4 бита и идентифицирует версию протокола IP. Сейчас повсеместно используется версия 4 (IPv4), хотя все чаще встречается и новая версия (IPv6).

Значение длины заголовка IP-пакета также занимает 4 бита и измеряется в 32-битных словах. Обычно заголовок имеет длину в 20 байт (пять 32-битных слов), но при добавлении некоторой служебной информации это значение может быть увеличено за счет дополнительных байтов в поле параметров. Наибольшая длина заголовка составляет 60 байт.

Поле типа сервиса (Type of Service, ToS) имеет и другое, более современное название — байт дифференцированного обслуживания, или DS-байт. Этим двум названиям соответствуют два варианта интерпретации этого поля. В обоих случаях данное поле служит одной цели — хранению признаков, которые отражают требования к качеству обслуживания пакета. В прежнем варианте первые три бита содержат значение приоритета пакета: от самого низкого — 0 до самого высокого — 7. Маршрутизаторы и компьютеры могут принимать во внимание приоритет пакета и обрабатывать более важные пакеты в первую очередь. Следующие три бита поля ToS определяют критерий выбора маршрута. Если бит D (Delay - задержка) установлен в 1, то маршрут должен выбираться для минимизации задержки доставки данного пакета, установленный бит T (Throughput — пропускная

способность) - для максимизации пропускной способности, а бит R (Reliability — надежность) — для максимизации надежности доставки. Оставшиеся два бита имеют нулевое значение.

Стандарты дифференцированного обслуживания, принятые в конце 90-х годов, дали новое название этому полю и переопределили назначение его битов. В DS-байте также используются только старшие 6 бит, а два младших бита остаются в качестве резерва.

Поле общей длины занимает 2 байта и характеризует общую длину пакета с учетом заголовка и поля данных. Максимальная длина пакета ограничена разрядностью поля, определяющего эту величину, и составляет 65 535 байт, однако в большинстве компьютеров и сетей столь большие пакеты не используются. При передаче по сетям различного типа длина пакета выбирается с учетом максимальной длины пакета протокола нижнего уровня, несущего IP-пакеты. Если это кадры Ethernet, то выбираются пакеты с максимальной длиной 1500 байт, умещающиеся в поле данных кадра Ethernet. В стандартах TCP/IP предусматривается, что все хосты должны быть готовы принимать пакеты длиной вплоть до 576 байт (независимо от того, приходят ли они целиком или фрагментами). Идентификатор пакета занимает 2 байта и используется для распознавания пакетов, образовавшихся путем делен'ия на части (фрагментации) исходного пакета. Все части (фрагменты) одного пакета должны иметь одинаковое значение этого поля.

Флаги занимают 3 бита и содержат признаки, связанные с фрагментацией. Установленный в 1 бит DF (Do not Fragment — не фрагментировать) запрещает маршрутизатору фрагментировать данный пакет, а установленный в 1 бит MF (More Fragments — больше фрагментов) говорит о том, что данный пакет является промежуточным (не последним) фрагментом. Оставшийся бит зарезервирован.

Поле смещения фрагмента занимает 13 бит и задает смещение в байтах поля данных этого фрагмента относительно начала поля данных исходного (нефрагментированного) пакета. Используется при сборке/разборке фрагментов пакетов. Смещение должно быть кратно 8 байт.

Поле времени жизни (Time To Live, TTL) занимает один байт и используется для задания предельного срока, в течение которого пакет может перемещаться по сети. Время жизни пакета измеряется в секундах и задается источником. По истечении каждой секунды пребывания на каждом из маршрутизаторов, через которые проходит пакет во время своего «путешествия» по сети, из его текущего времени жизни вычитается единица; единица вычитается и в том случае, если время пребывания было меньше секунды. Поскольку современные маршрутизаторы редко обрабатывают

пакет дольше, чем за одну секунду, то время жизни можно интерпретировать как максимальное число транзитных узлов, которые разрешено пройти пакету. Если значение поля времени жизни становится нулевым до того, как пакет достигает получателя, пакет уничтожается. Таким образом, время жизни является своего рода часовым механизмом самоуничтожения пакета.

Поле протокола верхнего уровня занимает один байт и содержит идентификатор, указывающий, какому протоколу верхнего уровня принадлежит информация, размещенная в поле данных пакета [14]. Значения идентификаторов для разных протоколов приводятся в документе RFC 1700.

Контрольная сумма заголовка занимает 2 байта (16 бит) и рассчитывается только по заголовку. Поскольку некоторые поля заголовка меняют свое значение в процессе передачи пакета по сети (например, поле времени жизни), контрольная сумма проверяется и повторно рассчитывается на каждом маршрутизаторе и конечном узле как дополнение к сумме всех 16-битных слов заголовка. При вычислении контрольной суммы значение самого поля контрольной суммы устанавливается в нуль. Если контрольная сумма неверна, то пакет отбрасывается, как только обнаруживается ошибка.

Поля IP-адресов источника и приемника имеют одинаковую длину — 32 бита.

Поле параметров является необязательным и используется обычно только при отладке сети. Это поле состоит из нескольких подполей одного из восьми предопределенных типов. В этих подполях можно указывать точный маршрут, регистрировать проходимые пакетом маршрутизаторы, помещать данные системы безопасности или временные отметки.
Так как число подполей в поле параметров может быть произвольным, то в конце заголовка должно быть добавлено несколько нулевых байтов для выравнивания заголовка пакета по 32-битной границе.
Далее приведена распечатка значений полей заголовка одного из реальных IP-пакетов, захваченных в сети Ethernet средствами анализатора протоколов сетевого монитора (Network Monitor, NM) компании Microsoft. В данной распечатке NM в скобках дает шестнадцате-ричные значения полей, кроме того, программа иногда представляет числовые коды полей в виде, более удобном для чтения. Например, дружественный программный интерфейс NM интерпретирует код 6 в поле протокола, помещая туда название соответствующего протокола — TCP.

IP: Version = 4 (0x4)
IP: Header Length = 20 (0x14)
IP: Service Type = 0 (0x0)
IP: Precedence = Routine
IP: .. .0____ = Normal Delay

IP: ____0... = Normal Throughput
IP: .....0.. = Normal Reliability
IP: Total Length = 54 (0x36)
IP: Identification = 31746 (0x7C02)
IP: Flags Summary = 2 (0x2)
IP: ....0 = Last fragment in datagram
IP: ......1. = Cannot fragment datagram
IP: Fragment Offset = 0 (0x0) bytes
IP: Time to Live = 128 (0x80)
IP: Protocol - TCP - Transmission Control
IP: Checksum = 0xEB86
IP: Source Address = 194.85.135.75
IP: Destination Address = 194.85.135.66
IP: Data: Number of data bytes remaining = 34 (0x0022)

В заголовке IP-пакета для хранения IP-адресов отправителя и получателя отводятся два поля, каждое имеет фиксированную длину 4 байта (32 бита). IP-адрес состоит из двух логических частей — номера сети и номера узла в сети.

Наиболее распространенной формой представления IP-адреса является запись в виде четырех чисел, представляющих значения каждого байта в десятичной форме и разделенных точками, например:

128.10.2.30.

Этот же адрес может быть представлен в двоичном формате:

10000000 00001010 00000010 00011110.

А также в шестнадцатеричном формате:

80.0A.02.1D.

Заметим, что запись адреса не предусматривает специального разграничительного знака между номером сети и номером узла. Вместе с тем при передаче пакета по сети часто возникает необходимость разделить адрес на эти две части. Например, маршрутизация, как правило, осуществляется на основании номера сети, поэтому каждый маршрутизатор, получая пакет, должен прочитать из соответствующего поля заголовка адрес назначения и выделить из него номер сети. Каким образом маршрутизаторы определяют, какая часть из 32 бит, отведенных под IP-адрес, относится к номеру сети, а какая — к номеру узла?

Можно предложить несколько вариантов решения этой проблемы:

- простейший из них состоит в использовании фиксированной границы. При этом все 32-битное поле адреса заранее делится на две части не

обязательно равной, но фиксированной длины, в одной из которых всегда будет размещаться номер сети, в другой — номер узла. Решение очень простое, но хорошее ли? Поскольку поле, которое отводится для хранения номера узла, имеет фиксированную длину, все сети будут иметь одинаковое максимальное число узлов. Если, например, под номер сети отвести один первый байт, то все адресное пространство распадется на сравнительно небольшое (28);

- число сетей огромного размера (224 узлов). Если границу передвинуть дальше вправо, то сетей станет больше, но все равно все они будут одинакового размера. Очевидно, что такой жесткий подход не позволяет дифференцированно удовлетворять потребности отдельных предприятий и организаций. Именно поэтому он не нашел применения, хотя и использовался на начальном этапе существования технологии TCP/IP (RFC 760);
- второй подход (RFC 950, RFC 1518) основан на использовании маски, которая позволяет максимально гибко устанавливать границу между номером сети и номером узла. При таком подходе адресное пространство можно использовать для создания множества сетей разного размера. Маска — это число, применяемое в паре с IP-адресом, причем двоичная запись маски содержит непрерывную последовательность единиц в тех разрядах, которые должны в IP-адресе интерпретироваться как номер сети. Граница между последовательностями единиц и нулей в маске соответствует границе между номером сети и номером узла в IP-адресе;
- последний способ, основанный на классах адресов (RFC 791). Этот способ представляет собой компромисс по отношению к двум предыдущим: размеры сетей хотя и не могут быть произвольными, как при использовании масок, но и не должны быть одинаковыми, как при установлении фиксированных границ. Вводится пять классов адресов: A, B, C, D, E. Три из них — A, B и C — предназначены для адресации сетей, а два — D и E — имеют специальное назначение. Для каждого класса сетевых адресов определено собственное положение границы между номером сети и номером узла.

2.3 Формат PPP

Протокол PPP (Point-to-Point Protocol — протокол двухточечной связи) является стандартным протоколом Интернета. Протокол PPP, так же как и HDLC, представляет собой целое семейство протоколов, в которое, в частности, входят:

- протокол управления линией связи (Link Control Protocol, LCP);
- протокол управления сетью (Network Control Protocol, NCP);
- многоканальный протокол PPP (Multi-Link PPP, MLPPP);
- протокол аутентификации по паролю (Password Authentication Protocol, PAP);
- протокол аутентификации по квитированию вызова (Challenge Handshake Authentication Protocol, CHAP).

При разработке протокола PPP за основу был взят формат кадров HDLC и дополнен несколькими полями. Эти дополнительные поля протокола PPP вложены в поле данных кадра HD0LC. Позже были разработаны стандарты, описывающие вложение кадра PPP в кадры Frame Relay и других протоколов глобальных сетей. Хотя протокол PPP и работает с кадром HDLC, он не поддерживает, подобно стандартной версии протокола HDLC, процедуры надежной передачи кадров и управления их потоком.

Особенностью протокола PPR отличающей его от других протоколов канального уровня, является сложная переговорная процедура принятия параметров соединения. Стороны обмениваются различными параметрами, такими как качество линии, размер кадров, тип протокола аутентификации и тип инкапсулируемых протоколов сетевого уровня.

В корпоративной сети конечные системы часто отличаются размерами буферов для временного хранения пакетов, ограничениями на размер пакета, списком поддерживаемых протоколов сетевого уровня. Физическая линия, связывающая конечные устройства, может варьироваться от низкоскоростной аналоговой до высокоскоростной цифровой линии с различными уровнями качества обслуживания.

Протокол, в соответствии с которым принимаются параметры соединения, называется протоколом управления линией связи (LCP). Чтобы справиться со всеми возможными ситуациями, в протоколе PPP имеется набор стандартных параметров, действующих по умолчанию и учитывающих все стандартные конфигурации. При установлении соединения два взаимодействующих устройства для нахождения взаимопонимания пытаются сначала использовать эти параметры. Каждый конечный узел описывает свои возможности и требования. Затем на основании этой информации принимаются параметры соединения, устраивающие обе стороны. Переговорная процедура протоколов может и не завершиться соглашением о каком-нибудь параметре. Если, например, один узел предлагает в качестве MTU значение 1000 байт, а другой отвергает это предложение и в свою очередь предлагает значение 1500 байт, которое отвергается первым узлом, то по истечении тайм-аута переговорная процедура может закончиться безрезультатно.

Одним из важных параметров соединения PPP является режим аутентификации. Для целей аутентификации PPP предлагает по умолчанию протокол аутентификации по паролю (PAP), передающий пароль по линии связи в открытом виде, или протокол аутентификации по квитированию вызова (CHAP), не передающий пароль по линии связи и поэтому обеспечивающий более высокий уровень безопасности сети. Пользователям также разрешается добавлять новые алгоритмы аутентификации. Кроме того, пользователи могут влиять на выбор алгоритмов сжатия заголовка и данных.

Многопротокольная поддержка — способность протокола PPP поддерживать несколько протоколов сетевого уровня — обусловила распространение PPP как стандарта де-факто. Внутри одного соединения PPP могут передаваться потоки данных различных сетевых протоколов, включая IP, Novell IPX и многих других, сегодня уже не употребляющихся, а также данные протоколов канального уровня локальной сети.

Каждый протокол сетевого уровня конфигурируется отдельно с помощью соответствующего протокола управления сетью (NCP). Под конфигурированием понимается, во-первых, констатация того факта, что данный протокол будет использоваться в текущем сеансе PPP, а во-вторых, переговорное согласование некоторых параметров протокола. Больше всего параметров устанавливается для протокола IP, включая IP-адреса взаимодействующих узлов, IP-адреса DNS-серверов, признак компрессии заголовка IP-пакета и т. д. Для каждого протокола конфигурирования протокола верхнего уровня, помимо общего названия NCP, употребляется особое название, построенное путем добавления аббревиатуры CP (Control Protocol — протокол управления) к имени конфигурируемого протокола, например для IP — это протокол IPCP, для IPX — IPXCP и т. п.

Расширяемость протокола. Под этим свойством PPP понимается как возможность включения новых протоколов в стек PPP, так и возможность применения собственных протоколов пользователей вместо рекомендуемых в PPP по умолчанию. Это позволяет наилучшим образом настроить PPP для каждой конкретной ситуации.

2.4 Формат HTTP-сообщений

В протоколе HTTP все сообщения состоят из текстовых строк. Сообщения как запросов, так и ответов имеют единую обобщенную структуру из трех частей: обязательной стартовой строки, а также необязательных заголовков и тела сообщения.

Как видно из таблицы, запросы и ответы имеют разные форматы стартовой строки. Каждая из них состоит из трех элементов, включающих поле версии протокола HTTP. И в запросе, и в ответе примера указана версия HTTP 1.1. Стартовая строка запроса включает в себя поле метода — это название операции, которая должна быть выполнена. Чаще всего в запросах используется метод GET, то есть запрос объекта. Именно он включен в наш пример запроса. Помимо этого метода в запросах протокол предусматривает и другие методы, Такие как POST, который используется клиентом, например, для отправки электронной почты или в поисковых машинах, когда клиент запрашивает у сервера не определенный объект, а объекты, содержащие ключевые слова, помещенные в теле сообщения [15].

В стартовой строке ответа, помимо уже упоминавшегося указания на версию протокола HTTP, имеется поле кода состояния и поле фразы для короткого текстового сообщения, поясняющего данный код пользователю.

В настоящее время стандарты определяют пять классов кодов состояния:

- 1xx — информация о процессе передачи;
- 2xx — информация об успешном принятии и обработки запроса клиента;
- 3xx — информация о том, что для успешного выполнения операции нужно произвести следующий запрос по-другому URL-адресу, указанному в дополнительном заголовке Locati on;
- 4xx — информация об ошибках на стороне клиента (читатель наверняка не раз сталкивался с ситуацией, когда при указании адреса несуществующей страницы браузер выводил на экран сообщение 404 Not Found);
- 5xx — информация о неуспешном выполнения операции по вине сервера (например, сообщение 505 http Version Not Supported говорит о том, что сервер не поддерживает версию HTTP, предложенную клиентом).

Среди кодов состояния имеется код 401, сопровождаемый сообщением authorization required. Если клиент получает такое сообщение в ответ на попытку доступа к странице или объекту, это означает, что доступ к данному ресурсу ограничен и требует авторизации 1 пользователя. Помимо поясняющей фразы сервер помещает в свой ответ дополнительный заголовок www-Authenticate:<...>, который сообщает клиенту, какую информацию он должен направить серверу для того, чтобы процедура авторизации могла быть выполнена. Обычно это имя и пароль. Веб-клиент с момента получения такого ответа сервера начинает добавлять во все свои запросы к ресурсам данного сервера дополнительный заголовок Authorization: <имя, пароль>, который содержит информацию, необходимую для авторизации доступа.

2.5 Формат UDP

При работе на хосте-отправителе данные от приложений поступают протоколу UDP через порт в виде сообщений (рис. 6). Протокол UDP добавляет к каждому отдельному сообщению свой 8-байтный заголовок, формируя из этих сообщений собственные протокольные единицы, называемые UDP-дейтаграммами, и передает их нижележащему протоколу IP. В этом и заключаются его функции по мультиплексированию данных.

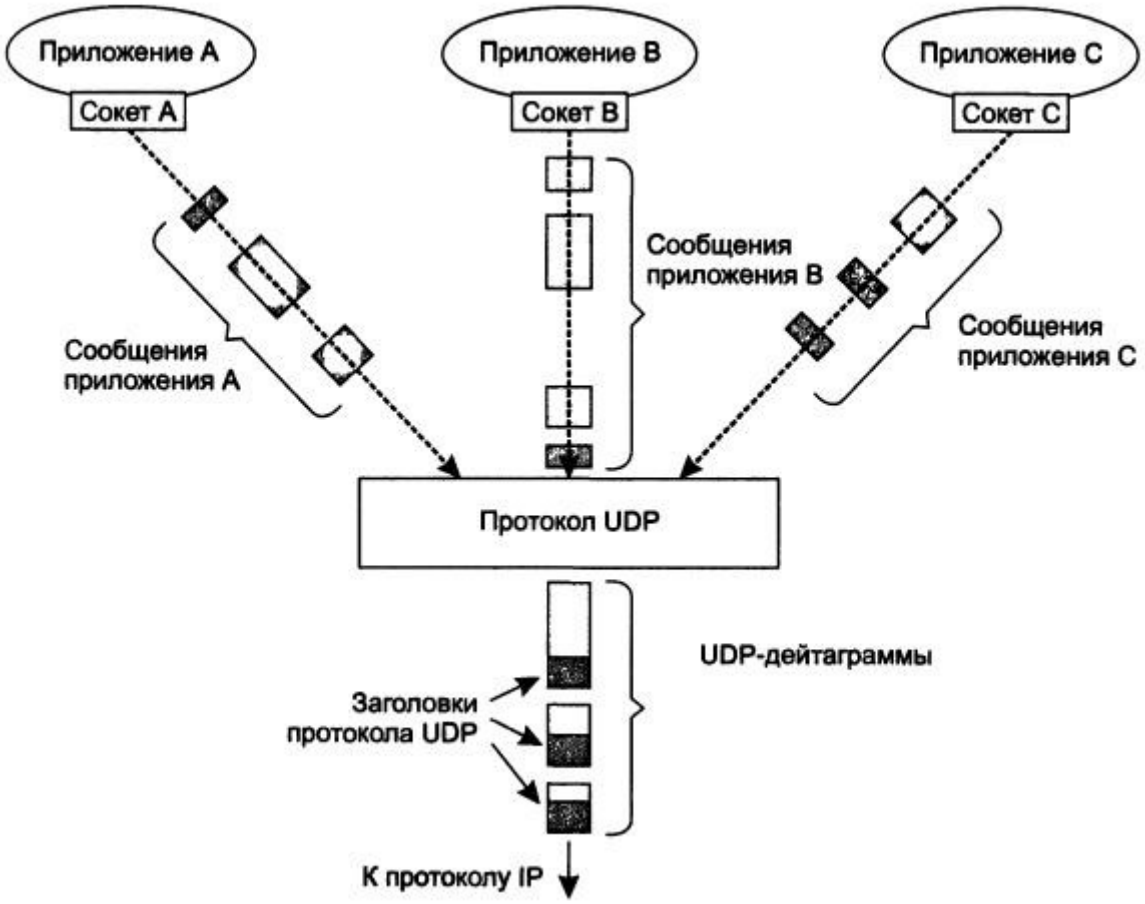

Рисунок 6 – Схема функционирования протокола UPD

Каждая дейтаграмма переносит отдельное пользовательское сообщение. Сообщения могут иметь различную длину, не превышающую однако длину поля данных протокола IP, которое, в свою очередь, ограничено размером кадра технологии нижнего уровня. Поэтому если буфер UDP переполняется, то сообщение приложения отбрасывается.

Заголовок UDP состоит из четырех 2-байтных полей:

- номер UDP-порта отправителя;
- номер UDP-порта получателя;
- контрольная сумма;

- длина дейтаграммы.

Далее приведен пример заголовка UDP с заполненными полями:

Source Port - 0x0035

Destination Port - 0x0411

Total length - 132 (0x84) bytes

Checksum = 0x5333

В этой UDP-дейтаграмме в поле данных, длина которого, как следует из заголовка, равна (132 - 8) байт, помещено сообщение DNS-сервера, что можно видеть по номеру порта источника (Source Port = 0—0035). В шестнадцатеричном формате это значение равно стандартному номеру порта DNS-сервера — 53.

Судя по простоте заголовка, протокол UDP не сложен. Действительно, его функции сводятся к простой передаче данных между прикладным и сетевым уровнями, а также примитивному контролю искажений в передаваемых данных. При контроле искажений протокол UDP только диагностирует, но не исправляет ошибку. Если контрольная сумма показывает, что в поле данных UDP-дейтаграммы произошла ошибка, протокол UDP просто отбрасывает поврежденную дейтаграмму.

Работая на хосте-получателе, протокол UDP принимает от протокола IP извлеченные из пакетов UDP-дейтаграммы. Полученные из IP-заголовка IP-адрес назначения и из UDP-заголовка номер порта используются для формирования UDP-сокета, однозначно идентифицирующего приложение, которому направлены данные. Протокол UDP освобождает дейтаграмму от UDP-заголовка. Полученное в результате сообщение он передает приложению на соответствующий UDP-сокет. Таким образом, протокол UDP выполняет демультиплексирование на основе сокетов.

2.6 Формат стека TCP/IP

Стек TCP/IP был разработан до появления модели взаимодействия открытых систем ISO/OSI, вследствие чего, хотя он также имеет многоуровневую структуру, соответствие уровней стека TCP/IP уровням модели OSI достаточно условно.

Протоколы TCP/IP делятся на 4 уровня.

Самый нижний (уровень IV) соответствует физическому и канальному уровням модели OSI. Этот уровень в протоколах TCP/IP не регламентируется, но поддерживает все популярные стандарты физического и канального уровня: для локальных сетей это Ethernet, Token Ring, FDDI, Fast 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. Обычно при появлении новой технологии локальных или глобальных сетей она быстро включается в стек TCP/IP за счет разработки соответствующего RFC, определяющего метод инкапсуляции пакетов IP в ее кадры.

Следующий уровень (уровень III) - это уровень межсетевого взаимодействия, который занимается передачей пакетов с использованием различных транспортных технологий локальных сетей, территориальных сетей, линий специальной связи и т.п.

В качестве основного протокола сетевого уровня (в терминах модели OSI) в стеке используется протокол IP, который изначально проектировался как протокол передачи пакетов в составных сетях, состоящих из большого количества локальных сетей, объединенных как локальными, так и глобальными связями. Поэтому протокол IP хорошо работает в сетях со сложной топологией, рационально используя наличие в них подсистем и экономно расходуя пропускную способность низкоскоростных линий связи. Протокол IP является дейтаграммным протоколом, то есть он не гарантирует доставку пакетов до узла назначения, но старается это сделать.

К уровню межсетевого взаимодействия относятся и все протоколы, связанные с составлением и модификацией таблиц маршрутизации, такие как протоколы сбора маршрутной информации RIP (Routing Internet Protocol) и OSPF (Open Shortest Path First), а также протокол межсетевых управляющих сообщений ICMP (Internet Control Message Protocol). Последний протокол предназначен для обмена информацией об ошибках между маршрутизаторами сети и узлом - источником пакета. С помощью специальных пакетов ICMP сообщается о невозможности доставки пакета, о превышении времени жизни или продолжительности сборки пакета из фрагментов, об аномальных величинах параметров, об изменении маршрута пересылки и типа обслуживания, о состоянии системы и т.п.

Следующий уровень (уровень II) называется основным. На этом уровне функционируют протокол управления передачей TCP (Transmission Control Protocol) и протокол дейтаграмм пользователя UDP (User Datagram Protocol). Протокол TCP обеспечивает надежную передачу сообщений между удаленными прикладными процессами за счет образования виртуальных соединений. Протокол UDP обеспечивает передачу прикладных пакетов дейтаграммным способом, как и IP, и выполняет только функции связующего звена между сетевым протоколом и многочисленными прикладными процессами.

Верхний уровень (уровень I) называется прикладным. За долгие годы использования в сетях различных стран и организаций стек TCP/IP накопил большое количество протоколов и сервисов прикладного уровня. К ним относятся такие широко используемые протоколы, как протокол копирования файлов FTP, протокол эмуляции терминала telnet, почтовый протокол SMTP, используемый в электронной почте сети Internet, гипертекстовые сервисы доступа к удаленной информации, такие как WWW и многие другие.

Нас интересуют четвертый и третий уровни данного стека, некоторые протоколы которых рассмотрим подробнее.

TCP/IP поддерживает все популярные стандарты физического и канального уровней: для локальных сетей - это Ethernet, Token Ring, FDDI, для глобальных - протоколы работы на аналоговых и выделенных линиях SLIP, PPP, RIP, EGP, BGP, IGP, OSPF протоколы территориальных сетей X.25 и ISDN.

ГЛАВА 3
РАЗРАБОТКА ПРОГРАМНОГО ОБЕСПЕЧЕНИЯ

3.1 Получение списка устройств

Как правило, первое, что делает приложение на основе SharpPcap, это получает список подключенных сетевых адаптеров. В библиотеке SharpPcap это обеспечивает класс CaptureDeviceList. Класс является одноэлементным экземпляром, который содержит кэшированный список сетевых адаптеров типа ICaptureDevice. В частности, свойства Name и Description содержат имя и описание, соответствующего устройства [16]. В данном коде C # показано, как получить список адаптеров и распечатать его на экране, напечатав ошибку, если адаптеры не найдены:

```
//Выводим версию SharpPcap
string ver = SharpPcap.Version.VersionString;
Console.WriteLine("SharpPcap {0}, Example1.IfList.cs", ver);
//Получаем список устройств
CaptureDeviceList devices = CaptureDeviceList.Instance;
//Если нет устройств выводим ошибку
if (devices.Count < 1)
{
Console.WriteLine("Нет сетевых устройств на компьютере");
return;
}
Console.WriteLine("\nДоступны следующие сетевые устройства:");
Console.WriteLine("-----------------------------------------------------\n");
//Печатаем все доступные устройства
foreach (ICaptureDevice dev in devices)
Console.WriteLine("{0}\n", dev.ToString());
Console.Write("Нажмите Enter для выхода...");
Console.ReadLine();
```

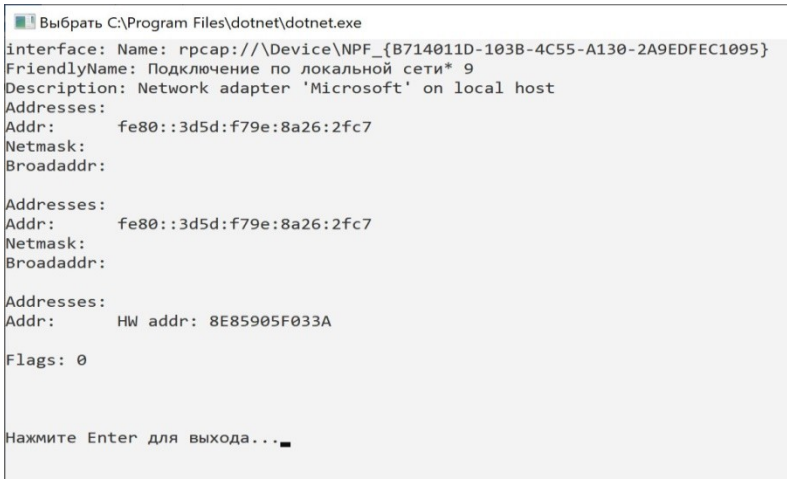

Рисунок 7 – Доступные сетевые устройства

3.2 Открытие адаптера и получение пакетов

Функция открывающая сетевой адаптер Open(). Она перегружена и имеет следующие аргументы:

- open();
- open(DeviceMode mode);
- open(DeviceMode mode, int read_timeout).

Последние два аргумента заслуживают детального рассмотрения.

DeviceMode в обычном режиме (**DeviceMode.Normal**), сетевой адаптер захватывает только те пакеты, которые адресованы непосредственно ему, обмен пакетами других хостов в сети игнорируется. Когда адаптер находится в беспорядочном режиме (**DeviceMode.Promiscuous**), он захватывает все пакеты, хотят они этого или нет. Это означает, что Libpcap / WinPcap сможет захватить пакеты адресованные другим хостам. Promiscuous(неразборчивый) режим используется по умолчанию для большинства приложений, так что мы также включим его в следующем примере. ПРИМЕЧАНИЕ: Использование этого режима в сети можно обнаружить, так что если захватывать в Promiscuous (беспорядочном) режиме, то действия могут быть обнаружены. **read_timeout**: Задает таймаут в миллисекундах. Чтения с адаптера (например, с помощью функции GetNextPacket () всегда будет возвращаться после **read_timeout** миллисекунд, даже если пакеты уже доступны. **read_timeout** также определяет интервал между статистическими отчетами, если адаптер в работает статистическом режиме (в режиме сбора статистики).

В следующем коде показано использование OnPacketArrival для приема пакетов. Мы создаем обработчик событий, который вызывается всякий раз, когда новый пакет проходит через ICaptureDevice:

CaptureDeviceList devices = CaptureDeviceList.Instance;

```
if (devices.Count < 1)
{
Console.WriteLine("Нет сетевых устройств на компьютере");
return;
}
foreach (ICaptureDevice dev in devices)
Console.WriteLine("{0}\n", dev.ToString());
int i = 0;
ICaptureDevice device = devices[i];
//Регистрируем событие для приёмки пакетов
device.OnPacketArrival +=
new SharpPcap.PacketArrivalEventHandler(device_OnPacketArrival);

//Открываем адаптер для получения пакетов
int readTimeoutMilliseconds = 1000;
device.Open(DeviceMode.Promiscuous, readTimeoutMilliseconds);
Console.WriteLine("-- Listening on {0}, hit 'Enter' to stop...",device.Description);

//Начинаем сбор пакетов
device.StartCapture();

//Ждем нажатия Enter
Console.ReadLine();

//Прекращаем процесс сбора пакетов
device.StopCapture();

//Закрываем pcap устройство
device.Close();
```

После этого, адаптер открыт, захват может быть запущен с помощью **StartCapture()** или **Capture(INT packetCount)** функции. Эти две функции схожи, разница в том, что **StartCapture()** не блокирует основной поток программы, а запускается в новом потоке, в то время **Capture(INT packetCount)** блокирует основной поток, до тех пор пока packetCount пакеты не будут получены. При использовании **StartCapture()** мы должны вызвать **StopCapture()**, чтобы завершить процесс захвата. Для бесконечного(постоянного) захвата необходимо вызывать метод **Capture()**.

Обе эти функции требуют, чтобы обработчик события для обработки пакетов был зарегистрирован(инициализирован) до их вызова. Этот

обработчик событий вызывается устройством **ICaptureDevice** для каждого нового пакета, проходящего через сеть. Он получает отправителя объекта, который вызывает этот обработчик (т.е. **ICaptureDevice** объекта) и фактически полученного пакета, включая все заголовки протокола. Обратите внимание, что CRC обычно не присутствует в пакете, поскольку он удаляется сетевым адаптером после процедуры проверки пакета(целостности). Стоит также отметить, что большинство адаптеров отбрасывает пакеты с неправильной контрольной суммой CRC, в следствии чего так WinPcap (и, следовательно SharpPcap), как правило, не в состоянии захватить их.

3.3 Захват пакетов без обработчика событий

ICaptureDevice.GetNextPacket() вместо метода регистрации обработчика события. Использование события **OnPacketArrival** является хорошей практикой и может быть хорошим выбором, в некоторых ситуациях, например, при захвате пакетов одновременно с нескольких устройств. Однако, использование обработки обратного вызова иногда не практично — это часто делает программу более сложной, особенно в ситуациях с многопоточными приложениями. В этих случаях **GetNextPacket()** возвращает пакет с прямым вызовом — с помощью GetNextPacket(), пакеты принимаются только тогда, когда мы хотим их получить.

```
CaptureDeviceList devices = CaptureDeviceList.Instance;
if (devices.Count < 1)
{
Console.WriteLine("Нет сетевых устройств на компьютере");
return;
}
foreach (ICaptureDevice dev in devices)
Console.WriteLine("{0}\n", dev.ToString());
int i = 0;
ICaptureDevice device = devices[i];
//Открываем устройство для захвата
int readTimeoutMilliseconds = 1000;
device.Open(DeviceMode.Promiscuous, readTimeoutMilliseconds);
Console.WriteLine();
Console.WriteLine("-- Listening on {0}...",
device.Description);
Packet packet = null;
```

```
RawCapture raw = device.GetNextPacket();
while (raw != null)
{
packet = Packet.ParsePacket(raw.LinkLayerType, raw.Data);
//Выводим время и длину полученного пакета
DateTime time = raw.Timeval.Date;
int len = raw.Data.Length;
Console.WriteLine("{0}:{1}:{2},{3} Len={4}",
time.Hour, time.Minute, time.Second,
time.Millisecond, len);
}

//Закрываем pcap устройство
device.Close();
Console.WriteLine(" -- Capture stopped, device closed.");

Console.ReadLine();
```

3.4 Запись пакетов в файл

Полезно, в случаях, когда необходимо записать перехваченные пакеты в файл для автономной их обработки, учета и другого. В последних версиях SharpPcap получил новый класс для записи захваченных пакетов в файлы, CaptureFileWriterDevice.

```
// Открываем файл для записи
captureFileWriter = new CaptureFileWriterDevice(device, capFile);
```

Теперь, когда captureFileWriter существует, при помощи класса OnPacketArrival, можно записывать захваченный пакеты на диск(в файл):

```
// Записываем пакет в файл
captureFileWriter.Write(e.Packet);
Console.WriteLine("Packet dumped to file.");
```

3.5 Считывание информации о пакетах из файла

Необходимо создать экземпляр **CaptureFileReaderDevice(string captureFilename)** и считывать пакеты так же, как с любого другого устройства.

```
// Подписываемся на событие
```

```
device.OnPacketArrival +=
    new PacketArrivalEventHandler( device_OnPacketArrival );
Console.WriteLine();
Console.WriteLine
("-- Capturing from '{0}', hit 'Ctrl-C' to exit...",      capFile);

// Запускаем бесконечный сбор пакетов
// Этот метод вернет EOF при конце файла
device.Capture();
```

ЗАКЛЮЧЕНИЕ

Таким образом, в ходе выполнения курсовой работы получены следующие результаты:

•рассмотрены основные протоколы сети интернет;

•изучены форматы структурных единиц протоколов;

•изучены возможности библиотеки SharpPcap.

•разработано приложение для захвата сетевого трафика.

Выявлены следующие возможности библиотеки SharpPcap:

•получение списка устройств;

•открытие адаптера и получение пакетов;

•захват пакетов без обработчика событий;

•запись пакетов в файл;

•считывание информации о пакетах из файлов.

В дальнейшем планируется реализация данных возможностей в приложении для анализа компьютерного трафика.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Бертсекас, Д. Сети передачи данных / Д. Бертсекас, Р. Галлагер. – М., Мир, 1989. – 542с.
2. Блэк, Ю. Сети ЭВМ: протоколы, стандарты, интерфейсы / Ю. Блэк. – М., Мир, 1990. – 452 с.
3. Бройдо, В. Л. Вычислительные системы, сети и телекоммуникации / В. Л. Бройдо. – Спб., Питер, 2002. – 688 с.
4. Гук, М. Аппаратные средства локальных сетей / М. Гук. – СПБ., Питер, 2000. – 576 с.
5. Гук, М. Аппаратные средства PC. Энциклопедия / – М., Мир, 2003. – 366 с.
6. Челлис, Дж Основы построения сетей / Дж. Челлис, Ч. Перкинс, М. Стриб. – М., Лори, 1997. – 205 с.
7. Дженнигс, Ф. Практическая передача данных: Модемы, сети и протоколы. / Ф. Дженнигс. – М., Мир, 1989. – 305 с.
8. Золотов, С. Протоколы Internet / С. Золотов. – Киев, BHV, 1998. – 340 с.
9. Хант, К Персональные компьютеры в сетях TCP/IP / К. Хант. – Киев, BHV, 1997. – 210 с.
10. Марк, А. Высокопроизводительные сети. Энциклопедия пользователя / А. Марк. – Киев, ДиаСофт, 1998. – 154 с.
11. Олифер, В. Г. Компьютерные сети / В. Г. Олифер, Н. А. Олифер. – СПб., Питер, 2000. – 672с.
12. Олифер, В. Г. Сетевые операционные системы / В. Г. Олифер, Н.А. Олифер – СПб., Питер, 2001. – 110 с.
13. Пасько, В. П. Энциклопедия ПК. Аппаратура. Программы. Интернет / В. П. Пасько. – Киев, BHV; 2004. – 800с.
14. Попов, И.И., Компьютерные сети: Учебное пособие / И. И. Попов, Н. В. Максимов. – М., ФОРУМ: ИНФРА-М, 2004. – 623 с.

15. Стек протоколов TCP/IP [Электронный ресурс] / Компьютерные сети. – 2017. – Режим доступа: http://iptcp.net/stek-protokolov-tcpip.html. – Дата доступа: 25. 12. 2018.

16. SharpPcap – A Packet Capture Fraemework for .NET [Electronic resource] / Code Project. – 2014. – Mode of access: https://www.codeproject.com/Articles/12458/SharpPcap-A-Packet-Capture-Framework-for-NET#dumpTCP. – Date of access: 25. 12. 2018.