ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

**Федеральное государственное
образовательное бюджетное учреждение
высшего профессионального образования
«САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ
им. проф. М. А. БОНЧ-БРУЕВИЧА»**

**Б. С. Гольдштейн, В. Ю. Гойхман
Ю. В. Столповская**

СЕТЕВЫЕ АНАЛИЗАТОРЫ IP СЕТЕЙ

УЧЕБНОЕ ПОСОБИЕ

СПбГУТ)))

**САНКТ-ПЕТЕРБУРГ
2013**

УДК 621.395(075.8)
ББК 32.882я73
Г63

Рецензенты
доктор технических наук, профессор, заведующий кафедрой АЭС ПГУТИ
А. В. Росляков

Утверждено
редакционно-издательским советом университета
в качестве учебного пособия

Гольдштейн, Б. С.

Г63 Сетевые анализаторы IP сетей : учебное пособие / Б. С. Гольдштейн, В. Ю. Гойхман, Ю. В. Столповская ; СПбГУТ. – СПб., 2013. – 56 с.

Содержит учебный материал по принципам тестирования телекоммуникационных протоколов и устройствам, предназначенным для их тестирования. Процесс обучения строится на основе исследовательского полигона технологий и протоколов СОТСБИ-У. Материал учебного пособия представлен в виде теоретической и практической частей, содержит рекомендации для проведения практических и лабораторных занятий, а также необходимую литературу.

Предназначены для студентов, обучающихся по направлению 210700 «Инфокоммуникационные технологии и системы связи».

УДК 621.395(075.8)
ББК 32.882я73

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

DNS	Domain Name System. Система доменных имён
ETSI	European Telecommunications Standards Institute. Европейский институт телекоммуникационных стандартов
FTP	File Transfer Protocol. Протокол передачи файлов
HTTP	HyperText Transfer Protocol. Протокол, обеспечивающий передачу данных через Интернет
HTTPS	Hypertext Transfer Protocol Secure. Расширение протокола HTTP, поддерживающее шифрование
ICMP	Internet Control Message Protocol. Протокол межсетевых управляющих сообщений
IETF	Internet Engineering Task Force. Комитет инженерных задач интернет
IP	Internet Protocol. Протокол сетевого уровня сети Интернет
ISO	International Organization for Standardization. Международная организация по стандартизации
NFS	Network File System. Протокол сетевого доступа к файловым системам
NGN	Next Generation Network. Сеть следующего поколения
NLM	Network Lock Manager. Менеджер блокировок сети
NP	Network Performance. Производительность сети
NSM	Network Status Monitor. Монитор состояния сети
NTP	Network Time Protocol. Сетевой протокол задания времени
QoS	Quality of Service. Качество обслуживания
RPC	Remote Procedure Call. Протокол удаленного вызова процедур
SIP	Session Initiation Protocol. Протокол инициирования сеансов связи
SSH	Secure Shell. «Безопасная оболочка» – это сетевой протокол, обеспечивающий защищенную аутентификацию, соединение и безопасную передачу данных в сети Интернет
TFTP	Trivial File Transfer Protocol. Простой протокол передачи файлов
TCP	Transmission Control Protocol. Протокол управления передачей
UDP	User Datagram Protocol. Протокол передачи дейтаграмм пользователя
XDR	External Data Representation. Международный стандарт передачи данных в Интернете
МСЭ	Международный союз электросвязи
СОТСБИ	Сертифицированное оборудование телекоммуникационных сетей – банк информации
СПС	Сеть подвижной связи

ТфОП Телефонная сеть общего пользования

1. ТЕСТИРОВАНИЕ ТЕЛЕКОММУНИКАЦИОННЫХ ПРОТОКОЛОВ

1.1. Стеки телекоммуникационных протоколов

Протоколы являются языком, на котором коммутационные узлы, станции и другие телекоммуникационные устройства общаются в сети. В более формальной трактовке протоколом является согласованная система правил и процедур, которая дает описание принципа взаимодействия множественных объектов.

Для определения протоколов Международной организацией стандартизации ISO разработана семиуровневая эталонная модель взаимодействия открытых систем (Open System Interconnection). В данной модели более низкий уровень всегда предоставляет услуги более высокому. Взаимодействие между разными уровнями одной системы осуществляется по средствам примитивов, а взаимодействие между одноименными уровнями разных систем по средствам протоколов (рис. 1.1). Совокупность этих протокольных уровней называется стеком протоколов.

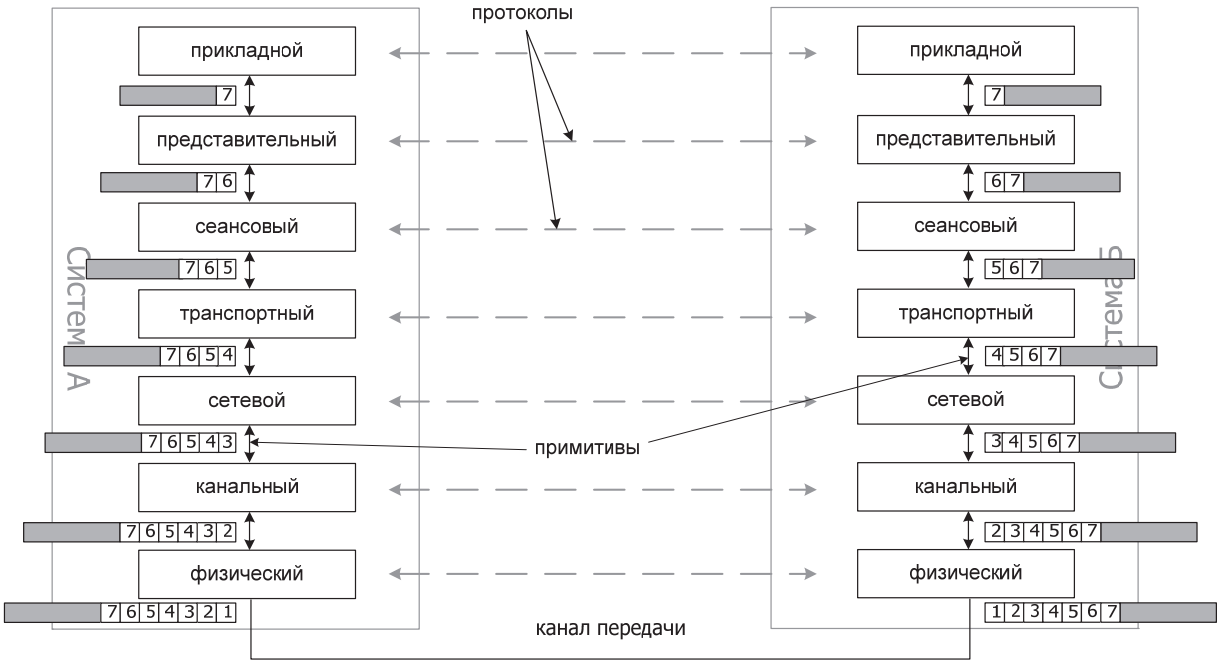

Рис. 1.1. Модель взаимодействия открытых систем. Протоколы и примитивы

1.2. Тестирование телекоммуникационных протоколов

Тестирование качества работы и совместимости компонентов современной телекоммуникационной сети приобретает все более важное значение в последние годы. При этом особое внимание уделяется проверке корректности реализации протоколов сигнализации.

Значение термина «тестирование» не определено никакими стандартами, поэтому многие используют для одних и тех же понятий разные термины. Кроме того, тестирование является больше искусством, чем наукой.

Это является следствием того:

- что лишь некоторые термины в этой области стандартизированы в мировом масштабе или имеют одинаковое толкование среди специалистов (например, при измерении параметров производительности можно встретить использование терминов «тестирование производительности» или «тестирование нагрузочной способности»);
- отдачу от применения тестового оборудования трудно рассчитать (например, дать ответ на вопрос о том, как много ошибок было выявлено с помощью тестового оборудования);
- практически невозможно оценить необходимый объем тестирования (например, выявить точную формулу зависимости обнаружения ошибок от проводимых тестов).

Тестирование в течение жизненного цикла сети связи

В общем виде тестирование и измерение в сетях связи являются необходимыми элементами эксплуатации сетей и сетевых элементов. Измерение определяет параметры эксплуатационной характеристики и осуществляется либо по запросу, либо непрерывно через специальное мониторинговое оборудование. Тестирование добавляет к этому процессу измерений еще и сравнение измеряемых параметров с нормативными значениями с последующим приемом/отказом (accept/reject thresholds). Тестирование и измерение сети сигнализации в течение различных фаз жизненного цикла сети выполняется системой мониторинга сети сигнализации. С помощью сетевого мониторинга сигнализации, в частности, определяется, работают ли сеть и сетевые элементы и услуги, как предусматривалось.

Жизненный цикл протокола сетевой сигнализации включает следующие основные фазы:

- фазу тестирования, в течение которой новый сетевой протокол внедряется и отлаживается, тестируется на соответствие требованиям национальных и международных стандартов;
- фазу внедрения, в течение которой протокол тиражируется по сети, тестируется совместимость сетевых элементов, проверяется устойчивость функционирования;
- эксплуатационную фазу, в течение которой протокол сигнализации находится в условиях эксплуатации и требует поддержания работоспособности и необходимого качества обслуживания QoS, для чего также предусматриваются тестирование и мониторинг.

Тестирование протоколов сигнализации на этих фазах жизненного цикла осуществляется с помощью протокол-тестеров.

1.3. Типы тестирования

Для тестирования протоколов сигнализации для обеспечения взаимодействия сетевых элементов в телекоммуникационной сети существует несколько методологий:

- проверка соответствия;
- проверка производительности;
- проверка возможности совместной работы;
- проверка взаимодействия;
- регрессивное тестирование;
- приемосдаточные испытания;
- мониторинг.

Каждый из этих механизмов гарантирует соответствие протокола требованиям той или иной стадии жизненного цикла.

Тестирование соответствия

Первым этапом в тестировании протокола является обеспечение того, чтобы он работал в соответствии со спецификацией, на основе которой он был создан. Этот процесс называется проверкой согласованности.

На практике, роль проверки согласованности заключается в том, чтобы увеличить уверенность в том, что протокол соответствует своей спецификации, и уменьшить риск неправильного срабатывания. Проверка согласованности представляет собой хорошо отлаженную методологию тестирования, основанную на международном стандарте ISO 9646. Главная идея стандарта ISO 9646 состоит в том, что спецификация нового протокола должна содержать комплект тестовых сценариев его проверки. Тестирование на соответствие заданной спецификации является единственным стандартизированным и широко распространенным методом проверки корректности реализации протокола. Этот метод основан на применении специализированного языка написания тестов TTCN.

Тесты соответствия включают в себя проверку корректности работы протокольных объектов, т. е. соблюдения очередности следования сообщений, правильности перехода объектов из одного состояния в другое под воздействием определенных внешних событий, кодировки обязательных информационных элементов.

Тестирование производительности

Одно лишь тестирование на соответствие не может гарантировать корректность реализации протокола полностью, так как не предполагает проведение тестирования под нагрузкой и проверку поведения системы при неопределенных значениях параметров спецификации, оставленных для возможного применения в будущем.

При тестировании производительности измеряются те параметры, которые зависят от поступающей на систему нагрузки, и производится их сравнение с допустимыми значениями (например, измерение интенсивности потерь вызовов). Этот вид тестирования сводится к измерению параметров качества обслуживания (QoS) или производительности сети (NP, Network Performance) при известных значениях параметров поступающей нагрузки.

Тестирование производительности производится с использованием эталонной системы (что не всегда возможно и дорого) или с использованием системы тестирования, имитирующей эталонную систему. Для имитации эталонной системы, с которой должно стыковаться тестируемое оборудование, используются симуляторы протоколов и генераторы вызовов. При использовании реальной системы в качестве эталонной применяются анализаторы протоколов, осуществляющие мониторинг интерфейса, который соединяет тестируемую систему с эталонной.

Для измерения значений параметров QoS и NP используются генераторы вызовов (сигнального трафика), создающие нагрузку определенного вида на тестируемую систему посредством генерации последовательностей сообщений определенного протокола и измеряющие значения интенсивности потерь вызовов, интенсивности появления ошибок протокола, интервалы времени между передачей и приемом сообщений (таймеры) и т. п.

Тестирование совместимости

Тестирование возможности совместной работы является следующим логическим этапом после выполнения проверок согласованности и производительности.

Спецификация протокола нередко содержит области неоднозначного понимания, подверженные различной интерпретации разработчиками и, следовательно, различной реализации. Такими областями спецификации являются опциональные процедуры и параметры, разные значения параметров и величины таймеров. Неоднозначность спецификации приводит к тому, что реализации протоколов от разных производителей оборудования не работают совместно, даже если каждая реализация успешно прошла предварительное тестирование на соответствие.

Тестирование совместного функционирования является ключевым аспектом для сетевых операторов, эксплуатирующих оборудование разных производителей. Очевидно, что сетевые элементы одного производителя должны корректно работать с сетевыми элементами другого производителя. Проверка этой возможности может проводиться в лабораторных условиях или непосредственно в сети оператора.

На этапе тестирования совместного функционирования проверяется, в какой степени и при каких условиях разные реализации одного и того же протокола могут совместно работать, производя ожидаемый результат. Те-

сты этого вида могут применяться как ко всем протоколам стека, используемого на интерфейсе, так и к какому-либо одному выбранному протоколу.

Тестирование взаимодействия

Тестирование взаимодействия разных протоколов и систем сигнализации приобретает важное значение для современных телекоммуникационных сетей. Тестирование взаимодействия охватывает весь процесс обслуживания вызова и предоставления дополнительных услуг. Иными словами тестирование взаимодействия является итоговым тестированием, обеспечивающим проверку функционирования системы в целом.

Цель тестов взаимодействия показать, что функциональность из конца в конец между двумя связанными системами отвечает требованием стандартов, на которых эти системы основаны.

Как показывает практика, проведение тестов взаимодействия существенно увеличивает вероятность того, что в одной сети будет успешно взаимодействовать оборудование, выпущенное разными производителями.

Регрессионное тестирование

Редко бывает, чтобы только одна версия программного обеспечения узла коммутации работала в течение его жизненного цикла. Последующие программные версии, как правило, включают зафиксированные и исправленные ошибки программирования, усовершенствования, новые функции, дополнительные услуги и т. д.

Регрессивное тестирование является методикой, обеспечивающей по мере развития и замены версий программного обеспечения надлежащее мигрирование ранее оттестированных протокольных реализации IUT.

Приемосдаточные испытания

Приемосдаточные испытания объединяют серию стандартных тестов, определяемых программой и методиками приемосдаточных испытаний. Обычно эти тесты выполняются на сети Оператора после того, как телекоммуникационное оборудование установлено, смонтировано и запущено. Приемосдаточные испытания могут включать регрессивное тестирование каждый раз, когда устанавливается новая версия программного обеспечения с новыми функциями или градациями емкостей.

Мониторинг

Мониторинг телекоммуникационных протоколов является не только последней фазой тестирования протоколов, но и самой длительной и, пожалуй, самой важной.

Мониторинг интерфейса между находящимися в эксплуатации сетевыми элементами обеспечивает:

• выявление ошибок при взаимодействии протоколов, не обнаруженных на других этапах тестирования;

• обнаружение несанкционированного доступа к ресурсам со стороны отдельных абонентов;

• сбор информации о вызовах (CDR) и транзакциях (TDR);

• трассировку вызовов;

• обнаружение зацикливания сообщений;

• контроль источников и маршрутов прохождения трафика.

Системы мониторинга и анализа сигнализации декодируют принимаемые от многочисленных каналов сети сигнализации сообщения и сигналы, проверяют их на предмет соответствия заданной спецификации, выделяют (как правило, красным цветом) сообщения или их отдельные параметры, не соответствующие спецификации, точно таким же образом они отображают перегрузки, аварийные ситуации и многое другое.

1.4. Анализаторы протоколов

На каждом из этапов, через которые проходит любой компонент сети связи, начиная от разработки и кончая эксплуатацией в условиях реальной нагрузки, используются специальные методы тестирования и анализа качества функционирования, которые реализуются специализированными программно-аппаратными средствами, представляющими собой отдельные приборы (протокол-тестеры или анализаторы) и целые системы распределенного наблюдения.

Эволюционные изменения сетей связи нашли отражение в развитии анализаторов протоколов. От простых устройств мониторинга протоколов передачи данных на скоростях 1200–2400 бит/с они эволюционировали до сложнейших интеллектуальных приборов с разнообразными функциональными возможностями и гибкой настройкой.

В процессе конвергенции сетей связи к анализаторам протоколов предъявляются все новые требования, а область их применения постоянно расширяется. Этому соответствует целый ряд предпосылок:

• концепция NGN породила целый ряд новых типов телекоммуникационных узлов и систем, взаимодействующих друг с другом в сети и, следовательно, требующих тщательного анализа их совместной работы, наблюдения, проверки и тестирования;

• концепция NGN не отдает предпочтение какой-то одной сетевой структуре с использованием единственного протокольного стека для всего разнообразия инфокоммуникационных услуг, в связи с чем всегда существует необходимость обеспечения взаимодействия программно-аппаратных компонентов различных производителей, поддерживающих протоколы и интерфейсы разных стандартов;

• концепция NGN подразумевает обеспечение параметров качества обслуживания QoS и показателей надежности, как минимум, на уровне существующих сетей фиксированной (ТфОП) и мобильной (СПС) телефонной связи, для чего также необходимы средства тестирования и контроля выполнения требований по корректному и бесперебойному функционированию NGN.

На сегодняшний день на рынке телекоммуникационного оборудования представлен довольно большой выбор анализаторов протоколов, применяемых для тестирования VoIP протоколов. Они отличаются друг от друга по конструктивному исполнению, реализованным функциям и возможностям, набору поддерживаемых протоколов.

Существуют следующие функции анализаторов протоколов:

• функции мониторинга;

• функции декодирования и анализа сообщений тестируемого протокола, выловленных из сети;

• функции генерации большого потока вызовов;

• функции симуляции оборудования, поддерживающего тестируемый протокол;

• возможность создания тестовых сценариев;

• возможность создания ошибочных ситуаций.

Анализатор протокола может обладать одной или несколькими из вышеперечисленных функций.

2. АНАЛИЗАТОР СЕТЕВЫХ ПРОТОКОЛОВ WIRESHARK

2.1. Назначение Wireshark

Wireshark – это программа-анализатор сетевых протоколов, имеющая пользовательский графический интерфейс, задача которой состоит в мониторинге сетевого трафика в реальном времени, детальном отображении принятых и отправленных пакетов данных, а также сохранении собранных данных для последующего анализа. В Wireshark реализована мощная система поиска и фильтрации пакетов по множеству критериев.

Первые разработки программы-анализатора сетевых протоколов появились в 1998 г. Изначально проект имел название Ethereal, но в 2006 г. он был переименован в Wireshark.

Программа распространяется под свободной лицензией GNU GPL. Существуют версии для большинства типов операционных систем UNIX, в том числе GNU/Linux, Solaris, FreeBSD, NetBSD, OpenBSD, Mac OS X, а также для Windows.

Wireshark осуществляет мониторинг огромного числа сетевых протоколов: IP, TCP, UDP, FTP, TFTP, DNS, HTTP, HTTPS, ICMP, SSH, NTP, NFS, SIP, RTP, RTCP, MEGACO (H.248), MGCP, стека протоколов H.323, SIGTRAN и т. д. и поддерживает такие технологии физического и канального уровней, как Ethernet, Token Ring и FDDI, ATM.

Wireshark используется:

- для выявления и решения проблем в сети;
- для отладки сетевых протоколов;
- для изучения сетевых протоколов.

2.2. Запуск Wireshark

Для запуска WireShark необходимо дважды нажать левую клавишу мыши на соответствующем ярлыке, расположенном на Рабочем столе ПК рабочего места исследовательского полигона «СОТСБИ-У».

На полигоне «СОТСБИ-У» предусмотрен запуск Wireshark в двух режимах:

- запуск программы непосредственно на рабочем месте (ярлык Wireshark). В данном режиме Wireshark захватывает все IP-пакеты, принимаемые и отправляемые сетевым интерфейсом ПК рабочего места;
- удаленный запуск программы на сервере (ярлык Wireshark на сервере). В данном режиме Wireshark захватывает все IP-пакеты, принимаемые и отправляемые сетевыми интерфейсами виртуальных серверов полигона.

2.3. Пользовательский интерфейс Wireshark

После запуска программы появляется Главное окно.

Процесс мониторинга пакетов начнется после того, как с помощью меню Capture будет выбран сетевой интерфейс и нажата кнопка Start.

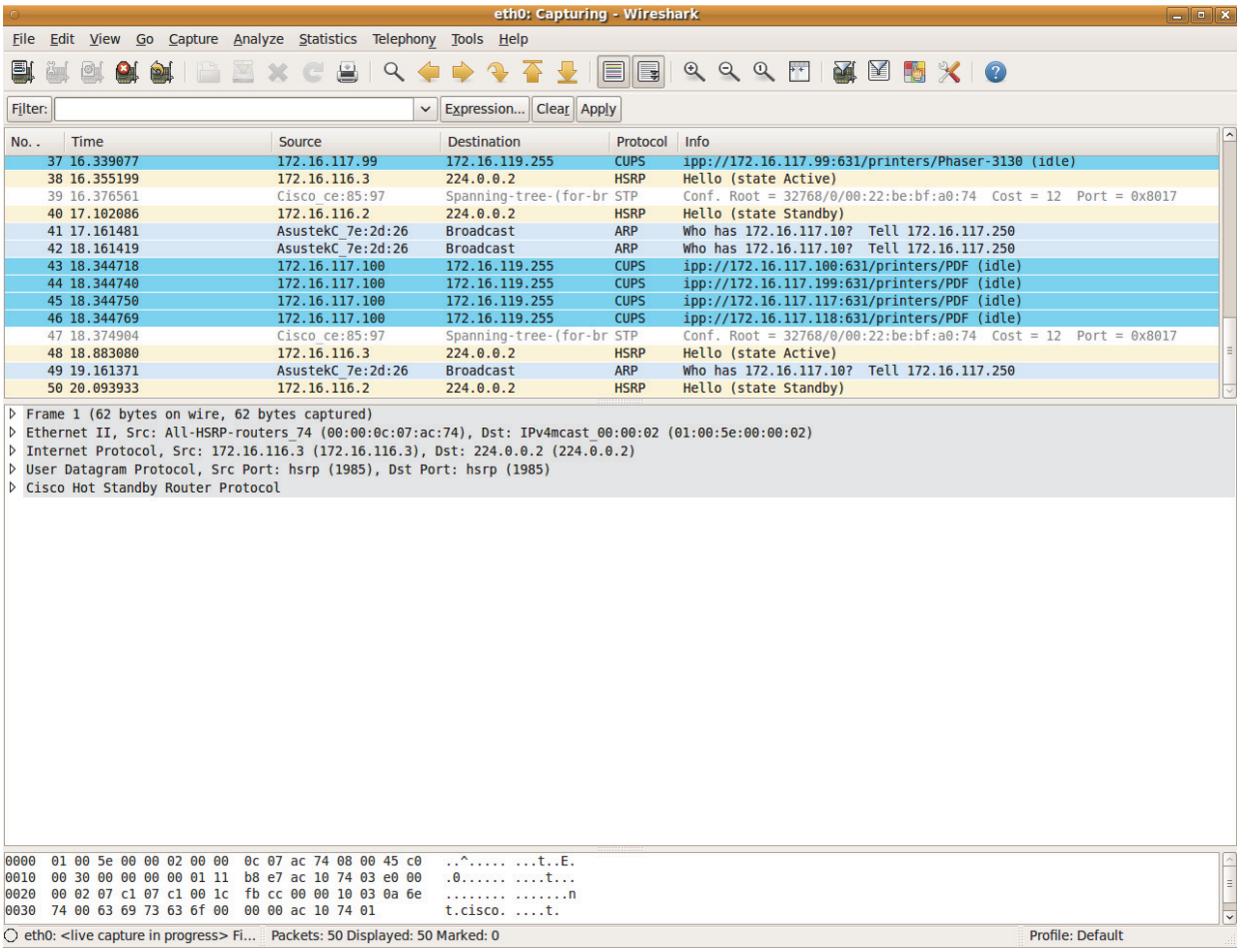

Рис. 2.1. Главное окно Wireshark в режиме мониторинга пакетов

Рассмотрим вид Главного окна программы Wireshark в режиме мониторинга пакетов (рис. 2.1):

- строка меню, содержит команды основного меню;
- панель инструментов, предоставляющая удобное средство для быстрого выполнения команд и процедур;
- панель Filter (фильтр) используется для задания критерия фильтрации пакетов, например по типу протокола;
- окно отслеживаемых пакетов, отображающее все отслеживаемые пакеты в формате списка – каждая строка списка соответствует одному пакету. При нажатии на соответствующую строку данного окна можно управлять данными, отображаемыми в нижних окнах: окне детального представления пакета и окне побайтового отображения пакета. В окне отслеживаемых пакетов доступна такая информация, как: номер пакета (столбец №), относительное время получения пакета (столбец Time, отсчет производится от первого пакета; параметры отображения времени можно изменить в настройках), IP-адрес отправителя (столбец Source),

IP-адрес получателя (столбец Destination), протокол, по которому пересылается пакет (столбец Protocol), а также дополнительная информация о нем (столбец Info). Для наглядности пакеты разных протоколов подсвечены разными цветами, что значительно упрощает анализ;

- окно детального представления пакета. Окно содержит детальную информацию о пакете, который был выбран в окне отслеживаемых пакетов. Данное окно отображает информацию о пакете (адреса и протоколы нижних уровней) и поля пакета в виде многоуровневой структуры, характерной для конкретного протокола, которую можно при необходимости свернуть или развернуть;
- окно побайтового отображения пакета. Окно отображает пакет, выбранный в окне отслеживаемых пакетов в виде HEX кода, т. е. побайтово;
- панель Статус содержит информационное сообщение: слева отображается детальная информация о текущем состоянии программы, а справа номер текущего отслеживаемого пакета.

Конфигурация интерфейса может быть легко изменена в меню View. Например, можно убрать окно побайтового отображения пакета (View -> Packet Bytes), так как в большинстве случаев (кроме анализа данных в пакете) оно не нужно и только дублирует информацию из окна детального представления пакета.

2.4. Мониторинг пакетов сетевого трафика сигнальных протоколов

Запуск режима мониторинга трафика

Для выбора сетевого интерфейса, с которого будет выполняться мониторинг пакетов, необходимо в строке Меню выбрать Capture > Interfaces (рис. 2.2) или нажать кнопку Interfaces, расположенную на панели инструментов.

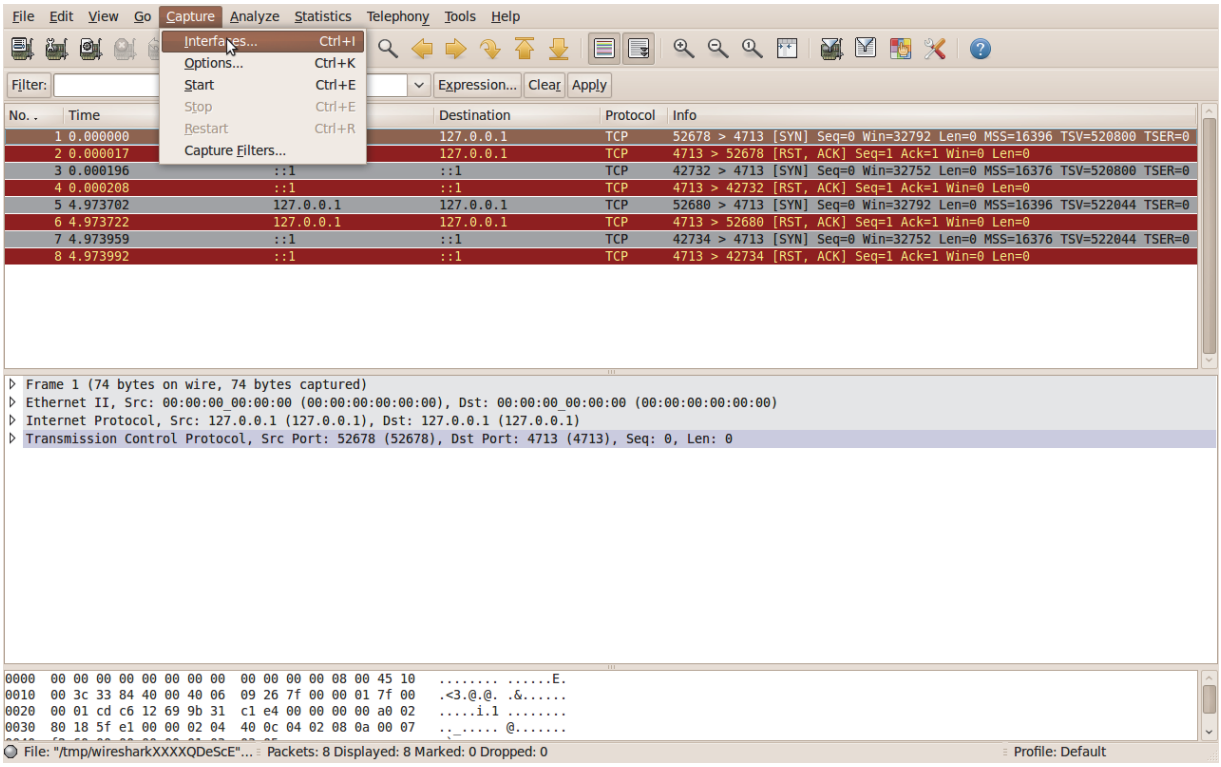

Рис. 2.2. Запуск режима мониторинга пакетов

После осуществления одного из вышеперечисленных действий появится диалоговое окно Capture Interfaces со списком сетевых интерфейсов, на которых может быть осуществлен мониторинг протоколов (рис. 2.3).

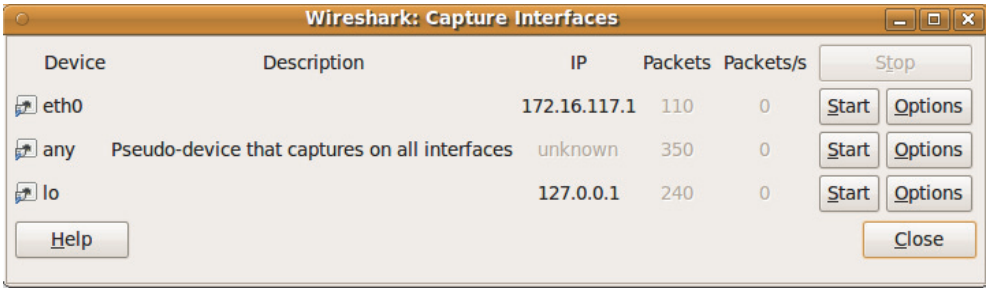

Рис. 2.3. Диалоговое окно Capture Interfaces

Диалоговое окно Capture Interfaces содержит информацию об интерфейсе

Description	Описание интерфейса

IP	IP-адрес интерфейса. В случае если IP-адрес интерфейса неизвестен (например, DHCP сервер недоступен), то вместо IP-адреса отображается значение «unknown»
Packets	Общее количество пакетов, отслеженных на конкретном интерфейсе с момента открытия диалогового окна
Packets/s	Количество пакетов, отслеженных в последнюю секунду

Кнопки для настройки режима мониторинга пакетов

Stop	Остановить текущий мониторинг пакетов
Start	Начать мониторинг пакетов на данном интерфейсе, используя настройки предыдущего режима мониторинга
Options	Открыть диалоговое окно Capture options с настройками режима мониторинга пакетов на данном интерфейсе
Close	Закрыть диалоговое окно

Диалоговое окно Capture options

Для настройки режима мониторинга пакетов на выбранном конкретном интерфейсе используется диалоговое окно Capture options (рис. 2.4). Для обращения к диалоговому окну Capture options необходимо в меню Capture выбрать пункт Options или в диалоговом окне Capture Interfaces (рис. 2.3) нажать кнопку Options.

Состав Capture

Interface	Определяет интерфейс, на котором будет осуществляться мониторинг пакетов. Необходимый интерфейс выбирается из выпадающего списка
IP address	IP-адрес выбранного интерфейса
Link-layer header type	Значение оставляется по умолчанию
Capture packets in promiscuous mode	Установка флажка в данном поле позволяет перевести интерфейс в режим приема всех сетевых пакетов (по умолчанию флажок должен быть установлен в данном поле)
Limit each packet to n bytes	Поле позволяет установить максимальную величину данных, которая будет отслеживаться для каждого пакета
Capture Filter	Поле определяет фильтр мониторинга. По умолчанию поле пустое

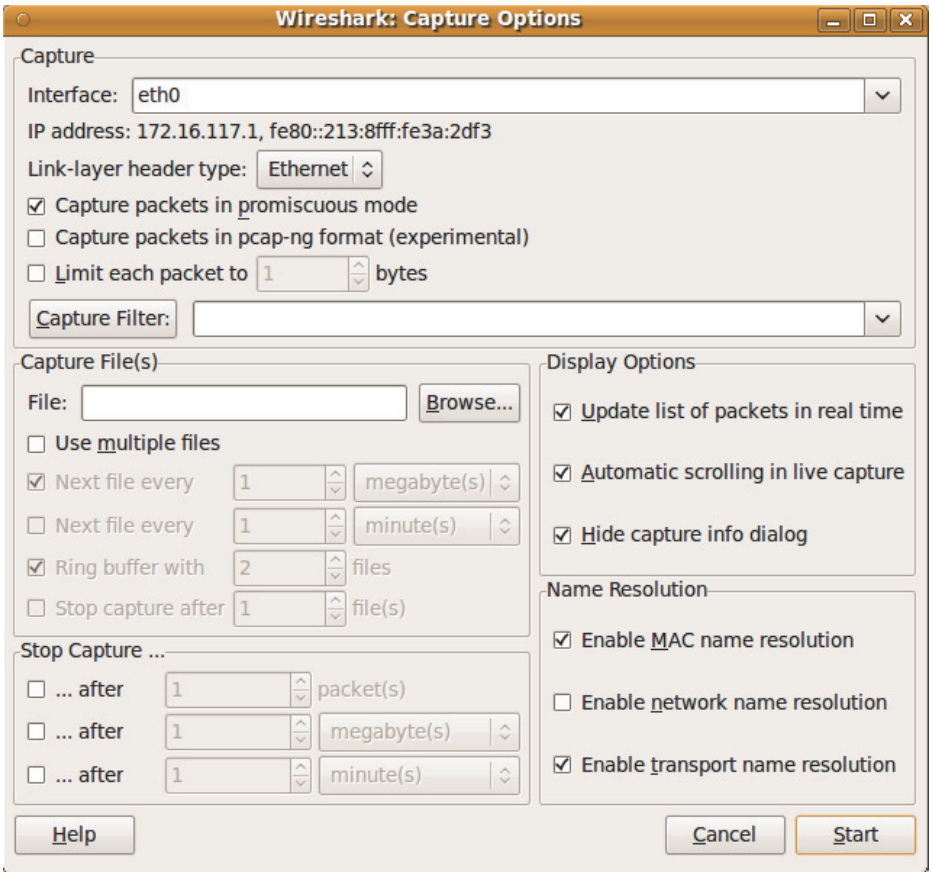

Рис. 2.4. Диалоговое окно Capture options

Состав Capture File(s)

File	Поле определяет имя файла для сохранения отслеживаемых данных. По умолчанию это поле пустое
Use multiple files	Вместо использования одного файла для сохранения отслеживаемых данных Wireshark автоматически переключается с одного файла на другой при достижении различных заданных условий
Next file every n megabyte(s)	Поле определяет момент переключения на следующий файл после того, как будет отслежено заданное количество байт, килобайт, мегабайт, гигабайт/с [1]
Next file every n minute(s)	Поле определяет момент переключения на следующий файл после истечения заданного времени (секунд, минут, часов, дней/с) [1]
Ring buffer with n files	Поле определяет круговой буфер из файлов сохранения отслеживаемых данных с заданным числом файлов
Stop capture after n file(s)	Остановить мониторинг данных после выполнения заданного количества переходов с одного файла на другой

Состав Stop Capture

after n packet(s)	Поле определяет количество пакетов, при достижении которого необходимо остановить мониторинг данных
after n megabytes(s)	Поле определяет количество байт, килобайт, мега-

[1] Только в случае, если в поле Use multiple files установлен соответствующий флажок.

	байт, гигабайт/с данных, при достижении которого необходимо остановить мониторинг данных
after n minute(s)	Поле определяет момент времени (секунды, минуты, часы, дни/с), по истечении которого необходимо остановить мониторинг данных

Состав Display Options

Update list of packets in real time	Поле определяет, что Wireshark должен обновлять окно отслеживаемых пакетов в режиме реального времени (по умолчанию флажок должен быть установлен в данном поле)
Automatic scrolling in live capture	Поле определяет, что Wireshark должен прокручивать (пролистывать, сдвигать) окно отслеженных пакетов при появлении новых пакетов так, чтобы всегда был виден последний отслеженный пакет (по умолчанию флажок должен быть установлен в данном поле)
Hide capture info dialog	Если данная опция выбрана, то диалоговое окно информации мониторинга Capture Info будет скрыто (по умолчанию флажок должен быть установлен в данном поле)

Name Resolution frame

Enable MAC name resolution	Данная опция позволяет контролировать транслирует ли Wireshark MAC адреса в имена или нет (по умолчанию флажок должен быть установлен в данном поле)
Enable network name resolution	Данная опция позволяет контролировать транслирует ли Wireshark сетевые адреса в имена или нет
Enable transport name resolution	Данная опция позволяет контролировать транслирует ли Wireshark транспортные адреса в протоколы или нет (по умолчанию флажок должен быть установлен в данном поле)

После того как были выбраны все требуемые опции и выставлены все величины, простое нажатие кнопки Start позволяет начать мониторинг данных, а нажатие кнопки Cancel – отменить мониторинг данных.

Когда Wireshark запущен в режим мониторинга пакетов и требуемые пакеты отслежены, остановить режим мониторинга пакетов можно нажатием кнопки Stop, расположенную на панели инструментов, либо в диалоговом окне информации мониторинга Capture Info, либо в пункте главного меню Capture.

Режим мониторинга

Когда мониторинг трафика запущен, появляется диалоговое окно информации мониторинга Capture Info [2] Данное диалоговое окно информирует о числе отслеженных пакетов и о времени начала мониторинга (рис. 2.5).

Запущенный сеанс мониторинга пакетов может быть остановлен следующими способами:

- кнопкой Stop диалогового окна информации мониторинга Capture Info (рис. 2.5);

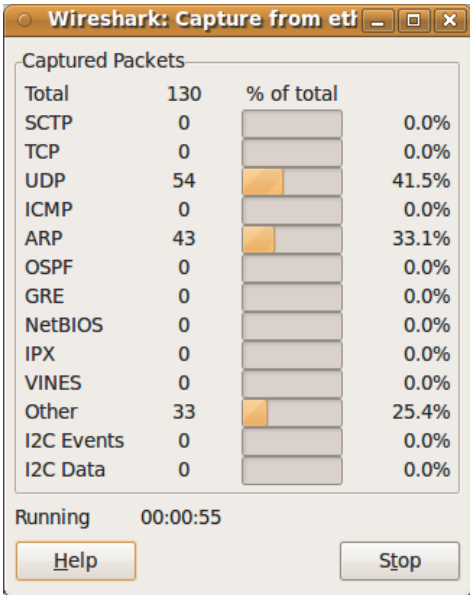

Рис. 2.5. Окно Capture Info

- кнопкой Capture/ Stop главного меню;
- кнопкой Stop панели инструментов (рис. 2.6).

[2] Только в случае, если не активированы функции Update list of packets in real time и Automatic scrolling in live capture при настройке режима мониторинга пакетов.

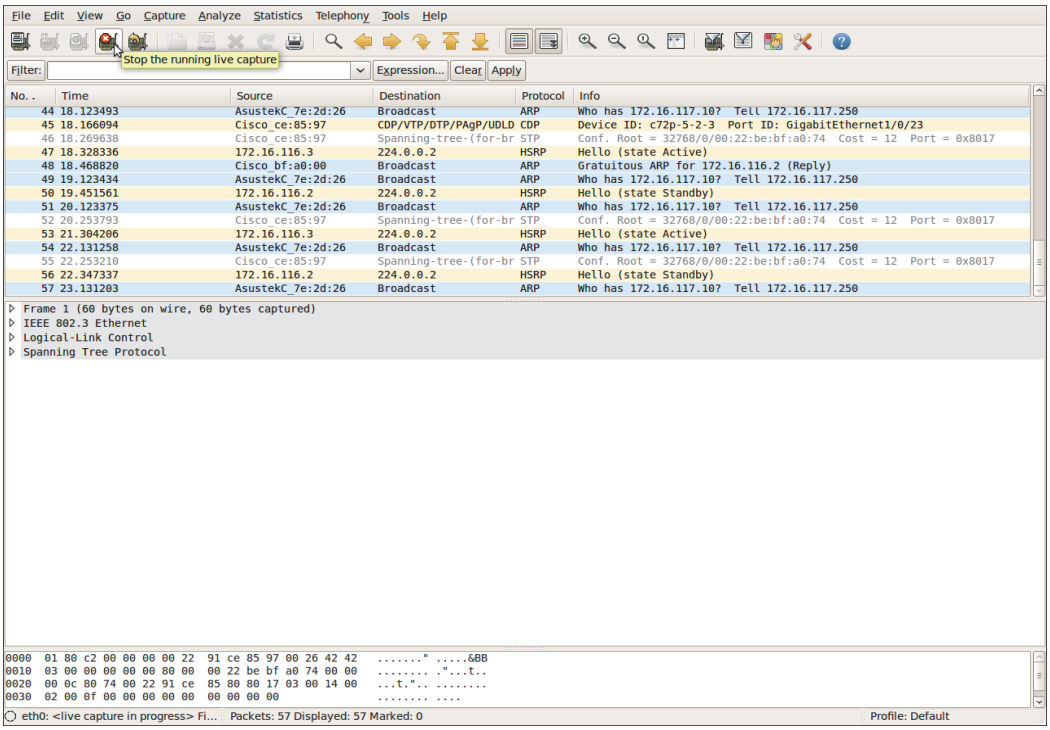

Рис. 2.6. Панель инструментов для остановки мониторинга пакетов

Перезапуск сеанса мониторинга пакетов

Запущенный сеанс мониторинга пакетов может быть перезапущен с предыдущими характеристиками фильтра, при этом все ранее отслеженные пакеты будут удалены. Данная опция полезна в случае, если в процессе мониторинга были отслежены не вызывающие интереса пакеты, которые нет необходимости сохранять.

Перезапуск может осуществляться одним из следующих способом:

- кнопкой главного меню Capture/ Restart;
- кнопкой Restart панели инструментов, расположенной справа от кнопки Stop.

Фильтрация трафика в режиме мониторинга пакетов

Wireshark имеет два языка фильтрации: один используется в режиме мониторинга пакетов, а другой – в режиме просмотра пакетов.

В Wireshark предусмотрена функция фильтрации информации в процессе мониторинга, таким образом, пользователь может задать нужный фильтр и анализировать только те пакеты, которые его интересуют, остальные отслеженные пакеты будут скрыты.

Эти фильтры позволяют выбирать пакеты:

1) по протоколу;
2) по IP адресу;
3) по определенному сообщению протокола;
4) по значению любого поля;

5) по величине поля и т. д.

Фильтр мониторинга вводится в поле панели Filter (Фильтр) Главного окна программы Wireshark (рис. 2.7).

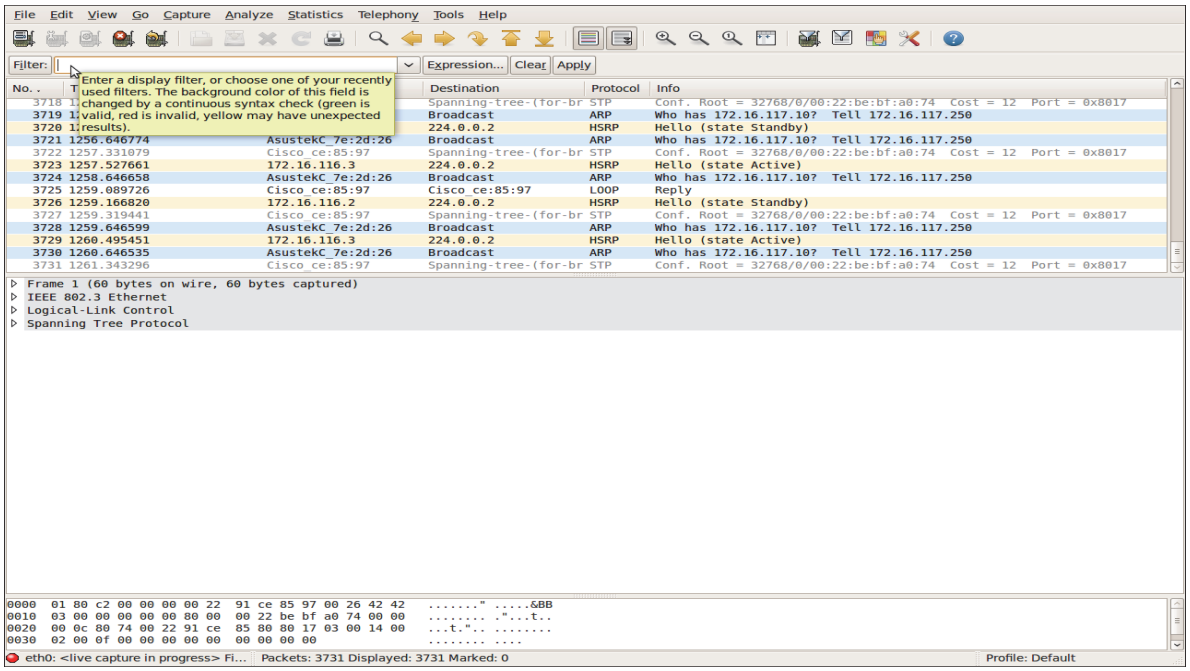

Рис. 2.7. Панель Filter Главного окна

Фильтр мониторинга записывается в виде последовательности элементарных выражений, соединенных логическими операндами (and/or или &/||) иногда с предшествующим операндом not. Все выражения пишутся прописными буквами.

Для выбора пакетов по типу протокола необходимо просто набрать строчными буквами в поле Filter панели инструментов, название соответствующего протокола, например sip и нажать Enter. Таким образом, будут отображены только пакеты протокола SIP (т. е. остальные отслеженные пакеты скрыты).

Для выбора пакетов по IP-адресу необходимо набрать в поле Filter панели инструментов выражение: *ip.addr==IP-адрес*, где IP-адрес – это адрес, по которому необходимо осуществить фильтрацию.

В случае, если необходимо фильтровать по нескольким IP-адресам, то необходимо ввести следующее выражение:
ip.addr==192.168.0.1 || ip.addr==192.168.0.2 || ip.addr==192.168.0.3 и т. д., либо
ip.addr==192.168.0.1 or ip.addr==192.168.0.2 or ip.addr==192.168.0.3

При использовании фильтров в режиме просмотра пакетов все пакеты сохраняются в файле отслеженных пакетов. Фильтры меняют только режим отображения файлов с отслеженными пакетами, а содержимое файлов остается неизменным.

Фильтрацию можно осуществлять по любому протоколу, который Wireshark понимает, также можно фильтровать по некоторым полям, содержащимся в отслеженных пакетах, и по определенным сообщениям протокола. Список таких полей доступен в диалоговом окне Add Expression. Например, для фильтрации по сообщению INVITE протокола sip в поле Filter панели инструментов необходимо ввести выражение:

sip.method==INVITE

Диалоговое окно Filter Expression

Для запуска процесса фильтрации в поле Filter (Фильтр) панели инструментов необходимо ввести соответствующее выражение. Язык записи выражения для фильтрации достаточно прост, в случае возникновения проблем с составлением выражения для фильтрации пакетов следует обратиться к диалоговому окну Filter Expression (нажать кнопку Expression, расположенную рядом с полем ввода Filter) (рис. 2.8).

Окно Filter Expression содержит следующие поля

Field Name	Содержит многоуровневый список полей для различных протоколов. При нажатии на знак рядом с названием протокола появляется список названий полей, доступных для фильтрации данного протокола
Relation	Содержит различные выражения отношения. Is present – простейшее отношение, которое является верным, если выбранное поле представлено в пакете. Все остальные перечисленные отношения бинарные (двойные), для которых требуются дополнительные величины

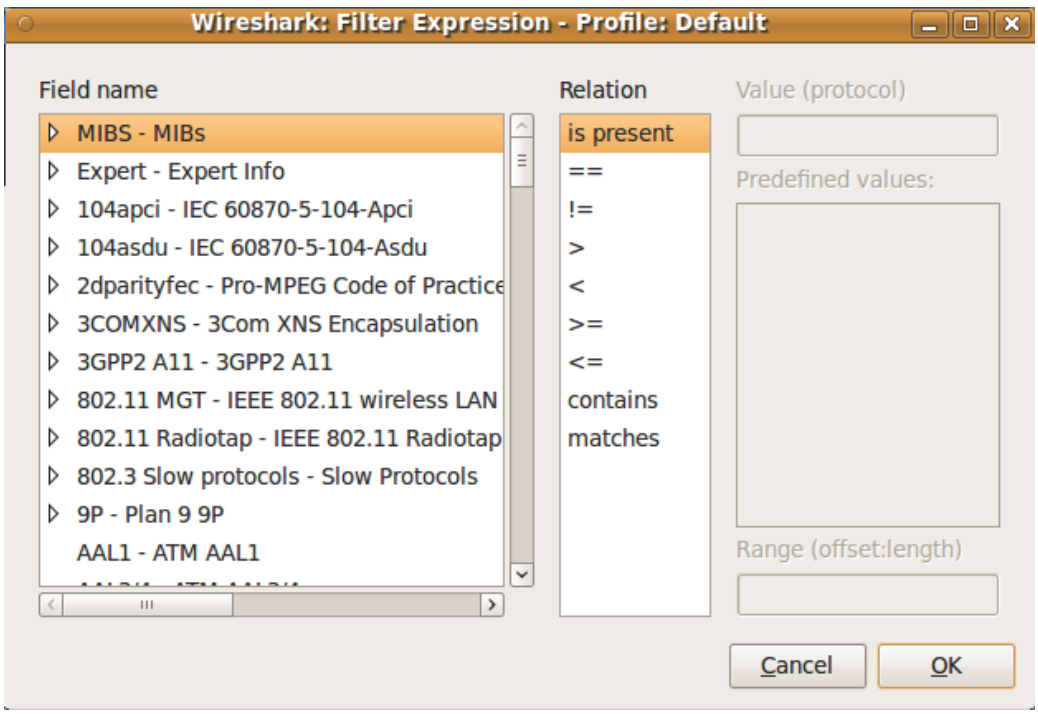

Рис. 2.8. Диалоговое окно Filter Expression

После выбора поля из списка Field Name и бинарного отношения (например, равное отношение ==) у пользователя появляется возможность ввести какое-либо значение или величину и возможно какую-либо дополнительную информацию.

Value	В это поле водится подходящая величина. Это поле также может указывать тип величины для выбранного Field Name
Predefind values	Некоторые поля протоколов имеют предопределенные доступные величины. Если для выбранного поля протокола такие величины определены, то одну из них можно выбрать в данном поле
Range	XXX – добавляется объяснение в данное поле
OK	Когда приемлемое выражение сформировано, необходимо нажать кнопку OK и строка поля Filter (Фильтр) будет заполнена
Отменить	При нажатии кнопки «Отменить» пользователь покидает окно Filter Expression

Сохранение отслеженной информации

Чтобы сохранить отслеженные пакеты, необходимо в главном меню выбрать «File/Save as» (рис. 2.9), либо нажать кнопку «Save this capture file…» на панели инструментов.

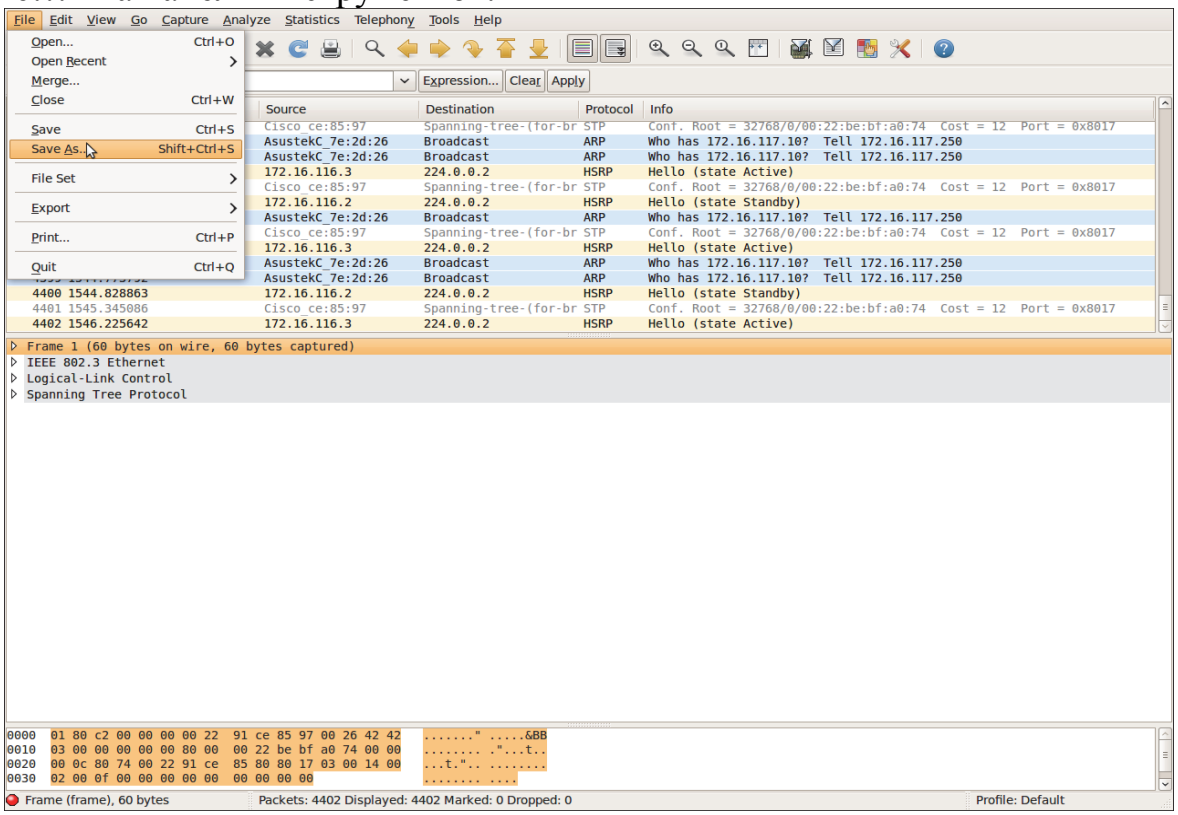

Рис. 2.9. Сохранение данных

Далее появится диалоговое окно Save Capture File As, в котором необходимо ввести имя файла и выбрать место сохранения. Также в данном окне имеется возможность выбрать файл для сохранения данных, выбрать пакеты данных, которые необходимо сохранить (т. е. будет сохраняться не вся отслеженная информация, а только заданные пакеты).

2.5. Работа с отслеженными пакетами

Открытие файла с отслеженной информацией

Wireshark отображает сохраненные ранее файлы с отслеженной информацией. Для открытия файла необходимо в Главном меню выбрать File/ Open и выбрать соответствующий файл.

Отображение отслеженных пакетов

После мониторинга некоторого числа пакетов или после открытия ранее сохраненного файла с отслеженными пакетами пользователь может посмотреть и проанализировать любой пакет, находящийся в окне отслеживаемых пакетов, более подробно. Для этого необходимо просто нажать левой кнопкой мыши на соответствующую строку данного окна и в окне детального представления пакетов и в окне побайтового отображения пакетов появится более подробная информация о выбранном пакете (рис. 2.10).

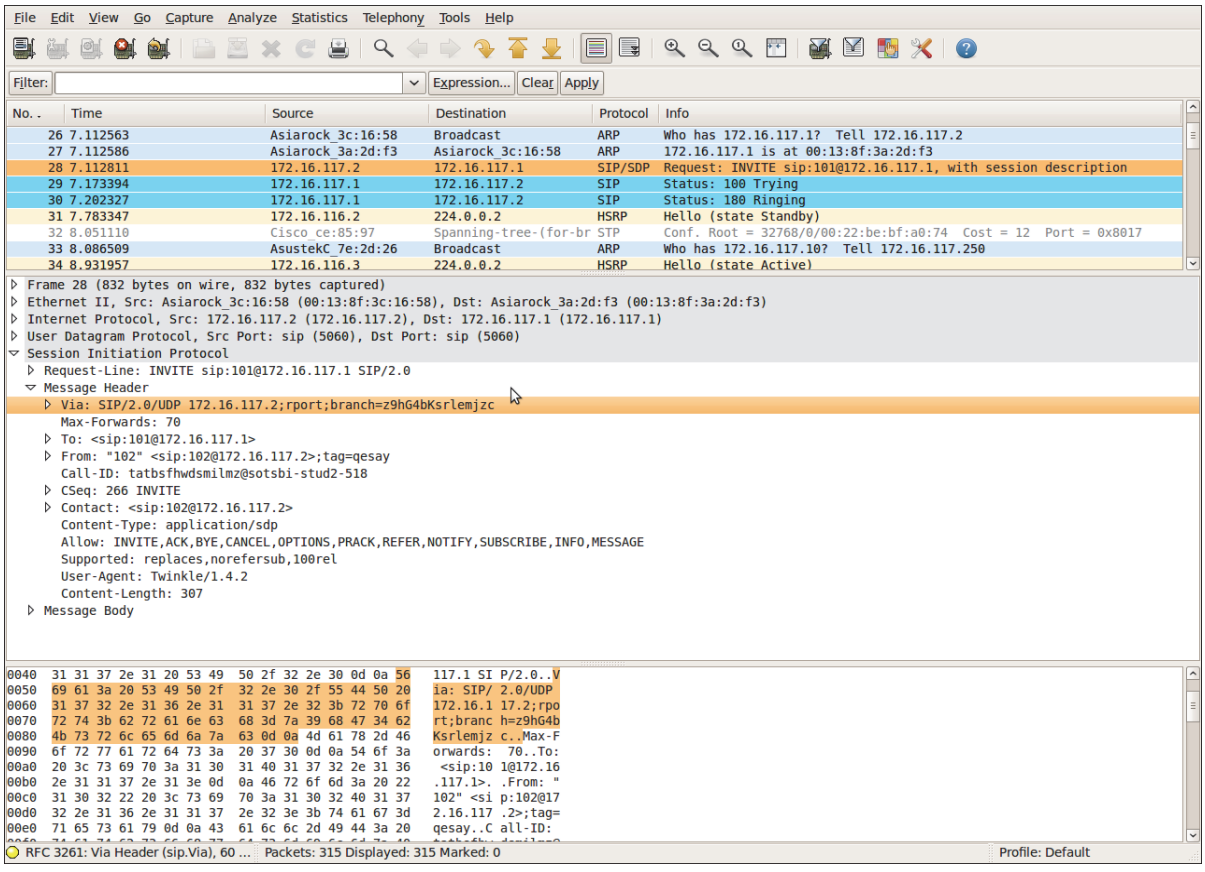

Рис. 2.10. Главное окно Wireshark в режиме мониторинга пакетов

В окне детального отображения пакетов имеется возможность раскрыть некоторые поля многоуровневой структуры нажатием на символ стрелки, расположенный слева.

Таким же образом можно выбирать и просматривать пакеты, когда Wireshark находится в режиме мониторинга пакетов, для этого необходимо выбрать Update list of packets in real time в диалоговом окне Wireshark Capture Preferences.

Фильтрация пакетов в режиме просмотра

Фильтр в режиме просмотра пакетов позволяет пользователю анализировать только те пакеты, которые его интересуют, остальные пакеты будут просто скрыты.

При использовании фильтров в режиме просмотра пакетов все пакеты сохраняются в файле отслеженных пакетов. Фильтры меняют только режим отображения файлов с отслеженными пакетами, а содержимое файлов остается неизменным.

Отображение отслеженной информации в виде графиков и диаграмм

Wireshark позволяет отображать и сохранять отслеженную информацию в виде графиков и диаграмм. Раздел Главного меню Statistics→Flow Graph позволяет вывести на экран диаграмму обмена сообщениями различных протоколов. На рис. 2.11 представлен пример построения диаграммы обмена сообщениями протокола SIP.

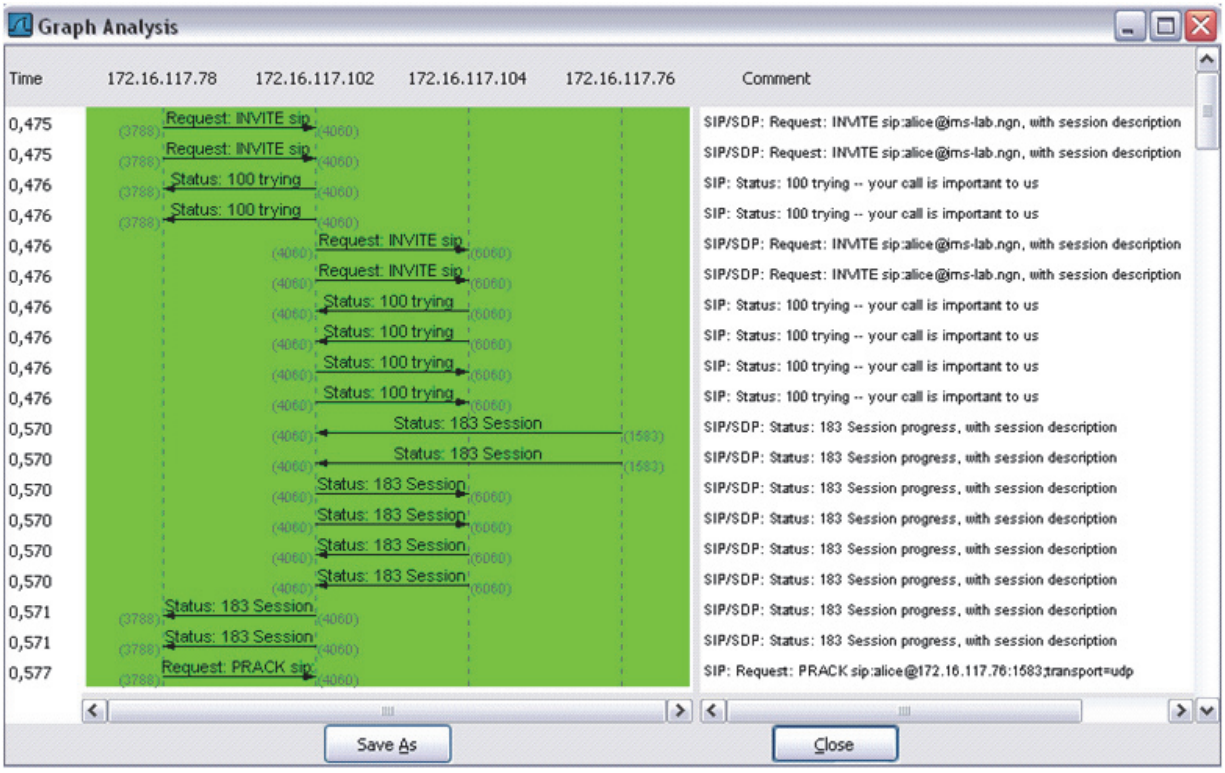

Рис. 2.11. Диаграмма обмена сообщениями протокола SIP

2.6. Формат отображения времени и временные метки

В процессе мониторинга пакетов каждому пакету присваивается временная метка. Эти временные метки сохраняются в файле с отслеженными пакетами и доступны для дальнейшего анализа.

Формат временной метки и точность представления в окне отслеженных пакетов может быть выбрана в пункте Главного меню View.

Доступные форматы отображения

Date and Time of Day: 1970-01-01 01:02:03.123456	Абсолютная дата дня и время, когда пакет был отслежен
Time of Day: 01:02:03.123456	Абсолютное время, когда пакет был отслежен
Seconds Since Beginning of Capture: 123.123456	Время относительно запуска отслеживаемого файла или первой привязки по времени перед этим пакетом
Seconds Since Previous Packet: 1.123456	Время относительно предыдущего пакета

Доступная точность отображения данных

Automatic	Точность отображения определяется автоматически (по умолчанию)
Seconds, Deciseconds, Centiseconds, Milliseconds, Microseconds or Nanoseconds	Точность отображения данных определяется заданными настройками

3. ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИГОН ТЕХНОЛОГИЙ И ПРОТОКОЛОВ «СОТСБИ OSI»

3.1. Назначение

Исследовательский полигон технологий и протоколов «СОТСБИ OSI» предназначен для проведения практических работ по изучению модели взаимодействия открытых систем (OSI) на примере популярных сервисов Интернет. Исследовательский полигон может выступать как в качестве независимого полигона OSI, позволяющего детально рассмотреть работу протоколов на любом уровне, так и являться составляющей частью более сложных полигонов (NGN, GUARD и т. д.)

3.2. Компоненты полигона СОТСБИ OSI

Сеть полигона СОТСБИ OSI, используемая для проведения практических работ состоит из следующих основных элементов:

- сервер OSI со следующими приложениями: HTTP, HTTPS, FTP, NTP, NFS, TFTP, SSH;
- сервер DNS;
- рабочие места учащихся.

Рабочее место

Комплектация рабочего места магистра зависит от типа полигона, в который интегрирован комплекс «СОТСБИ OSI». Для проведения практических работ используются приложения, указанные в табл. 3.1.

Таблица 3.1

Пиктограмма	Название	Описание
	Wireshark	«Wireshark» – ярлык для запуска программы мониторинга сетевого трафика Wireshark локально
	Wireshark на сервере	«Wireshark на сервере» – ярлык для запуска программы мониторинга сетевого трафика Wireshark на сервере
	Модель OSI	«Модель OSI» – ярлык для запуска различных сервисов Интернет на сервере OSI
	Библиотека	«Библиотека» – ярлык электронной библиотеки СОТСБИ-Lib, содержащей различную дополнительную литературу, необходимую при выполнении практической и исследовательской работы

4. ПРАКТИЧЕСКИЕ ЗАНЯТИЯ

Для выполнения практических занятий формируются бригады из расчета одно рабочее место на одну бригаду (далее РМ1, РМ2, РМ3 и т. д.).

При выполнении практических заданий используются следующие элементы полигона СОТСБИ-У:

- сервер DNS;
- сервер FTP;
- сервер NTP;
- сервер NFS;
- сервер TFTP;
- программа мониторинга сетевого трафика Wireshark.

В процессе выполнения практических занятий необходимо изучить уровни модели OSI на примере протоколов DNS, FTP, TFTP, NTP, NFS, HTTP, HTTPS, SSH. Описания протоколов представлены ниже.

4.1. DNS

Для обращения к хостам (компьютерам или устройствам) в сети Internet используются IP-адреса, однозначно идентифицирующие любой сетевой компьютер в этой глобальной сети. Однако для пользователей применение IP-адресов при обращении к хостам не слишком удобно. Поэтому в Internet принято присваивать имена всем компьютерам в сети. Применение в Internet мнемонически понятных для пользователей имен породило проблему их преобразования в IP-адреса. Такое преобразование необходимо, так как на сетевом уровне адресация пакетов осуществляется не по именам, а по IP-адресам, следовательно, для непосредственной адресации сообщений в Internet имена не годятся. На раннем этапе развития Internet для решения проблемы преобразования имен в адреса использовался специальный файл (hosts.txt), в который вносились имена и соответствующие им IP-адреса всех хостов в сети. Данный файл регулярно обновлялся и распространялся по всей сети. Но по мере развития Internet число хостов, объединенных в сеть, увеличивалось, и данная схема становилась все менее работоспособной. В 1983 г. Полом Мокапетрисом была создана новая система преобразования имен, позволяющая пользователю в случае отсутствия у него информации о соответствии имен и IP-адресов получить необходимые сведения от ближайшего информационно-поискового сервера. Эта система получила название распределенной системы имен доменов – DNS (Domain Name System) и стандартизирована в 1987 г. RFC 1034 и 1035.

Для реализации системы DNS был создан специальный сетевой протокол DNS, по которому взаимодействуют информационно-поисковые DNS-серверы. В упрощенном виде алгоритм работы DNS по поиску адресов web-сайтов можно описать следующим образом. Когда пользователь вводит в адресной строке браузера адрес web-сайта, компьютер выполняет запрос к тому или иному известному этому компьютеру серверу DNS, «спрашивая» сервер о том, какой IP-адрес связан с «доменным адресом», указанным пользователем. В ответ сервер DNS, проверив соответствие по своим внутренним таблицам или выполнив запрос к другим серверам DNS,

присылает искомый IP-адрес. Далее браузер устанавливает соединение с web-сайтом уже по IP-адресу.

Передача сообщений

В DNS сообщения передаются либо UDP-дейтограммами, либо по виртуальному каналу протокола TCP. Дейтограммы являются предпочтительными для передачи запросов, так как они меньше загружают сеть и обладают более высокой производительностью, тогда как обновление базы данных сервера должно осуществляться с использованием виртуального канала, который обеспечивает надежную передачу данных. Для доступа к серверу имен используется порт 53 протоколов TCP и UDP.

Сообщения, посылаемые по UDP-протоколу, ограничиваются 512 байтами (не включая IP и UDP заголовки). Сообщения большей длины разбиваются, и в заголовке устанавливается бит TC.

Формат сообщения DNS

Все взаимодействия, определенные протоколом DNS, осуществляются с использованием сообщений определенного формата. DNS-запрос и DNS-ответ представляют собой единственные два типа сообщений, используемые протоколом DNS. Форматы этих сообщений совпадают.

Каждое сообщение состоит их пяти секций (некоторые из которых при определенных обстоятельствах остаются незаполненными):

- секция заголовка имеет фиксированную длину 12 байт и присутствует в сообщении всегда;
- секция запроса (question) содержит поля, описывающие запрос: тип запроса (QTYPE), класс (QCLASS) и доменное имя запроса (QNAME);
- секция ответов (answer) содержит записи о ресурсах, которые отвечают на запрос;
- секция полномочий (authority) содержит записи о ресурсах, которые указывают на полномочные серверы; дополнительно может содержать запись о ресурсе «начала полномочий» (start of authority или SOA) для авторизованных данных в секции ответов;
- дополнительная секция (additional) содержит записи о ресурсах, которые связаны с запросом, но не точно отвечают на запрос.

4.2. FTP

Появившись в 1971 г., протокол FTP (File Transfer Protocol – протокол передачи файлов) сегодня является стандартным протоколом, предназначенным для передачи файлов по TCP-сетям (например, Интернет). Спецификация протокола определена в RFC 959.

Протокол имеет «клиент-серверную» архитектуру, FTP-сеанс представляет собой обмен файлами, находящимися на двух хостах – клиента и

сервера. Для получения доступа к серверу пользователю необходимо ввести свои логин и пароль. Пользователи FTP могут пройти аутентификацию, передавая логин и пароль открытым текстом, или, если это разрешено на сервере, они могут подключиться анонимно. Можно использовать протокол SSH для безопасной передачи, скрывающей (шифрующей) логин и пароль, а также шифрующей содержимое. После распознавания пользователя сервером начинается процесс передачи файлов в нужном направлении.

Протокол FTP использует два параллельных TCP-соединения: управляющее соединение и соединение данных. Управляющее соединение служит для пересылки управляющей информации между двумя хостами: имени пользователя и пароля, команд смены текущего удаленного каталога, передачи и запроса файлов. Соединение данных предназначено для передачи самих файлов. Соединение передачи сигналов управления остается открытым в течение всей интерактивной сессии FTP. Соединение передачи данных каждый раз открывается командой, чтобы вызвать передаваемый файл, и затем закрывается, когда файл передан. Каждое соединение позволяет передать только один файл; таким образом, множественный обмен вызывает необходимость многократной установки соединения данных.

Управляющее соединение

Соединение для передачи команд управления создается в два этапа (рис. 4.1):

1) сервер пассивно открывается, подключается к заданному порту (порт 21) и ждет клиента;

2) клиент использует временный порт, и сессия активно открывается.

Рис. 4.1. Управляющее соединение

Через управляющее соединение передаются команды, посылаемые клиентом серверу, и ответы сервера (отклики) в кодировке ASCII для 7-разрядных символов. Таким образом, команды FTP могут быть легко про-

читаны человеком. Для разделения команд используются пары символов перехода на новую строку (возврат каретки и перевод строки). Имя команды представляет собой четыре символа в верхнем регистре, за которым могут следовать параметры.

Команды

Все команды можно разделить на шесть групп: команды доступа, команды управления файлами, команды форматирования данных, команды определения порта, команды передачи файла и прочие команды.

• Команды доступа. Эти команды позволяют пользователю обращаться к удаленной системе:

- USER – пользовательская информация;
- PASS – пароль;
- ACCT – учетная информация;
- REIN – перезапуск;
- QUIT – выход из системы;
- ABOR – прерывание предыдущей команды.

• Команды управления файлом. Эти команды дают пользователю возможность обращаться к удаленному компьютеру, передвигаться по структуре директории, создавать новые директории, удалять файлы и т. д.:

- CWD – изменение другой директории;
- CDUP – изменение вышестоящей директории;
- DELE – удаление файла;
- LIST – список поддиректорий и файлов;
- NLIST – список имен поддиректорий или файлов, не имеющих атрибутов;
- MKD – создать новую директорию;
- PWD – имя текущей директории на дисплее;
- RMD – удалить директорию;
- RNER – идентификатор файла, который будет переименован;
- RNTO – переименование файла;
- SMNT – вершина системы.

• Команды форматирования данных. Эти команды дают пользователю возможность определить данные о структуре, типе файла и режиме передачи. Определенный формат затем используется командами передачи файлов:

- TYPE – определяет тип файла, если необходим формат для печати;
- STRU – определяет организацию данных;
- MODE– определяет режим передачи.

• Команды определения порта. Эти команды определяют номер порта для соединения передачи данных на стороне клиента:

- PORT – клиент выбирает порт;

- PASV – сервер выбирает порт.

• Команды передачи файла. Эти команды обеспечивают передачу файлов:

- RETR – извлечение файла: файл(ы) передан(ы) от сервера к клиенту;
- STOR – накопление файла: файл(ы) передан(ы) от клиента к серверу;
- APPE – совпадает с STOR за исключением того, что если файл существует, то данные могут быть прикреплены к нему;
- STOU – то же самое, что STORE, за исключением того, что имя файла будет уникальным в этой директории; однако существующий файл не должен быть переписан;
- AALLO– распределение места для накопления файлов в сервере;
- REST – установка отметки в определенной точке данных;
- STAT – возврат состояния файла.

• Различные команды. Эти команды доставляют информацию к пользователю FTP на стороне клиента:

- HELP – запрос информации;
- NOOP – проверка, является ли сервер действующим;
- SITE – определение сайта заданных команд;
- SYST – запрос об операционной системе, используемой сервером.

Отклики

Каждая FTP-команда вызывает по крайней мере один отклик. Отклик имеет две части: номер из трех цифр и текст. Числовая часть определяет код отклика; текстовая часть определяет необходимые параметры или дополнительные пояснения.

Первая цифра определяет состояние команды. В этой позиции может быть использована одна из пяти цифр:

1yz (положительный предварительный ответ). Действие началось. Сервер будет посылать другие отклики перед принятием другой команды;

2yz (положительный отклик завершения). Действие завершено. Сервер будет принимать другую команду;

3yz (положительный промежуточный отклик). Команда принята, но нужна дальнейшая информация;

4yz (отклик отрицательного переходного завершения). Действие не произошло, но ошибка временная. Та же самая команда будет послана позднее;

5yz (отклик отрицательного постоянного завершения). Команда не принята и должна быть повторена позже;

Вторая цифра также определяет состояние команды. В этой позиции может быть использована одна из шести цифр:

x0z (синтаксис);
x1z (информация);
x2z (подключение);
x3z (идентификация и учет);
x4z (не определено);
x5z (система файлов).
Третья цифра обеспечивает дополнительную информацию.

Соединение для передачи данных

Соединение для передачи данных использует заданный порт 20. Однако создание соединения для передачи данных отличается от предыдущего. FTP создает соединение для передачи данных следующим образом (рис. 4.2):

1) клиент (не сервер) вызывает пассивное открытие кратковременного порта. Это может быть сделано клиентом, потому что клиент вызывает команды для передачи файлов.

2) клиент посылает номер этого порта серверу, используя команду PORT.

3) сервер получает номер порта, вызывает активное открытие заданного порта 20 и получает номер временного порта.

Для передачи данных необходимо определить три атрибута: тип, структура данных и режим передачи.

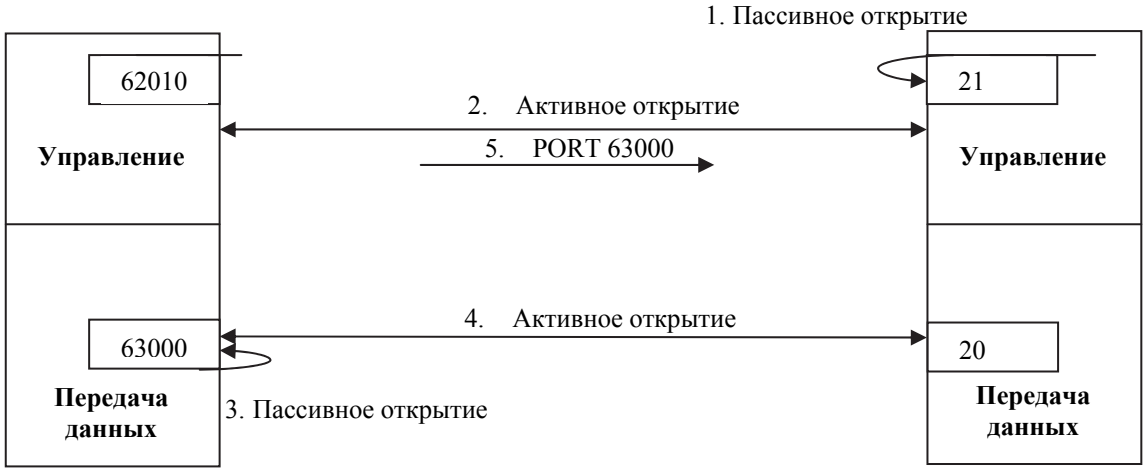

Рис. 4.2. Создание соединения для передачи данных

FTP может передавать через соединение для передачи данных следующие типы файлов:

- ASCII-файл. Это формат, используемый по умолчанию для трансляции текстовых файлов;
- EBCDIC-файл. Используется для передачи обычного текста в кодировке EBCDIC;

• Image-файл. Этот файл по умолчанию – формат для передачи двоичных файлов. Файл передается, как непрерывный поток бит без всякой интерпретации и кодирования. Он в большинстве случаев используется для передачи двоичных файлов, таких как компилированная программа.

FTP может передавать файл по соединению для передачи данных, используя одну из следующих интерпретаций структуры данных:

• файловая структура (по умолчанию). Файл не имеет структуры, представляет собой непрерывный поток данных;

• структура записи. Файл представляет собой серию последовательных записей;

• страничная структура. Это файл, разделенный на страницы, каждая страница имеет номер и заголовок страницы.

FTP может передавать файл по соединению для передачи данных, используя один из трех следующих режимов передачи:

• поточный режим. Это режим по умолчанию. Данные посылаются в виде непрерывного потока, освобождая FTP от выполнения какой бы то ни было обработки. Вместо этого, вся обработка выполняется TCP. Индикатор конца файла не нужен, за исключением разделения данных на записи;

• блочный режим. FTP разбивает данные на несколько блоков (блок заголовка, количество байт, поле данных) и затем передает их TCP;

• сжатый режим. Если файл большой, данные могут быть сжаты единым алгоритмом (обычно кодированием длин серий).

4.3. TFTP

Существует много случаев, когда требуется просто копировать файл без необходимости использовать все функциональные возможности протокола FTP. Например, когда необходимо загрузить загрузочный и конфигурационный файлы на бездисковые рабочие станции или маршрутизаторы. Тривиальный протокол передачи файла (Trivial File Transfer Protocol – TFTP) разработан для данного типа передач. Спецификация протокола определена в RFC 1350.

Сообщения

Имеется пять типов TFTP-сообщений:

• Read Request (RRQ, #1) – запрос на чтение файла используется клиентом для установления соединения для чтения данных от сервера;

• Write Request (WRQ, #2) – запрос на запись файла используется клиентом для установления соединения для записи данных на сервер;

• Data (DATA, #3) — данные, передаваемые через TFTP;

• Acknowledgment (ACK, #4) – подтверждение пакета;

• Error (ERR, #5) – ошибка.

Установление соединения

Обмен между клиентом и сервером начинается с того, что клиент запрашивает сервер либо прочитать, либо записать файл для клиента. В стандартном варианте загрузки бездисковой системы первый запрос – это запрос на чтение.

Установление соединения для чтения файла отличается от установления соединения для записи файлов:

чтение. Чтобы установить соединение для чтения, клиент TFTP посылает сообщение RRQ. Имя файла и режим передачи определяются в этом сообщении. Если сервер может передавать файл, он отвечает положительно сообщением DATA, содержащим первый блок данных. Если имеются проблемы, такие как трудности в открытии файла или ограничения в разрешении, сервер отвечает отрицательно посылкой сообщения ERROR;

запись. Чтобы установить соединение для записи, клиент TFTP использует сообщение WRQ. Имя файла и режим передачи определяется в этом сообщении. Если сервер может принять копию файла, он отвечает положительно сообщением ACK, используя значение 0 для блока данных. Если имеются проблемы, сервер отвечает отрицательно посылкой сообщения ERROR.

Завершение соединения

После того как передан полный файл, соединение может быть завершено. TFTP не имеет специального сообщения о завершении. Завершение сопровождается посылкой последнего блока данных, который содержит менее чем 512 байт.

Передача данных

Для передачи данных TFTP использует службу UDP, которая ненадежна. Файл разделяется на блоки данных, в которых каждый блок, исключая последний, содержит точно 512 байтов. Последний блок должен быть между 0 и 511 битами. TFTP может передавать данные в ASCII или в двоичном формате.

UDP не имеет механизма для управления потоком и контроля над ошибками, поэтому TFTP должен создать эти механизмы, чтобы осуществить передачу файла непрерывными блоками данных.

Управление потоками

TFTP посылает блоки данных, используя сообщение DATA, и ожидает сообщение ACK. Если отправитель получает сигналы подтверждения прежде, чем сработал таймер (time-out), он посылает следующий блок. Таким образом, управление потоком – это продвижение нумерации блоков данных и ожидание ACK, прежде чем послан следующий блок данных.

Контроль ошибок

TFTP-механизм контроля над ошибками отличается от других протоколов. Он симметричный – это означает, что отправитель и получатель оба используют тайм-аут. Отправитель применяет его для сообщений данных;

получатель – для подтверждения сообщений. Если сообщение данных потеряно, отправитель повторяет его после истечения тайм-аута. Если подтверждение потеряно, получатель повторяет его после истечения тайм-аута. Эта гарантирует нормальную работу.

Поврежденное сообщение

Если блок данных поврежден, это обнаруживается приемником и блок удаляется. Передатчик ожидает подтверждение и не получает его в период тайм-аута. Блок посылается опять.

Потеря сообщения

Если блок потерян, он никогда не достигнет приемника, и не будет послано подтверждение. Передатчик повторяет блок после тайм-аута.

Потеря подтверждения

Если потеряно подтверждение, возможны две ситуации. Если таймер приемника сработает раньше таймера передатчика, приемник повторит подтверждение; в другом случае передатчик повторит данные.

Дублирование сообщения

Дублирование блока может быть обнаружено приемником по номеру блока. Если блок дублирован, он просто удаляется приемником.

4.4. HTTP

HTTP (HyperText Transfer Protocol) – один из наиболее важных протоколов, который обеспечивает передачу данных через Интернет. Протокол HTTP является важнейшей частью веб-приложений. HTTP был предложен в марте 1991 г. Тимом Бернерсом-Ли, как механизм для доступа к документам в Интернете. Самая ранняя версия протокола HTTP/0.9 была впервые опубликована в январе 1992 г. В мае 1996 г. для практической реализации HTTP был выпущен информационный документ RFC 1945, что послужило основой для реализации большинства компонентов HTTP/1.0. Текущая версия протокола, принята в июне 1999 г. и определена в RFC 2616.

Протокол HTTP находится на седьмом прикладном уровне модели OSI и в качестве транспортного протокола использует протокол TCP с портом 30.

Протокол HTTP предполагает передачу данных в режиме «запрос-ответ». При этом в рамках такого взаимодействия могут передаваться данные практически любого типа – обычный текст, гипертекст (HTML), таблицы стилей, клиентские сценарии, изображения, документы в различных форматах, бинарная информация и т. д.

В рамках протокола HTTP всегда четко выделяется клиент и сервер. Клиент всегда является инициатором взаимодействия. Сервер, в свою очередь, прослушивает все входящие соединения и обрабатывает каждое из них. Поскольку HTTP-взаимодействие функционирует по схеме «запрос-ответ», то для инициации сеанса передачи данных необходимо сгенериро-

вать HTTP-запрос. В рамках этого запроса клиент описывает то, какой ресурс он хочет получить от сервера, а также указывает различные дополнительные параметры. После этого запрос отправляется серверу и тот, в свою очередь, обрабатывает запрос и генерирует HTTP-ответ, в котором содержится служебная информация и содержимое того ресурса, который был запрошен.

HTTP-запрос и HTTP-ответ сходны по своей структуре и называются HTTP-сообщениями. Фактически, все взаимодействие в рамках протокола HTTP сводится к пересылке HTTP-сообщений. Каждое HTTP-сообщение является обычной текстовой информацией, представленной в определенном формате.

HTTP-сообщение состоит из трех частей, которые передаются в указанном порядке:

1) стартовая строка (Starting line) – определяет тип сообщения;
2) заголовки (Headers) – характеризуют тело сообщения, параметры передачи и прочие сведения;
3) тело сообщения (Message Body) – непосредственно данные сообщения. Обязательно должно отделяться от заголовков пустой строкой.

Заголовки и тело сообщения могут отсутствовать, но стартовая строка является обязательным элементом, так как указывает на тип запроса/ответа. Исключением является версия 0.9 протокола, у которой сообщение запроса содержит только стартовую строку, а сообщение ответа – только тело сообщения.

HTTP-запрос

HTTP-запрос формируется клиентом и отправляется на сервер с целью получения информации от него. В нем содержится информация о ресурсе, который необходимо загрузить, а также дополнительные сведения. Первая строка содержит метод запроса, имя ресурса (с указанием относительного пути на сервере), а также версию протокола. Например, вид приветственной строки может быть определен как «GET /images/corner1.png HTTP/1.1». Такой запрос обращается к серверу с требованием выдать методом GET изображение, расположенное в папке «images» и имеющее название «corner1.png».

HTTP-заголовки имеют важное значение для HTTP-запроса, поскольку в них указывается уточняющая информация о запросе – версия браузера, возможности клиента принимать сжатое содержимое, возможности кэширования и другие важные параметры, которые могут влиять на формирование ответа. В теле HTTP-запроса обычно содержится информация, которую необходимо передать на сервер. Например, если требуется загрузить файл на сервер, то содержимое файла будет находиться в теле HTTP-запроса. Однако, размещение данных в теле HTTP-запроса допускается не для всех HTTP-методов. Например, тело HTTP-запроса всегда пустое, если используется метод GET.

Метод HTTP-запроса

Метод HTTP-запроса определяет, каким образом будет обрабатываться указанный HTTP-запрос, т. е. в каком-то смысле определяет его семантику. Поскольку HTTP-запросы могут иметь самый разнообразный смысл, то указание метода является важной частью построения HTTP-запроса. HTTP-запросы могут иметь следующие значения: запрос ресурса от сервера, создание или изменение ресурса на сервере, удаление ресурса на сервере и т. д.

Наиболее распространенными методами HTTP-запроса являются следующие типы методов (табл. 4.1).

Таблица 4.1

GET	Позволяет получить информацию от сервера, тело запроса всегда остается пустым
HEAD	Аналогичен GET, но тело ответа остается всегда пустым, позволяет проверить доступность запрашиваемого ресурса и прочитать HTTP-заголовки ответа
POST	Позволяет загрузить информацию на сервер, по смыслу изменяет ресурс на сервере, но зачастую используется и для создания ресурса на сервере, тело запроса содержит изменяемый/создаваемый ресурс
PUT	Аналогичен POST, но по смыслу занимается созданием ресурса, а не его изменением, тело запроса содержит создаваемый ресурс
DELETE	Удаляет ресурс с сервера

Кроме указанных методов HTTP, существует еще большое количество других методов, определенных в спецификации протокола HTTP. Однако, несмотря на это, браузерами зачастую используются только методы GET и POST. Тем не менее, другие прикладные приложения могут использовать HTTP-методы по своему усмотрению.

HTTP-ответ

HTTP-ответ генерируется веб-сервером в ответ на поступивший HTTP-запрос. По своей структуре он схож с HTTP-запросом, но имеет определенные отличия. Главное отличие содержится в первой строке. Вместо имени запрашиваемого ресурса и метода запроса в ней указывается статус ответа. Статус указывает на то, насколько успешно выполнился HTTP-запрос. Например, в случае, если документ найден на сервере и может быть выдан клиенту, то статус имеет значение «OK», которое говорит о том, что запрос выполнился успешно. Таким образом, первая строка HTTP-ответа может принимать значение «HTTP/1.1 200 OK». Однако могут появляться исключительные ситуации – например, документ отсутствует на сервере или у пользователя отсутствуют права на получение ресурса.

HTTP-заголовки в HTTP-ответе также являются важным элементом. Они характеризуют содержимое, которое передается клиенту. Например, в этих HTTP-заголовках может содержаться информация о типе содержимого (HTML-документ, изображение и т. д.), длине содержимого (размер в байтах), дате модификации, режиме кэширования и др.

Коды состояния

Код состояния является частью первой строки ответа сервера. Он представляет собой целое число из трех арабских цифр. Первая цифра указывает на класс состояния. За кодом ответа обычно следует отделенная пробелом поясняющая фраза на английском языке, которая разъясняет причину именно такого ответа.

В настоящее время выделено пять классов кодов состояния:

- 1xx Informational (Информационный). В этот класс выделены коды, информирующие о процессе передачи. В HTTP/1.0 сообщения с такими кодами должны игнорироваться. В HTTP/1.1 клиент должен быть готов принять этот класс сообщений как обычный ответ, но ничего отправлять серверу не нужно. Сами сообщения от сервера содержат только стартовую строку ответа и, если требуется, несколько специфичных для ответа полей заголовка;
- 2xx Success (Успех). Сообщения данного класса информируют о случаях успешного принятия и обработки запроса клиента. В зависимости от статуса сервер может еще передать заголовки и тело сообщения;
- 3xx Redirection (Перенаправление). Коды класса 3xx сообщают клиенту что для успешного выполнения операции необходимо сделать другой запрос (как правило по другому URI). Из данного класса пять кодов 301, 302, 303, 305 и 307 относятся непосредственно к перенаправлениям (редирект). Адрес, по которому клиенту следует произвести запрос, сервер указывает в заголовке Location;
- 4xx Client Error (Ошибка клиента). Класс кодов 4xx предназначен для указания ошибок со стороны клиента. При использовании всех методов, кроме HEAD, сервер должен вернуть в теле сообщения гипертекстовое пояснение для пользователя;
- 5xx Server Error (Ошибка сервера). Коды 5xx выделены под случаи неудачного выполнения операции по вине сервера. Для всех ситуаций, кроме использования метода HEAD, сервер должен включать в тело сообщения объяснение, которое клиент отобразит пользователю.

4.5. HTTPS

HTTPS (Hypertext Transfer Protocol Secure) – расширение протокола HTTP, поддерживающее шифрование. Данные, передаваемые по протоколу HTTPS, «упаковываются» в криптографический протокол SSL или TLS, тем самым обеспечивается защита этих данных. В отличие от HTTP, для HTTPS по умолчанию используется TCP-порт 443. Спецификация протокола определена в RFC 2818.

Как известно, существуют классические криптостойкие алгоритмы шифрования, которые шифруют данные на основе существующего ключа. Для шифрования и расшифровки данных используется один и тот же ключ – если кто-либо знает ключ к зашифрованной информации, то он может расшифровать ее. Ключ – это обычная последовательность бит определенной длины.

Чем больше длина ключа, тем сложнее взломать алгоритм шифрования. Таким образом, для того чтобы защитить свою информацию, необходимо хранить в секрете ключ шифрования.

В случае взаимодействия по протоколу HTTP, если необходимо защитить передаваемую информацию, используют дополнительно другой вид шифрования – асимметричный. В этом случае существует пара ключей – открытый и закрытый. С помощью открытого ключа можно только зашифровать информацию, а с помощью закрытого – расшифровать. Обычно при таком подходе закрытый ключ хранится в секрете, а открытый ключ является общедоступным. Однако асимметричный алгоритм работает медленнее, чем симметричный, поэтому его используют для первоначального обмена симметричными ключами.

4.6. ICMP

Протокол ICMP (Internet Control Message Protocol) используется хостами и маршрутизаторами для обмена управляющей информацией, например, об ошибках или о процессе начальной загрузки. ICMP рассматривается как самостоятельный прикладной протокол, хоть он и является неотъемлемой частью любой реализации протокола IP, поскольку управляет его работой.

Все ICMP-сообщения передаются с помощью IP-дейтаграмм, которые состоят их пяти частей:

- IP-заголовок;
- тип – это поле определяет тип ICMP-сообщения;
- код – это поле содержит текст ICMP-сообщения;
- контрольная сумма – это поле хранит сведения о целостности ICMP-сообщения;
- данные – это поле имеет переменную длину и содержит дополнительную информацию о сообщении.

Всего протоколом ICMP определено около дюжины типов сообщений. Каждое ICMP сообщение вкладывается в IP-пакет. Подробности о ICMP сообщениях и полное описание их формата – в RFC 1256, 1122, 792 и др. Наиболее часто используемые сообщения представлены в табл. 4.2.

Таблица 4.2

Тип сообщения	Сообщение	Описание
0	Echo Reply	Ответ на запрос отклика
3	Destination Unreachable	Дейтаграмма не может быть доставлена
4	Source Squench	Дейтаграмма уничтожена
8	Echo Request	Запрос отклика («спросить у хоста жив ли он»)
11	Time Exceeded	Значение поля «Время жизни» уменьшилось до нуля
12	Parameter Problem	Неправильный IP-заголовок
13	Timestamp Request	То же, что и ответ на запрос отклика, но с временной меткой

14	Timestamp Reply	То же, что запрос отклика, но с временной меткой

Некоторые ICMP-сообщения генерируются только маршрутизаторами, а остальные – и маршрутизаторами, и хостами. Обычно маршрутизаторы посылают ICMP-сообщения для отчета об ошибках. Если маршрутизатор не обрабатывает дейтограмму из-за недостатка ресурсов (например, памяти), то посылает ее отправителю ICMP-сообщение Source Squench. Если маршрутизатор получает пакет данных для неизвестного адресата, то посылает ICMP-сообщение Destination Unreachable. Хосты обычно посылают ICMP-сообщения для проверки качества связи и оценки времени отзыва. Для этого используется ICMP-сообщение Echo. Хост, получивший такое сообщение, должен отправить сообщение Echo reply.

Рассмотрим работу протокола на примере утилиты Ping – утилита для проверки соединений в сетях на основе TCP/IP, а также обиходное наименование самого запроса. Ping является одним из основных диагностических средств в сетях TCP/IP и входит в поставку всех современных сетевых операционных систем. Утилита была написана Майком Мууссом, ученым Исследовательской лаборатории баллистики США в декабре 1983 г.

Принцип действия утилиты ping построен на анализе времени задержки между моментом отправки удаленному узлу запроса по протоколу ICMP (сообщение Echo Request) и получением от этого узла ответа (сообщение Echo-Reply). Данный временной диапазон носит название RTT (Route Trip Time) и позволяет оценить скорость передачи информации, а также количество потерянных пакетов. На основе этих сведений можно сделать вывод о доступности удаленного узла и текущей нагрузке на используемые для соединения с ним каналы связи. Потеря ста процентов отправленных удаленному узлу пакетов может свидетельствовать о том, что данный узел недоступен, либо о выходе из строя промежуточного сетевого оборудования. Это может также означать, что какое-либо из осуществляющих маршрутизацию промежуточных устройств блокирует обработку ICMP-сообщений Echo.

4.7. SSH

SSH (Secure SHell – «безопасная оболочка») – это сетевой протокол прикладного уровня, обеспечивающий защищенную аутентификацию, соединение и безопасную передачу данных между хостами сети, путем шифрования, проходящего через него трафика, с возможной компрессией данных. Еще одной важной функциональной особенностью является возможность создания защищенных, шифрованных туннелей, для безопасной передачи через небезопасную среду (например, Интернет), других сетевых протоколов, также с возможностью сжатия трафика.

Существуют две версии протокола SSH, не совместимые между собой. Первая реализация протокола SSH-1 была разработана в 1995 г. исследователем Тату Улёненом из Технологического университета Хельсинки

(Финляндия). Вторая версия, SSH-2, выпущена в 1996 г. Спецификация протокола SSH-2 содержится в RFC 4251. В 2006 г. протокол SSH был принят ассоциацией IETF в качестве интернет стандарта.

Для работы по SSH нужен SSH-сервер и SSH-клиент. Сервер прослушивает соединения от клиентских машин и при установлении связи производит аутентификацию, после чего начинает обслуживание клиента. Клиент используется для входа на удаленную машину и выполнения команд.

Для соединения сервер и клиент должны создать пары ключей – открытых и закрытых – и обменяться открытыми ключами. Обычно используется также и пароль.

Архитектура протокола SSH

Протокол включает три основных компоненты.

Протокол транспортного уровня [SSH-TRANS] обеспечивает аутентификацию серверов, конфиденциальность и целостность. Для аутентификации сервера в SSH используется протокол аутентификации сторон на основе алгоритмов электронно-цифровой подписи RSA (криптографический алгоритм с открытым ключом) или DSA (алгоритм с использованием открытого ключа для создания электронной подписи). Этот протокол может также обеспечивать сжатие информации. Транспортный уровень работает в основном с использованием соединений TCP/IP, но может быть реализован и на базе иных потоков данных с гарантированной доставкой.

Протокол аутентификации пользователей [SSH-USERAUTH] используется на серверах для проверки полномочий клиентов. Для аутентификации клиента также может использоваться алгоритм электронно-цифровой подписи RSA или DSA, но допускается также аутентификация с помощью пароля и даже ip-адреса хоста. Аутентификация по паролю наиболее распространена; она безопасна, так как пароль передается по зашифрованному виртуальному каналу. Аутентификация по ip-адресу небезопасна, эту возможность чаще всего отключают. Протокол [SSH-USERAUTH] работает на основе протокола транспортного уровня.

Протокол соединений [SSH-CONNECT] обеспечивает мультиплексирование шифрованного туннеля в несколько логических каналов и работает поверх протокола аутентификации пользователей.

Сообщения протокола SSH

Пакеты SSH используют номера сообщений от 1 до 255. Эти номера распределены между различными компонентами:

- протокол транспортного уровня:

1 – 19 – базовые сообщения транспортного уровня (например, disconnect, ignore, debug и т. п.),

20 – 29 – согласование алгоритма,

30 – 49 – сообщения, связанные с обменом ключами (допускается совпадение номеров для разных методов аутентификации);

• протокол аутентификации пользователей:

50 – 59 – базовые сообщения протокола аутентификации,

60 – 79 – сообщения, связанные с методом аутентификации (допускается совпадение номеров для различных методов);

• протокол соединений:

80 – 89 – базовые сообщения протокола,

90 – 127 – сообщения, связанные с каналом;

• зарезервировано для клиентских протоколов:

128 – 191 – резерв;

• локальные расширения:

192 – 255 – локальные расширения.

4.8. NTP

Сетевой протокол задания времени NTP (Network Time Protocol) служит для осуществления синхронизации работы различных процессов в серверах и программах клиента. NTP использует для своей работы протокол UDP.

NTP – один из старейших используемых протоколов, разработан Дэвидом Л. Миллсом из университета Дэлавера в 1985 г. и в настоящее время продолжает совершенствоваться. Текущая версия протокола – NTP 4, описана в RFC 5905.

Протокол NTP создан с целью определения трех величин: смещения часов (clock offset), RTT (Round Trip Time) – время от момента посылки запроса до момента получения ответа и дисперсии; все они вычисляются по отношению к выбранным эталонным часам. Смещение часов определяет поправку, которую необходимо внести в показания местных часов, чтобы результат совпал с показанием эталонных часов. Дисперсия характеризует максимальную ошибку локальных часов по отношению к эталонным.

Клиент посылает NTP-сообщения одному или нескольким серверам и обрабатывает отклики по мере их получения. Сервер изменяет адреса и номера портов, переписывает содержимое некоторых полей, заново вычисляет контрольную сумму и немедленно посылает отклик. Информация, заключенная в сообщение NTP, позволяет клиенту определить показания часов сервера по отношению к часам клиента и соответствующим образом скорректировать рабочие параметры местных часов. Кроме того, эта информация содержит данные, позволяющие оценить точность и надежность часов сервера и выбрать наилучший эталон времени.

NTP использует иерархическую систему «часовых уровней» (stratum). Уровень 1 синхронизирован с высокоточными часами, например, с системой GPS, ГЛОНАСС (Единая государственная шкала времени РФ) или атомным эталоном времени. Уровень 2 синхронизируется с одной из машин уровня 1 и так далее.

Наиболее широкое применение протокол NTP находит для реализации серверов точного времени. Для достижения максимальной точности пред-

почтительна постоянная работа программного обеспечения NTP в режиме системной службы.

4.9. NFS

NFS (Network File System) – протокол сетевого доступа к файловым системам, первоначально разработан Sun Microsystems в 1984 г. Позволяет подключать (монтировать) удаленные файловые системы через сеть, описан в RFC 1094, RFC 1813, RFC 3530 и RFC 5661. Текущая версия протокола – NFS v4.1.

NFS предоставляет клиентам прозрачный доступ к файлам и файловой системе сервера. В отличие от FTP, протокол NFS осуществляет доступ только к тем частям файла, к которым обратился процесс, и основное достоинство его в том, что он делает этот доступ прозрачным. Это означает, что любое приложение клиента, которое может работать с локальным файлом, с таким же успехом может работать и с NFS файлом, без каких либо модификаций самой программы.

Протокол NFS использует клиент-серверную модель взаимодействия. В ранних версиях NFS для транспортирования данных использовался UDP-протокол, в современных – используется TCP. NFS клиенты получают доступ к файлам на NFS сервере путем отправки RPC-запросов на сервер.

Компоненты NFS

Реализация NFS состоит из нескольких компонентов. Некоторые из них локализованы либо на сервере, либо на клиенте, а некоторые используются и тем и другим.

Протокол NFS определяет набор запросов (операций), которые могут быть направлены клиентом к серверу, а также набор аргументов и возвращаемые значения для каждого из этих запросов.

Протокол удаленного вызова процедур RPC (Remote Procedure Call) определяет формат всех взаимодействий между клиентом и сервером. Каждый запрос NFS посылается как пакет RPC.

Внешнее представление данных (XDR–External Data Representation) обеспечивает машинно-независимый метод кодирования данных для пересылки через сеть. Все запросы RPC используют кодирование XDR для передачи данных.

Программный код сервера NFS отвечает за обработку всех запросов клиента и обеспечивает доступ к экспортируемым файловым системам.

Программный код клиента NFS реализует все обращения клиентской системы к удаленным файлам путем посылки серверу одного или нескольких запросов RPC.

Протокол монтирования MOUNT определяет семантику монтирования и размонтирования файловых систем NFS.

Менеджер блокировок сети (NLM – Network Lock Manager) и монитор состояния сети (NSM– Network Status Monitor) вместе обеспечивают средства

для блокировки файлов в сети. Эти средства, хотя формально не связаны с NFS, можно найти в большинстве реализаций NFS.

4.10. Практическая работа «Модель OSI»

Порядок выполнения работы

1. На РМ запустить Wireshark (локально) со следующим фильтром: ip.addr==IP_адрес_РМУ and not tcp.port==2000 and not ip.addr==IP_адрес_РМП, например, ip.addr==172.16.117.3 and not tcp.port==2000 and not ip.addr==172.16.117.99;

2. Преподаватель запускает скрипт, который производит обращение с РМ пользователя к сервисам полигона СОТСБИ OSI по различным протоколам, обращение осуществляется последовательно к четырем сервисам. При этом на рабочем столе РМ пользователя появляется диалоговое окно содержащее сообщение «Выполняется обращение к сервису»;

3. По окончании работы скрипта (появление четырех сообщений «Выполняется обращение к сервису» в диалоговом окне) необходимо остановить мониторинг сетевого трафика;

4. Проанализировать трейс, полученный с помощью Wireshark:

- определить четыре сервиса, к которым производилось обращение;
- для каждого сервиса определить параметры, указанные в табл. 4.3.

Таблица 4.3

Название сервиса	Параметры
DNS	Определить: • назначение обращения к сервису DNS; • уровень DNS по модели OSI; • транспортный протокол для сообщений DNS; • IP-адрес и порт источника запроса сервиса; • IP-адрес и порт получателя запроса сервиса; • тип запроса и ответ на запрос
FTP	Определить: • назначение обращения к сервису FTP; • уровень FTP по модели OSI; • транспортный протокол для сообщений FTP; • IP-адрес и порт источника запроса сервиса; • IP-адрес и порт получателя запроса сервиса; • этапы установления управляющего соединения (проанализировать команды и отклики на них); • тип, название передаваемого файла и режим передачи

Продолжение табл. 4.3

Название сервиса	Параметры
TFTP	Определить: • назначение обращения к сервису TFTP;

Название сервиса	Параметры
	• уровень TFTP по модели OSI; • транспортный протокол для сообщений TFTP; • IP-адрес и порт источника запроса сервиса; • IP-адрес и порт получателя запроса сервиса; • назначение переданных сообщений • количество блоков передаваемых данных
HTTP	Определить: • назначение обращения к сервису HTTP; • уровень HTTP по модели OSI; • транспортный протокол для сообщений HTTP; • IP-адрес и порт источника запроса сервиса; • IP-адрес и порт получателя запроса сервиса
HTTPS	Определить: • назначение обращения к сервису HTTPS; • уровень HTTPS по модели OSI; • транспортный протокол для сообщений HTTPS; • Используемый криптографический протокол; • IP-адрес и порт источника запроса сервиса; • IP-адрес и порт получателя запроса сервиса
PING	Определить: • назначение обращения к сервису PING; • уровень PING по модели OSI; • IP-адрес и порт источника запроса сервиса; • IP-адрес и порт получателя запроса сервиса
SSH	Определить: • назначение обращения к сервису SSH; • основные этапы обращения к сервису; • уровень SSH по модели OSI; • транспортный протокол для сообщений SSH; • IP-адрес и порт источника запроса сервиса; • IP-адрес и порт получателя запроса сервиса
NTP	Определить: • назначение обращения к сервису NTP; • уровень NTP по модели OSI; • транспортный протокол для сообщений NTP; • IP-адрес и порт источника запроса сервиса; • IP-адрес и порт получателя запроса сервиса

Окончание табл. 4.3

Название сервиса	Параметры
NFS	Определить: • назначение обращения к сервису NFS; • уровень NFS по модели OSI; • IP-адрес и порт источника запроса сервиса; • IP-адрес и порт получателя запроса сервиса; • название загруженного файла с NFS сервера; • адрес NFS сервера

5. Отчет должен содержать названия четырех сервисов, которые были определены в процессе выполнения практической работы, и их параметры (табл. 4.3).

СПИСОК ЛИТЕРАТУРЫ

1. *Гольдштейн, А. Б.* Softswitch / А. Б. Гольдштейн, Б. С. Гольдштейн. – СПб. : БХВ-Санкт-Петербург, 2006.

2. *Гольдштейн, Б. С.* Сети связи / Б. С. Гольдштейн, Н. А. Соколов, Г. Г. Яновский. – СПб. : БХВ-Санкт-Петербург, 2010.

3. Тестирование и анализ телекоммуникационных протоколов [Электронный ресурс] / И. М. Ехриель, Р. Д. Рерле. ФГУП ЛОНИИС – Режим доступа : www.niits.ru/public/2002/200221.pdf, свободный.

4. Анализаторы протоколов для NGN [Электронный ресурс] / Ю. С. Исаченко ; Электрон. журн. // Вестник связи. – 2006. № 9. – Режим доступа : http://www.niits.ru/public/2006/2006-039.pdf, свободный. – Режим доступа к журн.: http://www.vestnik-sviazy.ru

5. RFC 1034 DOMAIN NAMES – CONCEPTS AND FACILITIES

6. RFC 1035 DOMAIN NAMES – IMPLEMENTATION AND SPECIFICATION

7. RFC 959 FILE TRANSFER PROTOCOL (FTP)

8. RFC 1350 The TFTP Protocol (Revision 2)

9. RFC 1945 Hypertext Transfer Protocol -- HTTP/1.0

10. RFC 2818 HTTP Over TLS

11. RFC 4251 The Secure Shell (SSH) Protocol Architecture

12. RFC 5905 Network Time Protocol Version 4: Protocol and Algorithms Specification

13. RFC 1094 NFS: Network File System Protocol Specification

14. RFC 1813 NFS Version 3 Protocol Specification

15. RFC 3530 NFS version 4 Protocol

16. RFC 5661 Network File System (NFS) Version 4 Minor Version 1 Protocol

17. RFC 1122 Requirements for Internet Hosts – Communication Layers

18. RFC 1256 ICMP Router Discovery Messages

19. RFC 792 INTERNET CONTROL MESSAGE PROTOCOL

20. WireShark [Электронный ресурс] / Режим доступа: http://www.wireshark.org/, свободный.

21. http://www.skri.sut.ru

22. http://www.niits.ru

23. http://www.sotsbi.spb.ru

24. http://www.wireshark.org/

Гольдштейн Борис Соломонович
Гойхман Вадим Юрьевич
Столповская Юлия Владимировна

СЕТЕВЫЕ АНАЛИЗАТОРЫ IP СЕТЕЙ

Учебное пособие

Редактор *Л. А. Медведева*

Верстка *Н. А. Ефремовой*

План 2013 г., п. 1

Подписано к печати 05.07.2013
Объем 3,5 усл.-печ. л. Тираж 115 экз. Заказ 343
РИЦ СПбГУТ. 191186 СПб., наб. р. Мойки, 61
Отпечатано в СПбГУТ