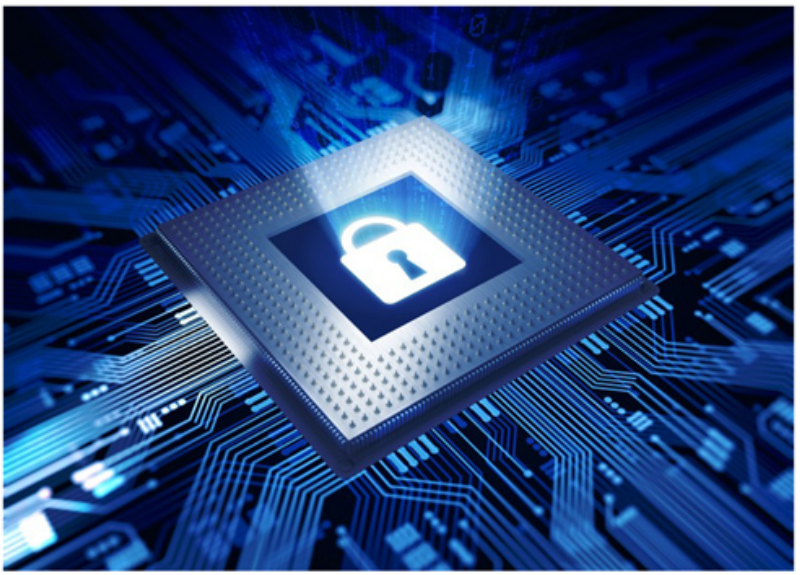
УНИВЕРСИТЕТ ИТМО

А.С. Куракин, А.В. Жуковский, Е.И. Зозуля

«ЭКСПЕРТНЫЕ СИСТЕМЫ КОМПЛЕКСНОЙ ОЦЕНКИ БЕЗОПАСНОСТИ ОБЪЕКТА»

методические указания
к выполнению лабораторных работ по дисциплине



Санкт-Петербург
2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

А.С. Куракин, А.В. Жуковский, Е.И. Зозуля

МЕТОДИЧЕСКИЕ УКАЗАНИЯ к выполнению лабораторных работ по дисциплине «ЭКСПЕРТНЫЕ СИСТЕМЫ КОМПЛЕКСНОЙ ОЦЕНКИ БЕЗОПАСНОСТИ ОБЪЕКТА»

УЧЕБНОЕ-МЕТОДИЧЕСКОЕ ПОСОБИЕ

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург

2016

А.С. Куракин, А.В. Жуковский, Е.И. Зозуля. Методические указания к выполнению лабораторных работ по дисциплине «Экспертные системы комплексной оценки безопасности объекта». Учебное пособие / Под ред. профессора, д.т.н. Ю.А. Гатчина. – СПб: НИУ ИТМО, 2016. – 25 с.

Учебно-методическое пособие подготовлено на кафедре проектирования и безопасности компьютерных систем для студентов НИУ ИТМО, обучающихся по направлению 10.04.01 «Информационная безопасность» и 11.04.03 «Конструирование и технология электронных средств»

Рекомендовано к печати по решению Ученого совета факультета КТиУ от 10.11.15 (протокол №9)

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года СПб НИУ ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель СПб НИУ ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Программное обеспечение (ПО) компьютерных сетей - сложное в отладке вследствие затруднений при анализе корректности передаваемых и принимаемых данных в виде циркулирующих по сети пакетов; проблемы имеют место даже при использовании стандартных протоколов, методов и процедур обмена данными. Одним из путей создания надежных сетевых программ является применение проверенных систем разработки, основанных на поддержании технологией объектно-ориентированного программирования (ООП) механизма сетевых сокетов. В сложных случаях необходим тщательный анализ трафика сети с применением, например, снифферов (анализаторов содержимого трафика сети).

Sniffer (от англ. to sniff – нюхать) – это сетевой анализатор трафика, программа или программно-аппаратное устройство, предназначенное для перехвата и последующего анализа, либо только анализа сетевого трафика, предназначенного для других узлов. Он может быть установлен на маршрутизаторе (шлюзе, при этом устройство перехватывает проходящий через интерфейсы этого шлюза трафик) и на конечном узле сети (перехватывается трафик данного сегмента сети).

Перехват трафика может осуществляться:

- обычным «прослушиванием» сетевого интерфейса (метод эффективен при использовании в сегменте концентраторов (хабов) вместо коммутаторов (свичей), в противном случае метод малоэффективен, поскольку на сниффер попадают лишь отдельные фреймы);
- подключением сниффера в разрыв канала;
- ответвлением (программным или аппаратным) трафика и направлением его копии на сниффер;
- через анализ побочных электромагнитных излучений и восстановление таким образом прослушиваемого трафика;
- через атаку на канальном (2-й) или сетевом (3-й) уровне, приводящую к перенаправлению трафика жертвы или

всего трафика сегмента на сниффер с последующим возвращением трафика в надлежащий адрес.

В начале 1990-х широко применялся хакерами для захвата пользовательских логинов и паролей. Широкое распространение хабов позволяло захватывать трафик без больших усилий в больших сегментах сети.

Снифферы применяются как в благих, так и в деструктивных целях. Анализ прошедшего через сниффер трафика, позволяет:

- отслеживать сетевую активность приложений;
- отлаживать протоколы сетевых приложений;
- локализовать неисправность или ошибку конфигурации;
- обнаружить паразитный, вирусный и закольцованный трафик, наличие которого увеличивает нагрузку сетевого оборудования и каналов связи;
- выявить в сети вредоносное и несанкционированное ПО, например, сетевые сканеры, флудеры, троянские программы, клиенты пиринговых сетей и другие;
- перехватить любой незашифрованный (а порой и зашифрованный) пользовательский трафик с целью узнавания паролей и другой информации.

Поскольку в «классическом» снифферe анализ трафика происходит вручную, с применением лишь простейших средств автоматизации (анализ протоколов, восстановление TCP-потока), то он подходит для анализа лишь небольших его объёмов.

Снизить угрозу сниффинга пакетов можно с помощью таких средств, как:

Аутентификация - процедура проверки подлинности. Например,
проверка подлинности пользователя путём сравнения введённого им пароля с паролем, сохранённым в базе данных пользователей; подтверждение подлинности электронного письма путём проверки цифровой подписи письма по открытому ключу отправителя; проверка контрольной суммы файла на соответствие сумме, заявленной автором этого файла.

Криптография -наука о методах обеспечения конфиденциальности (невозможности прочтения информации

посторонним), целостности данных (невозможности незаметного изменения информации), аутентификации (проверки подлинности авторства или иных свойств объекта), а также невозможности отказа от авторства.

Антиснифферы - работающие в сети передачи данных аппаратные или программные средства, которые помогают распознать снифферы.

Коммутируемая инфраструктура - способом борьбы со сниффингом пакетов в сетевой среде является создание коммутируемой инфраструктуры.

Анализ вышеприведенной перехваченной информации в реальном случае (анализ многих тысяч пакетов) чрезвычайно затруднен, поэтому разработаны специальные системы для фильтрации и компоновки перехваченного трафика (например, визуализация имеющегося HTTP-трафика WEB-браузером). Одной из несложных систем подобного рода является комплект SpyNet, включающий два модуля - CaptureNet (собственно перехват трафика, рисунок 1) и PeepNet (фильтрация пакетов и визуальное представление содержимого, рисунок 2). CaptureNet определяет MAC- и IP-адреса сетевой карты и по нажатию Start capture переводит ее в выбранный режим (задается набором флажков Hardware Filter); при этом в верхнем правом фрейме окна выводятся параметры каждого захваченного пакета (показываются время, MAC- и IP-источника и адресата, используемый протокол, порт и др.), в правом нижнем фрейме можно просмотреть содержимое выбранного пакета (восьмеричное), при работе CaptureNet записывает отфильтрованные пакеты в файл (по умолчанию CAPTURE.CAP). Заметим, что фактический сброс информации в этот файл происходит в момент превышения объема буфера (задается выбором вариантов меню Capture|Settings) или при остановке захвата пакетов нажатием Stop capture).

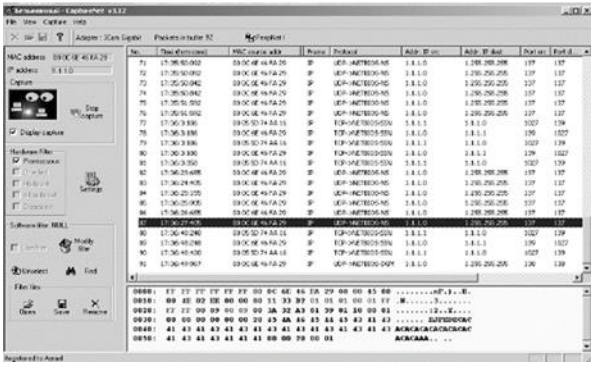


Рисунок 1. Главное окно утилиты CaptureNet v3.12

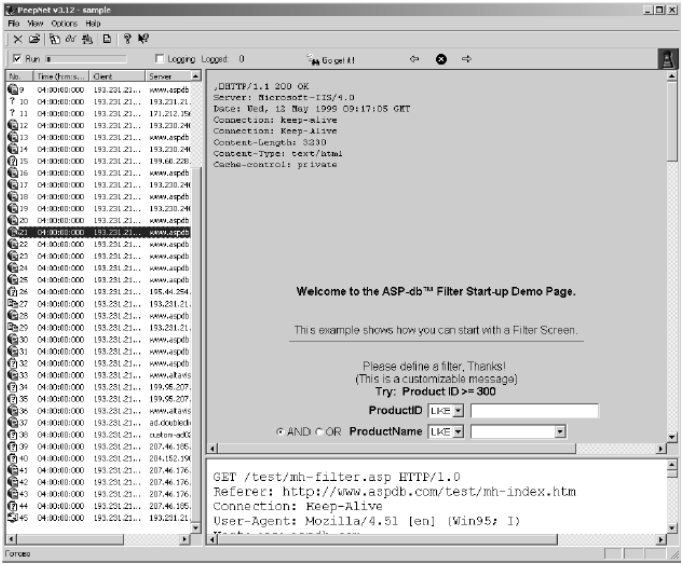


Рисунок 2.Окно визуализации перехваченного сообщения утилитой PeepNet v3.12

Ниже приведен максимально упрощенный исходный текст небольшой программы - сниффера для Unix/Linux (единственный параметр командной строки – имя сетевого интерфейса карты).

```
/* сниффер пишет на stdout все, что захватывает */
#include <sys/socket.h>
#include <netinet/in.h>
#include <net/if.h>
#include <unistd.h>
#include <signal.h>
#include <stdio.h>

static volatile int done;

void
handler(int signum)
{
done = 1;
} /* конец обработчика сигнала HANDLER */

main(int argc, char **argv)
{
char buff[0x10000];
struct ifreq ifr;
int s, n;
if (argc < 2)
{
fprintf(stderr, "Usage: %s <interface>\n", argv[0]);
return 1;
}
s = socket(PF_INET, SOCK_PACKET, htons(0x0003));
if (s == -1)
{
```

```
perror("socket");
return 1;
}
strcpy(ifr.ifr_name, argv[1]);
if (ioctl(s, SIOCGIFFLAGS, &ifr) < 0)
{
perror("ioctl(SIOCGIFFLAGS)");
return 1;
}
/*************************************************
*****************************************/
ifr.ifr_flags |= IFF_PROMISC; /* установка режима
перехвата */
/* ВСЕХ пакетов, поступающих */
/* на сетевую карту */
if (ioctl(s, SIOCSIFFLAGS, &ifr) < 0)
{
perror("ioctl(SIOCGIFFLAGS)");
return 1;
}
signal(SIGINT, handler);
puts("starting capturing:\n");
fflush(stdout);
for (done = 0; !done; )
{
n = read(s, buff, sizeof(buff)); /* считываем перехваченный
трафик в буфер buff */
if ( n != -1 )
write(STDOUT_FILENO, buff, n);
}
ifr.ifr_flags &= ~IFF_PROMISC; /* снятие режима перехвата
всех пакетов */
if (ioctl(s, SIOCSIFFLAGS, &ifr) < 0)
{
```

```
perror("ioctl(SIOCGIFFLAGS)");
return 1;
}
close(s);
printf("Finished\n");
return 0;
} /* конец MAIN */
```

Один из примеров выдачи (определенным образом скомпонованной) информации сниффером приведен ниже; распечатка содержимого перехваченного пакета (датаграммы) состоит из разделенных двоеточием трех колонок: формат пакета-носителя, имя поля, содержимое поля в десятичном и восьмеричном представлении. Этот пакет содержит 14-байтовый заголовок Ethernet, 20-байтовый IP-заголовок, 20-байтовый TCP-заголовок, заголовок HTTP, оканчивающийся двумя подряд CRLF (0D 0A 0D 0A) и далее собственно данные прикладного уровня (WEB-трафик по протоколу HTTP версии 1.1, [2,4]):

```
ETHER: Destination address: 0000BA5EBA11
ETHER: Source address: 00A0C9B05EBD
ETHER: Frame Length: 1514 (0x05EA)
ETHER: Ethernet Type: 0x0800 (IP)
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 = 1500 (0x5DC)
IP: Identification = 7652 (0x1DE4)
IP: Flags Summary = 2 (0x2)
IP:.......0 = Last fragment in datagram
8
IP:.....1 = Cannot fragment datagram
IP: Fragment Offset = 0 (0x0) bytes
```

IP: Time to Live = 127 (0x7F)
IP: Protocol = TCP — Transmission Control
IP: Checksum = 0xC26D
IP: Source Address = 10.0.0.2 IP: Destination Address = 10.0.1.201 TCP: Source Port = Hypertext Transfer Protocol
TCP: Destination Port = 0x0775
TCP: Sequence Number = 97517760 (0x5D000C0)TCP: Acknowledgement Number = 78544373 (0x4AE7DF5)
TCP: Data Offset = 20 (0x14)
TCP: Reserved = 0 (0x0000)
TCP: Flags = 0x10:.A....
TCP:..0..... = No urgent data
TCP:...1.... = Acknowledgement field significant
TCP:....0... = No Push function
TCP:.....0.. = No Reset
TCP:......0. = No Synchronize
TCP:.......0 = No Fin
TCP: Window = 28793 (0x7079)
TCP: Checksum = 0x8F27
TCP: Urgent Pointer = 0 (0x0)
HTTP: Response (to client using port 1909)
HTTP: Protocol Version = HTTP/1.1
HTTP: Status Code = OK
HTTP: Reason = OK
...и так далее...

Постепенно из инструментов, предназначенных только для диагностики, снифферы постепенно превратились в средства для исследований и обучения. Например, они постоянно используются для изучения динамики и взаимодействий в сетях. В частности, они позволяют легко и наглядно изучать тонкости сетевых протоколов. Наблюдая за данными, которые посылает протокол, вы можете глубже понять его функционирование на практике, а заодно увидеть, когда некоторая конкретная реализация работает не в соответствии со спецификацией.

На сегодняшний момент существует достаточно большое количество хороших реализаций снифферов, подробнее с которыми мы ознакомимся ниже. Некоторое из них:

• Tcpdump (http://www.tcpdump.org/) – консольный вариант сниффера. Портирован почти подо все наиболее распространенные ОС;

• Wireshark (http://www.wireshark.org/) до недавнего момента был известен под названием Ethreal;

• WinDump (http://www.winpcap.org/windump);

• IP Sniffer и др.

Программа Wireshark является одной из самых удобных реализаций снифферов. Портирована на большое количество платформ. Распространяется абсолютно бесплатно. На лабораторных работах мы будем использовать данный сниффер для изучения и анализа сетевых протоколов.

Базовый принцип работы снифферов

Давайте рассмотрим с вами рисунок 3. На нем изображена схематично структура сетевой подсистемы ОС. Вся базовая инфраструктура реализована в виде драйверов и работает в режиме ядра. Пользовательские процессы и реализации прикладных протоколов, в частности интерфейс сниффера работают в пользовательском режиме.

На рисунке 3 отображены 2 пользовательских процесса («сетевой процесс 1» и «сетевой процесс 2»).

Основными компонентами сниффера являются: драйвер для захвата пакетов (libpcap драйвер), интерфейсная библиотека (libpcap) и интерфейс пользователя (Wireshark). Библиотека libpcap (реализация под ОС Windows носит название WinPcap - http://www.winpcap.org) – универсальная сетевая библиотека, самостоятельно реализующая большое количество сетевых протоколов и работающая непосредственно с NDIS (Network Driver Interface Specification) драйверами сетевых устройств. На базе данной библиотеки реализовано большое количество сетевых программ, в частности сниффер Wireshark.

Сниффер использует библиотеку в режиме «захвата» пакетов, т.е. может получать копию ВСЕХ данных проходящих через драйвер сетевого интерфейса. Изменения в сами данные не вносятся!

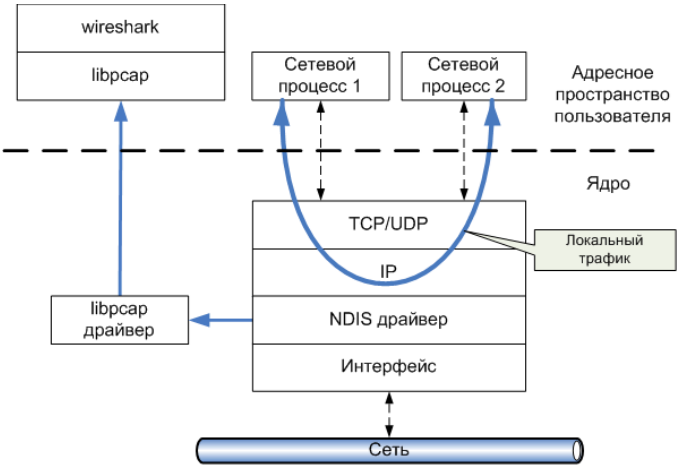


Рисунок. 3. Принцип «захвата» сниффером сетевого трафика

Основной нюанс использования сниффера заключается в том, что он не позволяет производить анализ локального трафика, т.к. он не проходит через драйвер сетевого устройства.Т.е., если вы захотите проанализировать сниффером трафик между 2-ми сетевыми процессами на локальной машине (например, ftp-сервер и ftp-клиент), то у вас ждет разочарование. Однако, например, при использовании виртуальных машин, сниффер будет работать без проблем, т.к. виртуальные машины эмулируют реальную среду и сетевые адаптеры, поэтому трафик идет через драйвер, как и в нормальной ситуации при взаимодействии с другими физическими сетевыми машинами.

Сниффер может анализировать только то, что проходит через его сетевую карту. Внутри одного сегмента сети, все пакеты рассылаются всем машинам, из-за этого возможно перехватывать чужую информацию. Использование коммутаторов (switch, switch-hub) и их грамотная конфигурация уже является защитой от прослушивания. Между сегментами информация передаётся через коммутаторы. Коммутация пакетов - форма передачи, при которой данные, разбитые на

отдельные пакеты, могут пересылаться из исходного пункта в пункт назначения разными маршрутами. Так что если кто-то в другом сегменте посылает внутри его какие-либо пакеты, то в ваш сегмент коммутатор эти данные не отправит.

Также к недостаткам большинства снифферов стоит отнести и тот факт, что, позволяя анализировать трафик, проходящий через сетевой интерфейс, они не могут указать, какое именно приложение генерирует или получает его. Это объясняется те, что информация об этом хранится на сетевом (например, IP) уровне сетевого стека, а большинство снифферов использует собственную реализацию стека протоколов (например, библиотеку WinPcap), которая (как уже было показано) работает непосредственно с драйверами устройств.

Стоит также отметить, что снифферы вносят дополнительную нагрузку на процессор, т.к. могут обрабатывать достаточно объемный сетевой трафик, в особенности для высокоскоростных соединений (Fast Ethernet, Gigabit Ethernet и др.).

Единственным решением, препятствующим сниффингу, является шифрование. В Song, Dug. "dsniff FAQ" для dsniff советуют не допускайте использования фирменных небезопасных прикладных протоколов или унаследованных протоколов, передающих данные явным образом в ваших сетях". Это очень важный совет. Замена небезопасных протоколов (таких как telnet) на их надежные шифрованные аналоги (такие как ssh) представляется серьезным барьером от перехвата. Замена всех небезопасных протоколов в большинстве случаев маловероятна.

Вместо прекращения использования протоколов, передающих данные явным образом, остается только одна возможность - шифрование всего сетевого трафика на 3 уровне, используя IPSec (Taylor, Laura."Understanding IPSec". June 2002). Осуществляя шифрование на 3 уровне, возможно продолжать использовать небезопасные протоколы, поскольку все данные будут инкапсулированы IPSec и зашифрованы при передаче по сети.

Таким образом, унаследованные приложения, которые используют старые протоколы, не пострадают. IPSec полностью прозрачен для приложений и пользователей. Это открытый

стандарт, поддерживаемый многими вендорами, включая Microsoft и Cisco. Кроме того, многие реализации Unix поддерживают IPSec.

Легкая настройка IPSec в Win'2k/XP дополнительно увеличивает его доступность.

Осуществление технологии шифрования на 3 уровне, таких как IPSec решает проблему снифинга полностью. Масштабируемость, распространенность, доступность IPSec выделяет его как прагматическое решение проблемы перехвата сетевого трафика.

Теперь подробнее рассмотрим реализаций снифферов:

Tcpdump - утилита UNIX (есть клон для Windows), позволяющая перехватывать и анализировать сетевой трафик, проходящий через компьютер, на котором запущена данная программа.

Для выполнения программы требуется наличие прав суперпользователя и прямой доступ к устройству (так, например, запуск из Jail во FreeBSD невозможен).

Основные назначения tcpdump: отладка сетевых приложений; отладка сети и сетевой конфигурации в целом.

Программа состоит из двух основных частей: части захвата пакетов (обращение к библиотеке, libpcap (Unix) или pcap (Windows)) и части отображения захваченных пакетов (которая на уровне исходного кода является модульной и для поддержки нового протокола достаточно добавить новый модуль).

Часть захвата пакетов (при запуске) передаёт «выражение выбора пакетов» (идущее после всех параметров командной строки) напрямую библиотеке захвата пакетов, которая проверяет выражение на синтаксис, компилирует его (во внутренний формат данных), а затем копирует во внутренний буфер программы сетевые пакеты, проходящие через выбранный интерфейс и удовлетворяющие условиям в выражении.

Часть отображения пакетов выбирает захваченные пакеты по одному из буфера, заполняемого библиотекой, и выводит их (в воспринимаемом человеком виде) на стандартный вывод построчно, в соответствии с заданным (в командной строке) уровнем детальности.

Если задан подробный вывод пакетов, программа проверяет для каждого сетевого пакета, имеется ли у неё модуль расшифровки данных, и, в случае наличия, соответствующей подпрограммой извлекает (и отображает) тип пакета в протоколе или передаваемые в пакете параметры.

Теперь рассмотрим кроссплатформенность, ведь изначально программа tcpdump была разработана для UNIX-подобных систем, позже - портирована на другие системы.

Для Windows в настоящее время известны:

tcpdump для Windows - коммерческая реализация, в виде одного файла tcpdump.exe

WinDump - реализация с открытым кодом, требующая установки библиотеки WinPcap (свободное ПО).

WinDump - определяет выбор отображаемых сетевых пакетов. Выдаются только те объекты, которые соответствуют выражению; если выражения не заданы, отображаться будут все пакеты. Выражение windump состоит из одной или нескольких директив, называемых примитивами, которые, в свою очередь, состоят из идентификатора и следующего за ним квалификатора.

Ниже приведены назначения типы квалификаторы:

Определяет, к чему относится идентификатор, заданный как имя или номер. Возможными типами служат host, net и port. Например, host foo, net 128.3 или port 20

Определяет направление трафика от определенного идентификатора. Возможными направлениями служат src; dst; src or dst и src and dst (src обозначает исходный адрес, dst - целевой)

Протокол windump позволяет определить протокол для фильтрации. Возможными протоколами являются ether, fddi, tr, ip, ipv6, arp, rarp, decnet, tcp и udp. Если протокол не задан, то допустимы все протоколы, совместимые с остальной частью выражения. При помощи фильтров с этим квалификатором можно определить, какая машина делает чрезмерное количество arp-запросов, или для отбрасывания на фильтре udp-запросов, которых немало во многих сетях, так как DNS использует udp

Ниже описаны комбинации примитивов:

dst host - показывает только трафик, адресованный хосту, который может быть задан IP-адресом или именем

src host - оказывает только трафик, исходящий из хоста

host – показывает, как исходящий, так и входящий трафик хоста

ether dst Ethernet - показывает трафик, предназначенный для указанного Ethernet-хоста, который может быть задан либо именем, либо MAC-адресом

ether src Ethernet - оказывает трафик, исходящий из Ethernet-хоста

ether host Ethernet – показывает, как исходящий, так и входящий трафик Ethernet-хоста

gateway - оказывает любой трафик, использующий хост в качестве шлюза. Иными словами, трафик, переправляемый с хоста. Так происходит, когда IP-адрес отправителя или получателя не соответствует Ethernet-адресу хоста. Данную возможность целесообразно использовать, когда необходимо отследить весь трафик, проходящий через Интернет-шлюз или некоторый конкретный маршрутизатор

dst net - фильтрует трафик, предназначенный для конкретной сети, заданной в нотации 0.0.0.0. Аналогично ether dst Ethernet-хост за исключением того, что это может быть значительно больше, чем один хост.

src net - фильтрует сеть отправителя

net - То же, что и две предыдущие инструкции, но трафик разрешен как в заданную сеть, так и из нее

net mask - сопоставляется с трафиком в заданную сеть или из нее, с указанной маской сети. Применяется для задания точного размера сети с шагом меньше, чем класс C. В этой комбинации допускается использование примитивов src и dst для указания направления потоков данных

net - сопоставляется с трафиком с сетевыми адресами из указанной сети и заданным числом бит в маске сети. Аналогична предыдущей комбинации

dst port - фильтрует трафик TCP и UDP с заданным целевым портом. Здесь можно также специфицировать тип перехватываемого трафика, TCP или UDP. По умолчанию отображается трафик обоих типов

src port - то же, что и предыдущая комбинация, только перехватывается трафик с заданным исходным портом

less - длина отображает пакеты с длиной, меньшей или равной заданной. Допустима также комбинация len <= длина

greater - же, что и предыдущая комбинация, только перехватывается трафик с длиной пакетов больше или равной указанной

ip proto - перехватывает трафик заданного протокола. Допустимыми протоколами служат icmp, icmpv6, igmp, igrp, pim, ah, esp, vrrp, udp и tcp. Имена tcp, udp и icmp должны помещаться между двумя обратными косыми чертами, чтобы они не читались как ключевые слова. Пример: ip proto \tcp;

ip6 proto - аналогично предыдущей комбинация, но для пакетов и типов IPv6

ip6 protochain - ищет пакеты IPv6, имеющие заголовок указанного протокола

ip protochain - то же, что и выше, но для пакетов IPv4

ip broadcast - идентифицирует только широковещательный трафик, то есть трафик, имеющий все нули или все единицы в поле целевого адреса

ether multicast - регистрирует вещательные пакеты Ethernet

ip multicast - регистрирует пакеты IP

ip6 multicast - регистрирует пакеты IPv6

ether proto - отображает трафик, который имеет указанный тип протокола Ethernet. Допустимыми именами протоколов служат ip, ipv6, arp, rarp, atalk, aarp, decnet, sca, lat, mopdl, moprc, iso, stp, ipx и netbeui. Эти имена являются также идентификаторами, поэтому они должны быть экранированы с помощью обратных косых черт

decnet src - перехватывает трафик DECnet с исходным адресом хоста

decnet dst - аналогична предыдущей комбинация, но фильтрует целевой адрес хоста

decnet - фильтрует трафик DECnet с исходным или целевым адресом хоста;

ip - овит трафик, соответствующий Ethernet-протоколу ip;

ip6 - ловит трафик ip6;

arp - ловит трафик arp;

rarp - ловит трафик rarp;

atalk - ловит трафик atalk;

aarp - ловит трафик aarp;

decnet - ловит трафик decnet;

iso- ловит трафик iso;

stp - ловит трафик stp;

ipx - ловит трафику ipx;

netbeui - ловит трафик netbeui;

vlan - идентификатор_ВЛВС Перехватывает пакеты на основе стандарта 802.1Q VLAN. Идентификатор виртуальной локальной сети можно опускать;

tcp - сокращенная форма комбинации ip proto tcp;

udp - сокращенная форма ip proto udp;

icmp - сокращенная форма ip proto icmp;

iso proto - перехватывает пакеты ВОС с заданным типом протокола - clnp, esis или isis;

clnp - сокращенная форма описанной выше комбинации с clnp в качестве протокола;

esis - сокращенная форма комбинации iso proto протокол с esis в качестве протокола;

isis - сокращенная форма комбинации iso proto протокол с isis в качестве протокола.

Существуют также более сложные выражения, которые можно строить с помощью булевых операций, таких как И, ИЛИ, НЕ, и операций сравнения (больше, меньше и т.п.).

IP Sniffer - бесплатный сниффер сети позволяющий получать подробную информация о всех проходящих пакетах по заданному протоколу. Кроме основных возможностей обычного сниффера, имеет возможность фильтрации обрабатываемых данных и декодирование содержимого пакетов. IP Sniffer содержит ряд встроенных сетевых утилит. Одной из них является монитор трафика, который занимается:

отображением диаграмм чаще всего используемых IP адресов и протоколов;

ARP (просмотр/удаление записей, отправка ответа);

Netstat (отображение сетевых подключений с возможностью принудительно завершать выбранные);

получение подробной информации о используемом сетевом адаптере;

Spoofing (TCP, UDP, ICMP, ARP протоколов);

WINS; DNS Query (using win32 DNSAPI);

поиск DHCP серверов;

служба WhoIs;

преобразование IP в Hostname и наоборот; PING (хостов или подсети);

сканирование определенного хоста или всей подсети на наличие открытых портов.

Netstat - утилита командной строки выводящая на дисплей состояние TCP-соединений (как входящих, так и исходящих), таблицы маршрутизации, число сетевых интерфейсов и сетевую статистику по протоколам. Она доступна на всех unix-подобных операционных системах, включая OS X, Linux, Solaris, и BSD, но также существует в основанных на Windows NT операционных системах начиная с Windows XP вплоть до Windows 10.

Основное назначение утилиты - это поиск сетевых проблем и определение производительности сети.

Следует иметь в виду, что в среде Linux утилита netstat, входящая в пакет net-tools, устарела. Вместо нее следует использовать утилиту ss из пакета iproute2.

Команда netstat выводит на экран содержимое различных структур данных, связанных с сетью, в различных форматах в зависимости от указанных опций.

Первая форма команды показывает список активных сокетов (sockets) для каждого протокола. Вторая форма выбирает одну из нескольких других сетевых структур данных. Третья форма показывает динамическую статистику пересылки пакетов по сконфигурированным сетевым интерфейсам; аргумент интервал задает, сколько секунд собирается информация между последовательными показами.

Значение по умолчанию для аргумента система - /unix; для аргумента core в качестве значения по умолчанию используется /dev/kmem.

Теперь ознакомимся с преобразование IP в Hostname:

Файл HOSTS используется в Windows для преобразования символьных имен доменов (google.com) в соответствующие им IP-адреса (64.233.167.99) и наоборот (точно такие же задачи в

сетях TCP/IP выполняет и DNS - Domain Name System - система доменных имен). То есть каждый раз, когда вы вводите в адресную строку браузера название сайта, ваш компьютер, прежде чем с ним соединится, должен преобразовать это «буквенное» название в соответствующие ему числа.

Файл HOSTS не имеет видимого расширения, но по сути его можно редактировать в любом текстовом редакторе (например, Notepad или Блокнот), как обычный файл текстового формата.

Его местоположение может отличаться в зависимости от вашей операционной системы, но по умолчанию файл HOSTS находится:

Windows 95/98/ME: WINDOWS\hosts;

Windows NT/2000: WINNT\system32\drivers\etc\hosts;

Windows XP/2003/Vista: WINDOWS\system32\drivers\etc\hosts;

В Windows NT/2000/XP/2003 это местоположение также можно изменить с помощью следующего ключа реестра:

HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters;

«DataBasePath»="%SystemRoot%\System32\drivers\etc";

Стандартный файл HOSTS Windows выглядит следующим образом:

Copyright © 1993—1999 Microsoft Corp

#

This is a sample HOSTS file used by Microsoft TCP/IP for Windows.

#

This file contains the mappings of IP addresses to host names. Each

#entry should be kept on an individual line. The IP address should

be placed in the first column followed by the corresponding host name.

The IP address and the host name should be separated by at least one

space.

#
Additionally, comments (such as these) may be inserted on individual
lines or following the machine name denoted by a '#' symbol.
#
For example:
#
102.54.94.97 rhino.acme.com # source server
38.25.63.10 x.acme.com # x client host
127.0.0.1 localhost

Файл HOSTS часто становится «жертвой» различного рода вредоносного ПО, которое вносит в него изменения (например, с целью перенаправления пользователя на свои веб-сайты, вместо тех, которые вводятся им в строку браузера или используются антивирусами для обновлений своих антивирусных баз).

Либо, если вы увидите в своем файле HOSTS следующие строки:

127.0.0.1 ftp.kasperskylab.ru;
127.0.0.1 ids.kaspersky-labs.com;
127.0.0.1 kaspersky.com;
127.0.0.1 kaspersky-labs.com;
127.0.0.1 liveupdate.symantec.com;
127.0.0.1 liveupdate.symanteclіveupdate.com;
127.0.0.1 www.symantec.com;
127.0.0.1 update.symantec.com;
127.0.0.1 updates.symantec.com;
127.0.0.1 updates1.kaspersky-labs.com;
127.0.0.1 updates1.kaspersky-labs.com;
127.0.0.1 updates2.kaspersky-labs.com;
127.0.0.1 updates3.kaspersky-labs.com;

То, это может означать только то, что кто-то хочет заблокировать вышеназванные сервера для вашей системы.

127.0.0.1 - это адрес вашего собственного компьютера.

Точно таким же образом вы и сами можете заблокировать нежелательные для вас веб-адреса, добавив в HOSTS-файл строку:

127.0.0.1 - нежелательный сайт.

Единственное, имейте в виду, что, если вы заблокируете таким образом много веб-сайтов (скажем, более сотни), будет очень желательно отключить службу DNS Client (DNS-клиент). Так как в противном случае интернет начнет ощутимо для вас тормозить.

Пример HOSTS-файла для блокировки нежелательных адресов также можно найти здесь: hpHosts (осторожнее, его размер более мегабайта). Если вы захотите заменить свой HOSTS файл на этот, на всякий случай сохраните копию оригинального HOSTS-файла, просто переименовав его, например, на hosts.original Большая часть назойливой рекламы и сайтов с нежелательным содержимым (где вы могли бы заразить свой компьютер) будет заблокирована. If you choose to download these files, please backup your original by renaming it to hosts.orig and saving the downloaded HOSTS file in its place. Using a HOSTS file such as these is highly recommended to protect your compute

Использование программы Wireshark

Программа Wireshark (ранее программа была известна под названием Ethereal) стала стандартом дефакто при исследовании сетей и анализе протоколов среди приложений с открытым исходным кодом. Данный сниффер (в дальнейшем в примерах и описании подразумевается версия 0.99.6a (SVN Rev 22276)) позволяет в режиме реального времени захватывать пакеты из сети, и анализировать их структуру. Также можно анализировать структуру пакетов из файла, содержащего трафик, полученный, например, программой «tcpdump» (unix/linux).

Очень частой проблемой при работе со стандартными настройками является то обстоятельство, что пользователю предоставляется огромный объем информации, а интересующую информацию становится очень сложно найти.

Большой объем информации уводит из поля зрения нужную информацию.

По этой причине фильтры так важны, ведь они могут помочь нам с поиском необходимой информации в обширном журнале данных:

Фильтры захвата: используются для указания на то, какие данные должны записываться в журнал данных. Эти фильтры задаются до начала захвата данных.

Фильтры отображения: используются для поиска внутри журнала данных. Эти фильтры могут быть изменены в процессе захвата данных.

На рисунке 4. изображено основное окно программы Wireshark. В стандартном режиме окно сниффера делится на 3 фрейма (панели): список захваченных пакетов, «анализатор» протоколов и исходные данные пакетов. Размер каждого фрейма можно менять по своему усмотрению.

Рассмотрим эти панели подробнее.

Верхняя панель содержит список пакетов, захваченных из сети. Список можно отсортировать по любому полю (в прямом или обратном порядке) – для этого нажать на заголовок соответствующего поля.

Каждая строка содержит следующие поля (по умолчанию):

- порядковый номер пакета (No.);
- время поступления пакета (Time);
- источник пакета (Source);
- пункт назначения (Destination);
- протокол (Protocol);
- информационное поле (Info).

Список полей настраивается в Edit/Perferencis/Columns. Для того, чтобы изменения возымели эффект необходимо перезапустить программу, предварительно нажав кнопку Save.

При нажатии правой кнопки мыши на том или ином пакете, появится контекстное меню. Нажатием на среднюю кнопку мыши можно

помечать группу интересующих нас пакетов.

Средняя панель содержит т.н. «дерево протоколов» для выбранного в верхнем окне пакета. В этой панели в иерархическом виде для выбранного в верхнем окне захваченного пакета отображается вложенность протоколов в соответствии с моделью взаимодействия открытых систем OSI. По нажатию на правую кнопку мыши вызывается контекстное

меню. При «раскрытии» каждого из протокола нажатием на значек «+» слева, выводятся поля данных соответствующих протоколов.

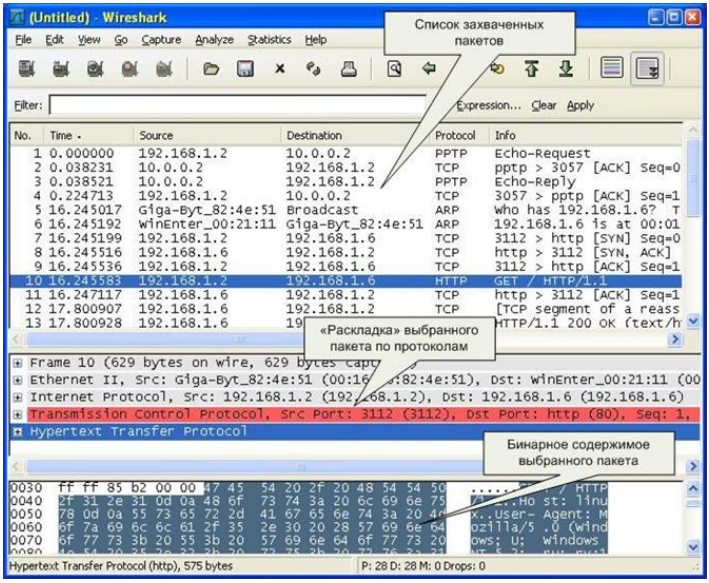


Рисунок 4. Основное окно сниффера Wireshark

Нижняя панель содержит шестнадцатеричное представление выбранного пакета. При выборе того или иного поля в средней панели автоматически будет подсвечиваться соответствующий участок 16-ого представления.

ЗАХВАТ ПАКЕТОВ

Для начала захвата пакетов необходимо задать параметры захвата. В частности, указать сетевой интерфейс, с которого и будет осуществляться захват пакетов. Это действие доступно через меню как «Capture\Options» или комбинации клавиш CTRL+K (см. рисунок 5). Интерфейс, задаваемый в поле «Interface:» можно выбрать из соответствующего поля. В примере на рис. 3. Показано, что доступны 3 интерфейса:

физический сетевой адаптер («Marvel…»), и интерфейсы для виртуальных каналов, в частности, установленного VPN-соединения («WAN (PPP/SLIP)»). В большинстве случае подходит выбор интерфейса сетевого адаптера.

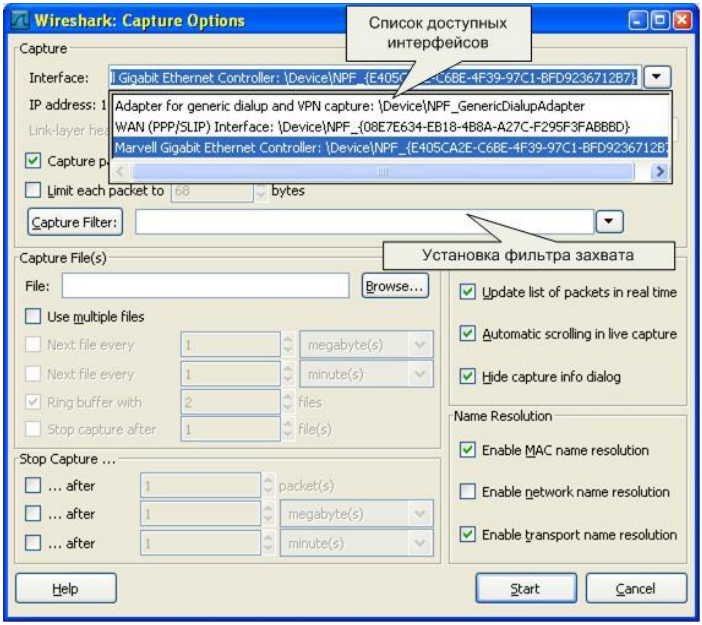


Рисунок 5. Выбор интерфейса и параметров захвата пакетов

В качестве дополнительных параметров захвата можно указать следующие:

- «Capture Filter» – фильтр захвата (будем рассматривать далее), служат для фильтрации еще на этапе захвата трафика. Но в таком случае, безусловно, можно безвозвратно потерять часть нужного трафика.

Фильтр представляет собой выражение, состоящее из встроенных значений, которые при необходимости могут объединяться логическими функциями (and, or, not).

По нажатию на соответствующую кнопку можно применить тот или иной фильтр отбора (из ранее сохраненных).

Если таковых не имеется, его можно указать явно в строке редактирования.

- «Update list of packets in real time» – обновление списка захваченных пакетов в режиме реального времени;
- «Stop Capture» – набор параметров, позволяющих задать то или иное значение при достижении, которого процесс захвата пакетов прекратится;
- «Name Resolution» – набор параметров разрешения имен позволяет определить какие из способов разрешения имен должны использоваться;

Для начала мониторинга сетевой активности нужно нажать «Start». После выбора интерфейса, который нас интересует, в дальнейшем можно начинать и останавливать захват пакетов через соответствующие команды в меню «Capture».

Фильтрация пакетов

Если запустить сниффер без дополнительных настроек, он будет «захватывать» все пакеты, проходящие через сетевые интерфейсы. Вообще, для общего ознакомления с процессами, происходящими в сети, очень полезно пронаблюдать активность сетевых протоколов в реальных условиях работы системы в сети. Пронаблюдать все разнообразие протоколов, запросов, ответов и др. событий.

При целенаправленном использовании сниффера очень часто необходимо выборочно отображать или захватывать пакеты по некоторым заданным критериям. Для этих целей служат фильтры отображения и захвата, соответственно.

Типы фильтрации трафика

Существует два варианта фильтрации пакетов: на этапе захвата и на этапе отображения пользователю. В первом случае эффективность работы сниффера и потребляемые им системные ресурсы значительно ниже, нежели во втором случае. Это объясняется тем, что при достаточно интенсивном сетевом трафике и продолжительном времени захвата все пакеты должны бить захвачены и сохранены либо в память, либо на дисковое устройство. Самые простые подсчеты могу показать, что даже для 100-мегабитной сети системных ресурсов хватит на непродолжительное время. Фильтрация захвата уже на

момент получения пакета гораздо эффективнее, однако в таком случае она должна быть реализована на уровне самих драйверов захвата. Данный факт, естественно, усложняет реализацию сниффера. Wireshark поддерживает оба варианта фильтрации.

Фильтры отображения

Фильтры отображения представляют собой достаточно мощное средство отображения трафика. Фильтры задаются в строке, располагающейся вверху основного экрана («Filter:»). Простейший фильтр отображения, позволяет отобрать пакеты по тому или иному протоколу. Для этого в строке требуется указать название протокола (например HTTP) и нажать кнопку «Apply». После этого в верхнем окне останутся пакеты, принадлежащие этому протоколу. Кнопкой «Reset» действие фильтра отменяется.

Для работы с фильтрами можно вызвать окно «Analyze/Display Filters». Можно сохранять созданные выражения под определенными именами для последующего использования и т.д.

С помощью логических операций (синтаксис языка Си) можно составлять логические выражения. Логическая истина - 1, ложь - 0.

Список поддерживаемых логических операций:

eq	==	равенство
ne	!=	не равно
gt	>	больше чем
Lt	<	меньше чем
ge	>=	больше равно
Le	<=	меньше равно

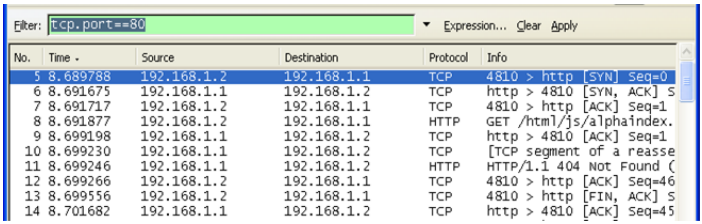


Рисунок 6. Пример задания простого фильтра отображения

Мастер построения фильтров отображения доступен через кнопку «Expression…» (см. рисунок 7).

Фильтры захвата

С помощью данных фильтров можно захватывать из сети только те пакеты, которые подходят под критерий отбора. Если не задано никакого фильтра (по умолчанию), то будут захватываться все пакеты. В противном случае только пакеты, для которых указанное выражение будет истинным. Синтаксис фильтров захвата несколько отличается от синтаксиса фильтров отображения. Выражение состоит из одного или более примитивов разделенных пробельными символами. На рисунке 7 приведен пример фильтра для захвата пакетов, адресованных на 80-й порт (http) узла с ip-адресом 10.197.0.11.

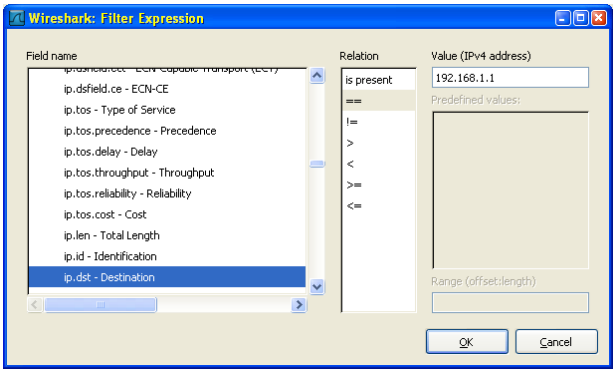


Рисунок 7. Построение фильтров отображения

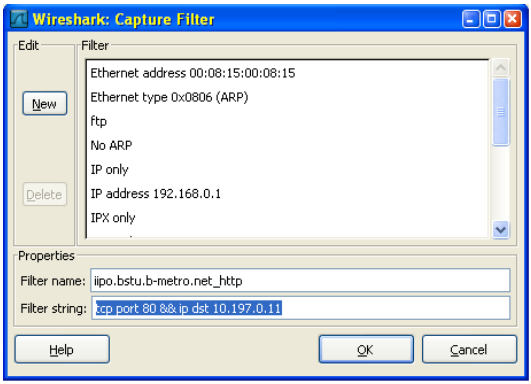


Рисунок 8. Пример фильтра захвата

Существует три различных типа примитивов: type, dir, proto.

Спецификатор type определяет тип параметра. Возможные параметры: host; net; port.

Например:

- host linux
- net 192.168.128
- port 80

Если не указано никакого типа предполагается что это параметр host.

Спецификатор dir определят направление передачи. Возможные направления: src; dst; src or dst; src and dst.

Например:

- src linux
- dst net 192.168.128
- src or dst port http

Если не определено направление то предполагается направление «src or dst». Для протоколов типа point-to-point используются спецификаторы inbound и outbound.

Спецификатор proto определят тип протокола, которому принадлежит пакет.

Возможные протоколы: ether; fddi; tr; ip/ipv6; arp/rarp; decent; tcp; udp.

Например:

- ether src linux
- arp net 192.168.128
- tcp port 80

Если протокол не определен, то будут захватываться пакеты всех протоколов. То есть: «src linux» означает «(ip or arp or rarp) src linux»; «net ctam» означает «(ip or arp or rarp) net ctam»; «port 53» означает «(tcp or udp) port 53».

Также существует несколько специальных спецификаторов, которые не попадают в описанные выше случаи:

- gateway;
- broadcast;
- less;
- greater;
- арифметические выражения.

Сложные фильтры захвата строятся с использованием логических выражений.

Список операций:

not	!	отрицание
and	&&	конкатенация (логическое И)
or	||	альтернатива (логическое ИЛИ)

Примеры фильтров захвата

Ниже рассмотрены некоторые примеры построения фильтров захвата:

- захват всех пакетов на сетевом интерфейсе хоста 192.168.1.2:

host 192.168.1.2

- захват трафика между хостом host1 И хостами host2 ИЛИ host3: host host1 and (host2 or host3)
- захват всех IP-пакетов между хостом host1 и каждым хостом за исключением hostX: ip host host1 and not hostX
- захват пакетов ни сгенерированных, ни адресованных локальными хостами: ip and not net localnet
- захват IP-пакетов размером больше чем 576 байт, проходящих через шлюз snup: gateway snup and ip[2:2]> 576
- захват всех ICMP пакетов, за исключением пакетов ping: icmp[0] != 8 and icmp[0] != 0

Статистическая обработка сетевого трафика

Сниффер Wireshark позволяет выполнять различную статистическую обработку полученных данных. Все доступные операции находятся в меню «Statistics».

Общая статистика – количество полученных/переданных пакетов, средняя скорость передачи и т.д. доступны через пункт «Statistics\Summary».

Получить информацию по статистике обработанных протоколов в полученных пакетах можно через пункт «Statistics\Protocol Hierarchy».

Статистику по типу ip-пакетов, их размеру и порту назначения можно получить выбрав подпункты меню «IP-address…», «Packet length» и «Port type» соответственно.

Одной из наиболее интересных возможностей является генерация диаграммы взаимодействия между узлами, которая доступна пунктом меню «Flow Graph…». В результате можно наблюдать в достаточно наглядной форме процесс взаимодействия на уровне протоколов. Например, на рисунке 9 приведена диаграмма взаимодействия при получении узлом win2003 статической web-странички с сервера http://linux.

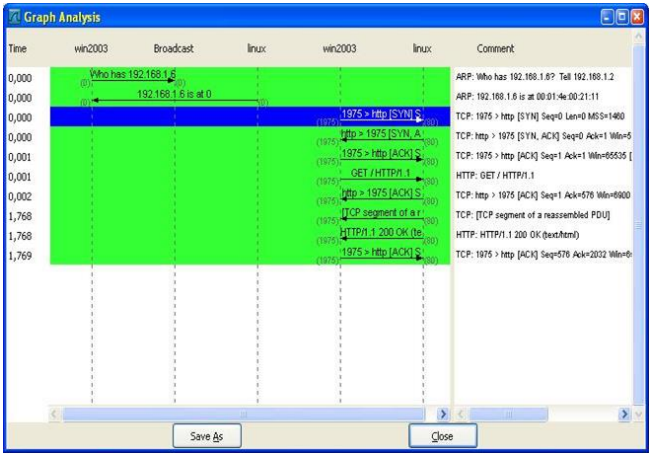


Рисунок 9. Диаграмма взаимодействия

Заключение

Программы-снифферы – это незаменимый инструмент для изучения того, что происходит в сети. Если знать, что в действительности посылается или принимается «по проводам», то трудные, на первый взгляд, ошибки удается легко найти и исправить. Сниффер представляет собой также важный инструмент для исследований динамики сети, а равно средство обучения.

ПРИМЕР ИСПОЛЬЗОВАНИЯ СНИФФЕРА

В качестве примера исследования некоторого протокола с использованием сниффера рассмотрим протокол ARP.

Протокол ARP

ARP (англ. Address Resolution Protocol - протокол разрешения адресов) - сетевой протокол, предназначенный для преобразования IP-адресов (адресов сетевого уровня) в MAC-адреса (адреса канального уровня) в сетях TCP/IP. Он определён в RFC 826.

Данный протокол очень распространенный и чрезвычайно важный. Каждый узел сети имеет как минимум два адреса, физический адрес и логический адрес. В сети Ethernet для идентификации источника и получателя информации используются оба адреса. Информация, пересылаемая от одного компьютера другому по сети, содержит в себе физический адрес отправителя, IP-адрес отправителя, физический адрес получателя и IP-адрес получателя. ARP-протокол обеспечивает связь между этими двумя адресами. Существует четыре типа ARP-сообщений: ARP-запрос (ARP request), ARP-ответ (ARP reply), RARP-запрос (RARP-request) и RARP-ответ (RARP-reply). Локальный хост при помощи ARP-запроса запрашивает физический адрес хоста-получателя. Ответ (физический адрес хоста-получателя) приходит в виде ARP-ответа. Хост-получатель, вместе с ответом, шлет также RARP-запрос, адресованный отправителю, для того, чтобы проверить его IP адрес. После проверки IP адреса отправителя, начинается передача пакетов данных.

Перед тем, как создать подключение к какому-либо устройству в сети, IP-протокол проверяет свой ARP-кеш, чтобы

выяснить, не зарегистрирована ли в нём уже нужная для подключения информация о хосте-получателе. Если такой записи в ARP-кеше нет, то выполняется широковещательный ARP-запрос. Этот запрос для устройств в сети имеет следующий смысл: «Кто-нибудь знает физический адрес устройства, обладающего следующим IP-адресом?» Когда получатель примет этот пакет, то должен будет ответить: «Да, это мой IP-адрес. Мой физический адрес следующий: ...» После этого отправитель обновит свой ARP-кеш, и будет способен передать информацию получателю.

RARP (англ. Reverse Address Resolution Protocol – обратный протокол преобразования адресов) – выполняет обратное отображение адресов, то есть преобразует аппаратный адрес в IP-адрес.

Протокол применяется во время загрузки узла (например, компьютера), когда он посылает групповое сообщение-запрос со своим физическим адресом. Сервер принимает это сообщение и просматривает свои таблицы (либо перенаправляет запрос куда-либо ещё) в поисках соответствующего физическому IP-адреса. После обнаружения найденный адрес отсылается обратно на запросивший его узел. Другие станции также могут «слышать» этот диалог и локально сохранить эту информацию в своих ARP-таблицах.

RARP позволяет разделять IP-адреса между не часто используемыми хост-узлами. После использования каким-либо узлом IP-адреса он может быть освобождён и выдан другому узлу. RARP является дополнением к ARP, и описан в RFC 903.

Для просмотра ARP-кеша можно использовать одноименную утилиту arp с параметром «-a».

Например:

```
D:\>arp -a
Interface: 192.168.1.2 --- 0x10003
 Internet Address Physical Address Type
 192.168.1.1 00-15-e9-b6-67-4f dynamic
192.168.1.6 00-01-4e-00-21-11 dynamic
```

Из данного результата команды arp видно, что в кеше на данный момент находится 2 записи и видны соответственно ip-адреса машин и MAC-адреса их сетевых адаптеров.

Записи в ARP-кеше могут быть статическими и динамическими. Пример, данный выше, описывает динамическую запись кеша. Хост-отправитель автоматически послал запрос получателю, не уведомляя при этом пользователя. Записи в ARP-кеш можно добавлять вручную, создавая статические (static) записи кеша. Это можно сделать при помощи команды:

arp -s <IP адрес> <MAC адрес>

Также можно удалять записи из ARP-кеша. Это осуществляется путем следующего вызова:

arp -d <IP адрес>

После того, как IP-адрес прошёл процедуру разрешения адреса, он остается в кеше в течение 2-х минут. Если в течение этих двух минут произошла повторная передача данных по этому адресу, то время хранения записи в кеше продлевается ещё на 2 минуты. Эта процедура может повторяться до тех пор, пока запись в кеше просуществует до 10 минут. После этого запись будет удалена из кеша и будет отправлен повторный ARP-запрос.

ARP изначально был разработан не только для IP протокола, но в настоящее время в основном используется для сопоставления IP- и MAC-адресов.

Посмотрим же на практике как работает протокол ARP/RARP. Для этого воспользуемся сниффером для захвата сетевого трафика.

Рассмотрим пример работы протокола ARP при обращении к машине с адресом 192.168.1.5, выполнив запрос с машины с адресом 192.168.1.2. Для успешного эксперимента предварительно очистим arp-кеш командой

arp -d 192.168.1.5

Для фильтрации ARP/RARP трафика воспользуемся фильтром захвата. В нашем случае это будет простой фильтр

arp or rarp

Далее запустим захват трафика командой «Start» и выполним обращение к заданной машине, например, «пропинговав» ее:

```
D:\>ping 192.168.1.5
Pinging 192.168.1.5 with 32 bytes of data:
Reply from 192.168.1.5: bytes=32 time<1ms TTL=64
Reply from 192.168.1.5: bytes=32 time<1ms TTL=64
Reply from 192.168.1.5: bytes=32 time<1ms TTL=64
Reply from 192.168.1.5: bytes=32 time<1ms TTL=64
Ping statistics for 192.168.1.5:
 Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
 Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Так как на момент начала работы утилиты ping в arp-кеше не было информации о MAC-адресе соответствующего узла, то первоначально система должна выполнить определение это самого MAC-адреса, сгенерировав ARP-запрос и отослав его в сеть широковещательным пакетом. После чего она будет ожидать ответа от заданного узла.

Посмотрим же, что мы получим на практике. После остановки сниффера мы должны увидеть результат схожий с тем, что отображен на рисунке 10. В нашем случае мы видим 2 захваченных пакета: ARP-запрос и ARP-ответ.

Проанализируем полученные пакеты. Сначала рассмотрим ARP-запрос (пакет №1). Выделив пакет курсором, мы получаем его раскладку по протоколам (Ethernet+ARP) в среднем окне. Wireshark очень наглядно «раскладывает» заголовок протокола по полям.

Мы можем видеть, что в пакете указаны MAC- и IP-адреса отправителя («Sender MAC address» и «Sender IP address» соответственно). Это параметры машины, с которой выполняется запрос. В данном случае запрос направлен на получения («Opcode: request» – запрос) MAC-адреса машины, у которой IP-адрес («Protocol type: IP») 192.168.1.5 («Target IP address»). При этом поле «Target MAC address» обнулено. Так как получатель ARP-запроса на момент запроса не известен, Ethernet-пакет отправляется всем машинам в данном локальном

сегменте, о чем сигнализирует MAC адрес Ethernet-пакета «ff:ff:ff:ff:ff:ff».

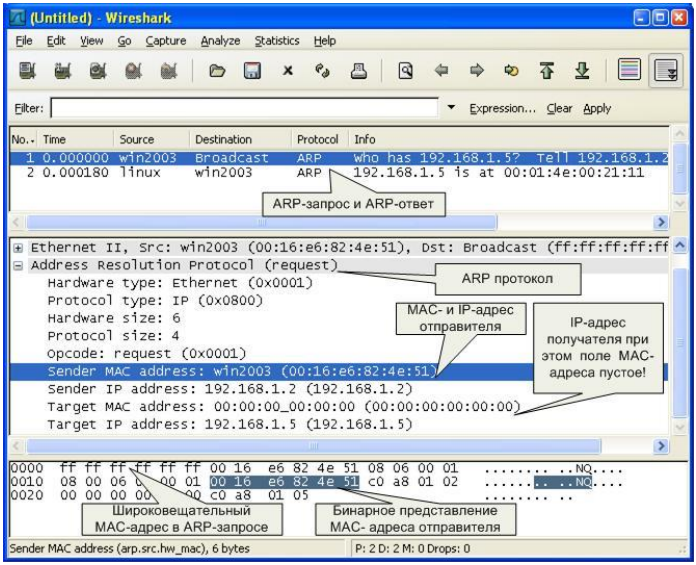


Рисунок 10. Анализ ARP-запроса

Примечание. Обратите внимание, что пакет представляет собой бинарную последовательность и сниффер выполняет большую работу по преобразованию полей из бинарного представления в удобочитаемый вариант.

Все работающие машины в сети получают пакет с ARP-запросом, анализируют его, а ответ отсылает только та машина, чей IP-адрес соответствует IP-адресу в запросе. Таким образом, второй полученный пакет является ARP-ответом (рисунок 11). Это следует из параметра поля «Opcode: reply». Обратите внимание, что данный пакет был отправлен именно той машиной, чей MAC-адрес нас и интересовал («Sender IP address: 192.168.1.5»). При этом поле «Sender MAC address» заполнено значением «00:01:4E:00:21:11».

Примечание. Обратите внимание на поле «Info» в списке захваченных пакетов. Сниффер и тут упрощает анализ сетевого трафика, подсказывая назначение пакетов.

Теперь мы можем повторно просмотреть ARP-кеш и сверить данные в нем с данными, которые мы узнали из анализа пакетов ARP-запрос/ответа:

```
D:\>arp -a
Interface: 192.168.1.2 --- 0x10003
 Internet Address Physical Address Type
 192.168.1.5 00-01-4e-00-21-11 dynamic
```

Стоит также отметить, что в реальных условиях в локальной сети с большим количеством машин arp/rarp трафик бывает гораздо более интенсивным.

Рассмотрим, что такое MAC-адрес:

MAC-адрес — управление доступом к среде, также Hardware Address) — уникальный идентификатор, присваиваемый каждой единице активного оборудования или некоторым их интерфейсам в компьютерных сетях Ethernet.

При проектировании стандарта Ethernet было предусмотрено, что каждая сетевая карта (равно как и встроенный сетевой интерфейс) должна иметь уникальный шестибайтный номер (MAC-адрес), прошитый в ней при изготовлении. Этот номер используется для идентификации отправителя и получателя фрейма, и предполагается, что при появлении в сети нового компьютера (или другого устройства, способного работать в сети) сетевому администратору не придётся настраивать MAC-адрес.

Уникальность MAC-адресов достигается тем, что каждый производитель получает в координирующем комитете IEEE Registration Authority диапазон из шестнадцати миллионов (224) адресов, и по мере исчерпания выделенных адресов может запросить новый диапазон. Поэтому по трём старшим байтам MAC-адреса можно определить производителя. Существуют таблицы, позволяющие определить производителя по MAC-адресу; в частности, они включены в программы типа arpalert.

В широковещательных сетях (таких, как сети на основе Ethernet) MAC-адрес позволяет уникально идентифицировать каждый узел сети и доставлять данные только этому узлу.

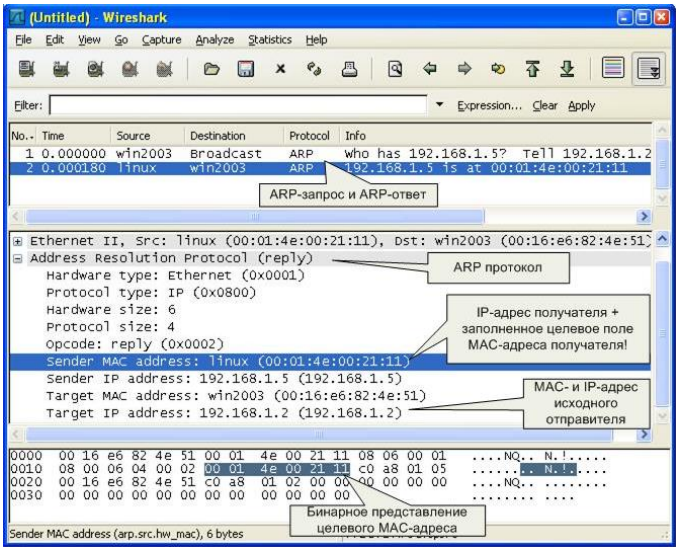


Рисунок 11. Анализ ARP-ответ

. Таким образом, MAC-адреса формируют основу сетей на канальном уровне, которую используют протоколы более высокого (сетевого) уровня. Для преобразования MAC-адресов в адреса сетевого уровня и обратно применяются специальные протоколы (например, ARP и RARP в сетях IPv4 и NDP в сетях на основе IPv6).

Большинство сетевых протоколов канального уровня используют одно из трёх пространств MAC-адресов, управляемых IEEE: MAC-48, EUI-48 и EUI-64. Адреса в каждом из пространств теоретически должны быть глобально уникальными. Не все протоколы используют MAC-адреса, и не все протоколы, использующие MAC-адреса, нуждаются в подобной уникальности этих адресов.

Адреса вроде MAC-48 наиболее распространены; они используются в таких технологиях, как Ethernet, Token ring, FDDI, WiMAX и др. Они состоят из 48 бит, таким образом, адресное пространство MAC-48 насчитывает 248 (или 281 474

976 710 656) адресов. Согласно подсчётам IEEE, этого запаса адресов хватит по меньшей мере до 2100 года.

EUI-48 от MAC-48 отличается лишь семантически: в то время как MAC-48 используется для сетевого оборудования, EUI-48 применяется для других типов аппаратного и программного обеспечения.

Идентификаторы EUI-64 состоят из 64 бит и используются в FireWire, а также в IPv6 в качестве младших 64 бит сетевого адреса узла.

Структура MAC-адреса представлена в традиционной бит-реверсной шестнадцатеричной записи

Стандарты IEEE определяют 48-разрядный (6 октетов) MAC-адрес, который разделен на четыре части.

Первые 3 октета (в порядке их передачи по сети; старшие 3 октета, если рассматривать их в традиционной бит-реверсной шестнадцатеричной записи MAC-адресов) содержат 24-битный уникальный идентификатор организации (OUI)[1], или (Код MFG — Manufacturing, производителя), который производитель получает в IEEE. При этом используются только младшие 22 разряда (бита), 2 старшие имеют специальное назначение:

первый бит (младший бит первого байта) указывает, для одиночного (0) или группового (1) адресата предназначен кадр

второй младший бит первого байта указывает, является ли MAC-адрес глобально (0) или локально (1) администрируемым.

Следующие три октета выбираются изготовителем для каждого экземпляра устройства. За исключением сетей системной сетевой архитектуры SNA.

Таким образом, глобально администрируемый MAC-адрес устройства глобально уникален и обычно «зашит» в аппаратуру.

Администратор сети имеет возможность, вместо использования «зашитого», назначить устройству MAC-адрес по своему усмотрению. Такой локально администрируемый MAC-адрес выбирается произвольно и может не содержать информации об OUI. Признаком локально администрируемого адреса является соответствующий бит первого октета адреса (см. выше).

Для того, чтобы узнать MAC-адрес сетевого устройства, используются следующие команды:

Windows — ipconfig /all — более подробно расписывает — какой MAC-адрес к какому сетевому интерфейсу относится

Windows — getmac /v — менее подробно расписывает — какой MAC-адрес к какому сетевому интерфейсу относится:

Linux — ifconfig -a | grep HWaddr;

FreeBSD — ifconfig |grep ether;

HP-UX — /usr/sbin/lanscan;

Mac OS X — ifconfig, либо в Системных Настройках > Сеть > выбрать подключение > Дополнительно > Ethernet > Идентификатор Ethernet;

QNX4 — netinfo –l;

QNX6 — ifconfig или nicinfo.

Существует распространенное мнение, что MAC-адрес жестко вшит в сетевую карту и сменить его нельзя или можно только с помощью программатора. На самом деле это не так. MAC-адрес легко меняется программным путём, так как значение, указанное через драйвер, имеет более высокий приоритет, чем зашитое в плату. Однако всё же существует оборудование, в котором смену MAC-адреса произвести невозможно иначе, как воспользовавшись программатором. Обычно это телекоммуникационное оборудование, например, приставки для IP-TV (STB).

В некоторых устройствах, оснащённых веб-интерфейсом управления, возможна смена MAC-адреса во время настройки. Так, большинство маршрутизаторов позволяют дублировать MAC-адрес сетевой платы, через которую он подключен к компьютеру.

В Windows смену MAC-адреса можно осуществить встроенными средствами ОС — в свойствах сетевой платы, во вкладке «Дополнительно» для редактирования доступно свойство «Сетевой адрес» (англ. «Network Address», у некоторых изготовителей сетевых плат это свойство называется «Locally Administered Address»), позволяющее принудительно присвоить нужный MAC-адрес.

В FreeBSD, OpenBSD MAC-адрес меняется одной командой от пользователя root:

```
# ifconfig re0 ether <mac-address>
```

где re0 - пример имени сетевого интерфейса

В Linux — такой командой от пользователя root:

ifconfig ethN hw ether <mac-адрес>

где ethN — имя сетевого интерфейса.

При этом после перезагрузки ОС смену MAC-адреса нужно произвести заново. Чтобы этого избежать, следует прописать смену MAC-адреса в стартовых конфигурационных файлах сетевых настроек. Например, в случае Debian-based дистрибутива Linux в файл /etc/network/interfaces нужно добавить строку

hwaddress ether <mac-адрес>

в блок конфигурации соответствующего сетевого интерфейса, либо заполнить пункт настроек MAC-адреса для соответствующего сетевого интерфейса в менеджере сетевых настроек графической оболочки.

Основу транспортных средств стека протоколов TCP/IP составляет протокол межсетевого взаимодействия - Internet Protocol (IP). К основным функциям протокола IP относятся:

- перенос между сетями различных типов адресной информации в унифицированной форме,
- сборка и разборка пакетов при передаче их между сетями с различным максимальным значением длины пакета.

Формат пакета IP

Пакет IP состоит из заголовка и поля данных. Заголовок пакета имеет следующие поля:

Поле Номер версии (VERS) указывает версию протокола IP. Сейчас повсеместно используется версия 4 и готовится переход на версию 6, называемую также IPng (IP next generation).

Поле Длина заголовка (HLEN) пакета IP занимает 4 бита и указывает значение длины заголовка, измеренное в 32-битовых словах. Обычно заголовок имеет длину в 20 байт (пять 32-битовых слов), но при увеличении объема служебной

информации эта длина может быть увеличена за счет использования дополнительных байт в поле Резерв (IP OPTIONS).

Поле Тип сервиса (SERVICE TYPE) занимает 1 байт и задает приоритетность пакета и вид критерия выбора маршрута. Первые три бита этого поля образуют подполе приоритета пакета (PRECEDENCE). Приоритет может иметь значения от 0 (нормальный пакет) до 7 (пакет управляющей информации). Маршрутизаторы и компьютеры могут принимать во внимание приоритет пакета и обрабатывать более важные пакеты в первую очередь. Поле Тип сервиса содержит также три бита, определяющие критерий выбора маршрута. Установленный бит D (delay) говорит о том, что маршрут должен выбираться для минимизации задержки доставки данного пакета, бит T - для максимизации пропускной способности, а бит R - для максимизации надежности доставки.

Поле Общая длина (TOTAL LENGTH) занимает 2 байта и указывает общую длину пакета с учетом заголовка и поля данных.

Поле Идентификатор пакета (IDENTIFICATION) занимает 2 байта и используется для распознавания пакетов, образовавшихся путем фрагментации исходного пакета. Все фрагменты должны иметь одинаковое значение этого поля.

Поле Флаги (FLAGS) занимает 3 бита, оно указывает на возможность фрагментации пакета (установленный бит Do not Fragment - DF - запрещает маршрутизатору фрагментировать данный пакет), а также на то, является ли данный пакет промежуточным или последним фрагментом исходного пакета (установленный бит More Fragments - MF - говорит о том пакет переносит промежуточный фрагмент).

Поле Смещение фрагмента (FRAGMENT OFFSET) занимает 13 бит, оно используется для указания в байтах смещения поля данных этого пакета от начала общего поля данных исходного пакета, подвергнутого фрагментации. Используется при сборке/разборке фрагментов пакетов при передачах их между сетями с различными величинами максимальной длины пакета.

Поле Время жизни (TIME TO LIVE) занимает 1 байт и указывает предельный срок, в течение которого пакет может

перемещаться по сети. Время жизни данного пакета измеряется в секундах и задается источником передачи средствами протокола IP. На шлюзах и в других узлах сети по истечении каждой секунды из текущего времени жизни вычитается единица; единица вычитается также при каждой транзитной передаче (даже если не прошла секунда). При истечении времени жизни пакет аннулируется.

Идентификатор Протокола верхнего уровня (PROTOCOL) занимает 1 байт и указывает, какому протоколу верхнего уровня принадлежит пакет (например, это могут быть протоколы TCP, UDP или RIP).

Контрольная сумма (HEADER CHECKSUM) занимает 2 байта, она рассчитывается по всему заголовку.

Поля Адрес источника (SOURCE IP ADDRESS) и Адрес назначения (DESTINATION IP ADDRESS) имеют одинаковую длину - 32 бита, и одинаковую структуру.

Поле Резерв (IP OPTIONS) является необязательным и используется обычно только при отладке сети. Это поле состоит из нескольких подполей, каждое из которых может быть одного из восьми предопределенных типов. В этих подполях можно указывать точный маршрут прохождения маршрутизаторов, регистрировать проходимые пакетом маршрутизаторы, помещать данные системы безопасности, а также временные отметки. Так как число подполей может быть произвольным, то в конце поля Резерв должно быть добавлено несколько байт для выравнивания заголовка пакета по 32-битной границе.

Максимальная длина поля данных пакета ограничена разрядностью поля, определяющего эту величину, и составляет 65535 байтов, однако 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 в 1500 байтов, умещающиеся в поле данных кадра Ethernet.

Управление фрагментацией

Протоколы транспортного уровня (протоколы TCP или UDP), пользующиеся сетевым уровнем для отправки пакетов, считают, что максимальный размер поля данных IP-пакета равен 65535, и поэтому могут передать ему сообщение такой

длины для транспортировки через интерсеть. В функции уровня IP входит разбиение слишком длинного для конкретного типа составляющей сети сообщения на более короткие пакеты с созданием соответствующих служебных полей, нужных для последующей сборки фрагментов в исходное сообщение.

В большинстве типов локальных и глобальных сетей определяется такое понятие как максимальный размер поля данных кадра или пакета, в которые должен инкапсулировать свой пакет протокол IP. Эту величину обычно называют максимальной единицей транспортировки - Maximum Transfer Unit, MTU. Сети Ethernet имеют значение MTU, равное 1500 байт, сети FDDI - 4096 байт, а сети X.25 чаще всего работают с MTU в 128 байт.

Работа протокола IP по фрагментации пакетов в хостах и маршрутизаторах иллюстрируется рисунком 4.1.

Пусть компьютер 1 связан с сетью, имеющей значение MTU в 4096 байтов, например, с сетью FDDI. При поступлении на IP-уровень компьютера 1 сообщения от транспортного уровня размером в 5600 байтов, протокол IP делит его на два IP-пакета, устанавливая в первом пакете признак фрагментации и присваивая пакету уникальный идентификатор, например, 486. В первом пакете величина поля смещения равна 0, а во втором - 2800. Признак фрагментации во втором пакете равен нулю, что показывает, что это последний фрагмент пакета. Общая величина IP-пакета составляет 2820 (размер заголовка IP), то есть 2820 байтов, что умещается в поле данных кадра FDDI. Маршрутизатор видит по сетевому адресу, что прибывшие два пакета нужно передать в сеть 2, которая имеет меньшее значение MTU, равное 1500. Вероятно, это сеть Ethernet. Маршрутизатор извлекает фрагмент транспортного сообщения из каждого пакета FDDI и делит его еще пополам, чтобы каждая часть уместилась в поле данных кадра Ethernet. Затем он формирует новые пакеты IP, каждый из которых имеет длину 1420 байтов, что меньше 1500 байтов, поэтому они нормально помещаются в поле данных кадров Ethernet.

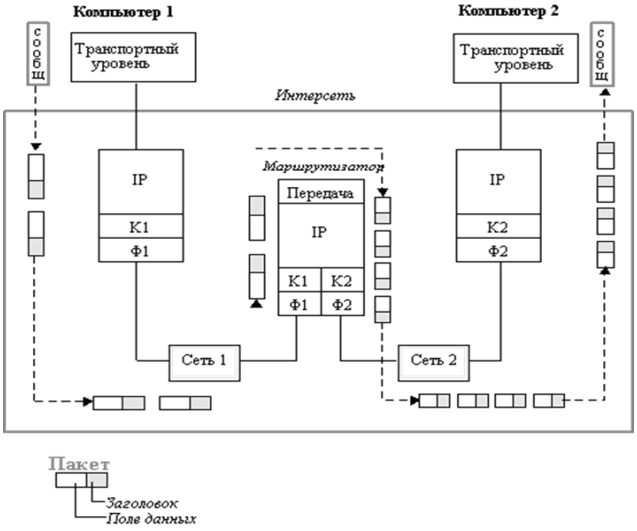


Рисунок 12. Фрагментация IP-пакетов при передаче между сетями с разными максимальными размерами пакетов. К1 и Ф1 канальный и физический уровень сети 1, К2 и Ф2 канальный и физический уровень сети 2 физический уровень Ф1, который отправляет их маршрутизатору, связанному с данной сетью.

В результате в компьютер 2 по сети Ethernet приходит четыре IP-пакета с общим идентификатором 486, что позволяет протоколу IP, работающему в компьютере 2, правильно собрать исходное сообщение. Если пакеты пришли не в том порядке, в котором были посланы, то смещение укажет правильный порядок их объединения.

Отметим, что IP-маршрутизаторы не собирают фрагменты пакетов в более крупные пакеты, даже если на пути встречается сеть, допускающая такое укрупнение. Это связано с тем, что отдельные фрагменты сообщения могут перемещаться по интерсети по различным маршрутам, поэтому нет гарантии, что все фрагменты проходят через какой-либо промежуточный маршрутизатор на их пути.

При приходе первого фрагмента пакета узел назначения запускает таймер, который определяет максимально допустимое время ожидания прихода остальных фрагментов этого пакета.

Если таймер истекает раньше прибытия последнего фрагмента, то все полученные к этому моменту фрагменты пакета отбрасываются, а в узел, пославший исходный пакет, направляется сообщение об ошибке с помощью протокола ICMP.

Маршрутизация с помощью IP-адресов

Рассмотрим теперь принципы, на основании которых в сетях IP происходит выбор маршрута передачи пакета между сетями.

Сначала необходимо обратить внимание на тот факт, что не только маршрутизаторы, но и конечные узлы - компьютеры - должны принимать участие в выборе маршрута. Пример, приведенный на рисунке 13., демонстрирует эту необходимость. Здесь в локальной сети имеется несколько маршрутизаторов, и компьютер должен выбирать, какому из них следует отправить пакет.

Рисунок 13. Выбор маршрутизатора конечным узлом

Длина маршрута может существенно измениться в зависимости от того, какой маршрутизатор выберет компьютер для передачи своего пакета на сервер, расположенный, например, в Германии, если маршрутизатор 1 соединен выделенной линией с маршрутизатором в Копенгагене, а маршрутизатор 2 имеет спутниковый канал, соединяющий его с Токио.

В стеке TCP/IP маршрутизаторы и конечные узлы принимают решения о том, кому передавать пакет для его

успешной доставки узлу назначения, на основании так называемых таблиц маршрутизации (routing tables).

В стеке TCP/IP принят так называемый одношаговый подход к оптимизации маршрута продвижения пакета (next-hop routing) - каждый маршрутизатор и конечный узел принимает участие в выборе только одного шага передачи пакета. Поэтому в каждой строке таблицы маршрутизации указывается не весь маршрут в виде последовательности IP-адресов маршрутизаторов, через которые должен пройти пакет, а только один IP-адрес - адрес следующего маршрутизатора, которому нужно передать пакет. Вместе с пакетом следующему маршрутизатору передается ответственность за выбор следующего шага маршрутизации. Одношаговый подход к маршрутизации означает распределенное решение задачи выбора маршрута. Это снимает ограничение на максимальное количество транзитных маршрутизаторов на пути пакета.

(Альтернативой одношаговому подходу является указание в пакете всей последовательности маршрутизаторов, которые пакет должен пройти на своем пути. Такой подход называется маршрутизацией от источника - Source Routing. В этом случае выбор маршрута производится конечным узлом или первым маршрутизатором на пути пакета, а все остальные маршрутизаторы только отрабатывают выбранный маршрут, осуществляя коммутацию пакетов, то есть передачу их с одного порта на другой. Алгоритм Source Routing применяется в сетях IP только для отладки, когда маршрут задается в поле Резерв (IP OPTIONS) пакета.)

В случае, если в таблице маршрутов имеется более одной строки, соответствующей одному и тому же адресу сети назначения, то при принятии решения о передаче пакета используется та строка, в которой указано наименьшее значение в поле "Расстояние до сети назначения".

При этом под расстоянием понимается любая метрика, используемая в соответствии с заданным в сетевом пакете классом сервиса. Это может быть количество транзитных маршрутизаторов в данном маршруте (количество хопов от hop - прыжок), время прохождения пакета по линиям связи, надежность линий связи, или другая величина, отражающая качество данного маршрута по отношению к конкретному

классу сервиса. Если маршрутизатор поддерживает несколько классов сервиса пакетов, то таблица маршрутов составляется и применяется отдельно для каждого вида сервиса (критерия выбора маршрута).

Для отправки пакета следующему маршрутизатору требуется знание его локального адреса, но в стеке TCP/IP в таблицах маршрутизации принято использование только IP-адресов для сохранения их универсального формата, не зависящего от типа сетей, входящих в интерсеть. Для нахождения локального адреса по известному IP-адресу необходимо воспользоваться протоколом ARP.

Конечный узел, как и маршрутизатор, имеет в своем распоряжении таблицу маршрутов унифицированного формата и на основании ее данных принимает решение, какому маршрутизатору нужно передавать пакет для сети N. Решение о том, что этот пакет нужно вообще маршрутизировать, компьютер принимает в том случае, когда он видит, что адрес сети назначения пакета отличается от адреса его собственной сети (каждому компьютеру при конфигурировании администратор присваивает его IP-адрес или несколько IP-адресов, если компьютер одновременно подключен к нескольким сетям).

Когда компьютер выбрал следующий маршрутизатор, то он просматривают кэш-таблицу адресов своего протокола ARP и, может быть, находит там соответствие IP-адреса следующего маршрутизатора его MAC-адресу. Если же нет, то по локальной сети передается широковещательный ARP-запрос и локальный адрес извлекается из ARP-ответа.

После этого компьютер формирует кадр протокола, используемого на выбранном порту, например, кадр Ethernet, в который помещает MAC-адрес маршрутизатора. Маршрутизатор принимает кадр Ethernet, извлекает из него пакет IP и просматривает свою таблицу маршрутизации для нахождения следующего маршрутизатора. При этом он выполняет те же действия, что и конечный узел.

Одношаговая маршрутизация обладает еще одним преимуществом - она позволяет сократить объем таблиц маршрутизации в конечных узлах и маршрутизаторах за счет использования в качестве номера сети назначения так

называемого маршрута по умолчанию - default, который обычно занимает в таблице маршрутизации последнюю строку. Если в таблице маршрутизации есть такая запись, то все пакеты с номерами сетей, которые отсутствуют в таблице маршрутизации, передаются маршрутизатору, указанному в строке *default*. Поэтому маршрутизаторы часто хранят в своих таблицах ограниченную информацию о сетях интерсети, пересылая пакеты для остальных сетей в порт и маршрутизатор, используемые по умолчанию. Подразумевается, что маршрутизатор, используемый по умолчанию, передаст пакет на магистральную сеть, а маршрутизаторы, подключенные к магистрали, имеют полную информацию о составе интерсети.

Особенно часто приемом маршрутизации по умолчанию пользуются конечные узлы. Хотя они также в общем случае имеют в своем распоряжении таблицу маршрутизации, ее объем обычно незначителен, так как маршрутизация для компьютера - не основное занятие. Главная роль в маршрутизации пакетов в концепции протокола IP отводится, естественно, маршрутизаторам, которые должны обладать гораздо более полными таблицами маршрутизации, чем конечные узлы. Конечный узел часто вообще работает без таблицы маршрутизации, имея только сведения об IP-адресе маршрутизатора по умолчанию. При наличии одного маршрутизатора в локальной сети этот вариант - единственно возможный для всех конечных узлов. Но даже при наличии нескольких маршрутизаторов в локальной сети, когда проблема их выбора стоит перед конечным узлом, задание маршрута по умолчанию часто используется в компьютерах для сокращения объема их маршрутной таблицы.

Другим способом разгрузки компьютера от необходимости ведения больших таблиц маршрутизации является получение от маршрутизатора сведений о рациональном маршруте для какой-нибудь конкретной сети с помощью протокола ICMP.

Кроме маршрута default, в таблице маршрутизации могут встретиться два типа специальных записей - запись о специфичном для узла маршруте и запись об адресах сетей, непосредственно подключенных к портам маршрутизатора.

Специфичный для узла маршрут содержит вместо номера сети полный IP-адрес, то есть адрес, имеющий ненулевую

информацию не только в поле номера сети, но и в поле номера узла. Предполагается, что для такого конечного узла маршрут должен выбираться не так, как для всех остальных узлов сети, к которой он относится. В случае, когда в таблице есть разные записи о продвижении пакетов для всей сети N и ее отдельного узла, имеющего адрес ND, при поступлении пакета, адресованного узлу ND, маршрутизатор отдаст предпочтение записи для N,D.

Записи в таблице маршрутизации, относящиеся к сетям, непосредственно подключенным к маршрутизатору, в поле "Расстояние до сети назначения" содержат нули.

Еще одним отличием работы маршрутизатора и конечного узла при выборе маршрута является способ построения таблицы маршрутизации. Если маршрутизаторы обычно автоматически создают таблицы маршрутизации, обмениваясь служебной информацией, то для конечных узлов таблицы маршрутизации создаются, как правило, вручную администраторами, и хранятся в виде постоянных файлов на дисках.

Существуют различные алгоритмы построения таблиц для одношаговой маршрутизации. Их можно разделить на три класса:

- алгоритмы фиксированной маршрутизации;
- алгоритмы простой маршрутизации;
- алгоритмы адаптивной маршрутизации.

Независимо от алгоритма, используемого для построения таблицы маршрутизации, результат их работы имеет единый формат. За счет этого в одной и той же сети различные узлы могут строить таблицы маршрутизации по своим алгоритмам, а затем обмениваться между собой недостающими данными, так как форматы этих таблиц фиксированы. Поэтому маршрутизатор, работающий по алгоритму адаптивной маршрутизации, может снабдить конечный узел, применяющий алгоритм фиксированной маршрутизации, сведениями о пути к сети, о которой конечный узел ничего не знает.

Фиксированная маршрутизация

Этот алгоритм применяется в сетях с простой топологией связей и основан на ручном составлении таблицы маршрутизации администратором сети. Алгоритм часто

эффективно работает также для магистралей крупных сетей, так как сама магистраль может иметь простую структуру с очевидными наилучшими путями следования пакетов в подсети, присоединенные к магистрали.

Различают одномаршрутные таблицы, в которых для каждого адресата задан один путь, и многомаршрутные таблицы, определяющие несколько альтернативных путей для каждого адресата. При использовании многомаршрутных таблиц должно быть задано правило выбора одного из них. Чаще всего один путь является основным, а остальные - резервными.

Простая маршрутизация

Алгоритмы простой маршрутизации подразделяются на три подкласса:

- случайная маршрутизация - пакеты передаются в любом, случайном направлении, кроме исходного;
- львиная маршрутизация - пакеты передаются во всех направлениях, кроме исходного (применяется в мостах для пакетов с неизвестным адресом доставки);
- маршрутизация по предыдущему опыту - таблицы маршрутов составляются на основании данных, содержащихся в проходящих через маршрутизатор пакетах. Именно так работают прозрачные мосты, собирая сведения об адресах узлов, входящих в сегменты сети. Такой способ маршрутизации обладает медленной адаптируемостью к изменениям топологии сети.
-

Адаптивная маршрутизация

Это основной вид алгоритмов маршрутизации, применяющихся маршрутизаторами в современных сетях со сложной топологией. Адаптивная маршрутизация основана на том, что маршрутизаторы периодически обмениваются специальной топологической информацией об имеющихся в интерсети сетях, а также о связях между маршрутизаторами. Обычно учитывается не только топология связей, но и их пропускная способность и состояние.

Адаптивные протоколы позволяют всем маршрутизаторам собирать информацию о топологии связей в сети, оперативно отрабатывая все изменения конфигурации связей.

Эти протоколы имеют распределенный характер, который выражается в том, что в сети отсутствуют какие-либо выделенные маршрутизаторы, которые бы собирали и обобщали топологическую информацию: эта работа распределена между всеми маршрутизаторами.

Пример взаимодействия узлов с использованием протокола IP

Рассмотрим на примере интерсети, приведенной на рисунке 14, каким образом происходит взаимодействие компьютеров через маршрутизаторы и доставка пакетов компьютеру назначения.

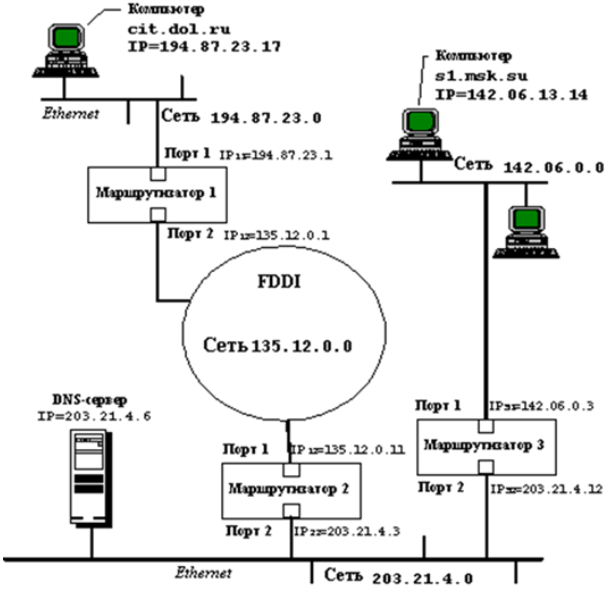


Рисунок 14. Взаимодействие компьютеров через маршрутизаторы и доставка пакетов компьютеру назначения.

Пусть в приведенном примере пользователь компьютера cit.dol.ru, находящийся в сети Ethernet с IP-адресом 194.87.23.0 (адрес класса C), хочет взаимодействовать по протоколу FTP с компьютером s1.msk.su, принадлежащем сети Ethernet с IP-адресом 142.06.0.0 (адрес класса B). Компьютер cit.dol.ru имеет IP-адрес 194.87.23.1.17, а компьютер s1.msk.su - IP-адрес 142.06.13.14.

В компьютере cit.dol.ru должны быть заданы некоторые параметры для стека TCP/IP, чтобы он мог выполнить поставленную перед ним задачу.

В число этих параметров должны входить собственный IP-адрес, IP-адрес DNS-сервера и IP-адрес маршрутизатора по умолчанию. Так как к сети Ethernet, к которой относится компьютер cit.dol.ru, подключен только один маршрутизатор, то таблица маршрутизации конечным узлам этой сети не нужна, достаточно знать IP-адрес маршрутизатора по умолчанию. В данном примере он равен 194.87.23.1.

Так как пользователь в команде ftp не задал IP-адрес узла, с которым он хочет взаимодействовать, то стек TCP/IP должен определить его самостоятельно. Он может сделать запрос к серверу DNS по имеющемуся у него IP-адресу, но обычно каждый компьютер сначала просматривает свою собственную таблицу соответствия символьных имен и IP-адресов. Такая таблица хранится чаще всего в виде текстового файла простой структуры - каждая его строка содержит запись об одном символьном имени и его IP-адресе. В ОС Unix такой файл традиционно носит имя HOSTS.

Будем считать, что компьютер cit.dol.ru имеет файл HOSTS, а в нем есть строка 142.06.13.14 s1.msk.su.

Поэтому разрешение имени выполняется локально, так что протокол IP может теперь формировать IP-пакеты с адресом назначения 142.06.13.14 для взаимодействия с компьютером s1.msk.su.

Протокол IP компьютера cit.dol.ru проверяет, нужно ли маршрутизировать пакеты для адреса 142.06.13.14. Так как адрес сети назначения равен 142.06.0.0, а адрес сети, к которой принадлежит компьютер, равен 194.87.23.0, то маршрутизация необходима.

Компьютер cit.dol.ru начинает формировать кадр Ethernet для отправки IP-пакета маршрутизатору по умолчанию с IP-адресом 194.87.23.1. Для этого ему нужен MAC-адрес порта маршрутизатора, подключенного к его сети. Этот адрес скорее всего уже находится в кэш-таблице протокола ARP компьютера, если он хотя бы раз за последнее включение обменивался данными с компьютерами других сетей. Пусть этот адрес в нашем примере был найден именно в кэш-памяти. Обозначим его MAC_{11}, в соответствии с номером маршрутизатора и его порта.

Кадр принимается портом 1 маршрутизатора 1 в соответствии с протоколом Ethernet, так как MAC-узел этого порта распознает свой адрес MAC_{11}. Протокол Ethernet извлекает из этого кадра IP-пакет и передает его программному обеспечению маршрутизатора, реализующему протокол IP. Протокол IP извлекает из пакета адрес назначения и просматривает записи своей таблицы маршрутизации. Пусть маршрутизатор 1 имеет в своей таблице маршрутизации запись 142.06.0.0 135.12.0.11 2 1, которая говорит о том, что пакеты для сети 142.06. 0.0 нужно передавать маршрутизатору 135.12.0.11, подключенному к той же сети, что и порт 2 маршрутизатора 1.

Структуризация сетей IP с помощью масок

Часто администраторы сетей испытывают неудобства, из-за того, что количество централизовано выделенных им номеров сетей недостаточно для того, чтобы структурировать сеть надлежащим образом, например, разместить все слабо взаимодействующие компьютеры по разным сетям.

В такой ситуации возможны два пути. Первый из них связан с получением от NIC дополнительных номеров сетей. Второй способ, употребляющийся более часто, связан с использованием так называемых *масок*, которые позволяют разделять одну сеть на несколько сетей.

Маска - это число, двоичная запись которого содержит единицы в тех разрядах, которые должны интерпретироваться как номер сети.

Например, для стандартных классов сетей маски имеют следующие значения:

255.0.0.0 - маска для сети класса A;
255.255.0.0 - маска для сети класса B;
255.255.255.0 - маска для сети класса C.

В масках, которые использует администратор для увеличения числа сетей, количество единиц в последовательности, определяющей границу номера сети, не обязательно должно быть кратным 8, чтобы повторять деление адреса на байты.

Пусть, например, маска имеет значение 255.255.192.0 (11111111 11111111 11000000 00000000). И пусть сеть имеет номер 129.44.0.0 (10000001 00101100 00000000 00000000), из которого видно, что она относится к классу B. После наложения маски на этот адрес число разрядов, интерпретируемых как номер сети, увеличилось с 16 до 18, то есть администратор получил возможность использовать вместо одного, централизованно заданного ему номера сети, четыре:

129.44.0.0 (10000001 00101100 00000000 00000000)
129.44.64.0 (10000001 00101100 01000000 00000000)
129.44.128.0 (10000001 00101100 10000000 00000000)
129.44.192.0 (10000001 00101100 11000000 00000000)

Например, IP-адрес 129.44.141.15 (10000001 00101100 10001101 00001111), который по стандартам IP задает номер сети 129.44.0.0 и номер узла 0.0.141.15, теперь, при использовании маски, будет интерпретироваться как пара:

129.44.128.0 - номер сети, 0.0. 13.15 - номер узла.

Таким образом, установив новое значение маски, можно заставить маршрутизатор по-другому интерпретировать IP-адрес. При этом два дополнительных последних бита номера сети часто интерпретируются как номера подсетей.

Еще один пример. Пусть некоторая сеть относится к классу B и имеет адрес 128.10.0.0 (рисунок 15). Этот адрес используется маршрутизатором, соединяющим сеть с остальной частью интерсети. И пусть среди всех станций сети есть станции, слабо взаимодействующие между собой. Их желательно было бы изолировать в разных сетях. Для этого сеть можно разделить на две сети, подключив их к соответствующим

портам маршрутизатора, и задать для этих портов в качестве маски, например, число 255.255.255.0, то есть организовать внутри исходной сети с централизовано заданным номером две подсети класса С (можно было бы выбрать и другой размер для поля адреса подсети). Извне сеть по-прежнему будет выглядеть, как единая сеть класса В, а на местном уровне это будут две отдельные сети класса С. Приходящий общий трафик будет разделяться местным маршрутизатором между подсетями.

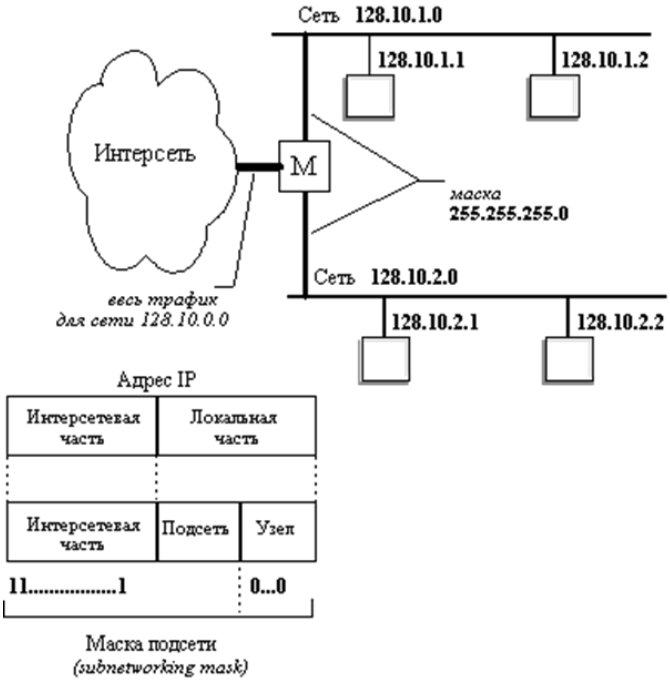


Рисунок 15. Пример использования масок для структурирования сети

Необходимо заметить, что, если принимается решение об использовании механизма масок, то соответствующим образом

должны быть сконфигурированы и маршрутизаторы, и компьютеры сети.

Протоколы обмена маршрутной информацией стека TCP/IP

Все протоколы обмена маршрутной информацией стека TCP/IP относятся к классу адаптивных протоколов, которые в свою очередь делятся на две группы, каждая из которых связана с одним из следующих типов алгоритмов:

- дистанционно-векторный алгоритм (Distance Vector Algorithms, DVA),
- алгоритм состояния связей (Link State Algorithms, LSA).

В алгоритмах *дистанционно-векторного типа* каждый маршрутизатор периодически и широковещательно рассылает по сети вектор расстояний от себя до всех известных ему сетей. Под расстоянием обычно понимается число промежуточных маршрутизаторов, через которые пакет должен пройти прежде, чем попадет в соответствующую сеть. Может использоваться и другая метрика, учитывающая не только число перевалочных пунктов, но и время прохождения пакетов по связи между соседними маршрутизаторами. Получив вектор от соседнего маршрутизатора, каждый маршрутизатор добавляет к нему информацию об известных ему других сетях, о которых он узнал непосредственно (если они подключены к его портам) или из аналогичных объявлений других маршрутизаторов, а затем снова рассылает новое значение вектора по сети. В конце-концов, каждый маршрутизатор узнает информацию об имеющихся в интерсети сетях и о расстоянии до них через соседние маршрутизаторы.

Дистанционно-векторные алгоритмы хорошо работают только в небольших сетях. В больших сетях они засоряют линии связи интенсивным широковещательным трафиком, к тому же изменения конфигурации могут отрабатываться по этому алгоритму не всегда корректно, так как маршрутизаторы не имеют точного представления о топологии связей в сети, а располагают только обобщенной информацией - вектором дистанций, к тому же полученной через посредников. Работа маршрутизатора в соответствии с дистанционно-векторным протоколом напоминает работу моста, так как точной топологической картины сети такой маршрутизатор не имеет.

Наиболее распространенным протоколом, основанным на дистанционно-векторном алгоритме, является протокол RIP.

Алгоритмы состояния связей обеспечивают каждый маршрутизатор информацией, достаточной для построения точного графа связей сети. Все маршрутизаторы работают на основании одинаковых графов, что делает процесс маршрутизации более устойчивым к изменениям конфигурации. Широковещательная рассылка используется здесь только при изменениях состояния связей, что происходит в надежных сетях не так часто.

Для того, чтобы понять, в каком состоянии находятся линии связи, подключенные к его портам, маршрутизатор периодически обменивается короткими пакетами со своими ближайшими соседями. Этот трафик также широковещательный, но он циркулирует только между соседями и поэтому не так засоряет сеть.

Протоколом, основанным на алгоритме состояния связей, в стеке TCP/IP является протокол OSPF.

Дистанционно-векторный протокол RIP

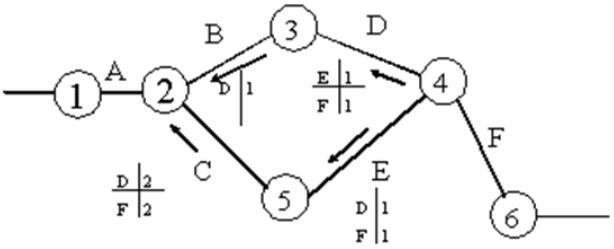
Представляет собой один из старейших протоколов обмена маршрутной информацией, однако он до сих пор чрезвычайно распространен в вычислительных сетях. Помимо версии RIP для сетей TCP/IP, существует также версия RIP для сетей IPX/SPX компании Novell.

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.

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.

На рисунке 16 приведен пример сети, состоящей из шести маршрутизаторов, имеющих идентификаторы от 1 до 6, и из шести сетей от A до F, образованных прямыми связями типа "точка-точка".



Начальная информация в узле 2:

Сеть	Расстояние	Сосед
A	1	-
B	1	-
C	1	-

После 2-х шагов:

A	1	-
B	1	-
C	1	-
D	2	3
E	2	5
D	3	5
F	3	5

Рисунок 16. Обмен маршрутной информацией по протоколу RIP

На рисунке приведена начальная информация, содержащаяся в топологической базе маршрутизатора 2, а также информация в этой же базе после двух итераций обмена маршрутными пакетами протокола RIP. После определенного числа итераций маршрутизатор 2 будет знать о расстояниях до всех сетей интерсети, причем у него может быть несколько альтернативных вариантов отправки пакета к сети назначения. Пусть в нашем примере сетью назначения является сеть D.

При необходимости отправить пакет в сеть D маршрутизатор просматривает свою базу данных маршрутов и выбирает порт, имеющий наименьшее расстояния до сети назначения (в данном случае порт, связывающий его с маршрутизатором 3).

Для адаптации к изменению состояния связей и оборудования с каждой записью таблицы маршрутизации связан таймер. Если за время тайм-аута не придет новое сообщение, подтверждающее этот маршрут, то он удаляется из маршрутной таблицы.

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.

На рисунке 17 показан случай неустойчивой работы сети по протоколу RIP при изменении конфигурации - отказе линии связи маршрутизатора М1 с сетью 1. При работоспособном состоянии этой связи в таблице маршрутов каждого маршрутизатора есть запись о сети с номером 1 и соответствующим расстоянием до нее.

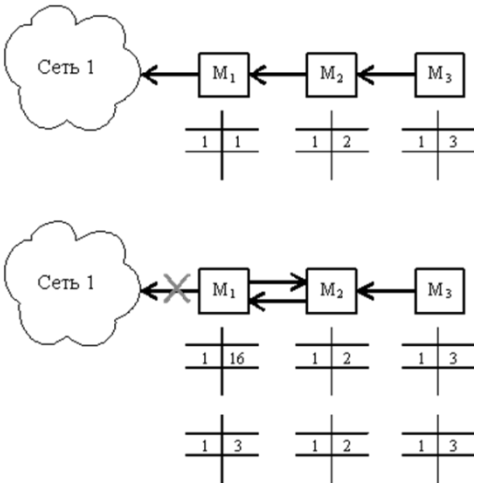


Рисунок 17. Пример неустойчивой работы сети при использовании протокола RIP

При обрыве связи с сетью 1 маршрутизатор M1 отмечает, что расстояние до этой сети приняло значение 16. Однако получив через некоторое время от маршрутизатора M2 маршрутное сообщение о том, что от него до сети 1 расстояние составляет 2 хопа, маршрутизатор M1 наращивает это расстояние на 1 и отмечает, что сеть 1 достижима через маршрутизатор 2. В результате пакет, предназначенный для сети 1, будет циркулировать между маршрутизаторами M1 и M2 до тех пор, пока не истечет время хранения записи о сети 1 в маршрутизаторе 2, и он не передаст эту информацию маршрутизатору M1.

Для исключения подобных ситуаций маршрутная информация об известной маршрутизатору сети не передается тому маршрутизатору, от которого она пришла.

Существуют и другие, более сложные случаи нестабильного поведения сетей, использующих протокол RIP, при изменениях в состоянии связей или маршрутизаторов сети.

Большинство протоколов маршрутизации, применяемых в современных сетях с коммутацией пакетов, ведут свое происхождение от сети Internet и ее предшественницы - сети ARPANET. Для того, чтобы понять их назначение и особенности, полезно сначала познакомится со структурой сети Internet, которая наложила отпечаток на терминологию и типы протоколов.

Internet изначально строилась как сеть, объединяющая большое количество существующих систем. С самого начала в ее структуре выделяли магистральную сеть **(**core backbone network**)**, а сети, присоединенные к магистрали, рассматривались как автономные системы (autonomous systems*)*. Магистральная сеть и каждая из автономных систем имели свое собственное административное управление и собственные протоколы маршрутизации. Общая схема архитектуры сети Internet показана на рисунке 8.3**.** Далее маршрутизаторы будут называться шлюзами для следования традиционной терминологии Internet.

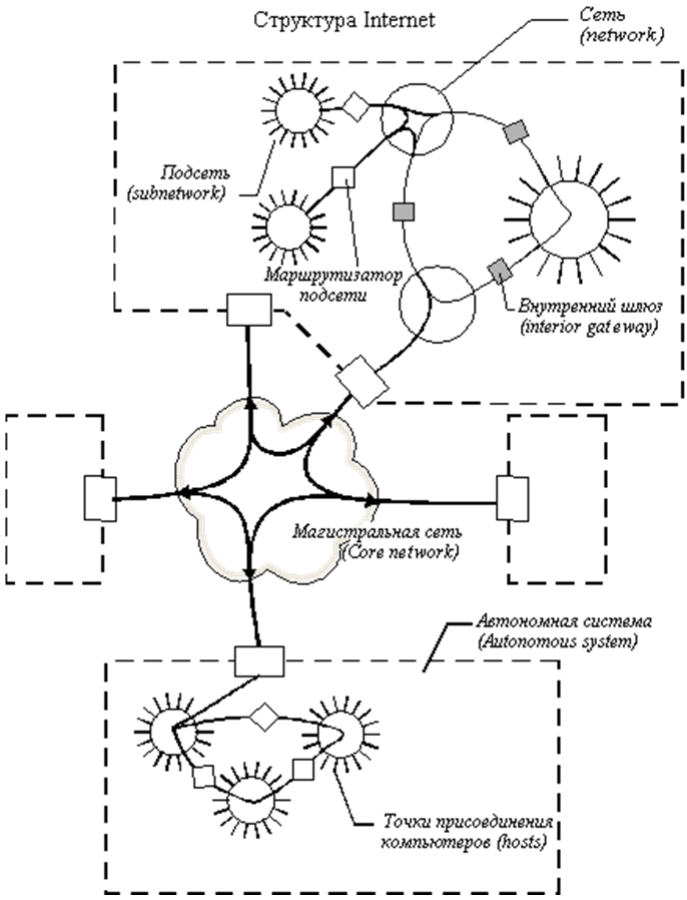


Рисунок 18. Архитектура сети Internet

Шлюзы, которые используются для образования подсетей внутри автономной системы, называются внутренними шлюзами **(**interior gateways**),** а шлюзы, с помощью которых автономные системы присоединяются к магистрали сети, называются внешними шлюзами (exterior gateways*)*. Непосредственно друг с другом автономные системы не соединяются. Соответственно, протоколы маршрутизации, используемые внутри автономных систем, называются

протоколами внутренних шлюзов (interior gateway protocol, IGP), а протоколы, определяющие обмен маршрутной информацией между внешними шлюзами и шлюзами магистральной сети - протоколами внешних шлюзов (exterior gateway protocol, EGP). Внутри магистральной сети также может использоваться любой собственный внутренний протокол IGP.

Смысл разделения всей сети Internet на автономные системы в ее многоуровневом представлении, что необходимо для любой крупной системы, способной к расширению в больших масштабах. Внутренние шлюзы могут использовать для внутренней маршрутизации достаточно подробные графы связей между собой, чтобы выбрать наиболее рациональный маршрут. Однако, если информация такой степени детализации будет храниться во всех маршрутизаторах сети, то топологические базы данных так разрастутся, что потребуют наличия памяти гигантских размеров, а время принятия решений о маршрутизации непременно возрастет.

Поэтому детальная топологическая информация остается внутри автономной системы, а автономную систему как единое целое для остальной части Internet представляют внешние шлюзы, которые сообщают о внутреннем составе автономной системы минимально необходимые сведения - количество IP-сетей, их адреса и внутреннее расстояние до этих сетей от данного внешнего шлюза.

При инициализации внешний шлюз узнает уникальный идентификатор обслуживаемой им автономной системы, а также таблицу достижимости **(**reachability table**)**, которая позволяет ему взаимодействовать с другими внешними шлюзами через магистральную сеть.

Затем внешний шлюз начинает взаимодействовать по протоколу EGP с другими внешними шлюзами и обмениваться с ними маршрутной информацией, состав которой описан выше. В результате, при отправке пакета из одной автономной системы в другую, внешний шлюз данной системы на основании маршрутной информации, полученной от всех внешних шлюзов, с которыми он общается по протоколу EGP, выбирает наиболее подходящий внешний шлюз и отправляет ему пакет.

В протоколе EGP определены три основные функции:

- установление соседских отношений;
- подтверждение достижимости соседа;
- обновление маршрутной информации.

Каждая функция работает на основе обмена сообщениями запрос-ответ.

Так как каждая автономная система работает под контролем своего административного штата, то перед началом обмена маршрутной информацией внешние шлюзы должны согласиться на такой обмен. Сначала один из шлюзов посылает *запрос на установление соседских отношений (acquisition request)* другому шлюзу. Если тот согласен на это, то он отвечает сообщением подтверждение *установления соседских отношений (*acquisition *confirm)*, а если нет - то сообщением *отказ от установления соседских отношений (*acquisition *refuse),* которое содержит также причину отказа.

После установления соседских отношений шлюзы начинают периодически проверять состояние достижимости друг друга. Это делается либо с помощью специальных сообщений (*привет (hello)* и *Я-услышал-тебя (I-heard-you)*)**,** либо встраиванием подтверждающей информации непосредственно в заголовок обычного маршрутного сообщения.

Обмен маршрутной информацией начинается с посылки одним из шлюзов другому сообщения *запрос данных (poll request)* о номерах сетей, обслуживаемых другим шлюзом и расстояниях до них от него. Ответом на это сообщение служит сообщение *обновленная маршрутная информация (routing update).* Если же запрос оказался некорректным, то в ответ на него отсылается сообщение об ошибке.

Все сообщения протокола EGP передаются в поле данных IP-пакетов. Сообщения EGP имеют заголовок фиксированного формата (рисунок 19).

Поля *Тип* и *Код* совместно определяют тип сообщения, а поле *Статус* - информацию, зависящую от типа сообщения. Поле *Номер автономной системы* - это номер, назначенный той автономной системе, к которой присоединен данный внешний шлюз. Поле *Номер последовательности* служит для синхронизации процесса запросов и ответов.

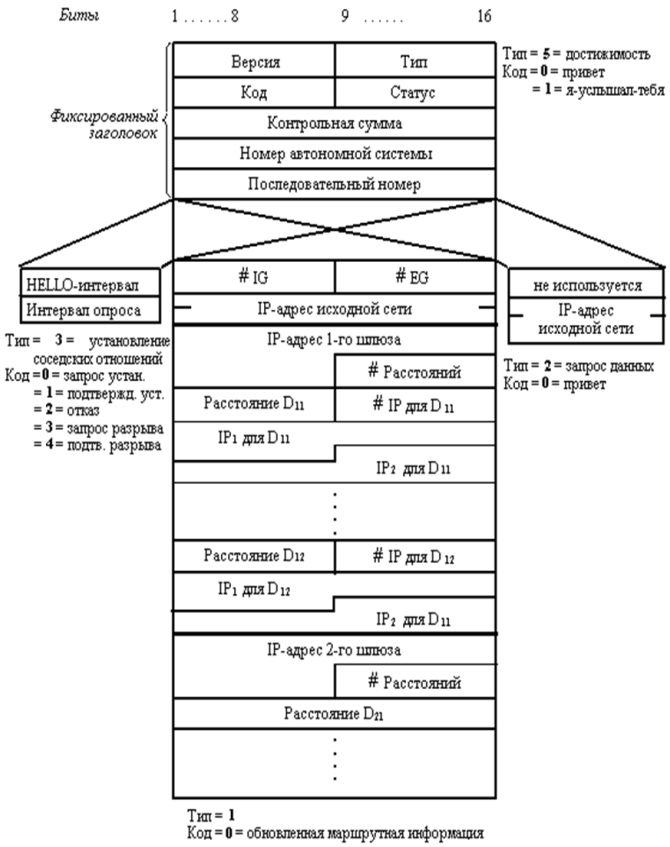


Рисунок 19. Формат сообщения протокола EGP

Поле IP-адрес исходной сети в сообщениях запроса и обновления маршрутной информации обозначает сеть, соединяющую два внешних шлюза (рисунок 20).

Сообщение об обновленной маршрутной информации содержит список адресов сетей, которые достижимы в данной автономной системе. Этот список упорядочен по внутренним шлюзам, которые подключены к исходной сети и через которые достижимы данные сети, а для каждого шлюза он упорядочен

по расстоянию до каждой достижимой сети от *исходной сети*, а *не от данного внутреннего шлюза*. Для примера, приведенного на рисунке 8.5, внешний шлюз R2 в своем сообщении указывает, что сеть 4 достижима с помощью шлюза R3 и расстояние ее равно 2, а сеть 2 достижима через шлюз R2 и ее расстояние равно 1 (а не 0, как если бы шлюз измерял ее расстояние от себя, как в протоколе RIP).

Протокол EGP имеет достаточно много ограничений, связанных с тем, что он рассматривает магистральную сеть как одну неделимую магистраль.

Развитием протокола EGP является протокол *BGP (Border Gateway Protocol),* имеющий много общего с EGP и используемый наряду с ним в магистрали сети Internet.

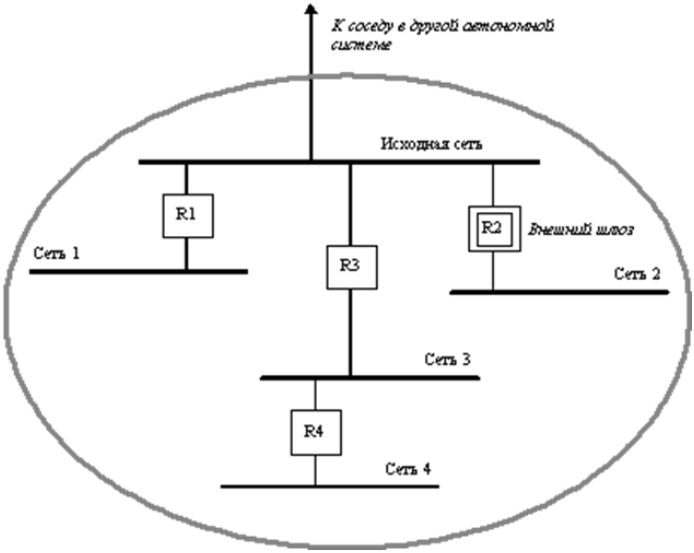


Рисунок 20. Пример автономной системы

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.

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.

Интерфейс OSPF (порт маршрутизатора, поддерживающего протокол OSPF) является обобщением подсети IP. Подобно подсети IP, интерфейс OSPF имеет IP-адрес и маску подсети. Если один порт OSPF поддерживает более, чем одну подсеть, протокол OSPF рассматривает эти подсети так, как если бы они были на разных физических интерфейсах, и вычисляет маршруты соответственно.

Интерфейсы, к которым подключены локальные сети, называются *широковещательными (broadcast) интерфейсами,* так как они могут использовать широковещательные возможности локальных сетей для обмена сигнальной информацией между маршрутизаторами. Интерфейсы, к которым подключены глобальные сети, не поддерживающие широковещание, но обеспечивающие доступ ко многим узлам через одну точку входа, например, сети X.25 или frame relay, называются не широковещательными интерфейсами с множественным доступом или *NBMA (non-broadcast multi-access).* Они рассматриваются аналогично широковещательным интерфейсам за исключением того, что широковещательная рассылка эмулируется путем посылки сообщения каждому соседу. Так как обнаружение соседей не является автоматическим, как в широковещательных сетях, NBMA-соседи должны задаваться при конфигурировании вручную. Как на широковещательных, так и на NBMA-интерфейсах могут быть заданы приоритеты маршрутизаторов для того, чтобы они могли выбрать выделенный маршрутизатор.

Интерфейсы "точка-точка", подобные PPP, несколько отличаются от традиционной IP-модели. Хотя они и могут иметь IP-адреса и под маски, но необходимости в этом нет.

В простых сетях достаточно определить, что пункт назначения достижим и найти маршрут, который будет удовлетворительным. В сложных сетях обычно имеется несколько возможных маршрутов. Иногда хотелось бы иметь возможности по установлению дополнительных критериев для выбора пути: например, наименьшая задержка, максимальная пропускная способность или наименьшая стоимость (в сетях с оплатой за пакет). По этим причинам протокол OSPF позволяет сетевому администратору назначать каждому интерфейсу определенное число, называемое *метрикой,* чтобы оказать нужное влияние на выбор маршрута.

Число, используемое в качестве метрики пути, может быть назначено произвольным образом по желанию администратора. Но по умолчанию в качестве метрики используется время передачи бита в 10-ти наносекундных единицах (10 Мб/с Ethernet'у назначается значение 10, а линии 56 Кб/с - число 1785). Вычисляемая протоколом OSPF метрика пути представляет собой сумму метрик всех проходимых в пути связей; это очень грубая оценка задержки пути. Если маршрутизатор обнаруживает более, чем один путь к удаленной подсети, то он использует путь с наименьшей *стоимостью пути.*

В протоколе OSPF используется несколько временных параметров, и среди них наиболее важными являются интервал сообщения HELLO и *интервал отказа маршрутизатора* (*router dead interval).*

HELLO - это сообщение, которым обмениваются соседние, то есть непосредственно связанные маршрутизаторы подсети, с целью установить состояние линии связи и состояние маршрутизатора-соседа. В сообщении HELLO маршрутизатор передает свои рабочие параметры и говорит о том, кого он рассматривает в качестве своих ближайших соседей. Маршрутизаторы с разными рабочими параметрами игнорируют сообщения HELLO друг друга, поэтому неверно сконфигурированные маршрутизаторы не будут влиять на работу сети. Каждый маршрутизатор шлет сообщение HELLO

каждому своему соседу по крайней мере один раз на протяжении интервала HELLO. Если интервал отказа маршрутизатора истекает без получения сообщения HELLO от соседа, то считается, что сосед неработоспособен, и распространяется новое объявление о сетевых связях, чтобы в сети произошел пересчет маршрутов.

Пример маршрутизации по алгоритму OSPF

Представим себе один день из жизни транзитной локальной сети. Пусть у нас имеется сеть Ethernet, в которой есть три маршрутизатора - Джон, Фред и Роб (имена членов рабочей группы Internet, разработавшей протокол OSPF). Эти маршрутизаторы связаны с сетями в других городах с помощью выделенных линий.

Пусть произошло восстановление сетевого питания после сбоя. Маршрутизаторы и компьютеры перезагружаются и начинают работать по сети Ethernet. После того, как маршрутизаторы обнаруживают, что порты Ethernet работают нормально, они начинают генерировать сообщения HELLO, которые говорят о их присутствии в сети и их конфигурации. Однако маршрутизация пакетов начинает осуществляться не сразу - сначала маршрутизаторы должны синхронизировать свои маршрутные базы (рисунок 21).

На протяжении интервала отказа маршрутизаторы продолжают посылать сообщения HELLO. Когда какой-либо маршрутизатор посылает такое сообщение, другие его получают и отмечают, что в локальной сети есть другой маршрутизатор. Когда они посылают следующее HELLO, они перечисляют там и своего нового соседа.

Когда период отказа маршрутизатора истекает, то маршрутизатор с наивысшим приоритетом и наибольшим идентификатором объявляет себя выделенным (а следующий за ним по приоритету маршрутизатор объявляет себя резервным выделенным маршрутизатором) и начинает синхронизировать свою базу данных с другими маршрутизаторами.

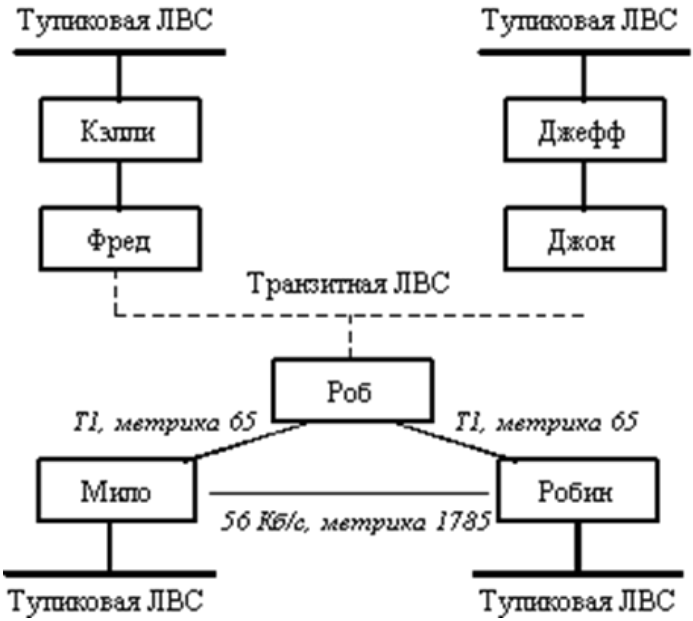


Рисунок 21. Гипотетическая сеть с OSPF маршрутизаторами

С этого момента времени база данных маршрутных объявлений каждого маршрутизатора может содержать информацию, полученную от маршрутизаторов других локальных сетей или из выделенных линий. Роб, например, вероятно получил информацию от Мило и Робина об их сетях, и он может передавать туда пакеты данных. Они содержат информацию о собственных связях маршрутизатора и объявления о связях сети.

Базы данных теперь синхронизированы с выделенным маршрутизатором, которым является Джон. Джон суммирует свою базу данных с каждой базой данных своих соседей - базами Фреда, Роба и Джеффа - индивидуально. В каждой синхронизирующейся паре объявления, найденные только в какой-либо одной базе, копируются в другую. Выделенный маршрутизатор, Джон, распространяет новые объявления среди

других маршрутизаторов своей локальной сети. Например, объявления Мило и Робина передаются Джону Робом, а Джон в свою очередь передает их Фреду и Джеффри. Обмен информацией между базами продолжается некоторое время, и пока он не завершится, маршрутизаторы не будут считать себя работоспособными. После этого они себя таковыми считают, потому что имеют всю доступную информацию о сети.

Посмотрим теперь, как Робин вычисляет маршрут через сеть. Две из связей, присоединенных к его портам, представляют линии Т-1, а одна - линию 56 Кб/с. Робин сначала обнаруживает двух соседей - Роба с метрикой 65 и Мило с метрикой 1785. Из объявления о связях Роба Робин обнаружил наилучший путь к Мило со стоимостью 130, поэтому он отверг непосредственный путь к Мило, поскольку он связан с большей задержкой, так как проходит через линии с меньшей пропускной способностью. Робин также обнаруживает транзитную локальную сеть с выделенным маршрутизатором Джоном. Из объявлений о связях Джона Робин узнает о пути к Фреду и, наконец, узнает о пути к маршрутизаторам Келли и Джеффа и к их тупиковым сетям.

После того, как маршрутизаторы полностью входят в рабочий режим, интенсивность обмена сообщениями резко падает. Обычно они посылают сообщение HELLO по своим подсетям каждые 10 секунд и делают объявления о состоянии связей каждые 30 минут (если обнаруживаются изменения в состоянии связей, то объявление передается, естественно, немедленно). Обновленные объявления о связях служат гарантией того, что маршрутизатор работает в сети. Старые объявления удаляются из базы через определенное время.

Представим, однако, что какая-либо выделенная линия сети отказала. Присоединенные к ней маршрутизаторы распространяют свои объявления, в которых они уже не упоминают друг друга. Эта информация распространяется по сети, включая маршрутизаторы транзитной локальной сети. Каждый маршрутизатор в сети пересчитывает свои маршруты, находя, может быть, новые пути для восстановления утраченного взаимодействия.

Использование других протоколов маршрутизации

Случай использования в сети только протокола маршрутизации OSPF представляется маловероятным. Если сеть присоединена к Internet'у, то могут использоваться такие протоколы, как *EGP (Exterior Gateway protocol), BGP (Border Gateway Protocol*, протокол пограничного маршрутизатора), старый протокол маршрутизации RIP или собственные протоколы производителей.

Когда в сети начинает применяться протокол OSPF, то существующие протоколы маршрутизации могут продолжать использоваться до тех пор, пока не будут полностью заменены. В некоторых случаях необходимо будет объявлять о статических маршрутах, сконфигурированных вручную.

В OSPF существует понятие *автономных систем* маршрутизаторов (autonomous systems), которые представляют собой домены маршрутизации, находящиеся под общим административным управлением и использующие единый протокол маршрутизации. OSPF называет маршрутизатор, который соединяет автономную систему с другой автономной системой, использующей другой протокол маршрутизации, *пограничным маршрутизатором автономной системы* (autonomous system boundary router, ASBR).

В OSPF маршруты (именно маршруты, то есть номера сетей и расстояния до них во внешней метрике, а не топологическая информация) из одной автономной системы импортируются в другую автономную систему и распространяются с использованием специальных внешних объявлений о связях.

Внешние маршруты обрабатываются за два этапа. Маршрутизатор выбирает среди внешних маршрутов маршрут с наименьшей внешней метрикой. Если таковых оказывается больше, чем 2, то выбирается путь с меньшей стоимостью внутреннего пути до ASBR.

Область OSPF - это набор смежных интерфейсов (территориальных линий или каналов локальных сетей). Введение понятия "область" служит двум целям - управлению информацией и определению доменов маршрутизации.

Для понимания принципа управления информацией рассмотрим сеть, имеющую следующую структуру: центральная локальная сеть связана с помощью 50 маршрутизаторов с

большим количеством соседей через сети X.25 или frame relay (рисунок 22). Эти соседи представляют собой большое количество небольших удаленных подразделений, например, отделов продаж или филиалов банка. Из-за большого размера сети каждый маршрутизатор должен хранить огромное количество маршрутной информации, которая должна передаваться по каждой из линий, и каждое из этих обстоятельств удорожает сеть. Так как топология сети проста, то большая часть этой информации и создаваемого ею трафика не имеют смысла.

Для каждого из удаленных филиалов нет необходимости иметь детальную маршрутную информацию о всех других удаленных офисах, в особенности, если они взаимодействуют в основном с центральными компьютерами, связанными с центральными маршрутизаторами. Аналогично, центральным маршрутизаторам нет необходимости иметь детальную информацию о топологии связей с удаленными офисами, соединенными с другими центральными маршрутизаторами. В то же время центральные маршрутизаторы нуждаются в информации, необходимой для передачи пакетов следующему центральному маршрутизатору. Администратор мог бы без труда разделить эту сеть на более мелкие домены маршрутизации для того, чтобы ограничить объемы хранения и передачи по линиям связи не являющейся необходимой информации. Обобщение маршрутной информации является главной целью введения областей в OSPF.

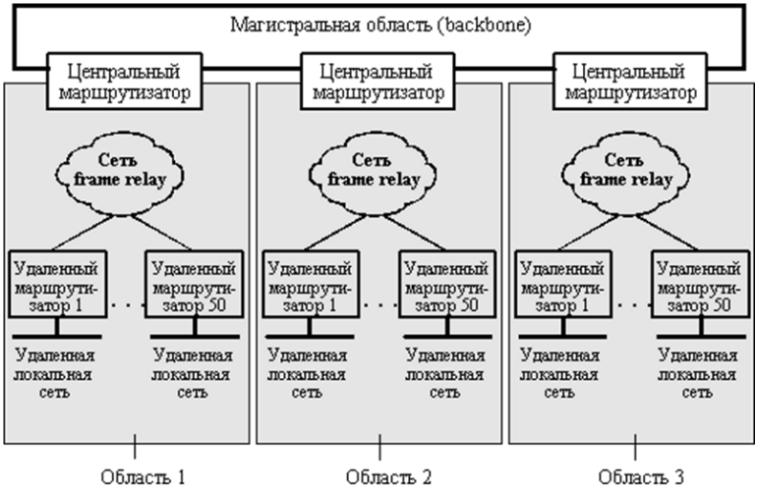


Рисунок 22. Большая сеть с топологией звезда

В протоколе OSPF определяется также *пограничный маршрутизатор области* (ABR, area border router). ABR - это маршрутизатор с интерфейсами в двух или более областях, одна из которых является специальной областью, называемой магистральной *(backbone area).* Каждая область работает с отдельной базой маршрутной информации и независимо вычисляет маршруты по алгоритму OSPF. Пограничные маршрутизаторы передают данные о топологии области в соседние области в обобщенной форме - в виде вычисленных маршрутов с их весами. Поэтому в сети, разбитой на области, уже не действует утверждение о том, что все маршрутизаторы оперируют с идентичными топологическими базами данных.

Маршрутизатор ABR берет информацию о маршрутах OSPF, вычисленную в одной области, и транслирует ее в другую область путем включения этой информации в обобщенное суммарное объявление (summary) для базы данных другой области. Суммарная информация описывает каждую подсеть области и дает для нее метрику. Суммарная информация может быть использована тремя способами: для

объявления об отдельном маршруте, для обобщения нескольких маршрутов или же служить маршрутом по умолчанию.

Дальнейшее уменьшение требований к ресурсам маршрутизаторов происходит в том случае, когда область представляет собой *тупиковую область* (stub area). Этот атрибут администратор сети может применить к любой области, за исключением магистральной. ABR в тупиковой области не распространяет внешние объявления или суммарные объявления из других областей. Вместо этого он делает одно суммарное объявление, которое будет удовлетворять любой IP-адрес, имеющий номер сети, отличный от номеров сетей тупиковой области. Это объявление называется маршрутом по умолчанию. Маршрутизаторы тупиковой области имеют информацию, необходимую только для вычисления маршрутов между собой плюс указания о том, что все остальные маршруты должны проходить через ABR. Такой подход позволяет уменьшить в нашей гипотетической сети количество маршрутной информации в удаленных офисах без уменьшения способности маршрутизаторов корректно передавать пакеты

.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каковы основные цели мониторинга сетевого трафика
2. Чем отличается мониторинг трафика от фильтрации?
3. Каково назначении класса программ-снифферов?
4. Какие основные функции выполняют снифферы?
5. Зачем используются фильтры отображения и фильтры захвата сниффера Wireshark? В чем их отличие?
6. Какие базовые функции статистической обработки захваченных пакетов имеет сниффер Wireshark?
7. Какие задачи рассчитан решать протокол ARP?

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

1. Изучить интерфейс программы Wireshark (ftp://iipo.tu-bryansk.ru/pub/Install/windows/net/wireshark/)

2. Захватить 100 произвольных пакетов. Определить статистические данные:

- процентное соотношение трафика разных протоколов в сети;
- среднюю скорость кадров/сек;
- среднюю скорость байт/сек;
- минимальный, максимальный и средний размеры пакета;
- степень использования полосы пропускания канала (загрузку сети).

3. Зафиксировать 20 IP-пакетов. Определить статистические данные:

- процентное соотношение трафика разных протоколов стека tcp/ip в сети;
- средний, минимальный, максимальный размеры пакета.

4. Выполнить анализ ARP-протокола по примеру из методических указаний.

5. На примере любого IP-пакета указать структуры протоколов Ethernet и IP, Отметить поля заголовков и описать их.

6. Проанализировать и описать принцип работы утилиты ping.

При этом описать все протоколы, используемые утилитой. Описать все поля протоколов. Составить диаграмму взаимодействия машин при работе утилиты ping.

Примечание. Данная утилита использует протокол ICMP (RFC 792 и RFC 960).

Миссия университета – генерация передовых знаний, внедрение инновационных разработок и подготовка элитных кадров, способных действовать в условиях быстро меняющегося мира и обеспечивать опережающее развитие науки, технологий и других областей для содействия решению актуальных задач.

КАФЕДРА ПРОЕКТИРОВАНИЯ И БЕЗОПАСНОСТИ КОМПЬЮТЕРНЫХ СИСТЕМ

1945–1966 РЛПУ (кафедра радиолокационных приборов и устройств).

Решением Советского правительства в августе 1945 г. в ЛИТМО был открыт факультет электроприборостроения. Приказом по институту от 17 сентября 1945 г. на этом факультете была организована кафедра радиолокационных приборов и устройств, которая стала готовить инженеров, специализирующихся в новых направлениях радиоэлектронной техники, таких как радиолокация, радиоуправление, теленаведение и др. Организатором и первым заведующим кафедрой был д. т. н., профессор С. И. Зилитинкевич (до 1951 г.). Выпускникам кафедры присваивалась квалификация инженер-радиомеханик, а с 1956 г. - радиоинженер (специальность 0705).

В разные годы кафедрой заведовали доцент Б. С. Мишин, доцент И. П. Захаров, доцент А. Н. Иванов.

1966–1970 КиПРЭА (кафедра конструирования и производства радиоэлектронной аппаратуры).

Каждый учебный план специальности 0705 коренным образом отличался от предыдущих планов радиотехнической специальности своей четко выраженной конструкторско-технологической направленностью. Оканчивающим институт по этой специальности присваивалась квалификация инженер-конструктор-технолог РЭА.

Заведовал кафедрой доцент А. Н. Иванов.

1970–1988 КиПЭВА (кафедра конструирования и производства электронной вычислительной аппаратуры).

Бурное развитие электронной вычислительной техники и внедрение ее во все отрасли народного хозяйства потребовали

от отечественной радиоэлектронной промышленности решения новых ответственных задач. Кафедра стала готовить инженеров по специальности 0648. Подготовка проводилась по двум направлениям - автоматизация конструирования ЭВА и технология микроэлектронных устройств ЭВА.

Заведовали кафедрой: д. т. н., проф. В. В.Новиков (до 1976 г.), затем проф. Г. А. Петухов.

1988–1997 МАП (кафедра микроэлектроники и автоматизации проектирования).

Кафедра выпускала инженеров, конструкторов, технологов по микроэлектронике и автоматизации проектирования вычислительных средств (специальность 2205). Выпускники этой кафедры имеют хорошую технологическую подготовку и успешно работают как в производстве полупроводниковых интегральных микросхем, так и при их проектировании, используя современные методы автоматизации проектирования. Инженеры специальности 2205 требуются микроэлектронной промышленности и предприятиям-разработчикам вычислительных систем.

Кафедрой с 1988 г. по 1992 г. руководил проф. С. А. Арустамов, затем снова проф. Г. А. Петухов.

С 1997 ПКС (кафедра проектирования компьютерных систем). Кафедра выпускает инженеров по специальности 210202 «Проектирование и технология электронно-вычислительных средств». Область профессиональной деятельности выпускников включает в себя проектирование, конструирование и технологию электронных средств, отвечающих целям их функционирования, требованиям надежности, проекта и условиям эксплуатации. Кроме того, кафедра готовит специалистов по защите информации, специальность 090104 «Комплексная защита объектов информатизации». Объектами профессиональной деятельности специалиста по защите информации являются методы, средства и системы обеспечения защиты информации на объектах информатизации.

С 1996 г. кафедрой заведует д. т. н., профессор Ю.А. Гатчин. За время своего существования кафедра выпустила 4364 инженеров. На кафедре защищено 65 кандидатских и семь докторских диссертаций.

На кафедре Проектирования Компьютерных систем осуществляется магистраторская подготовка по направлению 210200.05 «Информационные технологии и проектирование электронных средств».

2011 – переименование (кафедра проектирования и безопасности компьютерных систем)

Приказом №527-од от 07.10.2011 кафедра проектирования компьютерных систем переименована в кафедру проектирования и безопасности компьютерных систем (сокращенно —ПБКС).

С 2013 г. на кафедре ведется подготовка бакалавров по направлению 10.03.01 «Информационная безопасность» и 11.03.03 «Конструирование и технология электронных средств».

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Виснадул Б.Д., Лупин С.А., Сидоров С.В., Чумаченко П.Ю. Основы компьютерных сетей: учеб. пособие / Под ред. Л.Г. Гагариной. М.: ИД "ФОРУМ": ИНФРА-М, 2007. – 272 с.
2. Вишневский В.М. Теоретические основы проектирования компьютерных сетей. – М.: Техносфера, 2003. – 512 с.
3. Гребешков А. Ю. Стандарты и технологии управления сетями связи. – М.: Эко-Трендз, 2003. – 288 с.
4. Дымарский Я.С. Задачи и методы оптимизации сетей связи. Учеб. пособие. СПбГУТ, 2005. – 207 с.
5. Ивницкий В.А. Теория сетей массового обслуживания. – М.: Изд. физ-мат. литературы, 2004. – 772 с.
6. Йон Снейдер. Эффективное программирование TCP/IP. Библиотека программиста – СПб.: Питер, 2001. – 320 с.
7. Куроуз Дж., Росс К. Компьютерные сети. 2-е изд. – СПб.: Питер, 2004. – 765 с.
8. Одом, Уэнделл. Компьютерные сети. Первый шаг.: Пер. с англ. –М.: Издательский дом "Вильямс", 2006. – 432 с.
9. Олифер В., Олифер Н. Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов. 4-е издание – СПб.: Питер, 2010. – 944 с.
10. Филимонов А.Ю. Построение мультисервисных сетей Ethernet. – СПб.: БХВ-Петербург, 2007. – 592 с.

Куракин Александр Сергеевич
Жуковский Артем Валерьевич
Зозуля Евгений Игоревич

«Экспертные системы комплексной оценки безопасности объекта»

Учебное-методическое пособие

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе