Л4. Дополнительные возможности Wireshark для графического представления получаемых результатов

Итак, овладев основами приложения Wireshark, можете перейти к изучению его дополнительных возможностей для анализа пакетов и графического представления получаемых результатов. В этой главе будут рассмотрены некоторые из подобных возможностей и средств, включая окна Endpoints и Conversations, более сложные детали процесса преобразования имен, исследование протоколов, интерпретацию потоков, графическое представление ввода-вывода и многое другое.

Эти особые возможности Wireshark как графического средства анализа пакетов окажутся полезными на самых разных стадиях данного процесса. Поэтому постарайтесь хотя бы опробовать все рассматриваемые здесь дополнительные возможности Wireshark, прежде чем двигаться дальше, поскольку мы еще не раз вернемся к ним при рассмотрении примеров практического анализа пакетов в остальной части этой книги.

Конечные точки и сетевые диалоги

Для того чтобы данные передавались по сети, они должны перемещаться потоком хотя бы между двумя устройствами. Каждое устройство, передающее или принимающее данные в сети, представлено в приложении Wireshark так называемой *конечной точкой*. А передача данных между конечными точками

называется *диалогом*. Конечные точки и диалоги описываются в Wireshark на основании свойств передачи данных и, в частности, с помощью таких понятий, как адреса, применяемые в различных сетевых протоколах.

Конечные точки обозначаются несколькими адресами, которые присваиваются им на разных уровнях модели OSI. Например, на канальном уровне конечная точка получает MAC-адрес, который является однозначным адресом, встроенным в сетевое устройство, хотя он может быть видоизменен и тогда потенциально станет ненужным. А на сетевом уровне конечная точка получает IP-адрес, который может быть изменен в любой момент. О том, как используются оба эти типа адресов, речь пойдет в последующих главах.

На рис. 5.1 приведены два примера применения адресов для обозначения конечных точек в сетевых диалогах. В частности, диалог А состоит из двух конечных точек, обменивающихся данными на канальном уровне (по MAC-адресам). У конечной точки А имеется MAC-адрес **`00:ff:ac:ce:0b:de`**, тогда как у конечной точки Б – MAC-адрес **`00:ff:ac:e0:dc:0f`**. А диалог Б определяется двумя устройствами, обменивающимися данными на сетевом уровне (по IP-адресам). У конечной точки А имеется IP-адрес **`192.168.1.25`**, тогда как у конечной точки Б – IP-адрес **`192.168.1.30`**. Выясним, каким образом в Wireshark можно предоставить сведения о передаче данных по сети на основании конечных точек или диалогов.

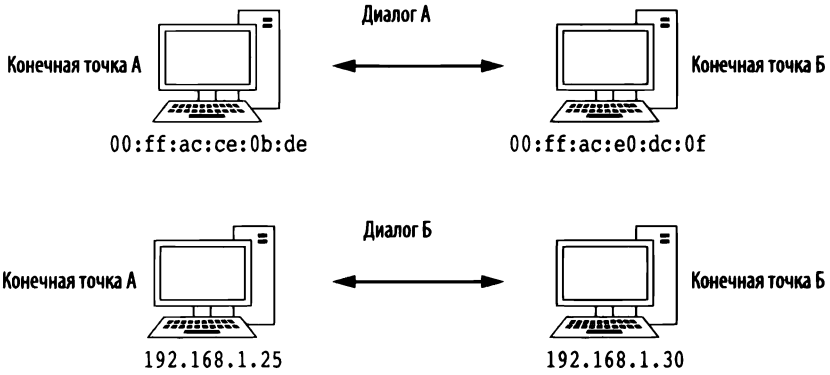


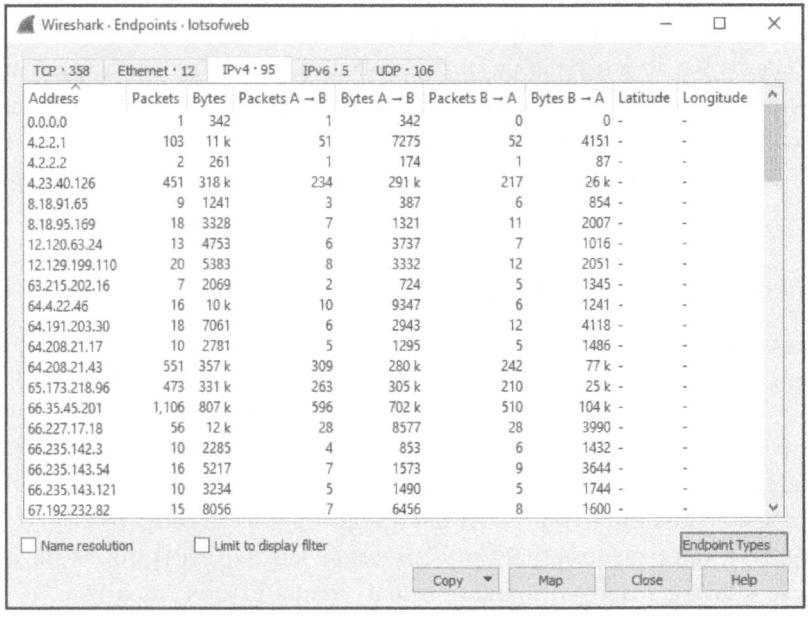
Рис. 5.1. Конечные точки и диалоги в сети

Просмотр статистики в конечных точках

Файл перехвата **`lotsofweb.pcapng`**

Анализируя сетевой трафик, можно выявить проблему в сети вплоть до конкретной конечной точки. В качестве примера откройте сначала файл перехвата `lotsofweb.pcapng`, а затем окно Endpoints, выбрав команду Statistics⇨Endpoints (Статистика⇨Конечные точки) из главного меню. В этом окне отображается ряд полез-

ных статистических данных по каждой конечной точке, включая адрес, количество переданных и принятых пакетов и байтов, как показано на рис. 5.2.



Address	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Latitude	Longitude
0.0.0.0	1	342	1	342	0	0	-	-
4.2.2.1	103	11 k	51	7275	52	4151	-	-
4.2.2.2	2	261	1	174	1	87	-	-
4.23.40.126	451	318 k	234	291 k	217	26 k	-	-
8.18.91.65	9	1241	3	387	6	854	-	-
8.18.95.169	18	3328	7	1321	11	2007	-	-
12.120.63.24	13	4753	6	3737	7	1016	-	-
12.129.199.110	20	5383	8	3332	12	2051	-	-
63.215.202.16	7	2069	2	724	5	1345	-	-
64.4.22.46	16	10 k	10	9347	6	1241	-	-
64.191.203.30	18	7061	6	2943	12	4118	-	-
64.208.21.17	10	2781	5	1295	5	1486	-	-
64.208.21.43	551	357 k	309	280 k	242	77 k	-	-
65.173.218.96	473	331 k	263	305 k	210	25 k	-	-
66.35.45.201	1,106	807 k	596	702 k	510	104 k	-	-
66.227.17.18	56	12 k	28	8577	28	3990	-	-
66.235.142.3	10	2285	4	853	6	1432	-	-
66.235.143.54	16	5217	7	1573	9	3644	-	-
66.235.143.121	10	3234	5	1490	5	1744	-	-
67.192.232.82	15	8056	7	6456	8	1600	-	-

Рис. 5.2. В окне Endpoints можно просмотреть статистические данные по каждой конечной точке в файле перехвата

На вкладках (TCP, Ethernet, IPv4, IPv6 и UDP), расположенных вверху данного окна, отображается количество конечных точек, организованных по отдельному сетевому протоколу. Чтобы отобразить конечные точки только по конкретному протоколу, щелкните на соответствующей вкладке. Чтобы добавить дополнительные вкладки для отсеивания конечных точек по отдельным протоколам, щелкните на кнопке Endpoint Types (Типы конечных точек) в правом нижнем углу экрана и выберите добавляемый сетевой протокол. Если же потребуется преобразование имен для просмотра адресов конечных точек (см. далее раздел "Преобразование имен"), установите флажок Name resolution слева внизу экрана. А если приходится анализировать крупный перехват и требуется отфильтровать отображаемые конечные точки, то можно применить фильтр отображения в главном окне Wireshark и установить флажок Limit to display filter (Наложить ограничение с помощью фильтра отображения) в окне Endpoints. Если этот флажок установлен, в данном окне будут отображаться только те конечные точки, которые соответствуют условию, заданному в фильтре отображения.

В окне Endpoints предоставляется еще одна удобная возможность отсеивать отдельные пакеты для отображения на панели Packet List. Это быстрый способ

углубленного анализа пакетов в отдельной конечной точке. С этой целью щелкните правой кнопкой мыши на конечной точке, чтобы выбрать имеющиеся варианты фильтрации. В открывшемся диалоговом окне вам представится возможность показать или скрыть пакеты, связанные с выбранной конечной точкой. Здесь же можно выбрать вариант Colorize (Выделить цветом), чтобы экспортировать адрес конечной точки непосредственно в правило выделения цветом (о правилах выделения цветом см. в главе 3, "Введение в Wireshark"). Таким образом, можно быстро выделить пакеты, связанные с отдельными конечными точками, чтобы оперативно находить их во время анализа.

Просмотр сетевых диалогов

Файл перехвата **`lotsofweb.pcapng`**

Если файл перехвата `lotsofweb.pcapng` все еще открыт, откройте окно Conversations, выбрав команду Statistics⇨Conversations (Статистика⇨Диалоги) из главного меню, чтобы отобразить все диалоги из данного файла перехвата, как показано на рис. 5.3. Окно Conversations аналогично окну Endpoints, но в нем отображаются по два адреса в каждой строке, чтобы представить сетевой диалог, а также пакеты и байты, передаваемые и принимаемые каждым устройством. В столбце **`Address A`** указывается конечная точка отправителя пакетов, а в столбце **`Address B`** – конечная точка получателя.

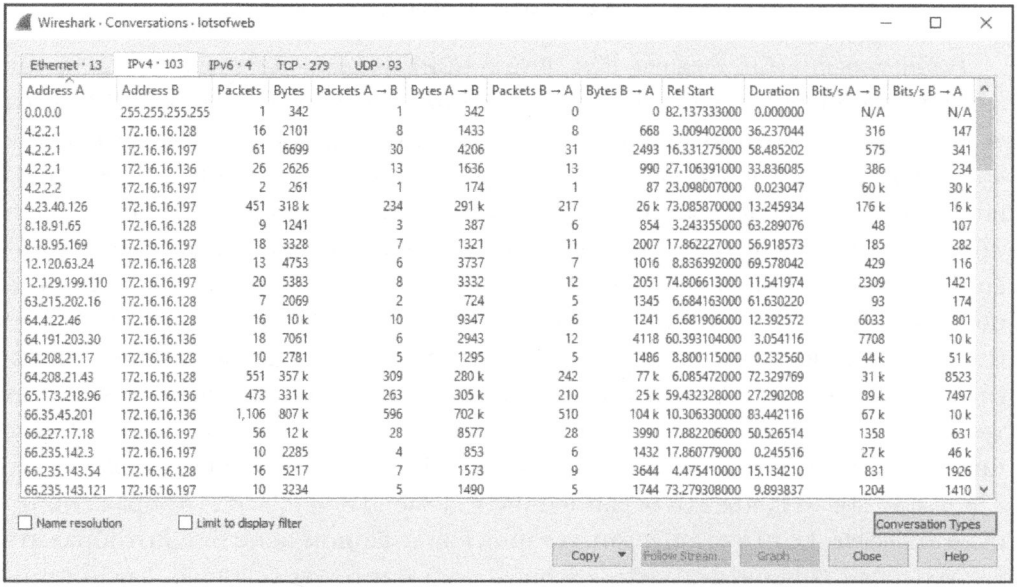


Рис. 5.3. В окне Conversations можно выделить каждый сетевой диалог из файла перехвата

Окно Conversations организовано по отдельным протоколам. Чтобы просмотреть сетевые диалоги по отдельному протоколу, щелкните на соответствующей вкладке у верхнего края данного окна или введите другие типы сетевых протоколов, щелкнув на кнопке Conversation Types справа внизу. Как и в окне Endpoints, здесь можно воспользоваться преобразованием имен, наложить ограничение на показываемые сетевые диалоги с помощью фильтра отображения, а также щелкнуть правой кнопкой мыши на отдельном диалоге, чтобы создать фильтры на основании конкретных диалогов. Фильтры, основанные на сетевых диалогах, удобны для углубленного анализа последовательностей обмена данными по сети, которые представляют особый интерес.

Выявление наиболее активных сетевых узлов с помощью конечных точек и диалогов

Файл перехвата **`lotsofweb.pcapng`**

Окнами Endpoints и Conversations удобно пользоваться при диагностике сети и особенно при попытке найти источник значительного объема сетевого трафика. Обратимся снова за примером к файлу перехвата `lotsofweb.pcapng`. Как подразумевает имя этого файла, он содержит сетевой трафик, формируемый по протоколу HTTP многими клиентами, просматривающими веб-сайты в Интернете. На рис. 5.4 приведен список конечных точек из этого файла, отсортированных по количеству байтов.

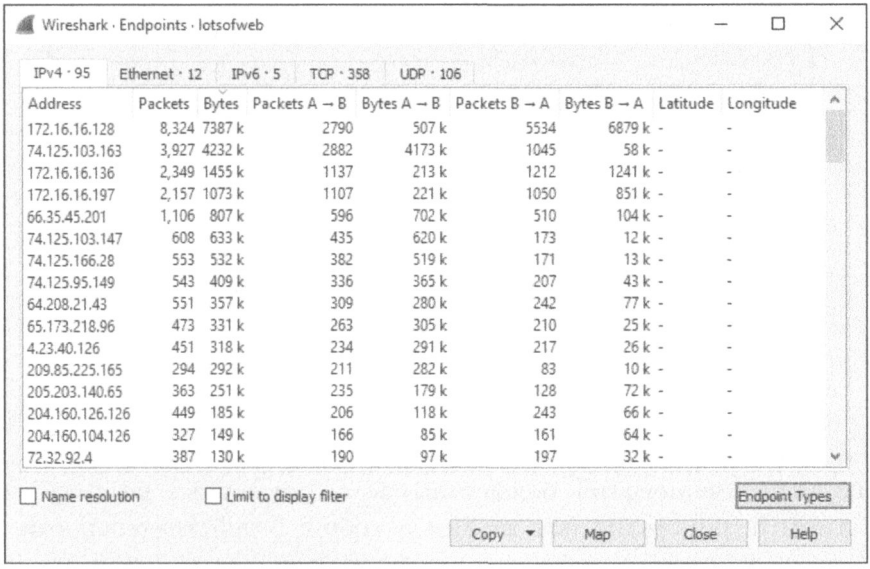


Рис. 5.4. В окне Endpoints показывается, какие сетевые узлы обмениваются данными больше всего

Обратите внимание на конечную точку, находящуюся по адресу **172.16.16.128** и служащую причиной самого интенсивного (в байтах) сетевого трафика. Адрес этой конечной точки является внутренним сетевым адресом, как поясняется в главе 7, "Протоколы сетевого уровня", а устройство, вызвавшее самый интенсивный обмен данными в перехваченном трафике, называется *наиболее активным сетевым узлом (top talker)*.

Вторая по интенсивности сетевого трафика конечная точка находится по адресу **74.125.103.163**, который является внешним (межсетевым) адресом. Если встретится внешний адрес, чтобы выяснить его сетевой узел, можно обратиться к базе данных WHOIS в поисках зарегистрированного владельца. В данном случае из реестра American Registry for Internet Numbers (ARIN – Американский регистратор интернет-адресов; `https://whois.arin.net/ui/`) выявится, что рассматриваемым здесь IP-адресом владеет компания Google (рис. 5.5).

Network	
Net Range	74.125.0.0 - 74.125.255.255
CIDR	74.125.0.0/16
Name	GOOGLE
Handle	NET-74-125-0-0-1
Parent	NET74 (NET-74-0-0-0-0)
Net Type	Direct Allocation
Origin AS	
Organization	Google Inc. (GOGL)
Registration Date	2007-03-13
Last Updated	2012-02-24
Comments	
RESTful Link	https://whois.arin.net/rest/net/NET-74-125-0-0-1
See Also	Related organization's POC records.
See Also	Related delegations.

*Рис. 5.5. Результаты поиска в реестре службы WHOIS по IP-адресу **74.125.103.163** указывают на то, что он принадлежит компании Google*

Принимая во внимание эту информацию, можно предположить одно из двух: конечные точки, находящиеся по адресам **172.16.16.128** и **74.125.103.163**, обмениваются большими массивами данных по сети с многими другими устройствами или же между собой. В действительности конечные точки, как правило, обмениваются данными между собой, что весьма характерно для самых активных пар конечных точек в сети. Чтобы убедиться в этом, откройте окно Conversations, выберите вкладку IPv4 и отсортируйте

ВЫЯВЛЕНИЕ ВЛАДЕЛЬЦЕВ IP-АДРЕСОВ СРЕДСТВАМИ WHOIS

Присваиванием IP-адресов занимаются разные органы в зависимости от их географического местоположения. В частности, ARIN отвечает за присваивание IP-адресов в Соединенных Штатах и соседних регионах, AfriNIC — на территории Африки, RIPE — в Европе, тогда как APNIC — Азии и Тихоокеанском регионе. Как правило, выявление владельца конкретного IP-адреса средствами WHOIS осуществляется на веб-сайте регистратора, отвечающего за этот IP-адрес. Разумеется, глядя на IP-адрес, трудно выяснить, какой региональный регистратор отвечает за него. Эту нелегкую задачу берут за себя веб-сайты вроде Robtex (`http://robtex.com/`), делая запрос в соответствующий реестр и предоставляя полученные результаты. Но даже если вы обратитесь к неверному регистратору, вас все равно перенаправят к верному регистратору.

список по количеству байтов. В итоге вы должны увидеть, что рассматриваемые здесь две конечные точки образуют диалог с наибольшим количеством переданных байтов. Сам характер передачи данных предполагает крупную загрузку, поскольку количество байтов, передаваемых из конечной точки с внешним адресом A (**`74.125.103.163`**), оказывается намного больше, чем количество байтов, передаваемых из конечной точки с внутренним адресом B (**`172.16.16.128`**), как показано на рис. 5.6.

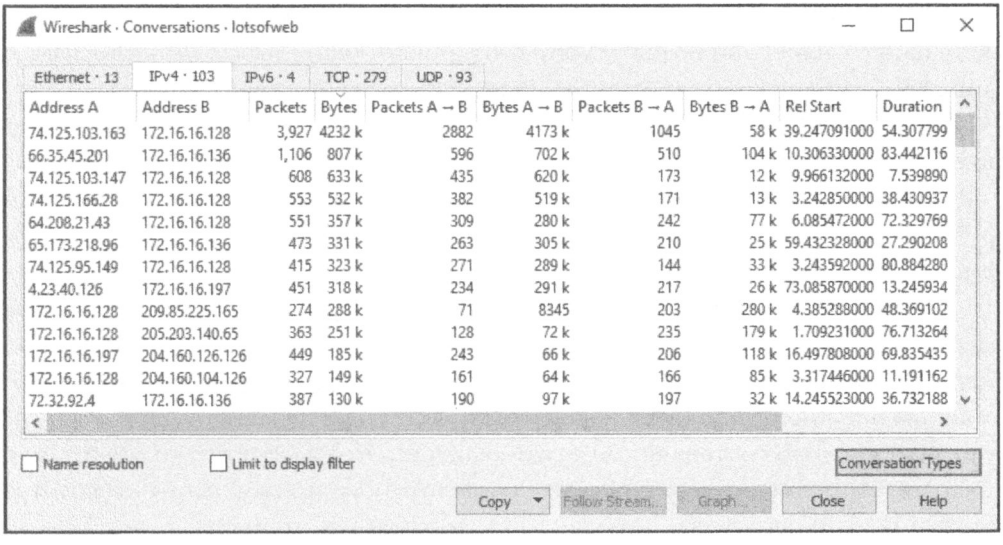


Рис. 5.6. Информация в окне Conversations подтверждает, что два наиболее активных сетевых узла обмениваются данными друг с другом

Этот сетевой диалог можно исследовать отдельно, применив следующий фильтр отображения:

```
ip.addr == 74.125.103.163 && ip.addr == 172.16.16.128
```

Прокрутив список пакетов, можно обнаружить несколько DNS-запросов к домену youtube.com в столбце **Info** на панели Packet List. Это вполне согласуется с представленными ранее выводами о том, что владельцем IP-адреса **74.125.103.163** является компания Google, поскольку ей принадлежит компания YouTube. О том, как пользоваться окнами Endpoints и Conversations в конкретных сценариях анализа пакетов, речь пойдет в остальных главах этой книги.

Статистические данные по иерархии сетевых протоколов

Файл перехвата **lotsofweb.pcapng**

Когда приходится иметь дело с незнакомыми файлами перехвата, иногда требуется выяснить распределение сетевого трафика по отдельным протоколам, т.е. ту долю в процентах, которая приходится на протокол TCP, IP, DHCP и т.д. в перехваченном трафике. Вместо того чтобы подсчитывать пакеты и подытоживать результаты вручную, эти сведения можно получить автоматически в Wireshark, открыв окно Protocol Hierarchy Statistics (Статистические данные по иерархии протоколов).

Так, если файл перехвата lotsofweb.pcapng все еще открыт, а любые применявшиеся ранее фильтры удалены, откройте окно Protocol Hierarchy Statistics (рис. 5.7), выбрав команду Statistics⇨Protocol Hierarchy из главного меню.

В окне Protocol Hierarchy Statistics предоставляется моментальный снимок того вида деятельности, который происходит в сети. Как следует из рис. 5.7, доля 100% приходится на сетевой трафик по протоколу Ethernet, 99,7% – на трафик по протоколу IPv4, 98% – на трафик по протоколу TCP и 13,5% – на трафик по протоколу HTTP, с помощью которого просматриваются веб-страницы в Интернете. Эти сведения служат отличным подспорьем для оценивания загрузки сети, особенно если имеется ясное представление о том, как обычно выглядит трафик в данной сети. Так, если заранее известно, что 10% сетевого трафика обычно приходится на протокол ARP, то из недавнего перехвата следует, что эта доля составляет 50%, это означает, что в сети что-то может быть неладно. Иногда само наличие пакетов некоторого протокола в сетевом трафике может представлять интерес. Так, если в сети отсутствуют устройства, настроенные на применение протокола STP (Spanning Tree Protocol – протокол связующего дерева), то его появление в иерархии сетевых протоколов может означать, что соответствующие устройства настроены неверно.

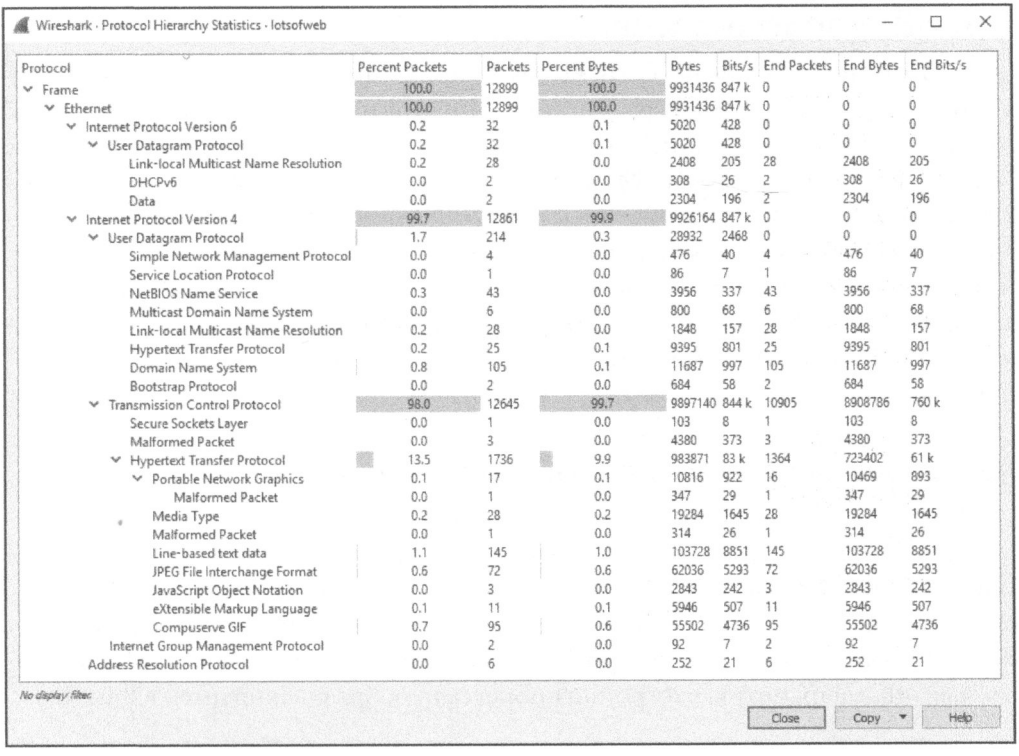


Рис. 5.7. В окне Protocol Hierarchy Statistics показывается распределение сетевого трафика по отдельным протоколам

Со временем вы обнаружите, что окном Protocol Hierarchy Statistics можно пользоваться для профилирования пользователей и устройств в сети, просто глядя на распределение применяемых протоколов. Например, большой объем сетевого трафика по протоколу HTTP может свидетельствовать об интенсивном просмотре веб-страниц в Интернете. Кроме того, вы сможете выявлять отдельные устройства в сети, просто глядя на трафик из сетевого сегмента, принадлежащего конкретному подразделению организации. Например, в отделе информационных технологий могут применяться дополнительные административные протоколы вроде ICMP или SNMP, в отделе обслуживания заказчиков – наблюдаться большой объем почтового трафика по протоколу SMTP, а вредный молодой специалист может незаметно наводнить сеть трафиком, тихо наслаждаясь в своем уголке многопользовательской ролевой игрой *World of Warcraft*!

Преобразование имен

Данные передаются по сети между конечными точками с помощью различных буквенно-цифровых систем адресации, которые зачастую оказываются слишком длинными и сложными для запоминания, например, MAC-адрес **`00:16:ce:6e:8b:24`**, адрес IPv4 **`192.168.47.122`** или адрес IPv6 **`2001:db8:a0b:12f0::1`**. Чтобы облегчить запоминание этих адресов им назначают удобные и запоминающиеся имена, а процесс нахождения адреса по имени называется *преобразованием имен* (*name resolution*). Например, запомнить адрес `google.com` намного легче, чем адрес **`216.58.217.238`**. Благодаря связыванию удобочитаемых имен с неудобочитаемыми адресами облегчается их запоминание и распознавание.

Активизация процесса преобразования имен

При отображении данных из пакетов в Wireshark можно пользоваться преобразованием имен, чтобы упростить их анализ. Чтобы воспользоваться в Wireshark преобразованием имен, выберите команду Edit⇨Preferences⇨Name Resolution (Правка⇨Глобальные параметры⇨Преобразование имен) из главного меню. В итоге откроется окно (рис. 5.8), в котором доступны основные, описываемые ниже параметры настройки процесса преобразования имен в Wireshark.

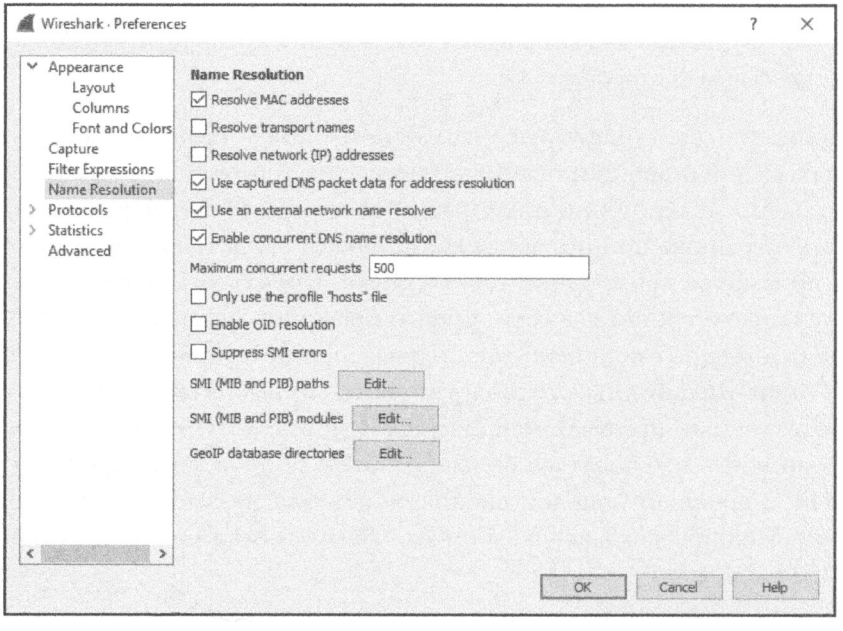


Рис. 5.8. Активизация процесса преобразования имен в окне Preferences. Среди первых трех флажков, относящихся к типам преобразования имен, здесь выбран только флажок Resolve MAC addresses

- **Resolve MAC addresses (Преобразовать MAC-адреса).** В этом режиме применяется протокол ARP, чтобы попытаться преобразовать MAC-адреса второго уровня (например, такие как `00:09:5b:01:02:03`) в адреса третьего уровня (например, `10.100.12.1`). Если подобные попытки потерпят неудачу, Wireshark выберет из своего каталога файл `ethers`, чтобы предпринять новую попытку преобразования адресов. А в качестве последнего средства Wireshark прибегнет к преобразованию первых трех байтов MAC-адреса в имя устройства, обозначаемое производителем по стандарту IEEE (например, `Netgear_01:02:03`).
- **Resolve transport names (Преобразовать транспортные имена).** В этом режиме предпринимается попытка преобразовать номер порта в связанное с ним имя протокола – например, отобразить порт **80** как **http**. Это удобно, когда в ходе анализа встречается неизвестный порт и непонятно, с какой службой он обычно связан.
- **Resolve network (IP) addresses (Преобразовать сетевые (IP) адреса).** В этом режиме предпринимается попытка преобразовать адрес третьего уровня (например, **192.168.1.50**) в удобочитаемое имя DNS вроде `MarketingPC1.domain.com`. Это удобно для выяснения назначения или владельца системы, при условии, что у нее имеется описательное имя.

В разделе Name Resolution окна Preferences, приведенного на рис. 5.8, имеются и другие глобальные, описываемые ниже параметры настройки преобразования имен.

- **Use captured DNS packet data for address resolution (Использовать данные из перехваченных пакетов DNS для преобразования адресов).** В этом режиме производится синтаксический анализ данных из перехваченных пакетов DNS для преобразования IP-адресов в имена DNS.
- **Use an external network name resolver (Использовать внешний преобразователь сетевых имен).** В этом режиме допускается формировать запросы к DNS-серверу той машиной, где выполняется анализ пакетов, для преобразования IP-адресов в имена DNS. Это удобно в том случае, если требуется воспользоваться службой DNS для преобразования имен, но анализируемый перехваченный трафик не содержит соответствующие пакеты DNS.
- **Maximum concurrent requests (Максимум параллельных запросов).** В этом режиме устанавливается ограничение на количество параллельных DNS-запросов, одновременно ожидающих обработки. Этот режим задается в том случае, если в перехваченном трафике формируется слишком много DNS-запросов, что может отрицательно сказаться на пропускной способности сети или производительности DNS-сервера.

- **Only use the profile "hosts" file (Использовать только профильные файлы сетевых узлов hosts).** В этом режиме работа системы DNS ограничивается только файлом **hosts**, связанным с активным профилем Wireshark. О том, как пользоваться этим файлом, речь пойдет в следующем разделе.

Изменения, вносимые в окне Preferences, сохраняются после закрытия и повторного открытия Wireshark. Чтобы оперативно вносить изменения в процесс преобразования имен, не сохраняя их, настраивайте параметры преобразования имен, выбирая команду View⇨Name Resolution из главного меню. У вас есть возможность активизировать или отменять преобразование имен для физических, транспортных и сетевых адресов.

Чтобы сделать файлы перехвата более удобочитаемыми, сэкономив немало времени в определенных случаях, воспользуйтесь различными инструментальными средствами преобразования имен. Например, преобразование имен DNS помогает сразу же выяснить имя компьютера, который вы пытаетесь идентифицировать как отправителя отдельного пакета.

Потенциальные недостатки преобразования имен

Принимая во внимание преимущества преобразования имен, его применение считается вполне очевидным, но у него имеется и ряд потенциальных недостатков. Прежде всего, преобразование сетевых имен может потерпеть неудачу, если отсутствует подходящий DNS-сервер для предоставления имен, связанных с IP-адресом. Сведения о преобразовании имен не сохраняются в файле перехвата, и поэтому процесс их преобразования должен происходить всякий раз, когда открывается этот файл. Если вы перехватываете пакеты в одной сети, а затем открываете файл перехвата пакетов в другой сети, то ваша система не сможет получить доступ к DNS-серверам из исходной сети, а следовательно, преобразование имен потерпит неудачу.

Кроме того, преобразование имен требует дополнительных издержек на обработку. Если приходится иметь дело с очень крупным файлом перехвата, то, возможно, придется отказаться от преобразования имен ради экономии системных ресурсов. Если же попробовать открыть крупный файл перехвата и при этом система не сможет справиться с нагрузкой или приложение Wireshark вообще завершится аварийно, в таком случае может помочь отмена преобразования имен.

Еще одно затруднение возникает, когда при преобразовании сетевых имен задействуется служба DNS. В этом случае могут быть сформированы лишние пакеты, способные засорить файл перехвата сетевым трафиком, посылаемым DNS-серверам для преобразования адресов. Дело усложняется еще и тем, что если анализируемый файл перехвата содержит зловредные IP-адреса, то

попытки преобразовать их в имена могут привести к формированию запросов к инфраструктуре, которая контролируется атакующим злоумышленником и может предупредить его о том, что о его злых намерениях сделать данную систему своей мишенью известно. Чтобы снизить риск засорения файла перехвата пакетов или непреднамеренного связывания с атакующим злоумышленником, сбросьте флажок Use an external network name resolver в разделе Name Resolution окна Preferences.

Применение специального файла hosts

Отслеживать трафик из многих сетевых узлов в крупных файлах перехвата может оказаться довольно утомительным занятием, особенно в том случае, когда преобразование имен недоступно. В таком случае может, в частности, помочь разметка систем вручную на основании их IP-адресов с помощью специального файла `hosts`, создаваемого для сетевых узлов (или хостов) в Wireshark. Это текстовый файл, содержащий список IP-адресов и соответствующих имен. Файл `hosts` может быть использован для назначения в Wireshark соответствующих имен адресам для краткой ссылки. Эти имена будут отображены на панели Packet List.

Чтобы воспользоваться файлом `hosts`, выполните следующие действия.

1. Выберите команду Edit⇨Preferences⇨Name Resolution из главного меню и установите флажок Only use the profile "hosts" file.
2. Создайте новый файл в с помощью программы Notepad в системе Windows или аналогичном текстовом редакторе. Этот файл должен содержать по одной записи в каждой строке с IP-адресом и тем именем, в которое он преобразуется (рис. 5.9). Имя, выбираемое справа в каждой строке, будет отображаться в списке пакетов всякий раз, когда в Wireshark встречается IP-адрес, указываемый слева.

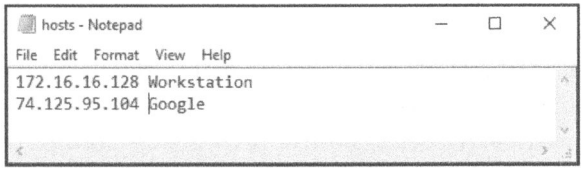


*Рис. 5.9. Создание файла **hosts** в Wireshark*

3. Сохраните созданный файл как обычный текстовый файл под именем **hosts** в соответствующем каталоге, как показано ниже. Обратите внимание, что у файла **hosts** нет расширения!
   - Windows:
   - `<USERPROFILE>\Application Data\Wireshark\hosts`.

- Mac OS X:
- /Users/<*имя_пользователя*>/.wireshark/hosts.
- Linux: /home/<username>/.wireshark/hosts.

Откройте далее файл перехвата, и тогда любые IP-адреса из файла hosts должны быть обозначены соответствующими именами, как показано рис. 5.10. Вместо IP-адресов в столбцах **Source** (Отправитель) и **Destination** (Получатель) на панели Packet List появятся более описательные имена.

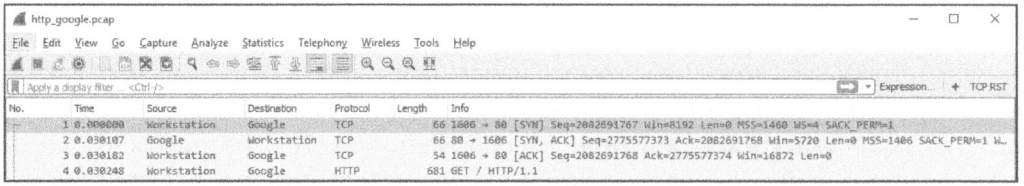


Рис. 5.10. Преобразование имен из специального файла ***hosts*** *в Wireshark*

Такое применение специальных файлов hosts может значительно улучшить вашу способность распознавать определенные сетевые узлы (или хосты) во время анализа пакетов. Работая в команде сетевых аналитиков, рассмотрите возможность обмениваться со своими коллегами файлом hosts, содержащим известные ресурсы. Это поможет вашей команде быстро распознавать системы со статическими адресами, в том числе серверы и маршрутизаторы.

ПРИМЕЧАНИЕ *Если ваш файл* **hosts** *окажется неработоспособным, убедитесь, не дополнили ли вы случайно его имя расширением файла* **.txt**. *Этот файл должен просто носить имя* **hosts** *без всякого расширения!*

Инициируемое вручную преобразование имен

В приложении Wireshark имеется также возможность по требованию активизировать на время преобразование имен. Для этого достаточно щелкнуть правой кнопкой мыши на пакете в панели Packet List и выбрать команду Edit Resolved Name (Изменить преобразуемое имя) из контекстного меню. В открывшемся окне можно указать имя, соответствующее конкретному адресу, как метку. Такое преобразование будет утрачено, как только файл перехвата будет закрыт, хотя это краткий способ пометить адрес, не внося никаких постоянных изменений, которые придется отменять впоследствии. Я часто пользуюсь данным способом, поскольку это немного проще, чем править файл hosts вручную при анализе каждого файла с перехваченными пакетами.

Дешифрирование сетевых протоколов

К числу самых сильных сторон Wireshark относится поддержка анализа тысяч сетевых протоколов. И такая возможность объясняется тем, что Wireshark является приложением с открытым исходным кодом, а следовательно, служит основанием для создания *дешифраторов протоколов*. Они дают возможность распознавать и декодировать различные поля сетевого протокола в Wireshark, чтобы отобразить сетевой протокол в пользовательском интерфейсе. Для интерпретации каждого пакета в Wireshark совместно используется несколько дешифраторов. Например, дешифратор сетевого протокола ICMP позволяет Wireshark выяснить, что IP-пакет содержит данные из протокола ICMP, извлечь тип и код протокола ICMP и отформатировать его поля для отображения в столбце **Info** панели Packet List.

Дешифратор можно рассматривать как интерпретатор исходных данных в приложении Wireshark. Чтобы сетевой протокол нашел поддержку в Wireshark, для него должен быть отдельный дешифратор в данном приложении, а иначе вам придется написать свой дешифратор сетевых протоколов.

Смена дешифратора

Файл перехвата **wrongdissector.pcapng**

Дешифраторы применяются в Wireshark для того, чтобы обнаружить отдельные протоколы и выяснить, как отобразить сетевую информацию. К сожалению, Wireshark далеко не всегда делает правильный выбор дешифратора для применения к пакетам. И это особенно справедливо, когда в сетевом протоколе применяется нестандартная конфигурация, в том числе нестандартный порт, который зачастую настраивается сетевыми администраторами из соображений безопасности или сотрудниками организации, пытающимися обойти средства управления доступом.

Если дешифраторы неверно применяются в Wireshark, их выбор можно переопределить. Например, откройте файл трассировки `wrongdissector.pcapng`, содержащий немало сведений об обмене данными между двумя компьютерами по сетевому протоколу SSL (Secure Socket Layer – уровень защищенных сокетов), применяемый для шифрованного обмена данными между хостами. При обычных условиях просмотр сетевого трафика по протоколу SSL в Wireshark не даст особенно полезных сведений в силу того, что они зашифрованы. Но здесь определенно что-то не так. Если вы просмотрите содержимое нескольких таких пакетов, щелкнув на них и исследовав содержимое панели Packet Bytes, то обнаружите сетевой трафик, представленный простым текстом. Так, если проанализировать пакет 4, то в нем можно обнаружить

упоминание о приложении FileZilla для работы с FTP-сервером. А в ряде следующих пакетов ясно показан запрос и ответ как имени пользователя, так и пароля.

Если бы это был сетевой трафик по протоколу SSL, вы не смогли бы прочитать любые данные, содержащиеся в пакетах, и вряд ли увидели бы все имена пользователей и пароли, передаваемые в виде открытого текста (рис. 5.11). Принимая во внимание представленные здесь сведения, можно с уверенностью предположить, что это скорее трафик по протоколу FTP, а не SSL. Этот трафик, скорее всего, будет интерпретирован в Wireshark как относящийся к протоколу SSL, поскольку в нем употребляется порт **443**, как следует из сведений, приведенных в столбце **Info**. А ведь это стандартный порт, используемый в сетевом протоколе HTTPS (т.е. надстройке HTTP над SSL).

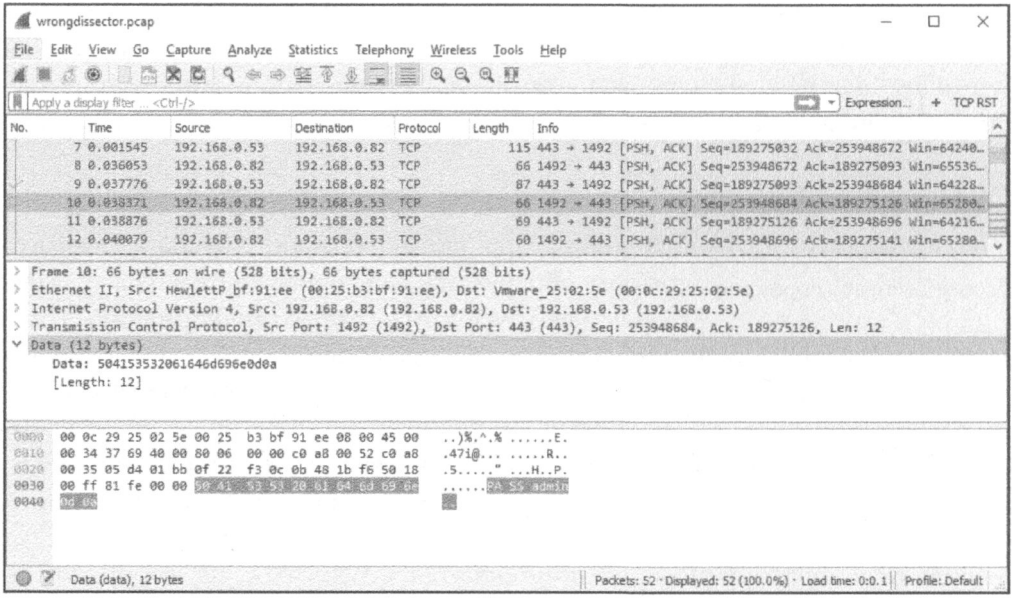


Рис. 5.11. Если имена пользователей и пароли представлены в виде открытого текста, это похоже скорее на протокол FTP, а не SSL!

Чтобы устранить подобное затруднение, можно применить в Wireshark *принудительную расшифровку* с целью воспользоваться дешифратором протокола FTP к анализируемым пакетам, выполнив следующие действия.

1. Щелкните правой кнопкой мыши на избранном пакете SSL (например, на пакете 30) в столбце **Protocol** и выберите из контекстного меню команду Decode As (Расшифровать как), чтобы открыть новое диалоговое окно.

2. Дайте Wireshark команду расшифровывать весь трафик TCP, проходящий через порт **443**, как FTP, выбрав вариант TCP port из списка в столбце **Field** (Поле), введя номер порта **443** в столбце **Value** (Значение) и выбрав вариант FTP из списка в столбце **Current** (Текущий протокол), как показано на рис. 5.12.

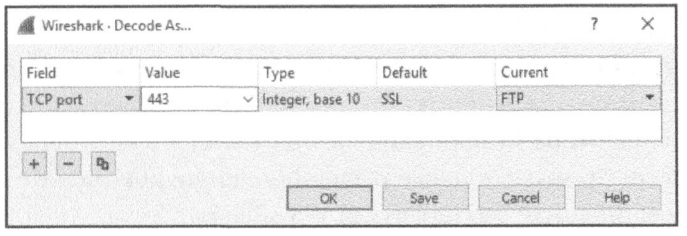


Рис. 5.12. В диалоговом окне Decode As... можно задать условия для принудительной расшифровки анализируемых пакетов

3. Щелкните на кнопке OK, чтобы увидеть изменения, немедленно внесенные в файл перехвата.

Данные будут декодированы как трафик по протоколу FTP, что даст возможность проанализировать их на панели Packet List, не особенно вдаваясь в отдельные байты (рис. 5.13).

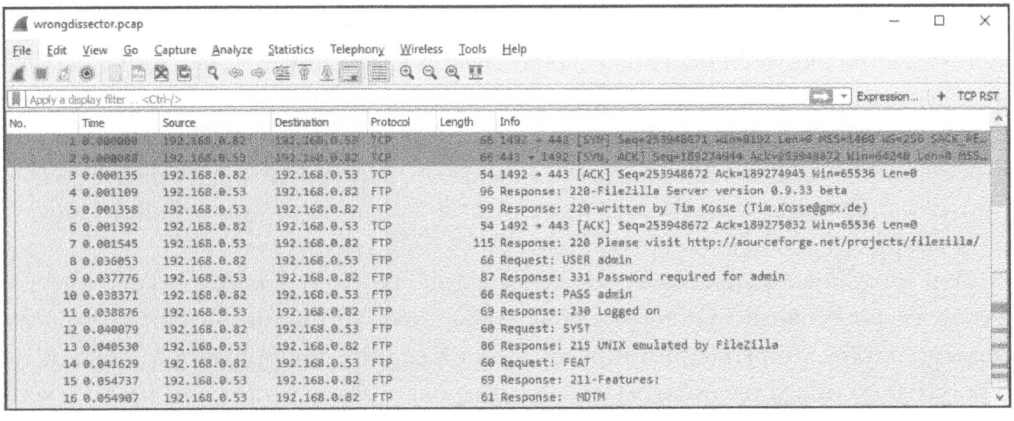


Рис. 5.13. Просмотр правильно расшифрованного трафика FTP

Возможностью принудительной расшифровки можно пользоваться многократно в одном и том же файле перехвата. Принудительные расшифровки, установленные в диалоговом окне Decode As..., будут автоматически отслеживаться в Wireshark. В этом окне можно просматривать и изменять все созданные до сих пор принудительные расшифровки.

По умолчанию принудительные расшифровки не сохраняются при закрытии файла перехвата. Но это положение можно исправить, щелкнув на кнопке Save в диалоговом окне Decode As.... В итоге правила расшифровки протоколов будут сохранены в профиле текущего пользователя Wireshark. Они будут применяться из данного профиля при открытии любого файла перехвата. Сохраненные правила расшифровки можно удалить, щелкнув на кнопке со знаком "минус" (-) в данном окне.

Сохраненные правила расшифровки можно легко забыть, что может привести к большому недоразумению для тех, кто к этому не готов, поэтому обращайтесь с правилами расшифровки благоразумно. Чтобы уберечься от подобной оплошности, принудительные расшифровки не рекомендуется сохранять в своем главном профиле пользователя Wireshark.

Просмотр исходного кода дешифраторов

Прелесть работы в приложении с открытым исходным кодом состоит в том, что при возникновении каких-нибудь недоразумений можно всегда просмотреть исходный код и выяснить их причину. И это особенно удобно при попытке выяснить причины, по которым конкретный протокол интерпретируется неверно. Ведь для этого достаточно проанализировать исходный код соответствующего дешифратора.

Просмотреть и проанализировать исходный код дешифраторов сетевых протоколов можно непосредственно на веб-сайте, посвященном приложению Wireshark, щелкнув сначала на ссылке Develop (Разработка), а затем на ссылке Browse the Code (Просмотр кода). По этой ссылке произойдет переход к хранилищу исходного кода Wireshark, где можно просмотреть код выпуска последних версий Wireshark. В частности, исходные файлы дешифраторов сетевых протоколов находятся в каталоге `epan/dissectors`, где исходный файл каждого дешифратора обозначен именем `packet-<`*`имя_протокола`*`>.c`.

Эти исходные файлы могут оказаться довольно сложными, но все они соответствуют общему шаблону и снабжены довольно подробными комментариями. Чтобы понять принцип действия каждого дешифратора, совсем не обязательно обладать опытом программирования на языке C. Если же требуется досконально разобраться в том, что отображается в Wireshark, рекомендуется начать просмотр и анализ с дешифраторов самых простых сетевых протоколов.

Отслеживание потоков

Файл перехвата **`http_google.pcapng`**

К числу наиболее удовлетворяющих потребностям анализа пакетов относится возможность повторно соби-

рать в Wireshark данные из многих пакетов в едином удобочитаемом формате, нередко называемом *выпиской из пакетов*. Благодаря этому отпадает необходимость просматривать данные, посылаемые от клиента к серверу, мелкими фрагментами при переходе от одного пакета к другому. При *отслеживании потока* данные сортируются с целью упростить их просмотр.

Ниже перечислены типы потоков, которые можно отслеживать.

- **Поток TCP.** В этот поток собираются данные из тех протоколов, где применяется протокол TCP, например, из сетевых протоколов HTTP и FTP.
- **Поток UDP.** В этот поток собираются данные из тех протоколов, где применяется протокол UDP, например, из сетевого протокола DNS.
- **Поток SSL.** В этот поток собираются данные из тех протоколов, где они шифруются. Для дешифровки сетевого трафика необходимо предоставить соответствующие ключи.
- **Поток HTTP.** В этот поток собираются данные из протокола HTTP. Это удобно для отслеживания данных HTTP через поток TCP без полной расшифровки полезной информации из протокола HTTP.

В качестве примера рассмотрим простую транзакцию по протоколу HTTP в файле перехвата `http_google.pcapng`. С этой целью щелкните сначала на любом из пакетов TCP или HTTP в этом файле, затем щелкните правой кнопкой мыши на выбранном пакете и выберите команду Follow⇨TCP Stream (Отслеживать⇨Поток TCP) из контекстного меню. В итоге будет получен единый поток TCP и открыта выписка из диалога в отдельном окне, как показано на рис. 5.14.

Текст, отображаемый в окне Follow TCP Stream, выделен двумя цветами: красным (более светлым оттенком серого на рис. 5.14) — текст, обозначающий сетевой трафик, проходящий от отправителя к получателю, а синим (более темным оттенком серого на рис. 5.14) — текст, обозначающий сетевой трафик, проходящий в противоположном направлении: от получателя к отправителю. Цвет связан с той стороной, которая инициировала обмен данными. В данном примере установление сетевого соединения с веб-сервером инициировал клиент, и поэтому его трафик выделен красным цветом.

Обмен данными в потоке TCP начинается с исходного запроса по методу GET корневого каталога (/) на веб-сервере и продолжается ответом сервера в форме `HTTP/1.1 200 OK` об успешной обработке запроса. По тому же самому образцу происходит обмен данными и в других потоках перехваченных пакетов по мере того, как клиент запрашивает отдельные файлы, а сервер присылает их в ответ. В данном примере можно увидеть, что пользователь просматривает начальную страницу веб-сайта Google. Но вместо того чтобы просматривать

пакеты по очереди, можно без труда прокрутить выписку из пакетов. По существу, вы видите то же самое, что и конечный пользователь, но только изнутри.

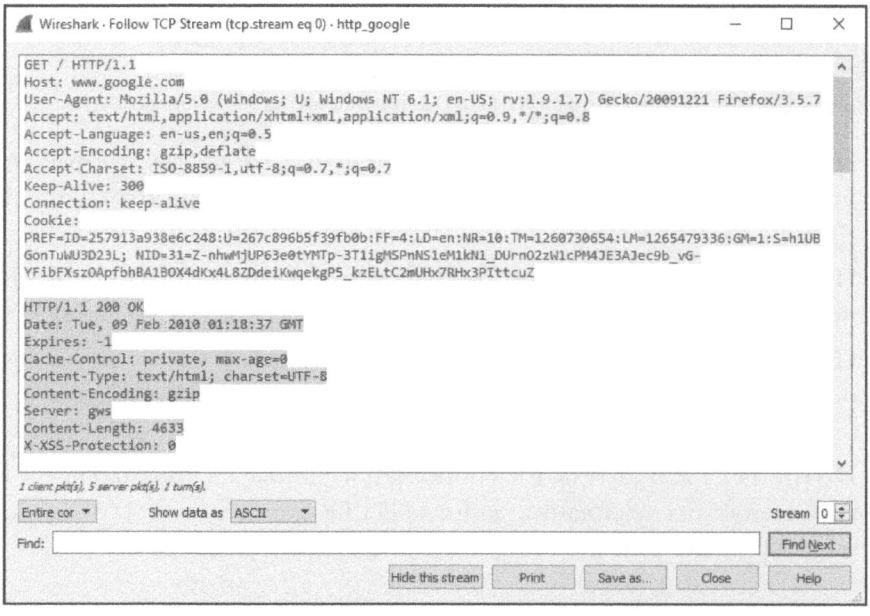


Рис. 5.14. В окне Follow TCP Stream повторно собранные передаваемые данные представлены в удобочитаемом формате

Помимо просмотра исходных данных, в этом окне можно производить поиск в тексте, сохранять его в файле, выводить на печать или выбирать представление данных в коде ASCII, EBCDIC, шестнадцатеричном виде или в форме массива на языке C. Все эти возможности, упрощающие анализ крупных массивов данных, находятся в нижней части данного окна.

Отслеживание потоков SSL

Чтобы отследить потоки TCP и UDP, достаточно выполнить всего пару щелчков, но для просмотра потоков SSL в удобочитаемом формате придется выполнить ряд дополнительных действий. Сетевой трафик по протоколу SSL зашифрован, поэтому необходимо предоставить секретный ключ, связанный с сервером, отвечающим за шифрованный трафик. Методика получения такого ключа зависит от конкретной серверной технологии, и поэтому ее рассмотрение выходит за рамки этой книги. Но как только вы получите секретный ключ, загрузите его в Wireshark, выполнив следующие действия.

1. Перейдите к глобальным параметрам настройки Wireshark, выбрав команду Edit⇨Preferences из главного меню.

2. Разверните раздел Protocols в открывшемся окне и щелкните на заголовке протокола **SSL**, как показано на рис. 5.15. Щелкните на кнопке Edit рядом с меткой RSA keys list (Список ключей RSA).

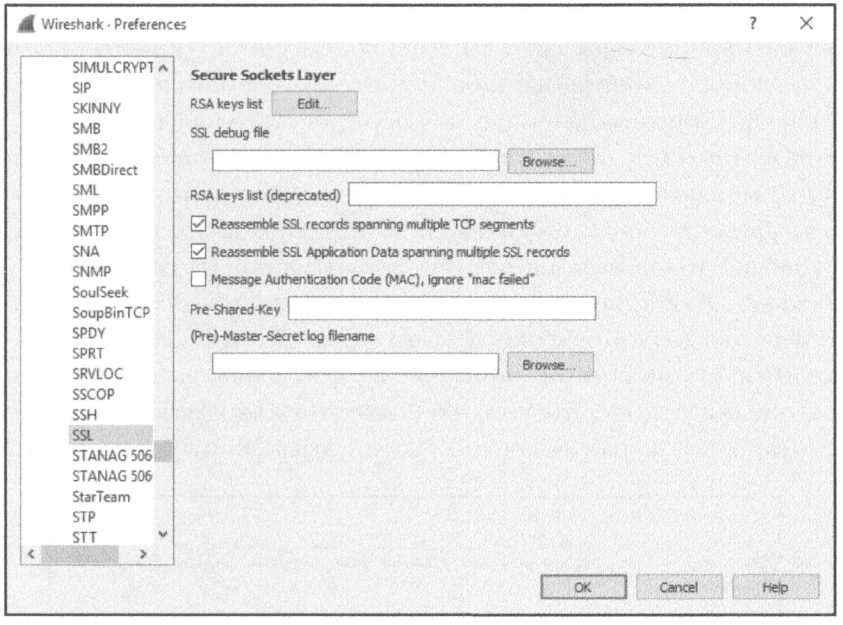


Рис. 5.15. Ввод сведений о дешифровке по протоколу SSL

3. Щелкните на кнопке со знаком "плюс" (+).
4. Предоставьте требующиеся сведения. К их числу относится IP-адрес сервера, отвечающего за шифрование, номер порта, сетевой протокол, местонахождение файла ключей и пароль к этому файлу, если таковой используется.
5. Перезапустите Wireshark.

В итоге у вас должна появиться возможность перехватывать зашифрованный сетевой трафик, проходящий между клиентом и сервером. Щелкните правой кнопкой мыши на пакете HTTPS и выберите команду Follow⇨SSL Stream (Отслеживать⇨Поток SSL), чтобы увидеть выписку из расшифрованного текста пакетов.

Возможность просматривать выписки из пакетов относится к числу наиболее употребительных при анализе пакетов в Wireshark, и вам придется ею часто пользоваться, чтобы быстро выяснить, какие сетевые протоколы при этом применяются. В последующих главах книги будет рассмотрен ряд других сценариев, основанных на просмотре выписок из пакетов.

Длина пакетов

Файл перехвата **download-slow.pcapng**

Размер одного пакета или группы пакетов может сообщить немало интересного о сложившейся ситуации в сети. При обычных обстоятельствах максимальный размер фрейма в сети Ethernet составляет 1518 байт. Если вычесть из этой числовой величины заголовки протоколов Ethernet, IP и TCP, то останется 1460 байт, предназначенных для передачи заголовка протокола седьмого уровня или данных. Если известны минимальные требования к передаче пакетов, то можно начать с анализа распределения длин пакетов в перехваченном трафике, чтобы сделать обоснованное предположение о составе данного трафика. Это очень помогает при попытках понять состав крупных файлов перехвата. Для просмотра распределения пакетов по длине в Wireshark предоставляется диалоговое окно Packet Lengths (Длины пакетов).

Обратимся к конкретному примеру из файла перехвата `download-slow.pcapng`. Открыв его, выберите команду Statistics⇨Packet Lengths из главного меню. В итоге откроется диалоговое окно Packet Lengths, приведенное на рис. 5.16.

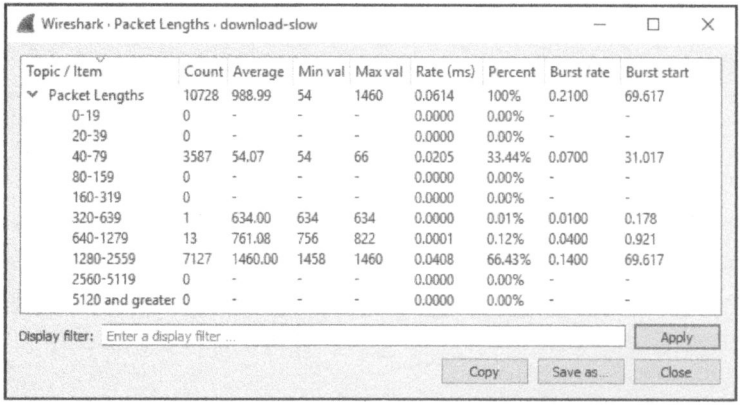


Рис. 5.16. Диалоговое окно Packet Lengths помогает сделать обоснованное предположение о сетевом трафике в файле перехвата

Обратите особое внимание на строку со статическими данными о пакетах длиной от 1280 до 2559 байт. Крупные пакеты вроде этих обычно свидетельствуют о передаче данных, тогда как более мелкие пакеты — о последовательностях управления протоколами. В данном случае наблюдается немалая доля крупных пакетов (66,43%). Даже не просматривая пакеты в файле перехвата, можно сделать вполне обоснованное предположение, что перехваченный трафик содержит одну или несколько передач данных, которые могут принимать форму загрузки по протоколу HTTP, выгрузки по протоколу FTP или любой другой операции в сети, где данные передаются между хостами.

Длина большинства оставшихся пакетов (33,44%) находится в пределах от 40 до 79 байт. К этой категории обычно относятся управляющие пакеты TCP, не несущие полезную информацию. Рассмотрим типичный размер заголовков протоколов. Так, заголовок протокола Ethernet занимает 14 байт (плюс 4 байта на циклический избыточный код CRC), заголовок протокола IP – как минимум 20 байт, а пакет TCP без данных или параметров – те же 20 байт. Это означает, что длина стандартных управляющих пакетов TCP (например, пакетов SYN, ACK, RST и FIN) составит около 54 байт и укладывается в рассматриваемые здесь пределы. Разумеется, эта длина увеличится, если добавить параметры протокола IP или TCP. Более подробно сетевые протоколы IP и TCP рассматриваются в главах 7, "Протоколы сетевого уровня", и 8, "Протоколы транспортного уровня", соответственно.

Анализ длин пакетов позволяет составить общее представление о крупном перехваченном трафике. Если в нем имеется много крупных пакетов, то можно с уверенностью предположить, что в сети передается большой объем данных. Если же длина большинства пакетов невелика, это означает, что передается немного данных, и можно предположить, что перехваченный трафик состоит из команд управления сетевыми протоколами. Но это не правила, которые следует считать непреложными, а всего лишь предположения, помогающие приступить к более углубленному анализу.

Составление графиков

Графики служат основой анализа пакетов и одним из лучших способов получения итогового представления о массиве данных. В состав Wireshark входит несколько средств для составления графиков, помогающих лучше понять перехваченные данные. И в первую очередь – это возможности графического представления ввода-вывода.

Просмотр графиков ввода-вывода

Файлы перехвата **download-fast.pcapng**, **download-slow.pcapng**, **http_espn.pcapng**

В окне IO Graph (График ввода-вывода) предоставляется возможность построить график передачи данных в сети. Такие графики позволяют быстро выявлять всплески и провалы в работе канала связи, обнаруживать задержки в производительности отдельных протоколов и сравнивать параллельные потоки данных.

В качестве примера построения графика ввода-вывода при загрузке файла на компьютер из Интернета обратимся к файлу перехвата `download-fast.pcapng`. Откройте этот файл, щелкните на любом пакете TCP, чтобы выбрать его, и выберите команду Statistics⇨IO Graph из главного меню.

В открывшемся окне IO Graph появится графическое представление потока данных во времени. Как следует из примера, приведенного на рис. 5.17, на данном графике загрузки файла передается около 500 пакетов в секунду. И этот показатель остается постоянным почти до конца графика, где он резко снижается.

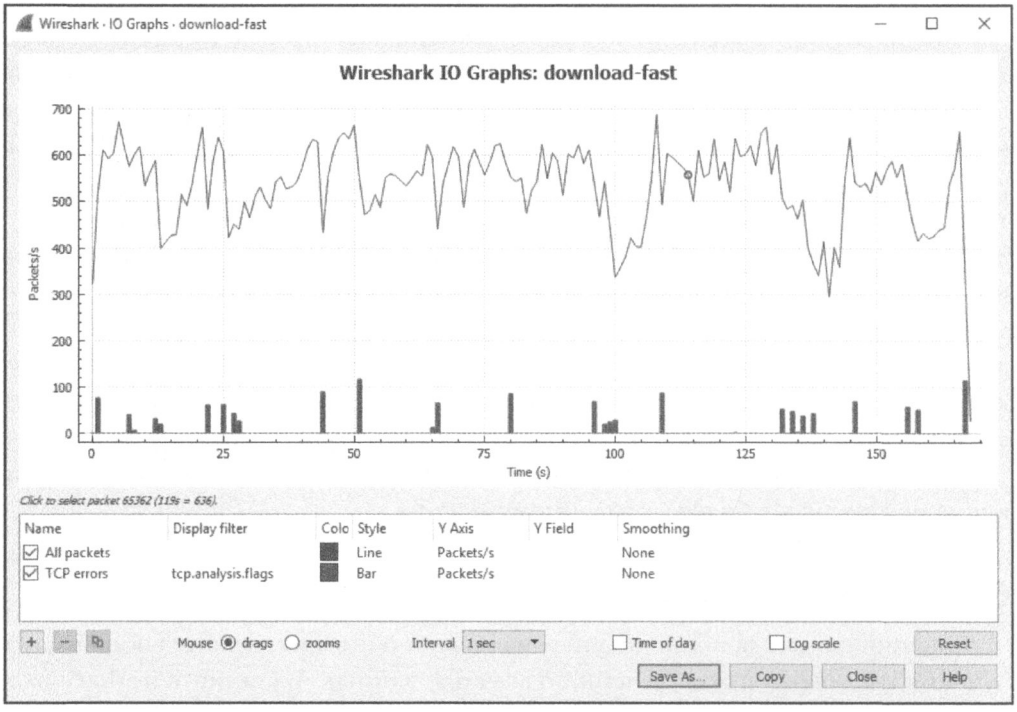


Рис. 5.17. Этот график ввода-вывода демонстрирует практически постоянную скорость передачи пакетов при быстрой загрузке файла

А теперь рассмотрим для сравнения пример более медленной загрузки файла. Оставив открытым текущий файл перехвата, откройте файл перехвата `download-slow.pcapng` в другом экземпляре Wireshark. Составив график ввода-вывода для данного примера загрузки файла описанным выше способом, вы увидите совсем другую картину, как показано на рис. 5.18.

Рассматриваемая здесь загрузка происходит со скоростью от 15 до 100 пакетов в секунду, и такая скорость далека от постоянной, а иногда даже падает до нуля пакетов в секунду. Подобное непостоянство можно лучше увидеть, если расположить рядом графики ввода-вывода обоих файлов, как показано на рис. 5.19. Сравнивая оба графика, обратите особое внимание на значения, откладываемые по осям X и Y, чтобы сравнивать сопоставимые величины. Масштаб в обоих случаях автоматически корректируется в зависимости от количества пакетов и/или объема переданных данных, что составляет главное

отличие сравниваемых графиков. Так, медленная загрузка файла демонстрируется в масштабе от 0 до 100 пакетов в секунду, тогда как быстрая загрузка – от 0 до 700 пакетов в секунду.

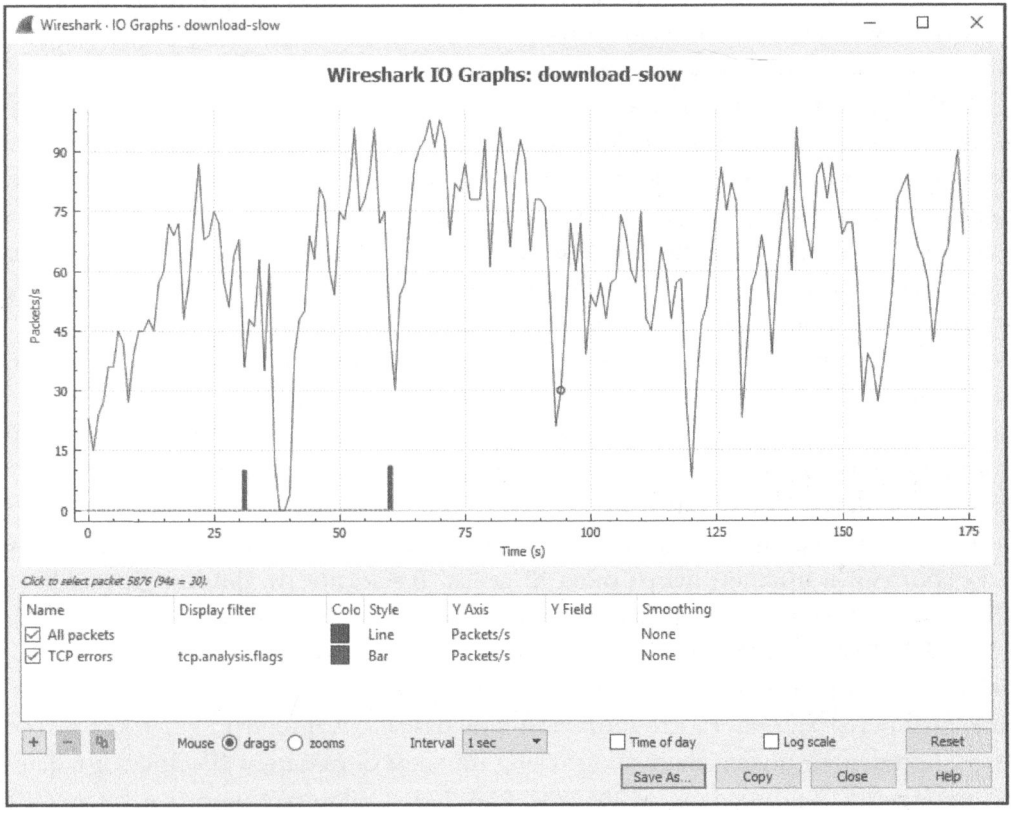


Рис. 5.18. Этот график ввода-вывода демонстрирует непостоянную скорость передачи пакетов при медленной загрузке файла

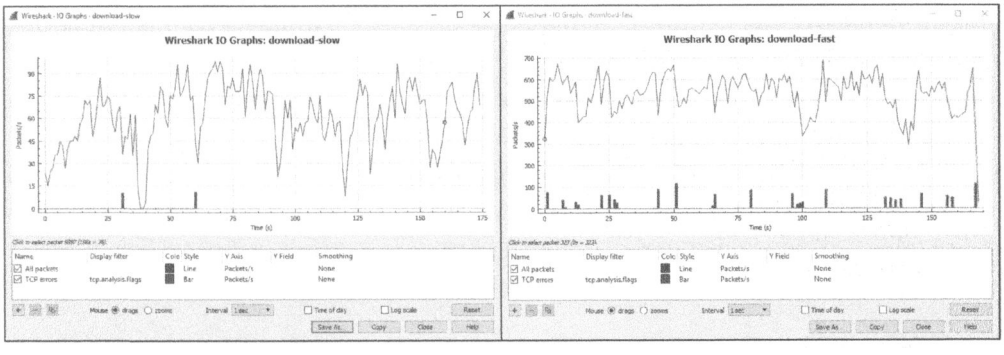


Рис. 5.19. Просмотр двух расположенных рядом графиков ввода-вывода может помочь в выявлении отклонений

Параметры, настраиваемые в нижней части окна IO Graph, позволяют применять ряд особых фильтров, составляемых с помощью того же самого синтаксиса, что и для фильтров отображения, а также выбирать цвета для отображения данных из этих фильтров. Например, можно создать фильтры конкретных IP-адресов и назначить для них особые цвета для просмотра отклонений в пропускной способности каждого устройства.

Опробуйте такую возможность, открыв файл перехвата `http_espn.pcapng`, который был получен при посещении начальной страницы американского кабельного спортивного телеканала ESPN из анализируемого устройства. В окне Conversations вы обнаружите, что наиболее активным оказывается сетевой узел с внешним IP-адресом **`205.234.218.129`**. Из этого можно сделать вывод, что данный сетевой узел, вероятнее всего, служит основным поставщиком содержимого, которое получается при посещении веб-страницы по адресу `espn.com`. Но в диалоге принимают участие и сетевые узлы по другим IP-адресам, вероятнее всего, потому, что дополнительное содержимое загружается из внешних его поставщиков и рекламодателей. Отличия в прямой и сторонней доставке содержимого можно продемонстрировать с помощью графика ввода-вывода, приведенного на рис. 5.20.

Оба фильтра, применяемых на этом графике, представлены отдельными строками в нижней части окна IO Graph. В частности, фильтр `Top Talker` (Наиболее активный сетевой узел) показывает ввод-вывод только по IP-адресу **`205.234.218.129`** основного в данном случае поставщика содержимого. Объем этого ввода-вывода отображается на графике черным цветом, заполняющим верхнюю часть столбиковой диаграммы. А фильтр `Everything Else` (Все остальные) показывает ввод-вывод по всем остальным IP-адресам в файле перехвата, кроме адреса **`205.234.218.129`**. Следовательно, он включает в себя всех сторонних поставщиков содержимого. Объем этого ввода-вывода отображается на графике красным цветом (светлым оттенком серого на рис. 5.20), заполняющим нижнюю часть столбиковой диаграммы. Обратите внимание на то, что единицы измерения по оси Y данного графика были изменены на байты в секунду. С учетом этих изменений очень легко увидеть отличия в объеме трафика основного и сторонних поставщиков содержимого, а также выяснить, сколько содержимого поступает из стороннего источника. Вам, вероятно, будет любопытно повторить это упражнение на часто посещаемых вами веб-сайтах и взять эту полезную стратегию на вооружение для сравнения объемов ввода-вывода в разных хостах.

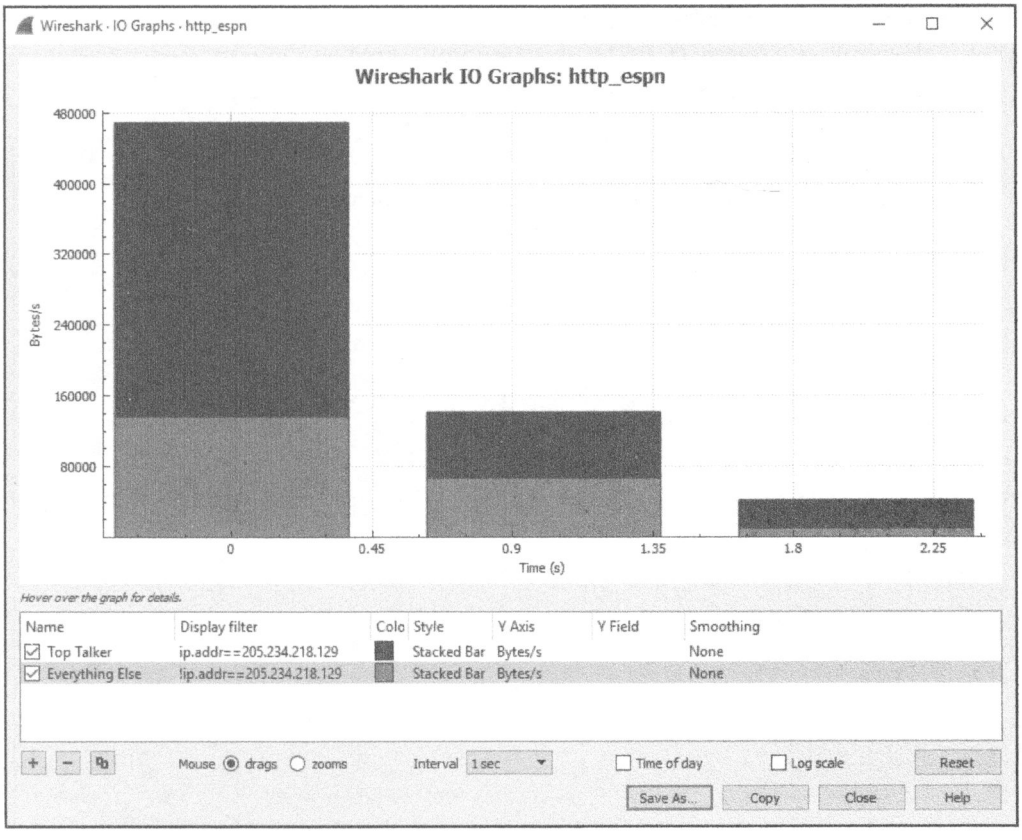


Рис. 5.20. График, демонстрирующий отличия ввода-вывода в двух отдельных устройствах

Составление графика времени круговой передачи пакетов

Файл перехвата **`download-fast.pcapng`**

В приложении Wireshark имеется также возможность составлять и просматривать график времени круговой (round-trip) передачи пакетов из заданного файла перехвата. *Время круговой передачи* (Round-Trip Time, RTT) — это время, которое требуется на получение подтверждения доставки пакета адресату. По существу, это время, которое требуется, чтобы пакет достиг получателя, а отправленное обратно подтверждение его получения — отправителя этого пакета. Анализ времени круговой передачи пакета зачастую выполняется с целью выявить места замедления или узкие места в передаче данных, а также любые задержки, возникающие в этой связи.

Чтобы опробовать такую возможность на конкретном примере, откройте файл перехвата `download-fast.pcapng`. Чтобы просмотреть график времени круговой передачи пакетов из этого файла, выберите сначала любой пакет

TCP, а затем команду Statistics⇨TCP Stream Graphs⇨Round Trip Time Graph (Статистика⇨Графики потоков TCP⇨График времени круговой передачи) из главного меню. В итоге появится график, приведенный на рис. 5.21.

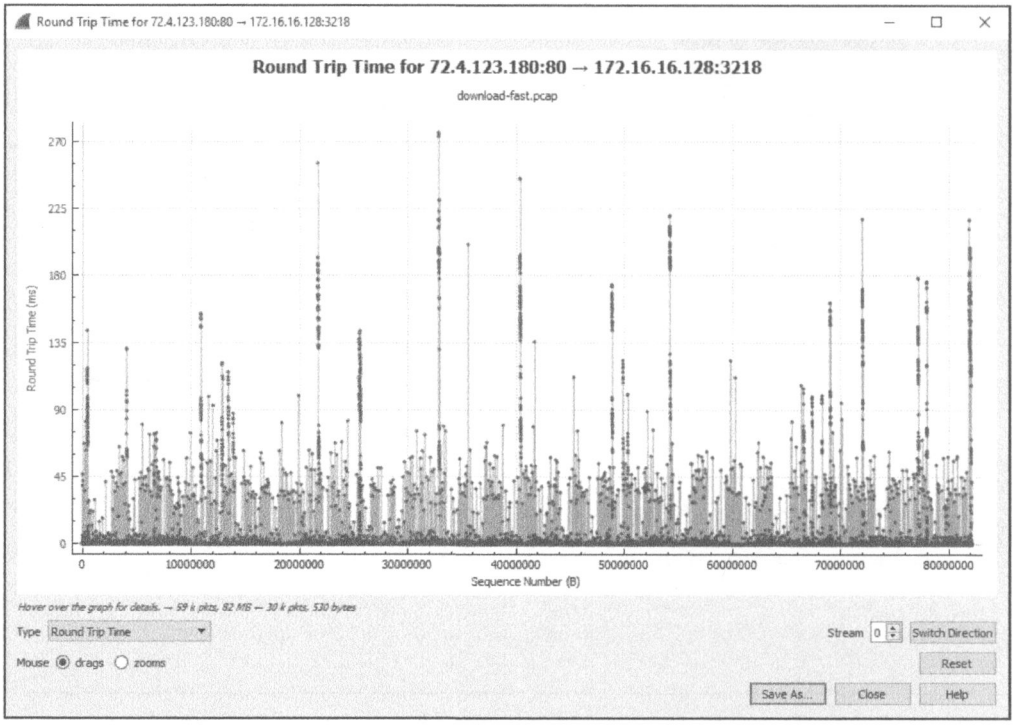


Рис. 5.21. График времени круговой передачи пакетов при быстрой загрузке файла демонстрирует практическое постоянство этой временной характеристики, за исключением нескольких случайных отклонений

Каждая точка на этом графике представляет время на передачу и подтверждение приема пакета. По умолчанию эти величины отсортированы по порядковым номерам пакетов. Чтобы перейти непосредственно к пакету на панели Packet List, достаточно щелкнуть на соответствующей точке графика.

ПРИМЕЧАНИЕ *График времени круговой передачи пакетов носит однонаправленный характер, поэтому очень важно выбрать для анализа нужное направление сетевого трафика. Если у вас этот график выглядит иначе, чем на рис. 5.21, попробуйте дважды щелкнуть на кнопке* Switch Direction *(Сменить направление).*

На этом графике, построенном для быстрой загрузки файла, величины времени круговой передачи пакетов в основном оказываются меньше 0,05 с,

и лишь в некоторых точках они находятся в пределах от 0,10 до 0,25 с. И несмотря на немалое количество больших величин, они в основном обозначают вполне допустимое для загрузки файла время на передачу и подтверждение приема. Анализируя график времени круговой передачи пакетов с точки зрения пропускной способности сети, следует выявлять большие величины времени задержки, обозначаемые множеством точек при больших значениях, откладываемых по оси Y.

Составление графиков потоков

Файл перехвата **dns_recursivequery_server.pcapng**

Возможность составлять графики потоков оказывается полезной для наглядного представления сетевых соединений и передачи потоков данных во времени. Подобные сведения облегчают понимание характера обмена данными между устройствами в сети. График потоков состоит из столбцов, обозначающих соединение между хостами, наглядно представляя сетевой трафик для удобства его интерпретации.

Чтобы создать график потоков, откройте файл перехвата `dns_recursivequery_server.pcapng` и выберите команду Statistics⇨Flow Graph (Статистика⇨График потоков) из главного меню. Полученный в итоге график приведен на рис. 5.22.

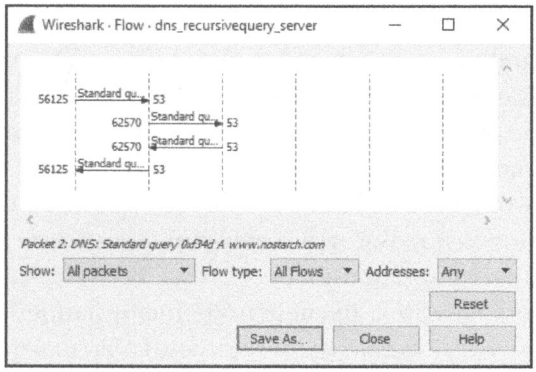


Рис. 5.22. График потоков TCP позволяет намного лучше представить наглядно сетевое соединение

На этом графике потоков наглядно представлен рекурсивный DNS-запрос, получаемый одним хостом и пересылаемый другому хосту (подробнее о протоколе DNS речь пойдет в главе 9, "Распространенные протоколы верхнего уровня"). Каждая вертикальная линия на этом графике обозначает отдельный хост. График потоков позволяет наглядно представить двухсторонний обмен данными между двумя устройствами в сети, а в данном примере — соотношение

обмена данными между несколькими устройствами. Такой график полезен и для понимания обычного потока данных, передаваемых по менее известным из вашего опыта сетевым протоколам.

Экспертная информация

Файл перехвата **download-slow.pcapng**

В дешифраторах каждого протокола в Wireshark определена *экспертная информация*, предупреждающая о конкретных состояниях в пакетах данного протокола. Эти состояния разделяются на следующие категории.

- **Chat (Текстовый диалог).** Основные сведения об обмене данными.
- **Note (Уведомление).** Необычные пакеты, которые могут быть частью обычного обмена данными.
- **Warning (Предупреждение).** Необычные пакеты, которые, вероятнее всего, не являются частью обычного обмена данными.
- **Error (Ошибка).** Ошибка в пакете или интерпретирующем его дешифраторе.

В качестве примера откройте файл перехвата `download-slow.pcapng` и выберите команду Analyze⇨Expert Information (Анализ⇨Экспертная информация) из главного меню, чтобы открыть окно Expert Information. Установите в этом окне флажок Group by summary (Группировать по сводке), чтобы организовать вывод экспертной информации по степени ее важности (рис. 5.23).

В этом окне имеются разделы по каждой категории экспертной информации. В данном случае ошибки отсутствуют, имеются 3 предупреждения, 19 уведомлений и 3 текстовых диалога.

Большинство сообщений из данного файла перехвата связаны с сетевым протоколом TCP просто потому, что к данному протоколу традиционно применялась экспертная информационная система. В то же время в данном окне отображается 29 сообщений с экспертной информацией, настроенных на протокол TCP, и они могут оказаться полезными для диагностики файлов перехвата. Эти сообщения помечают отдельные пакеты, когда они удовлетворяют определенным критериям, как перечислено ниже. (Это означает, что подобные сообщения станут более понятными при изучении сетевого протокола TCP в главе 8, "Протоколы транспортного уровня", и диагностики медленных сетей в главе 11, "Меры борьбы с медленной сетью".

- **Сообщения текстового диалога.**

  Сообщение `"Window Update"` (Изменение размера окна), посылаемое получателем, чтобы уведомить отправителя об изменении размера окна приема в протоколе TCP.

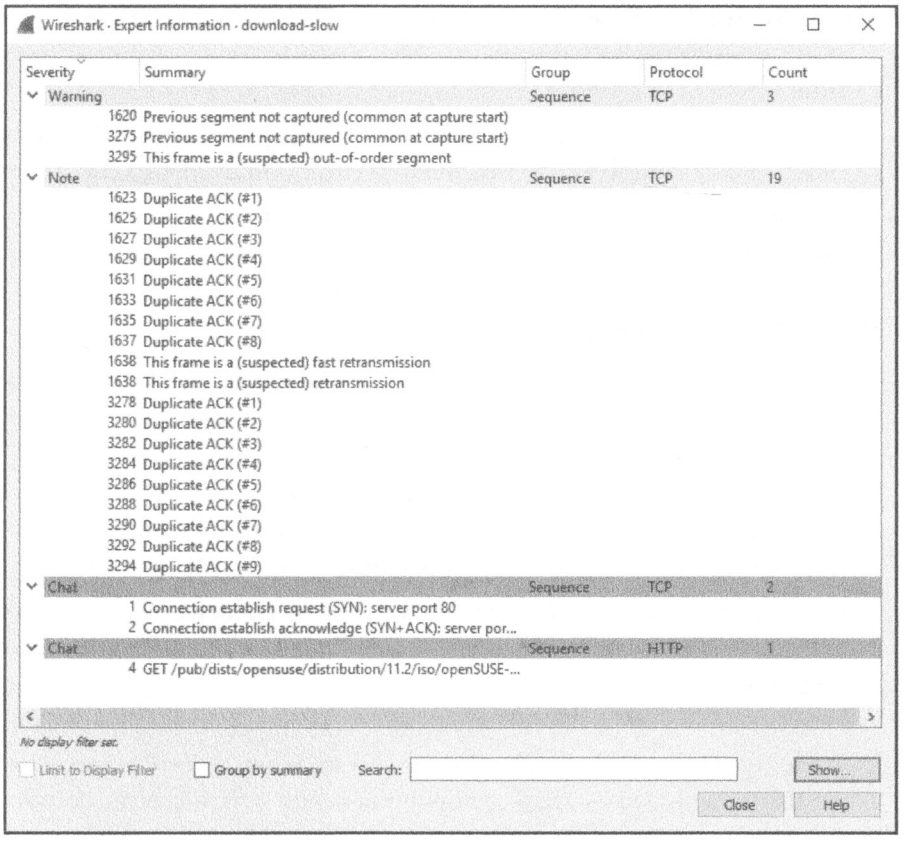


Рис. 5.23. В окне Expert Information отображаются сведения из экспертной системы, запрограммированной в дешифраторах сетевых протоколов

- **Сообщения с уведомлениями.**

  Сообщение "`TCP Retransmission`" (Повторная передача по протоколу TCP), посылаемое в результате потери пакетов. Оно появляется, когда получен дубликат подтверждения приема пакета или сработал таймер времени ожидания повторной передачи пакетов.

  Сообщение "`Duplicate ACK`" (Дубликат подтверждения), посылаемое в том случае, если хост не получает пакет с ожидаемым следующим порядковым номером и формирует дубликат подтверждения последних полученных данных.

  Сообщение "`Zero Window Probe`" (Проба нулевого окна), посылаемое в ходе текущего контроля состояния окна приема в протоколе TCP после передачи пакета с нулевым окном, как поясняется в главе 11, "Меры борьбы с медленной сетью".

Сообщение "Keep Alive ACK" (Подтверждение активности соединения), посылаемое в ответ на пакеты поддержания активным соединения.

Сообщение "Zero Window Probe ACK" (Подтверждение пробы нулевого окна), посылаемое в ответ на пакеты с пробой нулевого окна.

Сообщение "Window Is Full" (Окно заполнено), посылаемое для уведомления о заполнении на стороне получателя окна приема в протоколе TCP.

- **Предупреждающие сообщения.**

  Сообщение "Previous Segment Lost" (Предыдущий сегмент потерян), посылаемое в том случае, если пропущен пакет с ожидаемым порядковым номером в потоке данных.

  Сообщение "ACKed Lost Packet" (Подтверждение потери пакета), посылаемое в том случае, если обнаружен пакет подтверждения ACK, но не пакет, который он подтверждает.

  Сообщение "Keep Alive" (Поддержание активным соединения), посылаемое в том случае, если обнаружен пакет поддержания активным соединения.

  Сообщение "Zero Window" (Нулевое окно), посылаемое в том случае, если достигнут размер окна приема в протоколе TCP и послано уведомление о нулевом окне, запрашивающее отправителя остановить передачу данных.

  Сообщение "Out-of-Order" (Нарушение порядка следования), посылаемое в том случае, если на основании порядковых номеров обнаружено, что пакеты получаются не по порядку следования их номеров.

  Сообщение "Fast Retransmission" (Быстрая повторная передача), посылаемое в том случае, если повторная передача пакета происходит в течение 20 мс после получения дубликата подтверждения.

- **Сообщения об ошибках.**

  Сообщение "No Error Messages" (Сообщения об ошибках отсутствуют).

На первый взгляд может показаться, что некоторые возможности Wireshark, рассматриваемые в этой главе, пригодны лишь в особых, малоизвестных ситуациях, но на самом деле вам придется пользоваться ими чаще, чем вы могли бы предположить. Ознакомиться с рассмотренными здесь возможностями и окнами очень важно потому, что к ним придется еще не раз обращаться в ряде последующих глав.