

Практическая работа №8 «Изучение средств мониторинга и анализа сетевого трафика. Сниффер Wireshark»

ЦЕЛЬ РАБОТЫ:

1. Знать принципы анализа сетевого трафика.
2. Научиться использовать сетевой анализатор (сниффер Wireshark).
3. Научиться анализировать сетевой трафик на примере протоколов ARP, IP и ICMP.

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.

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.

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. Некоторое из них:

- Tcpdump (<http://www.tcpdump.org/>) – консольный вариант сниффера. Портитован 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.

СНИФФЕР WIRESHARK

Программа Wireshark является одной из самых удобных реализаций sniffеров. Портирована на большое количество платформ. Распространяется абсолютно бесплатно. Имеет смысл использовать данный sniffer для изучения и анализа сетевых протоколов.

Но для начала рассмотрим базовый принцип работы sniffера как раз на примере Wireshark.

Базовый принцип работы sniffеров

Давайте рассмотрим с вами рис. 1. 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 sniffера работают в пользовательском режиме.

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

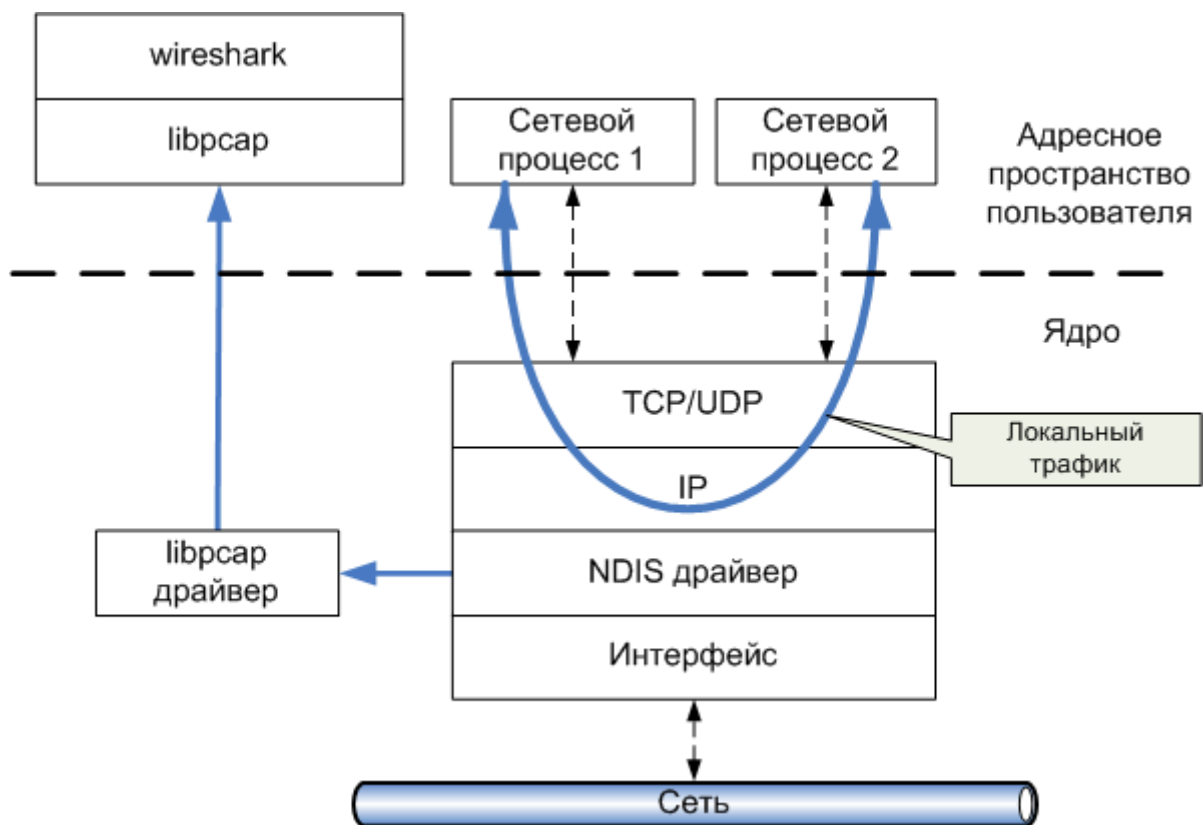


Рис. 1. Принцип «захвата» sniffером сетевого трафика

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!

Основной нюанс использования сниффера заключается в том, что он не позволяет производить анализ локального трафика, т.к. он не проходит через драйвер сетевого устройства (см. рис 1.). 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 драйвера как и в нормальной ситуации при взаимодействии с другими физическими сетевыми машинами.

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 тем, 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.

Также, 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.).

Использование программы Wireshark

Данный сниффер 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).

- пункт назначения (Destination);
- протокол (Protocol);
- информационное поле (Info).

Список отображаемых 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 пометить 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.

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. рис. 3). 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 случаев подходит выбор интерфейса сетевого адаптера.

В качестве дополнительных параметров захвата можно указать следующие:

- «*Capture Filter*» – фильтр захвата (будем рассматривать далее). 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».

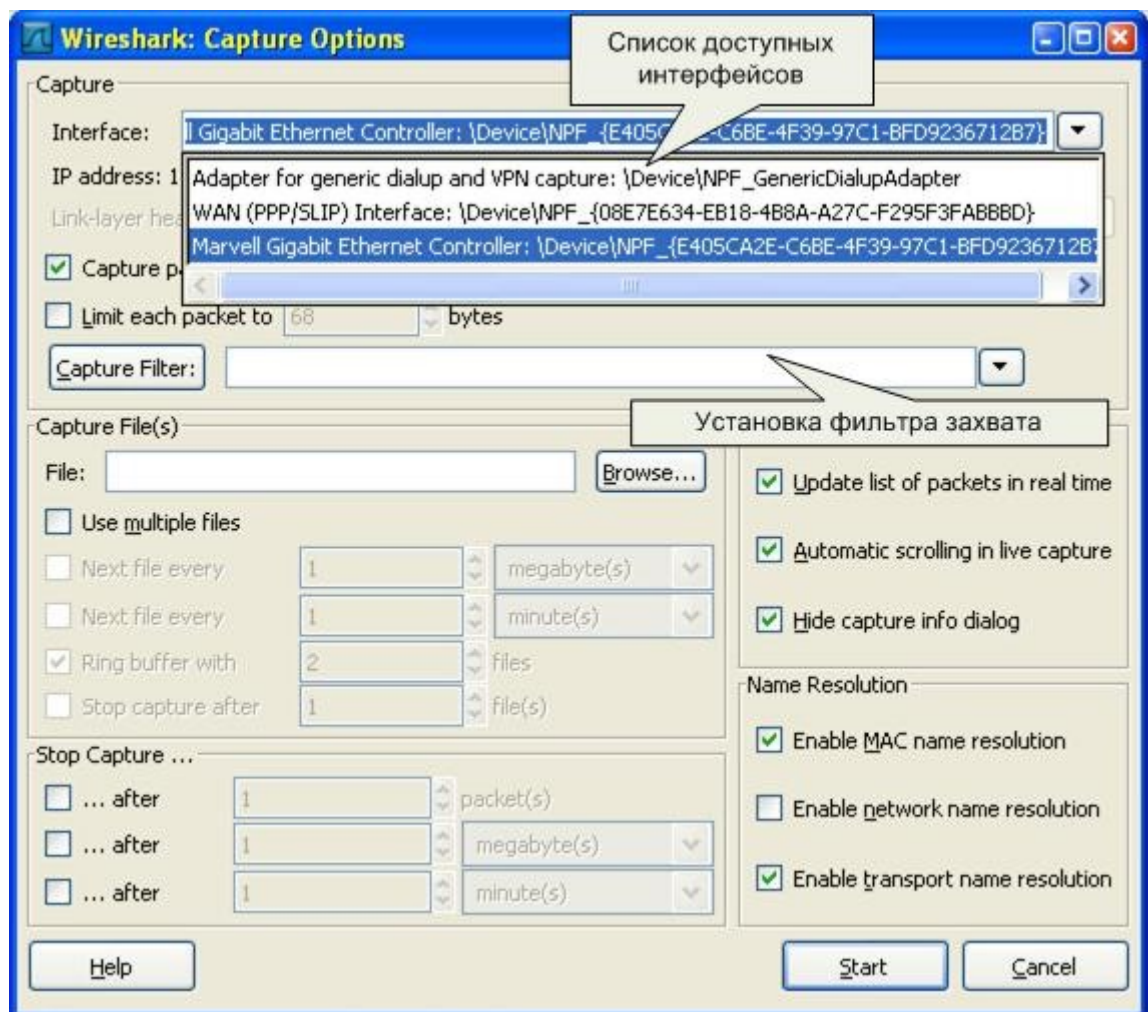


Рис. 3. Выбор интерфейса и параметров захвата пакетов

ФИЛЬТРАЦИЯ ПАКЕТОВ

Если запустить сниффер без дополнительных настроек, он будет «захватывать» все пакеты, проходящие через сетевые интерфейсы (см. рис. 1). 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 быть захвачены и сохранены либо в память, либо на дисковое устройство. Самые простые подсчеты могут 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. Рассмотрим

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.

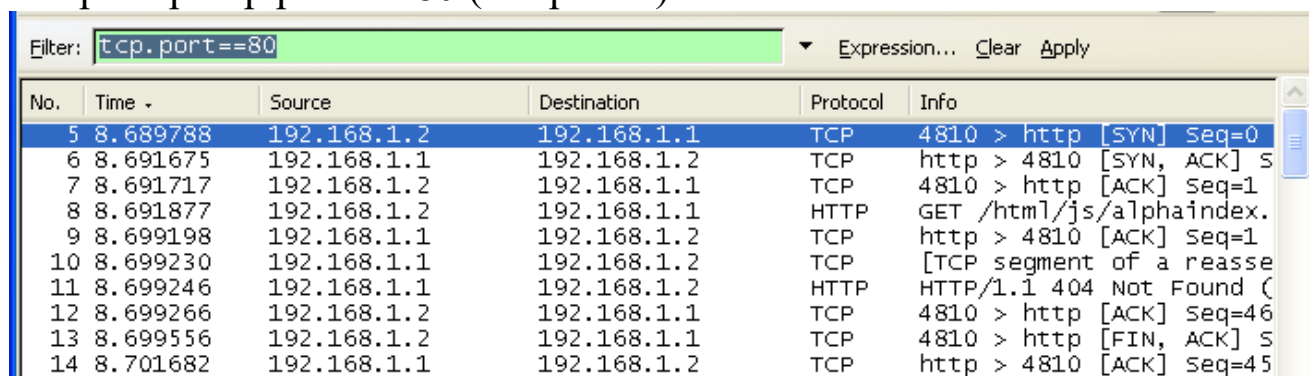
Для работы с фильтрами можно вызвать окно «Analyze/Display Filters». Можно сохранять созданные выражения под определенными именами для последующего использования и т.д.

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

Например: tcp.port == 80 (см. рис. 4).



No.	Time	Source	Destination	Protocol	Info
5	8.689788	192.168.1.2	192.168.1.1	TCP	4810 > http [SYN] Seq=0
6	8.691675	192.168.1.1	192.168.1.2	TCP	http > 4810 [SYN, ACK] S
7	8.691717	192.168.1.2	192.168.1.1	TCP	4810 > http [ACK] Seq=1
8	8.691877	192.168.1.2	192.168.1.1	HTTP	GET /html/js/alphaindex.
9	8.699198	192.168.1.1	192.168.1.2	TCP	http > 4810 [ACK] Seq=1
10	8.699230	192.168.1.1	192.168.1.2	TCP	[TCP segment of a reasse
11	8.699246	192.168.1.1	192.168.1.2	HTTP	HTTP/1.1 404 Not Found (
12	8.699266	192.168.1.2	192.168.1.1	TCP	4810 > http [ACK] Seq=46
13	8.699556	192.168.1.2	192.168.1.1	TCP	4810 > http [FIN, ACK] S
14	8.701682	192.168.1.1	192.168.1.2	TCP	http > 4810 [ACK] Seq=45

Рис. 4. Пример задания простого фильтра отображения

Мастер построения фильтров отображения доступен через кнопку «Expression...» (см. рис. 5).

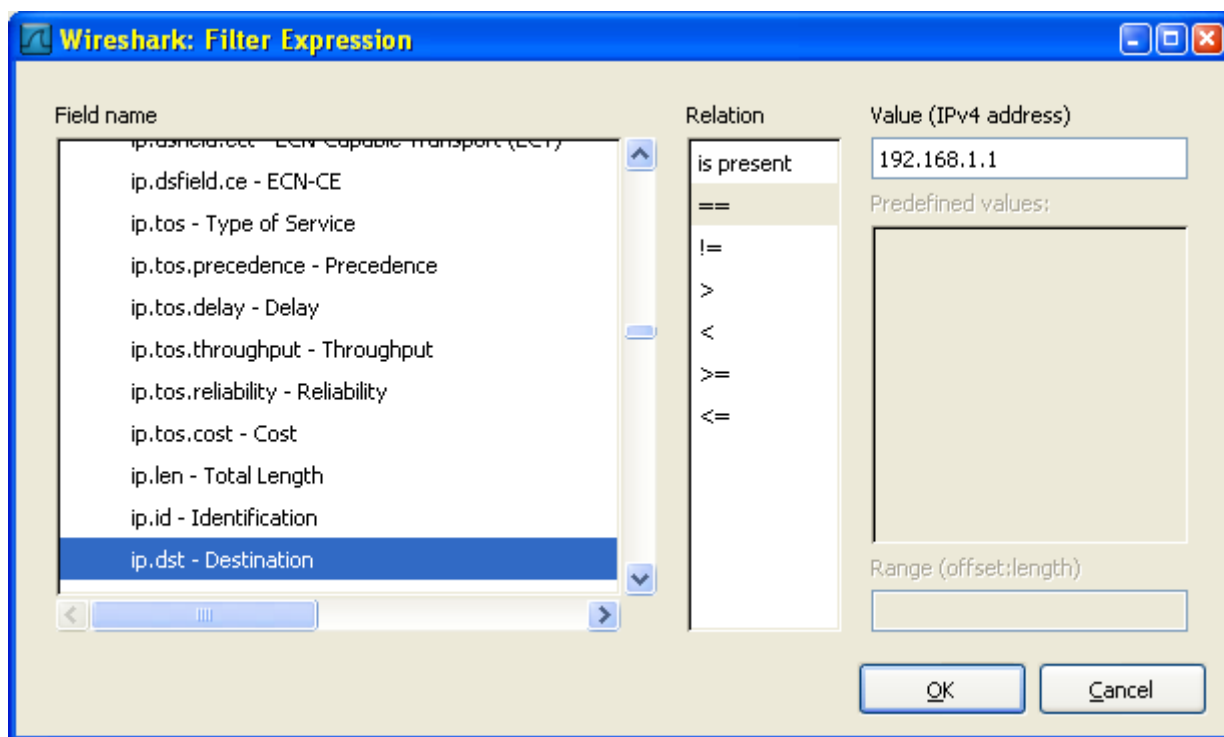


Рис. 5. Построение фильтров отображения

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 рис. 6 приведен пример фильтра для захвата пакетов, адресованных на 80-й порт (http) узла с ip-адресом 10.197.0.11.

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* определяют направление передачи. Возможные направления: **src**; **dst**; **src or dst**; **src and dst**.

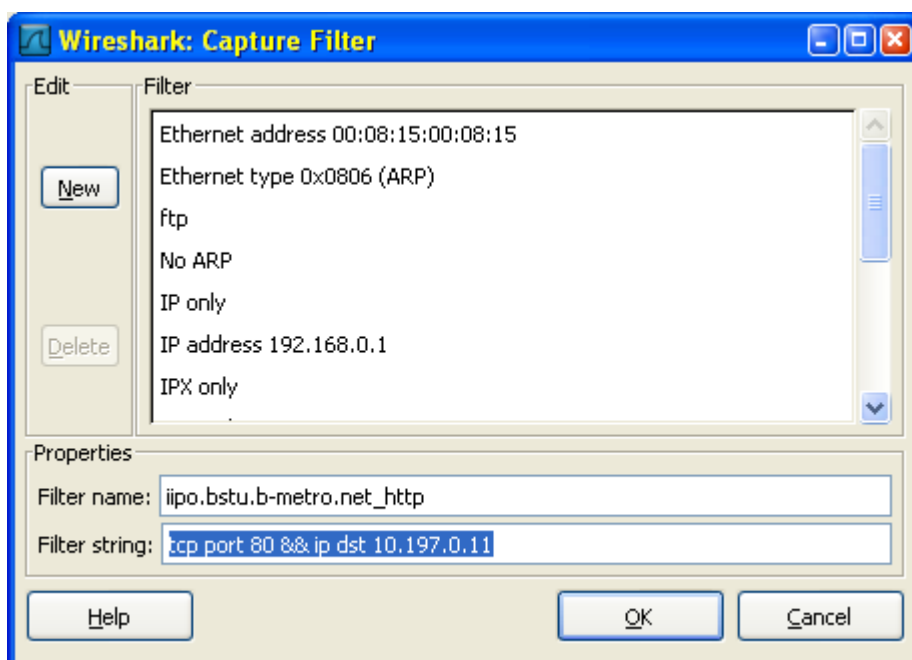


Рис. 6. Пример фильтра захвата

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*** определяют 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 рис. 7 приведена диаграмма взаимодействия при получении узлом win2003 статической web-странички с сервера <http://linux>.

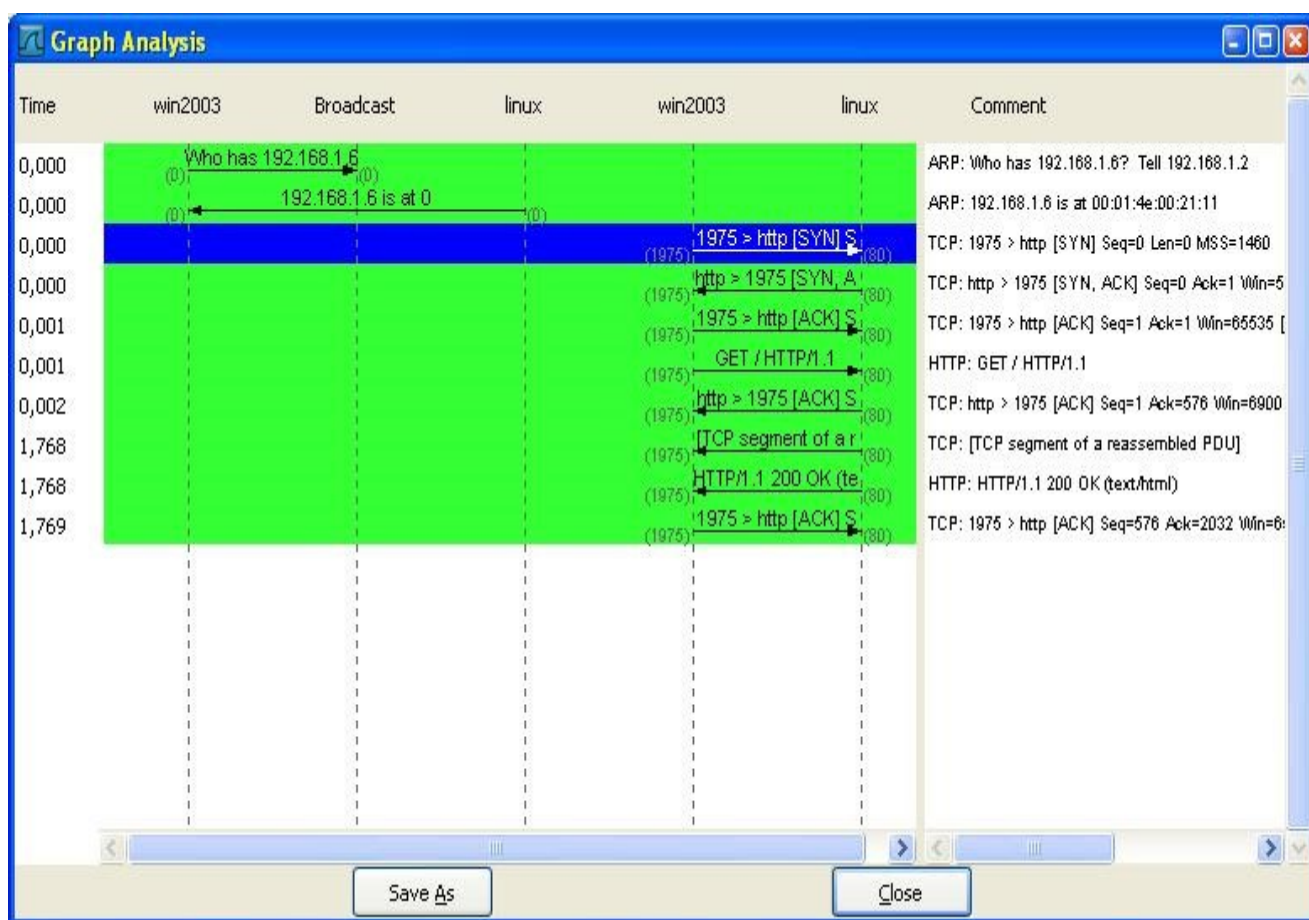


Рис. 7. Диаграмма взаимодействия

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 удастся легко найти и исправить. Сниффер представляет собой также важный инструмент для исследований динамики сети, а равно средство обучения.

ПРИМЕР ИССЛЕДОВАНИЯ ПРОТОКОЛА С ИСПОЛЬЗОВАНИЕМ 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 «-а». 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 рис. 8. В нашем случае мы видим 2 захваченных пакета: ARP-запрос и ARP-ответ.

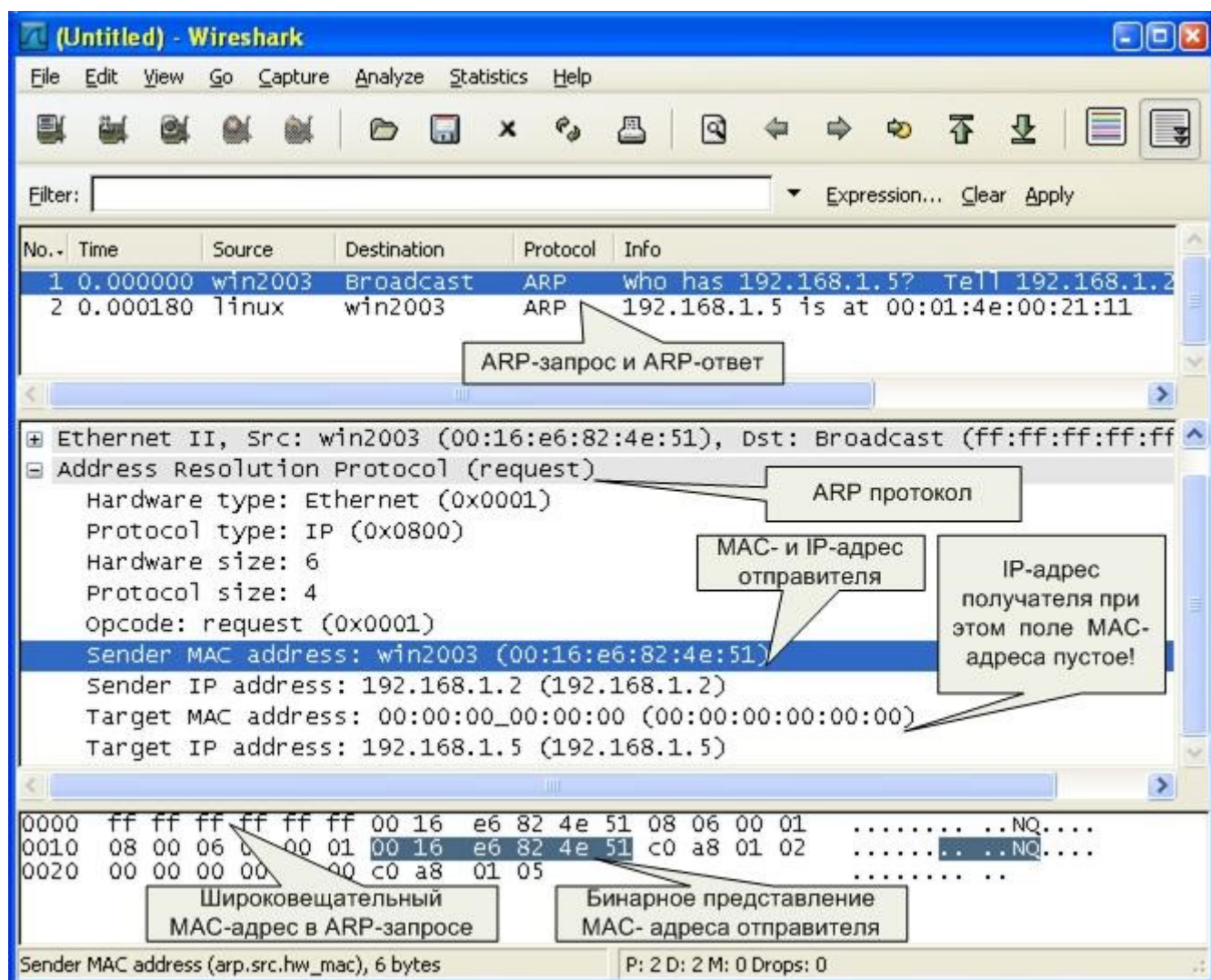


Рис. 8. Анализ ARP-запроса

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».

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 отправляет только та машина, чей IP-адрес соответствует IP-адресу в запросе. Таким образом, второй полученный пакет является ARP-ответом (см. рис. 9). 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».

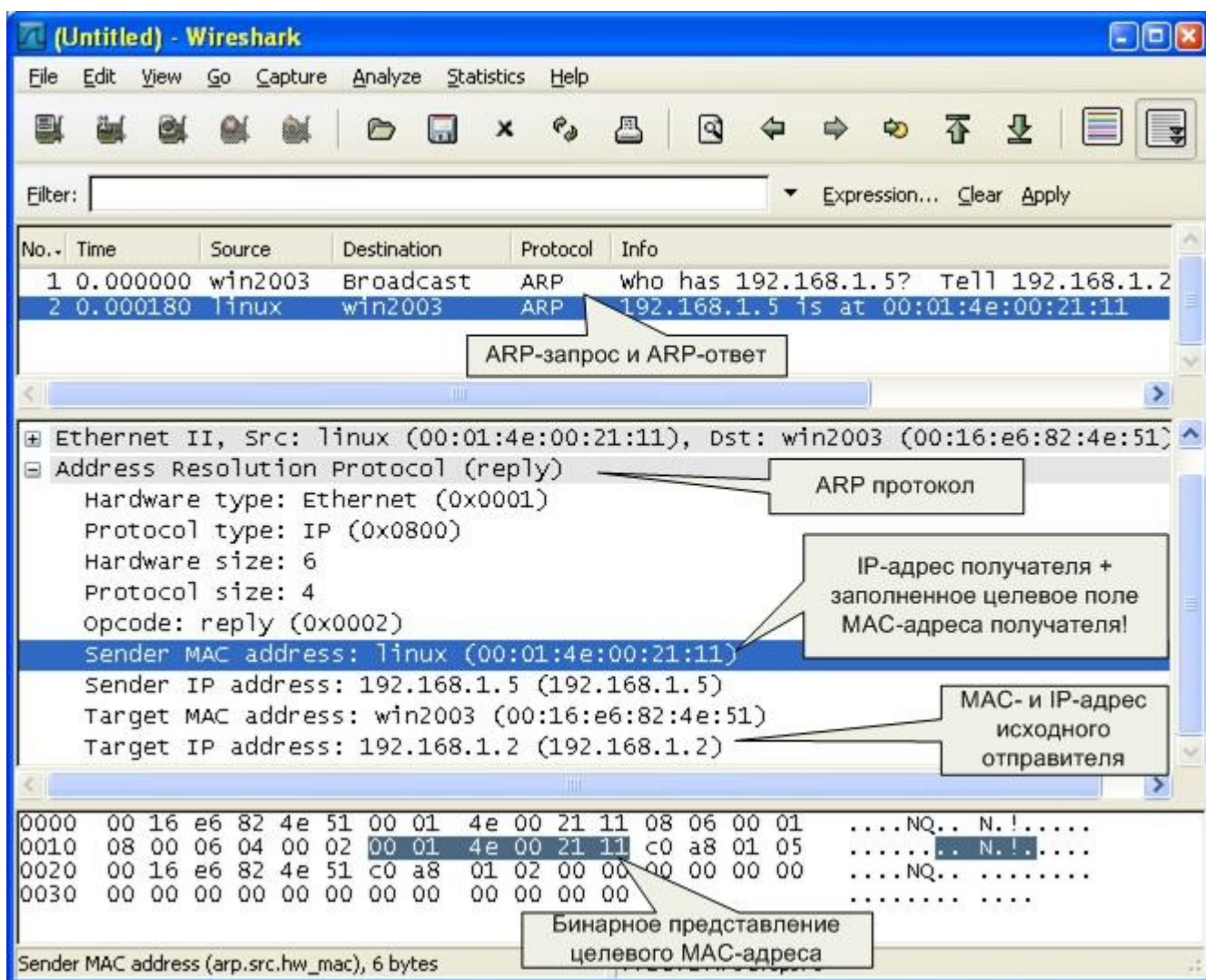


Рис. 9. Анализ ARP-ответа

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.

ЗАДАНИЕ НА ПРАКТИЧЕСКУЮ РАБОТУ

1. Изучить интерфейс программы Wireshark (<\\corp.mgkit.ru\dfs\work\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*.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каковы основные цели мониторинга сетевого трафика?
2. Чем отличается мониторинг трафика от фильтрации?
3. Каково назначение 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?