Изучив протоколы TCP и UDP, мы можем перейти к рассмотрению важных прикладных протоколов стека TCP/IP. Кратко вспомним, как прикладные протоколы используют один из транспортных протоколов стека TCP/IP: все приложения стека TCP/IP используют для передачи данных по сети один из двух протоколов host-to-host уровня модели TCP/IP, протокол TCP или протокол UDP.

Протоколом UDP пользуются приложения, либо отправляющие единичные пакеты данных на которые предполагается получить прикладные ответы, которые и будут выступать в роли подтверждающих квитанций, либо приложения, отправляющие данные широковещательно, так как при любых потребностях приложений TCP не поддерживает широковещательной отправки данных. При этом вопрос о гарантиях доставки данных должно решать само приложение, использующее UDP.

Адресами приложений на узлах являются порты TCP и UDP, вспомним, что использование UDP вместо непосредственной инкапсуляции прикладных данных в IP пакеты обусловлено малым размером поля Protocol в IP заголовке (1 байт), в то время как поля «порты» имеют размер 2 байта и позволяют адресовать достаточное количество прикладных протоколов. Напротив, приложения, передающие по сети большие объемы данных (в особенности это касается тех случаев, когда в основном данные передаются в одном направлении) пользуются надежным протоколом TCP, который устанавливает соединения перед передачей данных и квитирует доставляемые данные. В дальнейшем мы будем изучать различные прикладные протоколы, и он будут пользоваться как UDP так и TCP, в зависимости от рода решаемых прикладным протоколом задач и потребностей приложений.

Прежде чем рассматривать первый прикладной протокол, проведем небольшую условную классификацию используемых в стеке TCP/IP прикладных протоколов. Разделим прикладные протоколы стека TCP/IP на две категории: служебные протоколы, необходимые для упрощения управления сетью, упрощения использования сети и т.д. и прикладные протоколы, призванные непосредственно решать задачи, стоящие перед пользователями. К первой категории мы отнесем следующие протоколы:

- DHCP. Протокол, предназначенный для автоматизации конфигурирования стека TCP/IP на узлах. С его помощью можно назначить узлам IP адрес, маску подсети, адрес шлюза по умолчанию, сконфигурировать многие другие настраиваемые параметры стека TCP/IP, сделав соответствующие настройки на сервере, а не на каждом узле.
- DNS. Это протокол, предназначенный для установления соответствий между символьными структурированными именами узлов и их IP адресами (разумеется, сейчас не время говорить обо всех возможностях протокола, необходимо дать лишь первое общее представление), что позволяет пользователям не запоминать IP адреса узлов, а запоминать понятные имена, а с помощью протокола DNS прозрачно преобразовывать их в IP адреса.
- Telnet. Данный протокол предназначен для удаленного управления узлами или аппаратными устройствами (маршрутизаторами, коммутаторами).

Ко второй категории протоколы, предназначенные для решения тех задач, которые решает в сети конечный пользователь:

- SMTP, POP3 – почтовые протоколы. С помощью этих протоколов пользователи сети могут обмениваться электронными сообщениями. Почтовая служба – одна из важнейших сетевых служб.
- FTP, TFTP – протоколы передачи файлов, так же одна из важнейших сетевых служб.
- HTTP – относительно новый протокол, используется для передачи произвольного рода информации: гипертекстовых документов, изображений, видеоданных, аудиоданных и т.д., крайне популярен сегодня.
- LPR – протокол, с помощью которого пользователи могут отправлять задания на печать и управлять сетевыми принтерами.

Подчеркнем, что это далеко не полный перечень протоколов прикладных протоколов стека TCP/IP, однако перед нами набор ключевых современных прикладных протоколов, которые мы будем изучать в данном курсе.

Первый прикладной протокол, который мы с Вами будем изучать – протокол DHCP. Рассмотрим задачу: пусть имеется большая сеть, необходимо сконфигурировать стек TCP/IP на каждом клиентском компьютере, т.е. по меньшей мере, указать узлу IP адрес, маску подсети и адрес шлюза по умолчанию (а возможно и другие параметры стека, например, сконфигурировать таймеры

TCP). Сконфигурировать один узел, разумеется, не составляет проблем, а если узлов очень много? В таком случае конфигурирование займет очень много времени. К тому же при ручном конфигурировании каждого узла неизбежно появление ошибок, связанных с невнимательностью. Более того, при добавлении нового узла в сеть, при переносе узла из одной подсети в другую администратору придется каждый раз делать дополнительные настройки на узлах – очевидно, это не слишком удобно. Было бы эффективно организовать автоматизацию конфигурирования узлов, использовать в сети некоторую службу, которая конфигурируется однажды (возможно, даже не просто), зато впоследствии объем ручной работы, выполняемой администратором будет сведен к нули или к минимуму. Важно подчеркнуть, что если сеть невелика, то возможно проще сконфигурировать узлы вручную и впоследствии конфигурировать новые узлы или реконфигурировать перемещаемые узлы вручную. Однако если сеть, напротив, велика, имеет смысл однажды потратить время и силы на конфигурирование службы, автоматизирующей настройку узлов, после чего использовать эту службу вместо того, чтобы выполнять большой объем работы. Важно, понять, что подобную службу НЕ нужно внедрять всегда и везде только потому, что Вы умеете ее внедрять, каждое РЕШЕНИЕ должно быть сообразно ПРОБЛЕМЕ и не стоит использовать технологию там, где в этом нет потребности.

Следует рассмотреть еще одно преимущество от использования автоматического назначения IP адресов клиентам. Предположим, что в нашем распоряжении имеется несколько (например 30) «честных» IP адресов, приобретенных у LIR или провайдера. Предположим, что сеть клиента растет и сегодня в ней уже 40 узлов, но особенности работы компании таковы, что одновременно в сети работает не более 20 узлов. В таком случае при статическом назначении адресов возникнет серьезная проблема адресации в сети, в то время как использование динамической адресации позволит без труда решить данную проблему: сервер будет предоставлять адреса тем клиентам, которые в настоящее время работают в сети, те же узлы, которые отключены, не занимают «ценные» IP адреса.

Итак, подытожим преимущества от использования службы, автоматизирующей настройку стека TCP/IP на узлах:

- Упрощение жизни администратора, которому не придется конфигурировать большое количество узлов вручную
- Сведение к минимуму ошибок администратора при конфигурировании узлов
- Создание динамического окружения, в котором нет будет необходимости в конфигурировании узлов при их добавлении или перемещении.
- Возможность организации экономного расходования IP адресов в случае необходимости.

Какие решения по автоматизации конфигурирования стека TCP/IP на узлах нам уже знакомы? Вспомним протокол RARP: с помощью этого протокола можно было назначить станции, пославшей RARP запрос IP адрес с помощью специального RARP сервера. Вспомним формат пакета RARP: с помощью протокола RARP в принципе невозможно назначить интерфейсу ничего, кроме IP адреса. Автоматическое назначение только лишь IP адреса – не достаточно для того, чтобы говорить об автоматическом конфигурировании интерфейса. По крайней мере необходимо назначить еще маску подсети и адрес шлюза по умолчанию. Напомним, что не смотря на то, что протокол RARP не позволяет назначить станции ни маски, ни шлюза вследствие недостатка необходимых полей в заголовке, данная задача может быть решена с помощью ICMP. Действительно, стартующая станция, получающая IP адрес с помощью RARP посылает широковещательный RARP запрос и получает от RARP сервера RARP ответ с указанием назначенного ей IP адреса. После этого станция, получающая настройку от RARP сервера знает IP адрес RARP сервера (из RARP ответа) и может послать RARP серверу ICMP сообщение типа 17 (Get Address Mask). Так как RARP сервер по определению находится в одной сети со станцией клиентом RARP, то и маска подсети у RARP сервера и станции, получающей от него конфигурацию, совпадает. Соответственно, получив ответ от RARP сервера в ICMP сообщении типа 18 (Address Mask Reply) клиент может узнать свою маску подсети. Остается еще необходимость выяснить адрес шлюза, которым можно пользоваться для отправки пакетов в чужие сети. Для решения этой задачи так же можно привлечь протокол ICMP, а точнее сообщения типа 9 и 10 (IRDP). Послав ICMP сообщение типа 9 (Router Solicitation), станция клиент получит от маршрутизатора в ответ ICMP сообщение типа 10 (Router Advertisement), узнав таким образом адрес шлюза, которым можно пользоваться для отправки дейтаграмм в другие сети. Подведем

краткий итог: совокупность технологий RARP, ICMP типов 9, 10, 17, 18 позволяют автоматизировать конфигурирование клиентских систем, при этом узлам можно назначить IP адрес, маску подсети и адрес шлюза по умолчанию. Однако такая технология, совершенно очевидно, не слишком хороша, в ней можно найти целый ряд недостатков, попробуем кратко перечислить эти недостатки:

- Невозможность сконфигурировать другие настройки узлу и/или интерфейсу, нежели приведенные выше, т.е. самые базовые. Желательно иметь возможность назначить адреса серверов имен, возможно адреса каких-то сетевых служб (например, службы точного времени). Данный недостаток является ключевым и уже говорит о неэффективности рассматриваемого подхода.
- Невозможность расширения функциональности протокола RARP
- Необходимость поддержки на маршрутизаторе технологии ICMP 17/18, что не безопасно
- Необходимость поддерживать в каждой подсети RARP сервер, так как RARP пакеты в принципе не могут маршрутизироваться, так как не используют IP заголовка.

Из вышеперечисленных недостатков следует, что протокол RARP не достаточно эффективно справляется с поставленной задачей об автоматическом конфигурировании стека TCP/IP на узлах. Для более эффективного выполнения этой задачи был разработан специальный протокол BOOTP (Bootstrap Process), лишенный многих недостатков связки RARP/ICMP. Этот протокол поддерживал назначение узлу IP адреса, маски подсети, шлюза по умолчанию и других параметров стека, поддерживал расширение возможностей с помощью опций, поддерживал работу через маршрутизаторы после специальной доработки маршрутизаторов, возможность обеспечить начальную загрузку бездисковых станций или станций, на которых не инсталлирована операционная система. Однако протокол BOOTP имел некоторые недостатки, связанные с недостаточной расширяемостью в конечном счете был модифицирован таким образом, что новая версия протокола BOOTP получила название не BOOTPv2, а новое название – DHCP (Dynamic Host Configuration Protocol, протокол динамического конфигурирования узлов). Фактически DHCP – это просто переработанный BOOTP, формат заголовка практически не изменился, однако, тем не менее, новый протокол получил новое название, имеет расширенную функциональность и является сегодня стандартом de-facto для автоматизации конфигурирования IP узлов в локальных сетях. Наша задача: изучить протокол DHCP, принципы его работы, формат заголовка, конфигурирование DHCP сервера.

Для начала зададимся вопросом о том, КТО может назначать узлам IP адреса и прочие параметры? Очевидно, для выполнения этой задачи в сети необходим некоторый сервер, предварительно сконфигурированный администратором, на котором заранее настроены все параметры, которые необходимо передать узлам.

Второй вопрос: с помощью чего клиент сможет получить от сервера все необходимые параметры? Очевидно, с помощью какого-то прикладного протокола, предназначенного для выполнения этой задачи.

Третий вопрос: как узел, не имеющий IP адреса может общаться с сервером? От какого IP адреса узел, пока не имеющий IP адреса сможет отправлять пакеты? Напомним, что в протоколе IP описан так называемый неопределенный адрес 0.0.0.0, который может фигурировать в сетевых пакетах только в качестве адреса отправителя, и как раз предназначен для того, чтобы от этого адреса посылать пакеты в том случае, если станция пока не имеет своего собственного адреса. Итак, клиент сможет посылать IP пакеты от адреса 0.0.0.0

Четвертый вопрос: откуда клиент будет знать IP адрес сервера перед тем, как начинать взаимодействие? Вполне естественно, что клиент НЕ должен знать изначально адреса сервера, иначе этот адрес пришлось настраивать на клиентской машине, что в значительной степени дискредитирует саму идею об автоматизации настроек. Так что у клиента остается только один выход – начинать общение с сервером с посылки широковещательного пакета, причем адрес получателя должен быть широковещательным в заголовке канального уровня, и ограниченно широковещательным в заголовке сетевого уровня.

Пятый вопрос: как клиент сможет получить от сервера ответ, если он еще не имеет IP адреса? На канальном уровне клиент может получить как широковещательный кадр, так и кадр, направленный на адаптер клиента, так как MAC адрес клиента сервер узнает из его запроса. На сетевом уровне клиент, не имея пока адреса, может, тем не менее, получать IP пакеты,

адресованные на ограниченно широковещательный адрес 255.255.255.255 (вспомните самостоятельно почему это так).

Шестой вопрос: откуда DHCP сервер знает, какие адреса и дополнительные конфигурационные параметры предлагать клиенту? Для этого сервер должен быть заранее настроен, т.е. на сервере должно быть указано, из какого диапазона выдавать IP адреса клиентам (в терминах протокола DHCP такой диапазон называют «scope») и какие дополнительные параметры (маску, адрес шлюза по умолчанию и т.д.) необходимо выдавать клиентам. Отмечаем, что все клиенты должны получить различные адреса (в общем, и так понятно), но должны получить одинаковую маску, адрес шлюза и т.д. Поэтому на сервере конфигурируется скоп как диапазон адресов, разрешенных для выдачи клиентам, а все остальные параметры настраиваются как некие свойства данного скопа. Разумеется, детальнее обо всем этом будет сказано ниже, но пока эти сведения нам необходимы для правильного понимания принципов работы DHCP.

Итак, введение и основные идеи работы DHCP мы разобрали, переходим к детальному разбору коммуникаций с помощью протокола DHCP. Начинаем с того, каким транспортным протоколом пользуется DHCP. Вам уже должно быть ясно, что DHCP не может пользоваться TCP, так как мы уже выяснили, что, по крайней мере, часть коммуникаций в рамках DHCP взаимодействия проводится с использованием широковещания, а у одной из станций нет IP адреса, так что об установке TCP соединения речи не идет. Отсюда следует, что DHCP клиент и DHCP сервер должны обмениваться пакетами таким образом, чтобы обеим сторонам было ясно, что их пакеты доставлены партнеру, т.е. подобие гарантий доставки должен обеспечивать сам протокол DHCP.

Служба DHCP, как и большинство служб занимает на сервере хорошо известный порт, а именно UDP 67. Отметим, что DHCP клиент, в отличие от реализаций большинства других прикладных протоколов так же пользуется хорошо известным портом, а именно UDP 68, с чем это связано станет ясно чуть позднее. Итак, рассматриваем, как происходит собственно взаимодействие клиента и сервера. Клиент, стартуя интерфейс, который должен быть сконфигурирован с помощью DHCP, посылает в сеть специальный DHCP пакет, называемый DHCPDISCOVER, суть этого пакета можно отразить следующим образом: может ли кто-то предложить мне IP адрес. Пока мы не рассматриваем сам протокол DHCP (заголовки будут рассмотрены позднее), но скажем, что в заголовке есть некоторое поле, показывающее тип DHCP пакета, позволяющее отличить пакет DHCPDISCOVER от DHCP пакетов других типов, которые мы тоже рассмотрим. Как уже ясно, пакет DHCPDISCOVER посылается в кадре канального уровня с широковещательным адресом получателя, в этом кадре канального уровня находится IP пакет, IP получателя – 255.255.255.255, IP отправителя – 0.0.0.0, в IP пакете – UDP пакет, порт источника – 68, порт получателя – 67, внутри DHCP пакет типа DHCPDISCOVER. Зададимся вопросом: кто получает данный пакет? Очевидно, на канальном и сетевом уровне его получают все узлы локальной сети, а вот содержимое, т.е. DHCP пакет получают только те узлы, на которых открыт 67 UDP порт, т.е. все DHCP сервера. Очевидно, из соображений отказоустойчивости наличие в сети одного DHCP сервера не желательно. А раз в сети может быть несколько серверов, то в ответ на пакет DHCPDISCOVER может поступить, вообще говоря, много ответов с предложениями адреса от каждого активного DHCP сервера в сети. Когда DHCP сервер посылает пакет в ответ на поступивший пакет DHCPDISCOVER, он может послать кадр с этим пакетом на MAC адрес клиента (сервер знает его из пакета DHCPDISCOVER) или широковещательно, обычно используется широковещательная посылка. Поясним, с какой целью делается широковещательная посылка. Хоть в сети может быть несколько DHCP серверов, решающих, по сути, одну и ту же задачу, НИКАКИХ прямых коммуникаций между DHCP серверами в протоколе DHCP не предусмотрено! В рамках протокола DHCP возможны лишь коммуникации клиент – сервер. С другой стороны было бы полезно, если бы DHCP сервера знали, какие предложения клиентам делают ДРУГИЕ DHCP сервера сети. Именно для этого DHCP пакеты от серверов к клиентам посылаются широковещательно на канальном уровне (хотя могли бы быть посланы направлено): чтобы их могли получать и другие DHCP сервера, таким образом, имеет место некоторый косвенный обмен данными между DHCP серверами. Но и этого мало: если бы DHCP клиенты использовали динамические порты, серверам, желающим «подслушивать» взаимодействия других серверов с клиентами пришлось бы открывать на прослушивание все динамические порты, с которых работали клиенты. Именно для того, чтобы не создавать подобных проблем, DHCP клиенты ВСЕГДА работают на хорошо известном порту!

Вернемся к пакетам, посылаемым сервером DHCP в ответ на пакет DHCPDISCOVER. Такой пакет называется DHCPOFFER и содержит сведения, которые сервер может предложить клиенту, а

именно, IP адрес, маску и все остальное. Переводя смысл данного пакета на русский язык, можно было бы сказать: я, сервер, предлагаю тебе данный IP адрес, маску и прочие параметры конфигурации. На уровне IP такой пакет посылается, как уже упоминалось выше по ограниченно широковещательному адресу, так как своего IP адреса у клиента еще нет. Еще раз отметим, что пакетов типа DHCPOFFER может быть послано несколько в ответ на один пакет DHCPDISCOVER. Учитывая, что в сети так же одновременно может быть и несколько клиентов DHCP, а пакет DHCPOFFER может быть послан широковещательно, в самом DHCP пакете должен быть какой то идентификатор, позволяющий DHCP клиентам понять, что ответ предназначается именно им. Когда мы будем рассматривать заголовок DHCP мы увидим, что в этом заголовке есть специальное поле, идентификатор транзакции, клиент посылая DHCPDISCOVER заполняет в пакете это поле, а сервер его цитирует в пакете DHCPOFFER. Итак, клиент получает в ответ на свой пакет DHCPDISCOVER один или несколько пакетов DHCPOFFER. Нужно ли клиенту несколько адресов? Нет, достаточно одного, поэтому клиент в самом простом случае просто выбирает первый пришедший DHCPOFFER. Может ли клиент просто принять данный адрес и пользоваться им? Это крайне не желательно, важно, чтобы Вы поняли, почему это так! Действительно, в таком случае сервер, послав пакет DHCPOFFER не будет уверен, что именно его адрес взял себе клиент, а, учитывая, что в сети может быть несколько DHCP серверов, и если клиент не будет подтверждать, какое именно предложение от какого сервера он принял, КАЖДЫЙ DHCP сервер может полагать, что принято именно его предложение и должен вычеркнуть предложенный в пакете DHCPOFFER адрес из списка доступных для предоставления клиентам. Разумеется, это приведет к тому, что сервера будут быстро истощать свои скопы, так как на каждого DHCP клиента КАЖДЫЙ DHCP сервер будет расходовать один IP адрес, что не эффективно. Было бы хорошо, если бы DHCP клиент уведомлял тот сервер, чье предложение он принял, а так же уведомлял ВСЕ сервера, чье предложение не принято для более эффективного использования адресного пространства. Как сделать первое понятно – достаточно послать серверу, чье предложение принято некоторый DHCP пакет, но как уведомить все остальные DHCP сервера о том, что их предложение не принято? Послать каждому такому серверу другое DHCP сообщение? А если таких серверов много? Используется следующее решение: клиент действительно посылает тому серверу, чье предложение он принял специальный DHCP пакет, называемый DHCPREQUEST, но посылает его широковещательно на канальном уровне и ограниченно широковещательно на сетевом, что приводит к тому, что ВСЕ DHCP сервера получают соответствующий пакет DHCPREQUEST и делают из этого пакета выводы: сервера, чье предложение не принято, считают предложенные данному клиенту IP адреса снова свободными и пригодными для предложения другим клиентам, поведение сервера, чье предложение принято мы рассмотрим подробнее. Отмечаем, что в переводе на русский язык суть пакета DHCPREQUEST можно выразить следующей фразой: из поступивших мне предложений я принимаю твое, можно ли мне пользоваться данным адресом и набором конфигурационных параметров? Поясняем, почему DHCPREQUEST это вопрос, а не утверждение. Может ли клиент, послав DHCPREQUEST и выполнив свой «долг» по экономии IP адресов просто начать пользоваться адресом, предложенным ему тем сервером, чье предложение клиент посчитал нужным принять? Нет, и вот почему: сервер, посылая пакет DHCPOFFER еще не отдает клиенту адрес, понимая, что данное предложение совсем не обязательно будет принято. В принципе, в случае дефицита IP адресов у сервера он может (хотя и это не рекомендуется) предложить данный адрес и другому узлу! Следовательно, из того, что клиент решил выбрать данное предложение, еще не следует, что сервер готов это предложение подтвердить! Поэтому DHCP сервер, получив DHCPREQUEST, т.е. конкретный запрос на использование данного предложения должен подтвердить (или не подтвердить) данное предложение. Если сервер согласен с тем, что данный клиент будет пользоваться этим предложением, то он должен послать в ответ специальный DHCP пакет – DHCPACK, в противном случае – пакет DHCPNACK. Если клиент в ответ на свой запрос DHCPREQUEST получил в ответ DHCPACK, он может начинать пользоваться данным IP адресом, если же получен пакет DHCPNACK, то клиент пользоваться данным адресом не имеет права. Что в таком случае делать клиенту? Может ли он тогда выбрать другое предложение из поступивших ранее пакетов DHCPOFFER? Нет, так как клиент уже отказался от этих предложений, послав DHCPREQUEST на прошлый IP адрес, поэтому клиент должен получив пакет DHCPNACK снова послать пакет DHCPDISCOVER и начать все сначала.

Итог, подводим итог по типовому обмену данными в рамках протокола DHCP: клиент широковещательно шлет DHCPDISCOVER, получает в ответ один или несколько DHCPOFFER, причем эти же пакеты получают и все остальные DHCP сервера сети, клиент выбирает одно из

предложений, шлет широковещательно DHCPREQUEST, те сервера, чье предложение не принято узнают об этом, тот сервер, чье предложение принято посылает в ответ на него DHCPACK или DHCPNACK, в первом случае клиент начинает использовать полученный IP адрес, во втором случае клиент начинает все с начала посылкой нового DHCPDISCOVER.

Отмечаем еще раз, что, так как DHCP использует UDP, то клиенту и серверу необходимо контролировать, доставлены ли пакеты получателям. Рассматриваем кратко, как это делается. Когда клиент посылает пакет DHCPDISCOVER, то, не получив ответа на этот пакет в течение некоторого интервала времени (пакет либо потерян, либо в сети отсутствует хотя бы один DHCP сервер), клиент повторяет данный пакет через заданные интервалы времени заданное число раз, при чем количество повторов и интервалы определяются автором приложения DHCP сервера, а интервал чаще всего не растет как в случае таймера повторной передачи TCP, так как среда – не плохо предсказуемая составная сеть, а локальная сеть. Сервер, послав DHCPOFFER и не получив DHCPREQUEST не повторяет DHCPOFFER, так как формально клиент может послать DHCPREQUEST другому серверу, причем направленно, а не широковещательно. Если же пакет DHCPOFFER потерян, то клиент просто повторит свой DHCPDISCOVER. Клиент, получивший DHCPOFFER и пославший DHCPREQUEST ожидает получения DHCPACK или DHCPNACK, в случае отсутствия одного из этих пакетов клиент пошлет свой DHCPREQUEST несколько раз через некоторые интервалы времени, которые снова таки определяются параметрами DHCP клиента. Как видим вопросами восстановления потерянных пакетов в протоколе DHCP занимается клиент, но не сервер.

Рассматриваем, как ведет себя клиент при получении пакета DHCPACK. Клиент получает IP адрес и другие конфигурационные параметры, но как мы знает, даже при статическом назначении адресов клиент при старте интерфейса нередко проверяет отсутствие дубликатов данного адреса в сети, разумеется, получив данный адрес динамически, клиент так же должен проверить его с помощью ARP запросов. В случае, если проверка прошла удачно (ARP ответа не поступило) клиент начинает пользоваться данным адресом, в случае же, если проверка показала, что такой адрес уже используется в сети клиент должен снова попытаться получить адрес от DHCP сервера, послав пакет DHCPDISCOVER. Но перед тем, как это сделать, клиент должен уведомить DHCP сервер о том, что данный адрес нельзя впредь использовать. Действительно, если этого не сделать, то в лучшем случае сервер предложит данный адрес другому клиенту, породив при этом лишний сетевой трафик и вызвав дополнительную задержку в работу станции-клиента, в худшем же случае сервер может предложить этому же клиенту этот же адрес, что приведет к тому, что клиент вообще не сможет получить корректного адреса никогда.

Для уведомления сервера о невозможность использования некоторого адреса по причине его дублирования в сети существует специальный DHCP пакет – DHCPDECLINE. Клиент шлет его серверу, демонстрируя тем самым, что данный адрес используется в сети, а лишь затем отправляет серверу новый DHCPDISCOVER. Когда сервер получает пакет DHCPDECLINE, он должен исключить данный адрес из своего диапазона адресов, которые допустимы для предоставления клиентам. Важно, чтобы Вы поняли, почему нельзя исключить этот адрес навсегда: возможно, кто-то пользовался статически назначенным адресом по ошибке или недосмотру администратора, если исключить адрес навсегда, то он просто потеряется для дальнейших применений, поэтому сервер помечает адрес как «плохой» на некоторое время, а затем возвращает его в диапазон, пригодный для использования, при этом не исключено, что и следующее предложение данного адреса закончится DHCPDECLINE, но с этим ничего не поделаешь.

Раз мы уже коснулись вопроса экономного использования IP адресов DHCP сервером, необходимо продолжить рассмотрение данной темы. Предположим, станция заканчивает работу в сети, например, выключается. Должна ли она уведомить об этом DHCP сервер? Да, конечно, ведь IP адрес, которым она пользовалась теперь не будет использоваться, а станция возможно, включится не скоро и адрес, которым она пользовалась оказывается «потерянным». Если эта станция включится вновь, она, возможно, получит новый адрес, а старый окажется просто «утраченным», так как сервер будет думать, что адрес используется, а он, увы, ни кем не используется. При этом важно понимать, что один потерянный адрес – не проблема, а когда многие станции будут таким образом приводить к потере адресов сервером, то скоп адресов истощится крайне быстро. Для того, чтобы избежать потери адресов, в DHCP применяется ряд методов, рассмотрим первый из них: станция, прекращающая работу в сети посылает серверу специальный DHCP пакет, называемый DHCPRELEASE, который уведомляет сервер о том, что данный адрес освобождается клиентом. Важно понимать, что сервер не дает ответа на такой пакет, поэтому если пакет DHCPRELEASE

потеряется, то адрес все же не смотря на старания клиента не будет освобожден. Ясно, что такая технология освобождения адресов хоть и полезна, но не достаточна: пакет DHCPRELEASE может потеряться, или станция может выключиться из-за сбоя питания. Последняя ситуация самая неприятная – если в офисе отключилось питание, то множество станций неожиданно, без посылки DHCPRELEASE выключились, и когда они перезагрузятся, то получат, возможно, новые адреса, а множество адресов перейдут в разряд потерянных.

Для того чтобы избежать потери адресов в DHCP применяется следующий подход: сервер выдает клиенту адрес на ограниченное время, сообщаемое клиенту в пакетах DHCPOFFER и DHCPACK. Это время называется временем аренды адреса. Клиент получает не только само время аренды, но и еще два времени, через которые он должен подтвердить, что по-прежнему пользуется адресом. Если клиент не подтверждает использование адреса за оговоренное время, сервер считает адрес свободным, так что даже если многие станции неожиданно, без посылки DHCPRELEASE прекратят свою работу, то через время, равное времени аренды этих адресов сервер сможет вновь данные адреса. Сразу скажем, как клиент может подтвердить использование адреса. Очевидно, при этом не нужно начинать взаимодействие с пакета DHCPDISCOVER, так как у станции уже есть адрес и она просто хочет подтвердить, что пользуется им по-прежнему. В таком случае клиенту достаточно послать пакет DHCPREQUEST, в котором указать, какой именно адрес станция хочет подтвердить, при чем такой пакет отправляется направленно на DHCP сервер и не от адреса 0.0.0.0, а от адреса, который использует станция. Сервер отвечает на такой пакет DHCPACK и тем самым позволяет станции продолжить использование данного адреса. Так же возможен и случай, когда сервер отвечает на такой пакет DHCPNACK, если например, сервер был переконфигурирован и более не может предоставлять станции данный адрес (чуть позднее будет рассмотрена такая ситуация на практике), в таком случае станция не имеет права далее пользоваться этим адресом и начинает новое DHCP взаимодействие с посылки пакета DHCPDISCOVER. Ясно, что такой метод борьбы с потерями адресов исключает потери адресов навсегда, но в случае неожиданного отключения многих станций все равно будет наблюдаться потеря адресов на срок порядка времени аренды адресов. При этом значительное уменьшение срока аренды с целью сократить время, в течение которого адреса «потеряны» не желательно, так как приведет к излишнему DHCP трафику (который часто широковещательный) и нагрузке на DHCP сервера. Для борьбы с этой проблемой используется следующий подход: станции обычно запоминают, какими адресами они пользовались в «прошлый раз», а протокол DHCP, (как это будет показано при изучении заголовка DHCP) позволяет станции в пакете DHCPDISCOVER указать, какой адрес станция бы желала получить. Сервер выдает IP адреса и запоминает, какому MAC адресу выдан данный адрес, следовательно, если станция неожиданно перезагрузилась без посылки DHCPRELEASE сервер будет думать, что станция по прежнему пользуется данным адресом, при рестарте станция попросит данный адрес, сервер убедиться, что хотя адрес и используется, но используется той же самой станцией, которая его просит и выдаст станции тот же самый адрес. Таким образом, можно избежать потери адресов даже на срок времени аренды, кроме того, крайне полезно, когда одна и та же станция получает всегда один и тот же IP адрес, и для администратора, и для пользователей, вообще говоря.

Если станция, отправляя пакет DHCPDISCOVER не указывает желаемый IP адрес, то она принимает первое же предложение, поступившее в пакете DHCPOFFER. А вот если станция при отправке DHCPDISCOVER просит себе некоторый адрес, то в таком случае может быть несколько различных ситуаций, которые имеет смысл рассмотреть.

Для начала рассмотрим такой пример: пусть станция «с прошлого раза» запомнила, что использовала адрес 192.168.0.50. Пока станция была выключена никаких изменений ни в настройках DHCP сервера ни в местоположении станции в подсети не произошло, кроме того полагаем, что у DHCP сервера нет недостатка в количестве адресов в скопе, т.е. сервер не имел причины отдать адрес 192.168.0.50 другой станции. Поэтому когда станция при очередном старте просит DHCP сервер предоставить ей адрес 192.168.0.50, сервер этот адрес предоставляет – все очень просто и явно.

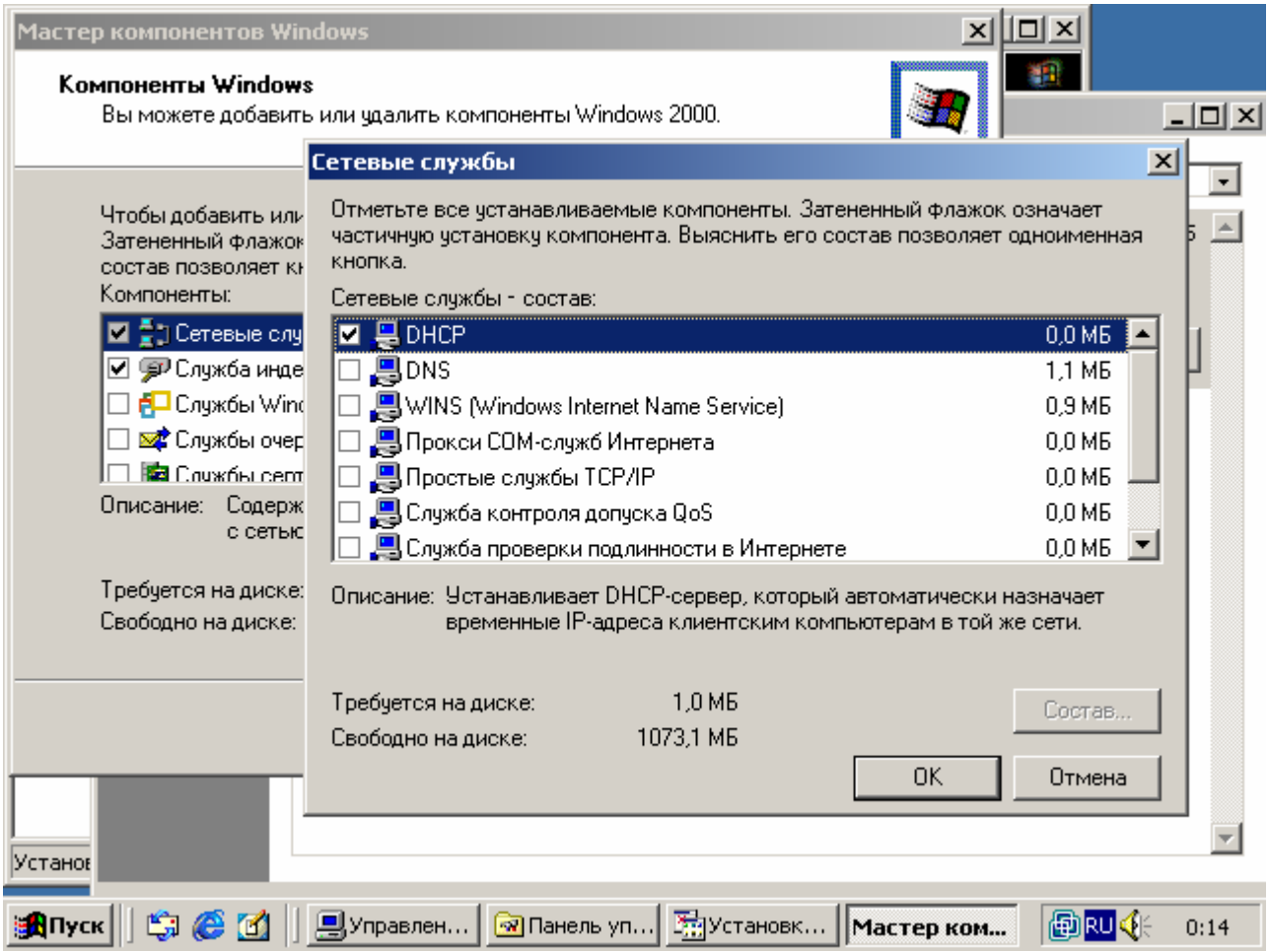
Второй пример: пусть станция снова запомнила, что пользовалась ранее адресом 192.168.0.50, но пока станция была отключена, ее перенесли в другую подсеть, и, разумеется, в другой подсети станция не имеет возможности пользоваться старым адресом. В таком случае станция в пакете DHCPDISCOVER просит адрес 192.168.0.50, сервер предоставить его не может, и предлагает в пакете DHCPOFFER например 10.0.0.25. Станция видит, что предложенный адрес принадлежит ДРУГОЙ IP сети и «понимает» что шансов получить старый адрес нет, поэтому

станция принимает такое предложение, высылает на него DHCPREQUEST и дальше все идет как и говорилось ранее.

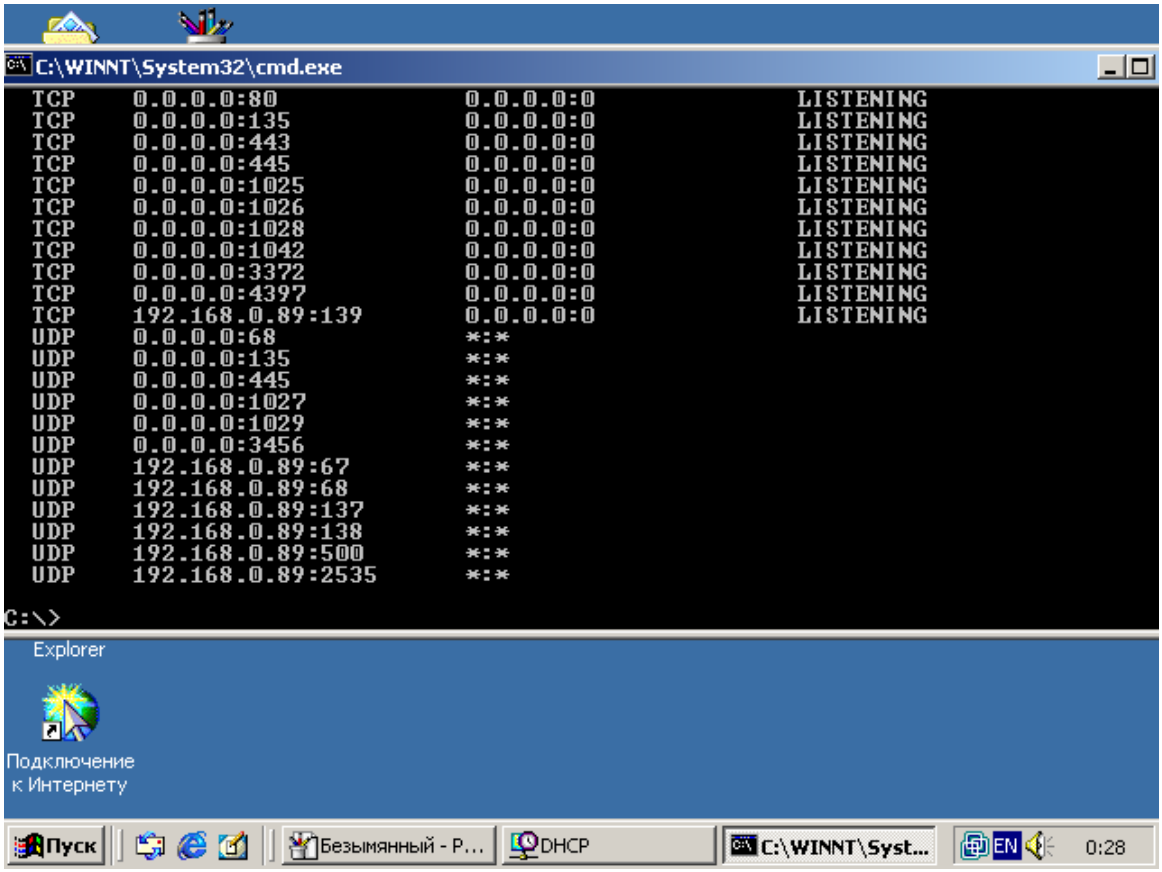
Третий пример: и снова пусть станция запомнила, что пользовалась ранее адресом 192.168.0.50. Предположим, что в сети имеет место дефицит адресов, и данный адрес уже выдан другому узлу, пока данный узел был отключен. Или, к примеру, за время отсутствия станции в сети произошла перенастройка DHCP сервера и адрес 192.168.0.50 больше не является допустимым адресом для назначения узлам сети. В любом случае цель мы преследуем одну – рассмотреть ситуацию, в которой клиент просит некоторый адрес, который ему не может быть назначен, хотя клиент ОСТАЛСЯ в той же сети. Рассмотрим особенности взаимодействия клиента и сервера в этом случае. Клиент посылает пакет DHCPDISCOVER с просьбой предоставить ему адрес 192.168.0.50. Сервер отвечает ему пакетом DHCPOFFER, в котором предлагает другой адрес, например 192.168.0.210. Клиент, видя, что адрес, который ему предлагают, принадлежит ТОЙ ЖЕ сети, надеется, что, ВОЗМОЖНО, поступит предложение пользоваться адресом 192.168.0.50 от ДРУГОГО сервера, поэтому НЕ принимает предложение адреса 192.168.0.210, а повторяет свой пакет DHCPDISCOVER столько раз, сколько повторял бы если бы никакого предложения не поступало (хотя возможно в стеке будут реализованы другие таймеры и другие количества повторов для этого случая). И лишь исчерпав положенное количество попыток послать DHCPDISCOVER, клиент, увидев, что адреса 192.168.0.50 ему так и не предложили, принимает предложение другого адреса, например 192.168.0.210. В таком случае трафик между клиентом и сервером будет, например, таков: DHCPDISCOVER, DHCPOFFER, DHCPDISCOVER, DHCPOFFER, DHCPDISCOVER, DHCPOFFER, DHCPREQUEST, DHCPACK (в случае клиента Windows 2000, который посылает пакет DHCPDISCOVER трижды).

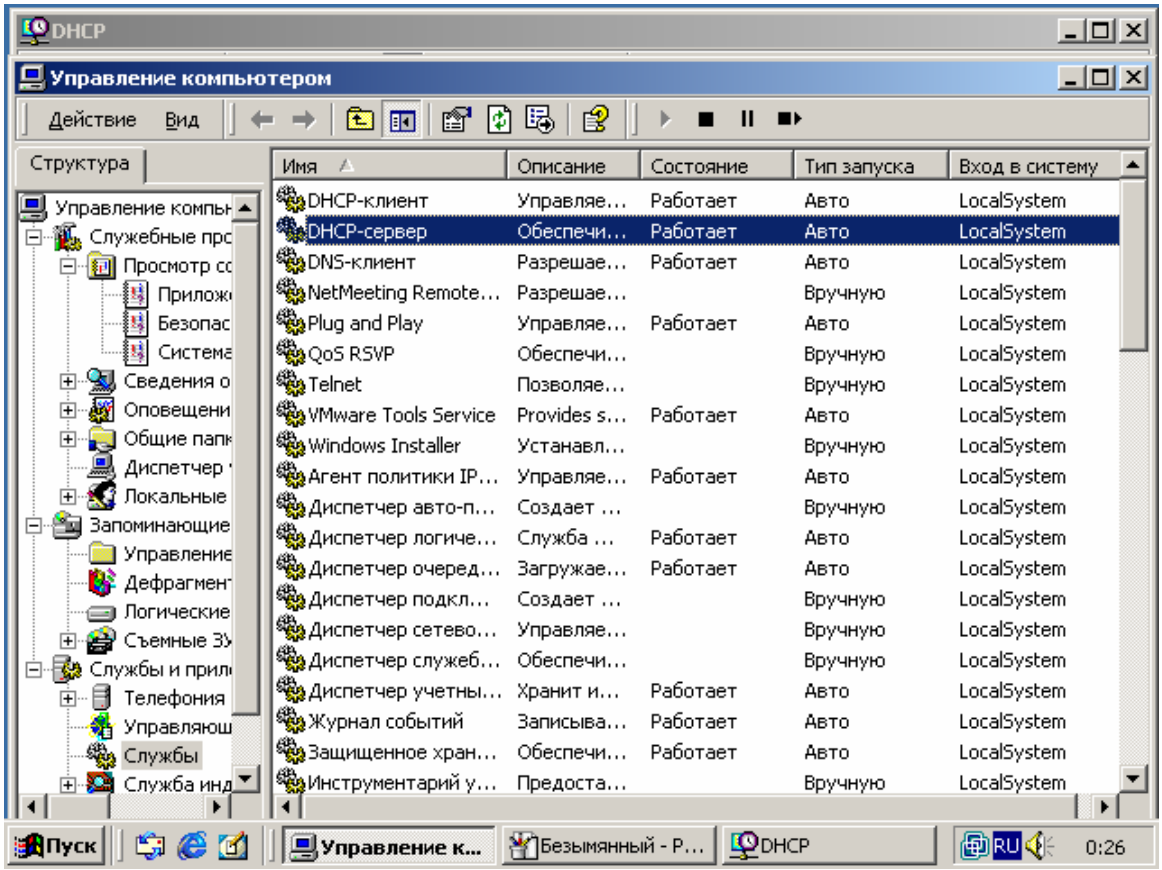
Итак, мы рассмотрели основные принципы, лежащие в основе работы протокола DHCP, теперь попытаемся закрепить полученные знания на практике. Для этого рассмотрим установку и конфигурирование сервера DHCP.

Для начала рассмотрим процесс установки DHCP сервера в операционной системе семейства Windows Server, это делается с помощью апплета Панели Управления «Установка и удаление программ», необходимо указать, что Вы хотите добавить компоненты Windows, затем выбрать сетевые службы:

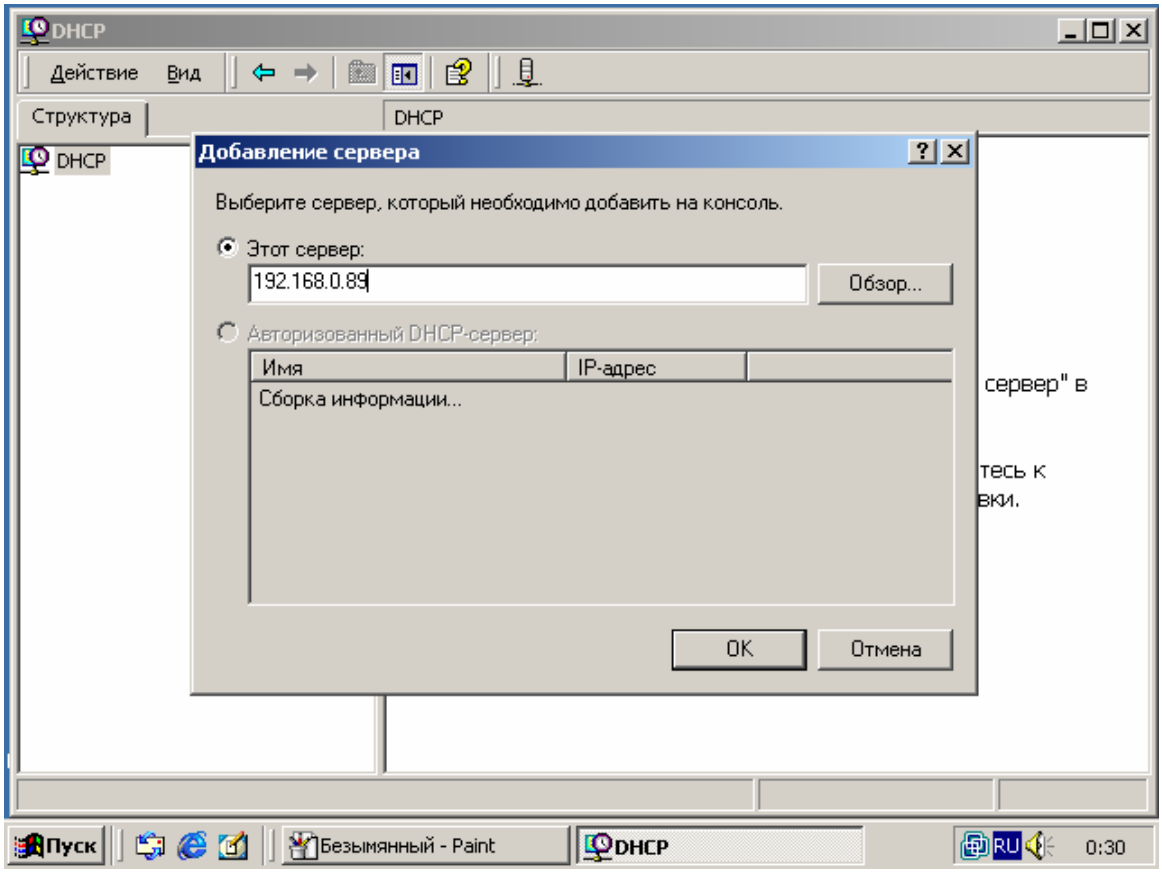


Убедимся с помощью утилиты netstat.exe, что на сервере прослушиваются 67 и 68 UDP порты, убедимся с помощью оснастки управления службами Windows, что на сервере запущена как служба DHCP сервера, так и служба DHCP клиента.

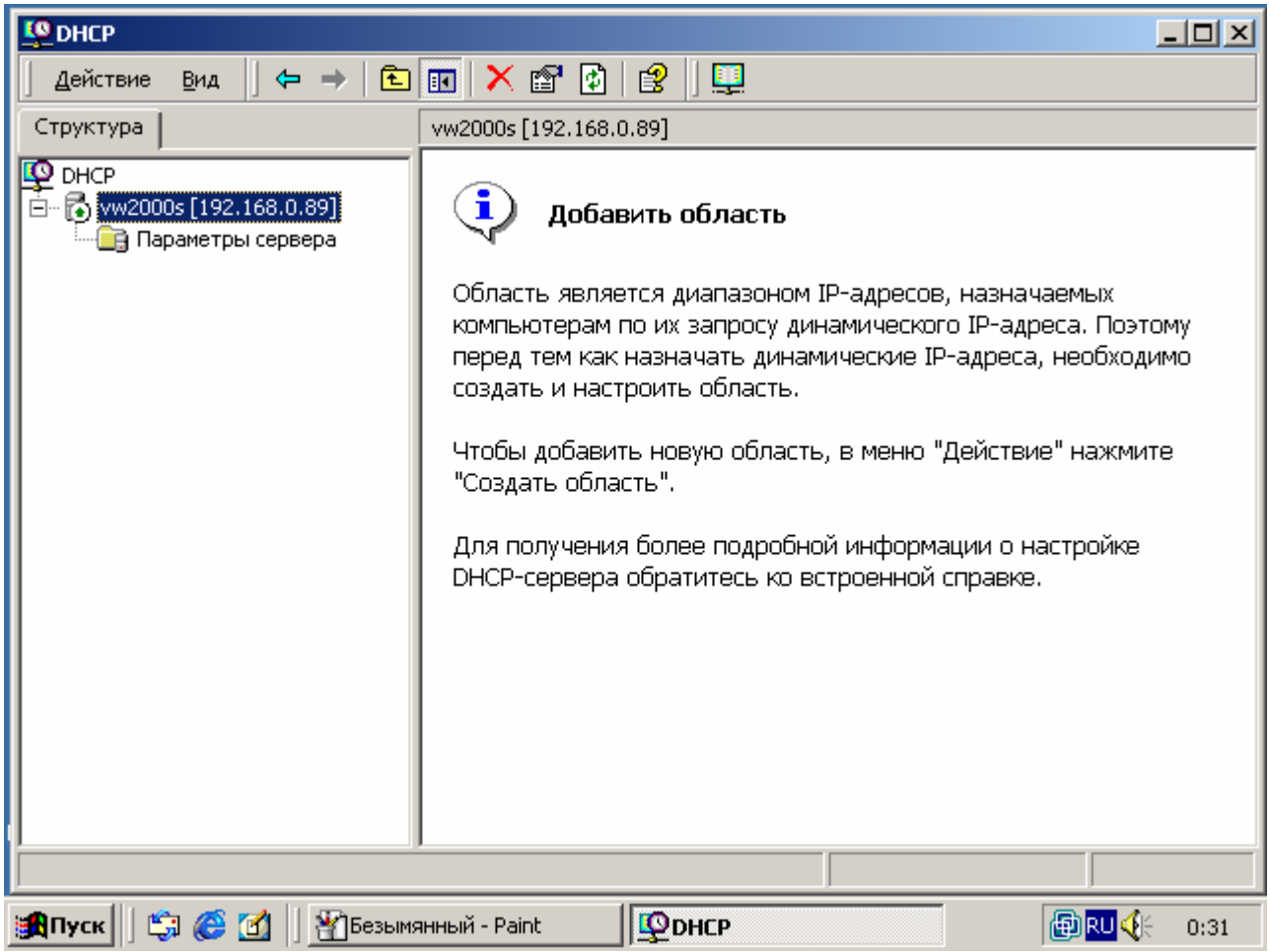




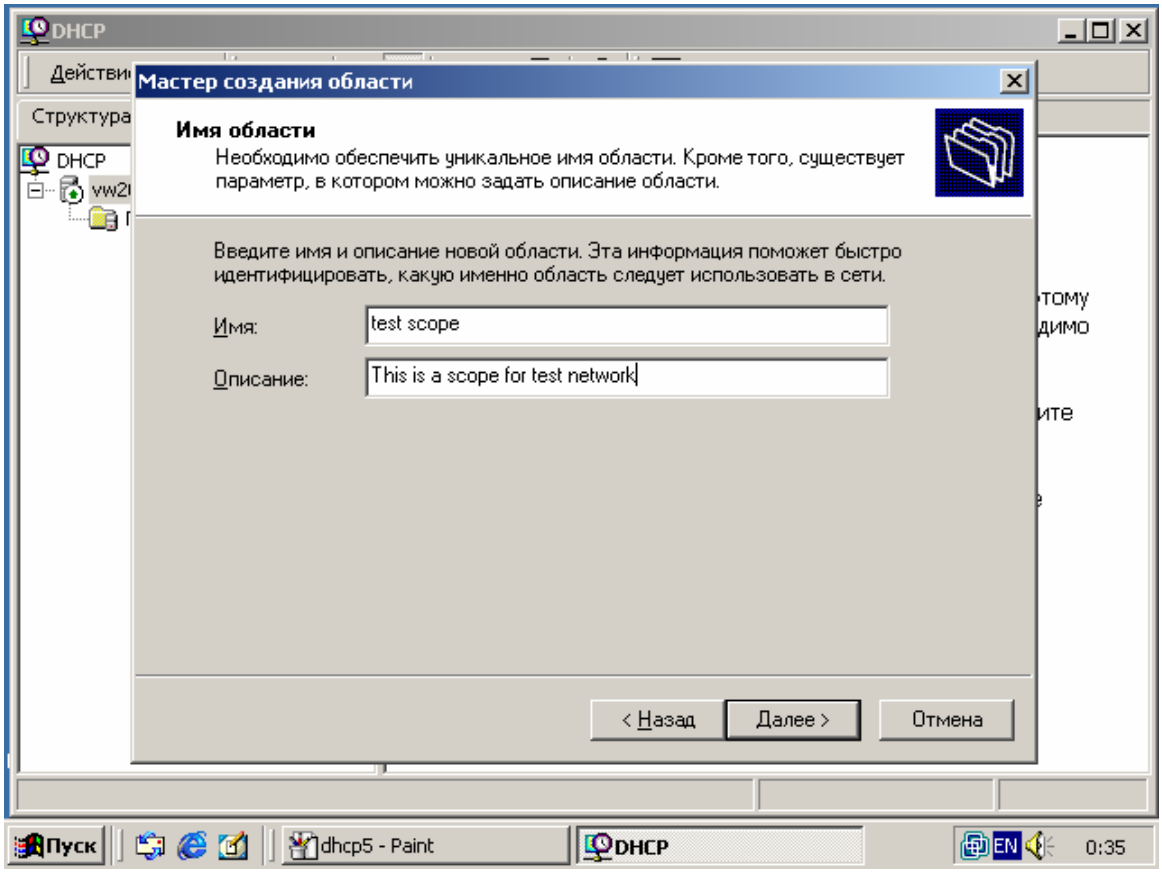
После инсталляции сервера DHCP запустим соответствующую оснастку управления, добавим к списку серверов, управляемых данной оснасткой наш сервер:



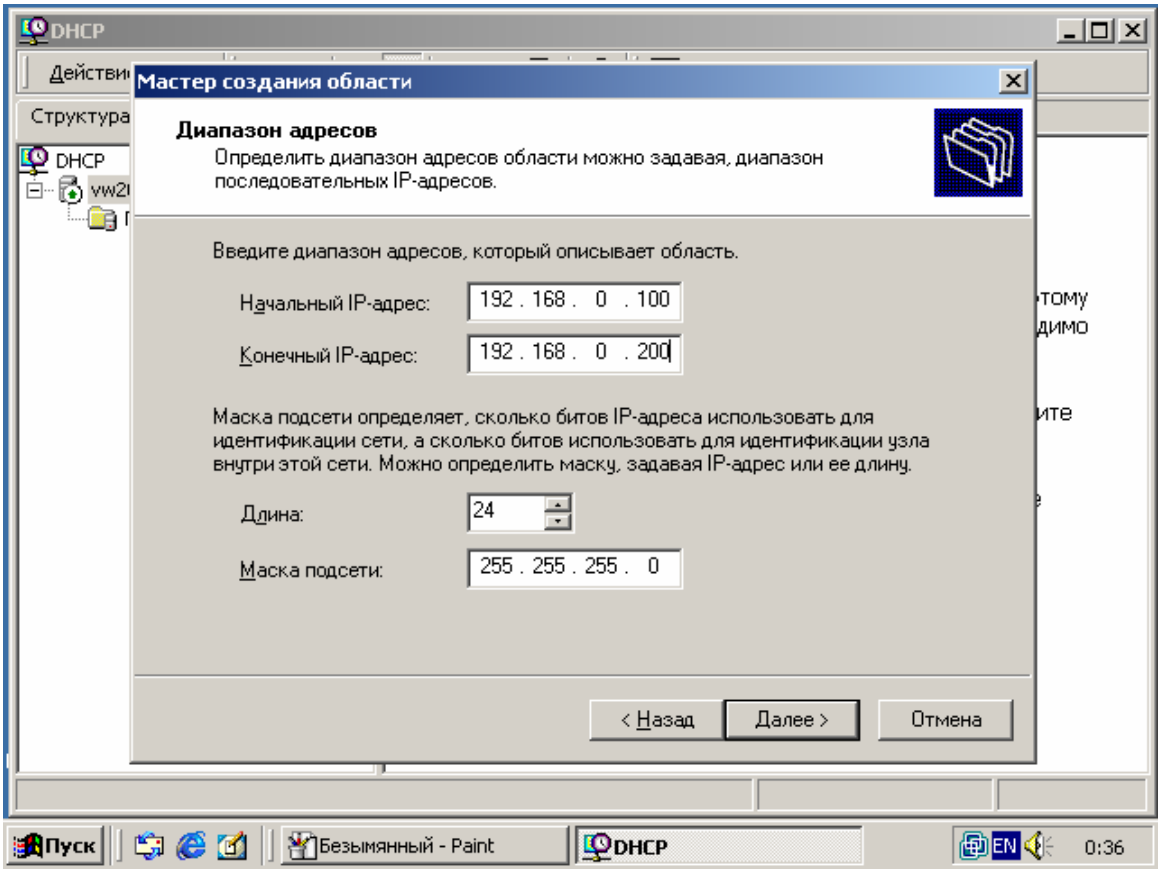
Еще раз подчеркнем, что DHCP сервер не может начать работать сразу после установки – для того, чтобы сервер мог раздавать клиентам IP адреса, необходимо предварительно выполнить хотя бы минимальную настройку сервера – создать по меньшей мере один диапазон адресов (скоп).



Рассмотрим как создается скоп адресов. Каждый создаваемый скоп должен быть снабжен именем, понятным администратору и текстовым описанием, эти параметры служат исключительно для удобства администрирования.

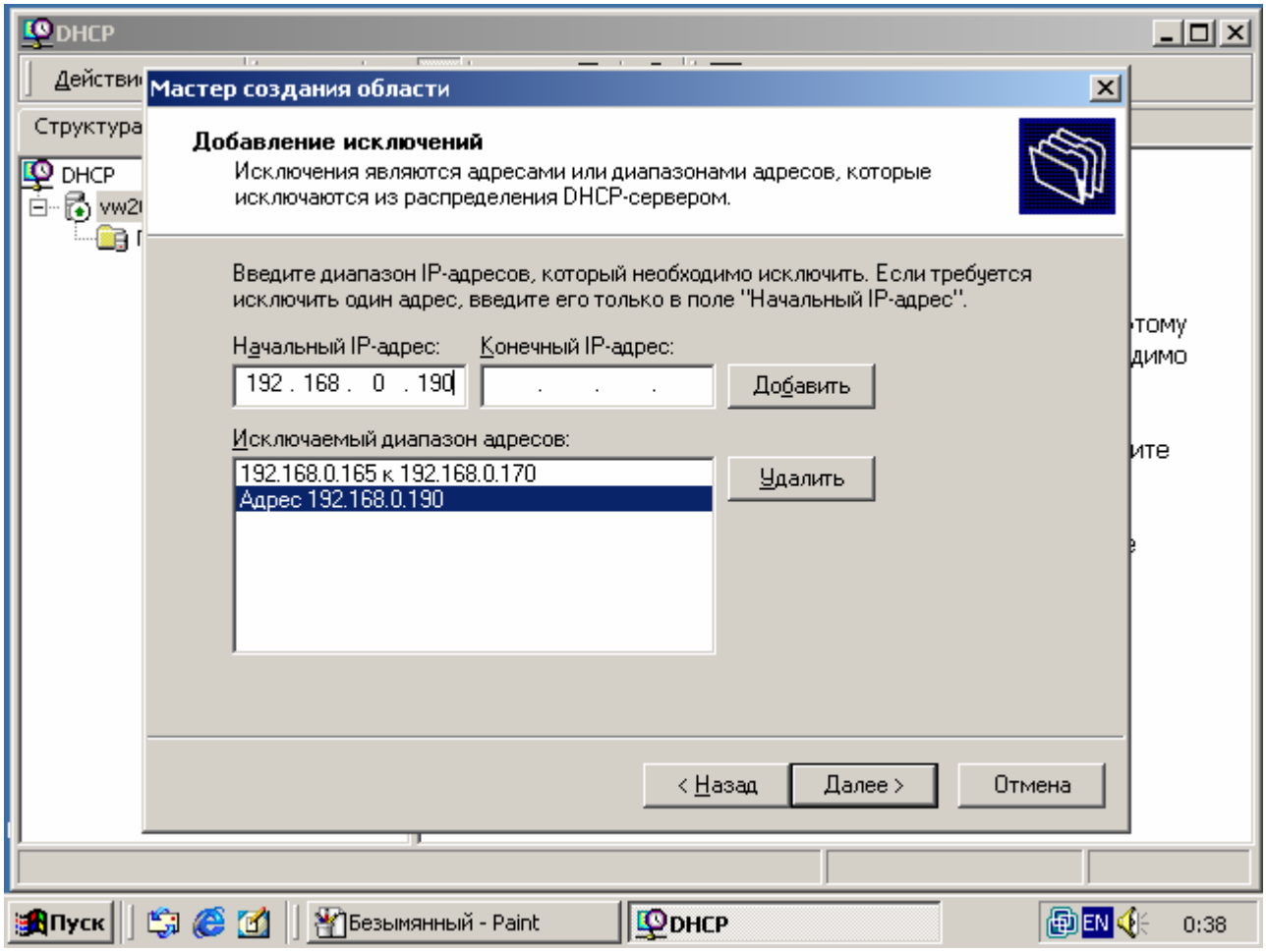


Отметим, что при создании области необходимо указать собственно диапазон адресов и необходимо указать маску подсети.



Формально говоря, стек TCP/IP может и не использовать маски, но данный параметр обязательно должен быть сконфигурирован при настройке скопа в Windows 2000 Server. Протокол DHCP передает в теле DHCP сообщения только адрес, для передачи ВСЕХ прочих конфигурационных параметров, в том числе и маски, используется механизм опций, отчасти подобный применяемым в протоколах IP или TCP. Так что, формально говоря, клиент даже может не поддерживать получение маски подсети. Во время изучения формата DHCP сообщения мы детально разберемся с тем, как передается в теле сообщения адрес и как с помощью опций передаются остальные параметры, в частности рассмотрим ситуацию, когда клиент не поддерживает ту или иную опцию. Пока же отмечаем, что хоть маска в DHCP формально говоря и является просто одной из опций, присваиваемых скопу для предоставления клиентам, важность этой опции сегодня такова, что в Windows 2000 Server эту опцию, в отличие от всех множества других опций ТРЕБУЕТСЯ присвоить скопу.

Следующим шагом создания скопа является указание адресов, исключаемых из скопа.

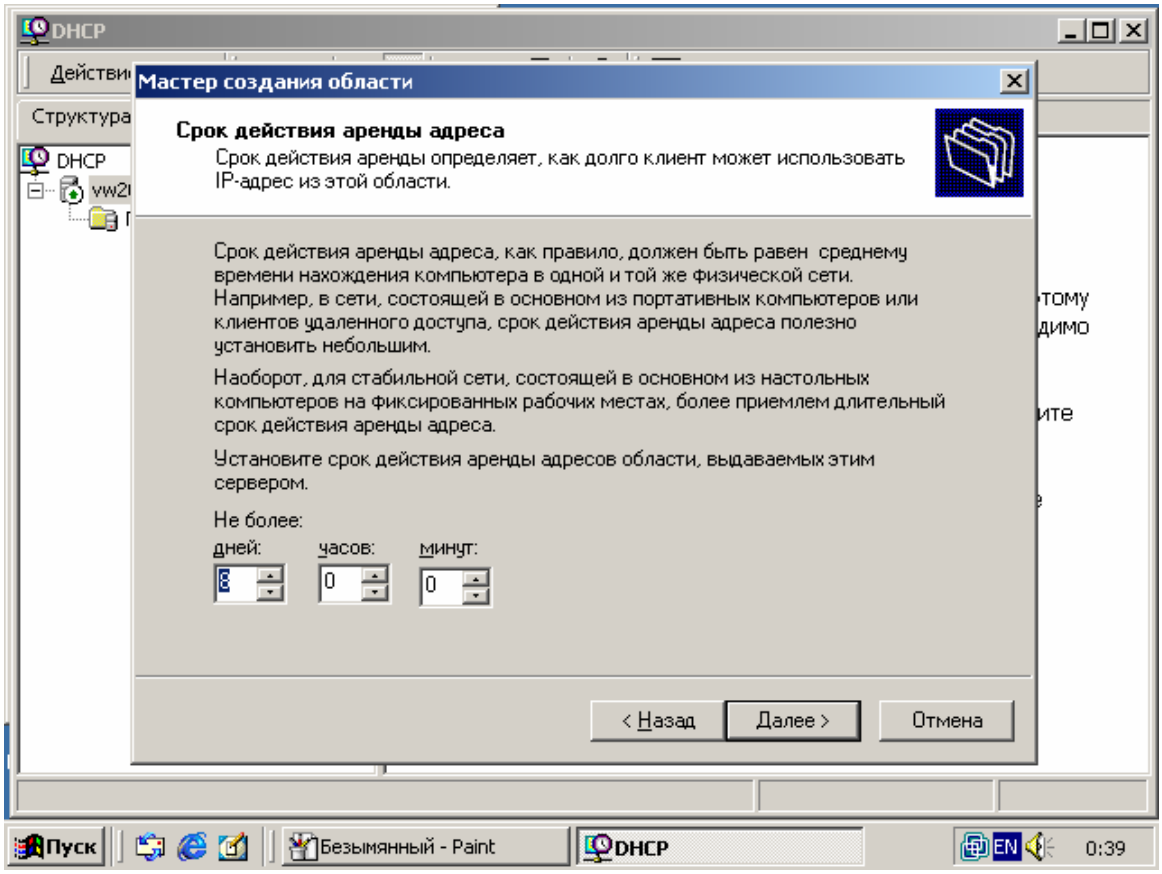


Исключать можно как идущие подряд диапазоны адресов, так и отдельные адреса. Зачем это может понадобиться? Представим себе, что в нашей сети мы применяем адреса из диапазоны 192.168.0.0/24, и первые 50 адресов зарезервированы за маршрутизаторами и серверами нашей сети, тогда, разумеется, в качестве скопа, обслуживающего клиентов мы выберем адреса из диапазона 192.168.0.51 – 192.168.0.254. Однако представим себе, что адрес 192.168.0.101 исторически оказался закреплен за неким сетевым принтером и менять адрес этому принтеру мы не можем или не предпочитаем. Выходит, что прямо в середине нашего скопа есть адрес, выдавать который клиенту никак нельзя, это завершится пакетом DHCPDECLINE и повторной транзакцией DHCP. Как поступить в таком случае? Ясно, что для решения проблемы можно создать два скопа:

192.168.0.51 – 192.168.0.100 и

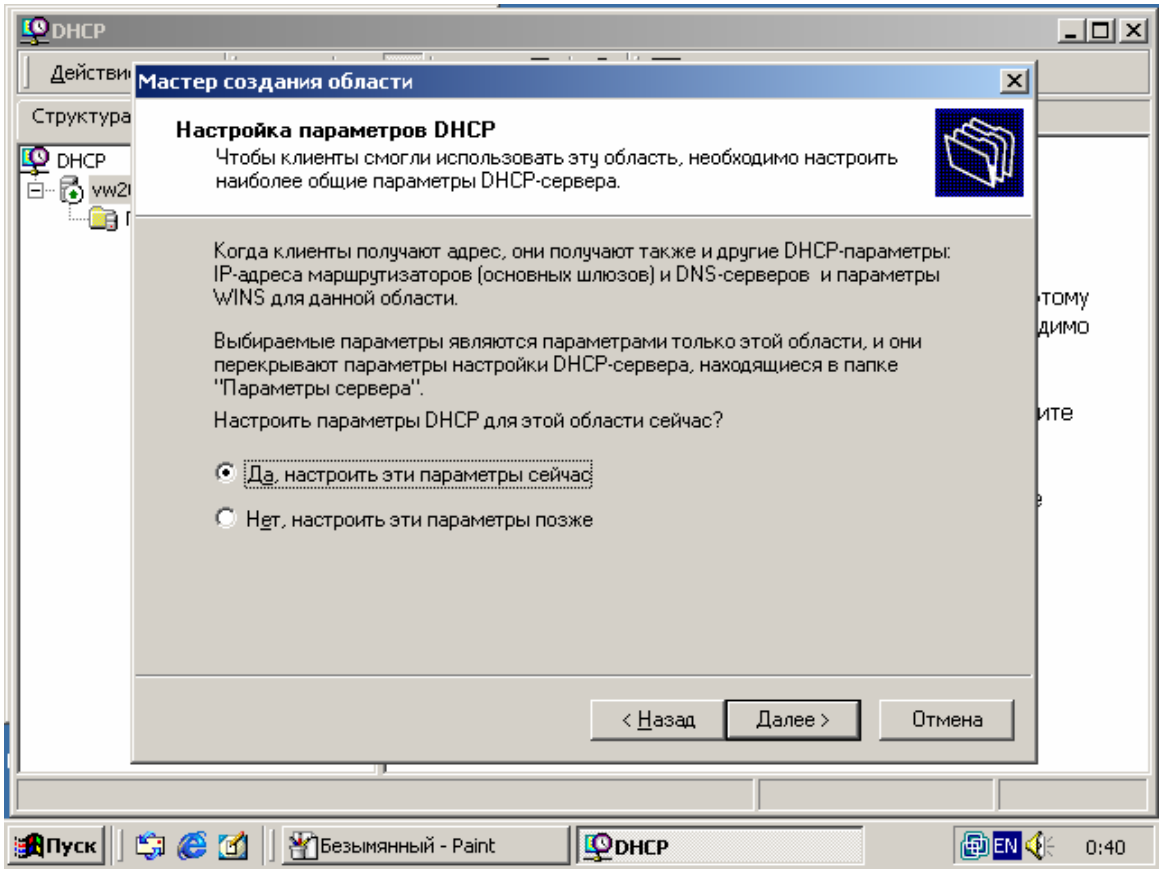
192.168.0.102 – 192.168.0.254, однако это будет не слишком удобно хотя бы потому, что администратору придется управлять двумя разными скопами. Вместо этого можно создать один единственный скоп и исключить нежелательный адрес 192.168.0.101 из этого скопа.

Для создаваемого скопа также необходимо указать время аренды адреса, которое будет передано клиенту с целями, описанными выше, при чем время аренды ТОЖЕ, как маска и все остальные конфигурационные параметры передаются в DHCP с помощью опций, эта опция тоже столь важна, что должна, как и маска обязательно сопоставляться скопу.



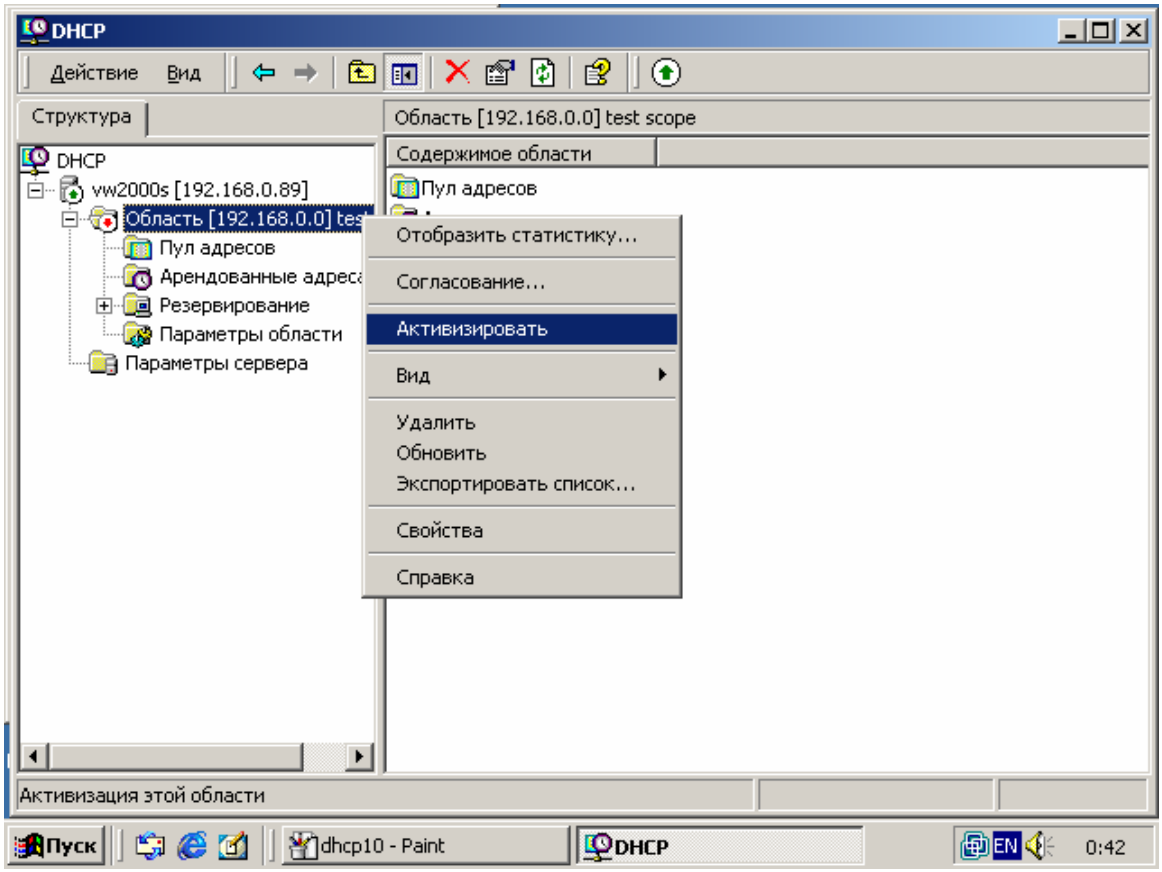
Это время, как мы уже говорили не должно быть слишком малым чтобы не провоцировать интенсивный DHCP трафик в сети, так же оно не должно быть слишком большим для того, чтобы не возникало ситуаций, когда неиспользуемые клиентами адреса не могут быть выданы новым клиентам.

После конфигурирования всех перечисленных выше настроек необходимо выбрать, будем ли мы сейчас настраивать прочие конфигурационные параметры, передаваемые протоколом DHCP с помощью опций (адрес шлюза по умолчанию, статические маршруты, адреса серверов имен и т.д.), это можно сделать сразу и потом, пока не настраиваем никаких дополнительных опций (конфигурационных параметров).

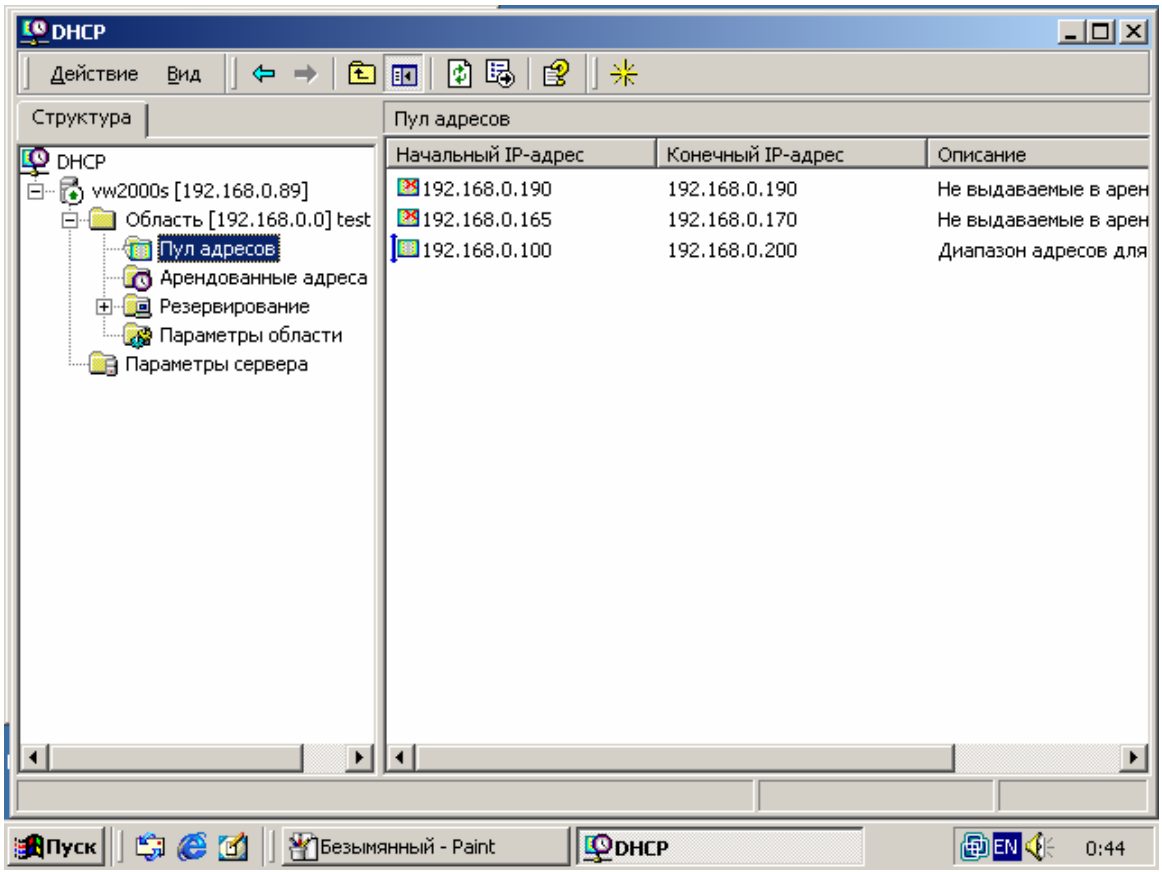


Подводим краткий итог: протокол DHCP содержит стационарную часть заголовка и опции, в стационарной части заголовка клиенту передается IP адрес (и еще кое-что, об этом позже, при детальном изучении заголовка), с помощью опций передаются ВСЕ остальные параметры, призванные настроить стек TCP/IP на узле. При этом в момент создания скопа в Windows 2000 Server этому скопу присваивается две обязательные опции – маска подсети и время аренды адреса как необходимые ВСЕГДА. Прочие опции можно сконфигурировать или этого не сделать в зависимости от потребностей, об этом мы детальнее поговорим позже.

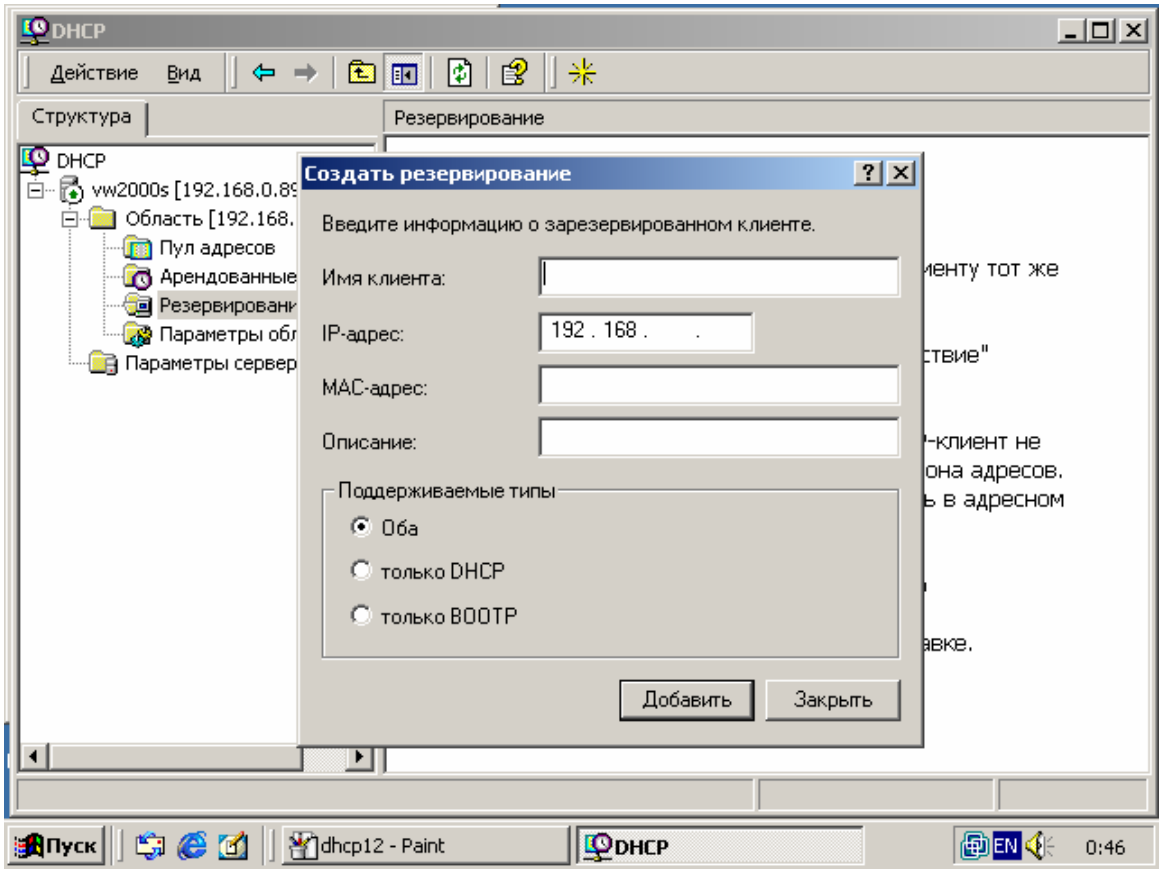
После того, как область создана, она еще не может использоваться клиентами, так как не является активной, для использования ее необходимо активировать. Возможность деактивировать скоп не удаляя его и всех настроек с ним связанных весьма удобна при переконфигурировании или поиске неисправностей:



Так же после того, как скоп создан, мы так же можем конфигурировать те или иные параметры этого скопа, например, диапазоны адресов и исключений в части «Пул адресов»:

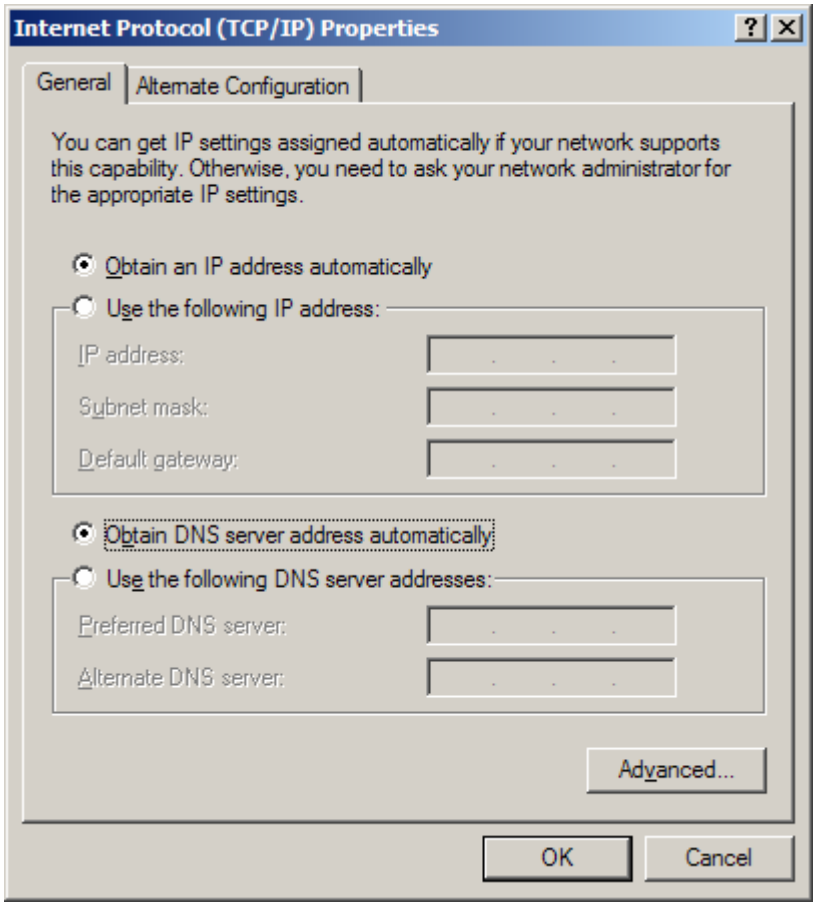


В «арендованных адресах» мы будем видеть узлам с какими MAC адресами присвоены какие IP адреса, когда наш сервер реально будет обслуживать клиентов. Так же DHCP сервера дают администратору возможность создать «резервирования – статические соответствия между MAC адресами узлов клиентов и их IP адресами, это весьма полезно если необходимо чтобы некоторые узлы, оставаясь в динамическом окружении всегда получали один и тот же адрес от DHCP сервера.



Наконец «параметры области» - возможность указать множество конфигурационных параметров, которые будут переданы клиентам данного скопа, получающим адреса. Отмечаем, что помимо возможности настроить параметры для области их так же можно настроить отдельно для резервирования и для всего сервера в целом, таким образом DHCP сервер Microsoft позволяет гибкое конфигурирование узлов, пользующихся услугами данного DHCP сервера.

Теперь, когда мы рассмотрели базовые принципы конфигурирования сервера DHCP, рассмотрим, каким образом можно управлять DHCP клиентом в Windows. По умолчанию все интерфейсы локальной сети в операционной системе Windows настроены на использование DHCP.



Заметим, что если интерфейс настроен на использование DHCP, то с помощью окна свойств TCP/IP этого интерфейса нельзя узнать, получил ли клиент какой-то IP адрес и если получил, то какой. Для этого необходимо пользоваться утилитой ipconfig.exe. Так же отмечаем, что у утилиты ipconfig.exe есть четыре дополнительных ключа для управления DHCP клиентом, два из них мы рассмотрим сейчас.

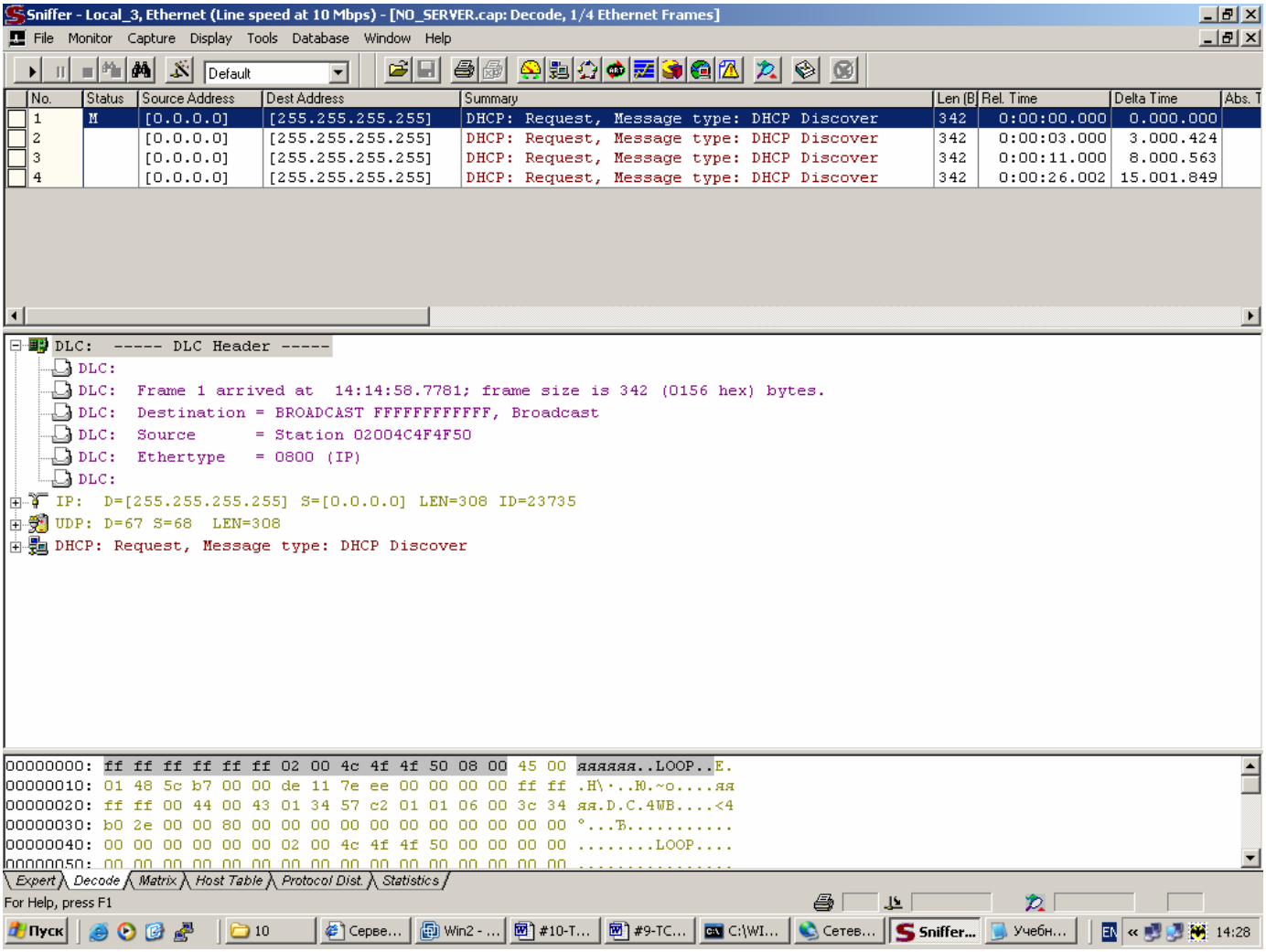
Ключ /renew позволяет сделать следующее:

- Если у клиента нет IP адреса, то команда ipconfig.exe /renew позволяет начать процедуру получения IP адреса, т.е. после этой команды DHCP клиент пошлет пакет DHCPDISCOVER и попытается получить IP адрес от DHCP сервера.
- Если у клиента уже есть адрес, полученный с помощью DHCP, то в таком случае клиент проведет процедуру обновления данного адреса, т.е. сделает то, что должен делать при наступлении таймаута аренды, о котором мы кратко говорили на прошлом занятии. Клиент пошлет пакет DHCPREQEST в надежде получить от сервера пакет DHCPACK.
- Если ни один интерфейс клиента не использует DHCP для конфигурирования, то мы получим сообщение об ошибке.

С помощью второго ключа - /release DHCP клиент освобождает используемый адрес, т.е. пошлет серверу пакет DHCPRELEASE и прекратит использование IP адреса, если же ни один интерфейс клиента не использует DHCP для конфигурирования, то мы снова получим сообщение об ошибке.

Попробуем НЕ устанавливая в своей сети DHCP сервера, активировать DHCP клиента на одной из станций и проанализируем соответствующий трафик в анализаторе протоколов (файл no_server.cap). Так как DHCP заголовок еще не изучен, а нам нужно сейчас продемонстрировать только принцип обмена пакетами, мы можем воспользоваться тем, что Sniffer Pro с строке с протоколом DHCP указывает тип сообщения, так что даже без раскрытия заголовка DHCP в анализаторе мы может рассмотреть особенности обмена пакетами. Клиент под управлением Windows XP (впредь везде в моих примерах в роли клиента DHCP будет выступать Windows XP SP1) посылает 4 сообщения DHCPDISCOVER. При этом клиент наращивает интервал ожидания прихода пакетов DHCPOFFER: после первого DHCPDISCOVER клиент ждет 3 секунды (видно на рисунке), после второго DHCPDISCOVER клиент ждет уже 8 секунд (не путаем Real time и Delta time), после третьего – 15 секунд. Сколько клиент ждет после четвертого DHCPDISCOVER из анализатора узнать нельзя, так как в момент, когда клиент прекращает ждать ответов на

DHCPDISCOVER с номерами 1-3 он шлет новый пакет DHCPDISCOVER, время отправки которого и является для нас способом отмерить время, а вот сколько клиент ждет после четвертого DHCPDISCOVER так не узнать – в этот момент клиент не шлет в провод никаких пакетов, но это можно замерить секундомером и выяснить, что клиент ждет 30 секунд, итого в сумме наш DHCP клиент пытался получить IP адрес от несуществующего или не отвечающего сервера в течение 56 секунд. Обращаем внимание, что пакет DHCPDISCOVER отправляется на широковещательный MAC адрес получателя, на ограниченно широковещательный IP адрес получателя от IP адреса 0.0.0.0, с UDP порта 68 на UDP порт 67. Так же видим, что в ПРОЦЕССЕ получения IP адреса утилита ipconfig.exe сообщала нам, что IP адрес интерфейса неопределенный:




```
C:\>ipconfig
Настройка протокола IP для Windows
Подключение по локальной сети 2 - Ethernet адаптер:

        DNS-суффикс этого подключения . . :
        IP-адрес  . . . . . . . . . . . . : 0.0.0.0
        Маска подсети . . . . . . . . . . : 0.0.0.0
        Основной шлюз . . . . . . . . . . :
```

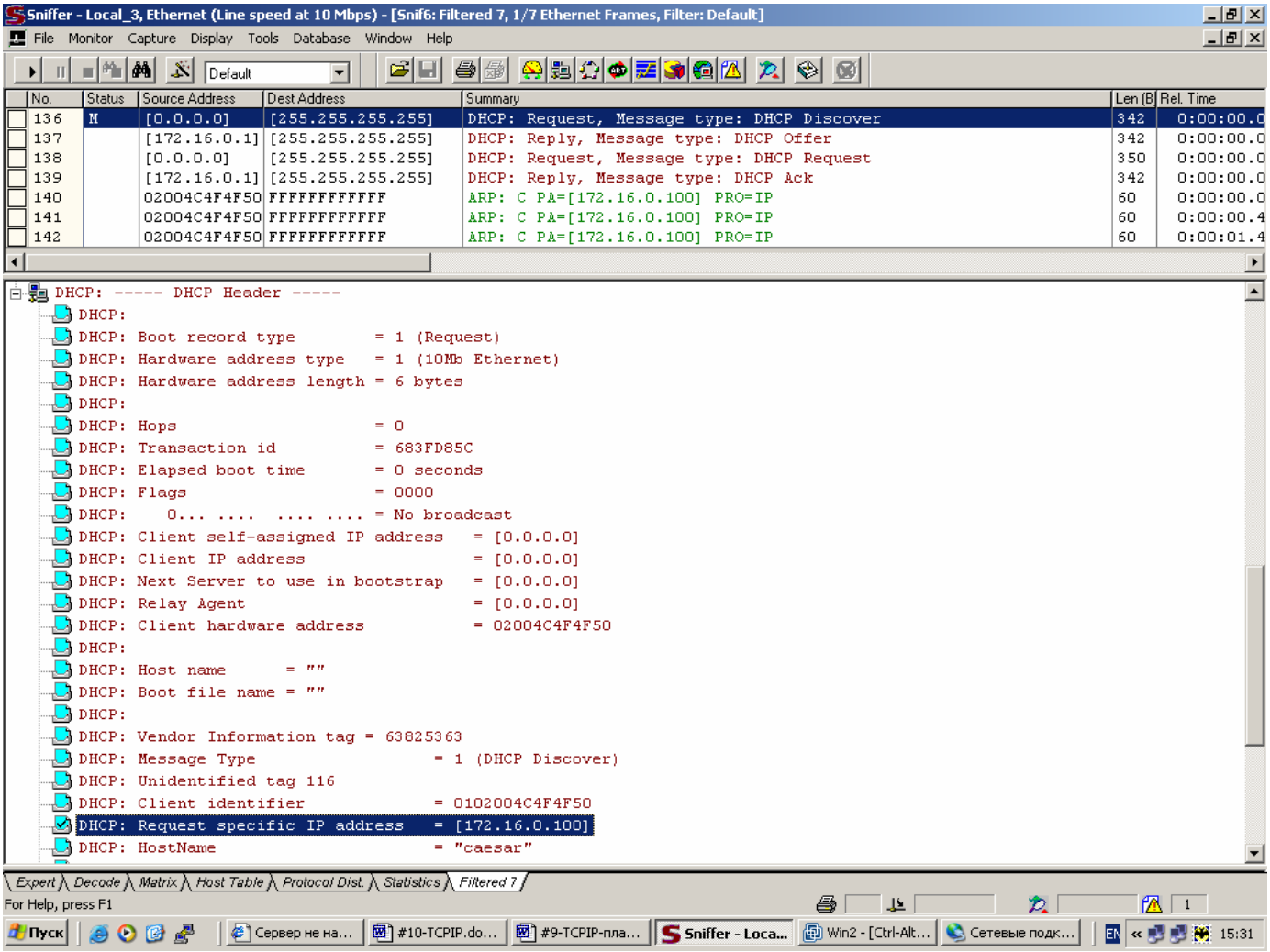
По окончании (неудачном) операции получения интерфейсом IP адреса, утилита ipconfig.exe выдает другой результат:

```
C:\>ipconfig
Настройка протокола IP для Windows
Подключение по локальной сети 2 - Ethernet адаптер:

        DNS-суффикс этого подключения . . :
        IP-адрес  . . . . . . . . . . . . : 169.254.25.129
        Маска подсети . . . . . . . . . . : 255.255.0.0
        Основной шлюз . . . . . . . . . . :
```

Вспоминаем, что это значит – в конце прошлого курса мы говорили об автоматическом конфигурировании интерфейсов с помощью специально выделенного диапазона адресов 169.254.0.0/16.

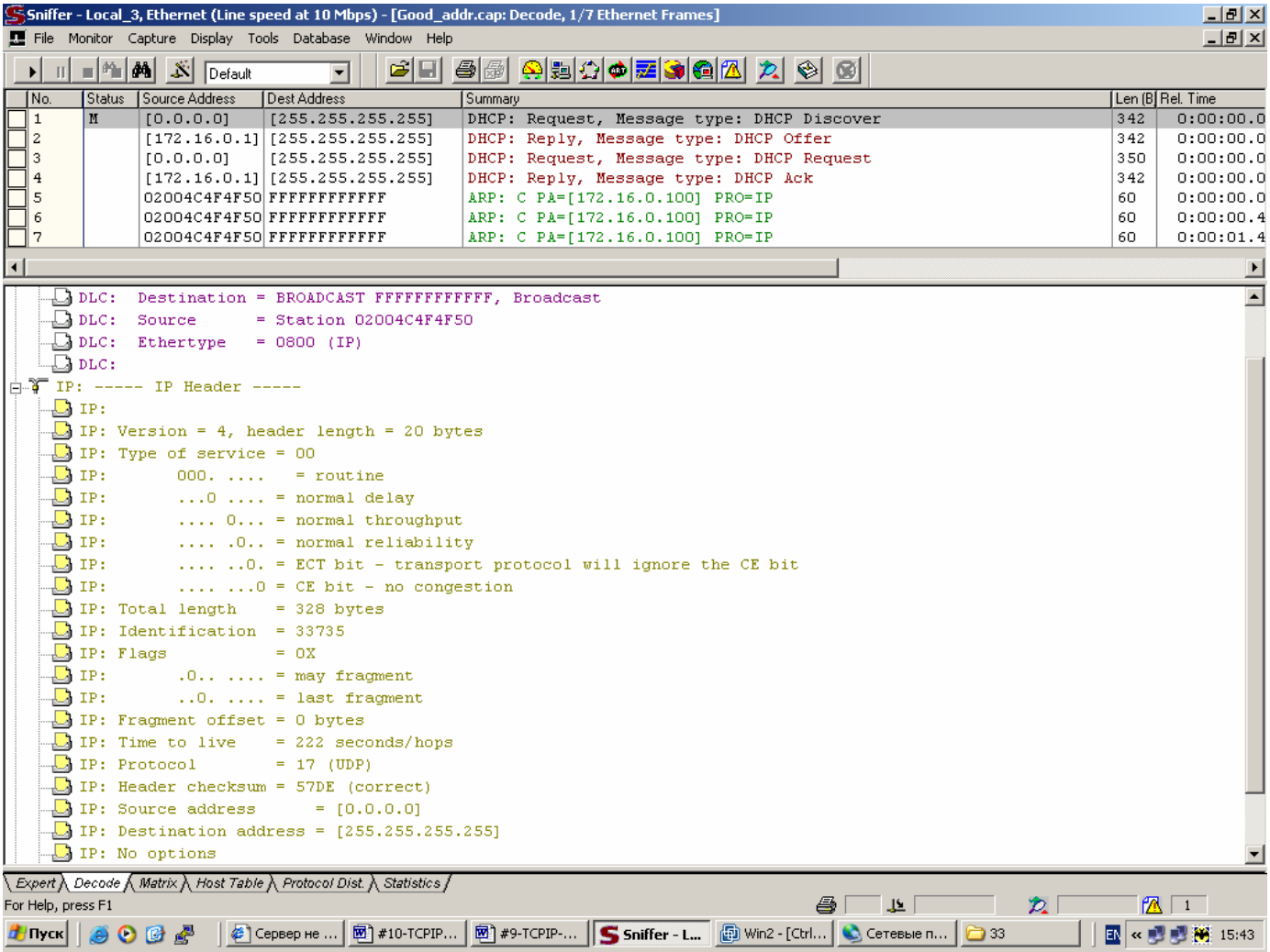
Переходим к следующему примеру: рассмотрим случай, когда клиент просит у сервера адрес, которым пользовался ранее и который может быть выдан клиенту. Вспомним, что клиенты запоминают адрес, которым пользовались ранее с целью получения этого же адреса в следующий раз, когда адрес понадобится, что удобно как само по себе, так и помогает экономнее расходовать адреса из скопа сервера. В моем примере DHCP сервер будет иметь адрес 172.16.0.1, сконфигурируем скоп 172.16.0.50 – 172.16.0.150. В рамках этой области пока не будем настраивать опций, просто настроим сервер таким образом, чтобы он выдавал адреса клиентам. Пусть клиент ранее использовал адрес 172.16.0.100, этого можно достичь, например, присвоив этот адрес клиенту статически, а затем переведя клиента на динамическое конфигурирование. Сделаем это и захватим трафик в анализаторе протоколов. Хотя мы пока не изучили формата пакета DHCP, однако мы уже сказали, что с помощью стационарной части заголовка DHCP пакета сервер и клиент оговаривают только IP адрес, который будет присвоен клиенту (и кое-что еще, но об этом позже), все же остальное делается в DHCP с помощью опций. Не вдаваясь пока в детали лишь отметим, что в заголовке DHCP может присутствовать специальная опция, с помощью которой клиент и просит у сервера желаемый адрес, более детально о формате пакета и опций в частности мы поговорим позднее.

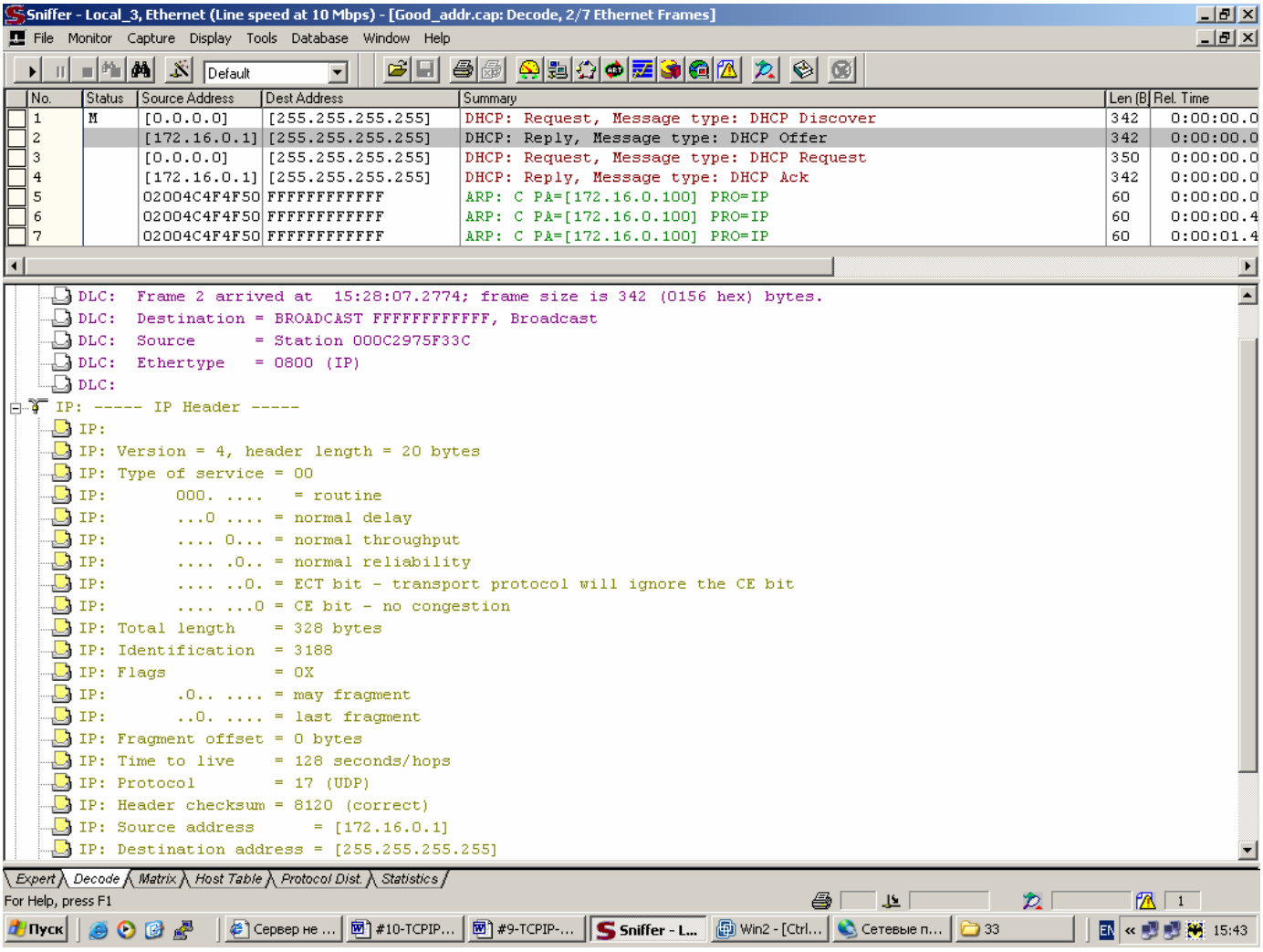
Рассмотрим пакет DHCPDISCOVER из данного примера, мы увидим опцию, с помощью которой клиент просит у сервера необходимый адрес, видно, что сервер предлагает именно этот адрес клиенту и клиент его получает (файл good_addr.cap):

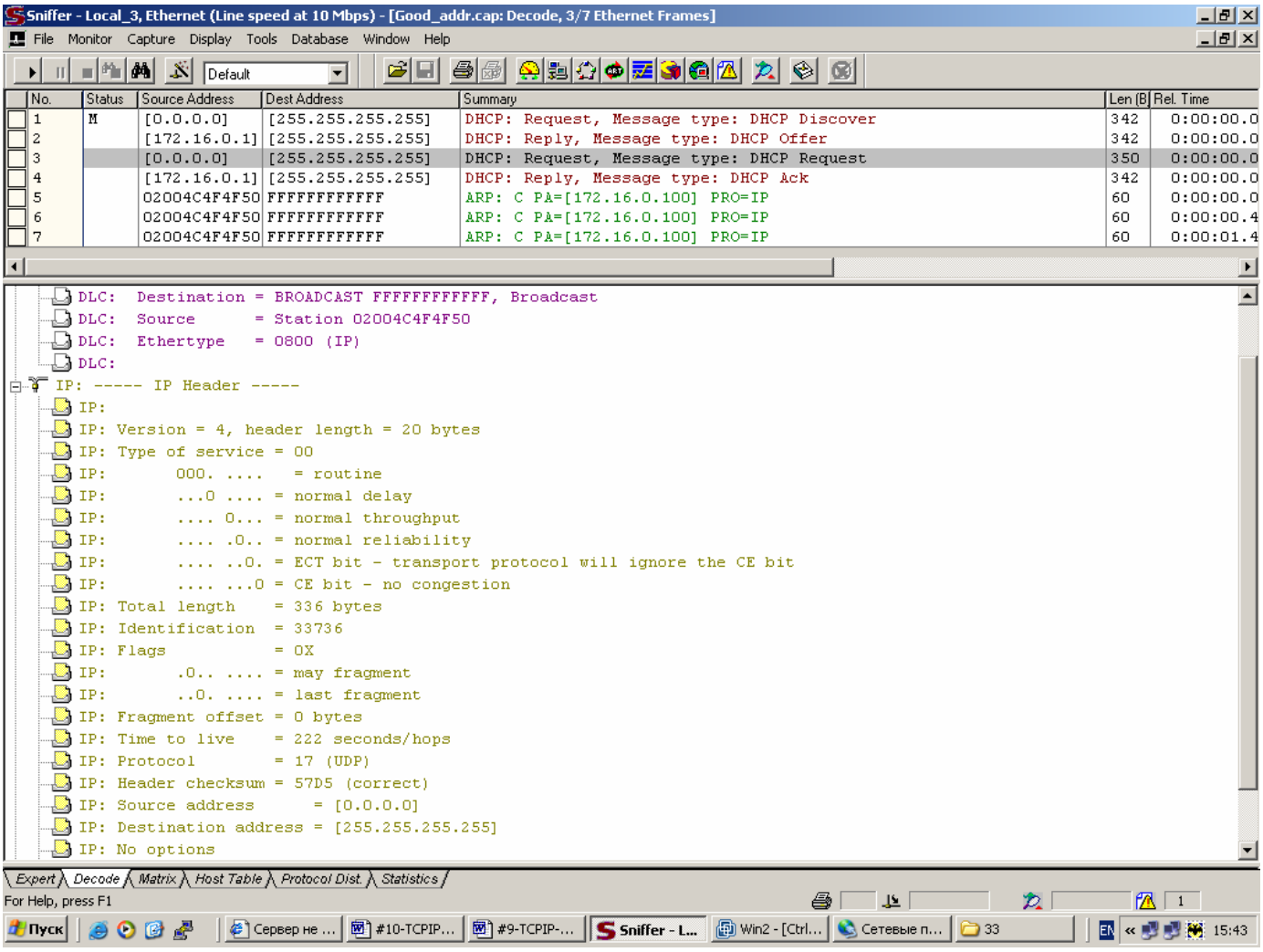


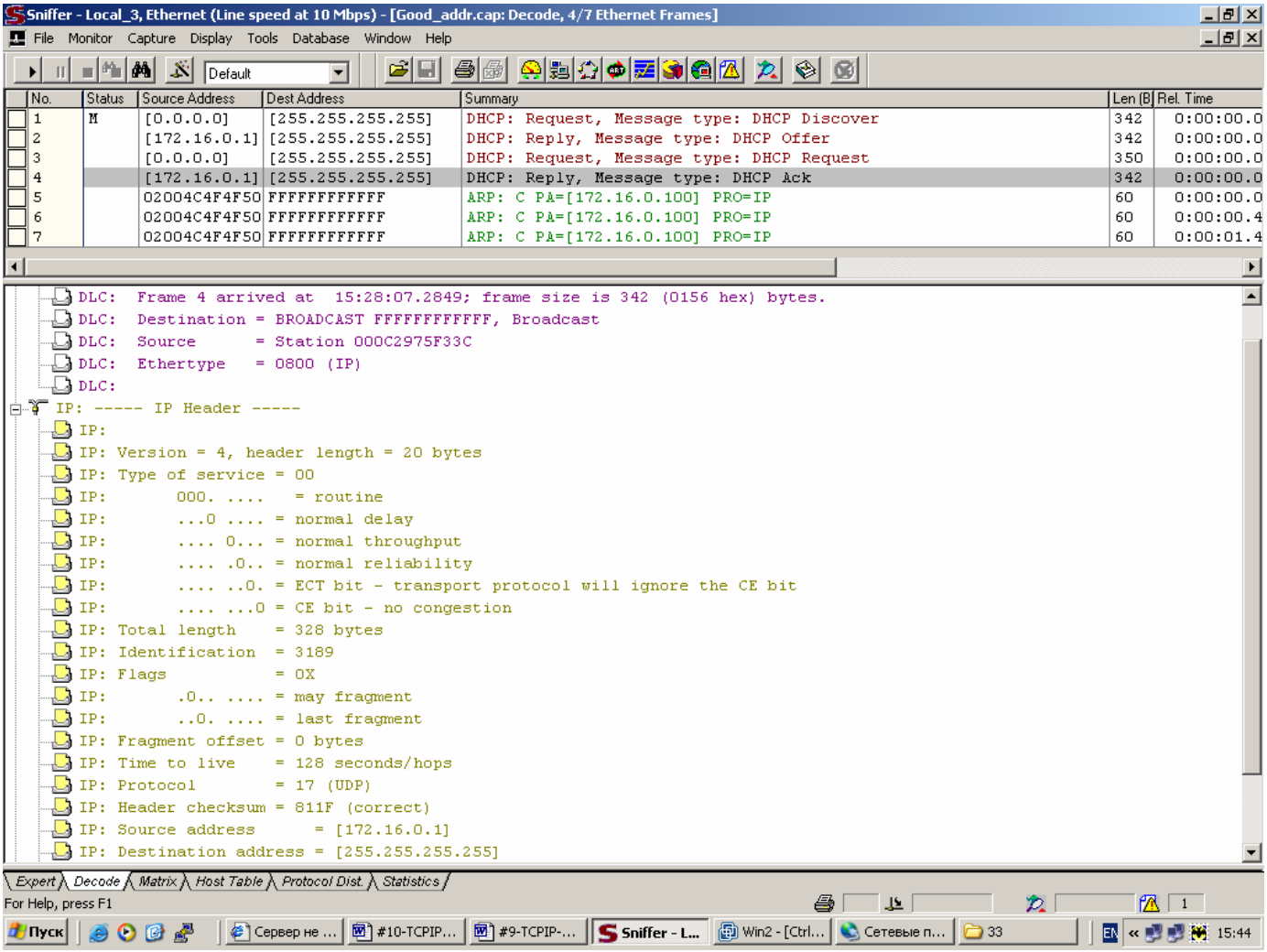
Рассмотрим пакет DHCPOFFER, подчеркнем, что данный пакет посылается широковещательно на канальном уровне и ограниченно широковещательно на третьем уровне, вспоминаем, для чего данный пакет отправляется именно с такими адресами получателя. Анализируем пакет DHCPREQEST, отмечаем, что и он отправляется широковещательно на втором и ограниченно широковещательно на третьем уровне, вспоминаем, какие цели преследует клиент,

посылая пакет таким образом. Так же анализируем пакет DHCPACK, подчеркиваем, что и он отправляется, так же как и все предыдущие пакеты в смысле адресов получателя на втором и третьем уровнях. Кроме того, отмечаем, что станция после получения адреса проверяет его с помощью ARP запросов, и, убедившись, что с адресом все в порядке начинает им пользоваться.




Sniffer - Local_3, Ethernet (Line speed at 10 Mbps) - [Good_addr.cap: Decode, 2/7 Ethernet Frames]
File Monitor Capture Display Tools Database Window Help
Default
No. Status Source Address Dest Address Summary Len (B) Rel. Time
1 M [0.0.0.0] [255.255.255.255] DHCP: Request, Message type: DHCP Discover 342 0:00:00.0
2 [172.16.0.1] [255.255.255.255] DHCP: Reply, Message type: DHCP Offer 342 0:00:00.0
3 [0.0.0.0] [255.255.255.255] DHCP: Request, Message type: DHCP Request 350 0:00:00.0
4 [172.16.0.1] [255.255.255.255] DHCP: Reply, Message type: DHCP Ack 342 0:00:00.0
5 02004C4F4F50 FFFFFFFFFFFF ARP: C PA=[172.16.0.100] PRO=IP 60 0:00:00.0
6 02004C4F4F50 FFFFFFFFFFFF ARP: C PA=[172.16.0.100] PRO=IP 60 0:00:00.4
7 02004C4F4F50 FFFFFFFFFFFF ARP: C PA=[172.16.0.100] PRO=IP 60 0:00:01.4
DLC: Frame 2 arrived at 15:28:07.2774; frame size is 342 (0156 hex) bytes.
DLC: Destination = BROADCAST FFFFFFFFFFFF, Broadcast
DLC: Source = Station 000C2975F33C
DLC: Ethertype = 0800 (IP)
DLC:
IP: ----- IP Header -----
IP:
IP: Version = 4, header length = 20 bytes
IP: Type of service = 00
IP: 000. .... = routine
IP: ...0 .... = normal delay
IP: .... 0... = normal throughput
IP: .... .0.. = normal reliability
IP: .... ..0. = ECT bit - transport protocol will ignore the CE bit
IP: ....0 = CE bit - no congestion
IP: Total length = 328 bytes
IP: Identification = 3188
IP: Flags = 0X
IP: .0.. .... = may fragment
IP: ..0. .... = last fragment
IP: Fragment offset = 0 bytes
IP: Time to live = 128 seconds/hops
IP: Protocol = 17 (UDP)
IP: Header checksum = 8120 (correct)
IP: Source address = [172.16.0.1]
IP: Destination address = [255.255.255.255]
Expert Decode Matrix Host Table Protocol Dist. Statistics
For Help, press F1
Пуск Сервер не ... #10-TCPIP... #9-TCPIP-... Sniffer - L... Win2 - [Ctrl... Сетевые п... 33 15:43

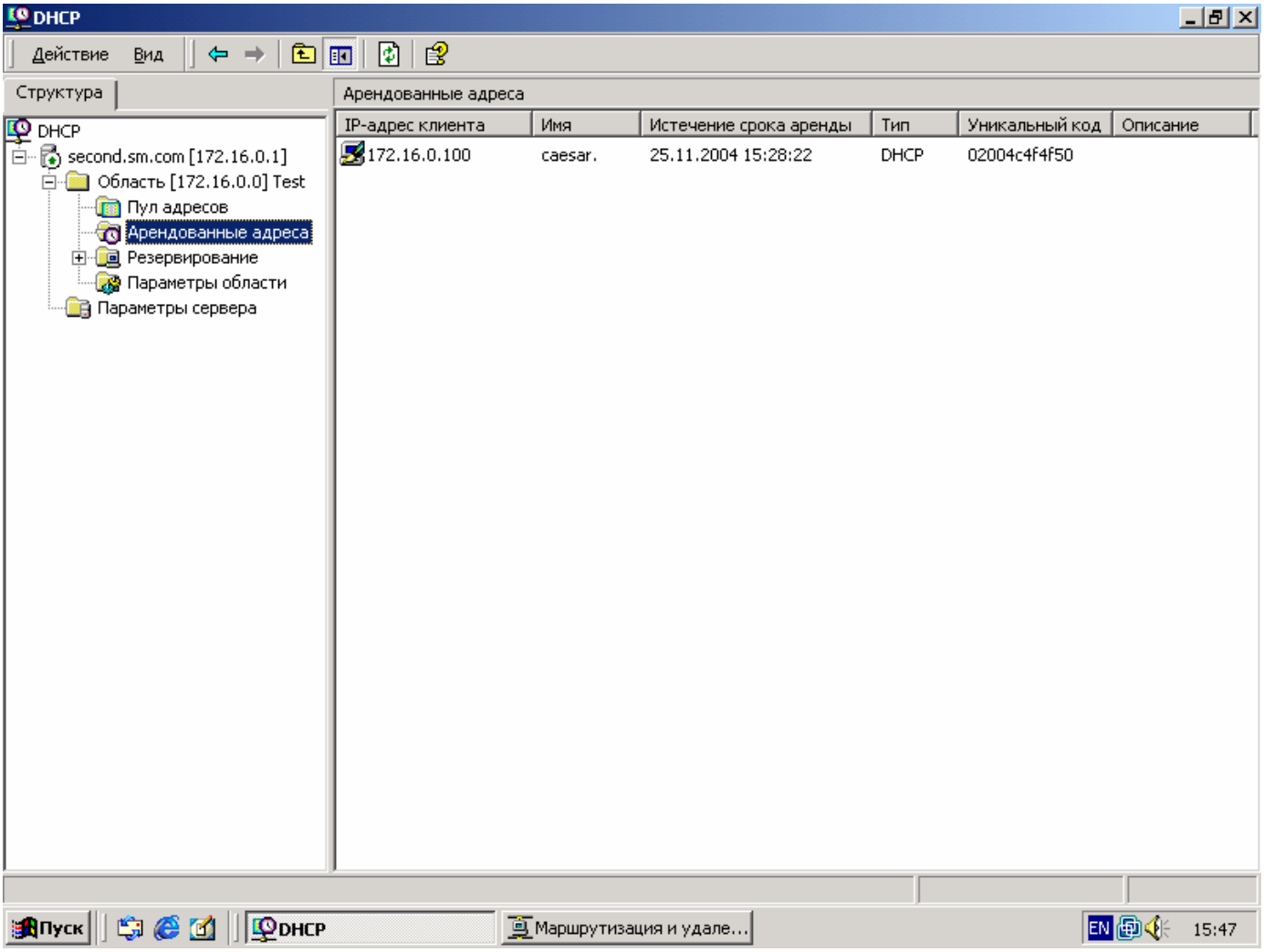

Sniffer - Local_3, Ethernet (Line speed at 10 Mbps) - [Good_addr.cap: Decode, 3/7 Ethernet Frames]
File Monitor Capture Display Tools Database Window Help
Default
No. Status Source Address Dest Address Summary Len (B) Rel. Time
1 M [0.0.0.0] [255.255.255.255] DHCP: Request, Message type: DHCP Discover 342 0:00:00.0
2 [172.16.0.1] [255.255.255.255] DHCP: Reply, Message type: DHCP Offer 342 0:00:00.0
3 [0.0.0.0] [255.255.255.255] DHCP: Request, Message type: DHCP Request 350 0:00:00.0
4 [172.16.0.1] [255.255.255.255] DHCP: Reply, Message type: DHCP Ack 342 0:00:00.0
5 02004C4F4F50 FFFFFFFFFFFF ARP: C PA=[172.16.0.100] PRO=IP 60 0:00:00.0
6 02004C4F4F50 FFFFFFFFFFFF ARP: C PA=[172.16.0.100] PRO=IP 60 0:00:00.4
7 02004C4F4F50 FFFFFFFFFFFF ARP: C PA=[172.16.0.100] PRO=IP 60 0:00:01.4
DLC: Destination = BROADCAST FFFFFFFFFFFF, Broadcast
DLC: Source = Station 02004C4F4F50
DLC: Ethertype = 0800 (IP)
DLC:
IP: ----- IP Header -----
IP:
IP: Version = 4, header length = 20 bytes
IP: Type of service = 00
IP: 000. .... = routine
IP: ...0 .... = normal delay
IP: .... 0... = normal throughput
IP: .... .0.. = normal reliability
IP: .... ..0. = ECT bit - transport protocol will ignore the CE bit
IP: ....0 = CE bit - no congestion
IP: Total length = 336 bytes
IP: Identification = 33736
IP: Flags = 0X
IP: .0.. .... = may fragment
IP: ..0. .... = last fragment
IP: Fragment offset = 0 bytes
IP: Time to live = 222 seconds/hops
IP: Protocol = 17 (UDP)
IP: Header checksum = 57D5 (correct)
IP: Source address = [0.0.0.0]
IP: Destination address = [255.255.255.255]
IP: No options
Expert Decode Matrix Host Table Protocol Dist. Statistics
For Help, press F1
Пуск Сервер не ... #10-TCPIP... #9-TCPIP-... Sniffer - L... Win2 - [Ctrl... Сетевые п... 33 15:43



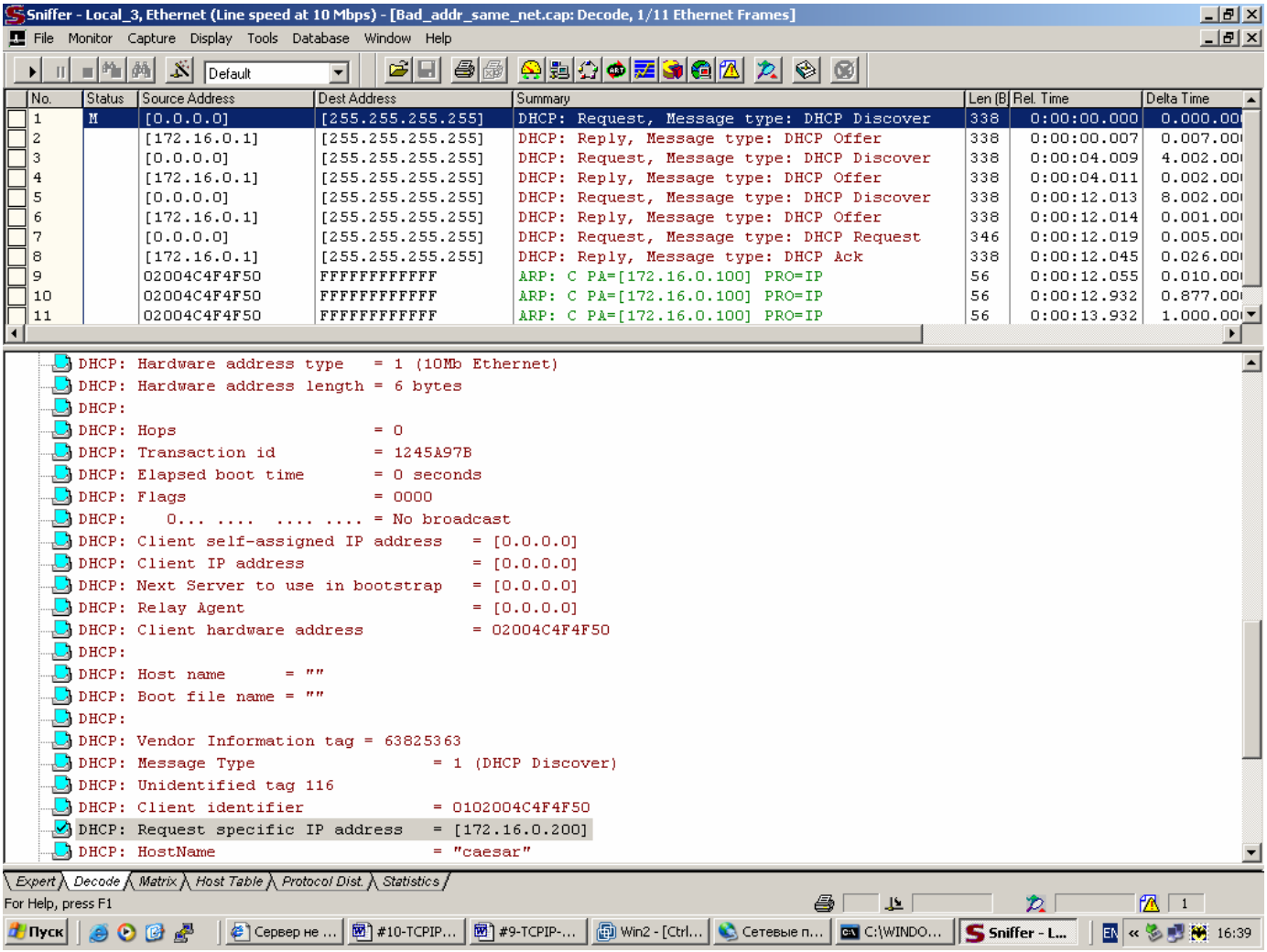
Утилита ipconfig.exe, запущенная на клиенте:

```
C:\>ipconfig /all
Настройка протокола IP для Windows
        Имя компьютера  . . . . . . . . . : caesar
        Основной DNS-суффикс  . . . . . . :
        Тип узла. . . . . . . . . . . . . : неизвестный
        IP-маршрутизация включена . . . . : да
        WINS-прокси включен . . . . . . . : да

Подключение по локальной сети - Ethernet адаптер:
        DNS-суффикс этого подключения . . :
        Описание  . . . . . . . . . . . . : Realtek RTL8139
        Физический адрес. . . . . . . . . : 02-00-4C-4F-4F-50
        Dhcp включен. . . . . . . . . . . : да
        Автонастройка включена  . . . . . : да
        IP-адрес  . . . . . . . . . . . . : 172.16.0.100
        Маска подсети . . . . . . . . . . : 255.255.0.0
        Основной шлюз . . . . . . . . . . :
        DHCP-сервер . . . . . . . . . . . : 172.16.0.1
        Аренда получена . . . . . . . . . : 17 ноября 2004 г. 15:28:08
        Аренда истекает . . . . . . . . . : 25 ноября 2004 г. 15:28:08
```

Так же убедимся, что в консоли управления DHCP сервером, что в папке «арендованные адреса» появилась запись о нашем клиенте. Отмечаем, что сервер в ходе коммуникаций с клиентом выяснил его MAC адрес (Уникальный код) и понятное имя (caesar).

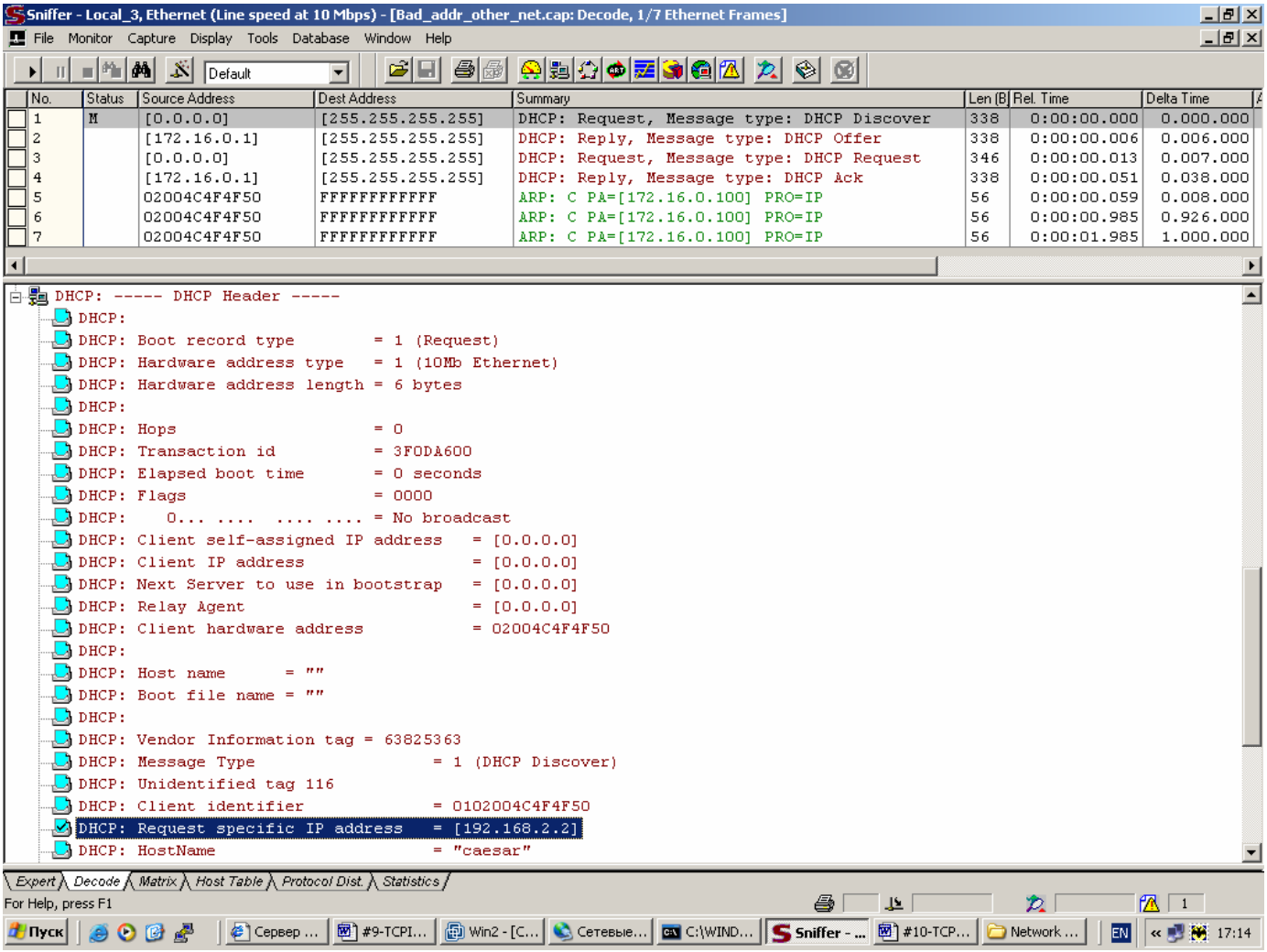


Рассматриваем следующий пример: пусть теперь клиент просит адрес, который не может быть ему выдан, но формально этот адрес принадлежит той же сети, в которой DHCP сервер выдает адреса из своего скопа. Для этого присвоим нашему клиенту статический адрес 171.16.0.200, а затем переведем клиента на динамическое конфигурирование. Убедимся, что клиент просит в пакете DHCPDISCOVERY тот адрес, которым пользовался ранее, а именно 172.16.0.200 (файл bad_addr_same_net.cap):



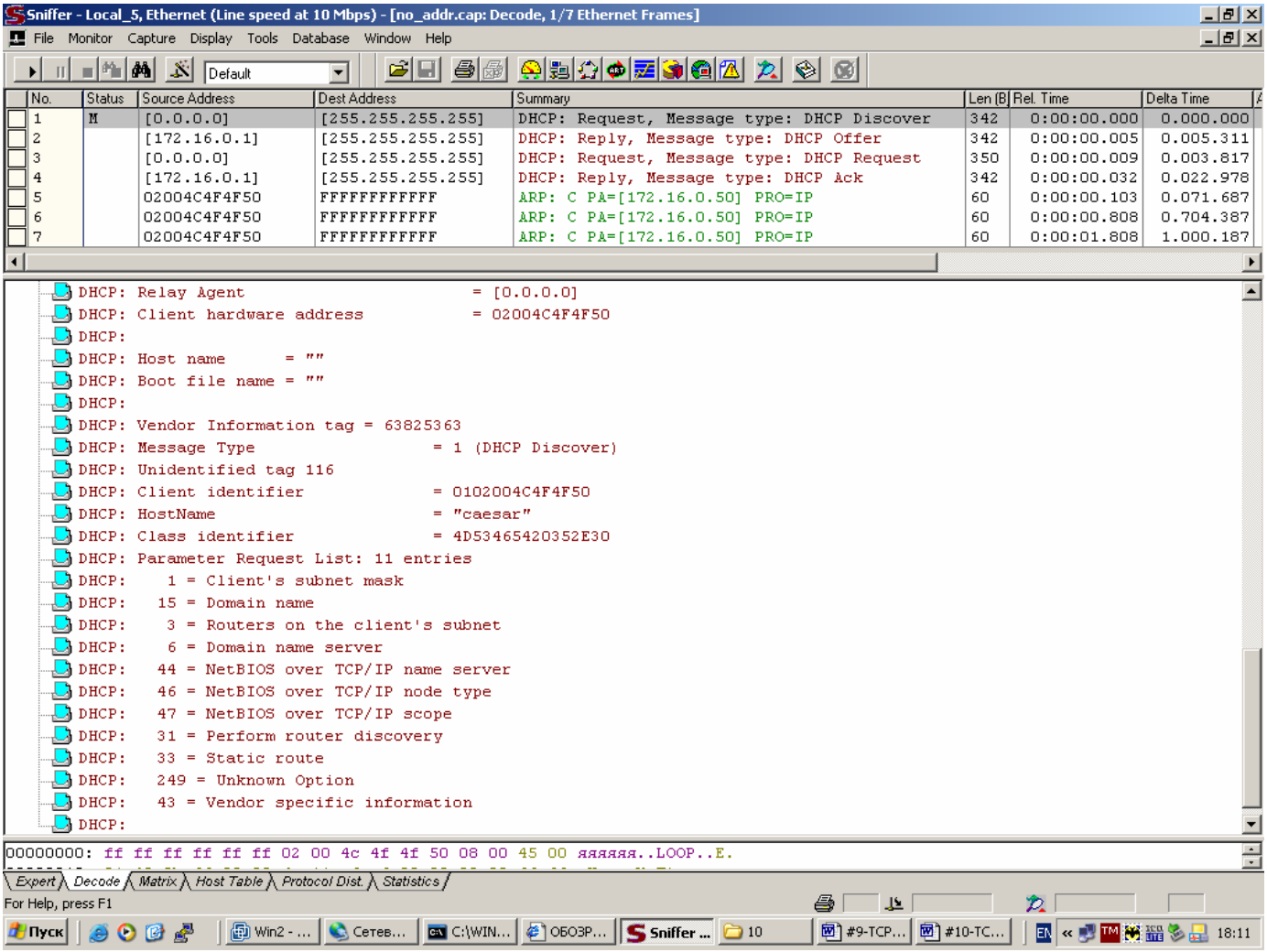
Очевидно, данный адрес не входит в скоп, настроенный нами на сервере, поэтому сервер не может предоставить клиенту данного адреса. Но сервер, как и клиент, стремится помнить, кому он ранее выдавал какие адреса, поэтому выдает клиенту не какой ни будь адрес, а тот адрес, который, по мнению сервера, клиент получал в прошлый раз (напоминаем, что сервер идентифицирует клиента по его MAC адресу). Клиент получает это предложение, но оно не соответствует тому, что клиент просил. При этом адрес, который выдал сервер, находится в той же сети, что и адрес, который просил клиент, а это дает клиенту надежду, что возможно в этой сети найдется другой DHCP сервер, который сможет выдать клиенту желаемый адрес 172.16.0.200. Поэтому клиент НЕ принимает поступившего предложения, а повторяет через некоторое время свой запрос DHCPDISCOVER! Из рисунка видно, что клиент повторяет свой DHCPDISCOVER дважды (т.е. всего посылает три DHCPDISCOVER), с интервалами 4 и 8 секунд соответственно, и когда на последний из этих DHCPDISCOVER не поступает того предложения, которого ждет клиент, только тогда лишь клиент соглашается принять предложение адреса 172.16.0.100, что и делает, посылая пакет DHCPREQEST. Заметим, что в прошлом примере вся процедура получения адреса заняла около 1,4 секунды, в данном же примере клиент получал адрес около 14 секунд! Важно, чтобы Вы четко понимали, с чем связаны рассмотренные особенности взаимодействия, т.е. важно, чтобы Вы учились правильно понимать природу задержек в сети: они легко объяснимы и часто легко устранимы. Очевидно, в следующий раз наш клиент ужу не будет просить адрес 172.16.0.200, а будет просить последний использовавшийся им адрес 172.16.0.100, т.е. впредь подобных задержек в сети не будет.

Если бы клиент изначально просил адрес, который сервер ему не может предоставить, но этот адрес принадлежал бы какой-то другой сети, то все было бы проще – рассматриваем на примере. Присвоим клиенту статический адрес 192.168.2.2, переведем клиента на динамическое получение адреса и проанализируем трафик (файл bad_addr_other_net.cap):

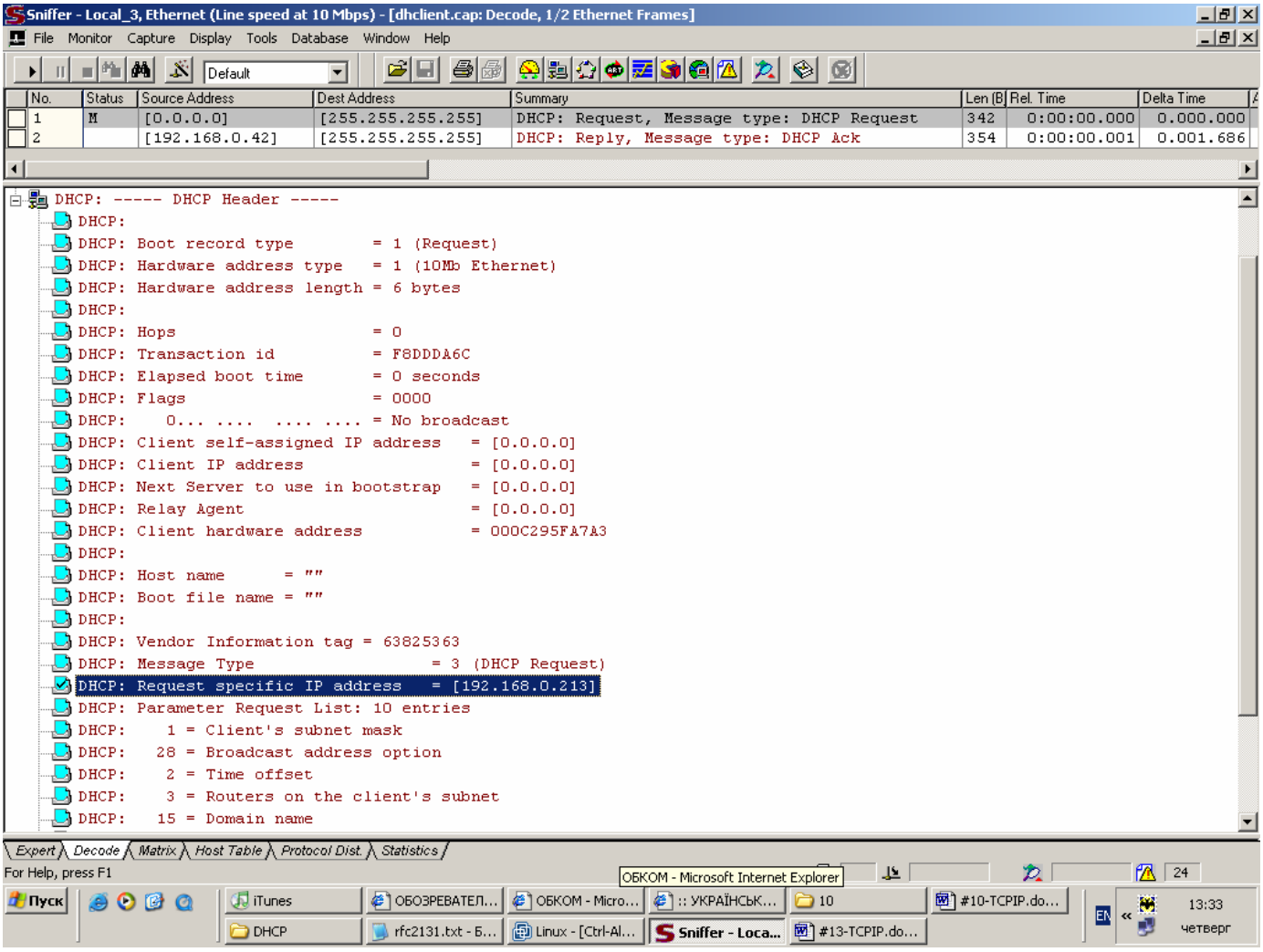


Видно, что клиент просит адрес 192.168.2.2, ему предлагают 172.16.0.100 (сервер помнит, каким адресом пользовался обладатель данного MAC адреса ранее), и клиент сразу принимает данное предложение, так как очевидно, в сети, где предлагают адреса из диапазона 172.16.0.0/16 не предложат адреса из сети 192.168.2.0/24, таким образом клиент без лишних проволочек получает необходимый адрес.

Так же можно попытаться сымитировать ситуацию, когда клиент не просит вообще никакого адреса в пакете DHCPDISCOVER, для этого необходимо, к примеру, удалить и вновь добавить интерфейс клиенту, имитируя тем самым новый адаптер без «истории». DHCP трафик в таком случае весьма прост – клиент ничего не просит в пакете DHCPDISCOVER и получает первый свободный адрес из скопа сервера (файл no_addr.cap):

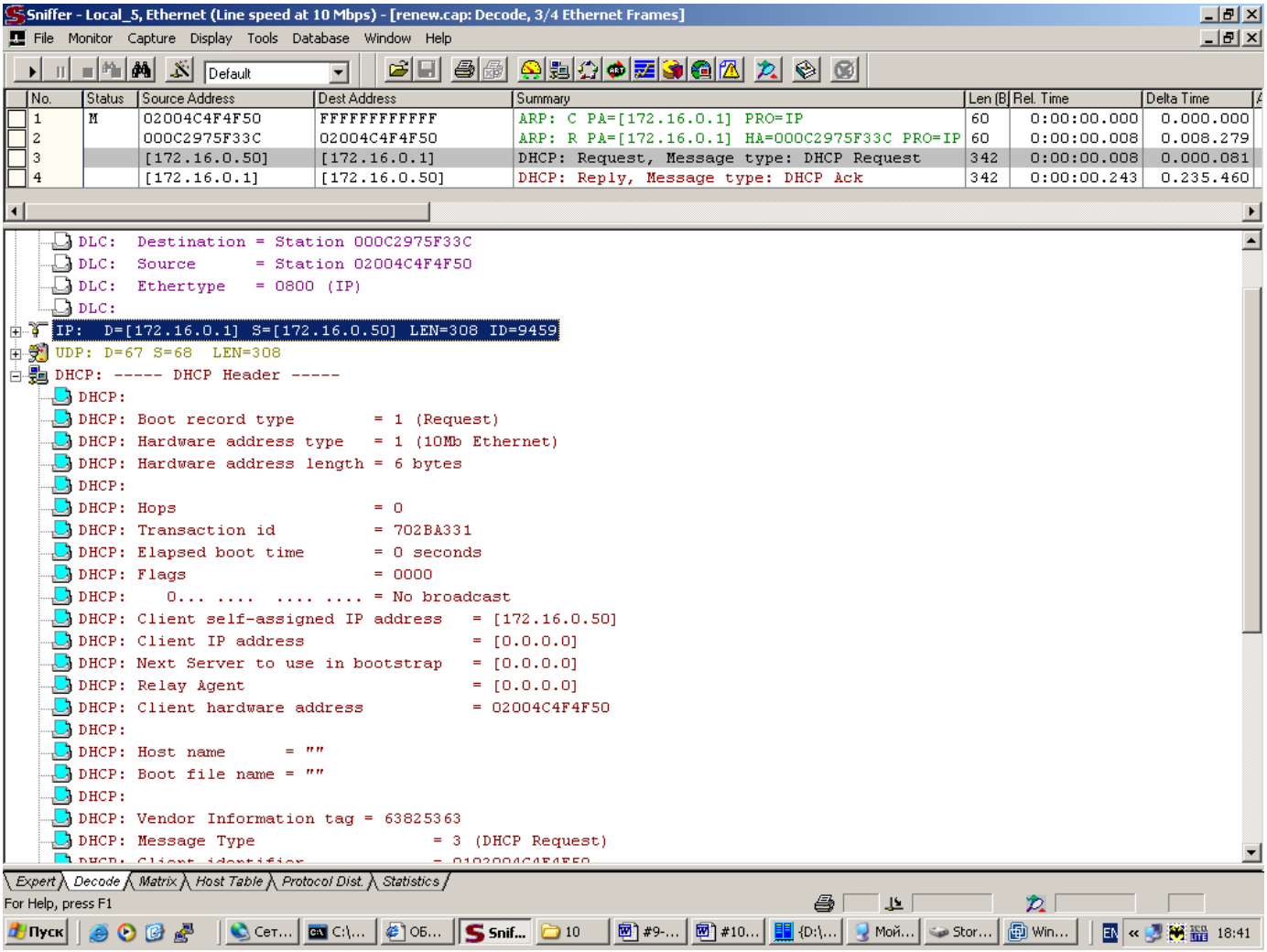


Далее рассмотрим еще один вариант DHCP взаимодействия, описанный в RFC2131. В том случае, если DHCP клиент помнит, какой адрес он использовал ранее, то он МОЖЕТ (но не обязан) пропустить некоторые рассмотренные нами ранее шаги. Клиент может широковещательно отправить пакет DHCPREQUEST, в котором НЕ должен с помощью упомянутой нами ранее опции сообщить серверу, какой адрес он хотел бы получить. Сервер НЕ должен проверять данный адрес на возможные конфликты с помощью ICMP эхо запросов (если вообще выполняет такие проверки, детальнее об этом мы поговорим позднее), так как клиент может ответить на этот запрос. Данный способ взаимодействия является весьма выгодным, так как уменьшает количество отправляемых в сеть пакетов, которые к тому же еще и широковещательные на канальном уровне и ограниченно широковещательные на сетевом уровне. Таким образом действуют многие DHCP клиенты в операционной системе Linux, как мы уже видели, DHCP клиент Windows выполняет полные коммуникации. Рассмотрим такое взаимодействие на практике в анализаторе протоколов. Для этого рассмотрим пример (файл dhclient.cap):



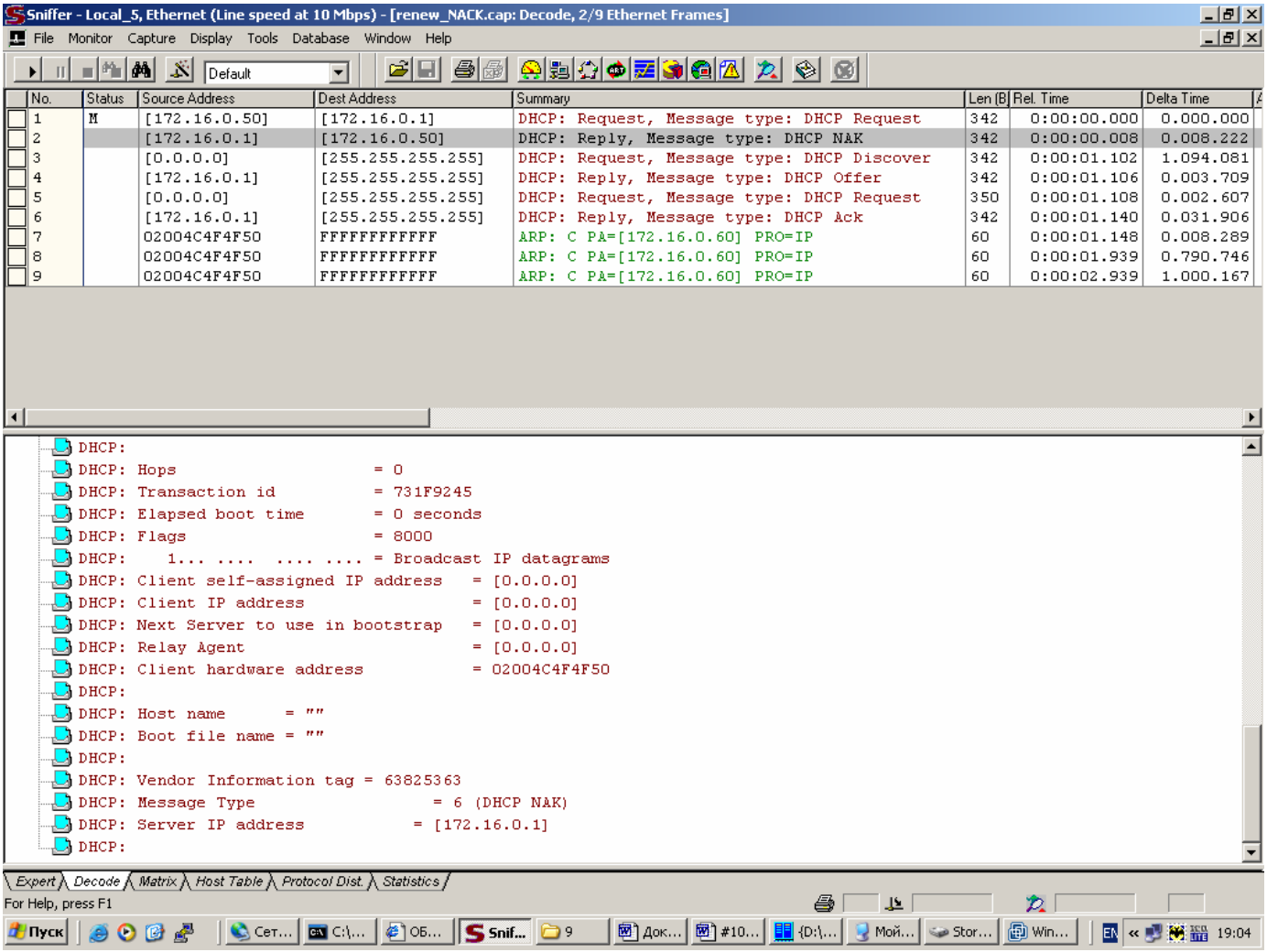
Видно, что клиент начинает взаимодействие сразу с пакета DHCPREQUEST, что вполне соответствует RFC2131 и наш сервер под управлением Windows 2000 поддерживает такой стиль работы клиента.

Теперь рассмотрим следующую штатную ситуацию – клиент обновляет адрес, которым пользуется, так как это необходимо сделать в соответствии со временем аренды, которое установил сервер. Разумеется, мы не будем ждать истечения этого времени, просто дадим команду ipconfig.exe /renew когда у клиента уже есть адрес, полученный от DHCP сервера. Проанализируем полученный трафик в анализаторе (файл renew.cap), что мы увидим? Во-первых, трафик состоит из двух пакетов (как уже говорилось нами ранее), DHCPREQEST и DHCPACK. Во-вторых, трафик не использует широковещание, а является направленным, что неудивительно, так как у клиента в этой ситуации уже есть IP адрес. Кроме того, мы видим, что перед обменом DHCP пакетами клиент посылает ARP запрос серверу – это обычный ARP запрос, посылаемый с самой что ни на есть прямой целью – выяснения MAC адреса DHCP сервера, что необходимо для отправки ему направленных кадров канального уровня. Так же отметим, что после такой проверки, если она конечно завершилась удачно, как в наем примере, ARP запросы для проверки дублирования адреса не посылаются, действительно, клиент просто продолжает пользоваться данным адресом.



Рассмотрим следующий пример: пусть клиент снова пытается продолжить пользоваться некоторым адресом, но сервер отказывает ему в этом. Одной из причин того, что сервер отказывает клиенту может быть изменение настроек на сервере. Например, у клиента был адрес 172.16.0.50, но администратор настроил резервирование, которое предполагает выдачу клиенту адреса 172.16.0.60. Когда клиент попытается продлить аренду своего адреса 172.16.0.50 сервер будет вынужден отказать клиенту и затем назначить ему адрес 172.16.0.60. Продемонстрируем это на практике (файл renew_NACK.cap). Мы видим, что клиент посылает DHCPREQEST в надежде обновить свой адрес, но в ответ получает от сервера DHCPNACK, после чего клиент вынужден начинать взаимодействие с DHCP сервером с самого начала, т.е. с посылки пакета DHCPDISCOVER, причем в этом пакете клиент уже не просит себе никакого адреса, так как тот адрес, который клиент хотел бы получить ему УЖЕ сервер отказался выдавать. Так же обратите внимание на то, что утилита ipconfig.exe работает не слишком корректно: она выдает на экран сообщение об ошибке, причем его текст крайне не внятен, в то время как на самом деле взаимодействие завершилось вполне корректно, т.к. клиент все же получил IP адрес, хотя и не тот, который просил. В том, что адрес получен, можно убедится, дав команду ipconfig.exe:

```
C:\>ipconfig /renew
Настройка протокола IP для Windows

Произошла ошибка при обновлении интерфейса "Подключение по локальной сети 4": От
казано в доступе.

C:\>ipconfig
Настройка протокола IP для Windows
Подключение по локальной сети - Ethernet адаптер:

        DNS-суффикс этого подключения . . :
        IP-адрес  . . . . . . . . . . . . : 172.16.0.60
        Маска подсети . . . . . . . . . . : 255.255.0.0
        Основной шлюз . . . . . . . . . . :
```

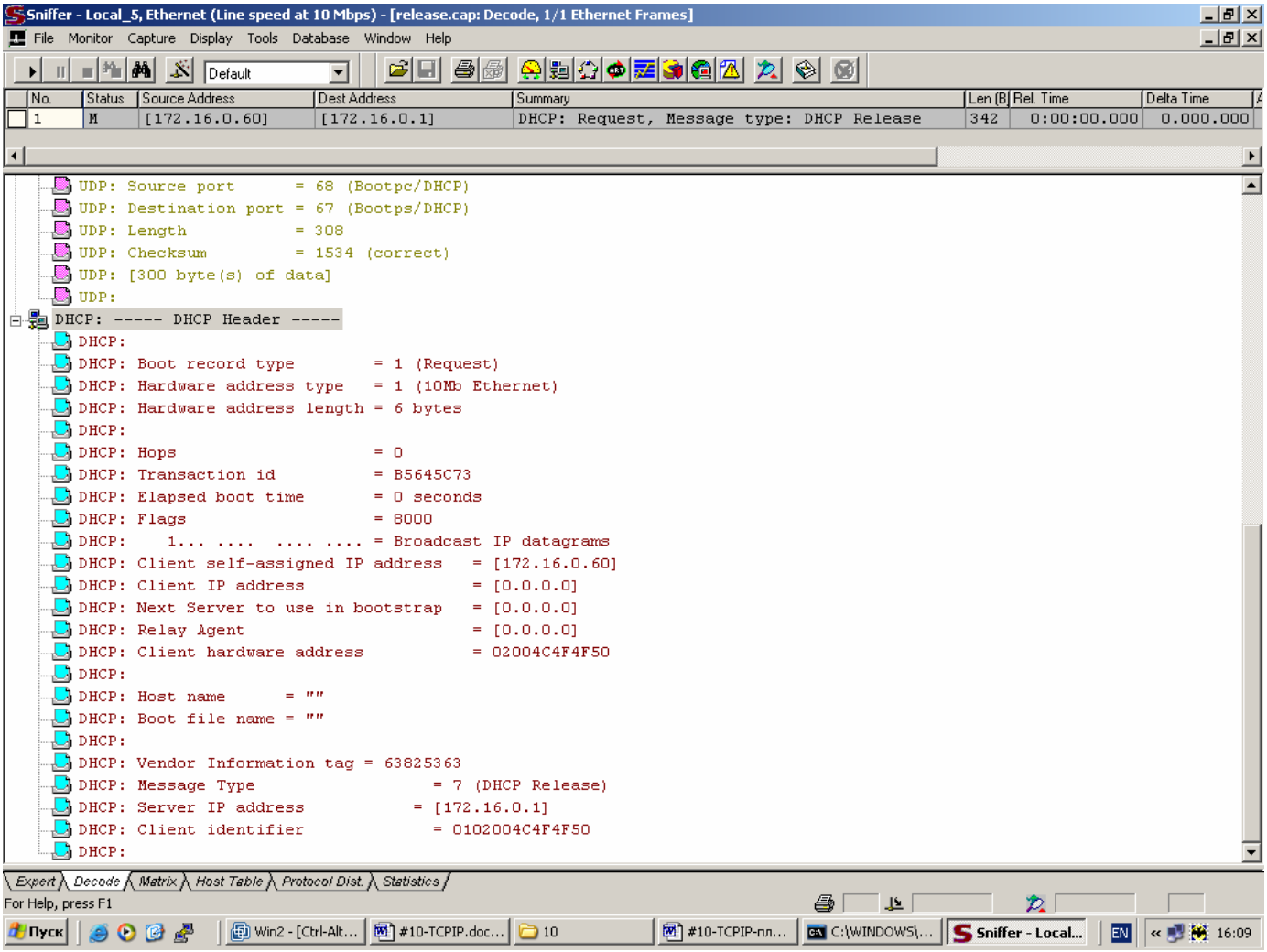


Рассматриваем следующий пример – использование сообщения DHCPRELEASE. С помощью этого сообщения, как говорилось выше, клиент сообщает серверу, что он больше не собирается пользоваться данным адресом. Например, такое сообщение должны посылать DHCP серверу клиенты, когда завершают работу в сети. Но с помощью утилиты ipconfig.exe можно заставить DHCP клиента послать такое сообщение и освободить адрес, используя ключ /release. Продемонстрируем как это работает. Заставим узел получить адрес от DHCP сервера, а затем дадим команду ipconfig.exe /release, проведем анализа трафика, убедимся, что клиент утратил адрес, а сервер считает данный адрес свободным (файл release.cap).

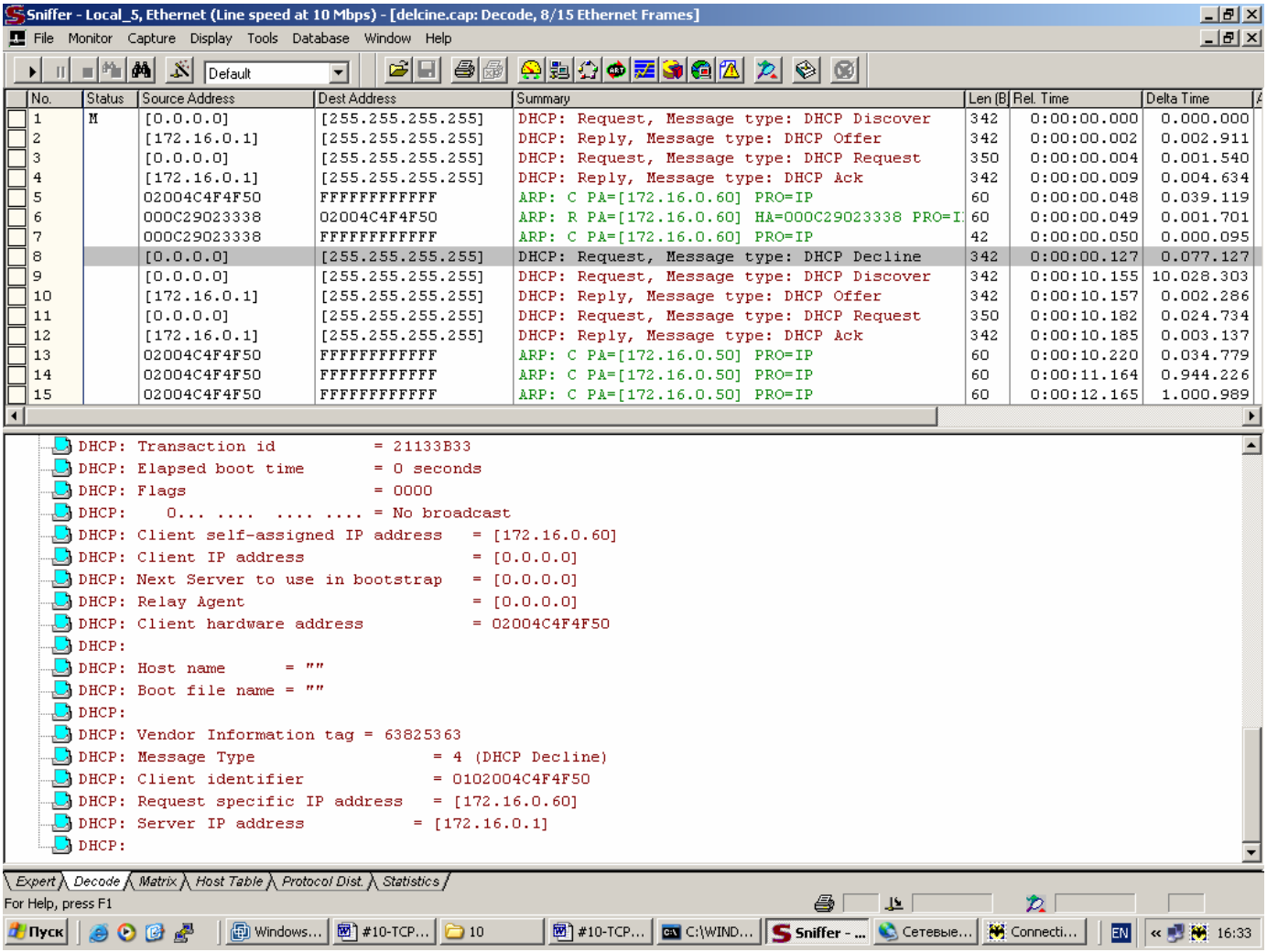
Результат работы утилиты ipconfig.exe:

```
C:\>ipconfig /release
Настройка протокола IP для Windows
Подключение по локальной сети - Ethernet адаптер:

        DNS-суффикс этого подключения . . :
        IP-адрес  . . . . . . . . . . . . : 0.0.0.0
        Маска подсети . . . . . . . . . . : 0.0.0.0
        Основной шлюз . . . . . . . . . . :
```

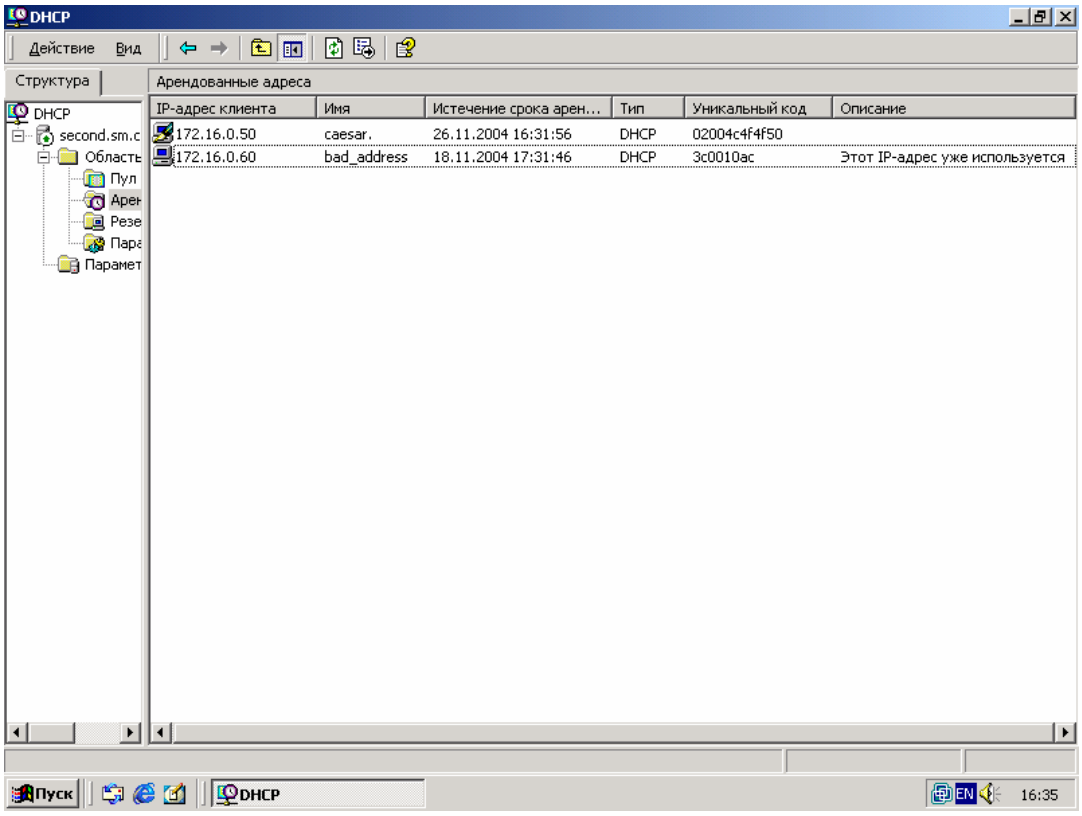
Трафик в анализаторе показывает нам, что данный пакет не подтверждается сервером, что означает, что в случае потери данного пакета сервер не сможет корректно освободить данный адрес, что согласуется с тем, о чем говорилось выше и в таком случае адрес будет снова задействован либо по окончании срока аренды, либо когда данный клиент снова попытается получить данный адрес.



Теперь построим сеть, в которой реализуем ситуацию, в которой посылаются сообщения DHCPDECLINE. Напоминаем, что это происходит в случае, если когда клиент получает от сервера IP адрес, но при проверке дублирования данного адреса с помощью ARP запросов клиент выясняет, что этот адрес уже используется кем-то и уведомляет с помощью сообщения DHCPDECLINE сервер об этом. Как может возникнуть такая ситуация? Например, на одном из узлов сети статически указан IP адрес, принадлежащий скопу DHCP сервера, и когда сервер выдает этот адрес клиенту, получается вышеописанная неприятность. Сконфигурируем один из узлов нашей сети статически на использование некоторого IP адреса, например, 172.16.0.60 и отключим пока интерфейс. «Поможем» клиенту «захотеть» получить от DHCP сервера именно этот адрес, предварительно тоже дав ему статический адрес 172.16.0.60, затем переведем клиента на использование динамического адреса и убедимся, что клиент получил от DHCP сервера именно адрес 172.16.0.60. Затем освободим данный адрес с помощью команды DHCPRELEASE и включим интерфейс второго узла, который статически настроен на использование адреса 172.16.0.60. Теперь все готово для эксперимента: в сети есть узел со статически сконфигурированным IP адресом 172.16.0.60, и есть узел, пока адреса не имеющий, но он попытается попросить у своего DHCP сервера именно этот адрес и получит его, так как DHCP сервер понятия не имеет о том, что адрес 172.16.0.60 статически используется в сети. Рассматриваем пример с помощью анализатора трафика, дав на DHCP клиенте команду ipconfig.exe /renew (файл decline.cap):



Как видим, сервер выдает клиенту адрес 192.168.0.60, клиент проверяет его ARP запросами, выясняет, что адресом пользоваться нельзя, уведомляет об этом сервер и начинает новую транзакцию по получению адреса. При этом сервер в своем скопе помечает данный адрес как «неисправный» сроком на один час:



Так же подчеркиваем, что диагностическое сообщение утилиты ipconfig.exe снова не вполне адекватно:

```
C:\>ipconfig /renew
Настройка протокола IP для Windows

Произошла ошибка при обновлении интерфейса "Подключение по локальной сети": DHCP-клиент запросил IP-адрес, который уже используется в сети. Локальный интерфейс будет отключен до тех пор, пока DHCP-клиент сможет получить новый адрес.
```

Звучит угрожающе, хотя на самом деле интерфейс получил адрес, и в этом легко убедиться просто еще раз запустив ipconfig.exe:

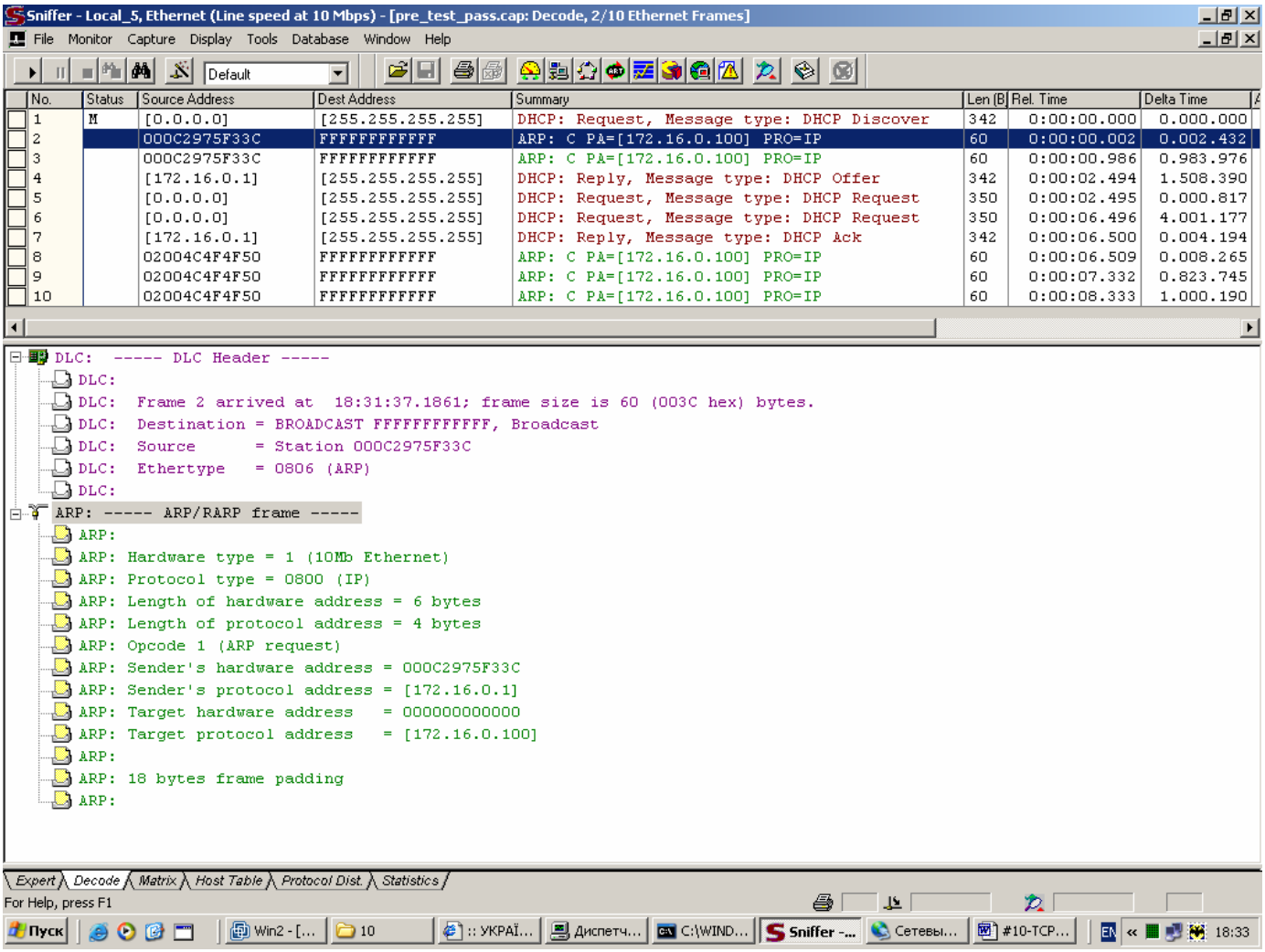
```
C:\Documents and Settings\Администратор>ipconfig
Настройка протокола IP для Windows
Подключение по локальной сети - Ethernet адаптер:

        DNS-суффикс этого подключения . . :
        IP-адрес  . . . . . . . . . . . . : 172.16.0.50
        Маска подсети . . . . . . . . . . : 255.255.0.0
        Основной шлюз . . . . . . . . . . :
```

Убедимся с помощью анализатора, что в пакете DHCPDISCOVER второй транзакции клиент уже не просит никакого адреса и поэтому получает первый свободный адрес, а именно 172.16.0.50.

Как видим, предоставление клиенту заведомо неподходящих адресов – неудачная практика. А может сервер мог бы сам, до того, как предлагает адрес клиенту, провести его проверку? Было бы это хорошо или плохо? На первый взгляд кажется, что это было бы хорошо, так как клиенты не получали бы неподходящих (занятых) адресов. С другой стороны в таком подходе есть минус – сервер перед каждой выдачей адреса должен проверить его допустимость, т.е. в каждой транзакции по протоколу DHCP появится ЛИШНЯЯ пауза – время, в течение которого DHCP сервер ждет ответа от потенциального владельца адреса, который сервер предполагает выдавать клиенту. При этом если владелец адреса в сети есть, то он пришлет совой ответ и сервер достаточно быстро поймет, что адрес предлагать нельзя, а если адрес, который сервер собирается предлагать, в сети не используется, то в этом случае задержка в коммуникациях будет больше, так как серверу придется ждать некоторый таймаут (относительно заметный) прежде чем понять, что адрес допустим к использованию. При этом такая проверка, проводимая сервером совершенно не избавляет клиента от проведения собственной проверки, так как клиент не знает, что сервер такую проверку выполнил! Следовательно, если проблема дублирования адресов возникает в сети редко, то лишняя проверка сервером – это просто лишняя задержка перед обслуживанием клиента. Отказ от таких проверок, выполняемых сервером приведет, как мы знаем к тому, что клиент, которому предложили неудачный адрес, получит нормальный адрес с задержкой, т.е. проведение проверки сервером замедляет НОРМАЛЬНЫЕ DHCP транзакции, а отказ от такой проверки замедляет НЕУДАЧНЫЕ DHCP транзакции. Следовательно, подобную проверку стоит проводить только в случае, если по некоторым причинам (о некоторых из них – позднее) в сети серверами DHCP часто предлагаются уже занятые адреса, и не стоит проводить, если такие ситуации редки.

Рассмотрите страницу свойств DHCP сервера, и Вы найдете там возможность сконфигурировать количество проверок, который может делать сервер, отправляя на проверяемый адрес ICMP эхо-пакеты перед тем, как предлагать этот адрес клиенту. Ясно, что чем больше проверок, выполняет сервер, тем надежнее он определяет, что адрес свободен (если пришел ответ – то это, во-первых, обычно происходит быстро, а во-вторых, точно означает, что адрес занят, но пакеты могут теряться), но тем большая задержка будет иметь место перед тем, как клиент получит IP адрес. Так же отмечаем, что сервер не удовлетворяется проверкой с помощью ARP запросов, а выполняет проверку посылкой ICMP эхо запроса на проверяемый IP адрес, очевидно, перед этим все равно необходимо послать ARP запрос.

Рассмотрим это на примере. Укажем серверу, что он должен делать две проверки для обнаружения конфликтов (больше не рекомендует Microsoft) и посмотрим для начала, как сервер делает проверку допустимости адреса, который действительно не используется в сети (файл pre_test_pass.cap):

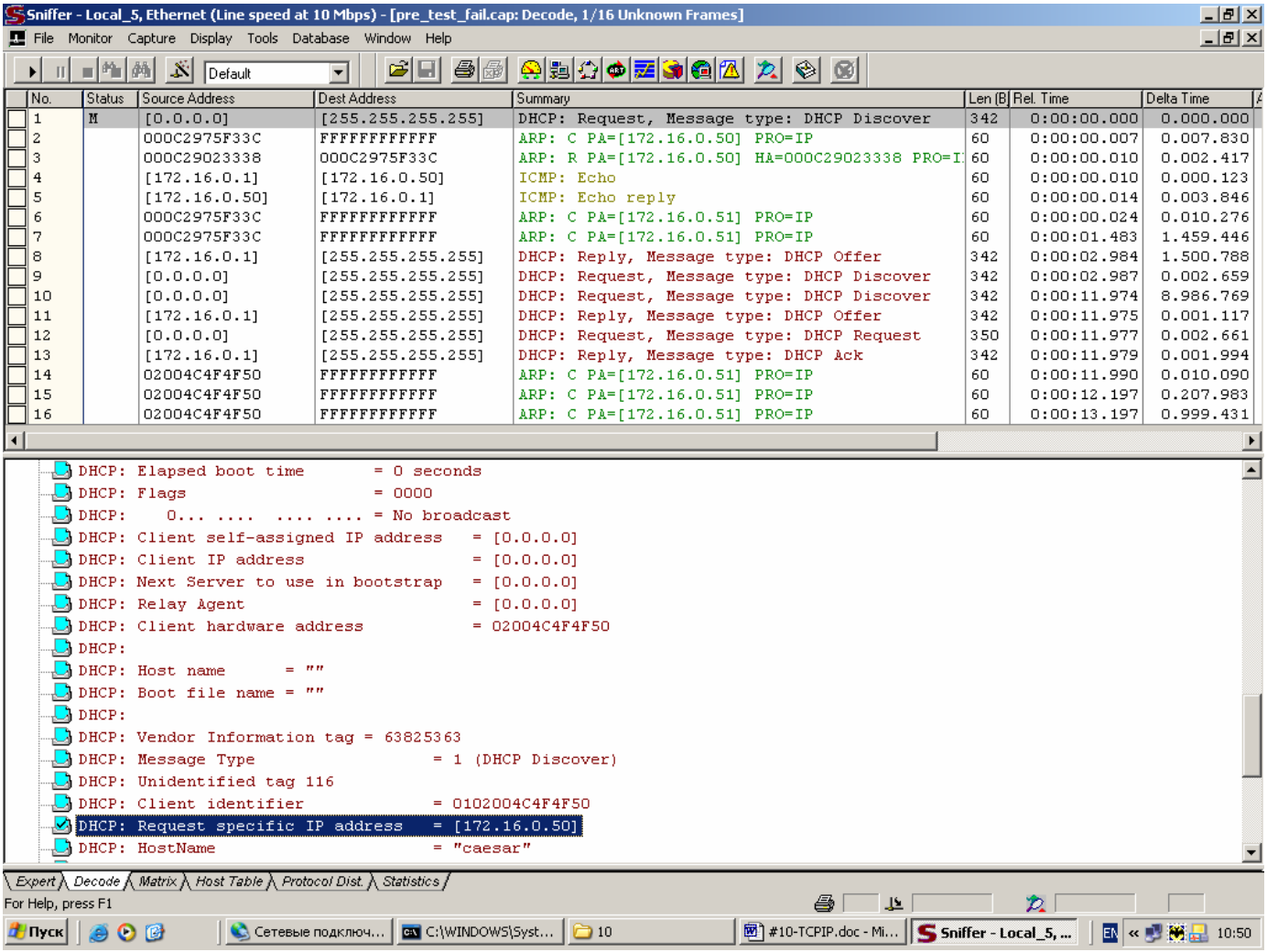


Отметим, что в данной транзакции мы видим два пакета DHCPREQEST: клиент, не получив ответа на первый из пакетов DHCPREQEST, через 4 секунды шлет второй пакет.

Оценим задержку, вносимую в каждую DHCP транзакцию двойной проверкой – достаточно взглянуть на рисунок, чтобы убедиться, что лишняя задержка составляет целых 3 секунды, что весьма не мало.

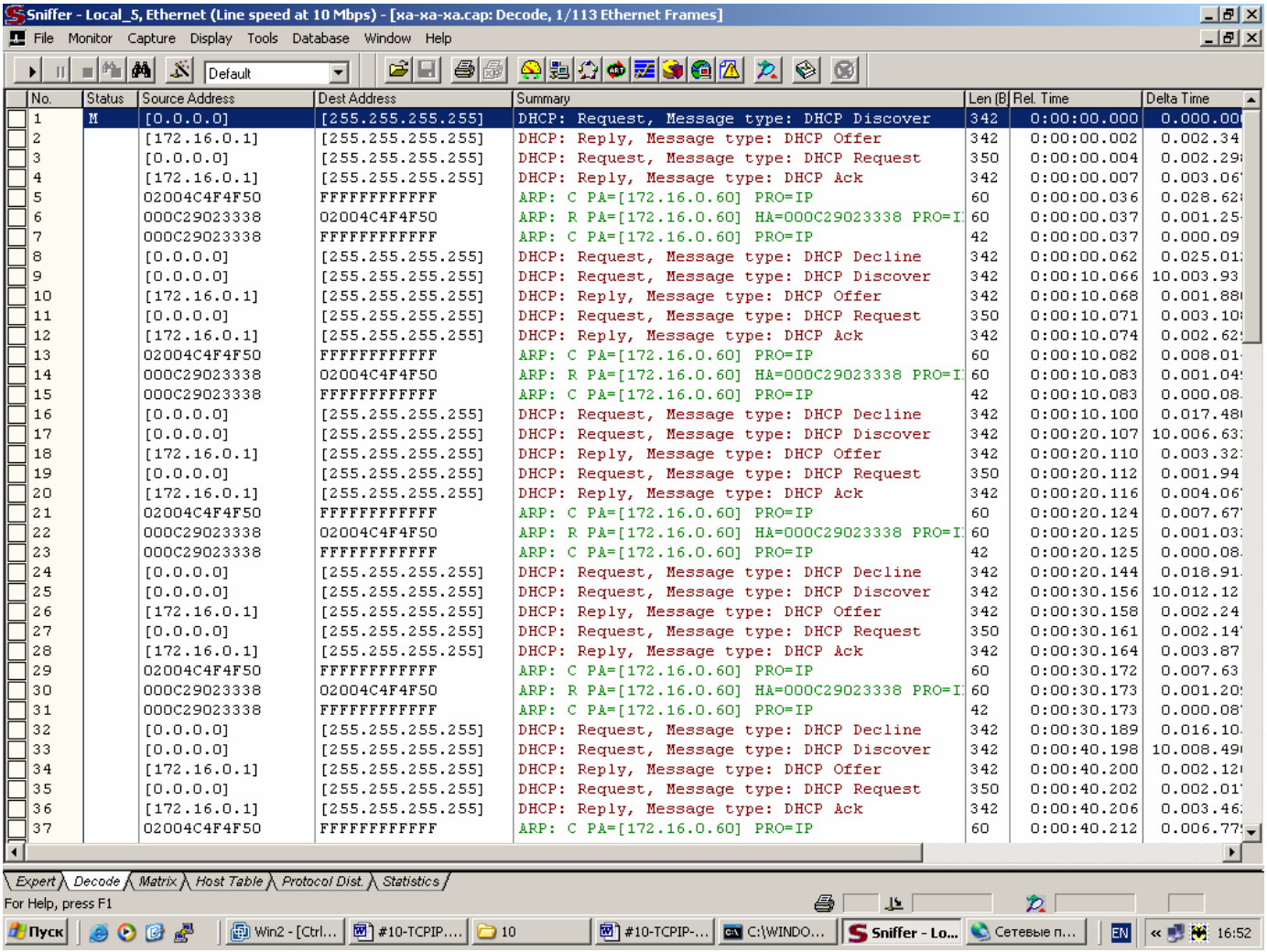
Заметим, что никого подтверждения того, что сервер использует для проверки ICMP эхо мы пока не увидели, в проводе – только ARP запросы. Но на самом деле так и должно быть и в случае, если проверка делается с помощью ICMP – перед отправкой ICMP эхо на адрес 172.16.0.100 необходимо узнать его MAC адрес, а так как ответов на ARP запросы нет, то и посылки ICMP эхо запросов нет. Вот если бы владелец адреса 172.16.0.100 нашелся бы в сети, тогда мы смогли бы увидеть ICMP эхо запросы и даже ответы ☺.

Сделаем и такой пример: пусть адрес 172.16.0.50 статически используется в сети, а клиент просит сервер предоставить ему как раз именно этот адрес. Проведем диагностику ситуации с помощью анализатора пакетов (файл pre_test_fail.cap). Видно, что клиент в первом пакете DHCPDESCOVER просит сервер предоставить ему IP адрес 172.16.0.50. Этот адрес не занят у сервера и поэтому он готов его предоставить, но сервер настроен на выполнение проверки, предупреждающей конфликты, поэтому сервер с помощью ARP запроса выясняет MAC адрес станции 172.16.0.50 и посылает ей ICMP эхо. Так как в ответ на ICMP эхо запрос приходит эхо ответ, сервер принимает решение предоставить клиенту другой адрес, а именно 172.16.0.51, для чего снова таки проверяет его дважды, и не найдя станцию с IP адресом 172.16.0.51 в сети предлагает данный адрес станции – клиенту. Как мы знаем, DHCP клиент, получая предложение, не соответствующее тому адресу, который клиент просил в пакете DHCPDESCOVER, посылает еще несколько пакетов DHCPDESCOVER в надежде, что какой то сервер все же предоставит ему желаемый адрес, но, не получая других предложений, клиент все же принимает предложение сервера с адресом 172.16.0.51, после чего происходит обычная DHCP транзакция, которая заканчивается тем, что клиент проверяет полученный адрес с помощью трех ARP запросов, что вроде как уже не нужно делать, но клиент об этом не знает.

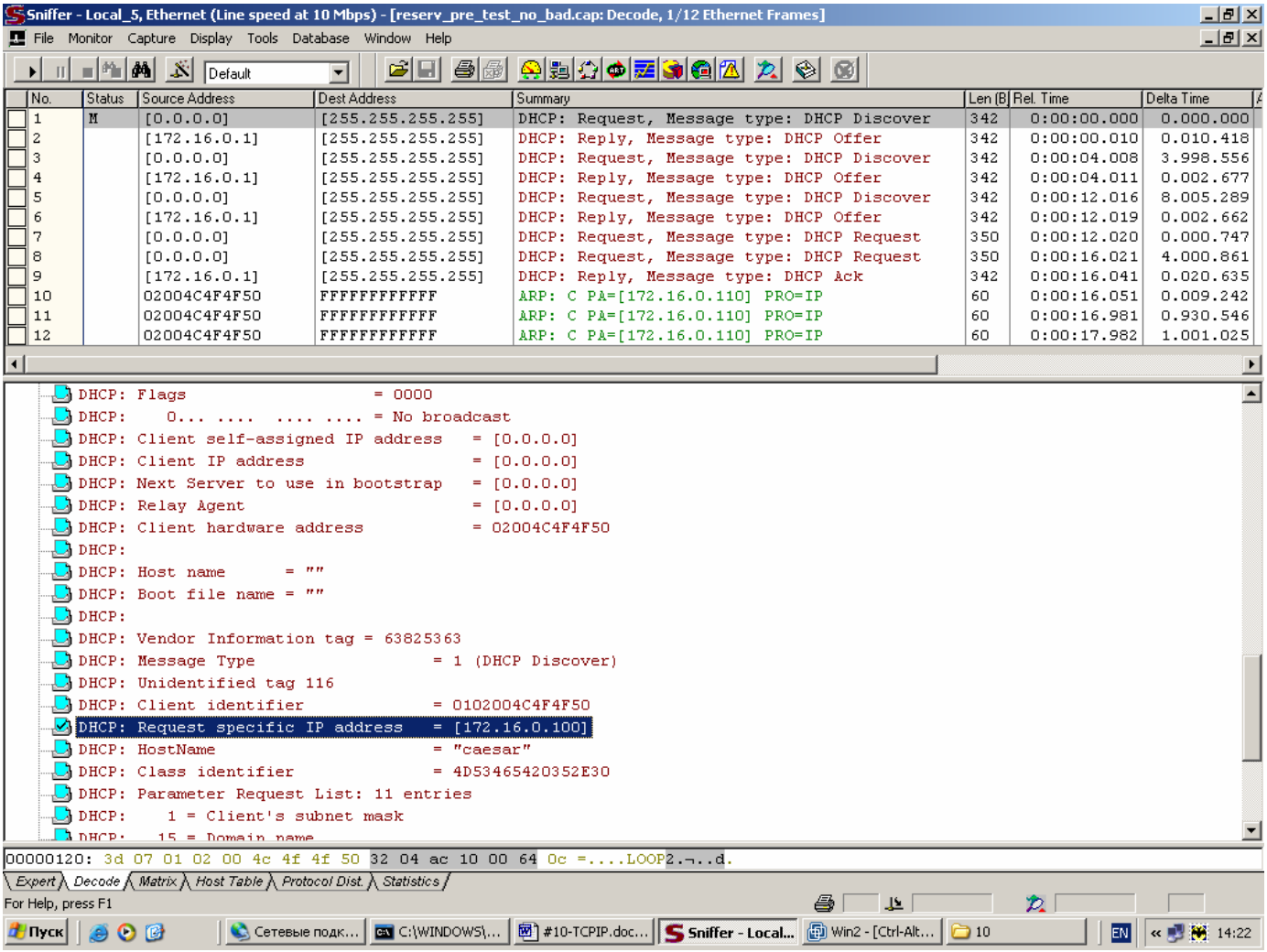


Подведем итог по использованию настройки сервера, позволяющей проверять адреса перед тем, как их предлагать: данная технология позволяет избежать использования пакетов DHCPDECLINE, но при этом каждая DHCP транзакция приобретает дополнительную задержку, что плохо, и поэтому включать поддержку данной технологии имеет смысл только в том случае, если имеются причины ожидать очень частого совпадения адресов в сети.

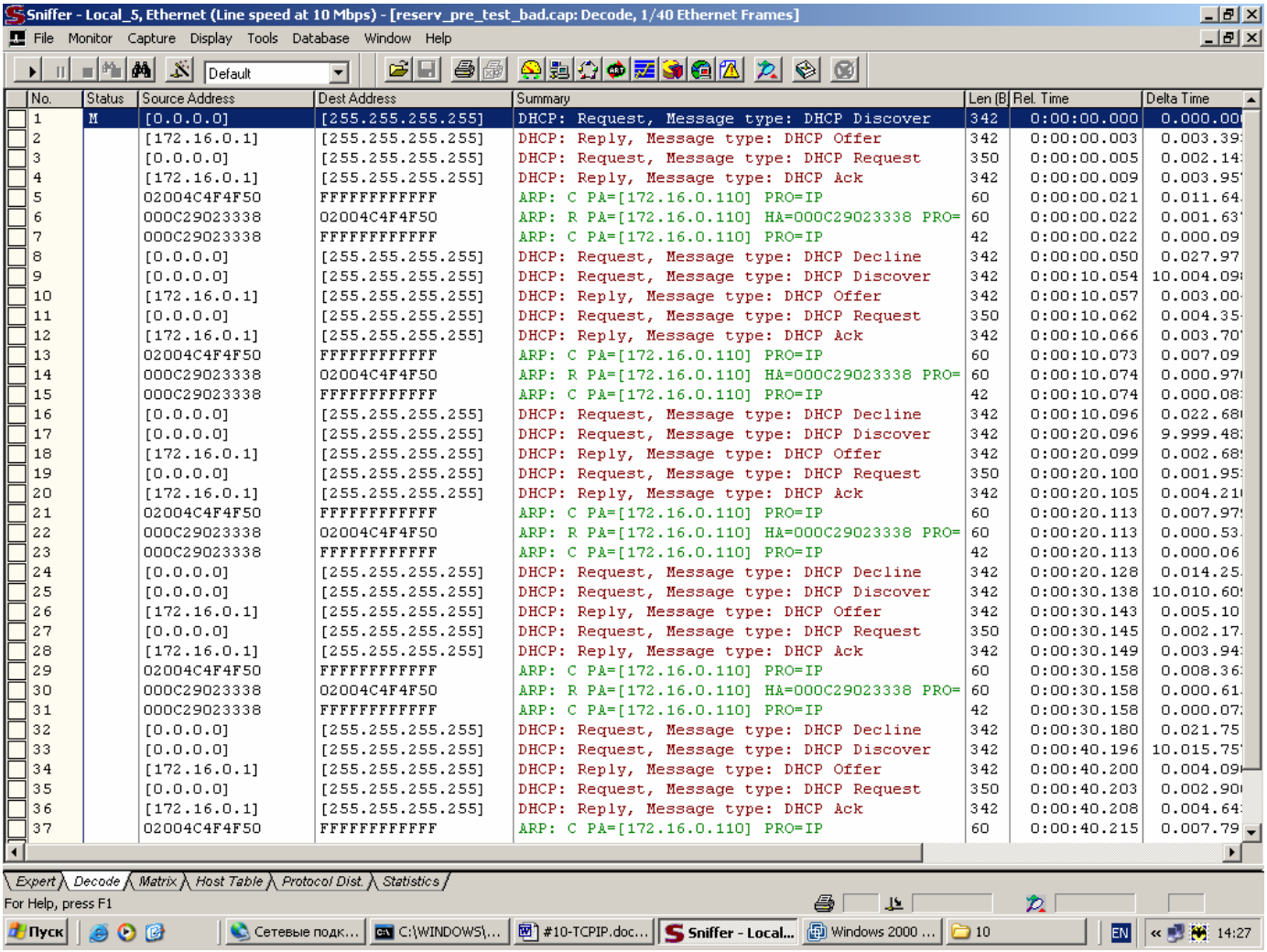
И, наконец, рассмотрим еще один весьма любопытный пример: пусть на DHCP сервере настроено резервирование для нашей станции IP адреса 172.16.0.60, но такой адрес статически используется в сети. В таком случае сервер предложит клиенту адрес 172.16.0.60, клиент проверит его с помощью ARP запроса, выяснит, что данным адресом нельзя пользоваться, пошлет серверу пакет DHCPECLINE, начнет новую транзакцию и … что предложит сервер? Посмотрим на практике, как DHCP сервер Microsoft выпутается из такого противоречия: со одной стороны он должен предлагать клиенту только адрес 172.16.0.60, с другой стороны данный адрес помечен в его скопе как «плохой». Сервер крайне плохо «справился» (файл ха-ха-ха.cap). Этот процесс будет продолжаться вечно: каждые 10 секунд клиент будет начинать новую транзакцию, получать адрес 172.16.0.60, убеждаться, что пользоваться им нельзя, посылать серверу DHCPDECLINE и … все с начала через 10 секунд. Обсудим со студентами, что такое поведение DHCP сервера крайне не удачно: во-первых клиент так и не получит никакого адреса, даже адреса автонастройки, а во-вторых создается постоянный широковещательный фон в сети: из 8 пакетов – 7 широковещательных за 10 секунд.



Может ли проверка допустимости адреса, выполняемая сервером помочь решить данную проблему? Проверим, зарезервируем некоторый IP адрес для некоторого MAC адреса, настроим сервер на выполнение проверки совпадения IP адресов в сети … и убедимся, что в таком случае сервер ВООБЩЕ не делает этой проверки, не смотря на то, что настроен ее выполнять! Заставим клиента просить себе адрес 172.16.0.100, и настроим на сервере резервирование адреса 172.16.0.110 для данного клиента, проанализируем трафик сети с помощью анализатора протоколов (файл reserve_pre_test_no_bad.cap).



Следовательно, можно ожидать, что если бы в сети существовал узел, уже использующий адрес 172.16.0.110, то ситуация с бесконечными попытками нашего клиента получить адрес повторилась бы и таким образом можно сделать вывод, что использование проверки сервером допустимости адреса перед тем как его предлагать не исключает в принципе возможности появления в линии связи пакетов DHCPDECLINE. Рассмотрим и этот пример. Проведем анализ происходящего в анализаторе протоколов, убедимся еще раз, что сервер, настроенный проверят адрес перед тем как предлагать его клиенту тем не менее не делает этого если за данным клиентом зарезервирован IP адрес и, как следствие, получаем все ту же картину – каждый 10 секунд транзакция в 8 пакетов, не приводящая к получению IP адреса.



Подводим итоги занятия: сегодня мы познакомились с основными принципами работы протокола DHCP, рассмотрели базовые вопросы конфигурирования DHCP сервера в Windows Server и на практике рассмотрели разнообразные взаимодействия между DHCP сервером и DHCP.

В следующем уроке мы продолжим изучение протокола DHCP.