Все, что ты хотел узнать о сетевых протоколах

Каждый раз, когда ты передаешь данные по сети, они как-то перетекают от твоего компьютера к другому. Как это происходит? Ты наверно скажешь, что по специальному сетевому протоколу и будешь прав. Но существует множество их разновидностей. Какой и когда используется? Зачем они нужны? Как они работают? Вот на эти вопросы я и постараюсь ответить в одной большой статье.

Модель взаимодействия открытых систем

Прежде чем разбираться с протоколами, нам необходимо узнать, что такое модель взаимодействия открытых систем (OSI - Open Systems Interconnection), которая была разработана Международной Организацией по Стандартам (ISO - International Organization for Standardization). В соответствии с этой моделью, сетевое взаимодействие делится на семь уровней:

1. Физический уровень - передача битов по физическим каналам (коаксиальный кабель, витая пара, оптоволоконный кабель). Здесь определяются характеристики физических сред и параметры электрических сигналов.

2. Канальный уровень - передача кадра данных между любыми узлами сетях типовой топологии или соседними узлами произвольной топологии. В качестве адресов на канальном уровне используются MAK-адреса.

3. Сетевой уровень - доставка пакета любому узлу в сетях произвольной топологии. На этом уровне нет никаких гарантий доставки пакета.

4. Транспортный уровень - доставка пакета любому узлу с любой топологией сети и заданным уровнем надежности доставки. На этом уровне имеются средства для установления соединения, буферизации, нумерации и упорядочивания пакетов.

5. Сеансовый уровень - управление диалогом между узлами. Обеспечена возможность фиксации активной на данный момент стороны.

6. Уровень представления - здесь возможно преобразование данных (шифрация, компрессия).

7. Прикладной уровень - набор сетевых сервисов (FTP, E-mail и др.) для пользователя и приложения.

Если ты внимательно прочитал все уровни, то наверно заметил, что первые три уровня обеспечиваются оборудованием, таким как сетевые карты, маршрутизаторы, концентраторами, мостами и др. Последние три обеспечиваются операционной системой или приложением. Четвертый уровень является промежуточным.

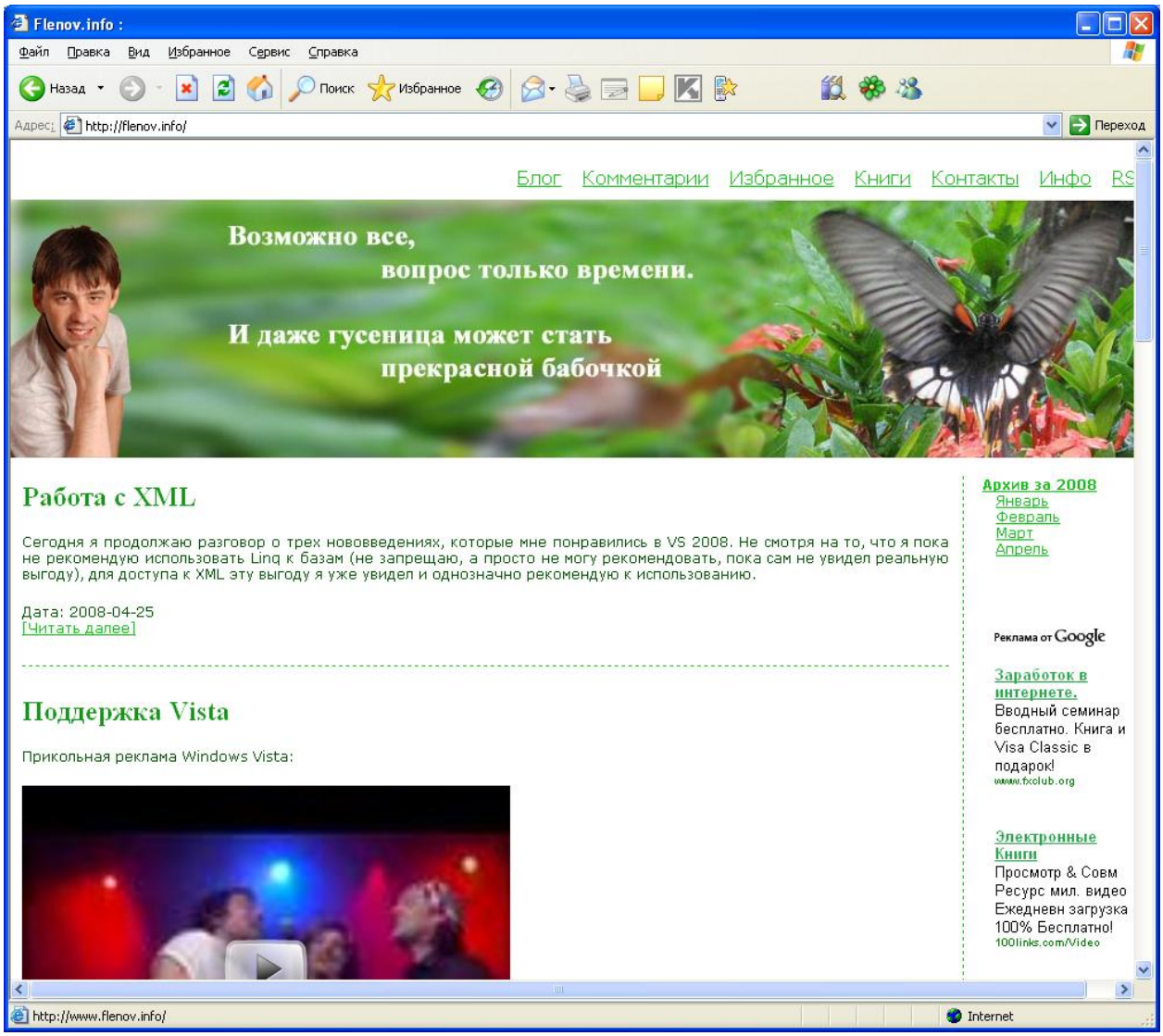
Связь модели OSI с сетевыми протоколами

Как работает протокол по этой модели? Все начинается с прикладного уровня. Пакет попадает на этот уровень и к нему добавляется заголовок. После этого, прикладной уровень отправляет этот пакет на следующий уровень (уровень представления). Здесь ему также добавляется свой собственный заголовок, и пакет отправляется дальше. Так до физического уровня, который занимается непосредственно передачей данных и отправляет пакет в сеть.

Другая машина, получив пакет, начинает обратный отсчет. Пакет с физического уровня попадает на канальный. Канальный уровень убирает свой заголовок и поднимает пакет выше (на уровень сети). Уровень сети убирает свой заголовок и поднимает пакет выше. Так пакет подымается до уровня приложения, где остается чистый пакет без служебной инфы которая была прикреплена на исходной машине перед отправкой пакета.

Передача данных не обязательно должна начинаться с седьмого уровня. Если используемый протокол работает на 4-м уровне, то процесс передачи начнется с него, и пакет будет подниматься вверх до физического уровня.

Чем ниже находиться протокол (ближе к прикладному уровню), тем больше у него возможностей и больше накладных расходов при передаче данных. Рассматриваемые сегодня протоколы будут находиться на разных уровнях, поэтому будут иметь разные возможности.



Модель OSI по методу M$

MS как всегда пошла своим путем и реализовала модель OSI в TCP/IP по-своему. Я понимаю, что модель OSI справочная и предназначена только в качестве рекомендации, но нельзя же было так ее уродовать.

У MS вместо семи уровней есть только четыре. Но это не значит, что остальные уровни позабыты и позаброшены, просто один уровень MS может выполнять все, что в OSI делает три уровня. Например, уровень приложения у MS выполняет все, что делает уровень приложения, уровень представления и уровень сеанса вместе взятые.

Рисунок 1. Модель OSI и вариант от MS

На рисунке 1 я графически сопоставил модель MS TCP и справочную модель OSI. Слева указаны названия уровней по методу MS, а справа уровни OSI. В центре показаны протоколы. Я постарался разместить их именно на том уровне, на котором они работают, в последствии нам это пригодиться.

Протокол IP

Некоторые считают, что протокол TCP/IP это одно целое. На самом деле это два разных протокола, которые могут работать совместно.

Если посмотреть на рисунок сетевой модели, то можно увидеть, что протокол IP находиться на сетевом уровне. Из этого можно сделать вывод, что IP выполняет сетевые функции - доставка пакета любому узлу в сетях произвольной топологии.

Протокол IP при передаче данных не устанавливает виртуального соединения и использует датаграммы для отправки данных от одного компьютера к другому. Это значит, что по протоколу IP пакеты просто отправляются в сеть без ожидания подтверждения о получении данных (ACK Acknowledgment), а значит без гарантии целостности данных. Все необходимые действия по подтверждению и обеспечении целостности данных должны обеспечивать протоколы, работающие на более высоком уровне.

Каждый IP пакет содержит адреса отправителя и получателя, идентификатор протокола, TTL (время жизни пакета) и контрольную сумму для проверки целостности пакета. Как видишь, здесь есть контрольная сумма, которая все же позволяет узнать целостность пакета. Но об этом узнает только получатель. Когда комп - получатель принял пакет, то он проверяет контрольную сумму только для себя. Если пакет верный, то он обрабатывается, иначе просто отбрасывается. А компьютер отправитель пакета не сможет узнать об ошибке в пакете и не сможет заново отправить пакет. Именно поэтому соединение по протоколу IP нельзя считать надежным.

Сопоставление адреса ARP и RARP

Протокол ARP предназначен для определения аппаратного (MAC) адреса компьютера в сети по его IP адресу. Прежде чем данные смогут быть отправлены на какой-нибудь комп, отправитель должен знать аппаратный адрес получателя. Именно для этого и предназначен ARP.

Когда комп посылает ARP запрос на поиск аппаратного адреса, то протокол сначала ищет этот адрес в локальном кэше. Если уже были обращения по данному IP адресу, то информация о MAC адресе должна сохраниться в кэше. Если ничего не найдено, то в сеть посылается широковещательный запрос, который получат все компы сети. Все они получат этот пакет и проверят, если искомый IP принадлежит им, то они ответят на запрос, указав свой MAC адрес.

Протокол RARP делает обратное - определяет IP адрес по известному MAC адресу. Процесс поиска адресов абсолютно такой же.

Быстрый UDP

На транспортном уровне у нас два протокола - UDP и TCP. Оба они работают поверх IP. Это значит, что когда пакет TCP или UDP опускается на уровень ниже для отправки в сеть, он попадает на уровень сети прямо в лапы протокола IP. Здесь пакету добавляется сетевой адрес, TTL и др. атрибуты протокола IP. После этого пакет идет дальше вниз для физической отправки в сеть. Голый пакет TCP не может быть отправлен в сеть, потому что он не имеет информации о получателе, эта инфа добавляется к пакету с IP заголовком на уровне сети.

Давай теперь рассмотрим каждый протокол в отдельности и начнем мы с UDP. Как и IP, протокол UDP для передачи данных не устанавливает соединения с сервером. Данные просто выплевываются в сеть и протокол даже не заботиться о доставке пакета. Если данные на пути к серверу испортятся или вообще не дойдет, то отправляющая сторона об этом не узнает. Так что по этому протоколу, как и по голому IP, не желательно передавать очень важные данные.

Благодаря тому, что протокол UDP не устанавливает соединения, он работает очень быстро (в несколько раз быстрее TCP, о котором чуть ниже). Из-за высокой скорости его очень удобно использовать там, где данные нужно передавать быстро, и не нужно заботиться об их целостности. Такими примерами может бить радиостанции в инете. Звуковые данные просто выплевываются в глобальную сеть и если слушатель не получит одного пакета, то максимум, что он услышит - небольшое заикание в месте потери. Но если учесть, что сетевые пакеты имеют небольшой размер, то это заикание будет практически незаметно.

Большая скорость - большие проблемы с безопасностью. Так как нет соединения между сервером и клиентом, то нет никакой гарантии в достоверности данных. Протокол UDP больше подвержен спуфингу (подмена адреса отправителя), поэтому построение на нем защищенных сетей затруднено.

Итак, UDP очень быстр, но его можно использовать только там, где данные не имеют высокой ценности (возможна потеря отдельных пакетов) и не имеют секретности (UDP больше подвержен взлому).

Медленный, но надежный TCP

Как я уже сказал, протокол TCP лежит на одном уровне с UDP и работает поверх IP (для отправки данных используется IP). Именно поэтому протоколы TCP и IP часто объединяют одним названием TCP/IP, потому что TCP неразрывно связан с IP.

В отличие от UDP протокол TCP устраняет недостатки своего транспорта (IP). В этом протоколе заложены средства установления связи между приемником и передатчиком, обеспечение целостности данных и гарантии их доставки.

Когда данные отправляются в сеть по TCP, то на отправляющей стороне включается таймер. Если в течение определенного времени приемник не подтвердит получение данных, то будет предпринята еще одна попытки отправки данных. Если приемник получит испорченные данные, то он сообщит об этом источник и попросит снова отправить испорченные пакеты. Благодаря этому обеспечивается гарантированная доставка данных.

Когда нужно отправить сразу большую порцию данных, не вмещающихся в один пакет, то они разбиваются на несколько TCP пакетов. Пакеты отправляются порциями по несколько штук (зависит от настроек стека). Когда сервер получает порцию пакетов, то

он восстанавливает их очередность и собирает данные вместе (даже если пакеты прибыли не в том порядке, в котором они отправлялись).

Из-за лишних накладных расходов на установку соединения, подтверждения доставки и повторную пересылку испорченных данных протокол TCP намного медленней UDP. Зато TCP можно использовать там, где нужна гарантия доставки и большая надежность. Хотя надежность нельзя назвать сильной (нет шифрования, и все же есть возможность взлома), но она достаточная и намного больше чем у UDP. По крайней мере тут спуфинг не может быть реализован так просто, как у UDP и в этом ты убедишься, когда прочтешь про процесс установки соединения.

Опасные связи TCP

Давай посмотрим, как протокол TCP обеспечивает надежность соединения. Все начинается еще на этапе попытки соединения двух компов:

1 шаг. Клиент, который хочет соединиться с серваком, отправляет SYN запрос на сервер, указывая номер порта, к которому он хочет приконнектиться, и специальное число (чаще всего случайное).

2 шаг. Сервер отвечает своим сегментом SYN, содержащим специальное число сервера. Он также подтверждает приход SYN пакета от клиента с использованием ACK ответа, где специальное число отправленное клиентом увеличено на 1.

3 шаг. Клиент должен подтвердить приход SYN от сервера с использованием ACK специальное число сервера плюс 1.

Получается, что при соединении клиента с сервером они обмениваются спец числами. Эти числа и используются в дальнейшем для обеспечения целостности и защищенности связи. Если кто-то другой захочет вклиниться в установленную связь (с помощью спуфинга), то ему надо будет подделать эти числа. Но так как они большие и выбираются случайным образом, то такая задача достаточно сложная, хотя Кевин Митник в свое время смог решить ее. Но это уже другая история и не будем уходить далеко в сторону.

Стоит еще отметить, что приход любого пакета подтверждается ACK ответом, что обеспечивает гарантию доставки данных.

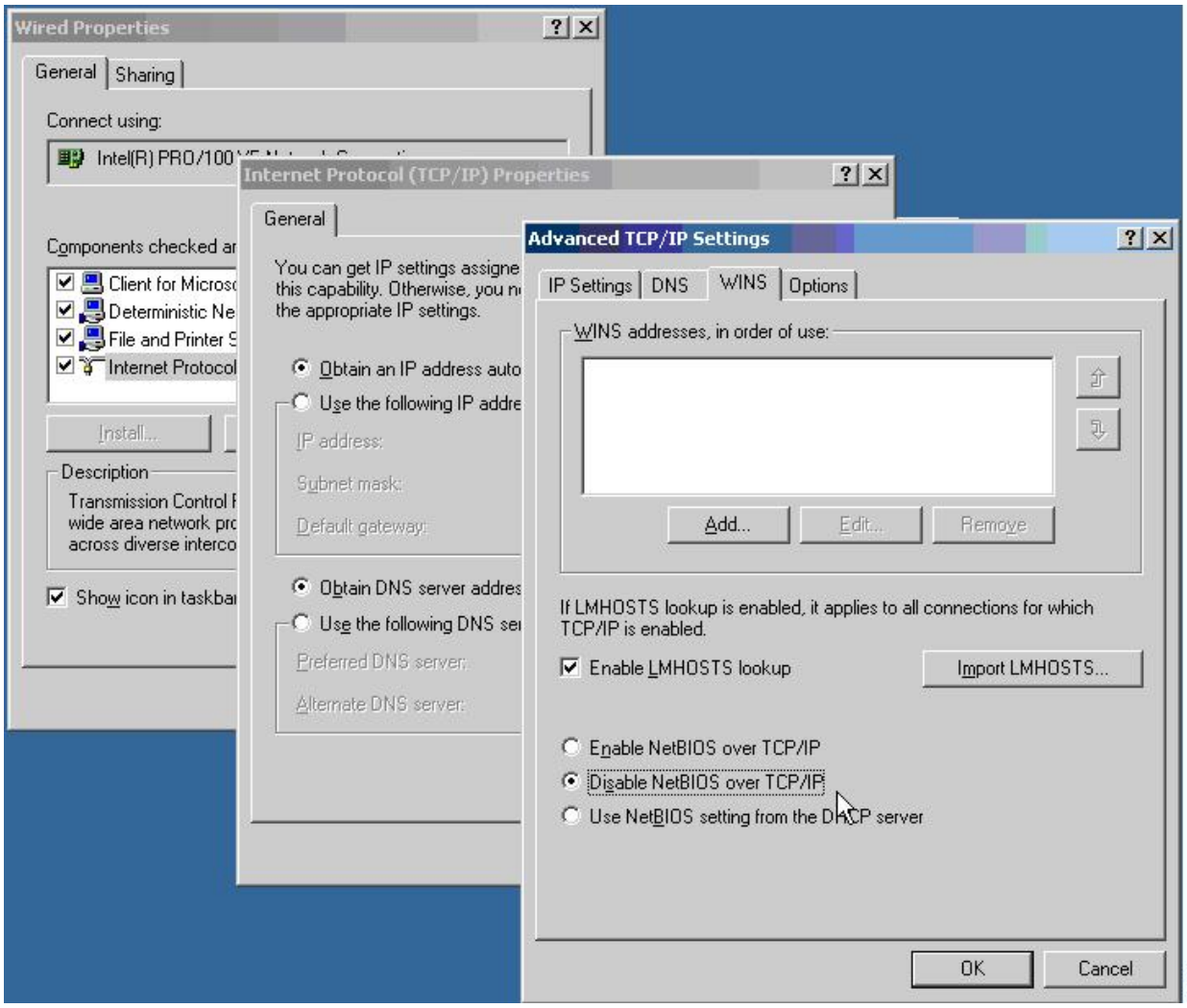
Загадочный NetBIOS

NetBIOS (Network Basic Input Output System - базовая система сетевого ввода вывода) - это стандартный интерфейс прикладного программирования. А проще говоря, это всего лишь набор API функций для работы с сетью (хотя весь NetBIOS состоит только из

одной функции, но зато какой...). NetBIOS был разработан в 1983 году компанией Sytek Corporation специально для IBM.

Система NetBIOS определяет только программную часть передачи данных, т.е. как должна работать программа для передачи данных посети. А вот как будут физически передаваться данные в этом документе не говориться ни слова, да и в реализации отсутствует что-нибудь подобное.

Если посмотреть на рисунок 1, то можно увидеть, что NetBIOS находиться в самом верху схемы (большой квадрат вверху справа). Этот квадрат расположен на уровнях сеанса, представления и приложения. Такое расположение квадрата - лишнее подтверждение моих слов.

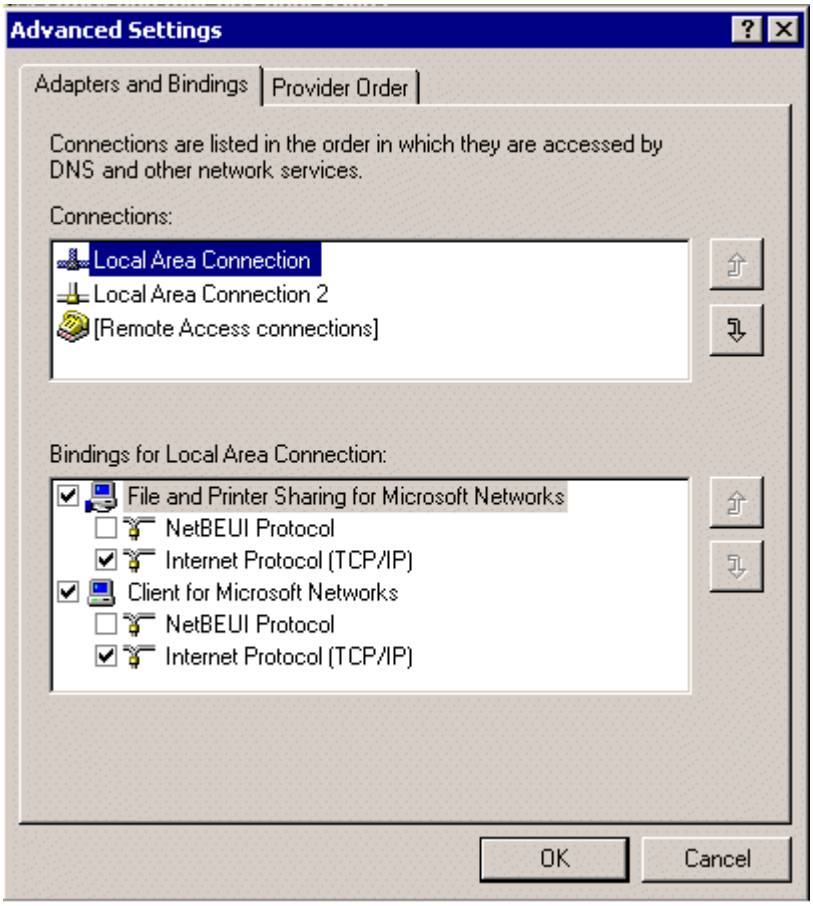


NetBIOS только формирует данные для передачи, а физически передаваться они могут только с помощью другого протокола, например TCP/IP, IPX/SPX и т. д. Это значит, что NetBIOS является независимым от транспорта. Если другие варианты протоколов верхнего уровня (только формирующих пакеты, но не передающих) привязаны к определенному транспортному протоколу, который должен передавать сформированные данные, то пакеты NetBIOS может передавать любой другой протокол. Прочувствовал силу? Представь, что ты написал сетевую прогу работающую через NetBIOS. А теперь запомни, что она будет прекрасно работать как в unix/windows сетях через TCP, так и в Novell сетях через IPX.

С другой стороны, для того, чтобы два компьютера смогли соединиться друг с другом по NetBIOS, необходимо чтобы на обоих компах стоял хотя бы один общий транспортный протокол. Если один комп будет посылать NetBIOS пакеты поверх TCP, а другой поверх IPX, то эти компьютеры друг друга не поймут. Транспорт должен быть одинаковый.

Стоит сразу же отметить, что не все варианты транспортных протоколов по умолчанию могут передавать по сети NetBIOS пакеты. Например, IPX/SPX сам по себе этого не умеет. Чтобы его обучить, нужно иметь "NWLink IPX/SPX/NetBIOS Compatible Transport Protocol".

Так как NetBIOS чаще всего использует в качестве транспорта протокол TCP, который работает с установкой виртуального соединения между клиентом и сервером, то по этому протоколу можно передавать достаточно важные данные. Целостность и надежность передачи будет осуществлять TCP/IP, а NetBIOS дает только удобную среду для работы с пакетами и кодинга сетевого софта. Так что если тебе нужно отправить в сеть какие-то файлы, то можно смело положиться на NetBIOS.



NetBEUI

В 1985 году уже сама IBM сделала попытку превратить NetBIOS в полноценный протокол, который умеет не только формировать данные для передачи, но и физически передавать их по сети. Для этого был разработан NetBEUI - (NetBIOS Extended User

Interface) и был предназначен именно для описания физической части передачи данных протокола NetBIOS.

Сразу хочу отметить, что NetBEUI является не маршрутизируемым протоколом и первый же маршрутизер будет отбиваться от таких пакетов как теннисистка от мячиков :). Это значит, что если между двумя компами стоит маршрутизатор и нет другого пути для связи, то им не удастся установить соединение через NetBEUI.

Сокеты Windows

Сокеты (Sockets) - всего лишь программный интерфейс, который облегчает взаимодействие между различными приложениями. Современные сокеты родились из программного сетевого интерфейса реализованного в ОС BSD Unix. Тогда этот интерфейс создавался для облегчения работы с TCP/IP на верхнем уровне.

С помощью сокетов легко реализовать большинство известных тебе протоколов, которые ты используешь каждый день, когда выходишь в инет. Достаточно только назвать HTTP, FTP, POP3, SMTP и в том же духе. Все они используют для отправки своих данных или TCP, или UDP и легко программируются с помощью библиотеки sockets/winsock.

Протокол IPX/SPX

Осталось только рассказать еще о нескольких протоколах, которые встречаются в повседневной жизни чуть реже, но зато они не менее полезны. Первый на очереди это IPX/SPX.

Протокол IPX (Internetwork Packet Exchange) сейчас используется наверно только в сетях фирмы Novell. В наших любимых окошках есть специальная служба "Клиент для сетей Novell" с помощью которой ты сможешь работать в таких сетях. IPX работает на подобие IP и UDP - без установления связи, а значит без гарантии доставки и всеми последующими достоинствами и недостатками.

SPX (Sequence Packet Exchange) - это транспорт для IPX, который работает с установлением связи и обеспечивает целостность данных. Так что если тебе понадобиться надежность при использовании IPX, то используй связку IPX/SPX или IPX/SPX11.

Сейчас IPX уже теряет свою популярность, а помнятся времена DOS, когда все сетевые игрулины работали через этот протокол.

Итого

Как видишь, в инете протоколов целое море, но большинство из них взаимосвязано, как например, HTTP-TCP-IP. Одни протоколы могут быть предназначены для одной цели, но абсолютно непригодны для другой, потому что создать что-то идеальное невозможно. У каждого будут свои достоинства и недостатки.

И все же, модель OSI принятая еще на заре появления инета не утратила своей актуальности до сих пор. По ней работает все и вся. Главное ее достоинство - скрывать сложность сетевого общения между компьютерами, счем старушка OSI справляется без особых проблем.

На этом считаю наш урок законченным, так что тушите свет и кидай гранату. Скоро увидимся!!!

Словарь

Датаграмма - синоним пакета данных. Чаще всего так называют пакеты протоколов IP или UDP, потому что они без установления соединения между приемником и передатчиком.

Хост, узел - любое устройство (компьютер, маршрутизатор и т.д.) подключенное к сети.

Loopback address - IP адрес, который начинается со 127 (например, 127.0.0.1). Такой адрес нельзя использовать в качестве адресов сетевых узлов, и всегда указывает только на локальный компьютер. Такие адреса нужны для тестирования прог без отправки данных в сеть.

MAC адрес - это физический адрес, который прописывается в любое сетевое устройство его производителем. Он состоит из 6 байт и абсолютно уникален. Никакие два сетевых устройства не могут иметь одинаковый физический адрес.

Маршрутизатор - устройство или программа, которые помогают двум разным сетям взаимодействовать друг с другом.

Протокол - правила, по которым информация передается в сеть.