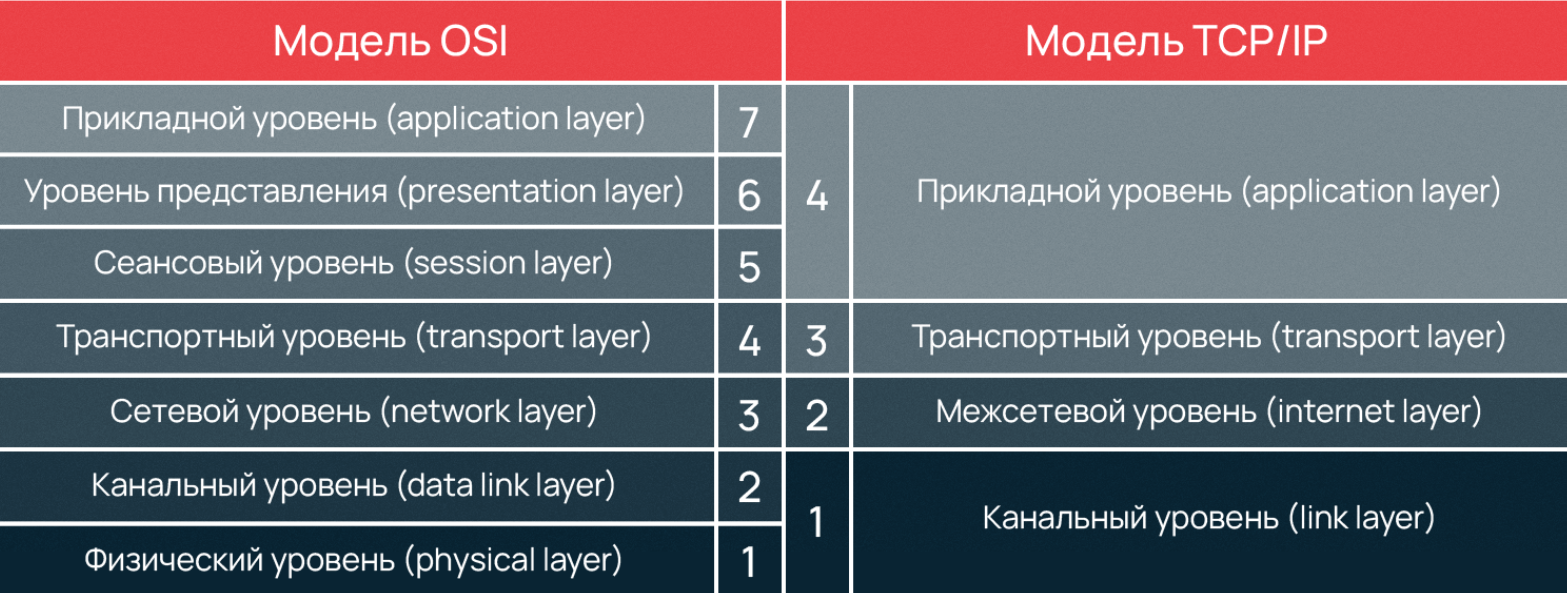
**1) Модели взаимодействия сетевых приложений (ISO/OSI, TCP/IP). Архитектура распределенного приложения (клиент/сервер). Основные технологии создания распределенных программных систем. Спецификация NDIS.**

**Шиман обращал внимание:** понятие распределенного приложения (в частности архитектура клиент\сервер); спецификации NDIS, POSIX и для чего нужны; ISO/OSI и TCP/IP уровни, для чего каждый предназначен, что на нём происходит и какие протоколы попадаются на уровне, где проходит граница программной и аппаратной, границы уровней.

Функции, обеспечивающие взаимодействие открытых систем в модели ISO/OSI(*актуальная ISO-7498)* распределены по семи уровням. Задача каждого уровня – предоставление услуг вышестоящему уровню таким образом, чтобы детали реализации этих услуг были скрыты. Наборы правил и соглашений, описывающих процедуры взаимодействия каждого уровня модели с соседними уровнями называются **протоколами**.

****

**Физический уровень (бит) –** определяет свойства среды передачи данных (коаксиальный кабель, витая пара, оптоволоконный канал и т.п.) и способы ее соединения с сетевыми адаптерами: технические характеристики кабелей (сопротивление, емкость, изоляция и т.д.), перечень допустимых разъемов, способы обработки сигнала и т.п.

**Канальный уровень (кадр) –** делится на два подуровня: подуровень управления доступом к среде передачи данных (MAC – Media Access Control) и подуровень управления логическим каналом (LLC – Logical Link Control). Основное назначение – подготовить блок данных (кадр) для следующего сетевого уровня.

**MAC** – определяет методы совместного использования \*сетевыми адаптерами\* среды передачи данных.

**LLC** – определяет понятие канала между двумя сетевыми адаптерами, а также способы обнаружения и исправления ошибок передачи данных. С этого уровня и выше протоколы никак не зависят от среды передачи данных.

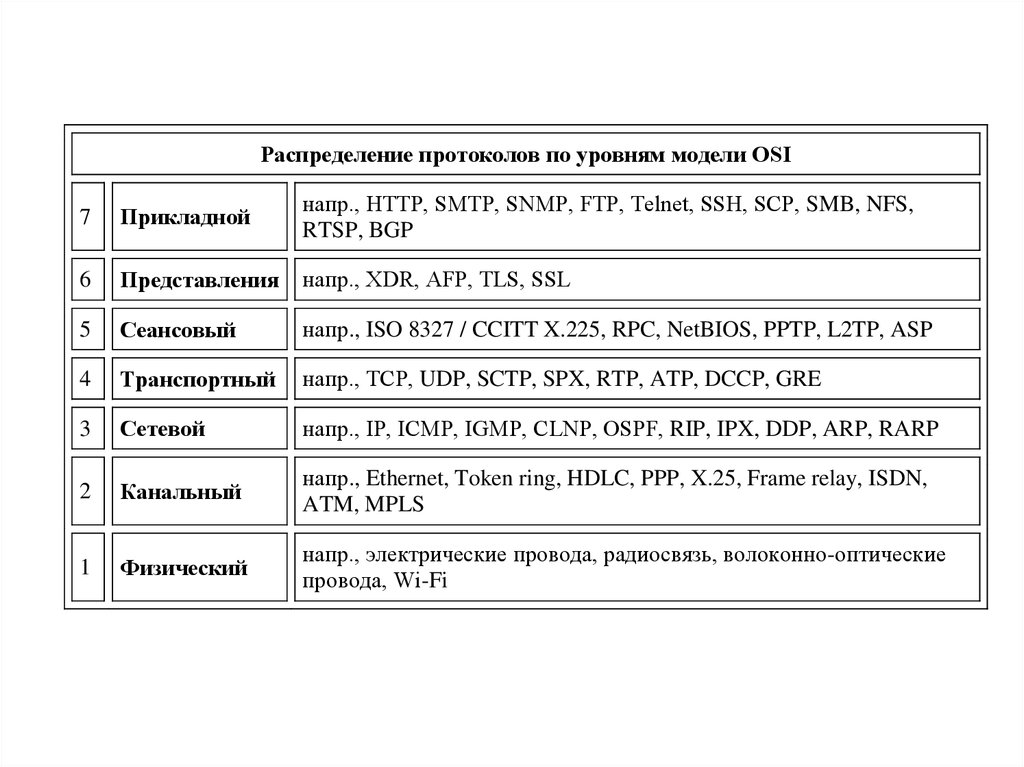
**Сетевой уровень (пакет) –** определяет: 1) методы адресации и маршрутизации компьютеров в сети; 2) единый метод адресации для всех компьютеров в сети не зависимо от способа передачи данных; 3) способы соединения компьютерных сетей. Результатом процедур сетевого уровня является пакет, который обрабатывается процедурами транспортного уровня.

**Транспортный уровень (дейтаграмма) –** подготовка и доставка пакетов данных между конечными точками без ошибок и в правильной последовательности. Формируются файлы для сеансового уровня из пакетов, полученных от сетевого уровня.

**Сеансовый уровень (данные) –** определяет способы установки и разрыва соединений (сеанса) двух приложений, работающих в сети.

**Представительский уровень (данные) –** определяется формат данных, используемых приложениями. Процедуры этого уровня описывают способы шифрования, сжатия и преобразования наборов символов данных.

**Прикладной уровень (данные) –** определение способа взаимодействия пользователей с системой (определить интерфейс).



Наиболее популярной архитектурой для распределенного программного приложения является **архитектура клиент-сервер** – все процессы можно разбить на две группы. Одна группа процессов называется серверами, другая – клиентами. Обмен данными осуществляется только между процессами-клиентами и процессами-серверами. Основное отличие между ними в том, что инициатором для обмена данными всегда является процесс-клиент. Для этого клиент должен обладать информацией о месте нахождения сервера или иметь механизмы для его обнаружения.

По методу обслуживания серверы подразделяются на итеративные (обслуживают запросы клиентов поочередно) и параллельные (обслуживают несколько клиентов одновременно).

**NDIS (Network Driver Interface Specification) –**драйвер для сопряжения драйверов сетевых адаптеров с операционной системой.

**POSIX** — набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой, библиотеку языка C и набор приложений и их интерфейсов.

**2) Стек протоколов TCP/IP. Публичные и частные пространства адресов, типы портов. Параметры настройки TCP/IP.**

**TCP (Transmission Control Protocol)** – обеспечивает надежную доставку данных в сети.

**IP (Internet Protocol) –** организует маршрутизацию сетевых передач от отправителя к получателю и отвечает за адресацию сетей и компьютеров.

TCP/IP была разработана задолго до ISO/OSI и является более простой. Уровни моделей похожи, но не идентичны.

Все уровни TCP/IP в общем и целом соответствуют уровням OSI/ISO как показано на рисунке выше. Но, например, некоторые функции сеансового уровня ISO/OSI берет на себя транспортный уровень TCP/IP.

**Протоколы уровня доступа к сети (канальный):**

1) LAN (Local-Area Network) – протокол для создания локальных сетей;

2) WAN (Wide-Area Network) – протокол для соединения с глобальными сетями;

3) Ethernet – протокол для передачи данных по локальной сети. Применяет метод доступа CSMA/CD, использует 48-битную адресацию и обеспечивает передачу данных до 1 гигабайта в секунду. Максимальная длина кадра составляет 1518 байт, при этом сами данные могут занимать от 46 до 1500 байт;

4) SLIP (Serial Line IP) – протокол для последовательного канала. Раньше использовался для подключения домашних компьютеров к интернету через последовательный порт RS-232. Имеет ряд недостатков: хост с одной стороны должен знать IP-адрес другого, т.к. SLIP не дает возможности сообщить свой IP-адрес; если линия задействована SLIP, то она не может быть использована никаким другим протоколом; SLIP не добавляет контрольной информации к пакету передаваемой информации – весь контроль возложен на протоколы более высокого уровня. Ряд недостатков были исправлены в новой версии протокола именуемой SCLIP (Compressed SLIP);

5) PPP (Point-to-Point Protocol) – протокол двухточечного соединения (Двухточечные соединения обычно используются для соединения двух систем в глобальной сети). PPP поддерживает многоканальные реализации: можно сгруппировать несколько каналов с одинаковой пропускной способностью между отправителем и получателем. Кроме того, PPP обеспечивает циклический контроль для каждого кадра, динамическое определение адресов, управление каналом. Наиболее широко используемый протокол для последовательного канала, обеспечивающий соединение компьютера с сетью Internet. Вытеснил SLIP.

**Протоколы межсетевого уровня:**

1) IP – доставка дейтаграмм. Не надежный и не поддерживающий соединение;

2) ICMP (Internet Control Message Protocol) – транспортировка информации о сетевой деятельности и маршрутизации. IPCM сообщения представляют собой специально отформатированные IP-дейтаграммы, которым соответствуют определенные типы (15 типов) и коды сообщений. С помощью этого протокола осуществляется деятельность утилит достижимости (ping, traceroute); регулируется частота отправки IP-дейтаграмм, оптимизируется MTU для маршрута передачи IP-дейтаграмм; доставляется хостам, маршрутизаторам и шлюзам всевозможная служебная информация; осуществляется поиск и переадресация маршрутизаторов; оптимизируются маршруты; диагностируются ошибки и оповещаются узлы IP-сети;

3) ARP (Address Resolution Protocol) – динамическая проекция IP-адресов в соответствующие MAC-адреса аппаратных средств. Каждый хост кэширует специальную ARP-таблицу. Время существования записи в таблице обычно составляет 10-20 минут с момента ее создания и может быть изменено с помощью параметров реестра;

4) RARP (Reverse ARP) – получение IP-адреса по MAC-адресу;

5) IPv6 (длина 128 бит) – новая версия протокола IPv4 призванная решить недостатки IPv4.

**Протоколы транспортного уровня:**

1) TCP;

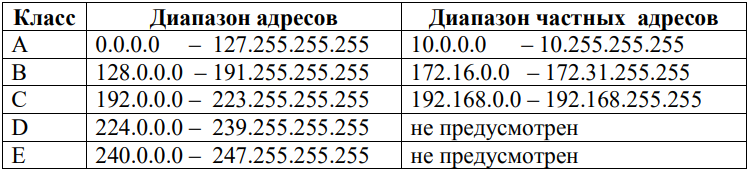
2) UDP (User Datagram Protocol) – протокол передачи дейтаграмм пользователя.

**Протоколы прикладного уровня:**

1) HTTP (HyperText Transfer Protocol) – протокол для передачи произвольных данных. Использует порт 80 по умолчанию.

2) FTP (File Transfer Protocol) – протокол передачи файлов по сети;

3) SMTP (Simple Mail Transport Protocol) – протокол для электронной почты.





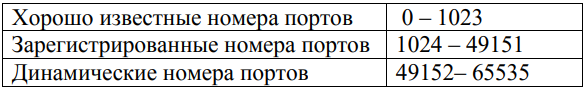
Принадлежность к классу определяется старшими битами адреса. Класс D – предназначен для использования групповых адресов, позволяющих отправлять сообщения группе хостов; E – исключительно для экспериментального применения.

**Публичные IP-адреса** – используются для выхода в интернет.

**Частные IP-адреса –** используются для неконтролируемого использования в организация. Они не маршрутизируются в интернете.

Альтернативой классов модели является **бесклассовая междоменная маршрутизация – CIDR (Classes Inter-Domain Routing).** CIDR позволяет произвольным образом назначить границу адреса сети (биты установлены) и адреса хоста (биты сброшены) IP-адреса с помощью маски подсети. Вычисление адреса сети происходит с помощью конъюнкции между IP-адресом и маской подсети.

Процесс, получающий или отправляющий данные с помощью транспортного уровня, идентифицируется номером, который называется **номер порта**.



Хорошо известные номера портов присваиваются базовым системным службам, имеющие системные привилегии.

Зарегистрированные номера портов присваиваются промышленным приложениям и процессам.

Динамические номера портов (эфемерные порты) выделяются, как правило, прикладным процессам специализированной службой операционной системы.

**Параметры настройки TCP/IP:**

1) Использовать фиксированный IP-адрес, при этом задать маску подсети и адрес основного шлюза или применить протокол DHCP;

2) Использовать фиксированный адрес DNS-сервера или получить автоматически.

**3) Основные характеристики протоколов Ethernet, SLIP, PPP, IPv4, IPv6, ICMP, ARP, RARP, TCP, UDP. Понятия: надежный и ненадежный протоколы, протоколы с установкой соединения или без установки соединения, протоколы, ориентированные на поток или на сообщения.**

**Шиман обращал внимание:** Главным параметром у протоколов на физическом уровне является МДУ?(максимальная длина кадров).Все параметры протоколов знать не обязательно, но только самого распространенного протокола Ethernet 1500 байт вот эту информацию в голове нужно держать, соответственно знать про IPv4,IPv6 структура пакета структуры в зависимости от реализации могут быть разные но в классике жанра 64 ip адрес и 64 мак адрес ну и мак адрес 48 поэтому 16 получается зарезервировано на будущее. ICMP нужно знать для того, чтобы знать о загруженности сети, ARP/RARP - сопоставление символьного имени и IP адреса, на основании этих двух работают DNS и службы DNS, TCP\UDP - сказать 6 основных свойств разницы их, но их больше, и потом если их принимать как постулаты закона, на все каверзные вопросы преподавателя можно отвечать, т.к на основании закона так-то так-то работает так-то. Вспомним что такое сокеты, как создаются сокеты, какие параметры нужно прописать, чтобы эти сокеты были ориентированы на сообщения или поток.

**Протоколы уровня доступа к сети (канальный):**

1) Ethernet – протокол для передачи данных по локальной сети. Применяет метод доступа CSMA/CD, использует 48-битную адресацию и обеспечивает передачу данных до 1 гигабайта в секунду. Максимальная длина кадра составляет 1518 байт, при этом сами данные могут занимать от 46 до 1500 байт;

2) SLIP (Serial Line IP) – протокол для последовательного канала. Раньше использовался для подключения домашних компьютеров к интернету через последовательный порт RS-232. Имеет ряд недостатков: хост с одной стороны должен знать IP-адрес другого, т.к. SLIP не дает возможности сообщить свой IP-адрес; если линия задействована SLIP, то она не может быть использована никаким другим протоколом; SLIP не добавляет контрольной информации к пакету передаваемой информации – весь контроль возложен на протоколы более высокого уровня. Ряд недостатков были исправлены в новой версии протокола именуемой SCLIP (Compressed SLIP);

3) PPP (Point-to-Point Protocol) – протокол двухточечного соединения (Двухточечные соединения обычно используются для соединения двух систем в глобальной сети). PPP поддерживает многоканальные реализации: можно сгруппировать несколько каналов с одинаковой пропускной способностью между отправителем и получателем. Кроме того, PPP обеспечивает циклический контроль для каждого кадра, динамическое определение адресов, управление каналом. Наиболее широко используемый протокол для последовательного канала, обеспечивающий соединение компьютера с сетью Internet. Вытеснил SLIP.

**Протоколы межсетевого уровня:**

1) Протокол IPv4. Его основной задачей является доставка дейтаграмм (так называется единица передачи данных в терминологии IP). При этом протокол по определению является ненадежным и не поддерживающим соединения. Ненадежность протокола IP обусловлена тем, что нет гарантии, что посланная узлом сети дейтаграмма дойдет до места назначения. Сбой, произошедший на любом промежуточном узле сети, может привести к уничтожению дейтаграмм. Предполагается, что необходимая степень надежности должна обеспечиваться протоколами верхних уровней. IP не ведет никакого учета очередности доставки дейтаграмм: каждая дейтаграмма обрабатывается независимо от остальных. Поэтому очередность доставки может нарушаться. Предполагается, что учетом очередности дейтаграмм должен заниматься протокол верхнего уровня.

2) Протокол IPv6. Наиболее распространенной на настоящий момент версией протокола IP является IPv4 – именно об этой версии говорилось выше. Этот протокол оказался самым удачным сетевым протоколом из всех, когда-либо созданных. Поэтому IPv4 быстро превратился в стандарт. Можно сказать, что протокол IPv4 стал жертвой собственной популярности, т.к. предлагаемое полезное пространство адресов практически исчерпано (2^32). Как результат усилий, направленных на решение этой проблемы появился протокол IPv6, в котором попутно было реализовано много других новых возможностей. Главным отличительным признаком протокола IPv6 является 128-битный адрес, позволяющий увеличить адресное пространство боле чем на 20 порядков. Основная концепция IPv6: каждый отдельный узел должен иметь собственный уникальный идентификатор интерфейса. Кроме того, протокол IPv6 требует соответствие идентификаторов интерфейсов формату IEEE EUI-64, позволяющему применять фиксированные (“зашитые” при изготовлении в специальную память сетевой платы) MAC-адреса сетевых плат. Например, 48-битный МАС-адрес платы Ethernet изначально предназначен для глобальной идентификации. Первые 24 бита этого адреса обозначают производителя платы (в соответствии с кодировкой ICANN) и индивидуальную партию изделия, а остальные 24 бита определяются производителем, с таким расчетом, чтобы каждый номер был уникален в пределах всей его продукции. Таким образом уникальный идентификатор интерфейса IPv6 на основе Ethernet cодержит в младших 64-х разрядах 128-битного адреса MAC-адрес платы Ethernet. Причем дополнение 48-бит адреса MAC-адреса до 64 бит осуществляется добавлением 16 бит (0xFFFF) между двумя его половина ми. На первоначальном этапе внедрения IPv6 предполагается совместное использование обеих версий IP-протокола. При этом предполагается использование, так называемых, IPv4-совместимых и IPv4-преобразованных адресов. Другой интересной особенностью IPv6 является возможность автоконфигурации. Автоконфигурация – это процесс, позволяющий хосту находить информацию для настройки собственных IP-параметров. В версии IPv6 основным средством, позволяющим выполнять подобную настройку является протокол DHCP. Пересмотр процесса автоконфигурации вызван сложность администрирования сетей с большим количеством хостов и в связи с необходимостью поддерживать мобильных (перемещающихся) пользователей. Большое внимание в новой версии протокола уделяется вопросам безопасности.

3) ICMP (Internet Control Message Protocol) – транспортировка информации о сетевой деятельности и маршрутизации. IPCM сообщения представляют собой специально отформатированные IP-дейтаграммы, которым соответствуют определенные типы (15 типов) и коды сообщений. С помощью этого протокола осуществляется деятельность утилит достижимости (ping, traceroute); регулируется частота отправки IP-дейтаграмм, оптимизируется MTU для маршрута передачи IP-дейтаграмм; доставляется хостам, маршрутизаторам и шлюзам всевозможная служебная информация; осуществляется поиск и переадресация маршрутизаторов; оптимизируются маршруты; диагностируются ошибки и оповещаются узлы IP-сети;

4) ARP (Address Resolution Protocol) – динамическая проекция IP-адресов в соответствующие MAC-адреса аппаратных средств. Каждый хост кэширует специальную ARP-таблицу. Время существования записи в таблице обычно составляет 10-20 минут с момента ее создания и может быть изменено с помощью параметров реестра;

5) RARP (Reverse ARP) – получение IP-адреса по MAC-адресу;

**Протоколы транспортного уровня:**

**1) Протокол TCP**. Протокол TCP является надежным байт-ориентированным протоколом с установлением соединения. При получении дейтаграммы, в поле Protocol указан код 6 (код протокола TCP) IP-протокол извлекает из дейтаграммы данные, предназначенные для Транспортного уровня, и переправляет их модулю протокола TCP. Модуль TCP анализирует служебную информацию заголовка сегмента, проверяет целостность (по контрольной сумме) и порядок прихода данных, а также подтверждает их прием отправляющей стороне. По мере получения правильной последовательности неискаженных данных процесса отправителя, используя поле Destination Port Number заголовка сегмента, модуль TCP переправляет эти данные процессу получателя.

Протокол TCP рассматривает данные отправителя как непрерывный не интерпретируемый (не содержащий управляющих для TCP команд) поток октетов. При этом TCP при отправке разделяет (если это необходимо) этот поток на части (TCP-сегменты) и объединяет полученные от протокола IP-дейтаграммы при приеме данных. Немедленную отправку данных может быть затребовано процессом с помощью специальной функции PUCH, иначе TCP сам решает, когда отправлять данные отправителя и когда их передавать получателю.

Модуль TCP обеспечивает защиту от повреждения, потери, дублирования и нарушения очередности получения данных.

Протокол TCP обеспечивает одновременно нескольких соединений. Поэтому говорят о разделении каналов. Каждый процесс Прикладного уровня идентифицируется номером порта. Заголовок TCP-сегмента содержит номера портов отправителя и получателя.

**2) Протокол UDP.** Протокол UDP является протоколом без установления соединения. Спецификация протокола описывается в документе RFC 768. Основными свойствами протокола являются:

1) отсутствие механизмов обеспечения надежности: пакеты не упорядочиваются, и их прием не подтверждается;

2) отсутствие гарантий доставки: пакеты оправляются без гарантии доставки, поэтому процесс Прикладного уровня (программа пользователя) должен сам отслеживать и обеспечивать (если это необходимо повторную передачу);

3) отсутствие обработки соединений: каждый оправляемый или получаемый пакет является независимой единицей работы; UDP не имеет методов установления, управления и завершения соединения между отправителем и получателем данных;

4) UDP может по требованию вычислять контрольную сумму для пакета данных, но проверка соответствия контрольной суммы ложится на процесс Прикладного уровня;

5) отсутствие буферизации: UDP оперирует только одним пакетом, и вся работа по буферизации ложится на процесс Прикладного уровня;

6) UDP не содержит средств, позволяющих разбивать сообщение на несколько пакетов (фрагментировать) – вся эта работа возложена на процесс Прикладного уровня.

Следует обратить внимания, что протокол UDP характеризуется тем, что он не обеспечивает. Все перечисленные отсутствующие характеристики присутствуют в протоколе TCP. Фактически UDP – это тонкая прослойка интерфейса, обеспечивающая доступ процессов Прикладного уровня непосредственно к протоколу IP.

Основным отличием протоколов TCP и UDP является, то, что TCP – надежный байт-ориентированный протокол на основе соединения (ориентированный на поток), а UDP – ненадежный протокол без установки соединения (ориентированный на сообщения).

**Основные свойства протокола UDP:**

1) Отсутствие механизмов обеспечения надежности: пакеты не упорядочиваются, и их прием не подтверждается;

2) Отсутствие гарантий доставки, поэтому процесс прикладного уровня должен сам отслеживать и обеспечивать, если необходимо, повторную передачу;

3) Отсутствие обработки соединений: каждый отправляемый или получаемый пакет является независимой единицей работы; UDP не имеет методов установления, управления и завершения соединения между отправителем и получателем данных;

4) UDP может по требованию вычислять контрольную сумму для пакета данных, но проверка соответствия контрольной суммы ложится на процесс прикладного уровня;

5) Отсутствие буферизации: UDP оперирует только одним пакетом, и вся работа по буферизации ложится на процесс прикладного уровня;

6) UDP не содержит средств, позволяющих разбивать сообщение на несколько пакетов (фрагментировать) – вся эта работа возложена на процесс прикладного уровня.

Все перечисленные отсутствующие характеристики присутствуют в протоколе TCP.

**Надежный и ненадежный протоколы:**

**Надежный протокол** обеспечивает доставку данных без потерь, дублирования или изменения порядка передачи. Он гарантирует, что данные будут успешно доставлены получателю и подтверждает получение каждого пакета данных. Примером надежного протокола является TCP (Transmission Control Protocol), который используется в Интернете для передачи данных с гарантией доставки.

**Ненадежный протокол**, напротив, не гарантирует доставку данных. Он может потерять, дублировать или изменить порядок пакетов данных без уведомления об этом. Примером ненадежного протокола является UDP (User Datagram Protocol), который широко используется для стриминга видео, аудио и игр, где небольшая задержка важнее, чем гарантированная доставка данных.

**Протоколы с установкой соединения** требуют предварительного установления логического канала связи между отправителем и получателем перед передачей данных. Этот процесс называется "установкой соединения" и обычно включает в себя обмен определенными управляющими сообщениями. Примером протокола с установкой соединения является TCP, который устанавливает виртуальное соединение между клиентом и сервером перед передачей данных.

**Протоколы без установки соединения**,наоборот, не требуют предварительной установки логического канала связи. Они передают данные независимо друг от друга. Примером протокола без установки соединения является UDP, который просто отправляет пакеты данных без необходимости установки соединения.

**Протоколы, ориентированные на поток**, предоставляют непрерывный поток данных между отправителем и получателем. Они гарантируют, что данные будут доставлены в том же порядке, в котором были отправлены, без разделения на отдельные сообщения. TCP является примером протокола, ориентированного на поток.

**Протоколы, ориентированные на сообщения**, передают данные в виде отдельных сообщений или пакетов данных. Каждое сообщение может быть обработано независимо от других сообщений. UDP является

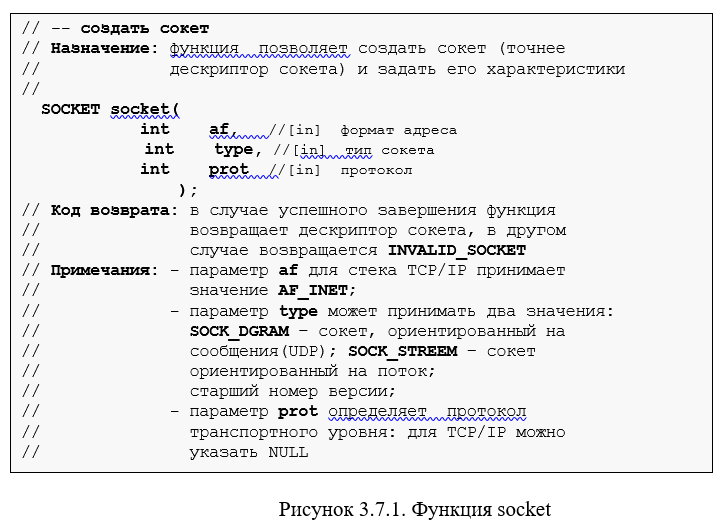
**4) Понятие сокета. Основные параметры сокета.**

**Шиман обращал внимание:** вспоминаем, что кроме ориентированных на сообщения и на поток есть другие (4 лаба - широковещательные). Что такое сокет, что он из себя представляет, Handle от Socketa чем отличается, параметры сокеты семейство адресов, протоколы и т. д. что можно вспомнить

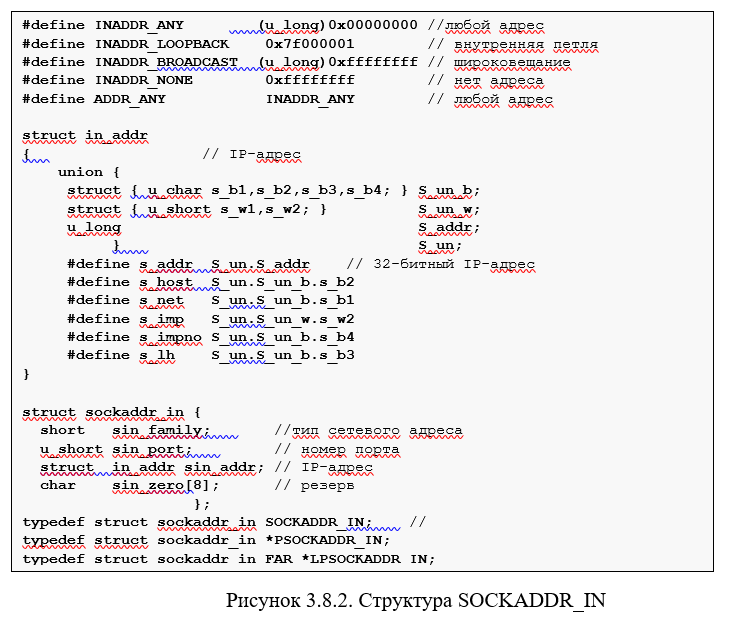
Интерфейс, который представляет совокупность IP-адреса и номера порта называется **сокетом**. Сокет однозначно идентифицирует прикладной процесс в сети TCP/IP. Следует помнить, что одни и те же номера портов могут быть использованы как для протокола UDP, так и для протокола TCP.

API сокетов – это название программного интерфейса, предназначенного для обмена данными между процессами, находящимися на одном или на разных объединенных сетью компьютерах.

В операционной системе Windows интерфейс сокетов имеет название **Windows Sockets API**. API сокетов включает в себя функции создания сокета (имеется в виду объект ОС, описывающий соединение), установки параметров сокета (сетевой адрес, номер порта и т.д.), функции создания канала и обмена данными между сокетами. Кроме того, есть набор функций, позволяющий управлять передачей данных, синхронизировать процессы передачи и приема данных, обрабатывать ошибки и т.п.



|  |  |
| --- | --- |
| **Наименование функции** | **Назначение** |
| **accept(**  **Socket s, сокет**  **Sockaddr\_in\* указатель на сокадр**  **Int\* la указатель на длину сокадр);** | Разрешить подключение к сокету |
| **bind( Socket s сокет**  **sockaddr\_in\* указатель на сокадр**  **int la длина сокадр в байтах)** | Связать сокет с параметрами (структура сокадр) |
| **closesocket(socket s, дескриптор сокета)** | Закрыть существующий сокет |
| **connect(**  **socket s сокет**  **sockaddr\_in\* указатель на сокадр**  **int la длина сокадр** | Установить соединение с сокетом |
| **gethostbyaddr(**  **char\* addr адрес в сетевом форм**  **int la длина адреса**  **int ta тип адреса (AF\_INET))** | Получить имя хоста по его адресу |
| **gethostbyname(**  **char\* name символическое имя хоста)** | Получить адрес хоста по его имени |
| **gethostname(**  **char\* name буфер имени**  **int ln длина имени)** | Получить имя хоста |
| **getsockopt** | Получить текущие опции сокета |
| **inet\_addr(char\* addr адрес в строке)** | Преобр-ть символьное представление IPv4-адреса в формат TCP/IP |
| **inet\_ntoa** | Преобр-ть сетевое представление IPv4-адреса в символьный формат |
| **ioctlsocket** | Установить режим ввода-вывода сокета |
| **listen(**  **socket s, сокет**  **int mcq максимальная длина очереди)** | Переключить сокет в режим прослушивания |
| **recv(**  **socket сокет**  **char\* buf буфер для данных**  **int lbuf, длина буфера**  **int flags флаги (индикатор))** | Принять данные по установленному каналу |
| **recvfrom(**  **socket s сокет**  **char\* buf буфер**  **int len длина буфера**  **int flag индикатор**  **sockaddr\* to указатель на сокадр**  **int tolen длина сокадра)** | Принять сообщение |
| **send(**  **socket s сокет**  **char\* buf буфер данных**  **int lbuf длина буфера**  **int flags индикатор особого режима)** | Отправить данные по установленному каналу |
| **sendto(**  **socket s сокет**  **char\* buf буфер**  **int len размер буфера**  **int flags индикатор**  **sockaddr\* to указатель на сокадр**  **int tolen длина сокадра)** | Отправить сообщение |
| **setsockopt(**  **socket s сокет**  **int level уровень действия режима**  **int optnamr режим сокета для установки**  **char\* optval значение режима сокета**  **int fromlen длина буфера optval)** | Установить опции сокета |
| **socket (  int af формат адреса**  **int type тип, int prot протокол)** | Создать сокет |
| **WSACleanup(void)** | Завершить использование библиотеки WS2\_32.DLL |
| **WSAGetLastError(void)** | Получить диагностирующий код ошибки |
| **WSAStartup(**  **WORD ver, версия Windows socket**  **lpWSAData wsd указатель на WSADATA)** | Инициализировать библиотеку WS2\_32.DLL |



IP-адрес и номер порта в структуре SOCKADDR\_IN хранятся в специальном сетевом формате. Этот формат отличается, от формата компьютеров с архитектурой Intel. В составе Winsock2 имеются функции, позволяющие преобразовывать форматы данных.

Для преобразования номера порта в формат TCP/IP следует использовать функцию htons. Функция ntohs является обратной функцией, предназначена для преобразования двух байтов в формате TCP/IP в формат u\_short.

Полезной является функция inet\_addr, предназначенная для преобразования символьного представления IPv4-адреса в формат TCP/IP. Функция inet\_ntoa предназначена для обратного преобразования из сетевого представления в символьный формат.

Структура **SOCKADDR\_IN** содержит три значения (параметры сокета):

• тип используемого адреса (константа AF\_INET используется для обозначения семейства IP-адресов);

• номер порта (устанавливается значение 2000 с помощью функции htons)

• адрес интерфейса.

Последний параметр определяет собственный IP-адрес сервера. При этом предполагается, что хост, в общем случае, может иметь несколько IP-интерфейсов. Если требуется использовать определенный IP-интерфейс хоста, то необходимо его здесь указать. Если выбор IP-адреса не является важным или IP-интерфейс один на хосте, то следует указать значение INADDR\_ANY (как это сделано в примере).

**struct sockaddr\_in {**

**short sin\_family;** //тип сетевого адреса

**u\_short sin\_port;** // номер порта

**struct in\_addr sin\_addr;** // IP-адрес

**char sin\_zero[8];** // резерв

**};**

**SOCKET socket (**

**int af,** //[in] формат адреса

**int type,** //[in] тип сокета

**int prot** //[in] протокол

**);**

- параметр **af** для стека TCP/IP принимает значение AF\_INET;

- параметр **type** может принимать два значения:

• **SOCK\_DGRAM** – сокет, ориентированный на сообщения (UDP);

• **SOCK\_STREAM** – сокет, ориентированный на поток;

• старший номер версии;

- параметр **prot** определяет протокол транспортного уровня:

• для TCP/IP можно указать NULL

Все функции интерфейса Winsock2 могут завершаться успешно или с ошибкой. При описании каждой функции будет указано, каким образом можно проверить успешность ее завершения. В том случае, если функция завершает свою работу с ошибкой, формируется дополнительный диагностирующий код, позволяющий уточнить причину ошибки.

Диагностирующий код может быть получен с помощью функции WSAGetLastError. Функция WSAGetLastError вызывается, непосредственно сразу после функции Winsock2, завершившейся с ошибкой.

**5) Интерфейс Named Pipe.**

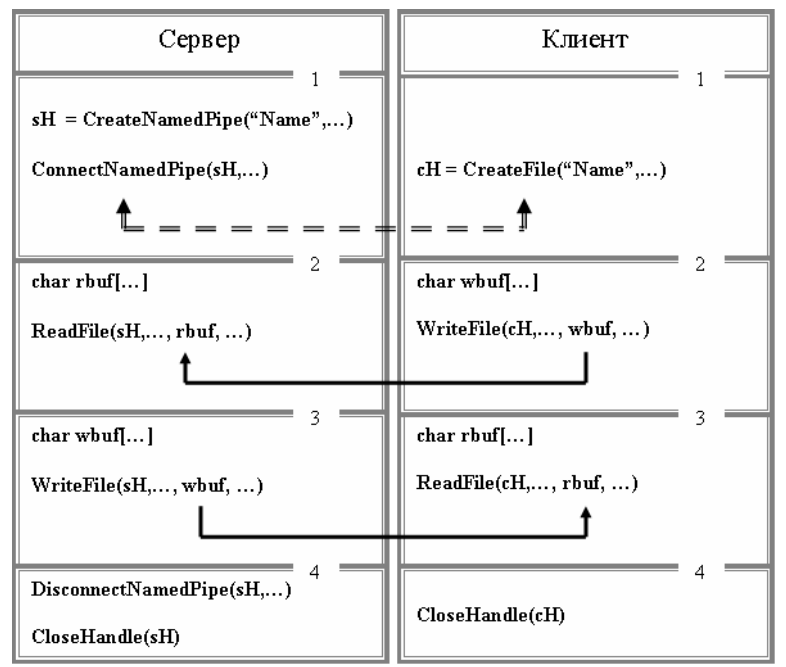
**Шиман обращал внимание:**  Вспоминаем что это такое, **для чего он предназначен главный интерфейс**(я не понял что он сказал) в каком направлении работает, что можно передавать, сообщения и пакеты, нужно вспомнить когда работают пакеты, когда сообщения. Функции, которые возможно применить к интерфейсу NamedPipe, они делятся на группы: управления каналом,передачи;запись\чтения\копирования\ transactnamedpipe чем отличается и функция callnamedpipe.

**Named Pipe –** объект ядра операционной системы, который обеспечивает обмен данными между процессами, выполняющимися на компьютерах в одной локальной сети. Любой именованный канал идентифицируется своим именем, которые задается при создании канала. Один из IPC-механизмов (Inter Process Communication) ОС Windows.

Именованные каналы бывают **дуплексные –** позволяют передать данные в обе стороны и **полудуплексные –** позволяющие передавать данные только в одну сторону. Передача данных в именованном канале может осуществляться как потоком, так и сообщениями. Обмен данными в канале может быть синхронным и асинхронным.

Все функции NamedPipeAPI можно разбить на **три группы**: функции управления каналом (создать канал, соединить сервер с каналом, открыть канал, получить информацию об именованном канале, получить состояние канала, изменить характеристики канала); функции обмена данными (писать в канал, читать из канала, копировать данные канала) и функции для работы с транзакциями.

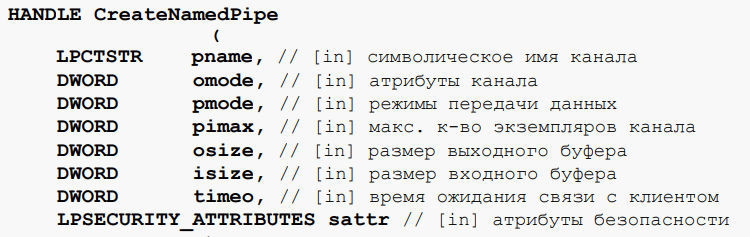




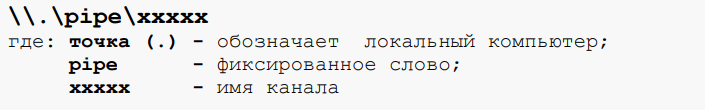
Функция ConnectNamedPipe() приостанавливает выполнение программы сервера до момента, пока программа клиента не выполнит функцию CreateFile().

В первом блоке программы клиента выполняется функция CreateFile(), одним из параметров которой является строка с именем канала.

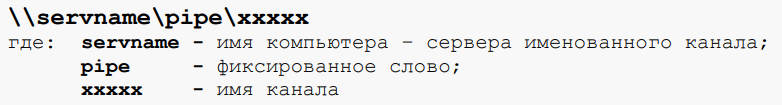
GetLastError() – получение кода системной ошибки Windows.

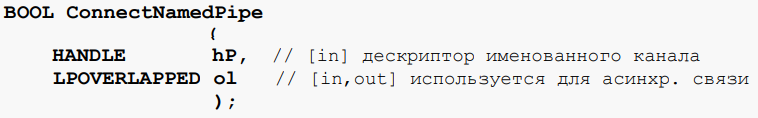


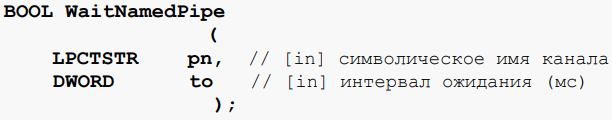
**Локальный формат имени канала:**



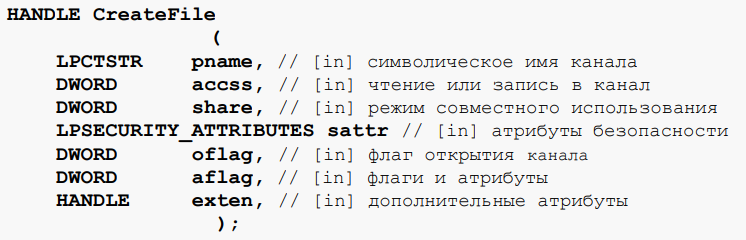
**Сетевой формат имени канала:**

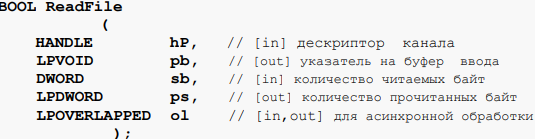
****

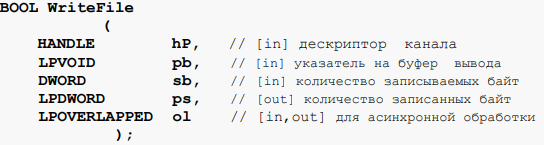
****

****

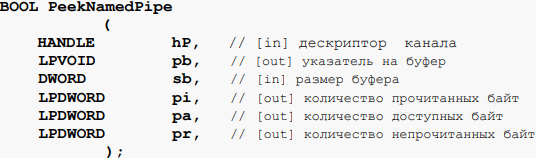
Если клиент локальный и использует сетевой формат имени, то обмен данными происходит сообщениями. Если клиент локальный и использует локальный формат имени, то обмен данными осуществляется потоком.



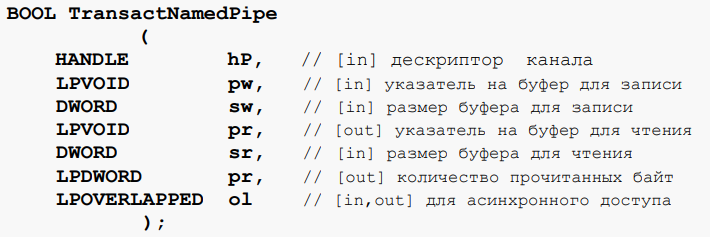




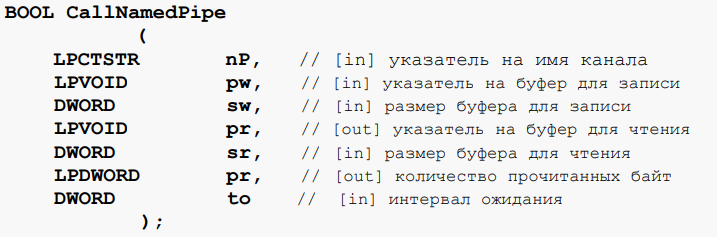
Функция PeekNamedPipe() копирует данные из канала в буфер, при этом данные не извлекаются и их еще можно считать (извлечь) с помощью функции ReadFile().



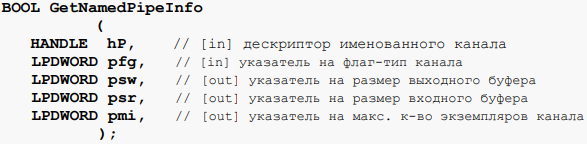
TransactNamedPipe() объединяет операции чтения и записи в одну операцию (транзакция).



CallNamedPipe() – устанавливает связь с именованным каналом, выполняет одну транзакцию и разрывает связь.



Для получения информации о созданном именованном канале можно использовать две функции: GetNamedPipeInfo() и GetNamedPipeHandleState().

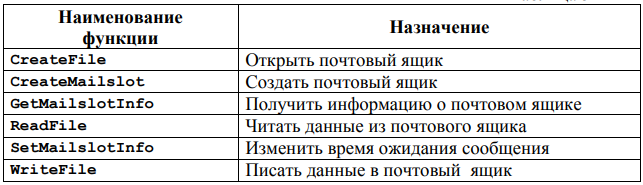


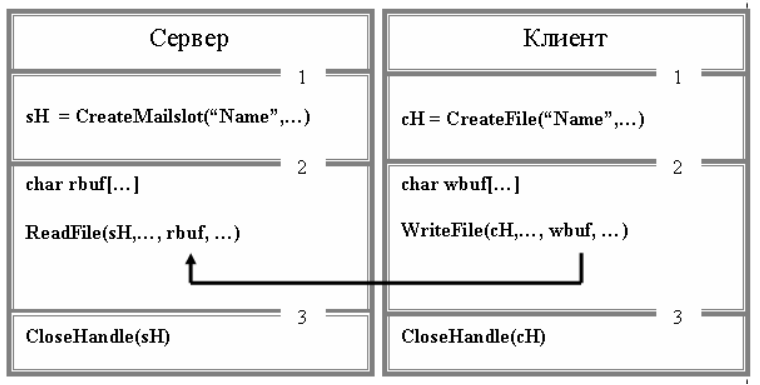
**6) Интерфейс MailSlot.**

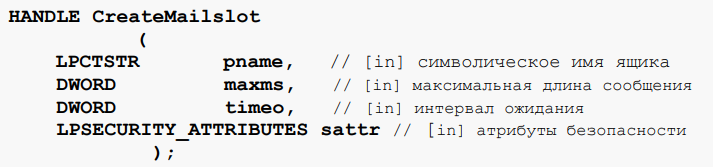
**MailSlot –** объект ядра операционной системы, который обеспечивает передачу данных от процессов-клиентов к процессам-серверам, выполняющимся на компьютерах в одной локальной сети. Один из IPC-механизмов (Inter Process Communication) ОС Windows.

Передача может осуществляться только в одном направлении – от клиента к серверу. Допускается создание нескольких серверов с одинаковым именем – в этом случае все отправляемые клиентом сообщения будут поступать во все почтовые ящики, имеющие имя, указанное клиентом. Такая рассылка сообщений возможно только в том случае, когда длина отправляемых сообщений не превышает 425 байт.

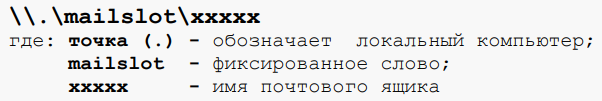
В случае если клиент отправляет сообщение меньше, чем 425 байт, то пересылка осуществляется без гарантий доставки. Пересылка сообщения размером более 425 байт возможна только от одного клиента к одному серверу.



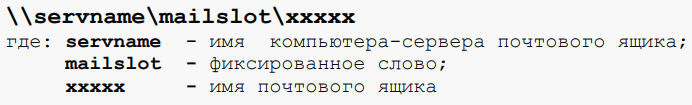




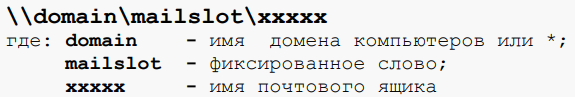
**Локальный формат имени почтового ящика:**

****

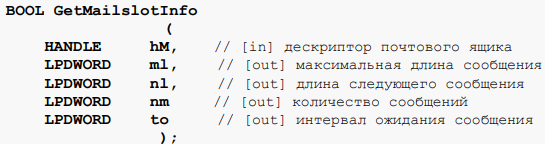
**Сетевой формат имени почтового ящика:**

****

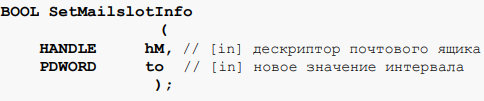
**Доменный формат имени почтового ящика:**

****

Получить информацию о характеристиках почтового ящика можно с помощью функции GetMailSlotInfo(). Эта функция может быть использована только на стороне сервера почтового ящика и параметр для дескриптора почтового ящика должен быть получен в результате выполнения функции CreateMailSlot().

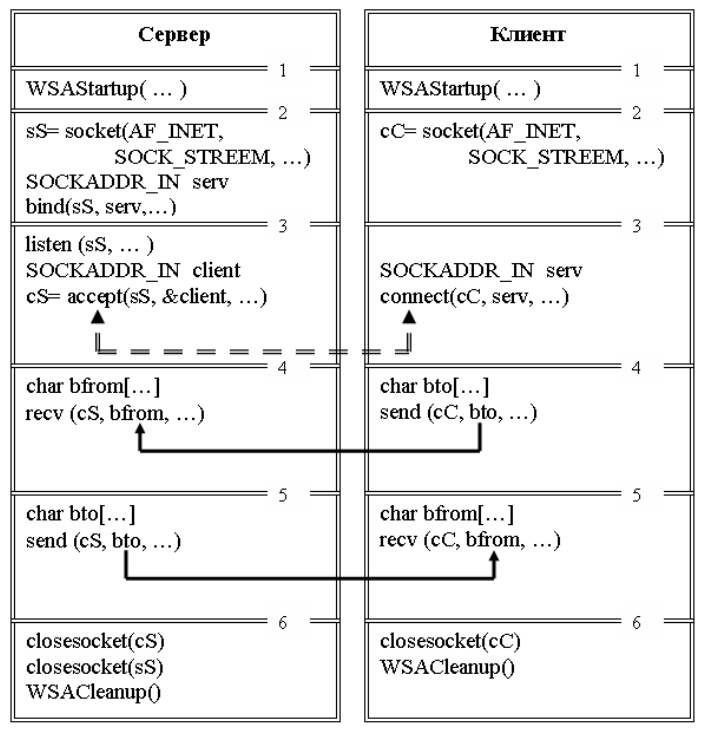


Функция SetMailsSlotInfo() предназначена для изменения интервала времени ожидания.



**7-8) Структура программы TCP-сервера/клиента.**

**Схема, ориентированная на поток –** между сокетами устанавливается TCP-соединение и весь обмен данными осуществляется в рамках этого соединения. Передача по каналу является надежной и данные поступают в порядке их отправления.

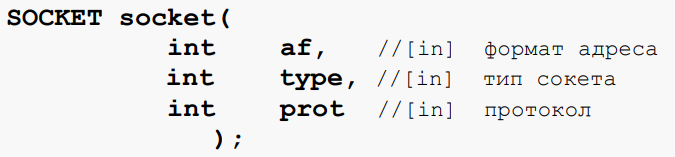


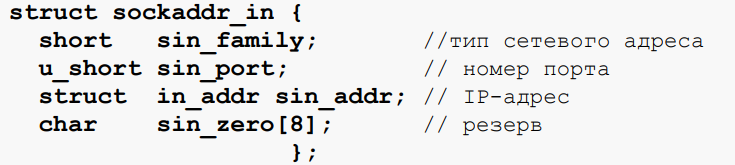
Второй блок сервера имеет тоже значение, что и в предыдущем случае. Единственным отличием является значение SOCK\_STREEM параметра функции socket(), указывающий, что сокет будет использоваться для соединения (сокет ориентированный на поток).

В третьем блоке программы сервера выполняются две функции Winsock2: listen() и accept(). Функция listen() переводит сокет, ориентированный на поток, в состояние прослушивания (открывает доступ к сокету) и задает некоторые параметры очереди соединений. Функция accept() переводит процесс сервера в состояние ожидания, до момента пока программа клиента не выполнит функцию connect() (подключится к сокету). Если на стороне клиента корректно выполнена функция connect(), то функция accept() возвращает новый сокет с эфемерным портом, который предназначен для обмена данными с подключившимся клиентом. Кроме того, автоматически заполняется структура SOCKADDR\_IN параметрами сокета клиента.

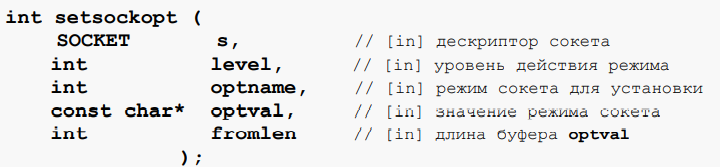
Четвертый и пятый блоки программы сервера предназначены для обмена данными по созданному соединению. Следует обратить внимание, что, во-первых, используется функция send() и recv(), а во-вторых, в качестве параметра эти функции используют сокет, созданный командой accept().

В третьем блоке программы клиента функция connect() предназначена для установки соединения с сокетом сервера. Функция в качестве параметров имеет, созданный в предыдущем блоке, дескриптор сокета (ориентированный на поток) и структуру SOCKADDR\_IN с параметрами сокета сервера.



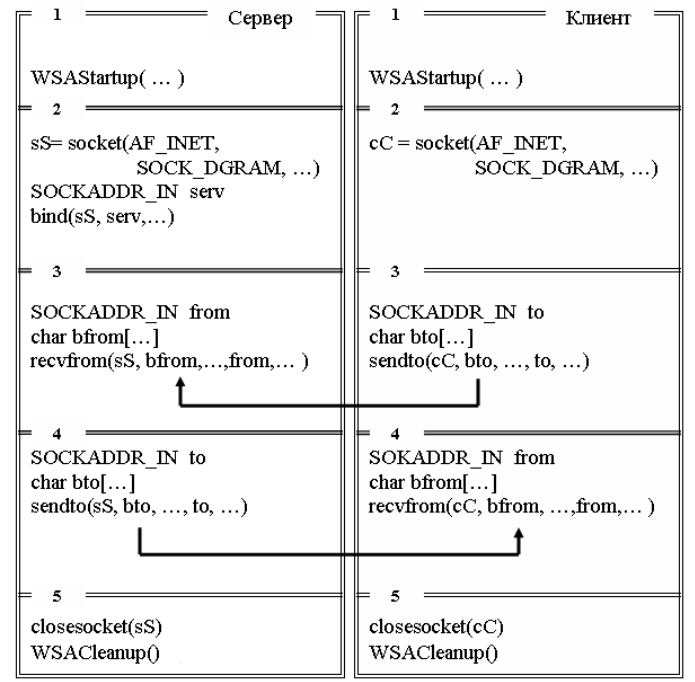


Использование широковещательных адресов возможно только в протоколе UDP. Установить различные режимы использования сокета можно с помощью функции setsockopt().



**9-10) Структура программы UDP-сервера/клиента.**

**Схема, ориентированная на сообщения –** между сокетами курсируют UDP-пакеты, и поэтому вся работа, связанная с обеспечением надежности и установкой правильной последовательности передаваемых пакетов возлагается на само приложение.



WSAStartup() – инициализация библиотеки WS2\_32.DLL.

Второй блок программы сервера создает сокет и устанавливает параметры этого сокета. Следует обратить внимание на параметр SOCK\_DGRAM, указывающий на тип сокета (в данном случае – ориентированный на сообщения). Для установки параметров сокета, используется функция bind(). Для хранения параметров сокета в Winsock2 предусмотрена структура SOCKADDR\_IN. Пока скажем, что в SOCKADDR\_IN хранится IP-адрес и номер порта сервера.

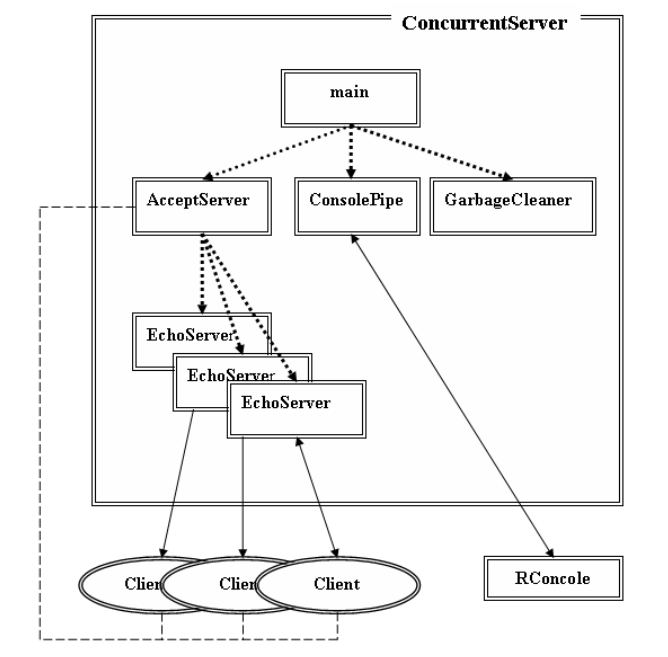
В третьем блоке программы сервера выполняется функция recvfrom(), которая переводит программу сервера в состояние ожидания, до поступления сообщения от программы клиента (функция sendto()). Функция recvfrom() тоже использует структуру SOCKADDR\_IN – в нее автоматически помещаются параметры сокета клиента, после приема от него сообщения. Данные поступают в буфер, который обеспечивает принимающая сторона (bfrom[…]). В качестве параметра функции recvfrom используется связанный сокет – именно через него осуществляется передача данных.

Четвертый блок программы сервера предназначен для пересылки данных клиенту. Пересылка данных осуществляется с помощью функции sendto(). В качестве параметров sendto() используется структура SOCKADDR\_IN с параметрами сокета принимающей стороны (в данном случае – клиента) и заполненный буфер с данными.

Пятые блоки одинаковые и предназначены для закрытия сокета и завершения работы с библиотекой WS2\_32.DLL.

Всем блокам программы клиента, кроме второго, есть аналог в программе сервере. Второй блок, в сравнении с сервером, не использует команду bind(). Здесь проявляется основное отличие между сервером и клиентом. Если сервер, должен использовать однозначно определенные параметры (IP-адрес и номер порта), то для клиента это не обязательно – ему Windows выделяет эфемерный порт. Т.к. инициатором связи является клиент, то он должен точно “знать” параметры сокета сервера, а свои параметры клиент получит от Windows и сообщит их вместе с переданным пакетом серверу.

**11-12) Структура параллельного сервера. AcceptServer. GarbageCleaner.**

****

**Процесс main –** основное назначение является запуск, инициализация и завершение работы сервера. Именно этот процесс первым получает управление от операционной системы. Процесс main запускает основные процессы: AcceptServer, ConsolePipe и GarbageCleaner.

**Процесс AcceptServer –** создается процессом main и предназначен для выполнения процедуры подключения клиентов к серверу, для исполнения команд консоли управления, а также для запуска процессов EchoServer, обслуживающих запросы клиентских программ по созданным соединениям. Кроме того, AcceptServer создает список подключений, который называется ListContact. При подключении очередного клиента, процесс AcceptServer добавляет в ListContact элемент, предназначенный для хранения информации о состоянии данного подключения.

**Процесс ConsolePipe –** создается процессом main и является сервером именованного канала, по которому осуществляется связь между программной RConsole и параллельным сервером.

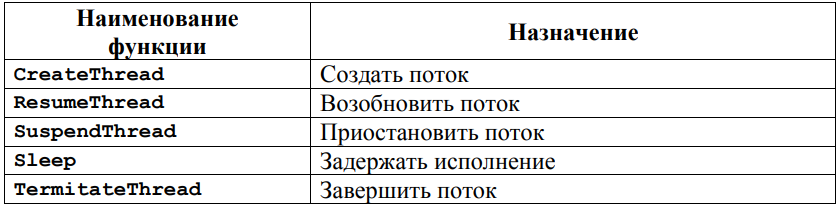
**Процесс GarbageCleaner –** удаление элемента списка подключений ListContact, после отключения программы клиента. Следует отметить, что ListContact является ресурсом, требующим последовательного использования. Одновременная запись и (или) удаление элементов списка может привести к разрушению списка ListContact.

**Процесс EchoServer –** создается процессом AcceptServer по одному для каждого успешного подключения программы клиента. Основным назначением процесса EchoServer является прием данных по созданному процессом AcceptServer подключению, и отправка этих же данных без изменения обратно программе клиента. Условием окончания работы сервера является получение от клиента пустого сегмента данных (имеющего нулевую длину).

**Процесс Client –** предназначена для пересылки данных серверу и получения ответа от сервера. Программа может работать, как на одном компьютере с сервером (будет использоваться интерфейс внутренней петли), так и на другом компьютере, соединенным с компьютером сервера сетью TCP/IP. Для окончания работы с сервером программа формирует и отправляет сегмент данных нулевой длины.

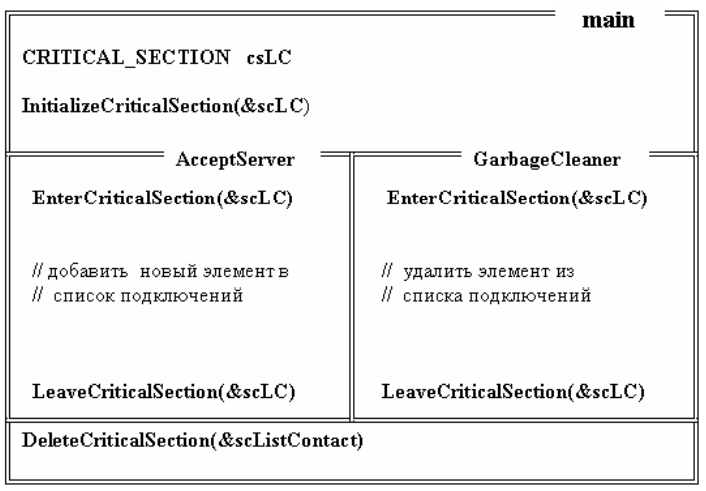
**Процесс RConsole –** предназначена для ввода команд управления сервером и для вывода диагностических сообщений, полученных от сервера. RConsole является клиентом именованного канала.

**Список подключений ListContact –** список создается на основе стандартного класса list и предназначен для хранения информации о каждом подключении. Список создается пустым при инициализации процесса AcceptServer. В рамках этого же процесса осуществляется добавление элементов списка, по одному для каждого подключения. При отключении программы клиента от сервера, соответствующий элемент списка помечается, как неиспользуемый. Удаление неиспользуемого элемента осуществляется процессом GarbageCleaner, который работает в фоновом режиме.

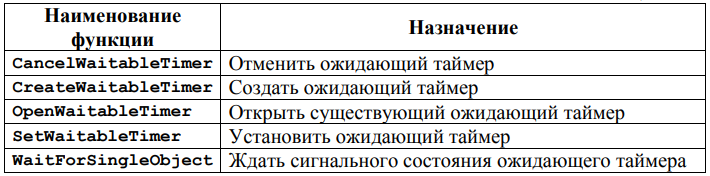


Синхронизация в параллельном сервере будет осуществляться между AcceptServer, который добавляет клиента в ListContact и GarbageCleaner, который удаляет клиента из ListContact.

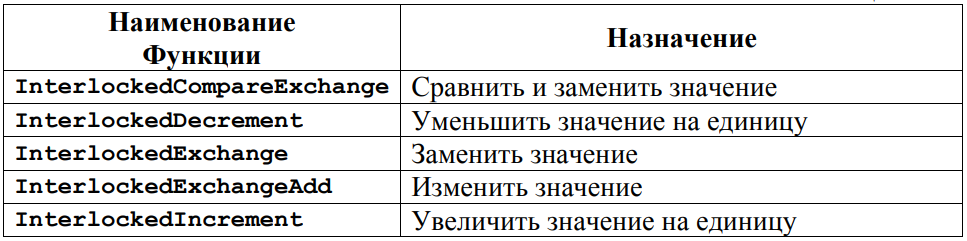
**Критические секции:**

****

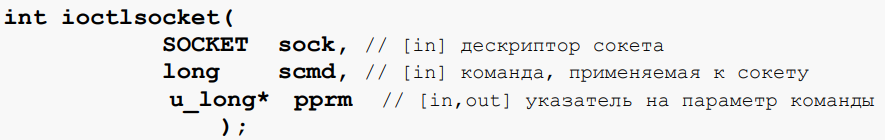
**Ожидающим таймер** в Windows называется объект синхронизации, который переходит в сигнальное состояние при наступлении заданного момента времени.



Блокирующие функции выполняют несколько элементарных операций, которые объединяются в одну неделимую операцию, называемую **атомарной операцией.**

****

**Переключение сокета в режим без блокировки**. В этом режиме выполнение функции accept(), не приостанавливает выполнение потока, как это было прежде, а возвращает значение нового сокета, если обнаружен запрос на создание канала (функция connect(), выполненная клиентом), или значение INVLID\_SOCKET, если запроса на создание канала нет в очереди запросов или возникла ошибка. Для переключение сокета применяется функция ioctlsocket().



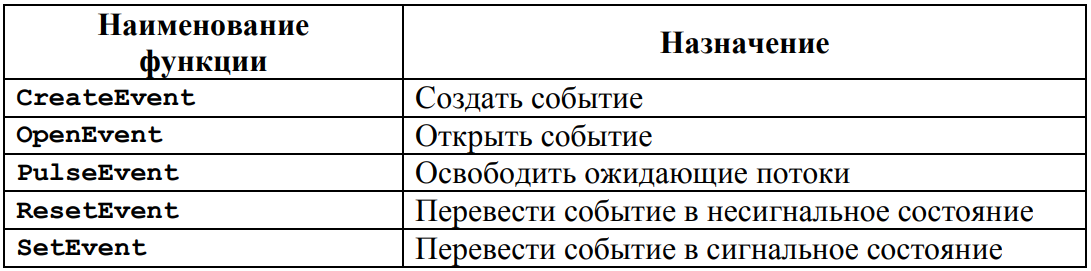


**Обработка запросов клиента.** Основным отличием новой структуры, является промежуточный поток DispatchServer между AcceptServer и обслуживающими потоками ServiceServer (раньше это был единственный поток EchoServer). Теперь предполагается, что сначала программа клиента осуществляет процедуру подключения (для этого используется поток AcceptServer), потом поток DispatchServer принимает от клиента запрос (команду) на обслуживание и после этого уже запускается соответствующий поток ServiceServer, который исполняет команду и в случае необходимости обменивается данными с клиентом.

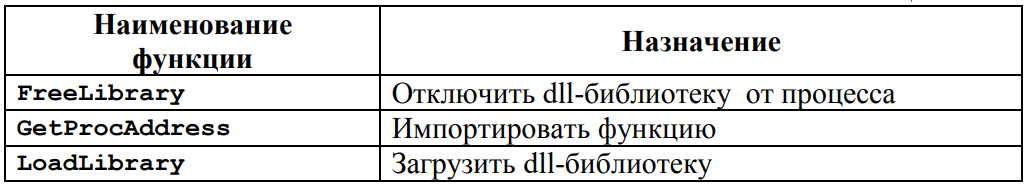
Введение промежуточного звена, обуславливается тем, что после этапа подключения, клиент (в общем случае) может достаточно долго не запрашивать у сервера услугу (не выполнять функцию send()), пересылающую команду сервера). Ожидать поступление команды в потоке AcceptServer не целесообразно, т.к. его основное назначение – подключение клиентов.

На поток DispatchServer возлагается прием первой команды от клиента после подключения.

**Механизм события,** позволяет оповестить поток о некотором выполненном действии, произошедшем за пределами потока.



**Динамически подключаемые библиотеки** позволяют менять поддерживаемый сервером сервис при неизменной логике управления сервером и обслуживания клиентов.



При разработке системы безопасности сервера, целесообразно использовать систему безопасности операционной системы. Это значительно снизит затраты на ее разработку.

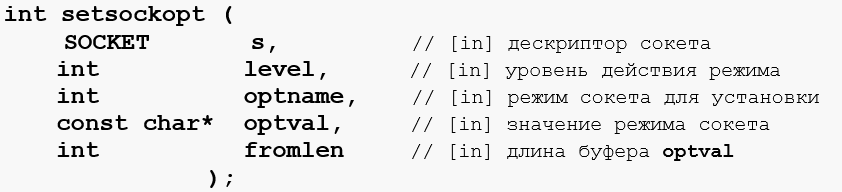
**13) Широковещание. Обнаружение сервера с помощью широковещания.**

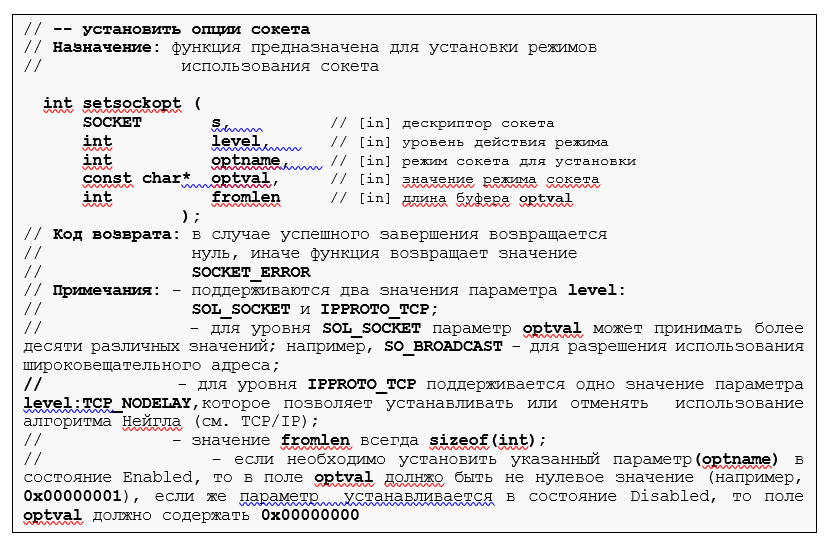
Для обеспечения независимости приложения от параметров сокета сервера (сетевой адрес и номер порта), как правило, номер порта делают одним из параметров инициализации сервера и хранят в специальных конфигурационных файлах, которые считываются сервером при загрузке (реже номер порта передается в виде параметра в командной строке). Так, например, большинство серверов баз данных в качестве одного из параметров инициализации используют номер порта, а при конфигурации (или инсталляции) клиентских приложений указывается сетевой адрес и порт сервера.

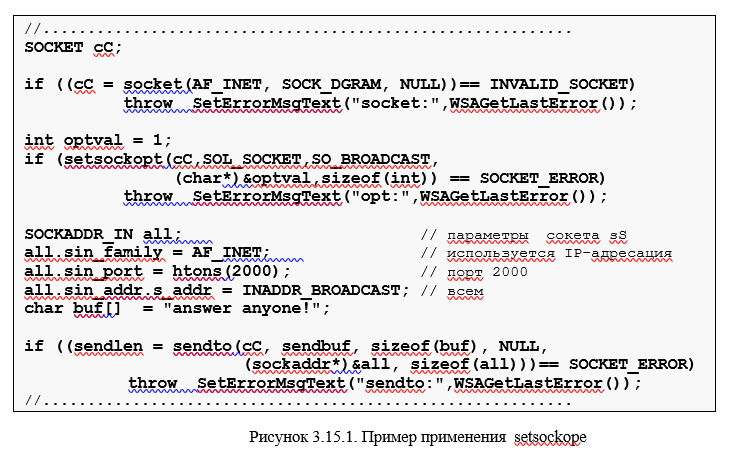
В некоторых случаях удобно возложить поиск сетевого адреса на само клиентское приложение (при условии, что номер порта сервера неизвестен). В этих случаях используются широковещательные сетевые адреса, позволяющие адресовать сообщение о поиске сервера всем компьютерам сети. Предполагается, что сервер (или несколько серверов) должен находиться в состоянии ожидания (прослушивания) на доступном в сети компьютере. При получении сообщения от клиента, сервер определяет параметры сокета клиента и передает клиенту необходимые данные для установки канала связи. В общем случае в сети может находиться несколько серверов, которые откликнутся на запрос клиента. В этом случае алгоритм работы клиента должен предполагать процедуру обработки откликов и выбора подходящего сервера. Сразу следует оговориться, что реально данный метод можно применять только внутри сегмента локальной сети, т.к. широковещательные пакеты, как правило, не пропускаются маршрутизаторами и шлюзами.

Использование широковещательных адресов возможно только в протоколе UDP.

Стандартный широковещательный адрес в формате TCP/IP задается с помощью константы INADDR\_BROADCAST, котораяопределена в Winsock2.h. По умолчанию использование стандартного широковещательного адреса не допускается и для его применения необходимо установить специальный режим использования сокета SO\_BROADCASTс помощью функции setsockopt(). Проверить установленные для сокета режимы можно с помощью функции getsockopt().





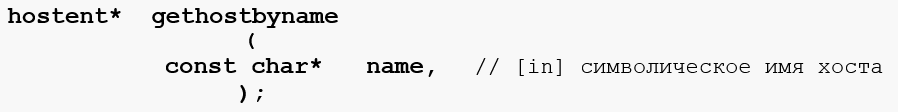


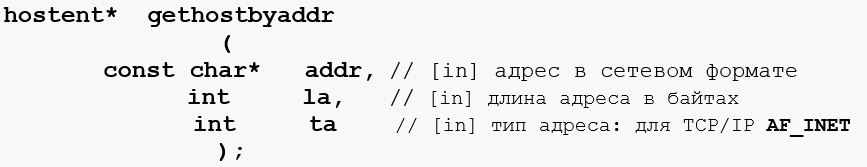
**14) Применение символического адреса хоста.**

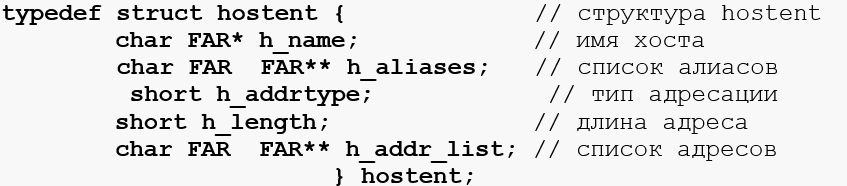
При наличии специальной службы (программная реализация протоколов прикладного уровня TCP/IP) в сети способной разрешить адрес компьютера по его символическому имени (например, DNS или некоторые протоколы, работающие поверх TCP/IP) поиск серверного компьютера можно осуществить с помощью функции gethostbyname(). При этом предполагается, что известно символическое имя компьютера, на котором находится программа сервера.

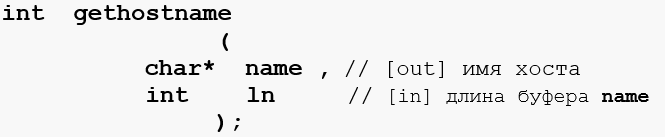
Такое решение довольно часто применяется разработчиками распределенных систем. Связав набор программ-серверов с определенными стандартными именами компьютеров, распределенное приложение становится независимым от адресации в сети. Естественно при этом необходимо позаботится, чтобы существовала служба, разрешающая адреса компьютеров по имени. Установка таких служб, как правило, возлагается на системного администратора сети.

Помимо функции gethostbyname() в составе Winsock2 имеется функция gethostbyaddr(), назначение которой противоположное: получение символического имени компьютера по сетевому адресу. Обе функции используют структуру hosten, содержащуюся в Winsock2.h.



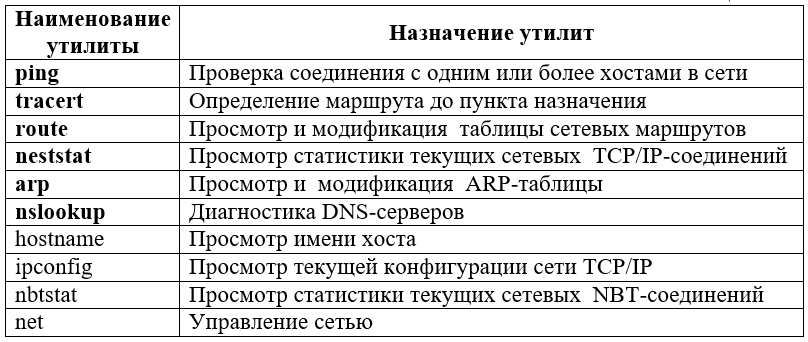






**15) Основные сетевые утилиты и их назначение.**

**Сетевые утилиты** представляют собой внешние команды операционной системы и предназначаются для диагностики сети. При этом утилиты, выделенные жирным шрифтом считаются стандартными для протокола TCP/IP и присутствуют в большинстве операционных систем.



**Route**. Утилита обеспечивает 4 команды: print (распечатка таблиц сетевых маршрутов), add (добавить маршрут в таблицу), change (изменение существующего маршрута), delete (удаление маршрута).

**Netstat.** Активные соединения TCP/IP можно просмотреть, набрав на консоли команду netstat с пар-ром -a.

**Arp.** Текущее состояние ARP-таблицы можно с помощью arp -a.

**Ipconfig.** Можно использовать ipconfig /all для получения полного отчета о конфигурации TCP/IP.

**Net.** Управление сетью. С помощью этой команды можно зарегистрировать пользователя в рабочей группе Windows, осуществить выход из сети, запустить или остановить сетевой сервис, управлять списком имен, пересылать сообщения в сети, синхронизировать время и т.д.

**16) Служба DNS.**

**DNS работает на прикладному уровне модели ISO/OSI.**

**Службу DNS (Domain Name System**) можно рассматривать, как распределенную иерархическую базу данных, основное назначение которой отвечать на два вида запросов: выдавать IP-адрес по символическому имени хоста и наоборот. База данных имеет древовидную структуру, в корне которой ничего нет, а сразу под корнем находятся первичные сегменты (домены): .com, .edu, .gov, …, .ru, .by, … Наименование этих первичных доменов отражает деление базы данных DNS по отраслевому и национальному признакам. Домен в терминологии DNS называется любое поддерево дерева базы данных DNS.

**Служба DNS состоит из 3 основных компонент:**

1) Пространство имен DNS и соответствующие ресурсные записи (RR, resource record) – это сама распределенная база данных DNS;

2) Серверы имен DNS – компьютеры, хранящие базу данных DNS и отвечающие на запросы DNS-клиентов;

3) DNS-клиенты – компьютеры, посылающие запросы серверам DNS для получения ресурсных записей.

Информация о доменах, хранящаяся в базе данных сервера DNS, организуется в особые единицы, называемые **зонами.**

**Типы зон:**

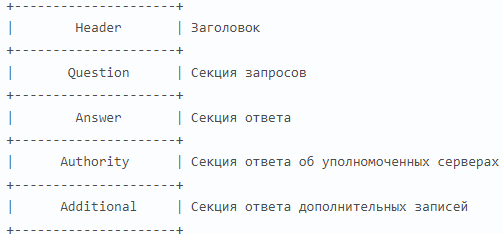
1) Стандартная основная (Standard primary) – главная копия стандартной зоны. Только в данном экземпляре зоны допускается производить какие-либо изменения, которые затем реплицируются на серверы, хранящие дополнительные зоны;

2) Стандартная дополнительная (Standard secondary) – копия основной зоны, доступная в режиме только чтение. Предназначена для повышения отказоустойчивости и распределения нагрузки между серверами, отвечающими за определенную зону;

3) Интегрированная в Active Directory (Active Directory-integrated) – вся информация о зоне хранится в виде одной записи в базе данных Active Directory;

4) Зона-заглушка (stub, только в Windows 2003) – особый тип зоны, которая для данной части пространства имен DNS содержит самый минимальный набор ресурсных записей.

**Структура DNS:**

****

**Типы DNS-записей (RR):**

1) A – связывает доменное имя и IP-адрес;

2) MX – управляет маршрутами почтовых сообщений для протокола SMTP;

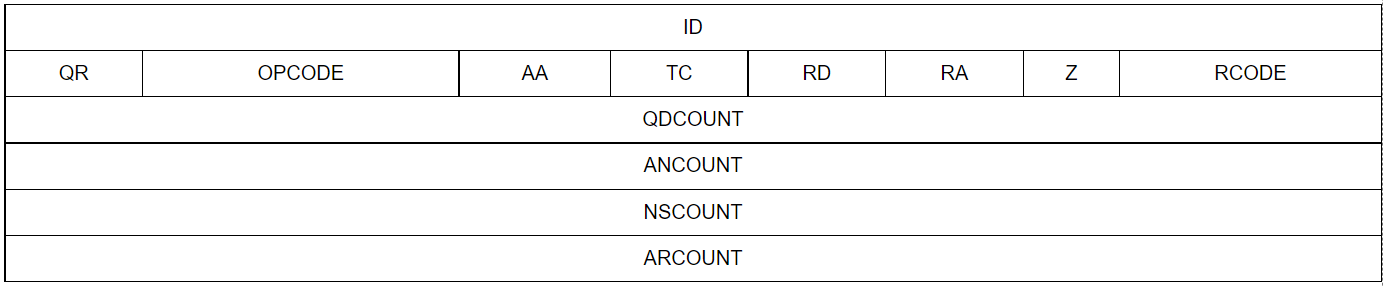
3) NS – указывает на серверы DNS, ответственные за конкретный домен и его поддомены;

4) SOA – используется для указания основного сервера для данной зоны и описания свойств зоны;

5) PTR – используется для обратного разрешения IP-адресов в имена узлов в домене in-addr.arpa;

6) SRV – используется для поиска серверов, на которых функционируют определенные службы;

7) CNAME – отображает одно имя на другое.

****

**ID –** идентификатор. 2 байта.

**QR** – идентификатор, обозначает, данное сообщение запрос от клиента или ответ от сервера.

**OPCODE** – клиент указывает тип запроса к серверу DNS.

**AA** – имеет смысл только в пакетах ответа от сервера клиенту. Если установлен данный флаг, значит сервер сгенерировал ответ.

**TC** – данный флаг устанавливается в пакете ответа в том случае, если сервер не смог поместить всю необходимую информацию в пакет из-за существующих ограничений.

**RD** – заголовок DNS. Клиент просит сервер не сообщать ему промежуточный ответ, а сообщить клиенту только конечный результат.

**RA** – данный флаг устанавливается в ответах. Служит для ответа клиенту поддерживает ли сервер рекурсивное обслуживание.

**RCODE** – поле кода ответа, либо с ошибкой, либо успешно. 0 – без ошибок, 1 – ошибка разрешения имен, и эта ошибка связана с DNS сервером который не смог понять формат запроса клиента, 2 – означает что сервер сигнализирует о неисправностях сервера, 3 – имени не существует в данном домене, 4 – сервер не может выполнить запрос данного типа.

**QDCOUNT** – количество запросов.

**ANCOUNT** – количество ответов.

**NSCOUNT** – количество записей в Authority Section.

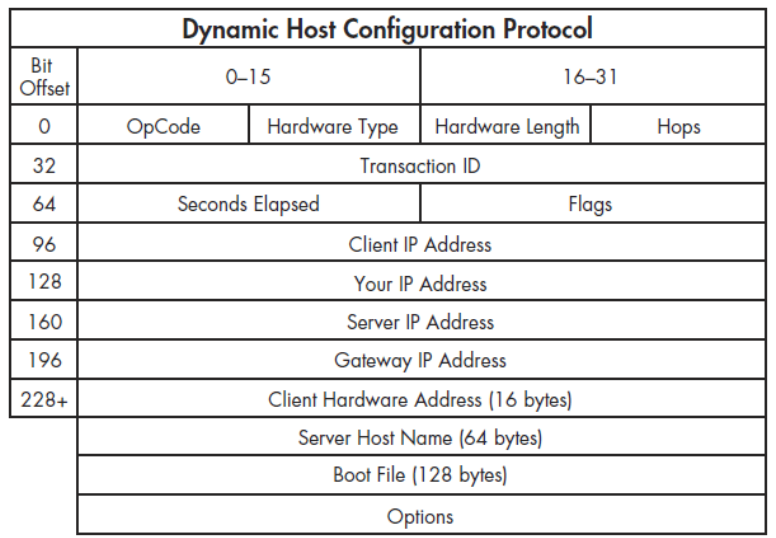
**ARCOUNT** – количество записей в Additional Record Section.

**17) Служба DHCP.**

**DHCP (Dynamic Host Configuration Protocol) –** это сетевая служба прикладного уровня TCP/IP, обеспечивающая выделение и доставку IP-адресов и сопутствующей конфигурационной информации (маска подсети, адрес локального шлюза, адреса серверов DNS и т.п.) хостам. Позволяет отказаться от фиксированных IP-адресов в зоне действия сервера DHCP.

**Служба DHCP состоит из 3 модулей:** сервера DHCP, клиента DHCP и ретранслятора DHCP (используется в том случае, если на первоначальном этапе подключения к сети широковещательные запросы DHCP-клиента не могут быть доставлены (по разным причинам) DHCP-серверу. Играет роль посредника между ними).

Для обнаружения DHCP-сервера DHCP-клиент выдает в сеть широковещательный запрос. Если в этом домене есть DHCP-сервер, то он окликается, посылая клиенту специальное сообщение, содержащее IP-адрес DHCP-сервера. Если доступны несколько DHCP-серверов, то, как правило, выбирается первый ответивший. Получив адрес сервера, клиент формирует запрос на выделение IP-адреса из пула адресов DHCP-сервера. В ответ на запрос, DHCP-сервер выделяет адрес клиенту на определенный период времени. После получения IP-адреса TCP/IP-стек клиента начинает его использовать. Продолжительность аренды адреса устанавливается специально или по умолчанию (может колебаться от нескольких часов до нескольких недель). После истечения срока аренды DHCP-клиент пытается снова договорится с DHCP-сервером о продлении срока аренды или о выделении нового IP-адреса.

****

**OpCode –** тип сообщения.

**Hops –** количество промежуточных маршрутизаторов.

**Second Elapsed –** время в секундах с момента начала процесса получения адреса.

**CSMA/CD (Carrier Sense Multiple Access with Collision Detection) –** множественный доступ с прослушиванием несущей и обнаружением коллизий.

Если во время передачи кадра рабочая станция обнаруживает другой сигнал, занимающий передающую среду, то она останавливает передачу, посылает сигнал преднамеренной помехи и ждет в течении случайного промежутка времени, перед тем как снова отправить кадр.

**18) Стандарты сообщений Internet.**

Сообщения, соответствующие данной спецификации, включают символы с десятичными кодами от 1 до 127, интерпретируемые в соответствии с кодировкой US-ASCII.

Данная спецификация вносит 2 ограничения на число символов в строке. Строка должна содержать не более 998 символов; следует использовать строки размером не более 78, без учета CRLF.

Поля заголовков представляют собой строки, начинающиеся с имени поля, за которым следует двоеточие, содержимое поля и знак завершения строки CRLF. Имя поля должно состоять только из печатаемых символов US-ASCII (т.е. символов с кодами от 33 до 126 включительно), исключая двоеточие. Значение поля может включать печатаемые символы US-ASCII, символы пробела (SP) и горизонтальные табуляции.

**Фальцовка –** разбиение строки на несколько строк.

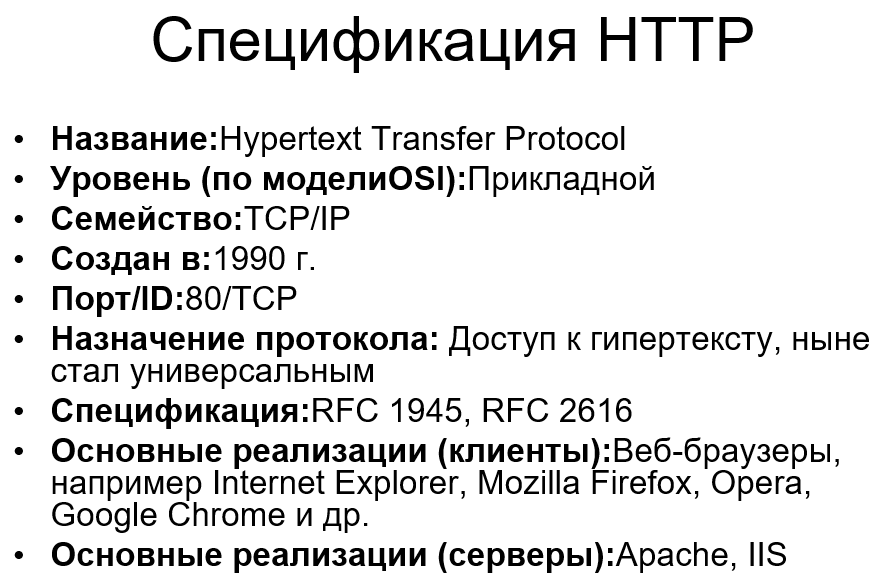
**Расфальцовка –** процесс преобразования фальцованного многострочного представления поля в обычное однострочное.

Некоторые символы имеют специальное значение (например, используются в качестве границ лексем). Для использования таких символов в общепринятом смысле служит механизм квотирования (добавления кавычек).

Поле addr-spec представляет собой специфический для Internet идентификатор, содержащий локально интерпретируемую строку, за которой следует символ “@” и доменное имя Internet. Это локально интерпретируемая строка представляет собой строку в кавычках или атом с точкой.

**19) Протокол HTTP.**

**HTTP (HyperText Transfer Protocol) –** протокол передачи данных.



**Каждое HTTP-сообщение состоит из 3 частей:**

1) Стартовая строка – определяет тип сообщения;

2) Заголовки – характеризуют тело сообщения, параметры передачи и прочие сведения;

3) Тело сообщения – непосредственно данные сообщения. Обязательно должно отделяться от заголовков пустой строки.

Заголовки и тело могут отсутствовать, но стартовая строка является обязательной.

**Методы HTTP:**

1) GET – запрашивает представление ресурса. Запросы с использованием этого метода могут только извлекать данные;

2) HEAD – запрашивает ресурс так же, как и метод GET, но без тела ответа;

3) POST – используется для отправки сущностей к определенному ресурсу. Часто вызывает изменение состояния или какие-то побочные эффекты на сервере;

4) PUT – заменяет все текущие представления ресурса данными запроса;

5) DELETE – удаляет указанный ресурс;

6) CONNECT – устанавливает “туннель” к серверу, определенному по ресурсу;

7) OPTIONS – используется для описания параметров соединения с ресурсом;

8) TRACE – выполняет вызов возвращаемого тестового сообщения с ресурса;

9) PATCH – используется для частичного изменения ресурса.

**Коды состояния:**

1) 1xx – коды, информирующие о процессе передачи;

2) 2xx – сообщения данного класса информируют о случаях успешного принятия и обработки запроса клиента (200 – успешно);

3) 3xx – сообщают клиенту, что для успешного выполнения операции необходимо сделать другой запрос;

4) 4xx – ошибки со стороны клиента (400 – плохой запрос, 401 – не авторизирован, 404 – не найден);

5) 5xx – ошибки со стороны сервера, должен включать в тело сообщение, которое клиент отобразит пользователю (500 – внутренняя ошибка сервера, 501 – не выполнено, 502 – плохой шлюз, 503 – недоступен, 505 – HTTP-версия не поддерживается).

**20) Служба RPC.**

**RPC (Remote Procedure Call) –** удаленный вызов процедур. Позволяет вызывать функции и процедуру в другом адресном пространстве.

**Порядок действий:**

1) Процедура клиента вызывает клиентскую заглушку;

2) Клиентская заглушка создает сообщение и вызывает функцию RPC локальной ОС;

3) Служба RPC пересылает сообщение серверу;

4) Служба RPC вызывает серверную заглушку и передает ей сообщение;

5) Серверная заглушка извлекает из сообщения параметры и вызывает удаленную процедуру;

6) Удаленная процедура выполняет код и возвращает параметры и значения серверной заглушке;

7) Серверная заглушка формирует сообщение и вызывает службу RPC своей локальной ОС;

8) Служба RPC сервера пересылает сообщение RPC ОС клиента;

9) RPC клиента возвращает сообщение заглушке;

10) Заглушка извлекает данные и передает их процессу.

**Маршалинг –** процесс укомплектования параметров в сообщение.

**Самый длительный процесс – процесс передачи.**

**Дополнительно**

Интерфейс RPC определяет программный механизм, который первоначально был разработан в компании Sun Microsystems и предназначался для того, чтобы упростить разработку распределенных приложений. Спецификация RPC компании Sun Microsystems содержится в документах RFC 1059, 1057, 1257.

RPC Sun Microsystems реализована в двух модификациях: одна выполнена на основе API сокетов для работы над TCP и UDP, другая, названная ***TI-RPC*** ***(Transport Independent RPC)***, использует API TLI (Transport Layer Interface, компании AT&T) и способна работать с любым транспортным протоколом, поддерживаемый ядром операционной системы.

Идея, положенная в основу RPC, заключается в разработке специального API, позволяющего осуществлять вызов ***удаленной процедуры*** (процедуры, которая находится и исполняется на другом хосте) способом, по возможности, ничем не отличающимся от вызова локальной процедуры из динамической библиотеки. Реализация этой идеи осложняется необходимостью учитывать возможность различия операционных сред, в которых работают вызывающая и вызываемая процедуры (отсюда, различные типы данных, невозможность обрабатывать адресные указатели и т.п.). Кроме того, следует предусмотреть обработку внепланового завершения процедуры на одной из сторон распределенного приложения. Все эти проблемы сделали интерфейс RPC достаточно сложным. Прозрачность механизма вызова достигается созданием вместо вызываемой и вызывающей процедур специальных программных заглушек, называемых ***клиентским*** и ***серверным*** ***стабами.***

Клиентским стабом называется тот стаб, который находится на хосте с вызывающей процедурой. Его основной задачей является преобразовать передаваемые параметры в формат стандарта ***XDR (External Data Representation)*** и скрыть (подменив вызываемую удаленную процедуру локальным вызовом стаба) от пользователя механизм RPC.

Серверный стаб находится на том же хосте, что и вызываемая процедура и предназначен для преобразования полученных параметров из формата XDR в формат, воспринимаемый вызываемой процедуры, а также для сокрытия (серверный стаб подменяет вызывающую процедуру на стороне сервера) RPC-механизма от вызываемой процедуры.

Стандарт XDR предназначен кодирования полей в запросах и ответах интерфейса RPC. Стандарт регламентирует все типы данных и уточняет способ их передачи в RPC-сообщениях. Спецификация стандарта XDR приведена в RFC 1014.

Число вызываемых удаленных процедур не регламентируется спецификацией RPC. Поэтому за ними не закрепляются конкретные TCP-порты. Порты получают сами удаленные процедуры динамическим образом (эфемерные порты). Учет соответствия портов вызываемым процедурам осуществляет специальная программа ***PortMapper*** (регистратор портов). Сам PortMapper доступен по 111 порту и тоже является удаленной процедурой. Процедура PortMapper – это связующее звено между различными компонентами системы. Всякая вызываемая процедура, должна быть зарегистрирована в базе данных PortMapper с помощью специальных служебных функций. Вызывающая сторона (клиентский стаб) с помощью все тех же служебных функций может получить спецификацию вызываемой процедуры.

Развитием технологии RPC для объектно-ориентированного программирования в операционной системе Windows являются технологии ***COM*** и ***DCOM***, которые позволяют создавать удаленные объекты.

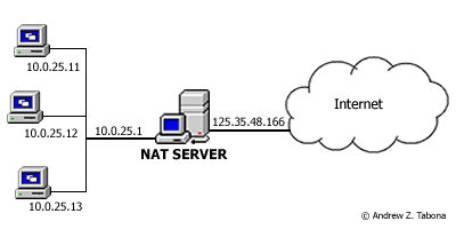
Аналогом RPC в Java-технологиях является механизм ***RMI (Remote Method Invocation)***, позволяющий работать с удаленными Java-объектами. Информацию об объекте и его методах вызывающая сторона может получить, обратившись к реестру RMI (аналогу PortMapper).

Сетевая файловая система ***NFS (Network File System)***, повсеместно используемая в качестве службы прозрачного удаленного доступа к файлам основана на механизме RPC Sun Microsystems.

Следует отметить, что кроме Sun-реализации интерфейса RPC, широко применяется и конкурирующий программный продукт, разработанный объединением OSF (Open Software Foundation).

**21) NAT, proxy-серверы, межсетевые экраны, ремайлеры.**

**NAT (Network Address Translation) –** это механизм в сетях TCP/IP, позволяющий преобразовывать IP-адреса транзитных пакетов.



Существует 3 вида **NAT** устройств:

* Статический – отображает не зарегистрированный на зарегистрированный на основании 1 к 1. Полезен когда необходимо скрыть доступ к интернету
* Динамический – отображает незарегистрированный адрес на зарегистрированный адрес из группы зарегистрированных IP адресов.
* Маскарадный (Перегруженный) – форма динамического НАТ которая отображает несколько не зарегистрированных адресов в единый зарегистрированный IP адрес (т.е. использует порты).

Преимущества:

* Позволяет предотвратить и ограничить обращение снаружи ко внутренним хостам, оставляя возможность обращения изнутри наружу.
* Позволяет скрыть определенные внутренние сервисы определенных внутренних хостов и серверов.

Недостатки:

* При использовании NAT хосты интернет взаимодействуют напрямую с NAT устройствами и не взаимодействуют напрямую с реальными хостами
* Использование NAT усложняет работу администратора
* Не все протоколы могут преодолевать NAT устройства

**Межсетевой экран (сетевой экран)** – это комплекс аппаратных или программных средств осуществляющий контроль и фильтрацию проходящих через него пакетов в соответствии с заданными правилами. (Фаервол и Брандмауэр). При передаче любых пакетов через фаерволы этот пакет проходит несколько стадий проверки. Если он проходит все стадии то пакет передается, если не проходит хотя бы одну, то пакет удаляется. В ISO / OSI находятся на сетевом, сеансовом, и прикладном уровнях. SPI брандмауэры объединяют в своей работе все 3 уровня с их фаерволами.

На сеансовом уровне блокируются TCP соединения. На прикладном уровне брандмауэры отвечают за доступ приложения в сеть, обмен почтовыми сообщениями, и определяют содержимое пакетов.

**ProxyServer**

Это служба в компьютерных сетях, позволяющая клиентам выполнять косвенные запросы к другим сетевым служба. Могут использоваться для обеспечения доступа компьютерам локальной сети в интернет. Кэширование данных. Для сжатия данных, что обеспечивают быструю скорость передачи. Используется для защиты локальной сети от внешнего доступа. На Proxy можно наложить некоторые правила на доступ из локальной сети во внешнюю. Анонимизация доступа к программам и службам.

**РЕМЭЙЛЕРЫ**

Это сервер получающий сообщение электронной почты и перенаправляющий его по адресу указанному отправителем. В процессе переадресации вся информация об отправителе уничтожается, поэтому получатель лишен возможности узнать, кто отправил сообщение. Делятся на анонимные и псевдо анонимные. Псевдо анонимные – сервер знает адрес электронной почты, который необходим для получения ответа на письмо. Анонимные ремэйлеры полностью уничтожают адрес отправителя. При этом обеспечивается очень высокая безопасность, и нет гарантии своевременной доставки.

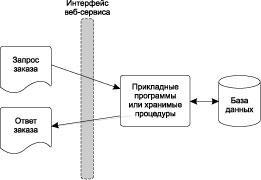
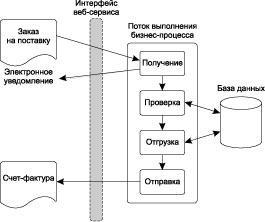
Стандарт MIXMINIOM анонимной перессылки почты, по этому стандарту можно отправлять электронную почту.

**22) Web-сервисы: SOAP, XML, WSDL, UDDI.**

Веб-сервисы преобразуют XML-документы (Extensible Markup Language, XML) в ИТ-системах. Веб-сервисы - это XML-приложения, осуществляющие связывание данных с программами, объектами, базами данных либо с деловыми операциями целиком. Между веб-сервисом и программой осуществляется обмен XML-документами, оформленными в виде сообщений. Стандарты веб-сервисов определяют формат таких сообщений, интерфейс, которому передается сообщение, правила привязки содержания сообщения к реализующему сервис приложению и обратно, а также механизмы публикации и поиска интерфейсов.

Стандарты и технологии веб-сервисов обычно подразумевают два основных типа моделей взаимодействия приложений:

* удаленный вызов процедуры (онлайновая);
* документно-ориентированный (пакетная).

SOAP - это XML-способ определения: какая информация должна пересылаться и как.

SOAP-сообщения содержат конверт, заголовок и тело сообщения. SOAP-сообщения состоят из нескольких основных частей.

* Envelope (конверт) - определяет начало и конец сообщения.
* Header (заголовок) - содержит любые дополнительные атрибуты сообщения, используемые в ходе обработки сообщения как посредником, так и конечным получателем.
* Body (тело сообщения) - содержит XML-данные, передаваемые данным сообщением.
* Attachment (вложение) - состоит из одного и более документов, "прикрепленных" к основному сообщению. (Относится только к SOAP withAttachments ("SOAP с вложениями").)
* RPC interaction (SOAP:RPC-взаимодействие) - определяет, как моделировать взаимодействия RPC-типа.
* Encoding (кодировка) - определяет, как будут представлены простые и сложные данные, передаваемые в сообщении.

Обязательными являются только конверт и тело сообщения.

WSDL - это XML-формат, описывающий состав веб-сервиса. WSDL предназначен для использования как в процедурно-ориентированных, так и в документно-ориентированных приложениях. Так же как и другие XML-технологии, WSDL является расширяемым языком и имеет такое количество параметров, что обеспечение совместимости при организации взаимодействия между различными реализациями может вызвать сложности. Полное взаимопонимание возможно лишь в том случае, если отправитель и получатель сообщения могут совместно использовать и одинаково интерпретировать один и тот же WSDL-файл.

WSDL в соответствии с уровнем абстрагирования состоит из трех элементов. WSDL можно разделить на три основные составляющие:

* определение типов данных;
* абстрактные операции;
* связывание сервисов.

UDDI регистрирует и публикует определения веб-сервисов. Структура UDDI определяет модель данных в программных интерфейсах (API) XML и SOAP для регистрации и обнаружения коммерческой информации, включая веб-сервисы.

SOAP — это стандарт для отсылки и получения сообщений по Internet. Изначально этот протокол был предложен фирмой Microsoft в качестве средства для удаленного вызова процедур (RPC, Remote Procedure Call) по протоколу HTTP, а спецификация SOAP 1.0 (Userland, Microsoft, Developmentor) была тесно связана с Component Object Model. Фирма IBM и ряд других компаний, в том числе Lotus, внесли определенный вклад в развитие этого протокола, и его спецификация была направлена на рассмотрение комитетом W3C.  
  
Спецификация SOAP определяет XML-«конверт» для передачи сообщений, метод для кодирования программных структур данных в формате XML, а также средства связи по протоколу HTTP.  
SOAP-сообщения бывают двух типов: запрос (Request) и ответ (Response). Запрос вызывает метод удаленного объекта, ответ возвращает результат выполнения данного метода.

**23) Национальная инфраструктура информационной безопасности.**

информационная безопасность - состояние защищенности сбалансированных интересов личности, общества и государства от внешних и внутренних угроз в информационной сфере;

**Статья 349. Несанкционированный доступ к компьютерной информации**

1. Несанкционированный доступ к информации, хранящейся в компьютерной системе, сети или на машинных носителях, сопровождающийся нарушением системы защиты (несанкционированный доступ к компьютерной информации), повлекший по неосторожности изменение, уничтожение, блокирование информации или вывод из строя компьютерного оборудования либо причинение иного существенного вреда, –

наказывается штрафом или арестом на срок до шести месяцев.

1. Несанкционированный доступ к компьютерной информации, совершенный из корыстной или иной личной заинтересованности, либо группой лиц по предварительному сговору, либо лицом, имеющим доступ к компьютерной системе или сети, –

наказывается штрафом, или лишением права занимать определенные должности или заниматься определенной деятельностью, или арестом на срок от трех до шести месяцев, или ограничением свободы на срок до двух лет, или лишением свободы на тот же срок.

1. Несанкционированный доступ к компьютерной информации либо самовольное пользование электронной вычислительной техникой, средствами связи компьютеризованной системы, компьютерной сети, повлекшие по неосторожности крушение, аварию, катастрофу, несчастные случаи с людьми, отрицательные изменения в окружающей среде или иные тяжкие последствия, –

наказываются ограничением свободы на срок до пяти лет или лишением свободы на срок до семи лет.

**Статья 350. Модификация компьютерной информации**

1. Изменение информации, хранящейся в компьютерной системе, сети или на машинных носителях, либо внесение заведомо ложной информации, причинившие существенный вред, при отсутствии признаков преступления против собственности (модификация компьютерной информации) –

наказываются штрафом, или лишением права занимать определенные должности или заниматься определенной деятельностью, или арестом на срок от трех до шести месяцев, или ограничением свободы на срок до трех лет, или лишением свободы на тот же срок.

1. Модификация компьютерной информации, сопряженная с несанкционированным доступом к компьютерной системе или сети либо повлекшая по неосторожности последствия, указанные в части третьей статьи 349 настоящего Кодекса, –

наказывается ограничением свободы на срок до пяти лет или лишением свободы на срок до семи лет с лишением права занимать определенные должности или заниматься определенной деятельностью или без лишения.

**Статья 351. Компьютерный саботаж**

1. Умышленные уничтожение, блокирование, приведение в непригодное состояние компьютерной информации или программы, либо вывод из строя компьютерного оборудования, либо разрушение компьютерной системы, сети или машинного носителя (компьютерный саботаж) –

наказываются штрафом, или лишением права занимать определенные должности или заниматься определенной деятельностью, или арестом на срок от трех до шести месяцев, или ограничением свободы на срок до пяти лет, или лишением свободы на срок от одного года до пяти лет.

1. Компьютерный саботаж, сопряженный с несанкционированным доступом к компьютерной системе или сети либо повлекший тяжкие последствия, –

наказывается лишением свободы на срок от трех до десяти лет.

**Статья 352. Неправомерное завладение компьютерной информацией**

Несанкционированное копирование либо иное неправомерное завладение информацией, хранящейся в компьютерной системе, сети или на машинных носителях, либо перехват информации, передаваемой с использованием средств компьютерной связи, повлекшие причинение существенного вреда, –

наказываются общественными работами, или штрафом, или арестом на срок до шести месяцев, или ограничением свободы на срок до двух лет, или лишением свободы на тот же срок.

**Статья 353. Изготовление либо сбыт специальных средств для получения неправомерного доступа к компьютерной системе или сети**

Изготовление с целью сбыта либо сбыт специальных программных или аппаратных средств для получения неправомерного доступа к защищенной компьютерной системе или сети –

наказываются штрафом, или арестом на срок от трех до шести месяцев, или ограничением свободы на срок до двух лет.

**Статья 354. Разработка, использование либо распространение вредоносных программ**

1. Разработка компьютерных программ или внесение изменений в существующие программы с целью несанкционированного уничтожения, блокирования, модификации или копирования информации, хранящейся в компьютерной системе, сети или на машинных носителях, либо разработка специальных вирусных программ, либо заведомое их использование, либо распространение носителей с такими программами –

наказываются штрафом, или арестом на срок от трех до шести месяцев, или ограничением свободы на срок до двух лет, или лишением свободы на тот же срок.

1. Те же действия, повлекшие тяжкие последствия, –

наказываются лишением свободы на срок от трех до десяти лет.

**24) Безопасность в сетях: конфиденциальность, аутентификация, обеспечение строгого выполнения обязательств, авторизация, обеспечение целостности, криптография, криптоанализ, криптология, шифр, код, ключ шифра, IPsec, SSL/TSL, HTTPS, DNSsec.**

• Секретность

• Аутентификация

• Обеспечение строго выполнения обязательств

• Обеспечение целостности

**Конфиденциальность** – предотвращение попадание информации неавторизованным пользователям.

**Аутентификация** – проверка принадлежности субъекту предъявленного им идентификатора, подтверждающего личность. Процесс аутентификации может осуществляться:

* логин и пароль
* электронный сертификат
* смарт-карт
* идентификация личности по биометрическим данным

**Идентификация** – процесс присвоения субъектам идентификатора и сравнение идентификатора с перечнем идентификаторов.

**Авторизация** – процесс проверки прав субъекта на выполнение некоторых действий.

Безопасность охватывает все уровни протоколов:

* 1. На физическом уровне можно поместить сетевой кабель в специальные герметические трубы.
  2. На канальном – аппаратное сжатие, шифрование, перемешивание….
  3. На сетевом – фаервол и брандмауэр
  4. На транспортном – можно поддерживать зашифрованное соединение между процессами
  5. На сеансовом – продолжительность действия ключей
  6. На представительском – методы шифрования
  7. На прикладном - процессы аутентификации.

**Обеспечение целостности(**Целостность информации (также целостность данных) — термин в информатике и теории телекоммуникаций, который означает, что данные полны, условие того, что данные не были изменены при выполнении любой операции над ними, будь то передача, хранение или представление.) данных с помощью хэш-кодов они используются для цифровых подписей.

**Криптография** – наука о методах обеспечения конфиденциальности и аутентичности.

• Криптосистема

• Криптоанализ

• Криптология

• Криптографическая стойкость

**Криптогра́фия**— наука о методах обеспечения конфиденциальности (невозможности прочтения информации посторонним) и аутентичности (целостности и подлинности авторства, а также невозможности отказа от авторства) информации. Изначально криптография изучала методы шифрования информации — обратимого преобразования открытого (исходного) текста на основе секретного алгоритма и/или ключа в шифрованный текст (шифротекст)( Самый эффективный способ борьбы со сниффингом пакетов не предотвращает перехвата и не распознает работу снифферов).

**Криптоанализ**— наука о методах получения исходного значения зашифрованной информации, не имея доступа к секретной информации (ключу), необходимой для этого. В большинстве случаев под этим подразумевается нахождение ключа.

**Криптоло́гия**  — наука, занимающаяся методами шифрования и дешифрования. Криптология состоит из двух частей — криптографии и криптоанализа. А также разработкой методов, позволяющих взламывать криптосистемы.

**Шифр** - какая-либо система преобразования текста (код) для обеспечения секретности передаваемой информации.

**Код** — совокупность алгоритмов криптографических преобразований (шифрования), отображающих множество возможных открытых данных на множество возможных зашифрованных данных, и обратных им преобразований.

**Ключ** — параметр криптографического алгоритма, обеспечивающий выбор одного преобразования из совокупности преобразований, возможных для этого алгоритма. В современной криптографии предполагается, что вся секретность криптографического алгоритма сосредоточена в ключе, но не деталях самого алгоритма.

**IPSec** – это комплекс протоколов касающихся вопросов шифрования, аутентификации и обеспечения защиты при транспортировке IP пакетов. Он включает около 20ти предложений по стандартам и 18ти RFC.

Основными функциями IPSec являются:

* Обеспечение конфиденциальности - отправитель должен иметь возможность шифровать пакеты до их отправки.
* Обеспечение целостности
* Обеспечение защиты от воспроизведения пакетов

Протоколы IPSec:

• IKE – обеспечение аутентификации сторон

• AH – обеспечивает аутентификацию пакетов и выявление их воспроизведение

• ESP – обеспечивает конфиденциальность

• HMAC – механизм аутентификации сообщений с использованием хэш функций

• DES – стандарты шифрования данных.

Существует 2 режима работы:

**Транспортный** – шифруется только информативная часть IP пакета.

**Туннельный** – IP пакет шифруется целиком. IP пакет вкладывается в другой IP пакет.

**SSL/TSL** TLS (что есть Transport Layer Security), он же ранее известный как SSL (Secure Sockets Layer), на данный момент является стандартом де-факто для защиты протоколов транспортного уровня от различных методов вмешательства извне. Много кто его использует.

**HTTPS** (Hypertext Transfer Protocol Secure) — расширение протокола HTTP, поддерживающее шифрование. Данные, передаваемые по протоколу HTTPS, «упаковываются» в криптографический протокол SSL или TLS, тем самым обеспечивается защита этих данных. В отличие от HTTP, для HTTPS по умолчанию используется TCP-порт 443.

**DNSSEC** (англ. Domain Name System Security Extensions) — набор спецификаций IETF, обеспечивающих безопасность информации, предоставляемой средствами DNS в IP-сетях. Он обеспечивает DNS-клиентам аутентификацию данных DNS либо аутентификацию информации о факте отсутствия данных и их целостность. Не обеспечивается доступность данных и конфиденциальность запросов.