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

Физический уровень. Физический уровень 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. На канальном уровне модели рассматривается два подуровня: подуровень управления доступом к среде передачи данных и подуровень управления логическим каналом. Управление доступом к среде передачи данных определяет методы совместного использования сетевыми адаптерами среды передачи данных. Подуровень управления логической связью определяет понятия канала между двумя сетевыми адаптерами, а также способы обнаружения и исправления ошибок передачи данных. Основное назначение процедур канального уровня подготовить блок данных (обычно называемый кадром) для следующего сетевого уровня.

Здесь следует отметить два момента:

1) начиная с подуровня управления логической связью и выше протоколы никак не зависят от среды передачи данных;

2) для организации локальной сети достаточно только физического и канального уровней, но такая сеть не будет масштабируемой (не сможет расширяться), т.к. имеет ограниченные возможности адресации и не имеет функций маршрутизации.

Сетевой уровень. Сетевой уровень определяет методы адресации и маршрутизации компьютеров в сети. В отличие от канального уровня сетевой уровень определяет единый метод адресации для всех компьютеров в сети не зависимого от способа передачи данных. На этом уровне определяются способы соединения компьютерных сетей. Результатом процедур сетевого уровня является пакет, который обрабатывается процедурами транспортного уровня.

Транспортный уровень. Основным назначением процедур транспортного уровня является подготовка и доставка пакетов данных между конечными точками без ошибок и в правильной последовательности. Процедуры транспортного уровня формируют файлы для сеансового уровня из пакетов, полученных от сетевого уровня.

Сеансовый уровень. Сеансовый уровень определят способы установки и разрыва соединений (называемых сеансами) двух приложений, работающих в сети.

Следует отметить, что сеансовый уровень - это точка взаимодействия программ и компьютерной сети.

Представительский уровень. На представительский уровне определяется формат данных, используемых приложениями. Процедуры этого уровня описывают способы шифрования, сжатия и преобразования наборов символов данных.

Прикладной уровень. Основное назначения уровня: определить способы взаимодействия пользователей с системой (определить интерфейс).

Структура TCP/IP является более простой: в ней выделяются Прикладной, Транспортный, Межсетевой и уровень доступа к сети. В общем и целом, Транспортные уровни обеих моделей соответствуют друг другу, но есть и некоторые различия. Например, некоторые функции Сеансового уровня модели ISO/OSI берет на себя Транспортный уровень TCP/IP. Содержимое Сетевого уровня модели ISO/OSI тоже примерно соответствует Межсетевому уровню TCP/IP. В большей или меньшей степени Прикладной уровень TCP/IP соответствует трем уровням Сетевому, Представительскому и Прикладному модели ISO/OSI. Уровень доступа к сети – совокупности Физического и Канального уровней.

Транспортный уровень - TCP (Transmission Control Protocol), чье имя присутствует в названии всего стека; **UDP** (User Datagram Protocol) – протокол передачи дейтаграмм пользователя.

Межсетевой уровень - IP (Internet Protocol), ***ICMP*** (Internet Control Message Protocol), ***ARP*** (Address Resolution Protocol), ***RARP*** (Reverse ARP).

Уровень доступа к сети - Ethernet, SLIP (Serial Line IP)**, PPP** (Point-to-Point Protocol)**.**

Наиболее популярной архитектурой для распределенного программного приложения является ***архитектура клиент-сервер***. Будем говорить, что распределенное приложение имеет архитектуру клиент-сервер, если все процессы распределенного приложения можно условно разбить на две группы. Одна группа процессов называется серверами другая – клиентами. Обмен данными осуществляется только между процессами-клиентами и процессами-серверами. Основное отличие процесса-клиента от процесса-сервера в том, что инициатором обмена данными всегда является процесс-клиент. Другими словами процесс-клиент обращается за услугой (сервисом) к процессу-серверу. Такая архитектура лежит в основе большинства современных информационных систем.

NDIS - спецификация интерфейса сетевого драйвера для сопряжения драйверов сетевых адаптеров с операционной системой.

NDIS — это специальный драйвер, который содержит функции, используемые низкоуровневыми сетевыми драйверами. NDIS как бы обволакивает низкоуровневые сетевые драйверы и является посредником в их общении между собой и с железом.

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.

Структура TCP/IP является более простой: в ней не выделяются Физический, Канальный, Сетевой и Представительский уровни. В общем и целом Транспортные уровни обеих моделей соответствуют друг другу, но есть и некоторые различия. Например, некоторые функции Сеансового уровня модели ISO/OSI берет на себя Транспортный уровень TCP/IP. Содержимое Сетевого уровня модели ISO/OSI тоже примерно соответствует Межсетевому уровню TCP/IP. В большей или меньшей степени Прикладной уровень TCP/IP соответствует трем уровням Сетевому, Представительскому и Прикладному модели ISO/OSI, а Уровень доступа к сети – совокупности Физического и Канального уровней.

Транспортный уровень - TCP (Transmission Control Protocol), чье имя присутствует в названии всего стека; **UDP** (User Datagram Protocol) – протокол передачи дейтаграмм пользователя.

Межсетевой уровень - IP (Internet Protocol), ***ICMP*** (Internet Control Message Protocol), ***ARP*** (Address Resolution Protocol), ***RARP*** (Reverse ARP).

Уровень доступа к сети - Ethernet, SLIP (Serial Line IP)**, PPP** (Point-to-Point Protocol)**.**

Публичные "белые" адреса используются в сети Интернет. Публичным IP-адресом называется IP-адрес, который используется для выхода в Интернет. Доступ к устройству с публичным IP-адресом можно получить из любой точки глобальной сети, т.к. публичные (глобальные) IP-адреса маршрутизируются в Интернете, в отличие от частных (серых) IP-адресов.

Однако, в связи с тем что "белых" адресов существует ограниченное количество, в локальных сетях используются частные (серые) IP-адреса. Они не маршрутизируются в Интернете и на них нельзя отправить трафик из Интернета, только из конкретной частной локальной

сети. Для домашних пользователей провайдер может предоставлять один или несколько публичных IP-адресов (как правило, это платная услуга). Маршрутизатор позволяет компьютерам локальной сети (с частными IP-адресами) использовать для выхода в Интернет публичный IP-адрес, установленный на внешнем интерфейсе (WAN) устройства. Именно этот внешний IP-адрес может быть использован для доступа из Интернета в локальную сеть.

Частные IP:
10.0.0.0 – 10.255.255.255 (префикс 10/8)
172.16.0.0 – 172.31.255.255 (префикс 172.16/12)
192.168.0.0 – 192.168.255.255 (префикс 192.168/16)

Параметры: IP, маска, шлюз, DNS

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.

Протокол Ethernet. Термин Ethernet обычно связывают со стандартом опубликованным в 1982 г. совместно корпорациями DEC, Intel и Xerox (DIX).

На сегодняшний день это наиболее распространенная технология локальных сетей. Ethernet применяет метод доступа CSMA/CD , использует 48-битную адресацию (стандарт IEE EUI-64) и обеспечивает передачу данных до 1 гигабита в секунду. Максимальная длина кадра, передаваемая в сети Ethernet составляет 1518 байт, при этом сами данные могут занимать от 46 до 1500 байт. Физически функции протокола Ethernet реализуются сетевой картой (***NIC***, Network Interface Card), которая может быть подключена к кабельной системе с фиксированным MAC-адресом производителя (в соответствии со стандартом ICANN).

Протокол SLIP(Serial Line IP). Аббревиатурой SLIP обозначают межсетевой протокол для последовательного канала. Раньше SLIP использовался для подключения домашних компьютеров к Internet через последовательный порт RS-232. Протокол использует простейшую инкапсуляцию кадра и 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 CSLIP (Compressed SLIP).

Протокол PPP (Point-to-Point Protocol). PPP – универсальный протокол двухточечного соединения: поддерживается TCP/IP, NetBEUI, IPX/SPX, DECNet и многими стеками протоколов. Протокол может применяться для технологии ISDN (Integrated Services Digital Network) и SONET (Synchronous Optical Network). 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. В настоящее время это наиболее широко используемый протокол для последовательного канала, обеспечивающий соединение компьютера с сетью Internet и практически вытеснил протокол SLIP.

Протокол IPv4. В семействе протоколов TCP/IP протоколу IP отведена центральная роль. Его основной задачей является доставка ***дейтограмм*** (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.

Протокол IPv6. 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 MAC-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 содержит 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 половинами. 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***. Большое внимание в новой версии протокола уделяется вопросам безопасности

Протокол ICMP. Спецификация протокола ICMP (Протокол контроля сообщений в Internet) изложена в документе RFC 792. ICMP является неотъемлемой частью TCP/IP и предназначен для транспортировки информации о сетевой деятельности и маршрутизации. IPCM сообщения представляют собой специально отформатированные IP-дейтаграммы, которым соответствуют определенные типы (15 типов) и коды сообщений. Описание типов и кодов IPCM-сообщений содержится в [5]. С помощью протокола ICMP 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.

Протокол ARP. IP-адреса могут восприниматься только на сетевом уровне и вышестоящих уровнях TCP/IP. На канальном уровне всегда действует другая схема адресации, которая зависит от используемого протокола. Например, в сетях Ethernet используются 48-битные адреса. Для установки соответствия между 32-разрядными IP-адресами и теми или иными MAC-адресами, действующими на канальном уровне, применяется механизм привязки адресов по протоколу ARP, спецификация которого приведена в документе RFC 826. Основной задачей ARP является динамическая (без вмешательства администратора, пользователя, прикладной программы) проекция IP-адресов в соответствующие MAC-адреса аппаратных средств. Эффективность работы ARP обеспечивается тем, что каждый хост кэширует специальную ARP-таблицу. Время существования записи в этой таблице составляет обычно 10- 20 минут с момента ее создания и может быть изменено с помощью параметров реестра [6]. Просмотреть текущее состояние ARP-таблицы можно с помощью команда ***arp***. Кроме того, протокол ARP используется для проверки существования в сети дублированного IP-адреса и разрешения запроса о собственном MAC-адресе хоста во время начальной загрузки.

Протокол RARP. Протокол RARP, как следует из названия (Reverse ARP), по своей функции противоположен протоколу ARP. RARP применятся для получения IP-адреса по MAC-

адресу. В настоящее время протокол заменен на протокол Прикладного уровня DHCP, предлагающий более гибкий метод присвоение адресов.

Протокол TCP. Протокол TCP является ***надежным*** байт-ориентированным протоколом с ***установлением соединения***. При получении дейтаграммы, в поле Protocol (со структурой IP-дейтаграммы можно ознакомиться в [5,6]) которой 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 (структура TCP-сегмента приведена в [5,6]), 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. Для выполнения этих задач все октеты в потоке данных пронумерованы в возрастающем порядке. Заголовок каждого сегмента содержит число октетов и порядковый номер первого октета данных в данном сегменте. Каждый сегмент данных сопровождается контрольной суммой, позволяющей обнаружить повреждение данных. При отправлении некоторого числа последовательных октетов данных, отправитель ожидает подтверждение приема. Если подтверждения не приходит, то предполагается, что группа октетов не дошла по назначению или была повреждена – в этом случае предпринимается повторная попытка переслать данные.

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

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 сумы 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 UDP и TCP является, то, что UDP – протокол без установления соединения (ориентированным на сообщения), а TCP – протокол на основе соединения (ориентированный на поток).

4. Понятие сокета. Основные параметры сокета.

Совокупность IP-адреса и номера порта называется ***сокетом***. Сокет однозначно идентифицирует прикладной процесс в сети TCP/IP.

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 операционной системы, 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	Разрешить подключение к сокету
bind	Связать сокет с параметрами
closesocket	Закрыть существующий сокет
connect	Установить соединение с сокетом
gethostbyaddr	Получить имя хоста по его адресу
gethostbyname	Получить адрес хоста по его имени
gethostname	Получить имя хоста
getsockopt	Получить текущие опции сокета
inet_addr	Преобразовать символьное представление IPv4-адреса в формат TCP/IP
inet_ntoa	Преобразовать сетевое представление IPv4-адреса в символьный формат
ioctlsocket	Установить режим ввода-вывода сокета
listen	Переключить сокет в режим прослушивания
recv	Принять данные по установленному каналу
recvfrom	Принять сообщение
send	Отправить данные по установленному каналу
sendto	Отправить сообщение
setsockopt	Установит опции сокета
socket	Создать сокет
WSACleanup	Завершить использование библиотеки WS2_32.DLL
WSAGetLastError	Получить диагностирующий код ошибки
WSAStartup	Инициализировать библиотеку WS2_32.DLL

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

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_STREEM** – сокет ориентированный на поток;
- старший номер версии;

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

- для TCP/IP можно указать NULL

5. Интерфейс Nimed Pipe.

Именованным каналом называется объект ядра операционной системы, который обеспечивает обмен данными между процессами, выполняющимися на компьютерах в одной локальной сети. Процесс, создающий именованный канал, называется ***сервером именованного канала***. Процессы, которые связываются с именованным каналом, называются ***клиентами именованного канала***. Любой именованный канал идентифицируется своим именем, которое задается при создании канала.

Именованные каналы бывают: ***дуплексные*** (позволяющие передавать данные в обе стороны) и ***полудуплексны***е (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***.

Для использования функций интерфейса Named Pipe в программе на языке C++ необходимо включить в ее текст заголовочный файл Windows.h. Сами функции интерфейса располагаются в библиотеке KERNEL32.DLL ядра операционной системы.

Все функции Named Pipe API 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.

Наименование функции	Назначение
CallNamedPipe	Выполнить одну транзакцию
ConnectNamedPipe	Соединить сервер с каналом
CreateFile	Открыть канал
CreateNamedPipe	Создать именованный канал
DisconnectNamedPipe	Закончить обмен данными
GetNamedPipeHandleState	Получить состояние канала
GetNamedPipeInfo	Получить информацию об именованном канале
PeekNamedPipe	Копировать данные канала
ReadFile	Читать данные из канала
SetNamedPipeHandleState	Изменить характеристики канала
TrasactNamedPipe	Писать и читать данные канала
WaitNamedPipe	Определить доступность канала
WriteFile	Писать данные в канал

6. Интерфейс MailSlot.

Почтовым ящиком (Mailslot) называется объект ядра операционной системы, который обеспечивает передачу данных от процессов-клиентов к процессам-серверам, выполняющимся на компьютерах в одной локальной сети. Процесс, создающий почтовый ящик называется ***сервером почтового ящика***. Процессы, которые связываются с почтовым ящиком, называются ***клиентами почтового*** ящика.

Каждый почтовый ящик имеет имя, которое определяется сервером при создании и используется клиентами для доступа. Передача может осуществляться только сообщениями и в одном направлении – от клиента к серверу. Обмен данными может происходить в синхронном и асинхронном режимах. Допускается создание нескольких серверов с одинаковым именем почтового ящика – в этом случае все отправляемые клиентом сообщения будут поступать во все почтовые ящики, имеющие имя, указанное клиентом. Однако, следует сказать, что такая рассылка сообщений возможна только в том случае, когда длина оправляемых сообщений не превышает 425 байт.

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

Наименование функции	Назначение
CreateFile	Открыть почтовый ящик
CreateMailslot	Создать почтовый ящик
GetMailslotInfo	Получить информацию о почтовом ящике
ReadFile	Читать данные из почтового ящика
SetMailslotInfo	Изменить время ожидания сообщения
WriteFile	Писать данные в почтовый ящик

7. Структура программы TCP-сервера.
8. Структура программы TCP-клиента.

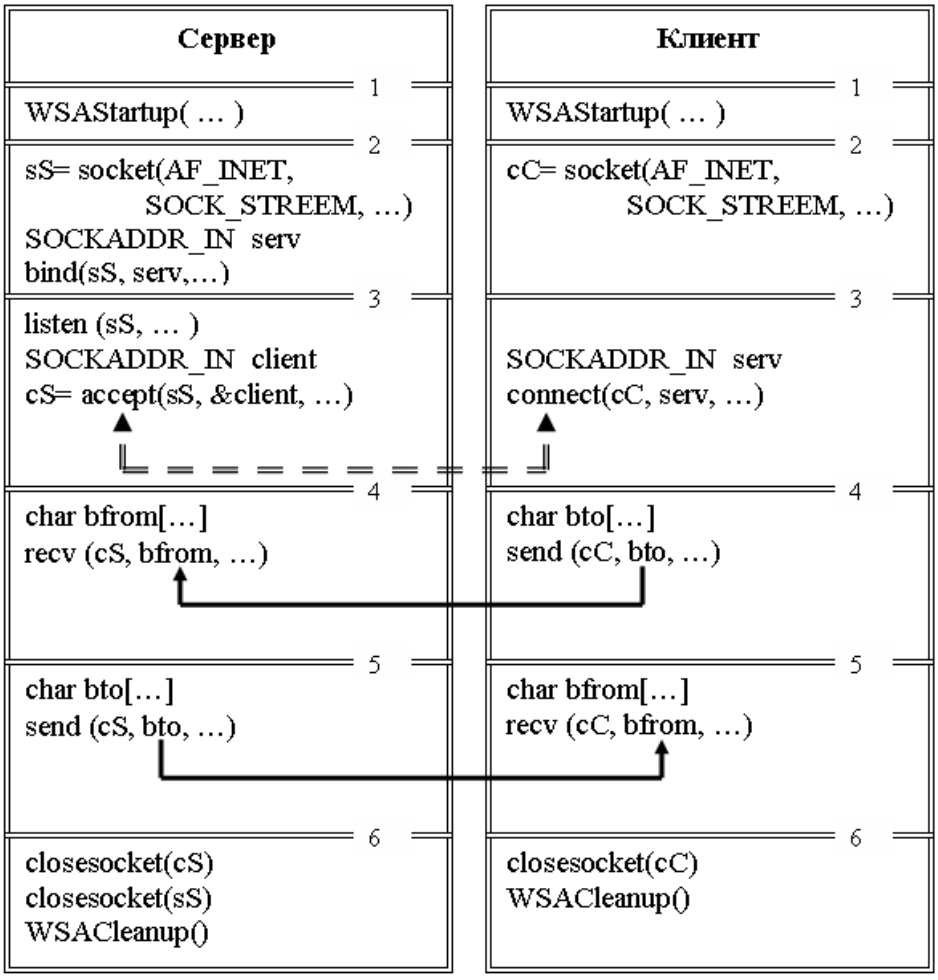
Первые блоки обеих программ идентичны и предназначены для инициализации библиотеки WS2_32.DLL.

Второй блок сервера имеет то же предназначение, 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).

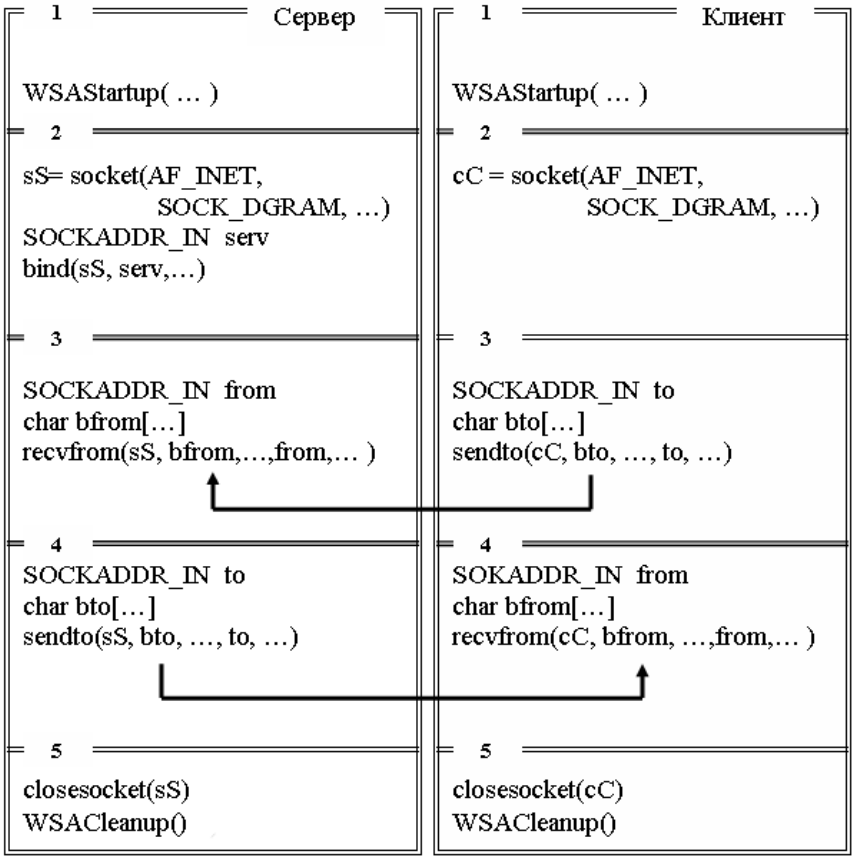
В третьем блоке программы сервера выполняются две функции Winsock2: listen и accept. Функция listen переводит сокет, ориентированный на поток, в состояния прослушивания (открывает доступ к сокету) и задает некоторые параметры очереди соединений. Функция accept переводит процесс сервера в состояние ожидания, 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, используются функции send и recv, а во-вторых, в качестве параметра эти функции используют сокет, созданный командой accept.

В программе клиента осталось пояснить, только работу третьего блока. В этом блоке выполняется функция connect, предназначенная для установки соединения с сокетом сервера. Функция в качестве параметров имеет, созданный в предыдущем блоке, дескриптор сокета (ориентированного на поток) и структуру SOCKADDR_IN с параметрами сокета сервера.



9. Структура программы UDP-сервера.
10. Структура программы UDP-клиента.



Первые блоки обеих программ одинаковые и предназначены для инициализации библиотеки WS2_32.DLL.

Второй блок программы сервера создает сокет (функция socket) и устанавливает параметры этого сокета. Следует обратить внимание на параметр SOCK_DGRAM функции socket, указывающий на тип сокета (в данном случае – сокет, ориентированный на сообщения). Для установки параметров сокета, используется функция bind. При этом говорят, что сокет ***связывают*** с параметрами. Для хранения параметров сокета в Winsock2 предусмотрена специальная структура SOCKADDR_IN (она тоже присутствует на рисунке). Перед выполнением функции bind, которая использует эту структуру в качестве параметра, необходимо ее заполнить данными. Пока скажем только, 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 (на рисунке символьный массив bfrom). Следует отметить, что 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 использует структуру SOCKADDR_IN с параметрами сокета принимающей стороны (в данном случае клиента) и заполненный буфер с данными.

Пятые блоки программ сервера и клиента одинаковые и предназначены для закрытия сокета и завершения работы с библиотекой WS2_32.DLL.

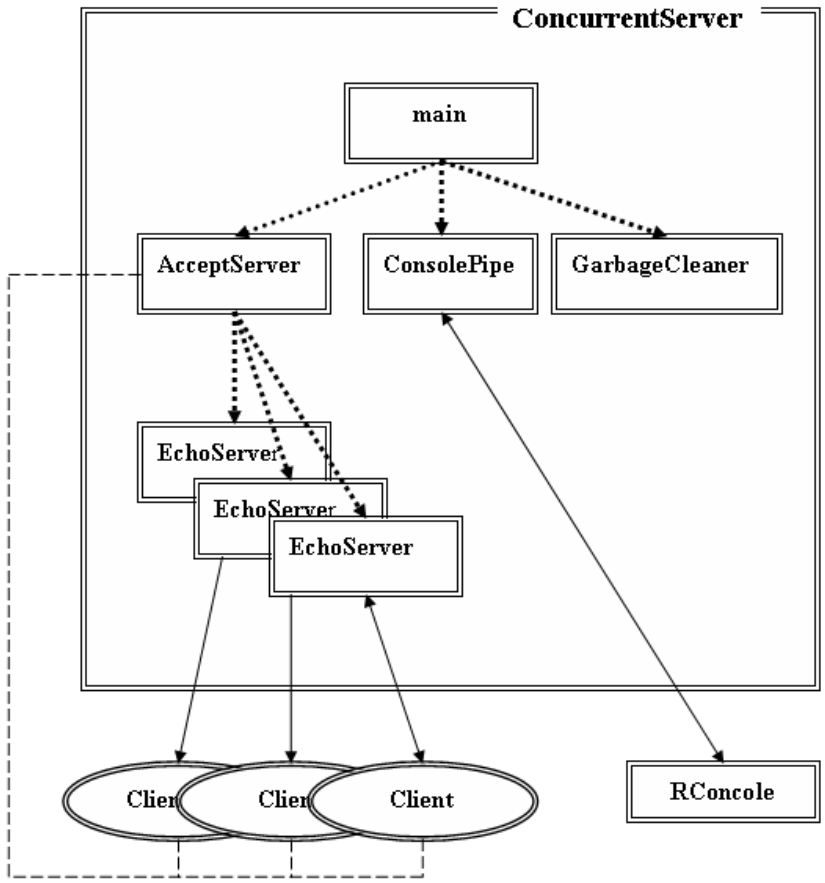
Всем блокам программы клиента, кроме второго, есть аналог в программе сервера. Второй блок, в сравнении с сервером, не использует команду bind. Здесь проявляется основное отличие между сервером и клиентом. Если сервер, должен использовать однозначно определенные параметры (IP-адрес и номер порта), то 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. Структура параллельного сервера.

12. Структура параллельного сервера. Accept Server. GarbageCleaner.

Структура параллельного сервера, зависит, в конечном счете, от характера решаемой сервером задачи, но все же существуют общие структурные свойства сервера, на которых следует остановиться. Для того, чтобы упростить изложение будем рассматривать далее конкретную реализацию (модель) параллельного сервера с именем ConcurrentServer.



Назначение сервера является, одновременное обслуживание нескольких клиентских программ. Обслуживание заключается в получении от клиента по установленному TCP-соединению последовательности символов и в возврате (пересылке) этой последовательности обратно. Кроме того, предполагается, что сервер может выполнять команды, введенные с консоли управления, с которой поддерживается связь через именованный канал (Named Pipe).

Процесс main. Основным назначением процесса main, является запуск, инициализация и завершение работы сервера. Как уже отмечалось, именно этот процесс первым получает управление от операционной системы. Процесс main запускает основные процессы: AcceptServer, ConsolePipe и RConsole.

Процесс AcceptServer. 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 далее будем называть ***ListContanct***. При подключении очередного клиента, процесс AcceptServer добавляет в ListContact элемент, предназначенный для хранения информации о состоянии данного подключения.

Процесс ConsolePipe. ConsolePipe создается процессом main и является сервером именованного канала, по которому осуществляется связь между программой RConsole (консоль управления сервером) и параллельным сервером.

Процесс GarbageCleaner. Основным назначением процесса GarbageCleaner является удаление элемента списка подключений ListContact, после отключения программы клиента. Следует сразу отметить, что ListContact является ресурсом, требующим последовательного использование. Одновременная запись и (или) удаление элементов списка может привести к разрушению списка ListContact.

Процесс EchoServer. Процессы EchoServer создаются 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).

Программа Client. Программа 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.

Программа RConsole. Программа RConsole предназначена для ввода команд управления сервером и для вывода диагностических сообщений полученных от сервера . RConsole является клиентом именованного канала.

Список подключений ListContact. Список ListContact (не изображен на рисунке) создается 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.

13. Широковещание. Обнаружение сервера с помощью широковещания.

Для обеспечения независимости приложения от параметров сокета сервера (сетевой адрес и номера порта), как правило, номер порта делают одним из параметров инициализации сервера и хранят в специальных конфигурационных файлах, которые считывается 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 сервера на само клиентское приложение (при условии, что номер порта сервера известен). 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, которая определена в Winsock2.h. По умолчанию использование стандартного широковещательного адреса не допускается и для его применения необходимо установить специальный режим использования сокета SO_BROADCAST с помощью функции setsockopt. Проверить установленные для сокета режимы можно с помощью функции getsockopt (описание здесь не приводится).

14. Применение символического адреса хоста.

При наличии специальной службы 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 (рисунок 3.16.1). При этом предполагается, что известно символическое имя компьютера, на котором находится программа сервера.

Такое решение достаточно часто применяется разработчиками распределенных систем. Связав набор программ-серверов с определенными стандартными именами компьютеров, распределенное приложение становится не зависимым от адресации в сети. Естественно при этом необходимо позаботиться, 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 противоположно**:** получение символического имени компьютера по сетевому адресу. Обе функции используют структуру hosten, содержащуюся в Winsock2.h.

Если с помощью функции gethostbyname получить адрес компьютера с именем localhost, то в будет собственный получен IP-адрес компьютера или адрес INADDR_LOOPBACK.

Кроме того, для получения действительного собственного имени компьютера (NetBIOS-имени или DNS-имени) можно использовать функцию gethostname.

15. Основные сетевые утилиты и их назначение.

Утилита ping. Как уже отмечалось раньше, ping в своей работе использует протокол ICMP и предназначена для проверки соединения с удаленным хостом.. Проверка соединения осуществляется путем посылки в адрес хоста специальных ICMP-пакетов, которые в соответствии с протоколом должны быть возвращены, отправляющему хосту (эхо-пакеты и эхо-ответы).

Для получения справки о параметрах утилиты ping следует выполнить команду ping без параметров. В простейшем случае команда может быть применена с одним параметром:

– **ping** hostname, где hostname – NetBIOS или DNS - имя хоста или его IP-адрес.

Утилит tracert. Как и утилита ping, tracert использует ICMP протокол для определения маршрута до пункта назначения. В результате работы утилиты на консоль выводятся все промежуточные узлы маршрута от исходного хоста до пункта назначения и время их прохождения.

Для получения справки о параметрах утилиты tracert следует выполнить команду tracert без параметров. В простейшем случае команда может быть применена с одним параметром:

tracert hostname, где hostname – NetBIOS или DNS - имя хоста или его IP-адрес.

Утилита route. Утилита route позволяет манипулировать таблицей сетевых маршрутов, которая имеется на каждом компьютере с TCP/IP-интерфейсом. Утилита обеспечивает выполнение четырех команд: print (распечатка таблицы сетевых маршрутов), add (добавить маршрут в таблицу), change (изменение существующего маршрута), delete (удаление маршрута).

Для получения справки о параметрах утилиты route следует выполнить команду route без параметров. В простейшем случае команда может быть использована для распечатки таблицы сетевых маршрутов:

route print, где параметр (команда) print, без уточняющих операндов, указывает на необходимость распечатки всей таблицы.

Утилита netstat. Утилита отражает состояние текущих TCP/IP-соединений хоста, а также статистику работы протоколов. С помощью утилиты netstat можно распечатать номера ожидающих портов всех соединений TCP/IP, имена исполняемых файлов, участвующих в подключениях, идентификаторы соответствующих Windows-процессов и т.д.

Активные соединения TCP/IP на компьютере можно просмотреть, набрав на консоли команду nestat с параметром -a.

netstat -a

Утилита arp. Утилита используется для просмотра и модификации ARP-таблицы, используемой для трансляции IP-адресов в адреса протоколов канального уровня (MAC-адреса). С помощью параметров команды можно распечатывать таблицу, удалять и добавлять данные ARP-таблицы. Корректировку ARP-таблицы может осуществлять только пользователь справами администратора.

Для получения справки о параметрах утилиты arp, следует выполнить команду arp без параметров. Получить текущее состояние ARP-таблицы можно с помощью следующей команды.

arp -a

Утилита nslookup. Утилита nslookup предназначена для проверки правильности работы DNS-серверов. С помощью утилиты, пользователь может выполнять запросы к DNS-серверам на получение адреса хоста по его DNS-имени, на получение адресов и имен почтовых серверов, ответственных за доставку почты для отдельных доменов DNS, на получение почтового адреса администратора DNS-сервера и т.д. и т.п. Утилита работает в двух режимах: в режиме однократного выполнения (при запуске в командной строке задается полный набор параметров) и в интерактивном режиме (команды и параметры задаются в режиме диалога).

Запуск утилиты в интерактивном режиме осуществляется запуском команды nslookup без параметров.

Утилита hostname. Утилита предназначена для вывода на консоль имени хоста, на котором выполняется данная команда. Команда hostname не имеет никаких параметров.

Утилита ipconfig. Утилита ipconfig является наиболее востребованной сетевой утилитой. С ее помощью можно определить конфигурацию IP-интерфейса и значения всех сетевых параметров. Особенно эта утилита полезна на компьютерах, работающих с протоколом DHCP: команда позволяет проверить параметры IP-интерфейсов установленные в автоматическом режиме.

Для получения справки о параметрах утилиты следует ввести следующую команду.

Короткий отчет о конфигурации TCP/IP можно получить выдав команду ipconfig без параметров. Для получения полного отчета, можно использовать ключ /all.

Утилита nbtstat. Утилита nbtstat позволяет просматривать статистику текущих соединений, использующих протокол NBT (NetBIOS over TCP/IP). Утилита в чем-то подобна утилите netstat, но применительно к протоколу NBT. Для получения справки о параметрах команды, необходимо ее выполнить без указания параметров.

Утилита net. Утилита net является основным средством управления сетью для сетевого клиента Windows. Команду net часто включают в скрипты регистрации и командные файлы. С помощью этой команды можно зарегистрировать пользователя в рабочей группе Windows, можно осуществить выход из сети, запустить или остановить сетевой сервис, управлять списком имен, пересылать сообщения в сети, синхронизировать время и т.д.

Для вывода списка параметров (команд) утилиты net следует выполнить следующую команду.

net help

Справка может быть уточнена для каждого отдельного параметра команды. Например, для того, чтобы получить справку для параметра send (пересылка сообщений в сети) следует добавить соответствующий параметр.

net help send

16. Служба DNS.

Служба ***DNS (Domain Name System)*** является одной из важнейших служб TCP/IP, само появление которой в 1980-х годах дало мощный толчок развитию TCP/IP и всемирной сети Internet. Дело в том, что DNS обеспечивает важную возможность преобразования символических доменных имен в соответствующие IP-адреса (***разрешение имен***). Например, для обращения к адресу серверу компании Microsoft, имеющему IP-адрес 207.46.230.229, можно обратиться, используя символическое имя microsoft.com. С одной стороны, это дает более наглядную нотацию, а с другой, появляется возможность не привязывать жестко получение услуг сервера к фиксированному адресу, который при реорганизации сети может измениться.

Службу DNS можно рассматривать, как распределенную иерархическую базу данных, основное назначение которой отвечать на два вида запросов: выдать IP-адрес по символическому имени хоста и наоборот – выдать символическое имя хоста по его IP-адресу. Облуживание этих запросов и поддержку базы данных в актуальном состоянии обеспечивают взаимодействующие глобально рассредоточенные в сети Internet серверы DNS. База данных имеет древовидную структуру, в корне которой нет ничего, а сразу под корнем находятся ***первичные*** сегменты ***(домены)***: **.com, .edu, .gov, …, .ru, .by, .uk, …** Наименование этих первичных доменов отражает деление базы данных DNS по отраслевому (домены, обозначенные трехбуквенным кодом) и национальному признакам (двухбуквенные домены в соответствии со стандартом ISO 3166). ***Доменом*** в терминологии DNS называется любое поддерево дерева базы данных DNS.

DNS-серверы, обеспечивающие работоспособность всей глобальной службы, тоже имеют древовидную структуру подчиненности, которая соответствует структуре распределенной базы данных. По своему функциональному назначению DNS-серверы бывают: первичные серверы (которые являются главными серверами, поддерживающими свою часть базы данных DNS), вторичные серверы (всегда привязан к некоторому первичному серверу и используются для дублирования данных первичного сервера), кэширующие серверы (обеспечивают хранение недавно используемых записей из других доменов и служат для увеличения скорости обработки запросов на разрешение имен).

При обработке запроса на разрешение имени, хост, как правило, обращается к первичному или вторичному DNS-серверу, обслуживающему данный домен сети. В зависимости от сложности запроса, DNS-сервер может сам ответить на запрос или переадресовать к другому серверу DNS. Последней инстанцией в разрешении имен являются пятнадцать (на сегодняшний день) корневых серверов имен, представляющих собой вершину всемирной иерархии DNS.

Разработчик приложения может обратиться за разрешением имени с помощью функций, имеющих, как правило, имена ***gethostbyname*** и ***gethostbyaddr***.

В одной доменной зоне может быть несколько доменных имен.

Существует определенный ряд записей, почтовый адрес админа, записи сопоставления имени и IPадреса, для замены одного имени на другое, для указания зоны, для поиска серверов где функционируют определенные службы.

Запись SOA: в ней указывается имя доменной зоны, период обновления, Ip адрес администратора, задержка перед следующей попыткой, минимальное время жизни

Запись A: связывает доменное имя и ip адрес. Если создать несколько записей с одним и тем же доменным именем но разными Ip адресами, то это позволит равномерно распределить нагрузки между несколькими хостами. Сервер выбирает ближайший IP адрес к клиенту.

Запись CNAME: позволяет одному IP адресу сопоставить несколько доменных имен.

Запись MX: связывает имена почтовых доменов с именами хостов.

Запись PTR: запись обратная записи A.

Запись NS: определяет полномочный сервер имен субдомена. Связывает имена субдомена с именем полномочного DNS сервера.

Запись SRV: отображает порядок и доступность служб в конкретной зоне. Определяет сетевой протокол, порт, доменное имя хоста.

Формат пакета DNS.

<table>
<tr><td colspan="8">ID</td></tr>
<tr><td>QR</td><td>OPCODE</td><td>AA</td><td>TC</td><td>RD</td><td>RA</td><td>Z</td><td>RCODE</td></tr>
<tr><td colspan="8">QDCOUNT</td></tr>
<tr><td colspan="8">ANCOUNT</td></tr>
<tr><td colspan="8">NSCOUNT</td></tr>
<tr><td colspan="8">ARCOUNT</td></tr>
</table>

ID – идентификатор. Состоит из 2х байт. Т.к. обмен данными носит диалоговый характер, каждый запрос идентифицируется. Клиент передавая запрос заполняет это поля, сервер генерирует ответ и помещает его в тоже поле.

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 ответе в том случае если сервер не смог поместить всю необходимую информацию в пакет из-за существующих ограничений.

RD – заголовок DNS. Клиент просит сервера 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.

Следующие 3 бита заголовка DNS Зарезервированы для дальнейшего использования и должны быть равны нулю во всех запросах и ответах.

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

Утилита NSLOOKUP

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 сети, адрес локального шлюза, адреса серверов DNS и т.п.) хостам. Применение DHCP дает возможность отказаться от фиксированных IP-адресов в зоне действия сервера DHCP. Описание протокола DHCP содержится в документах: RFC 1534, 2131, 2132, 2141.

Конструктивно служба DHCP состоит из трех модулей: ***сервера DHCP (DHCP Server)***, ***клиента DHCP (DHCP Client)*** и ***ретранслятора DHCP (DHCP Replay Agent)*** .

DHCP-серверы способны управлять одним или несколькими диапазонами IP-адресов (***адресными пулами***). В пределах одного пула можно всегда выделить адреса, которые не должны распределяться между хостами. DHCP-серверы используют для приема запросов от DHCP-клиентов порт 67. Выделение IP-адресов может быть трех типов: ***ручной***, ***автоматический*** и ***динамический*** [5,6]. Обычно DHCP-серверы устанавливают на компьютерах, исполняющих роль сервера в сети.

DHCP-клиенты представляет собой программный компонент, обычно реализуемый как часть стека протоколов TCP/IP и предназначен для формирования и пересылки запросов к DHCP-серверу на выделение IP-адреса, продления срока аренды IP-адреса и т.п. DHCP- клиенты используют для приема сообщений от DHCP-сервера порт 68.

Логика работы протокола DHCP достаточно проста. При физическом подключении к сети, хост пытается подсоединиться к сети, используя для этого DHCP-клиент. Для обнаружения DHCP-сервера DHCP-клиент выдает в сеть широковещательный запрос (это процесс называется ***DHCP-поиском***). 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 (***аренда адреса***). 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.

Ретранслятор 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. Ретранслятор в этом случае играет роль посредника между DHCP-клиентом и DHCP-сервером.

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 два 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, код ASCII - 32) и горизонтальной табуляции.

Для удобства и с учетом ограничения размеров строки (998/78 символов), значение поля может быть разбито на несколько строк; это называется «фальцовкой» (folding).

Например, поле заголовка

Subject: This is a test

можно записать в форме

Subject: This

is a test

Процесс преобразования фальцованного многострочного представления поля в обычное однострочное называется расфальцовкой (unfolding) и выполняется путем простого удаления всех последовательностей CRLF, непосредственно за которыми следуют пробельные символы (WSP).

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

quoted-pair = ("\" (VCHAR / WSP)) / obs-qp

Некоторые конструкции в теле структурированных полей заголовков представляют собой просто строки некоторых базовых символов. Такие конструкции называют атомами.

Поле addr-spec представляет собой специфический для Internet идентификатор, содержащий локально интерпретируемую строку, за которой следует символ @ (код ASCII - 64) и доменное имя Internet. Эта локально интерпретируемая строка представляет собой строку в кавычках или атом с точкой

addr-spec = local-part "@" domain

```
message     =  (fields / obs-fields)
               [CRLF body]
body        =  (*(*998text CRLF) *998text) / obs-body
text        =  %d1-9 /          ; Символы, за исключением CR и LF
               %d11 /
               %d12 /
```

19. Почтовая служба. Протоколы SMTP, POP3.

Протоколы Прикладного уровня ***SMTP (Simple Mail Transport Protocol)***, ***POP3 (Post Office Protocol)*** и ***IMAP4 (Internet Message Access Protocol)*** являются основой для создания современной электронной почты.

Основными компонентами системы электронной почты являются: ***MTA (Mail Transport Agent)***, ***MDA (Mail Delivery Agent)***, ***POA (Post Office Agent)*** и ***MUA (Mail User Agent)***.

MTA – транспортный агент, основное назначение которого: прием почтовых сообщений от пользовательских машин; отправка почтовых сообщений другим MTA (установленных на других почтовых системах); прием сообщений от других MTA; вызов MDA. Это компонент реализован в виде сервера, прослушивающего порт 25 и работающего по протоколу SMTP.

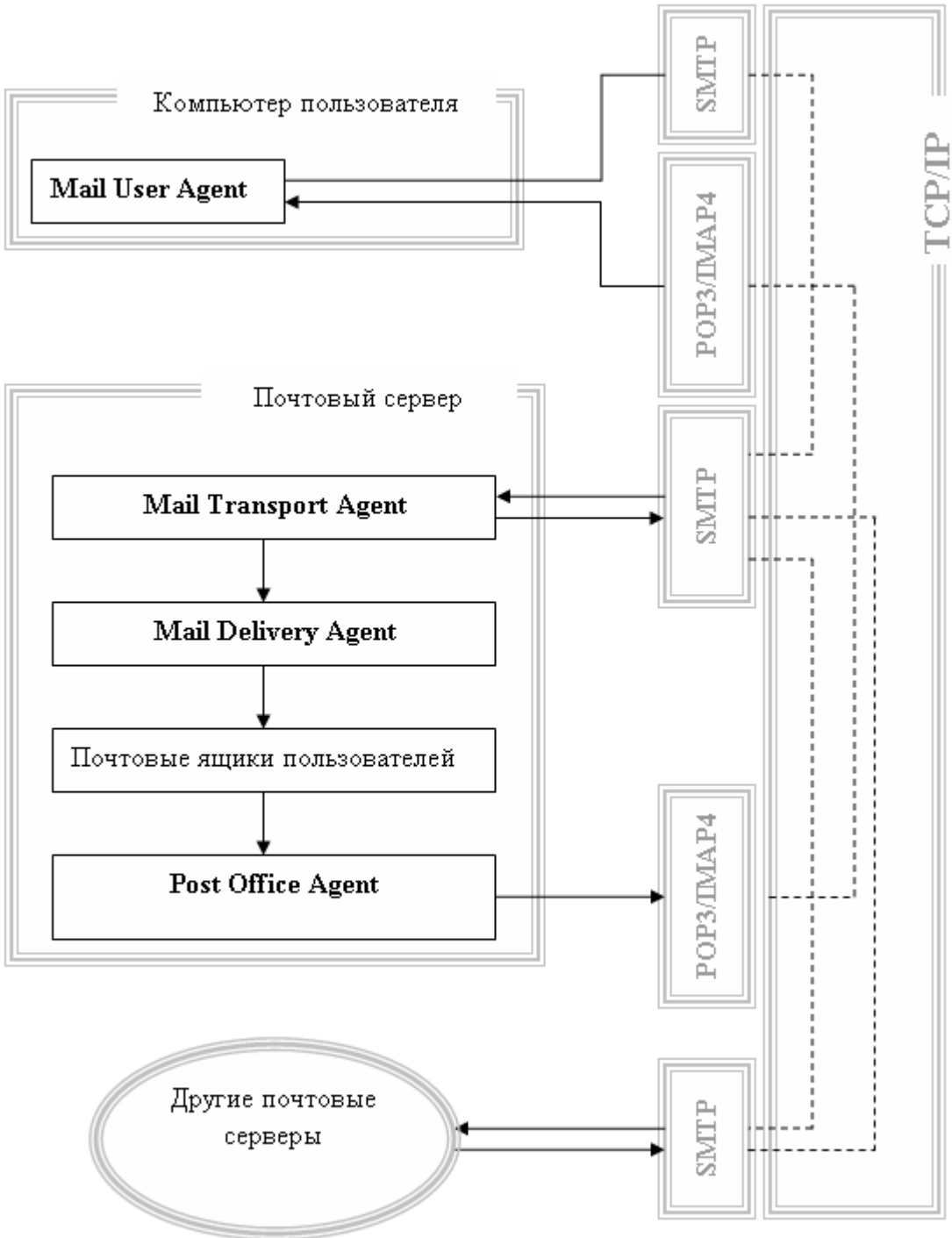
MDA – агент доставки, предназначенный для записи почтового сообщения в почтовый ящик. MDA реализован в виде отдельной программы, которую вызывает MTA по мере необходимости. Обычно, MDA располагают на том же компьютере, что и MTA.

POA – агент почтового отделения, позволяющий пользователю получить почтовое сообщение на свой компьютер. POA реализован в виде сервера, прослушивающего порты 110 и 143. При этом, порт 110 работает по протоколу POP3, порт 143 – IMAP4.

MUA – почтовый агент пользователя позволяет принимать почту по протоколам POP3 и IMAP4 и отправлять почту по протоколу SMTP.

Когда говорят о ***почтовом сервере***, то, обычно подразумевают совокупность серверов MTA, POA, программу MDA, а также систему хранения почтовых сообщений (почтовые ящики)

и ряд дополнительных программ, обеспечивающих безопасность и дополнительный сервис, расположенные на отдельном компьютере с TCP/IP-интерфейсом. Наиболее известными являются два почтовых сервера: ***Lotus Notes*** (IBM) и ***Microsoft Exchange Server. Почтовый клиент*** представляет собой программу, устанавливаемую на пользовательском компьютере и взаимодействующую с серверами MTA и POP3, почтового сервера, с помощью TPC/IP – соединения. Например, стандартным клиентом для отправления и организации работы с почтой в ОС Window является программа ***Outlook Express***.



20. Протокол HTTP.

Протокол ***HTTP (Hypertext Transfer Protocol)*** – это протокол Прикладного уровня, доставляющий информацию между различными ***гипермедийными*** системами. Под понятием ***гипермедийной системы*** понимается компьютерное представление системы данных, элементы которой представляются в различных форматах (гипертекст, графические изображения, видеоизображения, звук и т.д.) и обеспечивается автоматическая поддержка смысловых связей между представлениями элементов.

Протокол HTTP применяется в Internet с 1990 года. В настоящее время широкое распространение имеет версия HTTP 1.0, описанная в документе RFC 1945. Разработана новая версия HTTP 1.1 (документ RFC 2616), но пока она находится в стадии предложенного стандарта.

По умолчанию HTTP использует порт 80 и предназначен для построения систем архитектуры клиент-сервер. Запросы клиентов содержат ***URI (Uniform Resource Identifier)*** - универсальный идентификатор ресурса, позволяющий определить у сервера затребованный ресурс. URI представляет собой сочетание ***URL (Uniform Resource Locator)*** и ***URN (Uniform Resource Name)***. URL – унифицированный адресатор ресурсов: предназначен для указания места нахождения ресурса в сети. URN – унифицированное имя ресурса: идентифицирует ресурс, по указанному месту его нахождения (подразумевается, что по данному адресу может быть представлено несколько различных ресурсов).

HTTP (англ. *HyperText Transfer Protocol* — «протокол передачи гипертекста») — протокол прикладного уровня передачи данных (изначально — в виде гипертекстовых документов).

Основой HTTP является технология «клиент-сервер», то есть предполагается существование потребителей (клиентов), которые инициируют соединение и посылают запрос, и поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

HTTP в настоящее время повсеместно используется во Всемирной паутине для получения информации с веб-сайтов.

Достоинства HTTP:

- Простота
- Расширяемость
- Распространённость

Недостатки HTTP:

- Большой размер сообщений
- Отсутствие «навигации»
- Нет поддержки распределённости

Всё программное обеспечение для работы с протоколом HTTP разделяется на три больших категории:

- **Серверы** как основные поставщики услуг хранения и обработки информации (обработка запросов).
- **Клиенты** — конечные потребители услуг сервера (отправка запроса).
- **Прокси** для выполнения транспортных служб.

Каждое HTTP-сообщение состоит из трёх частей, которые передаются в указанном порядке:

1. **Стартовая строка** (англ. Starting line) — определяет тип сообщения;
2. **Заголовки** (англ. Headers) — характеризуют тело сообщения, параметры передачи и прочие сведения;
3. **Тело сообщения** (англ. Message Body) — непосредственно данные сообщения. Обязательно должно отделяться от заголовков пустой строкой.

Заголовки и тело сообщения могут отсутствовать, но стартовая строка является обязательным элементом, так как указывает на тип запроса/ответа. Исключением является версия

0.9 протокола, у которой сообщение запроса содержит только стартовую строку, а сообщения ответа только тело сообщения.

Стартовые строки различаются для запроса и ответа. Строка запроса выглядит так:

– **GET URI** — для версии протокола 0.9.
– **Метод URI HTTP/Версия** — для остальных версий.

Здесь:

- **Метод (англ. Method)** — название запроса, одно слово заглавными буквами. В версии HTTP 0.9 использовался только метод GET, список запросов для версии 1.1 представлен ниже.
- **URI** определяет путь к запрашиваемому документу.
- **Версия (англ. Version)** — пара разделённых точкой арабских цифр. Например: 1.0.

Для запроса страницы, клиент должен передать строку:

GET /net/index.html HTTP/1.0

21. Служба RPC и технология XML-RPC.

PRC – удаленный вывод процедур. Главная особенность – вычислительная машина делает вычисления, клиенты посылают данные серверу на вычисления. Работа бывает синхронная и асинхронная. Синхронная клиент ждет ответ от сервера, асинхронная клиент отправляет данные на вычисление и продолжат свою работу. Процесс укомплектования параметров в сообщения называется – маршалинг. На стороне клиента имеется СТАБ клиента – это заглушка которая осуществляет маршалинг. Ядро клиента находится в операционной системе клиента. Процесс клиента передает СТАБУ клиента функцию, СТАБ переводит все в сообщение и передает ядру. Ядро клиента передает ядру сервера, ядро сервера передает СТАБУ сервера. СТАБ распаковывает сообщение вызывает функцию и передает параметры. Процедура отправляет СТАБУ и т.д. … Клиентский СТАБ распаковывает данные и передает их процессу который отправлял сообщение.

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.

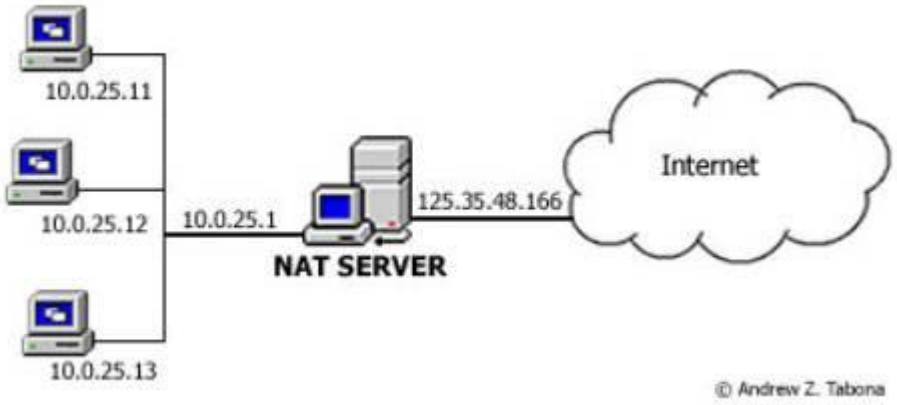
XML-RPC (сокр. от англ. Extensible Markup Language Remote Procedure Call —

XML-вызов удалённых процедур) — основанный на XML стандарт (протокол) вызова удалённых процедур, является прародителем SOAP, отличается исключительной простотой применения. XML-RPC, как и любой другой интерфейс RPC, определяет набор стандартных типов данных и команд, которые программист может использовать для доступа к функциональности другой программы, находящейся на другом компьютере в сети.

Протокол XML-RPC был изначально разработан Дэйвом Винером из компании «UserLand Software» в сотрудничестве с Майкрософт в 1995 году. Однако корпорация

Майкрософт вскоре сочла этот протокол слишком упрощённым, и начала расширять его функциональность. После нескольких циклов по расширению функциональности, появилась система, ныне известная как SOAP. Позднее Майкрософт начала широко рекламировать и внедрять SOAP, а изначальный XML-RPC был отвергнут. Но, несмотря на отвержение Майкрософт, стандарт XML-RPC очаровал многих программистов своей необычайной простотой и, за счёт этого, существует по сей день и даже постепенно набирает популярность.

22. NAT, proxy-серверы, межсетевые экраны, ремайлеры.



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.

Proxy Server

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 программа 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.

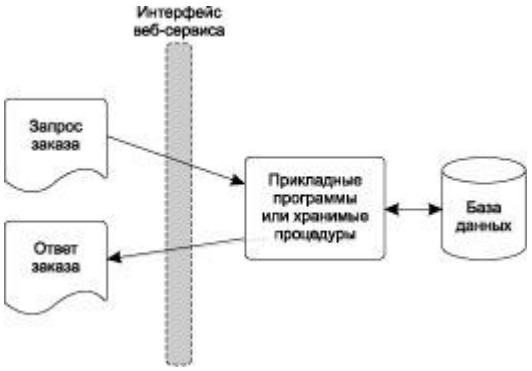
Стандарт MIXMINIOM анонимной перессылки почты, по этому стандарту можно отправлять электронную почту.

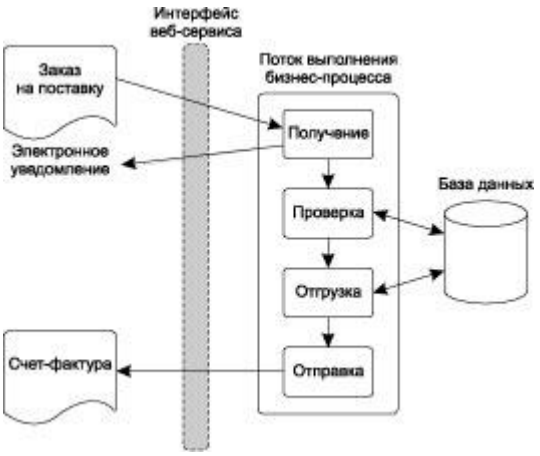
23. Web-сервисы: SOAP, XML, WSDL, UDDI.

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 with Attachments ("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.

24. 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.

2. 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.

3. 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.

2. 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.

2. 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.

2. Те же действия, повлекшие тяжкие последствия, –

наказываются лишением свободы на срок от трех до десяти лет.

25. 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.