

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

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

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.

4. Понятие сокета. Основные параметры сокета.

5. Интерфейс Named Pipe.

6. Интерфейс MailSlot.

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

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

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

13. Широковещание. Обнаружение сервера с помощью широковещания.

14. Применение символического адреса хоста.

15. Основные сетевые утилиты и их назначение.

16. Служба DNS.

17. Служба DHCP.

18. Стандарты сообщений Internet.

19. Протокол HTTP.

20. Служба RPC.

21. NAT, проху-серверы, межсетевые экраны, ремайлеры.

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

23. Национальная инфраструктура информационной безопасности.

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.

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

ISO/OSI	TCP/IP	
Прикладной	Прикладной	На этом уровне находятся службы TCP/IP и прикладные системы пользователя.
Представительский		
Сеансовый	Транспортный (TCP, UDP)	Обеспечивает сквозную доставку данных произвольного размера по сети между прикладными процессами, запущенными на узлах сети (UDP – ориентирован на сообщения, TCP – на соединение).
Транспортный		
Сетевой	Межсетевой (IP, ICMP, ARP, RARP)	Осуществляет перенос между сетями различных типов адресной информации в унифицированной форме. Сборка и разборка пакетов при передаче их между сетями с различными максимальными значениями длины пакета (IP, ICMP, ARP).
Ethernet, SLIP, PPP) Канальный		
Физический (LAN, WAN,	Уровень доступа к сети	Используются протоколы, обеспечивающие создание локальных сетей или соединений с глобальными сетями (Ethernet, PPP).

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 (MAC – определяет методы совместного использования *сетевыми адаптерами* среды передачи данных) и подуровень управления логическим каналом (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). Основное назначение – подготовить блок данных (кадр) для следующего сетевого уровня.

Сетевой уровень определяет методы адресации и маршрутизации компьютеров в сети. определяет единый метод адресации для всех компьютеров в сети не зависимо от способа передачи данных. На этом уровне определяются способы соединения компьютерных сетей. Результатом является пакет.

Транспортный уровень. Основным назначением процедур транспортного уровня является подготовка и доставка пакетов данных между конечными точками без ошибок и в правильной последовательности. формируются файлы из пакетов.

Сеансовый уровень определяют способы установки и разрыва соединений (называемых сеансами) двух приложений, работающих в сети. Сеансовый уровень - это точка взаимодействия программ и компьютерной сети.

Представительский уровень 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).

Архитектура распределенного приложения – распределение ролей между различными процессами распределенного приложения

Распределенное приложение, имеющее **архитектуру клиент-сервер**, подразумевает наличие в своем составе два вида процессов: процессы-серверы и процессы-клиенты. Некоторые процессы распределенного приложения могут выступать клиентом для некоторых процессов-серверов и одновременно являться серверами других процессов-клиентов.

Инициатором обмена данными между клиентом и сервером всегда является клиент. Для этого клиент должен обладать информацией о месте нахождения сервера или иметь механизмы для его обнаружения. Способы связи между клиентом и сервером могут быть различными и в общем случае зависят от интерфейсов, поддерживаемых операционной средой, в которой работает распределенное приложение. Клиент должен быть тоже распознан сервером, для того, чтобы сервер, во-первых, мог его отличить от других клиентов, а во-вторых, чтобы смог обмениваться с клиентом данными. Если основная вычислительная нагрузка ложится на сервер, а клиент лишь обеспечивает интерфейс пользователя с сервером, то такой клиент часто называют **тонким**.

По методу обслуживания серверы подразделяются на **итеративные** и **параллельные** серверы. *Принципиальная разница заключается в том, что параллельный сервер предназначен для обслуживания нескольких клиентов одновременно и поэтому использует специальные средства операционной системы, позволяющие распараллеливать обработку нескольких клиентских запросов. Итеративный сервер обслуживает запросы клиентов поочередно, заставляя клиентов ожидать своей очереди на обслуживание, или просто отказывает клиенту обслуживании.* По всей видимости, можно говорить о итеративно-параллельных серверах, когда сервер имеет ограниченные возможности по распараллеливанию своей работы. В этом случае только часть клиентских запросов будет обслуживаться параллельно.

NDIS – спецификация интерфейса сетевого драйвера, была разработана совместно фирмами Microsoft и 3Com 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.

TCP/IP	
Прикладной	На этом уровне находятся службы TCP/IP и прикладные системы пользователя.
Транспортный (TCP, UDP)	Обеспечивает сквозную доставку данных произвольного размера по сети между прикладными процессами, запущенными на узлах сети (UDP – ориентирован на сообщения, TCP – на соединение).
Межсетевой (IP, ICMP, ARP, RARP)	Осуществляет перенос между сетями различных типов адресной информации в унифицированной форме. Сборка и разборка пакетов при передаче их между сетями с различными максимальными значениями длины пакета (IP, ICMP, ARP).
Уровень доступа к сети	Используются протоколы, обеспечивающие создание локальных сетей или соединений с глобальными сетями (Ethernet, PPP).

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. 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 SCLIP

5) PPP (Point-to-Point Protocol) – протокол двухточечного соединения. PPP поддерживает многоканальные реализации: можно сгруппировать несколько каналов с одинаковой пропускной способностью между отправителем и получателем. 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. С помощью этого протокола осуществляется деятельность утилит достижимости (ping, traceroute); регулируется частота отправки 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;

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

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

Протоколы транспортного уровня: 1) TCP; 2) UDP (User Datagram Protocol) – протокол передачи дейтаграмм пользователя.

Протоколы прикладного уровня:

1) HTTP – протокол для передачи произвольных данных (порт 80 по умолчанию).

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

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

Класс	Диапазон адресов	Диапазон частных адресов
A	0.0.0.0 – 127.255.255.255	10.0.0.0 – 10.255.255.255
B	128.0.0.0 – 191.255.255.255	172.16.0.0 – 172.31.255.255
C	192.0.0.0 – 223.255.255.255	192.168.0.0 – 192.168.255.255
D	224.0.0.0 – 239.255.255.255	не предусмотрен
E	240.0.0.0 – 247.255.255.255	не предусмотрен

Класс D–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.

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

Хорошо известные (0-1023) номера портов присваиваются базовым системным службам, имеющие системные привилегии. Зарегистрированные (1024-49151) номера портов присваиваются промышленным приложениям и процессам. Динамические (49152-65535) (эфемерные) номера портов 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.

Протоколы уровня доступа к сети (канальный):

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:

2) ICMP (Internet Control Message Protocol) – транспортировка информации о сетевой деятельности и маршрутизации. ICMP 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.

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.

обмен без соединения (ориентированный на сообщения) Это такой обмен данными (сообщениями), особенностью которого является то, что протоколом, который обеспечивает такой обмен, **не гарантируется доставка и правильная последовательность** приема отправленных сообщений. Весь контроль надежности доставки сообщений возлагается на разработчика приложения. В связи с этим, обмен данными с помощью сообщений используется в основном для широковещательных сообщений или для пересылки коротких сообщений, последовательность получения которых не имеет значения.

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 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	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	Установить 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_STREAM** – сокет, ориентированный на поток;
- старший номер версии;

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

- для TCP/IP можно указать `NULL`

Разрешение на использование широковещательных адресов устанавливается функцией `setsockopt`. Функция `setsockopt` используется для установки опции сокета `SO_BROADCAST`(для разрешения использования широковещательного адреса), позволяющей использовать адрес `INADDR_BROADCAST`.

Использование широковещательных адресов возможно только в протоколе UDP.

```
int setsockopt (
    SOCKET          s,           // [in] дескриптор сокета
    int             level,       // [in] уровень действия режима
    int             optname,     // [in] режим сокета для установки
    const char*     optval,      // [in] значение режима сокета
    int             fromlen      // [in] длина буфера optval
);
```

5.Интерфейс Named Pipe.

Современные операционные системы имеют встроенные механизмы межпроцессного взаимодействия - **IPC (InterProcess Communication)**, предназначенные для обмена данными между процессами и для синхронизации процессов. **Named Pipe (именованный канал)** - программный интерфейс, который может быть использован для обмена данными между распределенными в локальной сети процессами.

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)
- 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 (сообщениями отправитель и получатель дожидаются друг друга для передачи каждого сообщения, и операция отправки считается завершенной только после того, как получатель закончит прием сообщения)

асинхронным (не происходит никакой координации между отправителем и получателем сообщения).

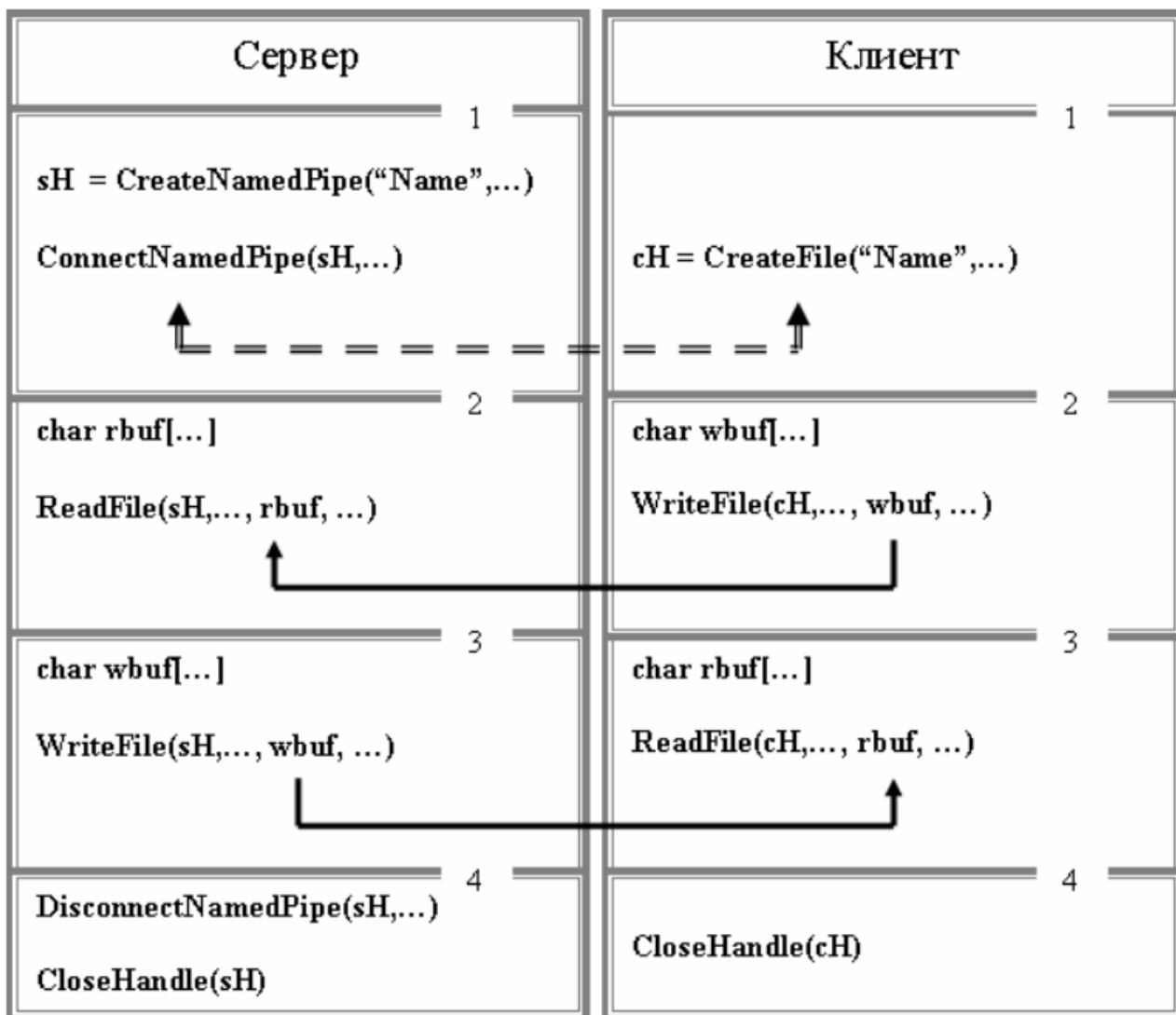
Все функции NamedPipeAPI можно разбить на **три группы**:

1. функции управления каналом (создать канал, соединить сервер с каналом, открыть канал, получить информацию об именованном канале, получить состояние канала, изменить характеристики канала);

2. функции обмена данными (писать в канал, читать из канала, копировать данные канала)

3. 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
TransactNamedPipe	Писать и читать данные канала
WaitNamedPipe	Определить доступность канала
WriteFile	Писать данные в канал



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.

```
HANDLE CreateNamedPipe
(
    LPCTSTR    pname, // [in] символическое имя канала
    DWORD      omode, // [in] атрибуты канала
    DWORD      pmode, // [in] режимы передачи данных
    DWORD      pimax, // [in] макс. к-во экземпляров канала
    DWORD      osize, // [in] размер выходного буфера
    DWORD      isize, // [in] размер входного буфера
    DWORD      timeo, // [in] время ожидания связи с клиентом
    LPSECURITY_ATTRIBUTES sattr // [in] атрибуты безопасности
)
```

Локальный формат имени канала:

`\\.\pipe\xxxxx`

где: **точка (.)** – обозначает локальный компьютер;
pipe – фиксированное слово;
xxxxx – имя канала

Сетевой формат имени канала: вместо точки имя сервера

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 **потокком**.

```
HANDLE CreateFile
(
    LPCTSTR    pname, // [in] символическое имя канала
    DWORD      accss, // [in] чтение или запись в канал
    DWORD      share, // [in] режим совместного использования
    LPSECURITY_ATTRIBUTES sattr // [in] атрибуты безопасности
    DWORD      oflag, // [in] флаг открытия канала
    DWORD      aflag, // [in] флаги и атрибуты
    HANDLE      exten, // [in] дополнительные атрибуты
);

BOOL ReadFile
(
    HANDLE      hP, // [in] дескриптор канала
    LPVOID      pb, // [out] указатель на буфер ввода
    DWORD      sb, // [in] количество читаемых байт
    LPDWORD     ps, // [out] количество прочитанных байт
    LPOVERLAPPED ol // [in,out] для асинхронной обработки
);

BOOL WriteFile
(
    HANDLE      hP, // [in] дескриптор канала
    LPVOID      pb, // [in] указатель на буфер вывода
    DWORD      sb, // [in] количество записываемых байт
    LPDWORD     ps, // [out] количество записанных байт
    LPOVERLAPPED ol // [in,out] для асинхронной обработки
);
```

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

```
BOOL PeekNamedPipe
(
    HANDLE      hP, // [in] дескриптор канала
    LPVOID      pb, // [out] указатель на буфер
    DWORD      sb, // [in] размер буфера
    LPDWORD     pi, // [out] количество прочитанных байт
    LPDWORD     pa, // [out] количество доступных байт
    LPDWORD     pr, // [out] количество непрочитанных байт
);
```

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

```
BOOL TransactNamedPipe
(
    HANDLE      hP, // [in] дескриптор канала
    LPVOID      pw, // [in] указатель на буфер для записи
    DWORD      sw, // [in] размер буфера для записи
    LPVOID      pr, // [out] указатель на буфер для чтения
    DWORD      sr, // [in] размер буфера для чтения
    LPDWORD     pr, // [out] количество прочитанных байт
    LPOVERLAPPED ol // [in,out] для асинхронного доступа
);
```

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

BOOL CallNamedPipe

```
(
    LPCTSTR      nP,    // [in] указатель на имя канала
    LPVOID       pw,    // [in] указатель на буфер для записи
    DWORD        sw,    // [in] размер буфера для записи
    LPVOID       pr,    // [out] указатель на буфер для чтения
    DWORD        sr,    // [in] размер буфера для чтения
    LPDWORD      pr,    // [out] количество прочитанных байт
    DWORD        to     // [in] интервал ожидания
);
```

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

BOOL GetNamedPipeInfo

```
(
    HANDLE  hP,    // [in] дескриптор именованного канала
    LPDWORD pfg,   // [in] указатель на флаг-тип канала
    LPDWORD psw,   // [out] указатель на размер выходного буфера
    LPDWORD psr,   // [out] указатель на размер входного буфера
    LPDWORD pmi,   // [out] указатель на макс. к-во экземпляров канала
);
```

Отличие Name Pipe Maislot:

- Именованные каналы ориентированы на соединение, а почтовые ящики - нет.
- Слоты электронной почты могут использоваться для широковещательной рассылки, именованные каналы - нет.
- почтовые ящики больше похожи на UDP, но именованные каналы ближе к TCP.
- **Почтовый ящик** - это механизм одностороннего межпроцессного взаимодействия (IPC). Приложения могут хранить сообщения в почтовом ящике. Владелец почтового ящика может получать сообщения, которые там хранятся. Эти сообщения обычно отправляются по сети либо на указанный компьютер, либо на все компьютеры в указанном домене.
- **Именованный канал** - это именованный односторонний или дуплексный канал для связи между сервером каналов и одним или несколькими клиентами каналов. Все экземпляры именованного канала имеют одно и то же имя канала, но каждый экземпляр имеет свои собственные буферы и дескрипторы и предоставляет отдельный канал для взаимодействия клиент / сервер.

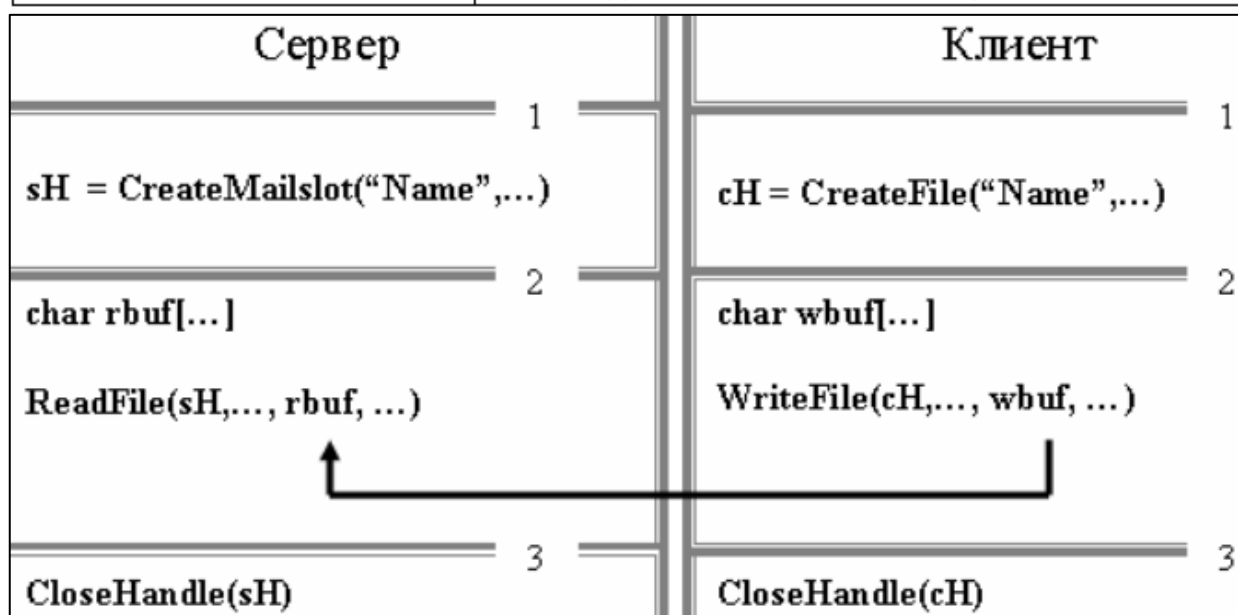
6.Интерфейс MailSlot.

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 сообщениями и 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.

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



```
HANDLE CreateMailslot
(
    LPCTSTR      pname,    // [in] символическое имя ящика
    DWORD        maxms,    // [in] максимальная длина сообщения
    DWORD        timeo,    // [in] интервал ожидания
    LPSECURITY_ATTRIBUTES sattrib // [in] атрибуты безопасности
);
```

Локальный формат имени почтового ящика:

`\\.\mailslot\xxxxx`

где: **точка (.)** – обозначает локальный компьютер;
mailslot – фиксированное слово;
xxxxx – имя почтового ящика

Сетевой формат имени почтового ящика:

```
\\servername\mailslot\xxxxxx
```

где: **servername** - имя компьютера-сервера почтового ящика;
mailslot - фиксированное слово;
xxxxxx - имя почтового ящика

Доменный формат имени почтового ящика:

```
\\domain\mailslot\xxxxxx
```

где: **domain** - имя домена компьютеров или *;
mailslot - фиксированное слово;
xxxxxx - имя почтового ящика

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().

```
BOOL GetMailslotInfo
```

```
(  
    HANDLE    hM,    // [in] дескриптор почтового ящика  
    LPDWORD   ml,    // [out] максимальная длина сообщения  
    LPDWORD   nl,    // [out] длина следующего сообщения  
    LPDWORD   nm      // [out] количество сообщений  
    LPDWORD   to      // [out] интервал ожидания сообщения  
);
```

Функция SetMailSlotInfo() предназначена для изменения интервала времени ожидания.

```
BOOL SetMailslotInfo
```

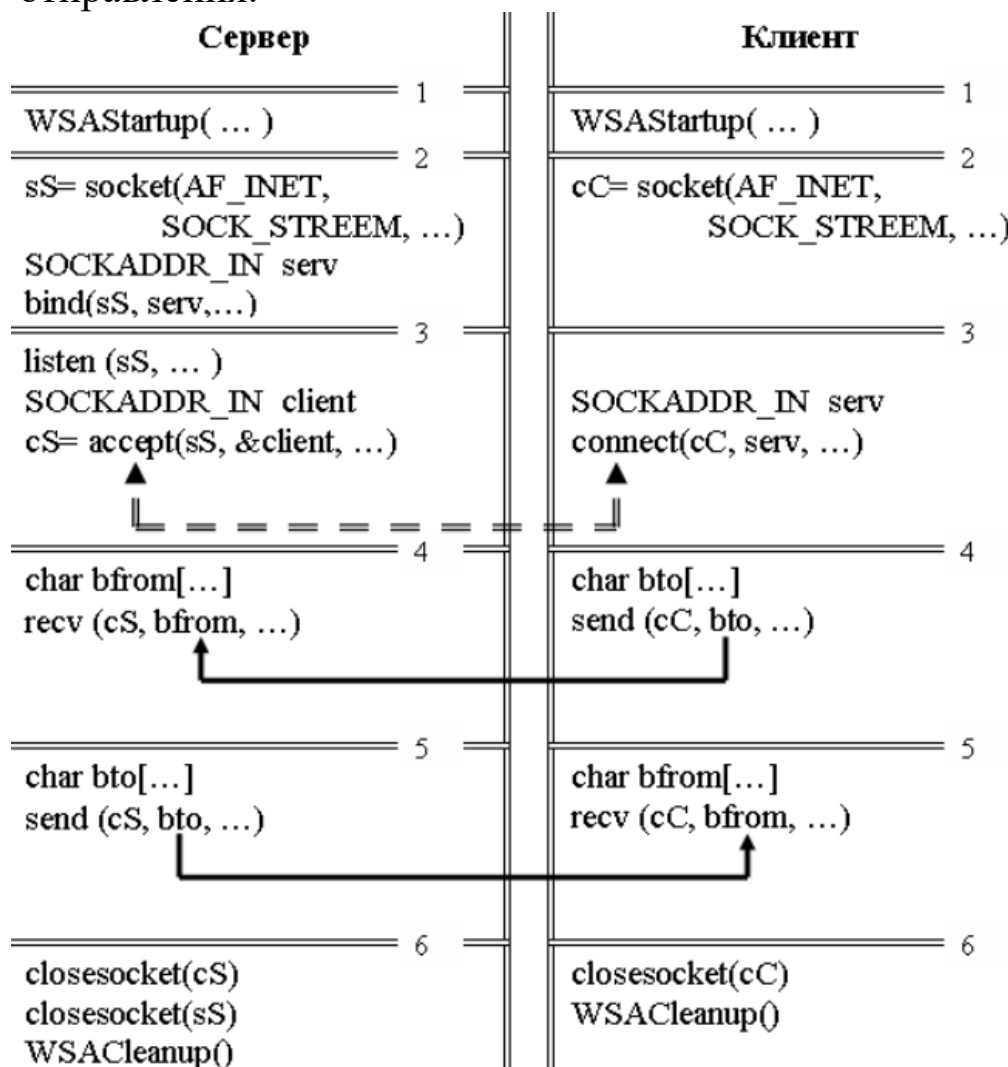
```
(  
    HANDLE    hM, // [in] дескриптор почтового ящика  
    PDWORD    to  // [in] новое значение интервала  
);
```

Отличие Name Pipe Maislot:

- Именованные каналы ориентированы на соединение, а почтовые ящики - нет.
- Слоты электронной почты могут использоваться для широковещательной рассылки, именованные каналы - нет.
- почтовые ящики больше похожи на UDP, но именованные каналы ближе к ТСР.
- Почтовый ящик** - это механизм одностороннего межпроцессного взаимодействия (IPC). Приложения могут хранить сообщения в почтовом ящике. Владелец почтового ящика может получать сообщения, которые там хранятся. Эти сообщения обычно отправляются по сети либо на указанный компьютер, либо на все компьютеры в указанном домене.
- Именованный канал** - это именованный односторонний или дуплексный канал для связи между сервером каналов и одним или несколькими клиентами каналов. Все экземпляры именованного канала имеют одно и то же имя канала, но каждый экземпляр имеет свои собственные буферы и дескрипторы и предоставляет отдельный канал для взаимодействия клиент / сервер.

7.-8. Структура программы ТСП-сервера/клиента.

Схема, ориентированная на поток – между сокетами устанавливается **ТСР-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_STREAM** 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.

```

SOCKET socket(
            int      af,      //[in]  формат адреса
            int      type,   //[in]  тип сокета
            int      prot    //[in]  протокол
            );

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
    };

```

Использование широковещательных адресов возможно только в протоколе UDP.

Установить различные режимы использования сокета можно с помощью функции `setsockopt()`.

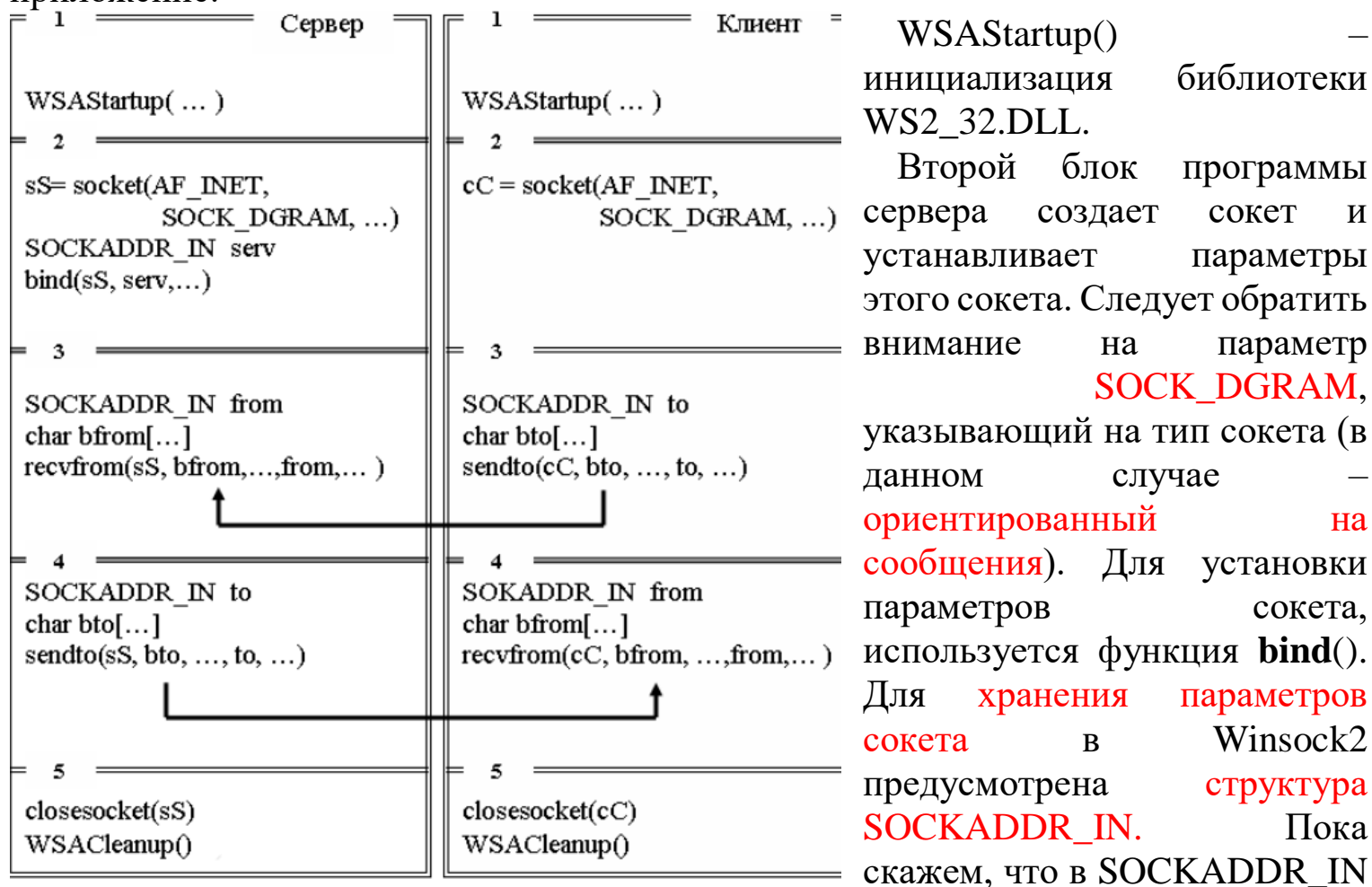
```

int setsockopt (
    SOCKET      s,           // [in] дескриптор сокета
    int         level,       // [in] уровень действия режима
    int         optname,     // [in] режим сокета для установки
    const char* optval,     // [in] значение режима сокета
    int         fromlen     // [in] длина буфера optval
    );

```

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.



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.**

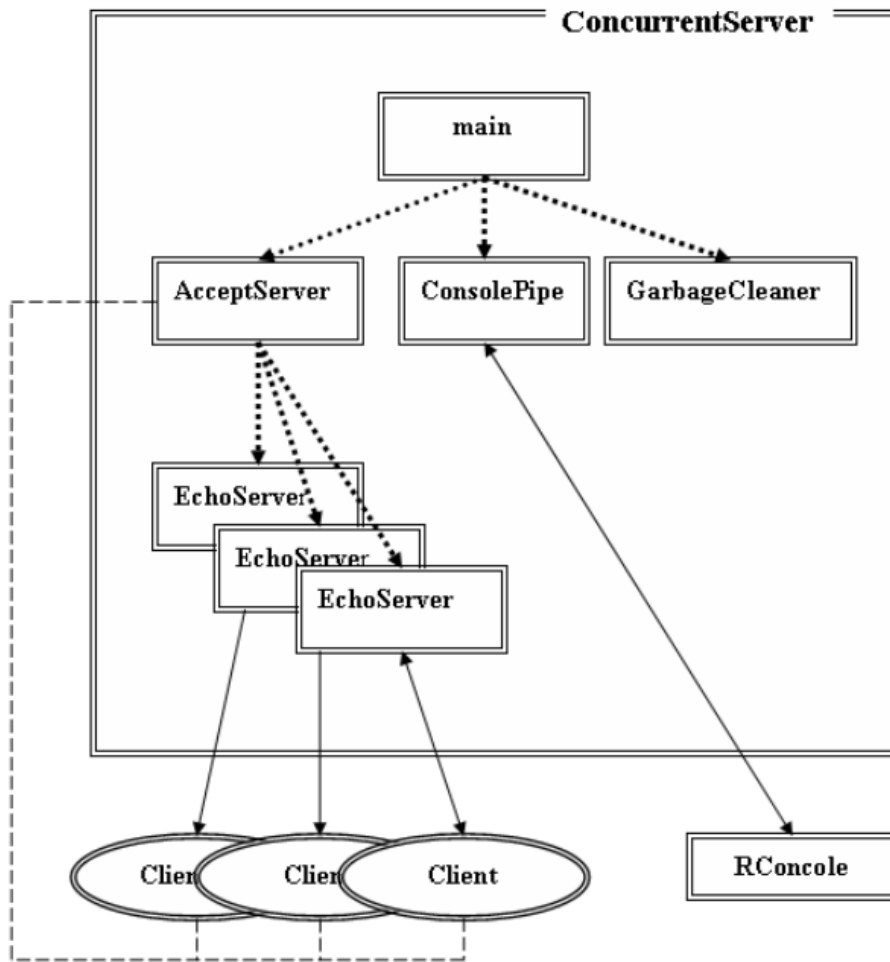
Четвертый блок программы сервера предназначен для пересылки данных клиенту. Пересылка данных осуществляется с помощью функции **sendto()**. В качестве параметров **sendto()** используется структура **SOCKADDR_IN** с параметрами сокета принимающей стороны (в данном случае – клиента) и заполненный буфер 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. AcceпtServer. 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: **AcceпtServer, ConsolePipe и GarbageCleaner**.

Процесс **AcceпtServer** – 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, **AcceпtServer** создает **список подключений**, который называется **ListContact**. При подключении очередного клиента, процесс **AcceпtServer** 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 программой **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 **AcceпtServer** по одному **для каждого успешного подключения** программы **клиента**. Основным назначением процесса **EchoServer** является **прием данных** по созданному процессом **AcceпtServer** 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.

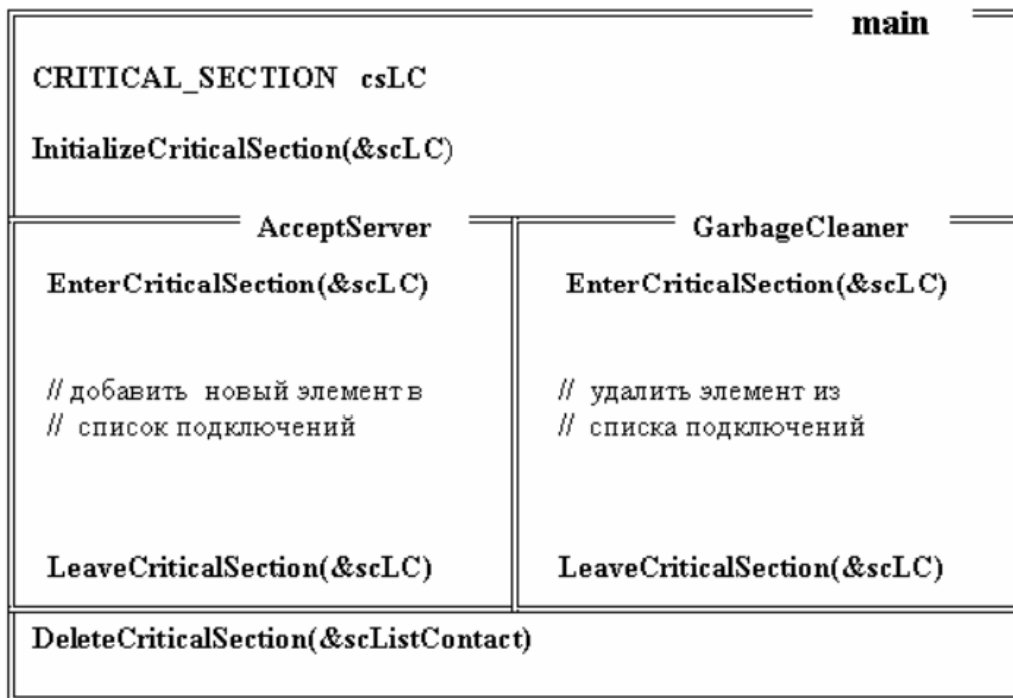
Процесс RConsole – предназначена для **ввода команд управления сервером и для вывода диагностических сообщений, полученных от сервера**. RConsole является клиентом именованного канала.

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.

Наименование функции	Назначение
CreateThread	Создать поток
ResumeThread	Возобновить поток
SuspendThread	Приостановить поток
Sleep	Задержать исполнение
TerminateThread	Завершить поток

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.

Наименование функции	Назначение
<code>CancelWaitableTimer</code>	Отменить ожидающий таймер
<code>CreateWaitableTimer</code>	Создать ожидающий таймер
<code>OpenWaitableTimer</code>	Открыть существующий ожидающий таймер
<code>SetWaitableTimer</code>	Установить ожидающий таймер
<code>WaitForSingleObject</code>	Ждать сигнального состояния ожидающего таймера

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

Наименование Функции	Назначение
<code>InterlockedCompareExchange</code>	Сравнить и заменить значение
<code>InterlockedDecrement</code>	Уменьшить значение на единицу
<code>InterlockedExchange</code>	Заменить значение
<code>InterlockedExchangeAdd</code>	Изменить значение
<code>InterlockedIncrement</code>	Увеличить значение на единицу

Переключение сокета в режим без блокировки. В этом режиме выполнение функции `ассепт()`, 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 `INVALID_SOCKET`, если запроса на создание канала нет в очереди запросов или возникла ошибка. Для переключения сокета применяется функция `ioctlsocket()`.

```
int ioctlsocket(
    SOCKET sock, // [in] дескриптор сокета
    long scmd, // [in] команда, применяемая к сокету
    u_long* pprm // [in,out] указатель на параметр команды
);
```

Наименование функции	Назначение
<code>GetPriorityClass</code>	Получить приоритет процесса
<code>GetThreadPriority</code>	Получить приоритет потока
<code>GetProcessPriorityBoost</code>	Определить состояние режима процесса
<code>GetThreadPriorityBoost</code>	Определить состояние режима потока
<code>SetPriorityClass</code>	Изменить приоритет процесса
<code>SetThreadPriority</code>	Изменить приоритет потока
<code>SetProcessPriorityBoost</code>	Установить или отменить динамический режим потоков процесса
<code>SetThreadPriorityBoost</code>	Установить или отменить динамический режим потока

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.

Наименование функции	Назначение
<code>CreateEvent</code>	Создать событие
<code>OpenEvent</code>	Открыть событие
<code>PulseEvent</code>	Освободить ожидающие потоки
<code>ResetEvent</code>	Перевести событие в несигнальное состояние
<code>SetEvent</code>	Перевести событие в сигнальное состояние

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

Наименование функции	Назначение
<code>FreeLibrary</code>	Отключить dll-библиотеку от процесса
<code>GetProcAddress</code>	Импортировать функцию
<code>LoadLibrary</code>	Загрузить dll-библиотеку

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`, 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()`.

```
int setsockopt (
    SOCKET      s,           // [in] дескриптор сокета
    int         level,       // [in] уровень действия режима
    int         optname,     // [in] режим сокета для установки
    const char* optval,      // [in] значение режима сокета
    int         fromlen      // [in] длина буфера optval
);
```

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 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 противоположное: **получение символического имени компьютера по сетевому адресу**. Обе функции используют структуру `hostent`, содержащуюся в `Winsock2.h`.

```
hostent* gethostbyname
(
    const char*    name,    // [in] символическое имя хоста
);

hostent* gethostbyaddr
(
    const char*    addr,    // [in] адрес в сетевом формате
    int            la,      // [in] длина адреса в байтах
    int            ta       // [in] тип адреса: для TCP/IP AF_INET
);

typedef struct hostent {
    char FAR* h_name;        // имя хоста
    char FAR  FAR** h_aliases; // список алиасов
    short h_addrtype;        // тип адресации
    short h_length;          // длина адреса
    char FAR  FAR** h_addr_list; // список адресов
} hostent;

int gethostname
(
    char*    name , // [out] имя хоста
    int      ln      // [in] длина буфера name
);
```


15. Основные сетевые утилиты и их назначение.

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

Наименование утилиты	Назначение утилит
ping	Проверка соединения с одним или более хостами в сети
tracert	Определение маршрута до пункта назначения
route	Просмотр и модификация таблицы сетевых маршрутов
netstat	Просмотр статистики текущих сетевых TCP/IP-соединений
arp	Просмотр и модификация ARP-таблицы
nslookup	Диагностика DNS-серверов
hostname	Просмотр имени хоста
ipconfig	Просмотр текущей конфигурации сети TCP/IP
nbtstat	Просмотр статистики текущих сетевых NBT-соединений
net	Управление сетью

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 параметром -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.

Определение ip адреса по имени компьютера

DNS работает на прикладном уровне модели ISO/OSI.

Утилита NSLOOKUP

```
> nslookup www.yandex.ru
Сервер:  dc1.imm.uorlan.ru
Address: 172.19.132.29
```

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 поддереву дерева базы данных DNS.**



Режимы работы DNS:

Итеративный

- Если сервер отвечает за данную доменную зону – он возвращает ответ
- Если не отвечает, то возвращает адрес DNS-сервера, у которого есть более точная информация

▶ Рекурсивный

- Сервер сам выполняет запросы к другим серверам DNS, чтобы найти нужный адрес

Запрос DNS	Ответ DNS
Имя	Имя
Тип записи	Тип записи
Класс записи	Класс записи
	Время жизни (TTL)
	Длина данных
	Данные

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:

+-----+	
Header	Заголовок
+-----+	
Question	Секция запросов
+-----+	
Answer	Секция ответа
+-----+	
Authority	Секция ответа об уполномоченных серверах
+-----+	
Additional	Секция ответа дополнительных записей
+-----+	

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.

Сервер должен находиться в одной подсети с клиентом.

Служба DHCP состоит из 3 модулей:

- сервера DHCP (комп, который выдает адрес и смотрит, чтобы не было дублирующихся),
- клиента DHCP (комп, который получает IP автоматически)
- 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

1. DHCP-клиент выдает в сеть широковещательный запрос, на получение адреса сервера.

2. 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.

3. Получив адрес сервера, клиент формирует запрос на выделение IP-адреса из пула адресов DHCP-сервера.

4. 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 договориться с DHCP-сервером о продлении срока аренды или о выделении нового IP-адреса.

Dynamic Host Configuration Protocol				
Bit Offset	0–15		16–31	
0	OpCode	Hardware Type	Hardware Length	Hops
32	Transaction ID			
64	Seconds Elapsed		Flags	
96	Client IP Address			
128	Your IP Address			
160	Server IP Address			
196	Gateway IP Address			
228+	Client Hardware Address (16 bytes)			
	Server Host Name (64 bytes)			
	Boot File (128 bytes)			
	Options			

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 кватирования (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.

Придумал Тим Бернес-Ли

- **Название:** Hypertext Transfer Protocol
- **Уровень (по модели OSI):** Прикладной
- **Семейство:** TCP/IP
- **Создан в:** 1990 г.
- **Порт/ID:** 80/TCP
- **Назначение протокола:** Доступ к гипертексту, ныне стал универсальным
- **Спецификация:** RFC 1945, RFC 2616
- **Основные реализации (клиенты):** Веб-браузеры, например Internet Explorer, Mozilla Firefox, Opera, Google Chrome и др.
- **Основные реализации (серверы):** Apache, IIS

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.

Запрос/статус ответа

- GET /courses/networks
- 200 OK

Заголовки (не обязательно)

- Host: www.asozykin.ru (обязательно в HTTP 1.1)
- Content-Type: text/html; charset=UTF-8

 Content-Length: 5161

Тело сообщения (не обязательно)

- Страница HTML
- Параметры, введенные пользователем

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.

GET – запрос Web-страницы

→ POST – передача данных на Web-сервер

HEAD – запрос заголовка страницы

PUT – помещение страницы на Web-сервер

DELETE – удаление страницы с Web-сервера

TRACE – трассировка страницы

OPTIONS – запрос поддерживаемых методов HTTP для ресурса

CONNECT – подключение к Web-серверу через прокси

В ответе сервера первое поле – статус обработки запроса

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 авторизован, 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).

1XX – информация

2XX – успешное выполнение (200 OK)

3XX – перенаправление (301 – постоянное перемещение, 307 – временное перенаправление)

4XX – Ошибка на стороне клиента (403 – доступ запрещен, 404 – страница не найдена)

5XX – Ошибка сервера (500 – внутренняя ошибка сервера)

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 вызываемой 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 процедурой, а также для сокрытия (серверный стаб подменяет вызываемую 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.

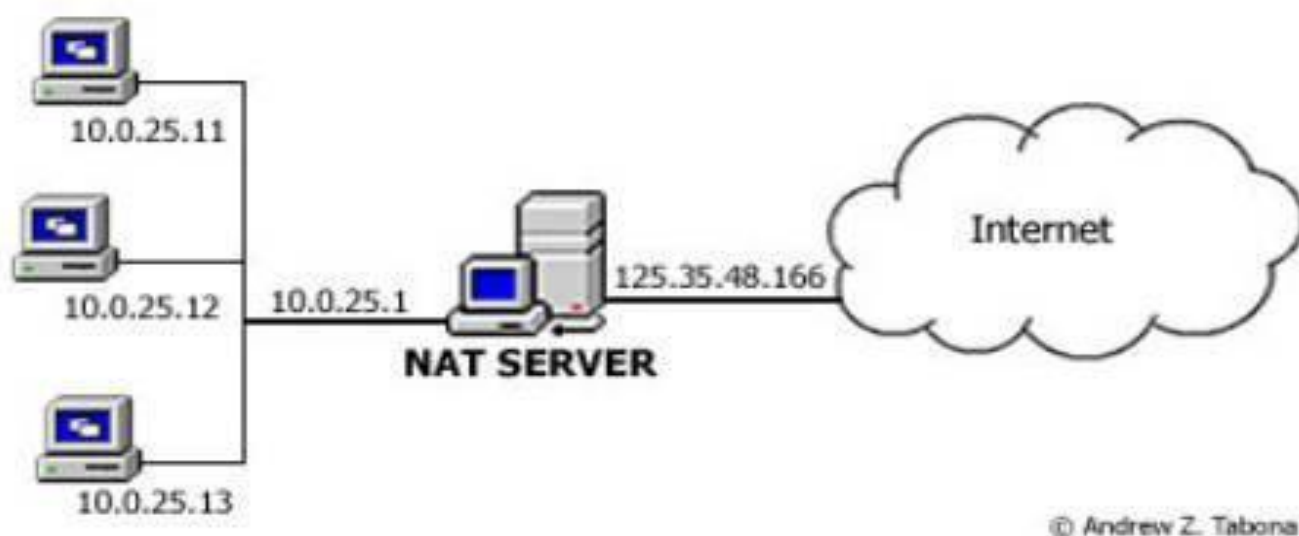
Аналогом RPC в Java-технологиях является механизм **RMI (Remote Method Invocation)**, позволяющий работать с удаленными Java-объектами. Информацию об объекте и его методах вызывающая сторона может получить, обратившись к реестру RMI (аналогу PortMapper).

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, проху-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 NAT 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 ТСР 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 службам. 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 пересылки 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.

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. Умышленное 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.

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.

Конфиденциальность – предотвращение попадания 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 (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 зашифрованный текст.

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.

Криптолoгия — наука, занимающаяся методами шифрования и дешифрования. состоит из двух частей — криптографии и криптоанализа.

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/TLS 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.