Интернет

Системное программирование/Сетевое программирование.

Что такое интернет?

Интернет - глобальная сеть компьютеров, соединенных друг с другом, которые обмениваются данными посредством стандартизированного набора протоколов.

Разработчику важно иметь четкое представление о том, что такое Интернет и как он работает. Это основа, на которой построено большинство современных программных приложений. Чтобы создавать эффективные, безопасные и масштабируемые приложения и сервисы, вам необходимо иметь четкое представление о том, как работает Интернет и как использовать его возможности

Как работает Интернет?

Прежде чем поговорить о Интернете, давайте поймем, что такое сеть. **Сеть** - это просто группа компьютеров или устройств, которые связаны друг с другом.

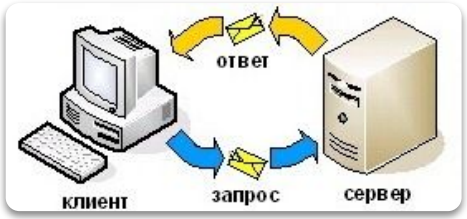
Например, у вас дома может быть своя сеть компьютеров, а у вашего соседа тоже своя. Когда все эти сети объединяются, они образуют **Интернет** - своего рода сеть сетей.

Интернет появился в конце 1960-х годов благодаря *Министерству обороны США.* Изначально задумывался как децентрализованная сеть связи, способная выжить при ядерной атаке. С течением времени он стал сложной системой, охватывающей весь мир.

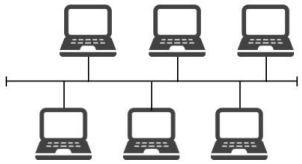
Понятие сервера и клиента.

Сервер - устройство или программа, предоставляющая услуги или ресурсы другим устройствам (клиентам) в сети.

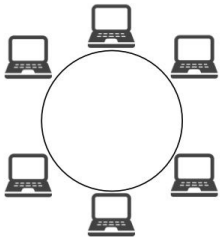
Клиент - устройство или программа, обращающаяся к серверу для получения доступа к услугам, данным или ресурсам.



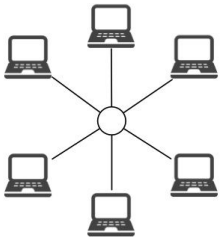
Классификация сетей.



Шинная топология



Кольцевая топология



Звездная топология

1. *<u>По размеру:</u>*

   1.1 **LAN (Local Area Network)** - маленькая сеть для дома, офиса или здания.

   1.2 **MAN (Metropolitan Area Network)** - сеть для города.

   1.3 **WAN (Wide Area Network)** - расширенная сеть для больших расстояний, между городами или странами.

2. *<u>По топологии:</u>*

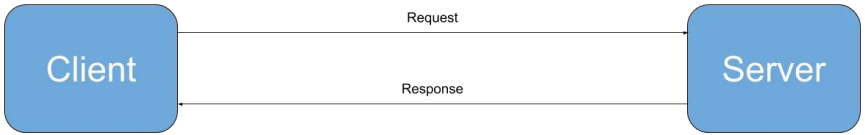
   2.1 **Звезда** - устройства подключены к центральному хабу или коммутатору.

   2.2 **Кольцо** - устройства соединены в кольцо, каждое с двумя соседями.

   2.3 **Шинная топология**

Что такое HTTP/HTTPs?

HTTP - это протокол передачи данных между клиентом (например, веб-браузером) и сервером (например, веб-сайтом). Когда вы посещаете веб-сайт, ваш браузер отправляет HTTP-запрос на сервер для получения нужных данных, а сервер в ответ отправляет HTTP-ответ с запрошенной информацией. **HTTPS** - это зашифрованная версия HTTP, обеспечивающая безопасное взаимодействие между клиентом и сервером.



HTTP коды.

HTTP-коды - числовые статусы, возвращаемые сервером в ответ на запрос клиента. Они указывают на успешность или ошибки взаимодействия. Например, код **200** означает успешный запрос, а **404** - что запрашиваемый ресурс не найден. Коды помогают определить состояние и обработать запрос корректным образом.

Браузеры и их работа.

Браузеры, такие как **Chrome** или **Firefox**, являются программами, позволяющими пользователям просматривать веб-сайты. Они отправляют **HTTP-запросы** на **серверы**, получают **HTTP-ответы** и отображают веб-страницы. Браузеры также обрабатывают HTML, CSS и JavaScript для создания интерактивных веб-страниц.

DNS.

DNS - это система, отвечающая за преобразование удобочитаемых доменных имен, таких как **google.com**, в **IP-адреса**. Когда вы вводите доменное имя в браузер, компьютер отправляет DNS-запрос на сервер, который возвращает соответствующий **IP-адрес**. Это необходимо для правильного направления запросов на сервер.

Также важно понимать что такое TCP/IP:

TCP/IP - это набор протоколов передачи данных, используемых для связи в сетях. Он обеспечивает надежную и упорядоченную передачу данных между устройствами в сети.

TCP (Transmission Control Protocol) отвечает за управление передачей данных, а **IP** (Internet Protocol) - за маршрутизацию и адресацию, обеспечивая глобальную связь в Интернете. TCP/IP является основой интернет-протокола.

Сокет. Дескриптор сокета.

Сокет - точка соединения для обмена данных между компьютерами в сети интернета или локальной сети. **Дескриптор сокета** - объект, который управляет сокетом, он же является обычным файловым дескриптором файловой системы.

Сокеты открываются через socket().

```
int sockfd = socket(int domain, int type, int protocol);
```

- domain – домен сокета (AF_INET для IPv4, AF_UNIX для UNIX и т.д.);
- type – тип сокета;
  - SOCK_STREAM – сокет потоковой передачи данных (TCP).
  - SOCK_DGRAM – сокет ненадежной передачи сообщений (UDP).
  - *SOCK_SEQPACKET - сокет надежной передачи.
  - и другие типы
- protocol – протокол соединения (0-auto).

close(sockfd); - для закрытия сокета;

Метод bind();

Метод bind() прямое предназначение - привязка адреса к сокету.

```
int bind(int sockfd, const sockaddr* addr, socklen_t addrlen);
```

Структура sockaddr:

- **sockaddr_in** исп-ся для сокетов AF_INET.
- **sockaddr_un** исп-ся для сокетов AF_UNIX.

```
struct sockaddr_in {
    sin_family,
    sin_port,
    sin_addr.s_addr
}
```

пример использования;

```
struct sockaddr in sa;
sa.sin_family = AF_INET;
sa.sin port = htons(123);
sa.sin addr.s addr =
inet_addr("127.0.0.1");
```

Метод connect();

В сетевом программировании connect() используется для установления соединения с удаленным сервером.

Например при TCP/IP:

```
connect(sockfd, (struct sockaddr *)&sa, sizeof(sa));
```

Протокол TCP.

функции которые использует TCP:

- send/recv - для отправки данных (send - отправка, recv - прием)
- read/write
- close

Протокол с установлением соединения: Перед началом передачи данных TCP устанавливает соединение между отправителем и получателем с помощью трехступенчатого рукопожатия (three-way handshake). Это обеспечивает надежное соединение для обмена данными.

Надежная доставка: TCP гарантирует доставку данных без потерь. Он использует механизмы подтверждения получения (ACKs) и повторной передачи потерянных или поврежденных пакетов.

Контроль потока: TCP управляет потоком данных между отправителем и получателем, чтобы избежать перегрузки сети. Это позволяет регулировать скорость передачи данных, предотвращая переполнение буфера получателя.

Контроль ошибок: TCP включает механизмы для проверки целостности данных, используя контрольные суммы.

Порядок доставки: TCP гарантирует, что пакеты данных будут доставлены в том порядке, в котором они были отправлены. Для этого он использует последовательные номера и механизмы управления окнами (windowing).

Управление перегрузкой: TCP включает алгоритмы управления перегрузкой, такие как алгоритмы медленного старта (slow start) и избегания перегрузки (congestion avoidance). Эти алгоритмы помогают регулировать объем передаваемых данных, чтобы избежать перегрузки сети.

Протокол UDP.

функции которые использует UDP:

- sendto/recvfrom - для отправки данных
- close

Протокол UDP (User Datagram Protocol) — это один из основных протоколов транспортного уровня, который используется для передачи данных в сетях. Вот его основные характеристики:

Простой протокол без установления соединения: UDP не требует установки соединения перед отправкой данных. Он просто отправляет пакеты (датаграммы) от отправителя к получателю.

Ненадежная доставка: UDP не гарантирует доставку пакетов. Пакеты могут потеряться, прийти в неправильном порядке или быть дублированы.

Отсутствие контроля ошибок: Нет встроенного механизма для проверки и исправления ошибок. Это оставляется на усмотрение приложения.

Меньшие задержки: Из-за своей простоты и отсутствия механизма установления соединения, UDP имеет меньшие задержки, что делает его подходящим для приложений, требующих быстрой передачи данных, таких как онлайн-игры, потоковое аудио и видео, VoIP.

send/sendto

send:

- Используется с TCP-сокетами.
- Отправляет данные через установленное соединение.
- Синтаксис: int send(int sockfd, const void *buf, size_t len, int flags);
- Пример:

```
char *message = "Hello, server!";
int bytes_sent = send(sockfd, message, strlen(message), 0);
```

sendto:

- Используется с UDP-сокетами.
- Отправляет данные на указанный адрес.
- Синтаксис: int sendto(int sockfd, const void *buf, size_t len, int flags, const struct sockaddr *dest_addr, socklen_t addrlen);

recv/recvfrom

recv:

- Используется с TCP-сокетами.
- Принимает данные из сокета.
- Синтаксис: int recv(int sockfd, void *buf, size_t len, int flags);

recvfrom:

- Используется с UDP-сокетами.
- Принимает данные и адрес отправителя.
- Синтаксис: int recvfrom(int sockfd, void *buf, size_t len, int flags, struct sockaddr *src_addr, socklen_t *addrlen);

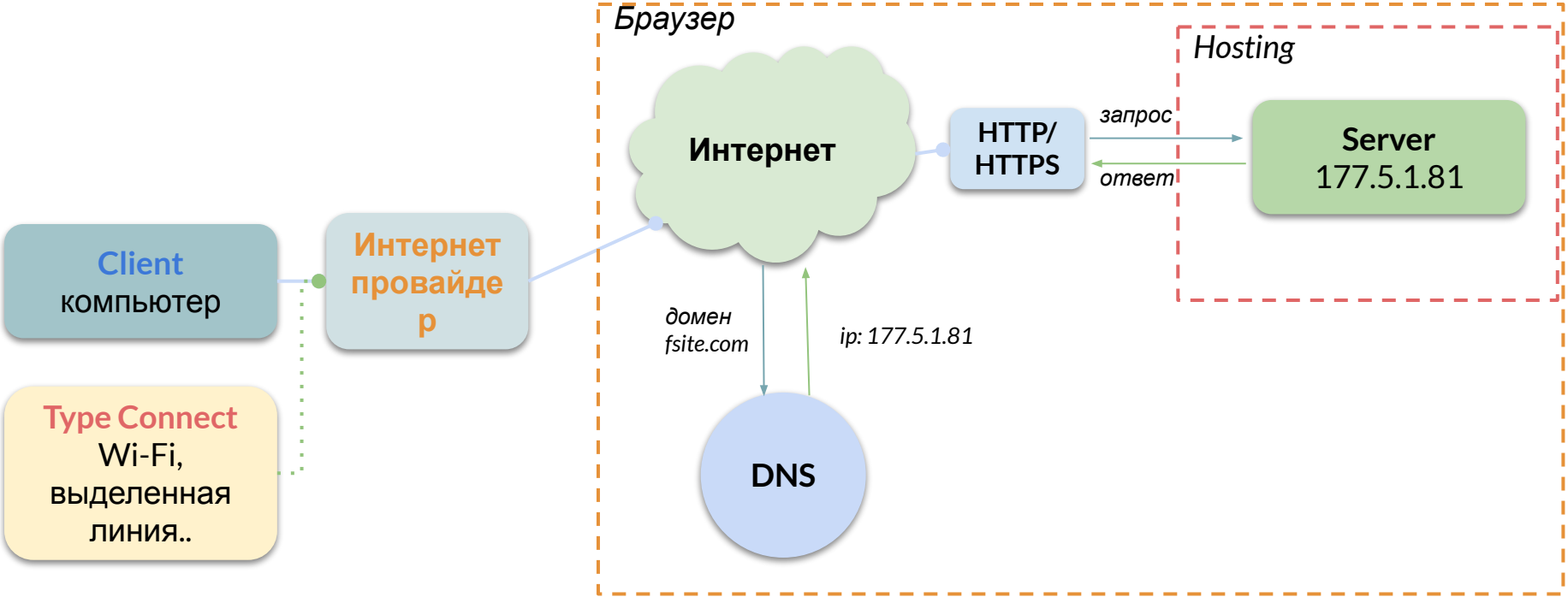
Доменное имя.

Доменное имя - это читаемое человеком имя, используемое для идентификации веб-сайта, например, google.com, gmail.com, и др. DNS преобразует доменные имена в IP-адреса, что позволяет устройствам найти правильные серверы.

Что такое хостинг?

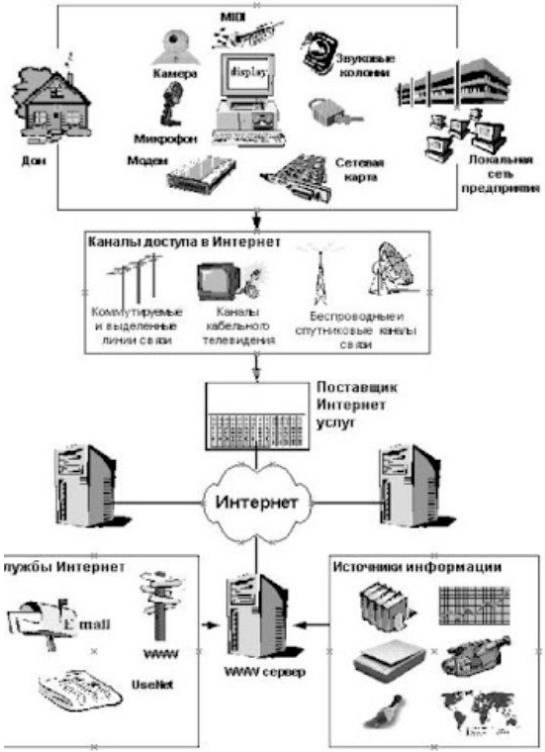
Хостинг - это услуга предоставления места на сервере для размещения веб-сайтов и приложений. Хостинг-провайдер предоставляет серверное пространство, обеспечивает соединение с Интернетом и поддерживает работу веб-сайта. Это позволяет владельцам сайтов делиться своим контентом онлайн, обеспечивает доступность и стабильную работу веб-проектов для пользователей по всему миру.

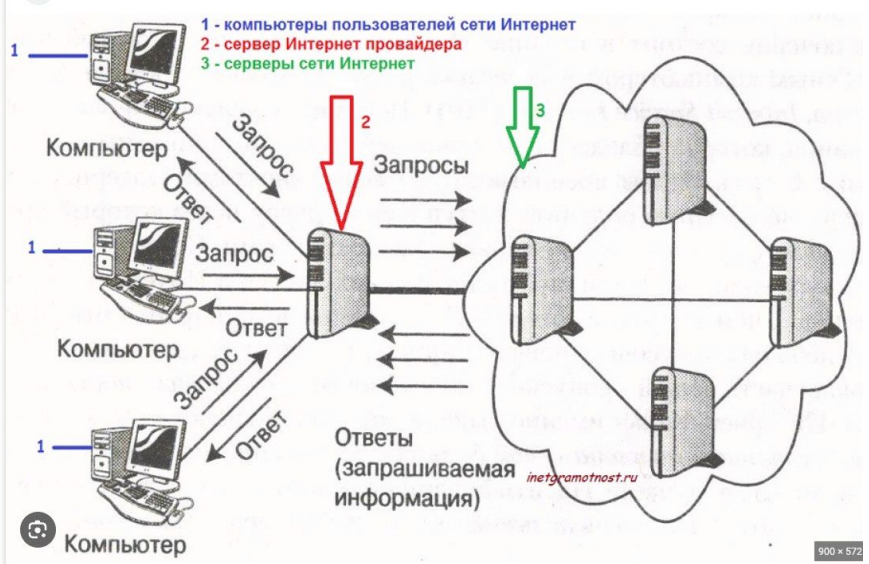
Упрощенная схема работы интернета.



интернет провайдер* - предоставляет доступ к сети интернет, а **type connect - тип соединения с провайдером.*
в более сложной схеме еще нужен маршрутизатор, proxy-server, или др сетевые устройства.

Схема-2


MIDI
Камера
display
Звуковые колонки
Микрофон
Модем
Сетевая карта
Локальная сеть предприятия
Дом
Каналы доступа в Интернет
Коммутируемые и выделенные линии связи
Каналы кабельного телевидения
Беспроводные и спутниковые каналы связи
Поставщик Интернет услуг
Интернет
Источники информации
E-mail
WWW
Usenet
WWW сервер


1 - компьютеры пользователей сети Интернет
2 - сервер Интернет провайдера
3 - серверы сети Интернет
1
Компьютер
Запрос
Ответ
2
3
Запросы
1
Компьютер
Запрос
Ответ
1
Компьютер
Запрос
Ответ
Ответы (запрашиваемая информация)
inetgramotnost.ru
900 × 572

ЗАДАЧИ;

1. написать функцию, которая соединяется с адресом 127.0.0.1 порт 123 и отправляется M нулевых байт по протоколу TCP.
2. написать функцию, которая соединяется с адресом 127.0.0.1 порт 123 и отправляет целое число по протоколу TCP.
3. написать функцию, которая соединяется с адресом 127.0.0.1 порт 123 и отправляет число по протоколу UDP.
4. написать функцию, которая соединяется с адресом 127.0.0.1 порт 123 по TCP и отправляет M случайных байт и завершается.
5. Написать функцию, которая соединяется с адресом 127.0.0.1, порт 123 и принимает сообщение по протоколу TCP, выводя его на экран.
6. Написать функцию, которая создает UDP-сокет, связывается с адресом 127.0.0.1, порт 123, и принимает сообщение по протоколу UDP, выводя его на экран.
7. Напишите функцию, которая создает один поток. Поток отправляет строку на адрес 127.0.0.1, порт 123 по протоколу UDP.