Рождение, жизнь и смерть сокетов

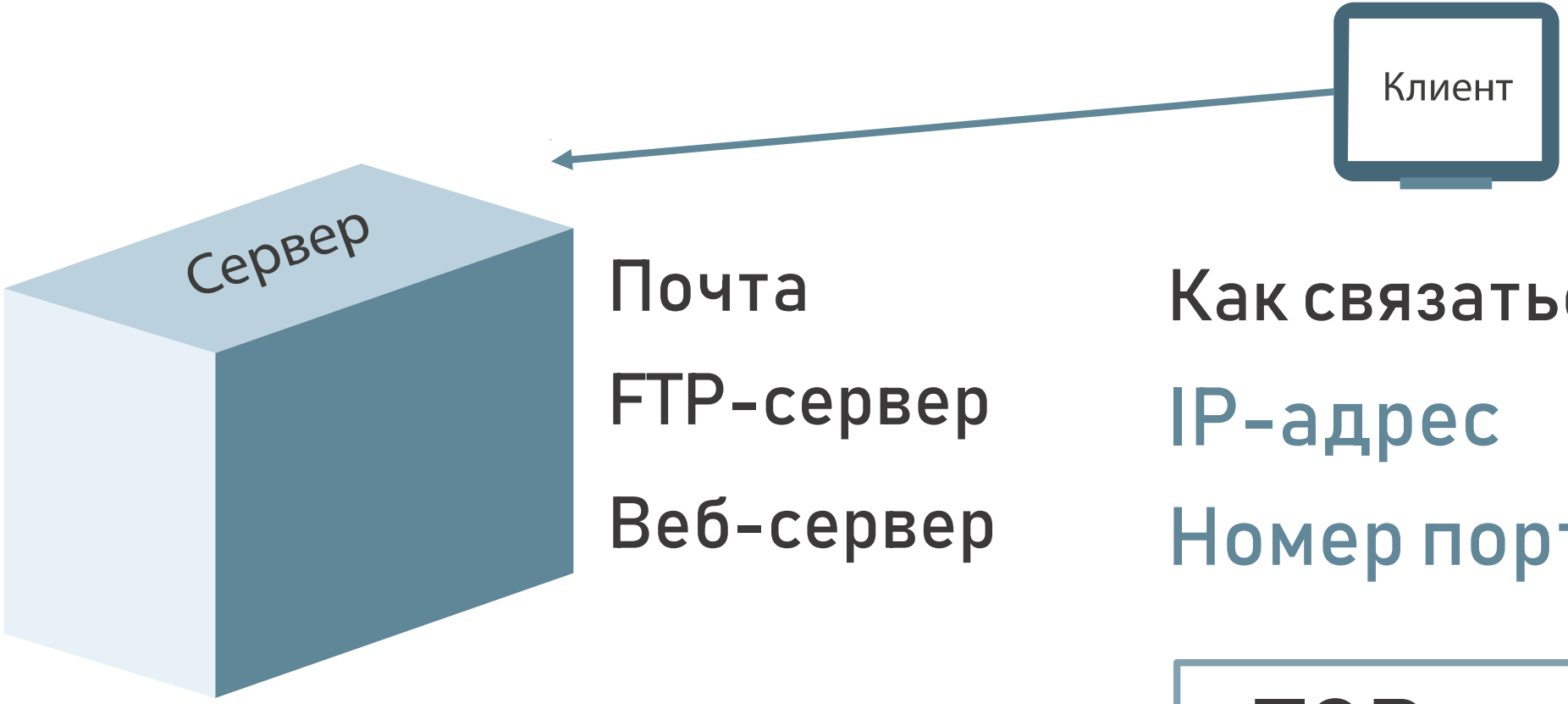
(на примере python)

Нина Пакшина

Глава 0

Начало

Что такое сокет?



Почта

FTP-сервер

Веб-сервер

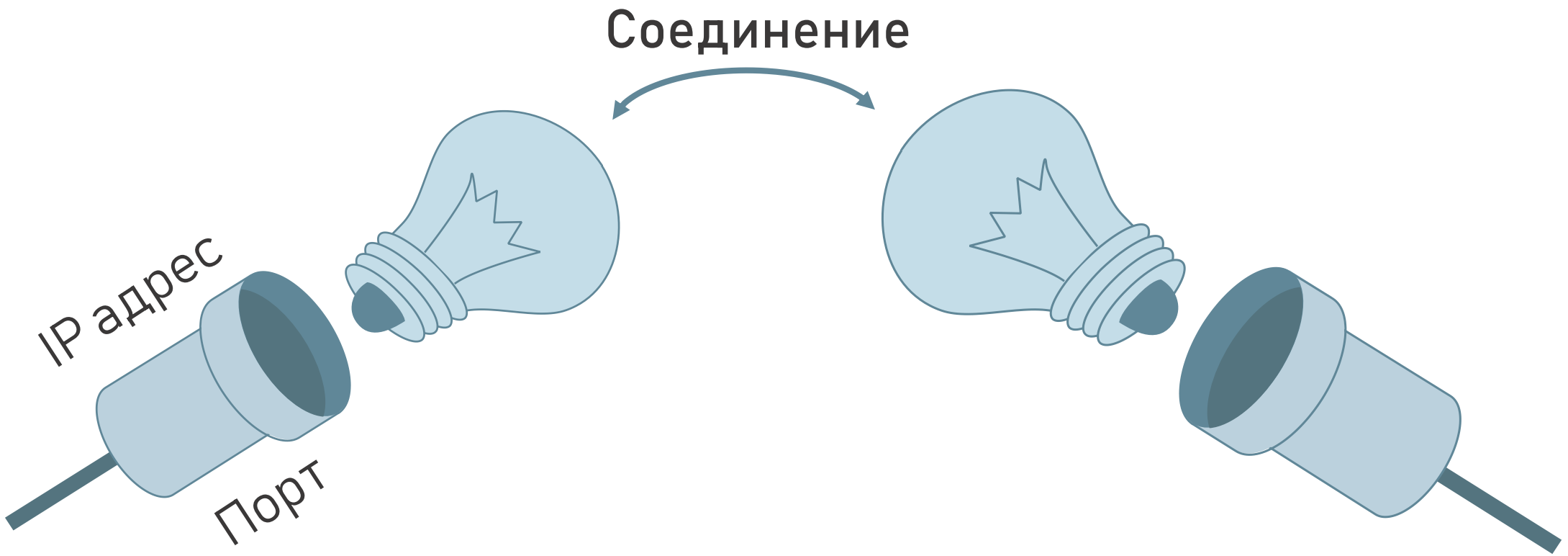
Как связаться?

IP-адрес +

Номер порта

= TCP сокет

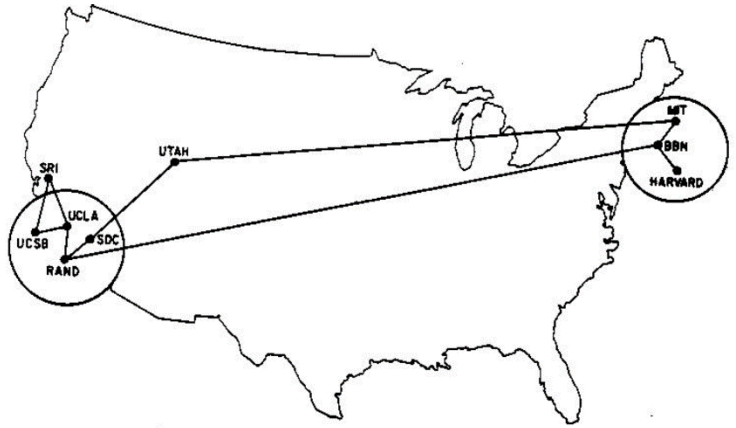
Что нужно для соединения?



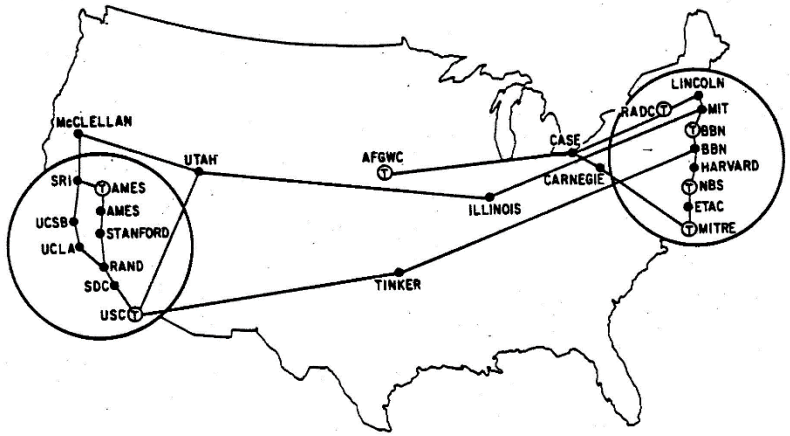
Немного истории

ARPANET

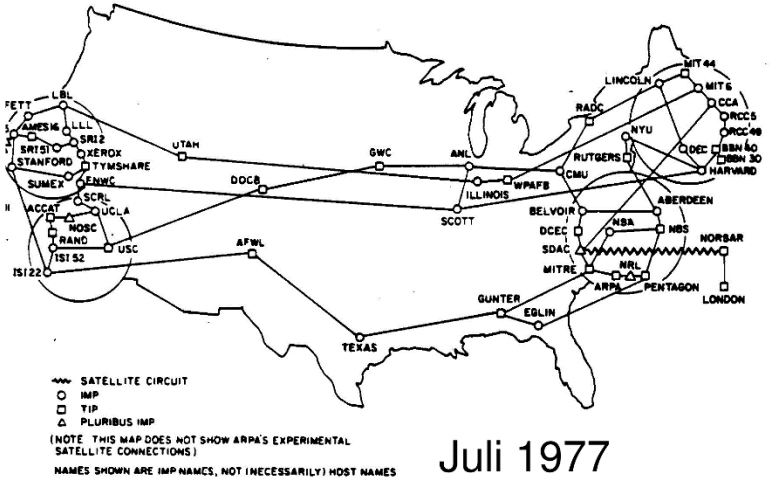
Dezember 1969



Juni 1970



März 1972



Juli 1977

RFC714

01.01.1983

ARPANET перешел на TCP

> …we concatenate a NETWORK identifier, and a **TCP identifier** with a **port name** to create a **SOCKET** name which will be unique throughout all networks connected together

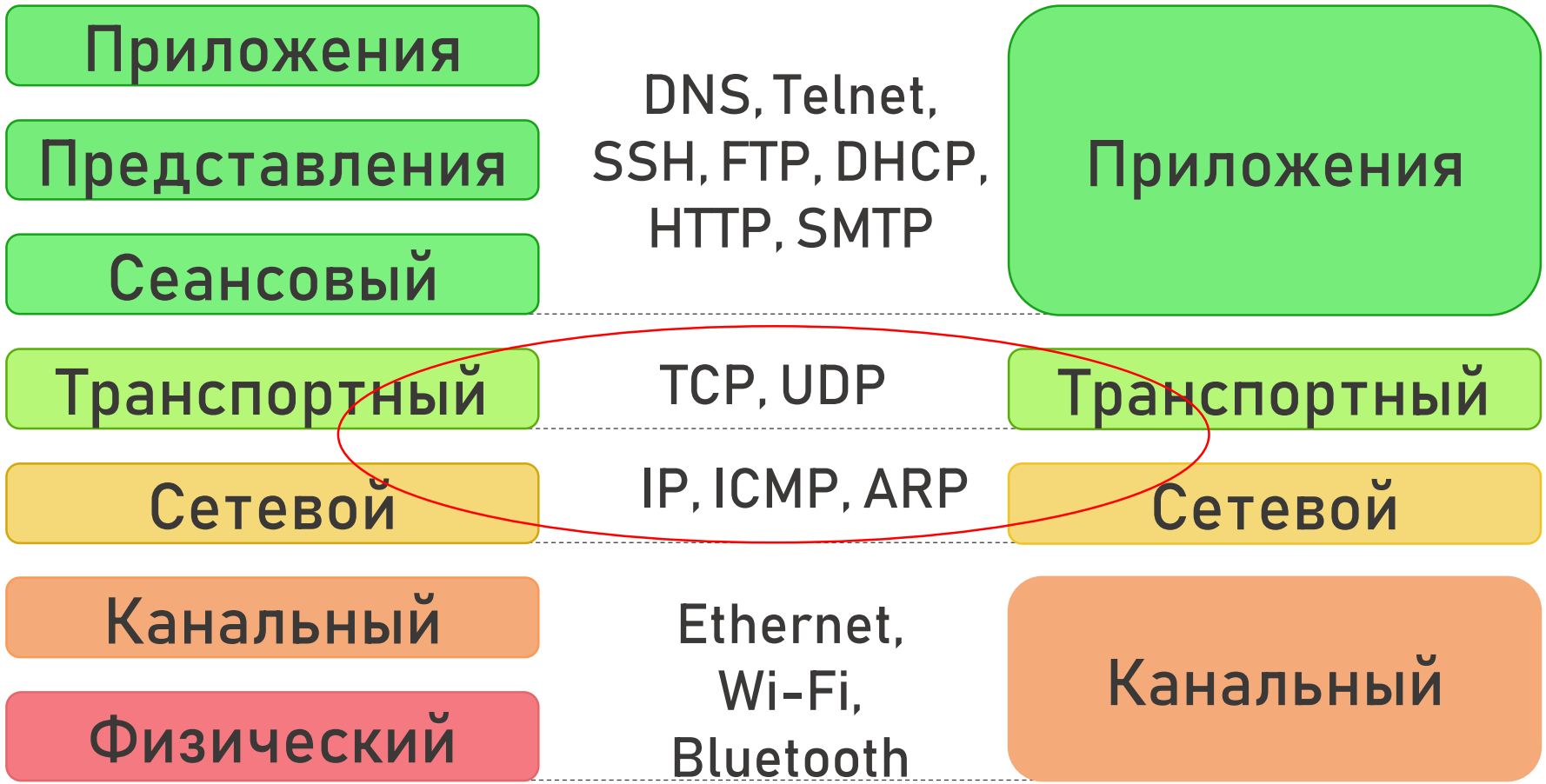
Сокеты Беркли

Berkeley
UNIVERSITY OF CALIFORNIA

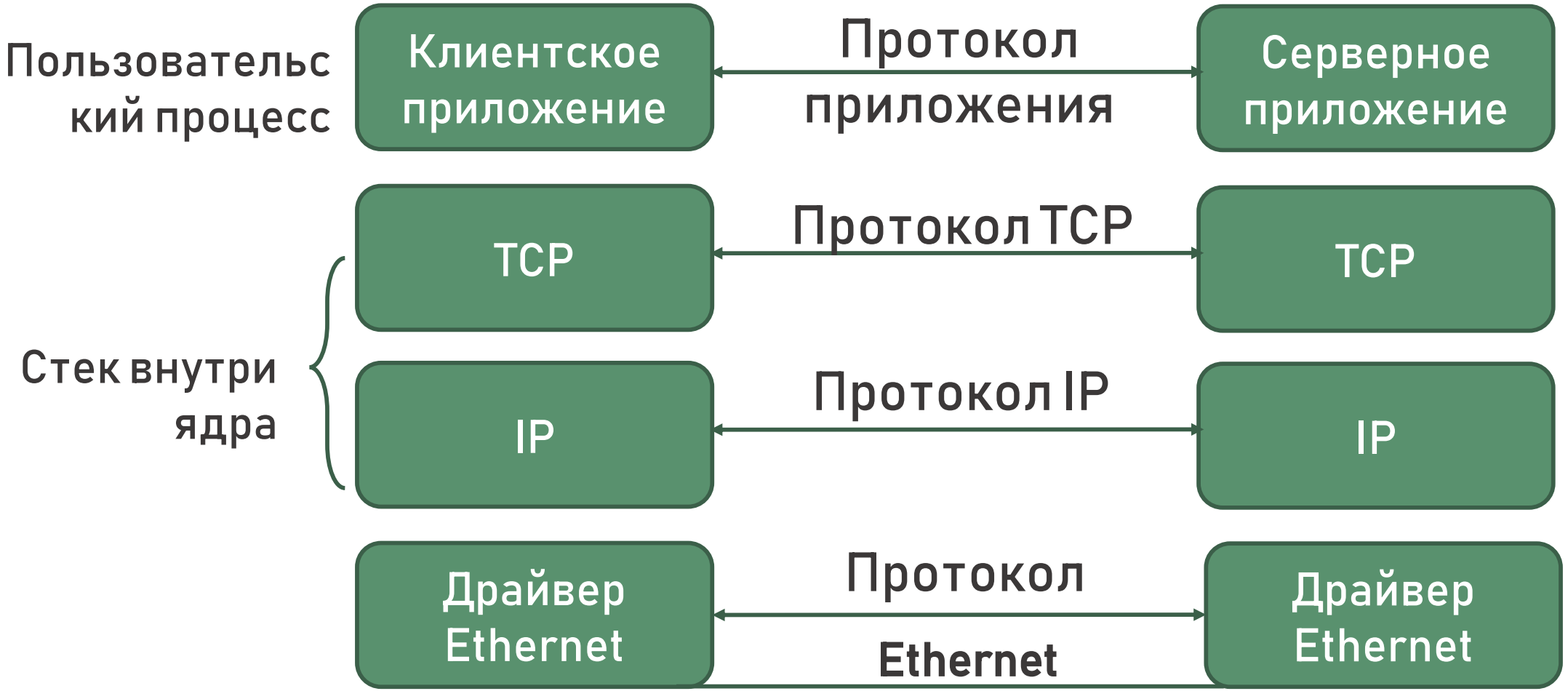
- 1983 (BSD Unix)
- 1988 стандарт POSIX.1

Зачем нужны сокеты?

Модель OSI и TCP/IP



Реальное взаимодействие



Глава 1

TCP сокеты

3 этапа

Рождение

Жизнь

Смерть

Рождение

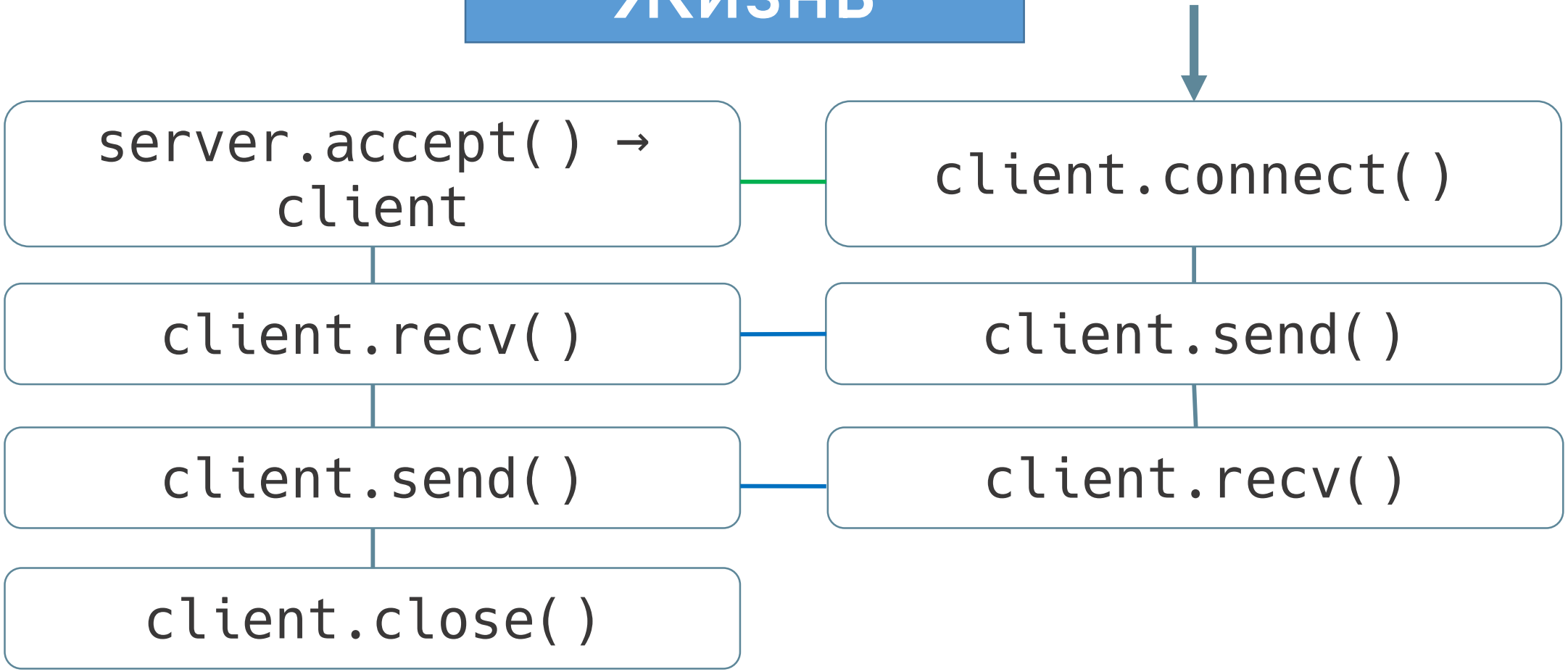
```
server = socket.socket()
```

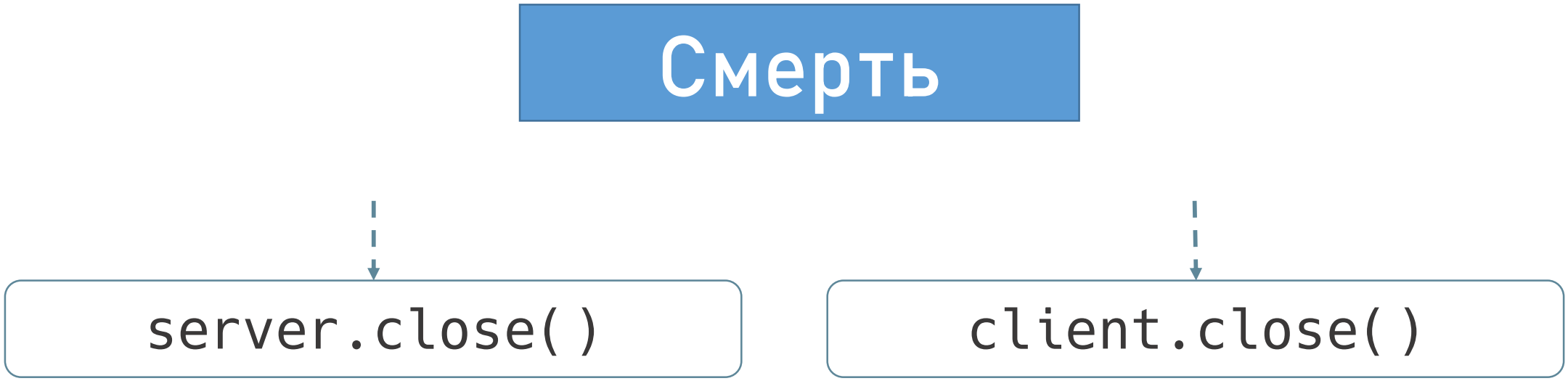
```
server.bind()
```

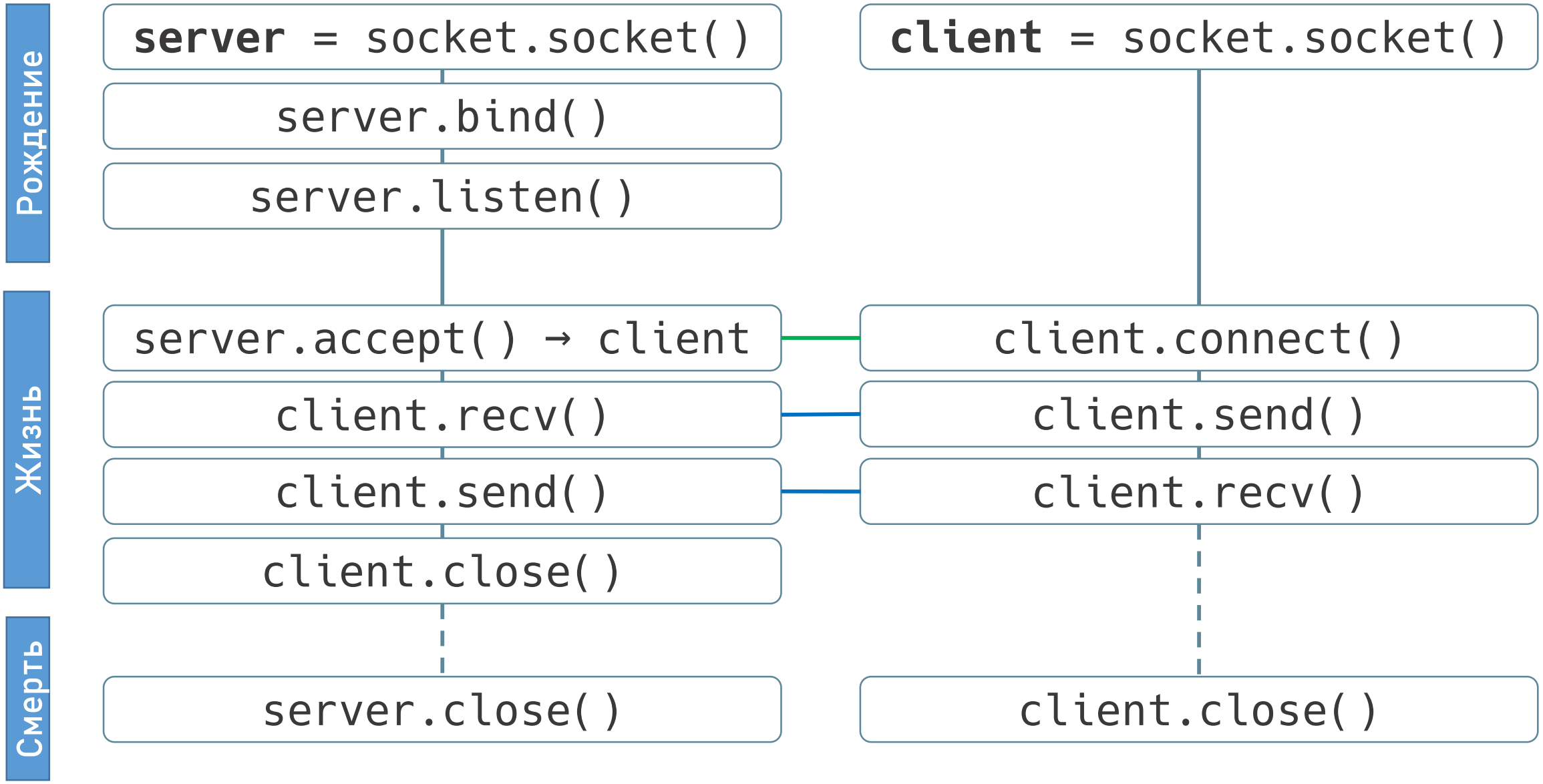
```
server.listen()
```

```
client = socket.socket()
```

Жизнь


server.accept() → client
client.connect()
client.recv()
client.send()
client.send()
client.recv()
client.close()


Смерть
server.close()
client.close()


Рождение
Жизнь
Смерть
server = socket.socket()
server.bind()
server.listen()
server.accept() → client
client.recv()
client.send()
client.close()
server.close()
client = socket.socket()
client.connect()
client.send()
client.recv()
client.close()

Что рождает socket.socket()?

UNIX: все есть файл!

Создан
дескрипт

open()
creat()
socket
accept()
pipe()

close()
select()

Глава 1

Примеры

TCP сервер

```
import socket

server = socket.socket()
server.bind(('192.168.220.5‘, 8888))
server.listen(1)
client, cl_addr = server.accept()
data_recv = client.recv(1024)
client.send(b‘Message from server’)

client.close()
server.close()
```

TCP клиент

```
import socket

client = socket.socket()
client.connect(('192.168.220.5‘, 8888))
client.send(b'Message from client')
data_recv = client.recv(1024)

client.close()
```

Глава 1

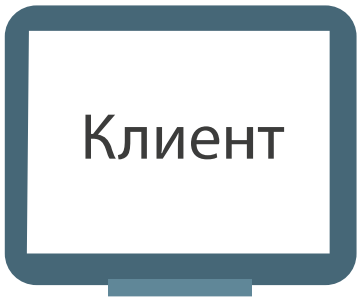
Особенности TCP сокетов

TCP сокеты

- Устанавливают соединение (тройное рукопожатие)
- Сегментация, контроль доставки и порядка сообщений
- Контроль ошибок
- Избавление от дублированных сегментов

Что может пойти не так?

Сеанс связи по протоколу TCP



Поболтаем?

Конечно, поболтаем

Отлично, подключились

Мне нужны твои данные

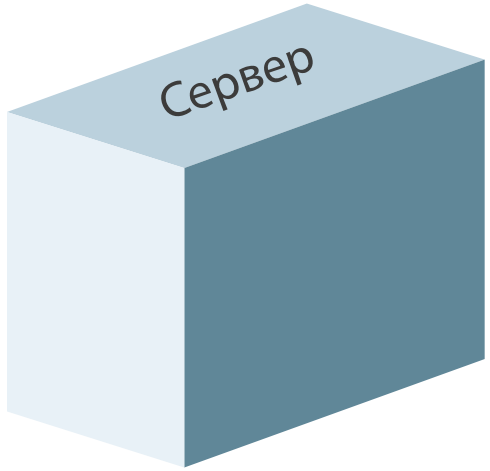
Спасибо запрос получен

Вот данные, которые ты просил

Спасибо, данные получил

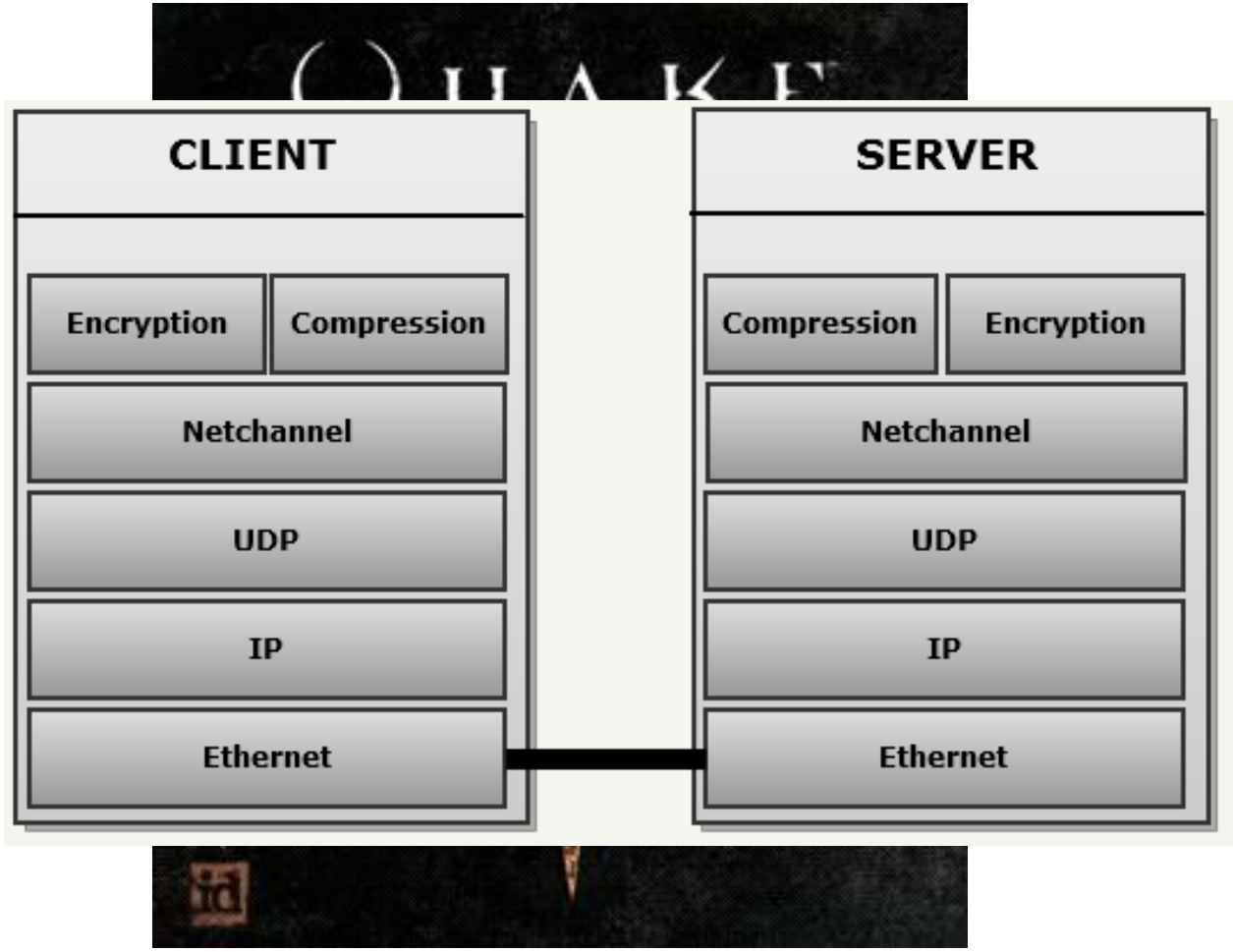
Я закончил, больше слать нечего

Я тоже закончил, спасибо



Онлайн игра: случай 1

Онлайн игра: случай 2



Глава 2

Не TCP единым: UDP

Инициализация: создание сокета

```
sock = socket.socket(
    family=…,
    type=…,
    proto=…
)
```

Домен

family=…

AF_INET	IPv4, порт
AF_INET6	IPv6, порт, метка потока, мультикаст
AF_UNIX*	Путь к файлу ('/tmp/process.s')
AF_CAN*	Имя интерфейса ('can0')
AF_BLUETOOTH*	MAC адрес, порт

Тип

type=…

SOCK_STREAM	Потоковый
SOCK_DGRAM	Датаграммный
SOCK_RAW	Сырой (нестандартные пакеты)
SOCK_SEQPACKET	Последовательные пакеты

Протокол

proto=…

IPPROTO_TCP	TCP
IPPROTO_UDP	UDP
IPPROTO_SCTP	SCTP

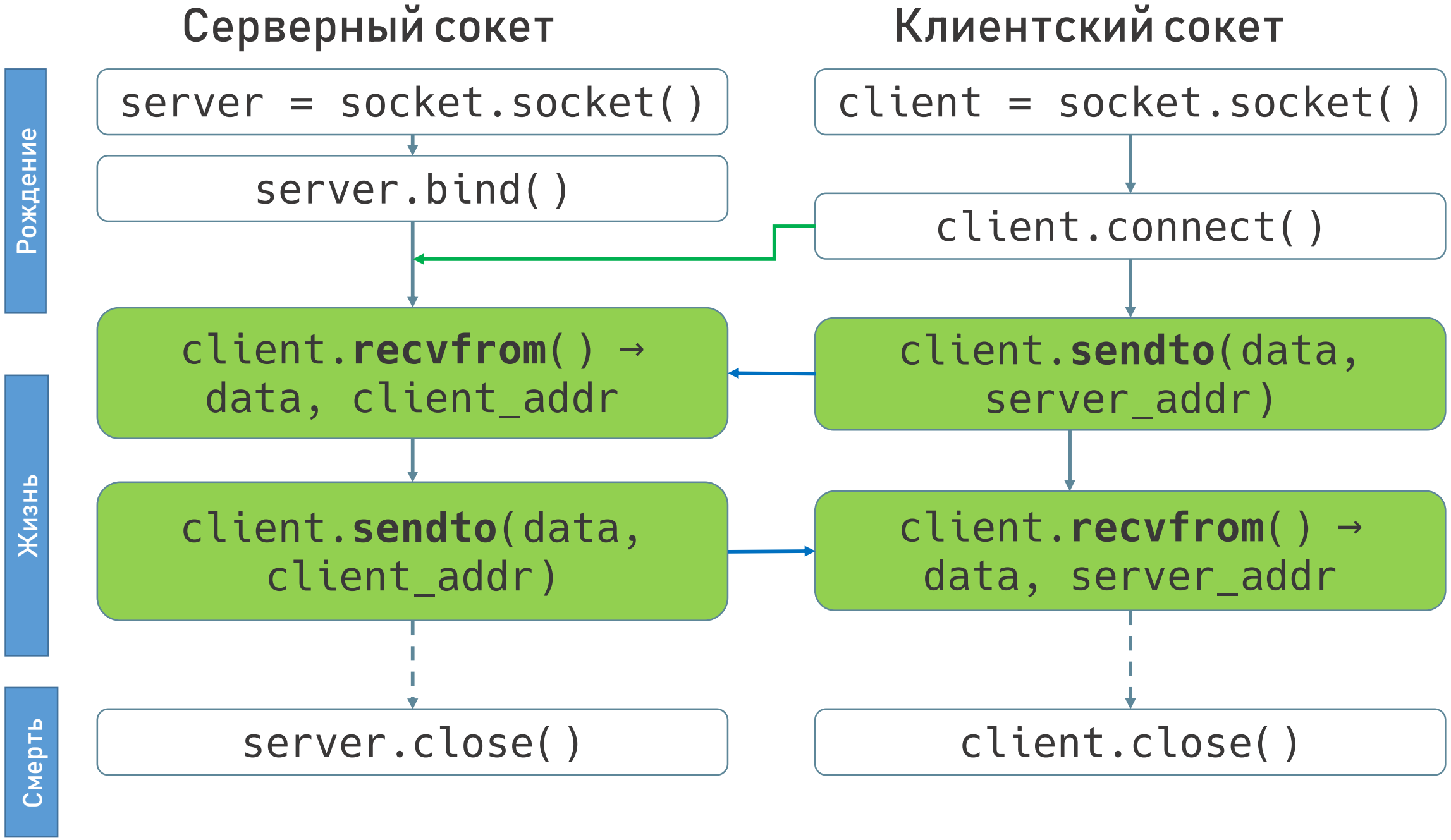
TCP и UDP сокеты

TCP

```
server = socket.socket(
    family=socket.AF_INET,
    type=socket.SOCK_STREAM,
    proto=socket.IPPROTO_TCP
)
```

UDP

```
server = socket.socket(
    family=socket.AF_INET,
    type=socket.SOCK_DGRAM,
    proto=socket.IPPROTO_UDP
)
```


Серверный сокет
Клиентский сокет
Рождение
Жизнь
Смерть
server = socket.socket()
server.bind()
client = socket.socket()
client.connect()
client.recvfrom() → data, client_addr
client.sendto(data, server_addr)
client.sendto(data, client_addr)
client.recvfrom() → data, server_addr
server.close()
client.close()

UDP сервер

```
server = socket.socket(
    family=socket.AF_INET,
    type=socket.SOCK_DGRAM,
    proto=socket.IPPROTO_UDP,
)
server.bind(('192.168.220.5', 8888))
data_recv, cl_addr = server.recvfrom(1024)
data_send = b'Message from server'
server.sendto(data_send, cl_addr)
server.close()
```

UDP клиент

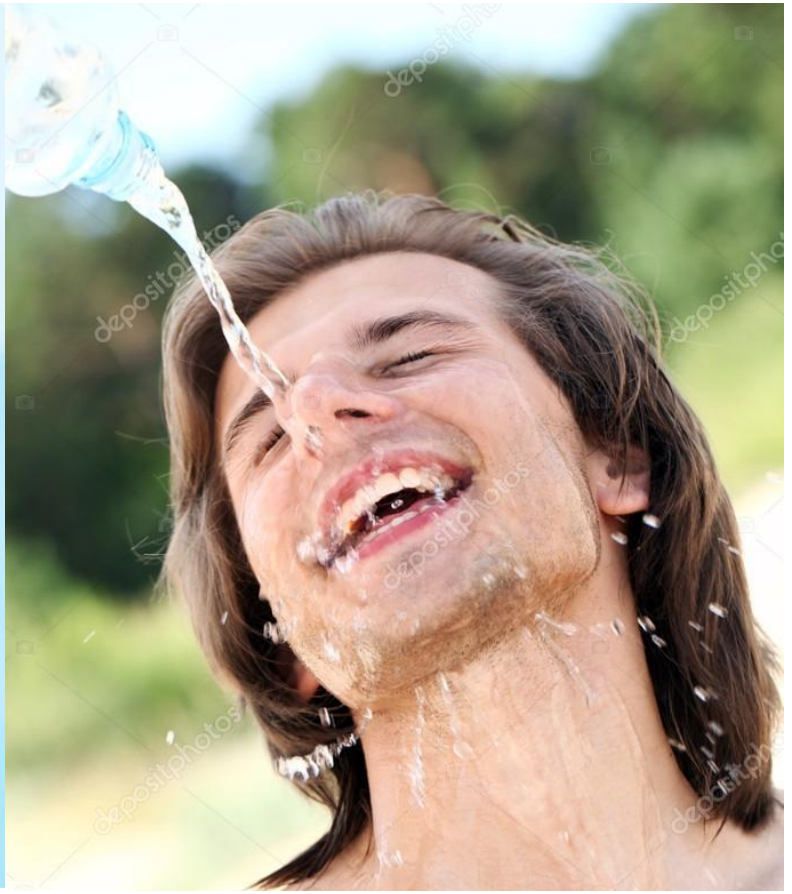
```
client = socket.socket(
    family=socket.AF_INET,
    type=socket.SOCK_DGRAM,
    proto=socket.IPPROTO_UDP,
)
client.connect((‘192.168.220.5’, 8888))
client.sendto(b’Hello’, server_addr)
data_recv, server_addr = client.recvfrom(1024)
client.close()
```

Какой сокет лучше?

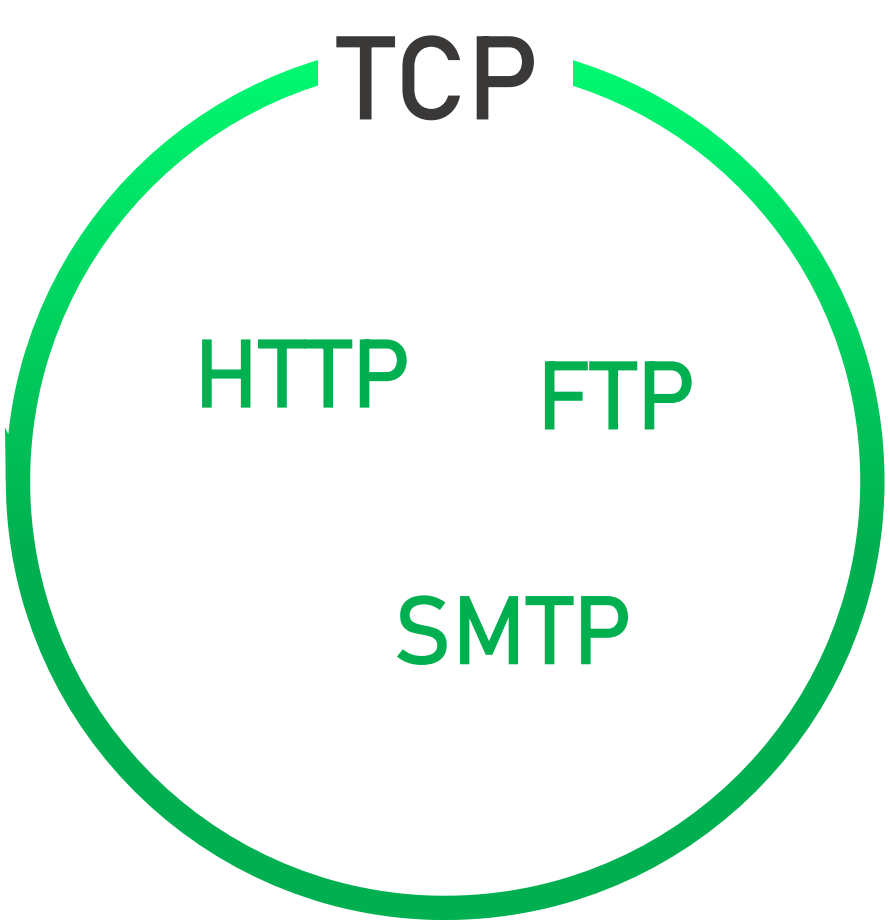
	UDP	TCP
Установка соединения	нет	да
Гарантия доставки сообщений	нет	да
Широковещательная рассылка	да	нет
Доставка сообщений в правильном порядке	нет	да

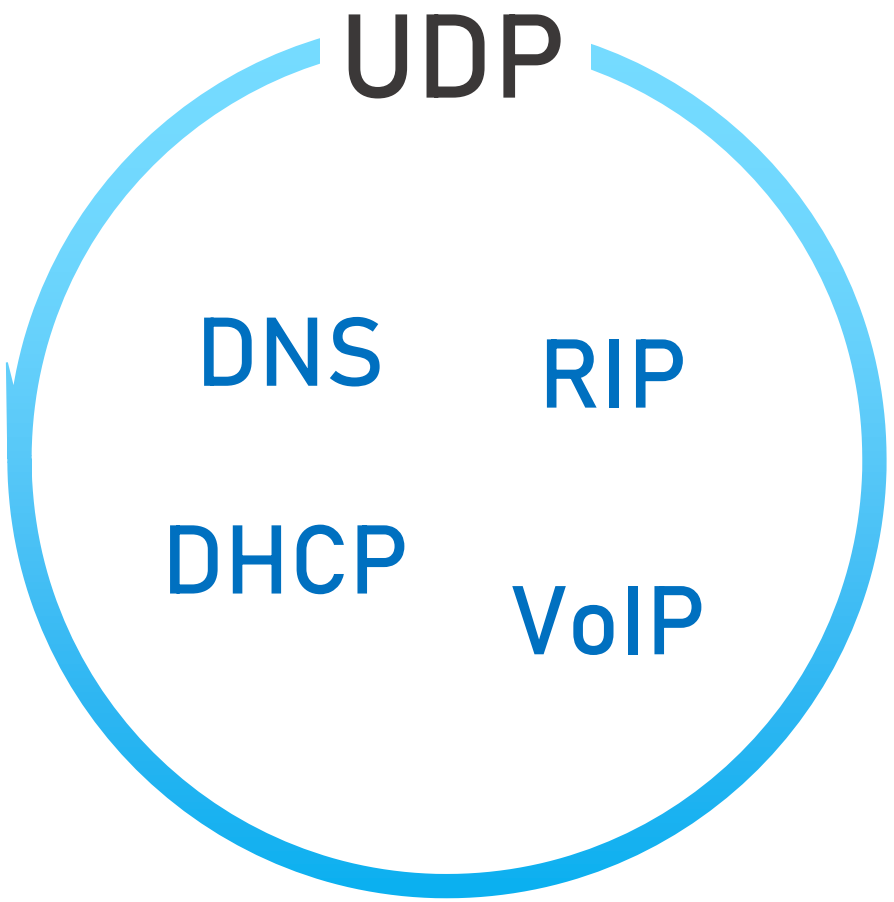
TCP

UDP



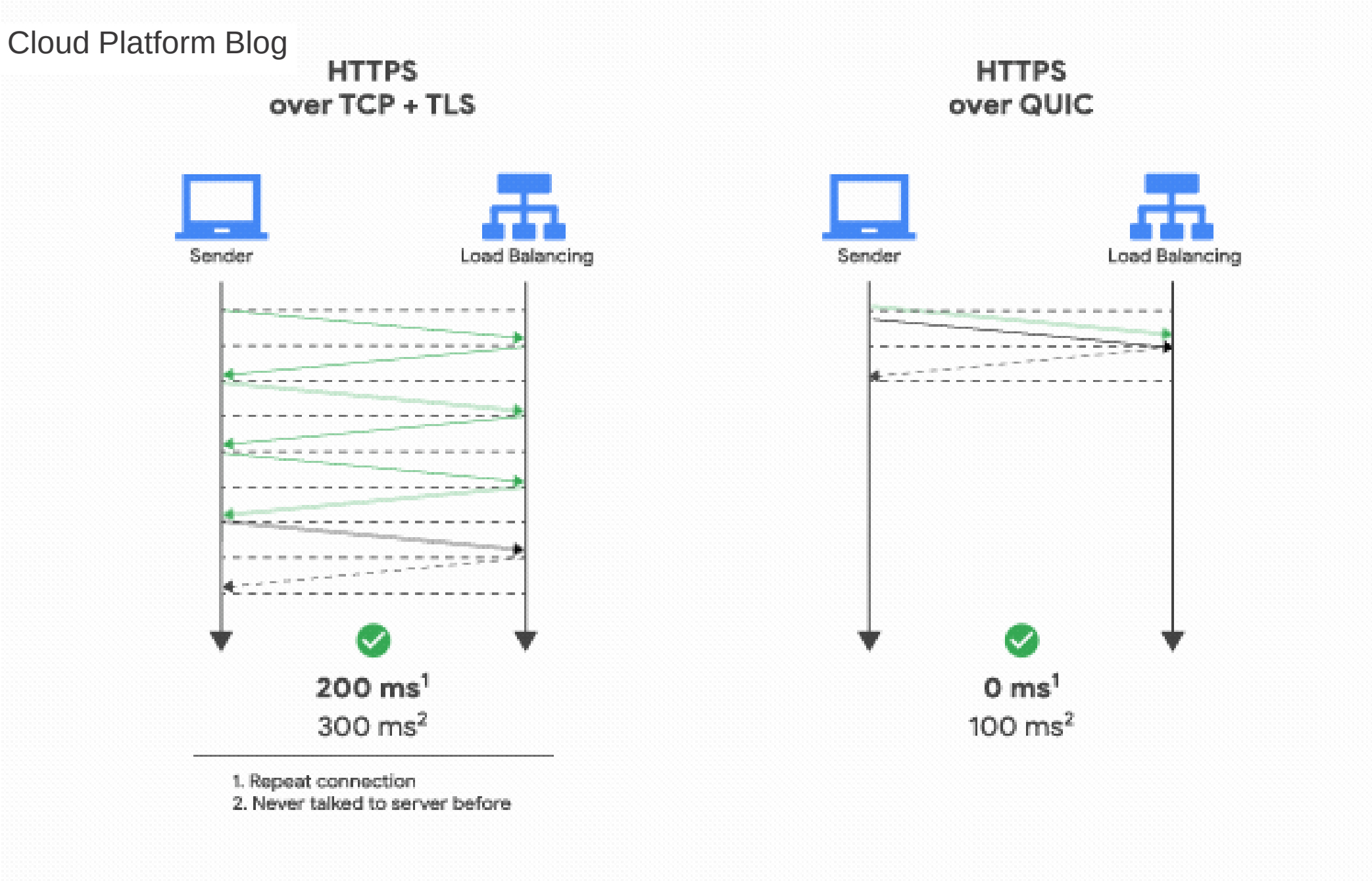
Протоколы





Quick UDP Internet Connections

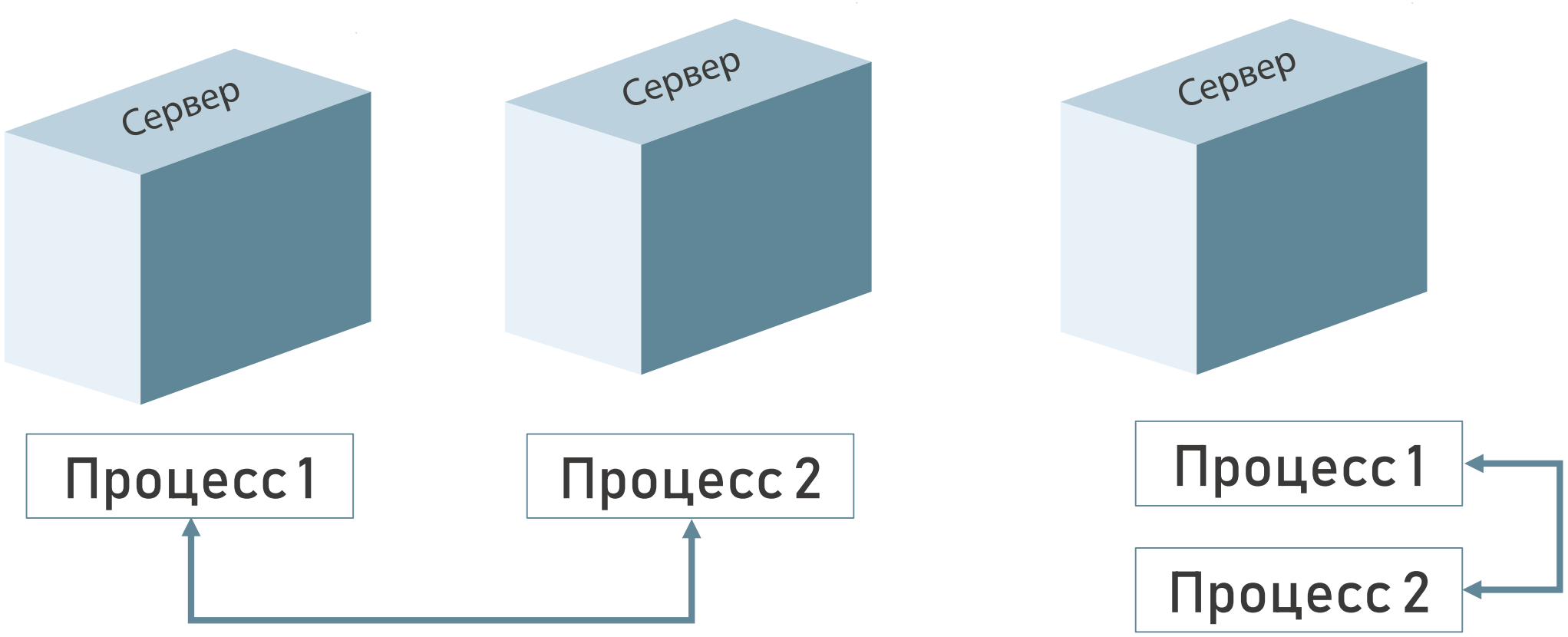
- Безопасность
- Почти мгновенное установление соединения
- Сокращение времени на переподключение к сетям для перемещающихся мобильных клиентов;


HTTPS
over TCP + TLS
Sender
Load Balancing
200 ms[1]
300 ms[2]
1. Repeat connection
2. Never talked to server before
HTTPS
over QUIC
Sender
Load Balancing
0 ms[1]
100 ms[2]

Глава 2

UNIX сокеты

Взаимодействие между сокетами



UNIX сервер

```
server = socket.socket(socket.AF_UNIX, socket.SOCK_DGRAM)
server.bind("/tmp/python_unix_sockets_example")

while True:
    message = server.recv(1024)
    if not message :
        break
    else:
        if "DONE" == message.decode('utf-8'):
            break
server.close()
```

UNIX на WINDOWS

AF_UNIX comes to Windows

Sarah

December 19th, 2017

Глава 3

Усложняем ~~жизнь~~ код

Переиспользование адреса

```
server = socket.socket()
server.bind()
server.close()
```

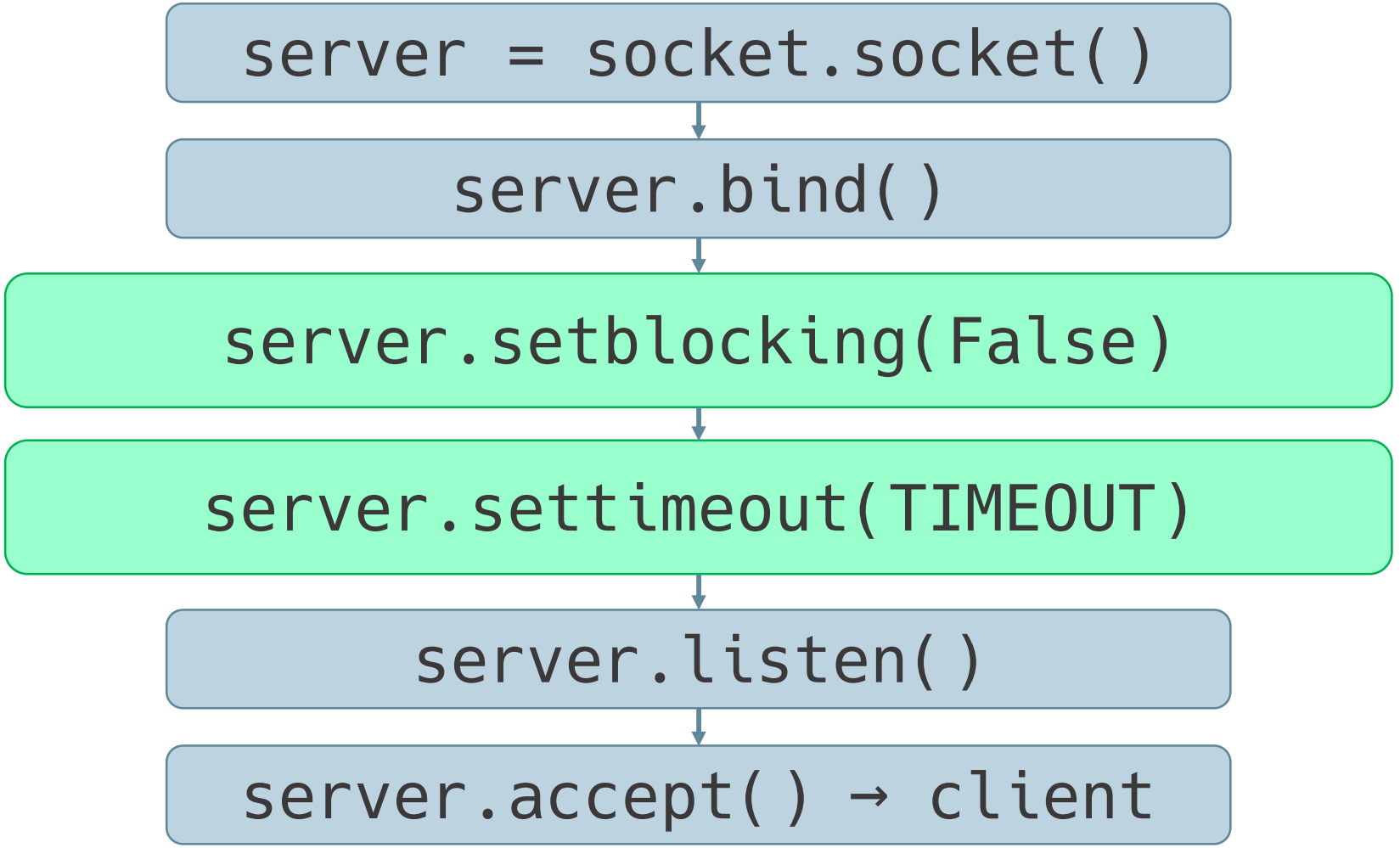
t ~ 2x60c

```
server = socket.socket()
server.setsockopt(
    socket.SOL_SOCKET,
    socket.SO_REUSEADDR,
    1
)
server.bind()
```

Блокирующая функция

```
server = socket.socket()
server.bind()
server.listen()
server.accept() → client
```

Неблокирующая с таймеров



Асинхронные сокеты

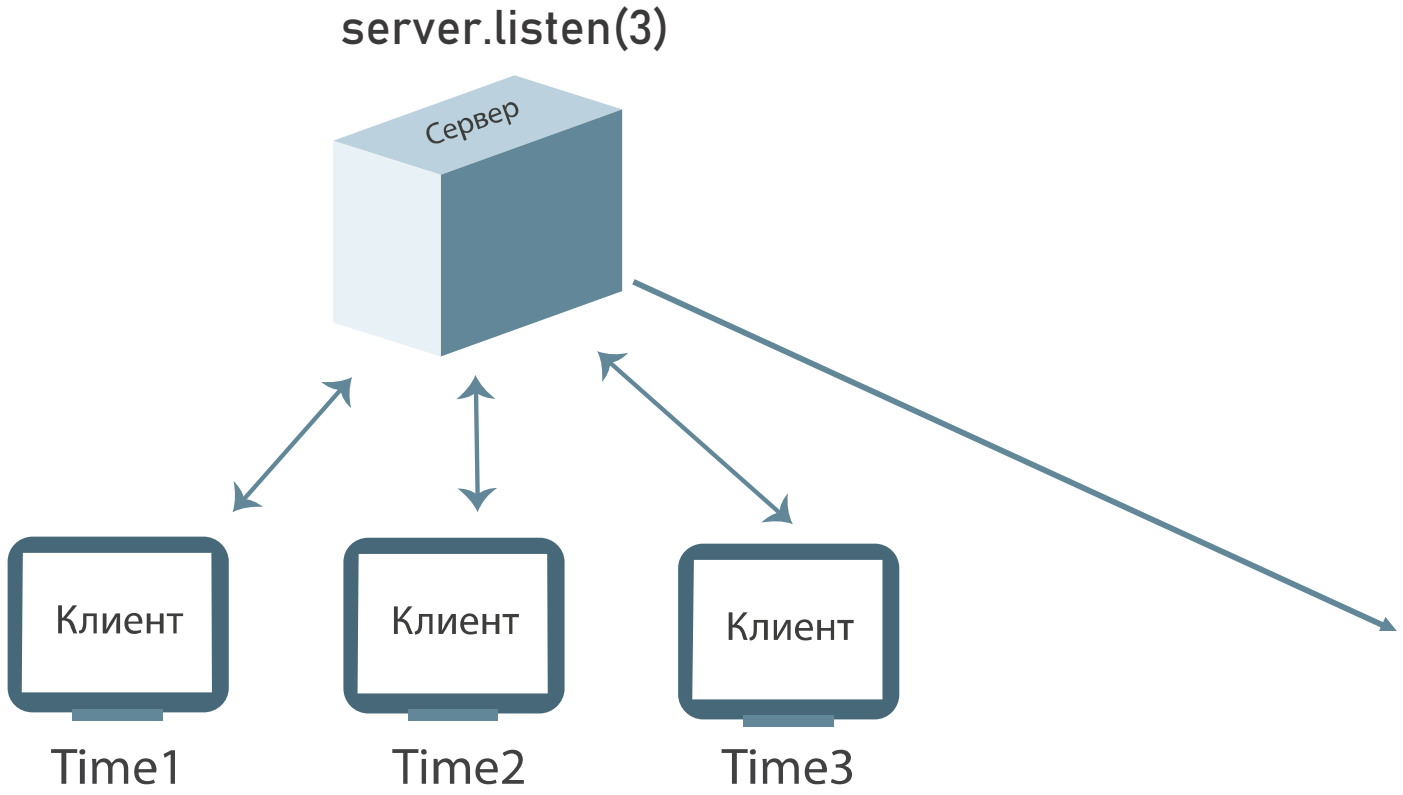
```
import asyncio
async def tcp_echo_client(message):
     reader, writer = await asyncio.open_connection(
          '127.0.0.1', 8888)
     writer.write(message.encode())
     data = await reader.read(100)
     writer.close()
     await writer.wait_closed()

asyncio.run(tcp_echo_client('Hello World!'))
```

Обработка нескольких клиентов

Обработка новых подключений

Очередь на подключение



Глава 4

Сокеты на других языках

Сервер на C/C++ (1/4)

```
#include <sys/socket.h>
…
int main()
{
    int client, server;
    struct sockaddr_in addr;
    char buf[1024];
    int bytes_read;

    server = socket(AF_INET, SOCK_STREAM, 0);
    if(server < 0)
    {
        perror("socket");
        exit(1);
    }
```

Сервер на C/C++ (2/4)

```
addr.sin_family = AF_INET;
addr.sin_port = htons(8888);
addr.sin_addr.s_addr = htonl(INADDR_ANY);
if(bind(server, (struct sockaddr *)&addr,
        sizeof(addr)) < 0)
{
    perror("bind");
    exit(2);
}
```

Сервер на C/C++ (3/4)

```
listen(listener, 1);

    while(1)
    {
        client = accept(server, NULL, NULL);
        if(client < 0)
        {
            perror("accept");
            exit(3);
        }
```

Сервер на C/C++ (4/4)

```
while(1)
        {
            bytes_read = recv(client, buf, 1024, 0);
            if(bytes_read <= 0) break;
            send(client, buf, bytes_read, 0);
        }

        close(client);
    }

    return 0;
}
```

TCP の , コネクション け け のソケットと , メッセ ジ のソケットは
される . socket() でオ プンしたソケットは , コネクション け け のソケットであり , アプリケ ションメッセ ジの には されない . socket() でオ プンしたソケットに して listen() するとコネクションの が される . TCP コネクションが されると しいソケットが られ , accept() で られたソケットのディスクリプタをることができる . このディスクリプタを ってアプリケ ションメッセ ジを することになる .

TCP ではコネクション accept() しなければサ バ ではメッセ ジを できない . accept() しなくても できる TCP コネクション の を listen() で できる .

メッセ ジの は send() , sendto() , sendmsg() , は recv() , recvfrom() , recvmsg() を する .

send() , recv() は , TCP のとき , または , UDP で connect() したときに する . つまり , コネクション である IP , ポ ト , IP , ポ トが されているときに する . sendto() , recvfrom() は , UDP で IP , ポ トが されていないときに する .

Глава 5

Итоги

Все гораздо сложнее…

SO_REUSEADDR
SO_KEEPALIVE
SO_DONTROUTE
SO_SNDBUF
SO_RCVBUF
SO_LINGER
SO_USELOOPBACK
SO_OOBINLINE

SO_SNDLOWAT
SO_RCVLOWAT
SO_SNDTIMEO
SO_RCVTIMEO
SO_TYPE
SO_ERROR
SO_BROADCAST
SO_REUSEPORT

…и это только опции…

Сокеты…

- Функциональны
- Гибки
- Универсальны

Где что применять

- TCP – для надежной передачи данных
- UDP – обмен короткими сообщениями, многоадресная рассылка
- UNIX – обмен данными на одной машине

Спасибо за внимание!