

Блокирующие ПОТОКОВЫЕ СОКЕТЫ

Курс «Информационные сети и телекоммуникации»

весенний семестр 2016 г.

кафедра Управления и информатики НИУ «МЭИ»

Потоковые сокет

- Данные передаются непрерывным потоком байт без деления на сообщения.
- Байты приходят в порядке отправления.
- Устанавливается соединение:
 - Пока оно активно, каждая сторона уверена, что другая присутствует в сети и достижима.
- Доставляются все байты
 - В противном случае соединение считается разорванным (аварийно).

Сравнительные характеристики

- **Преимущества:**

- гарантия доставки и порядка данных;
- нет дейтаграмм → их длина не ограничена;
- контроль связи по состоянию соединения.

- **Недостатки:**

- широковещательная рассылка невозможна
- приложению требуется отделять сообщения; в потоке друг от друга;
- повышенные накладные расходы.

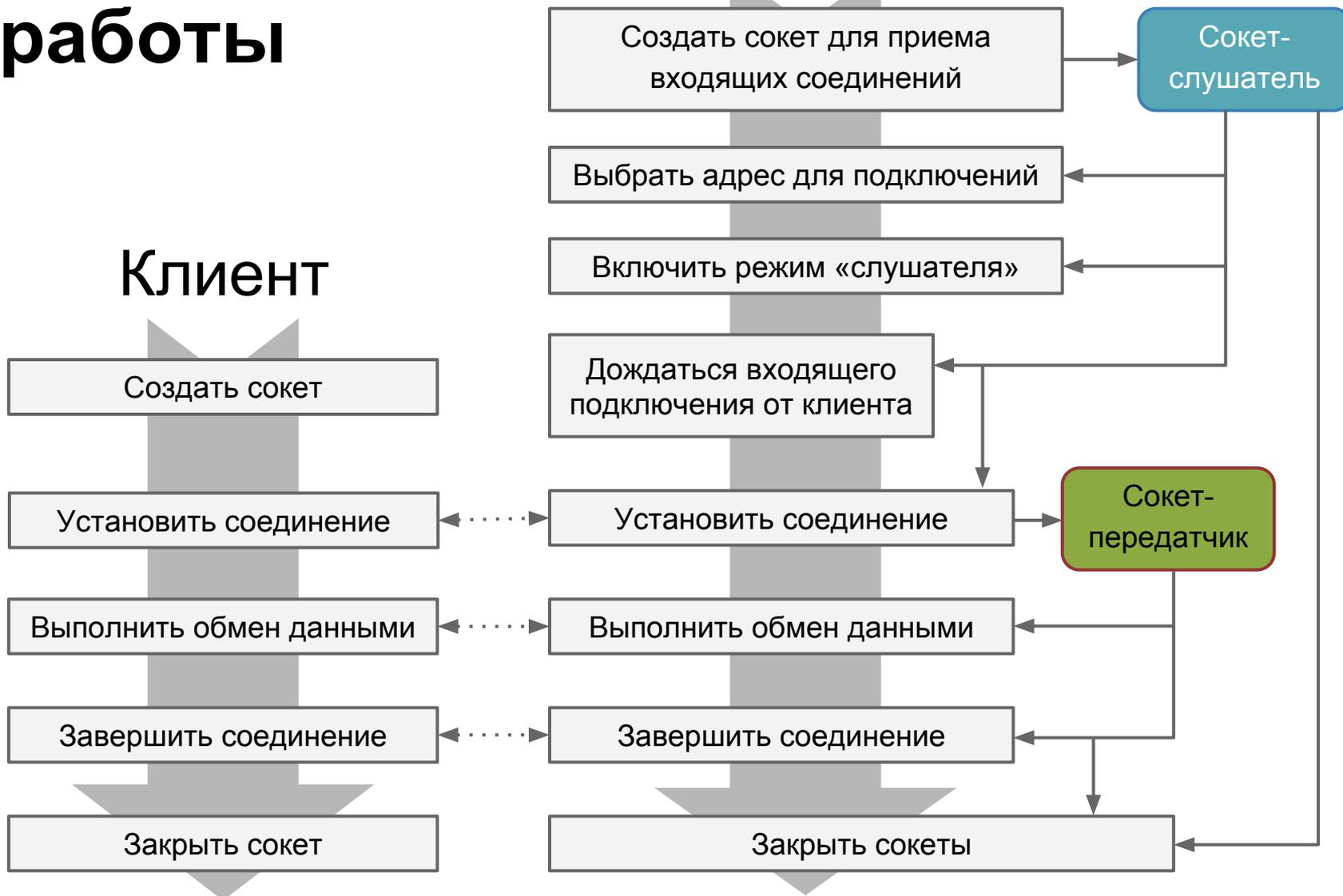
Область применения

- Для надежной доставки всех данных
- В сетях плохого качества
 - Потерь не происходит никогда,
 - но может оборваться соединение;
 - резко снижается скорость.
- При большом размере сообщений

Схема работы

Сервер

Клиент



Установка соединения клиентом

Успех: 0,

ошибка: SOCKET_ERROR (Windows) или -1 (*nix).

`int result = connect(`
`channel,` ← **Сокет-передатчик** `socket(`
`(const sockaddr*)(&server),` `AF_INET,`
`sizeof(server));` `SOCK_STREAM,`
 `IPPROTO_TCP)`

Указатель на структуру-адрес сервера `sockaddr_in` и размер этой структуры.

Подключение сервером клиента

Сокет-передатчик



```
auto client = accept(  
    listener,  
    (sockaddr*)&peer,  
    &peer_size);
```

```
socket(  
    AF_INET,  
    SOCK_STREAM,  
    IPPROTO_TCP)
```



Сокет-слушатель



Структура для записи
адреса присоединившегося
и её размер
(оба `nullptr`, если не нужны).



Перед вызовом `peer_size == sizeof(peer)`, если используется.

Включение режима «слушателя»

Необходимо до вызова `accept()`. Повторные вызовы безвредны, но не требуются.

Успех: 0,

ошибка: `SOCKET_ERROR` (Windows) или `-1` (*nix).

↓

```
int result = listen(  
    listener, backlog);
```

Сокет-слушатель

↑

Размер очереди входящих подключений
(накапливаются между вызовами `accept()`).

`SOMAXCONN` — «сделать размер наибольшим».

Потоковая отправка данных

Успех:

1. Количество успешно переданных байт данных.
2. Не превышает `count`, может быть меньше.
3. 0, если `count == 0`.

Ошибка: `SOCKET_ERROR` (Windows) или `-1` (*nix).

```
int bytes_send = send(
```

```
channel,
```

Сокет-передатчик

```
data,
```

Указатель на начало блока данных для отправки.

```
count,
```

Количество байт для отправки.

```
0);
```

Флаги настроек (см. документацию).

Потоковый прием данных

Успех:

1. Количество принятых байт данных.
2. Не превышает `max_count`, может быть меньше.
3. 0, если соединение закрыто удаленным узлом.

Ошибка: `SOCKET_ERROR` (Windows) или `-1` (*nix).

```
int bytes_received = recv(
```

```
channel,
```

Сокет-передатчик

```
data,
```

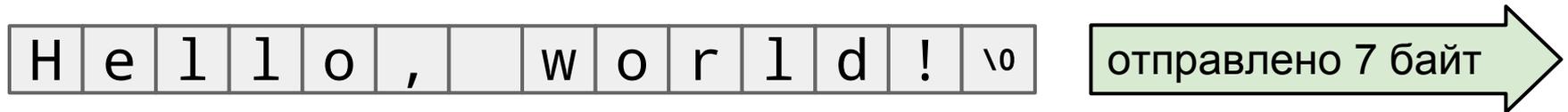
Указатель на начало блока
для записи принятых данных.

```
max_count,
```

Наибольшее число байт,
которые можно принять
и записать по адресу `data`.

```
0); ← Флаги настроек.
```

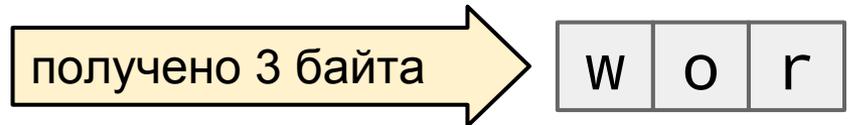
Проблемы потоковой передачи



Не все данные
удается отправить
сразу же.



Данные могут прибывать
иными порциями,
чем были отправлены.



Проблемы потоковой передачи

Данные могут прибывать иными порциями, чем были отправлены.



Решения проблем потоковой передачи

| Проблема | Решение |
|---|---|
| Не все данные удается отправить сразу же. | Отправителю досылать данные, пока не будет передано всё. |
| Данные прибывают иными порциями, чем были отправлены. | Получателю накапливать данные, пока не будет принято сообщение. |
| Теряются границы между сообщениями. | <ul style="list-style-type: none">● Зафиксировать размер сообщений;● передавать длины сообщений;● отмечать границы сообщений. |

Разделение сообщений в потоке

Фиксацией размера: очень просто, но почти не применимо.

Меткой окончания:  Получатель не может сразу выделить буфер под прием всего сообщения.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|--|---|---|---|---|---|---|----|
| Н | е | л | л | о | , | | в | о | р | л | д | ! | \0 |
|---|---|---|---|---|---|--|---|---|---|---|---|---|----|

14 байт

Указанием длины: 

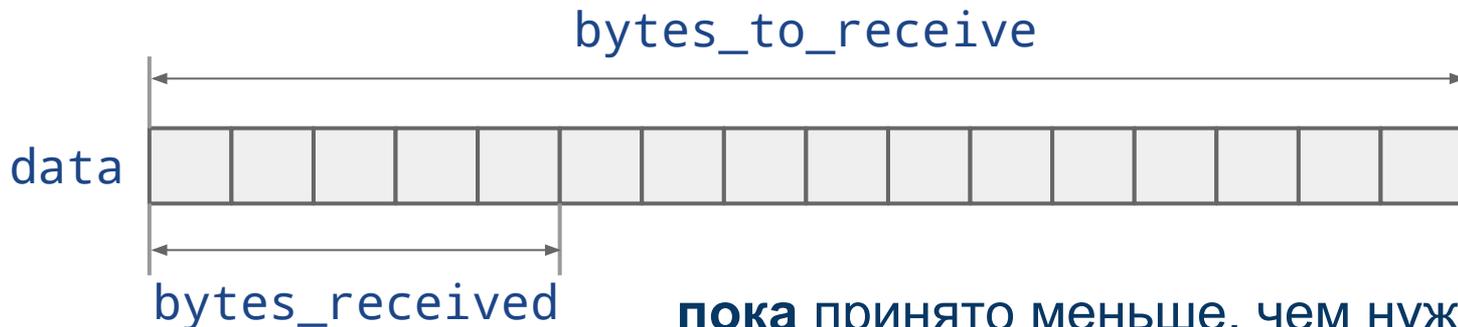
Длина может занимать несколько октетов и приходиться по частям.

| | | | | | | |
|----|----|----|----|----|----|----|
| 03 | 00 | 80 | 04 | 3f | 00 | dd |
|----|----|----|----|----|----|----|

3 байта

4 байта

Алгоритм приема данных



пока принято меньше, чем нужно:
попытаться *допринять остаток*;
запомнить, сколько стало принято.

В: Сколько байт нужно допринять?

О: Сколько осталось: `bytes_to_receive - bytes_received`.

В: Куда их нужно записать?

О: В `data`, начиная с байта `bytes_received`.

В: Как изменится алгоритм для отправки?

О: Никак!

Type, Length, Value (TLV)

- Типовая структура пакетов L7:
 - *Type* — тип пакета,
например, 1 — «запрос», 2 — «ответ».
 - *Length* — длина данных в пакете.
 - *Value* — данные переменной длины в пакете.
- *Type* и *Length* имеют фиксированный размер,
 - `bytes_to_receive` известна наперед.
- Размер *Value* зависит от *Length*,
 - например, `bytes_to_receive = Length`.
- *Type, Length* — минимальный практичный заголовок.

Завершение соединения

- Graceful shutdown —
корректное завершение (не аварийный разрыв).
- `close()` завершает соединение и закрывает сокет.
 - Доставленные, но не принятые данные теряются.
- Завершение приема или передачи:
 - `shutdown(channel, SHUT_RD)`
прекращает возможность приема сокетом;
 - `shutdown(channel, SHUT_WR)`
прекращает возможность передачи сокетом;
 - `shutdown(channel, SHUT_RDWR)`
- Windows: `closesocket()` и `SD_RECEIVE`, `SD_SEND`, `SD_BOTH`.