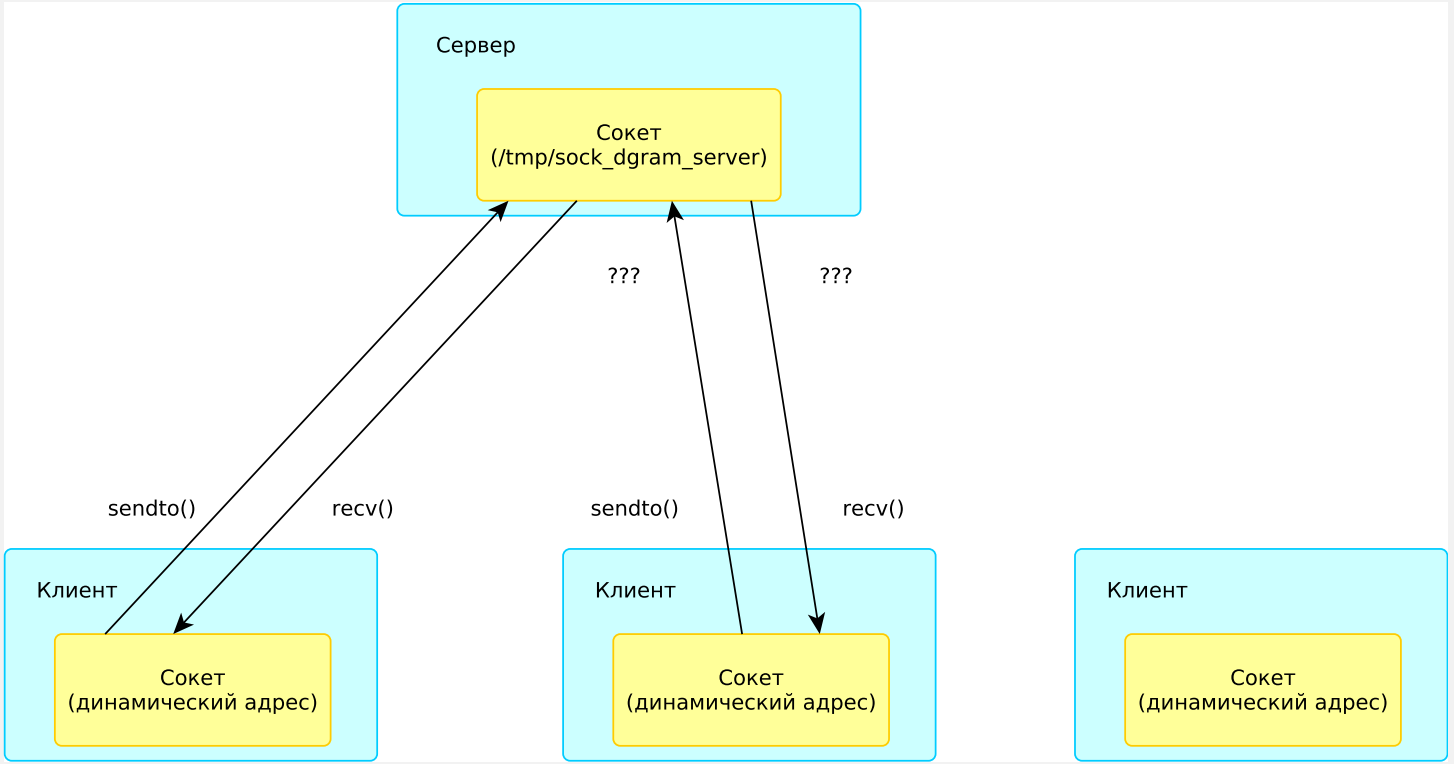
ПО сетевых устройств

Трещановский Павел Александрович, к.т.н.

26.04.19

Сервер на основе пакетного сокета



Привязка сокета к адресу

- Пример:

```
struct sockaddr_un addr;
addr.sun_family = AF_UNIX;
snprintf(addr.sun_path, sizeof(addr.sun_path), "%s", "/tmp/sock_dgram_server");

bind(sock_fd, (struct sockaddr *)&addr, sizeof(addr));
```

- Для каждого семейства адресов своя структура, описывающая адрес (`sockaddr_un`, `sockaddr_in`). При передаче в качестве аргумента `bind`, адрес приводится к абстрактному типу `sockaddr`.
- `bind` для Unix-сокета создает файл особого типа с указанным именем.
- Файл не хранит данные!
- Файл остается после закрытия сокета. Перед повторным вызовом `bind` его необходимо удалить.

Обработка запросов на сервере

- У клиента обычно нет публичного адреса.
- Сервер отправляет ответ на тот адрес, с которого пришел запрос.
- Пример:

```
struct sockaddr_un addr;
socklen_t addrlen = sizeof(addr);

recvfrom(sock_fd, rx_buf, rx_len, (struct sockaddr *)&addr, &addrlen);

sendto(sock_fd, tx_buf, tx_len, (struct sockaddr *)&addr, addrlen);
```

- `addrlen` должен быть инициализирован до вызова `recvfrom`. Он указывает размер буфера для хранения адреса. После вызова он указывает реальный размер адреса.

Отличия потокового сервера от пакетного

- Потоковый сокет требует установки соединения. Что дает соединение:
  - Сохранение порядка передачи данных за счет нумерации байтов потока.
  - Надежная доставка за счет повторной отправки неподтвержденных сегментов.
  - Детектирование разрыва соединения.
- Для AF_INET соединение устанавливается с помощью протоколу TCP, для AF_UNIX - средствами ядра Linux.
- Соединение устанавливается между 2 приложениями.
- Проблема: Каждый сокет может принадлежать 1 соединению, но серверу необходимо обслуживать несколько клиентов (несколько соединений).

Слушающий сокет

```
int listen(int sockfd, int backlog);
```

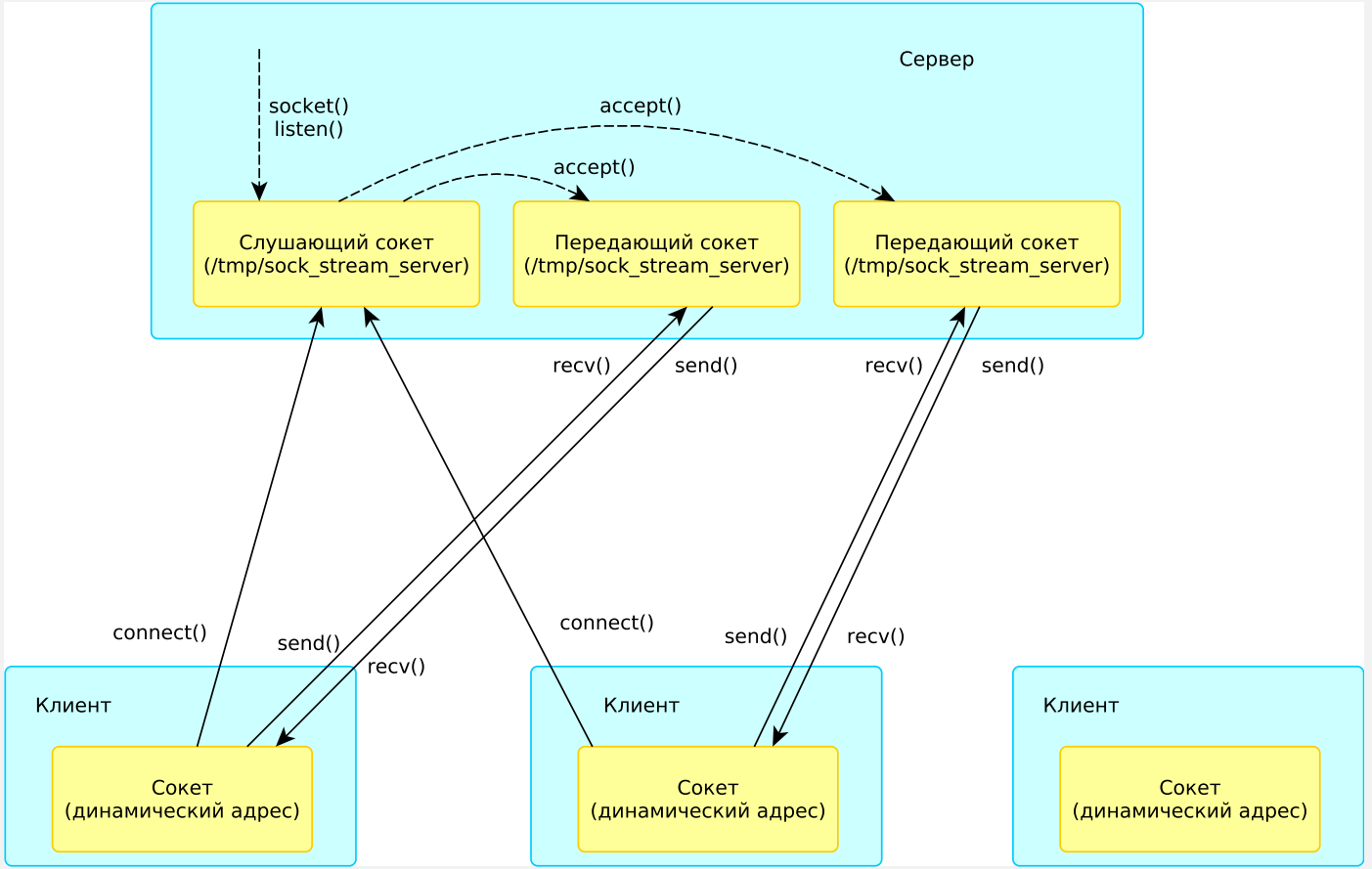
- Слушающий сокет отвечает только за прием соединений от клиентов.
- Функция `listen` переводит сокет, созданный с помощью `socket`, в слушающий режим.
- Слушающий сокет необходимо привязывать к публичному адресу.
- Аргумент `backlog` задает максимальное количество соединений, ждущих приема. При превышения этого количества запрос на соединение отклоняется.
- Прием соединения и извлечение его из очереди `backlog` осуществляется функцией `accept`.

Передающий сокет

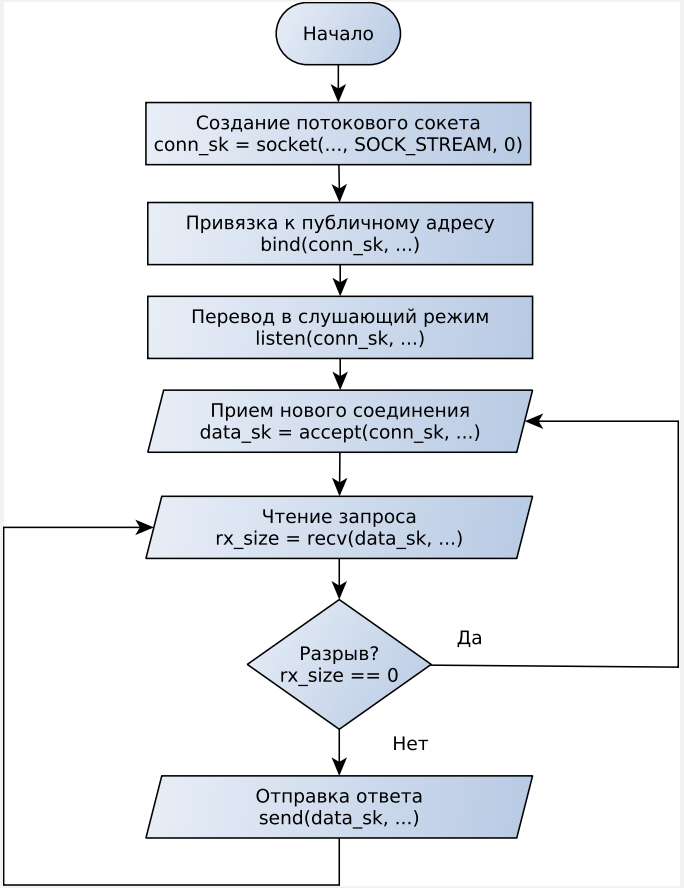
```
int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
```

- Функция `accept` возвращает дескриптор передающего сокета.
- Передача данных по каждому из установленных соединений производится через передающие сокеты. 1 передающий сокет = 1 соединение.
- В приложении может существовать несколько передающих сокетов, созданных из одного слушающего сокета.
- `accept` - блокирующая функция, которая возвращается только после приема нового соединения.
- Передающий сокет уже привязан (вызывать `bind` не нужно).
- Передача и прием производятся функциями `send` и `recv`.
- Когда клиент разрывает соединение, `recv` на передающем сокете сервера возвращает 0.

Сервер на основе потокового сокета



Алгоритм сервера с одним передающим потоком



Упражнения

- Разработать сервер на основе пакетного Unix-сокета, который принимает запросы на адресе /tmp/sock_dgram_server, и для каждого запроса выводит сообщение в терминал. Анализировать запросы и отправлять ответы не требуется.
- Разработать сервер на основе потокового Unix-сокета, который принимает соединения на адресе /tmp/sock_stream_server, и для каждого соединения выводит сообщение в терминал. Соединение следует сразу же закрывать.