ПО сетевых устройств

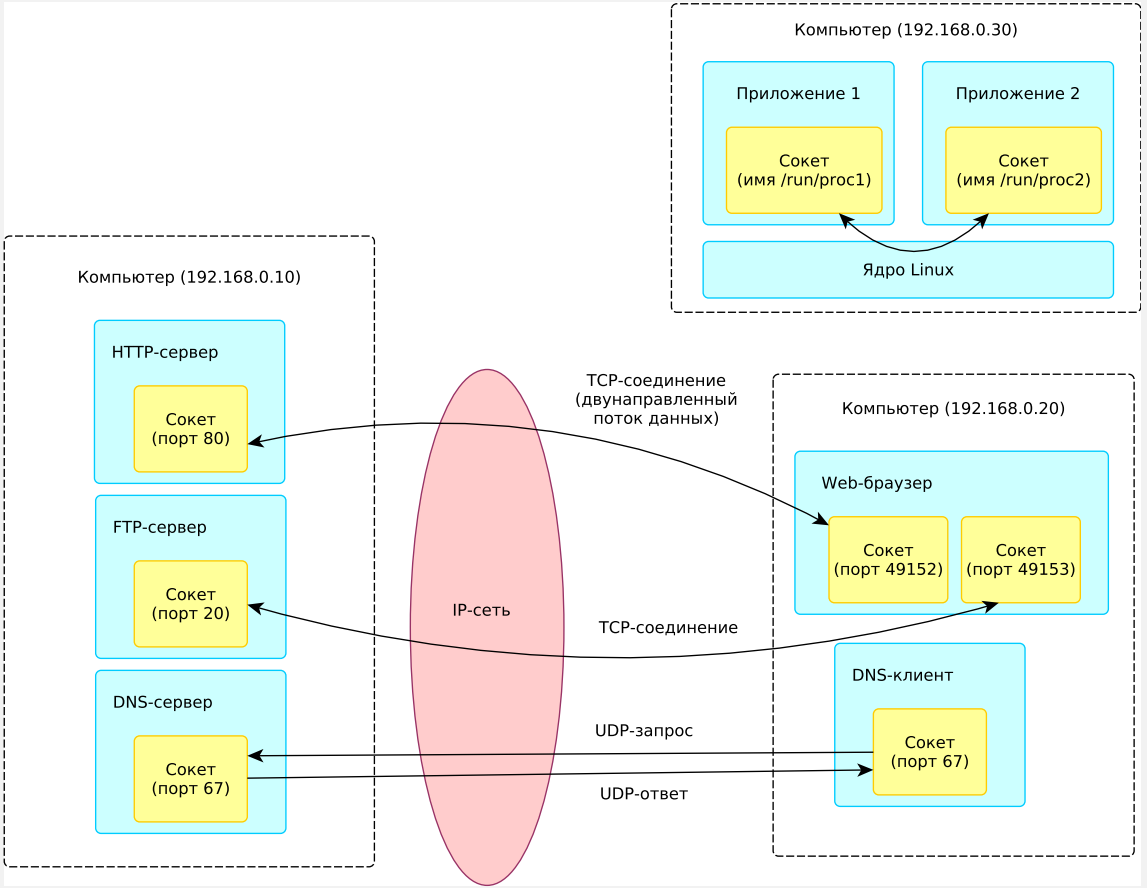
Трещановский Павел Александрович, к.т.н.

19.04.19

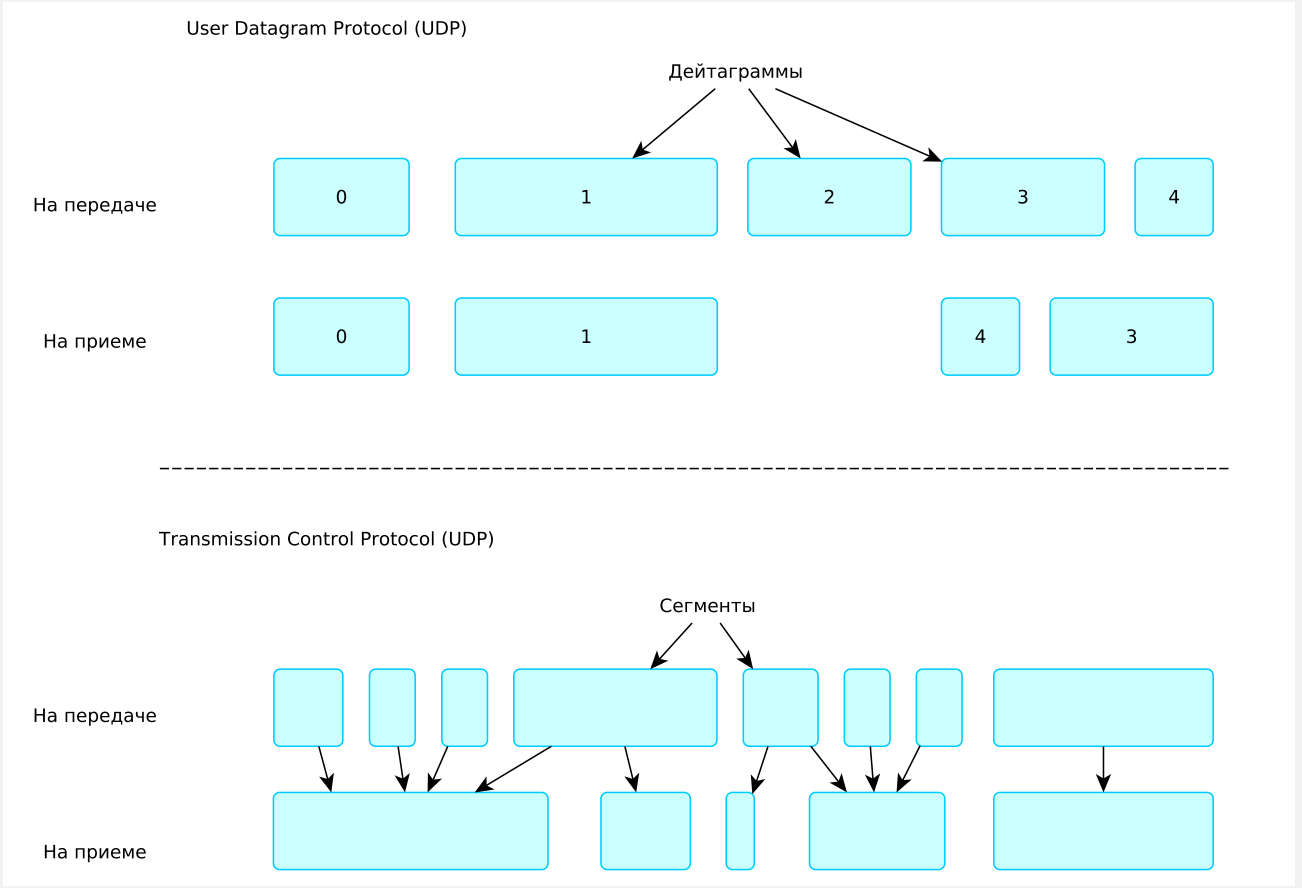
Сетевые сокеты

```
int sock_fd;
struct sockaddr_in addr;
unsigned char txbuf[100], rxbuf[100];

/* Инициализация addr и buf не показана. */

sock_fd = socket(AF_INET, SOCK_STREAM, 0);
connect(sock_fd, (struct sockaddr *)&addr, sizeof(addr));

write(sock_fd, txbuf, sizeof(txbuf));
read(sock_fd, rxbuf, sizeof(rxbuf));

send(sock_fd, txbuf, sizeof(txbuf), 0);
recv(sock_fd, rxbuf, sizeof(rxbuf), 0);

close(sock_fd);
```

Средства передачи данных между процессами



TCP vs UDP


User Datagram Protocol (UDP)
Дейтаграммы
На передаче
0
1
2
3
4
На приеме
0
1
4
3
Transmission Control Protocol (UDP)
Сегменты
На передаче
На приеме

Пакетная и потоковая передача данных

- При пакетной передаче границы сообщений сохраняются.
- UDP обеспечивает пакетную передачу.
- При потоковой передаче границы сообщений не сохраняются. Возможно объединение исходных сообщений в одно, разбиение исходного сообщения на несколько, а также любая комбинация этих операций.
- TCP обеспечивает потоковую передачу.
- Зачем вообще менять границы сообщений? Для некоторых приложений границы не существенны (например, для передачи файла). При этом использование более крупных сообщений снижает нагрузку на процессоры, а более мелких сообщений обеспечивает передачу через локальные сети с маленьким максимальным размером кадра.
- Потоковая передача данных требует установления соединения.

Что мы хотим от сетевых сокетов?

- Поддержка нескольких видов адресации: IP-адрес + порт для связи между процессами на разных компьютерах, строковые имена - между процессами на одном компьютере.
- Поддержка пакетной (с сохранением границ сообщений) и потоковой (без сохранения границ сообщения) передачи данных.
- Средства установки и разрыва соединения.
- Средства передачи и приема данных.
- Общий программный интерфейс для сетевого и не сетевого обмена.
- Сокрытие деталей реализации протоколов транспортного уровня в ядре операционной системы.

Классификация сокетов

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

- domain - семейство адресов:
  - AF_INET - адресом сокета является пара (IP-адрес, порт),
  - AF_UNIX - адресом сокета является строковые имя.
- type - тип сокета:
  - SOCK_DGRAM - пакетный (дейтаграммный) сокет (сохраняет границы сообщений, не гарантирует доставку, допускает передачу данных без установки соединения),
  - SOCK_STREAM - потоковый сокет (не сохраняет границы сообщений, гарантирует доставку, передача данных только после установки соединения).
- protocol - сетевой протокол (`IPPROTO_TCP`, `IPPROTO_UDP`, 0 - протокол по умолчанию).

Программный интерфейс сокетов

```
int socket(int domain, int type, int protocol);
int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
ssize_t send(int sockfd, const void *buf, size_t len, int flags);
ssize_t sendto(int sockfd, const void *buf, size_t len, int flags,
               const struct sockaddr *dest_addr, socklen_t addrlen);
ssize_t recv(int sockfd, void *buf, size_t len, int flags);
int close(int fd);

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

Клиентские и серверные приложения

- Пример: Web-сервер и Web-браузер.
- Будем называть клиентом то приложение, которое устанавливает соединение, передает запросы и принимает ответы.
- Будем называть сервером то приложение, которое принимает соединения, принимает запросы и передает ответы.
- Сервер имеет известный публичный адрес, клиенту адрес сокета (не путать с IP-адресом) назначается динамически.
- Клиент, использующий пакетный сокет, вызывает функции `sendto` и `recv`.
- Клиент, использующий потоковый сокет, вызывает функции `connect`, `send` и `recv`.
- Для назначения клиентскому сокету динамического адреса используется вызов `bind` со специальными аргументами (см. инструкцию к задаче).

Адресация Unix-сокетов

- Адрес Unix-сокета - строковое имя. Точнее - имя файла.
- Виртуальный файл с таким именем существует, но данные в нем не хранятся. Его назначение - идентификация сокета.
- Адрес задается структурой типа `struct sockaddr_un`:

```
struct sockaddr_un {
       sa_family_t sun_family;               /* AF_UNIX */
       char        sun_path[108];            /* pathname */
};
```

- Пример использования:

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

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

- `struct sockaddr` - абстрактный адрес. Конкретные адреса (`struct sockaddr_un` и др.) приводятся к этому типа при передаче в функцию.

Прием и передача данных через Unix-сокеты

```
ssize_t recv(int sockfd, void *buf, size_t len, int flags);
```

- Для пакетного сокета `recv` всегда принимает целый пакет. Буфер должен быть не меньше размера принимаемого пакета. Возвращаемое значение - размер пакета или индикация ошибки (-1).
- Для потокового сокета `recv` читает все имеющиеся данные, помещающиеся в данный буфер. Возвращаемое значение - количество прочитанный байтов или индикация ошибки (-1). Код 0 означает, что соединение было закрыто.
- `recv` - блокирующая функция. Она завершится только тогда, когда будут прочитаны данные. Если данных нет, то исполнение приложения блокируется до их появления.

Упражнения

- Написать программу, которая отправляет учебному серверу запрос на чтение строки из ячейки 0, принимает ответ, содержащий эту строку, и выводит полученное значение в терминал. Учебный сервер хранит массив ячеек и обрабатывает клиентские запросы по адресу /tmp/sock_dgram_server. Клиентское приложение может отправлять запросы на чтение и запись значений этих ячеек. Протокол описан в инструкции к практическому заданию.