Сети: протоколы и вызовы

Локальная передача данных (Ethernet)

- В пределах одной сети
- Доставка не гарантируется
- Знания:
  - MAC-адрес
  - ARP-таблица
  - ARP-запрос/ответ
  - ARP-атака
  - MTU, jumbo-frame

Межсетевая передача (IP)

- Передача маршрутизатору — локальная в каждой сети
- Доставка не гарантируется
- Знания:
  - IP-адрес
  - Маска сети
  - Таблица маршрутизации
  - Маршрутизатор
  - NAT

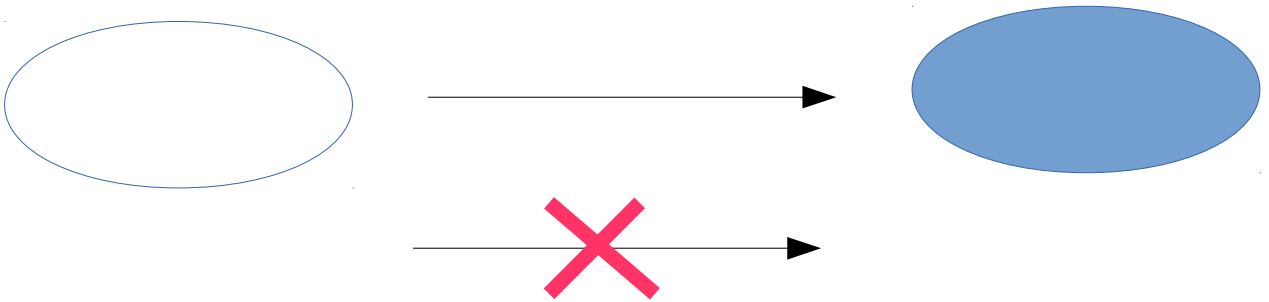
Инструменты

- wireshark (unix/win gui)/tshark (unix/win-cli)
  - Разбирает и анализирует содержимое пакета, для старых версий есть эксплойты.
- tcpdump (unix-cli), windump (win-cli)
  - Содержимое пакета как есть, анализ лишь простых протоколов (DNS, DHCP, …)

TCP/UDP/SCTP

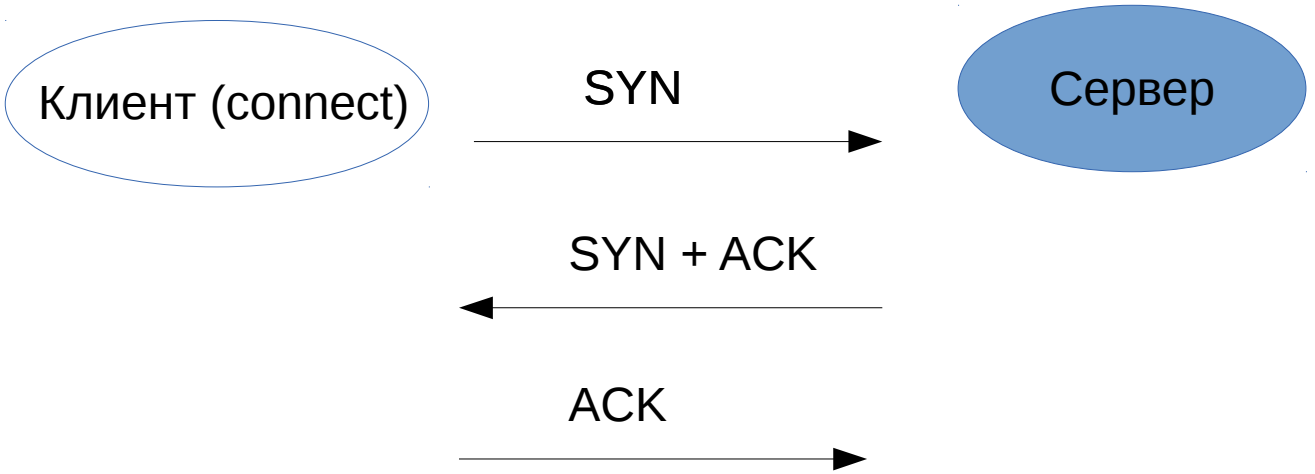
- UDP:
  - Без установления соединения, посылка сообщений (датаграмм)
- TCP:
  - Соединение (3-way), потоковая передача, контроль целостности
- SCTP: «улучшенный» TCP:
  - Соединение (4-way)
  - Многопоточность
  - Несколько интерфейсов
  - Сохраняются границы сообщений

UDP



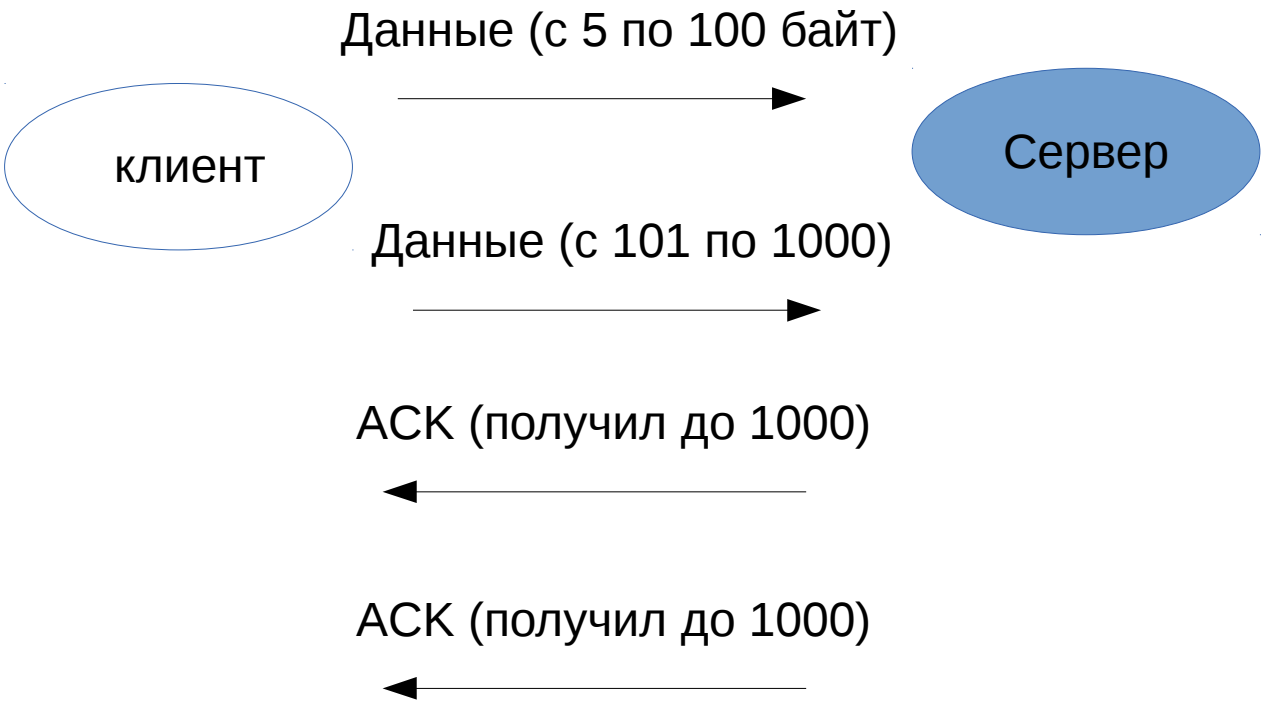
Если пакет не доставлен, то приложение должно о пересылке позаботиться само

TCP: установление соединения



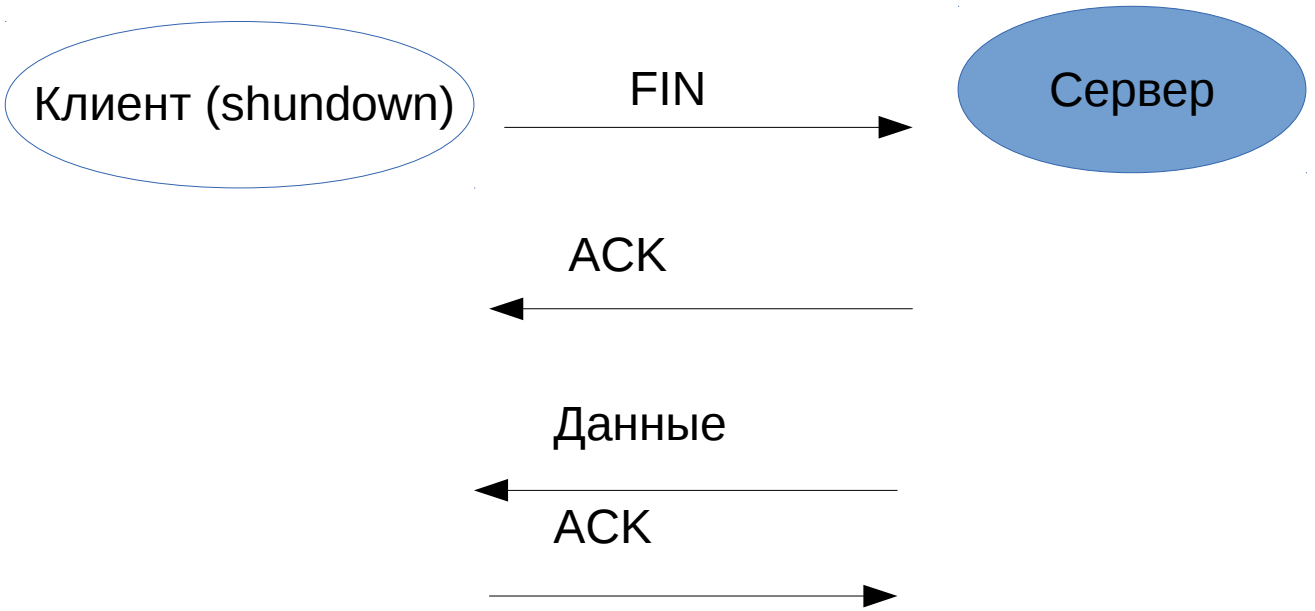
- SYN: флаг, говорящий о том, что хотим установить соединение
- ACK: флаг подтверждения получения данных

TCP: передача данных



- Если клиент посылал еще данные (с 1001), то по ответу (получено до 1000) поймет, что данные не дошли и перешлет их еще раз
- Возможно накопление данных, переразбивка на пакеты
- Размер окна (window size): сколько данных готов принять получатель

TCP: закрытие соединения

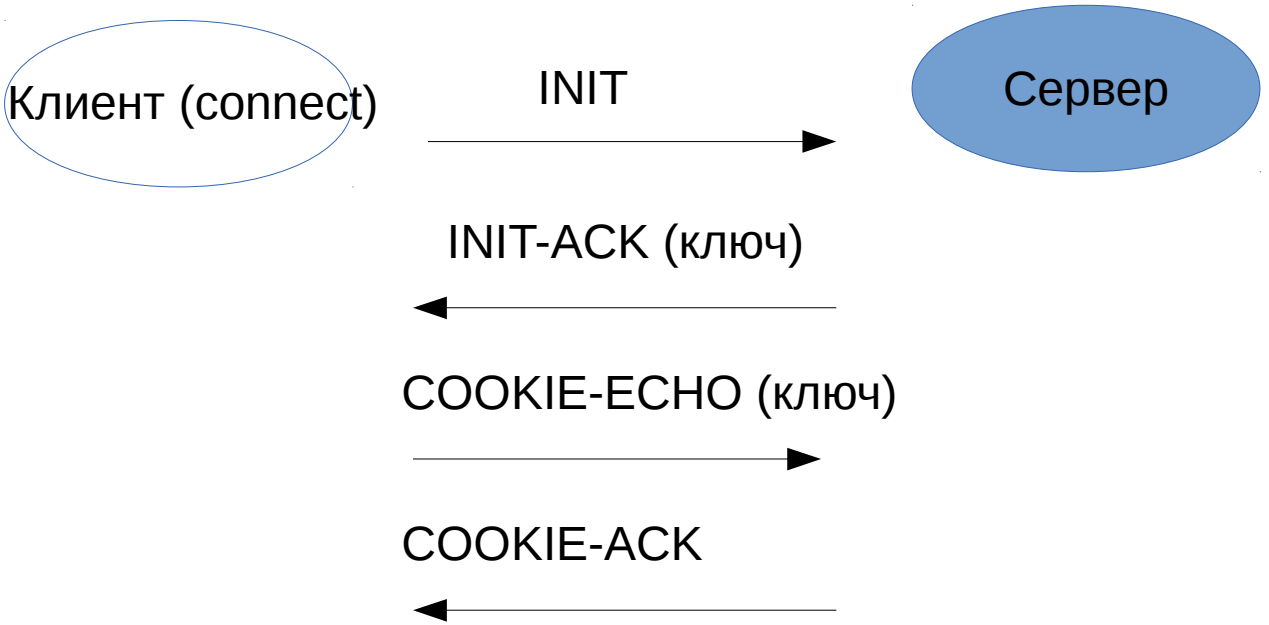


- Shutdown: для любого из направлений (можно сразу для обоих), в нашем примере клиент закрыл себе запись (отправку) данных.

SYN-атака на TCP

- Сервер получает SYN-пакет и готовится принять соединение (выделяет место в очереди, буфер под данные,...), но что если злоумышленник и не собирается соединение устанавливать:
  - Фальшивый обратный адрес
  - Ресурсы на отправку — ничтожны по сравнению с затратами сервера

SCTP: установление соединения



- 4 типа сообщений (а не комбинации флагов, как TCP)
- В COOKIE-ECHO должен придти ключ, который сервер должен признать (серверу не обязательно помнить ключ)

TCP vs UDP

TCP:

- Затраты на установление соединения (три пакета и еще ни одной передачи данных)
- Затраты на поддержание соединения (время от времени надо посылать пакеты, чтобы быть уверенным, что все хорошо)
- Надежность
- Более сложный (по сравнению с UDP)

UDP:

- Т.к. не надо устанавливать соединение, то можно передавать данные нескольким адресатам (broadcast/netcast/multicast)
- «Противоположен» TCP по остальным пунктам

Примеры

DNS:

- Имя → IP-адрес, IP-адрес → Имя
- Если сервер не знает ответ, то сам шлет запросы:
  - Вышестоящему DNS-серверу
  - Корневому DNS-серверу

Протоколы:

- Клиент ↔ Сервер: UDP
  - Время ответа сервера неизвестно, не нужно тратить пакеты на соединение (открытие, поддержку в рабочем состоянии,...)
  - Запрос и ответ — короткие
- Сервер ↔ Сервер (передача зоны): TCP
  - Данных много, требуется гарантия передачи

Примеры для TCP

- HTTP/FTP — TCP:
  - Большие объемы данных
  - Целостность файлов
- Работа с почтой:
  - Между серверами (SMTP: TCP)
  - Между пользователем и сервером (IMAP/POP3/SMTP: TCP)

Примеры для UDP

- DHCP — тоже UDP:
  - Сервер не известен
  - Запрос/ответ небольшие
- TFTP — UDP:
  - «Эмуляция» работы с жестким диском (пакеты по 512 байт)
  - В случае чего — можно и перезагрузиться (требуется только на начальном этапе)

Вызовы для TCP

Сервер:

- socket (Тип коммуникаций, Тип передачи, Протокол)
- bind
- listen
- accept
- read/write/recv/send
- shutdown
- close

Клиент:

- socket
- connect
- ...

Вызовы для SCTP

Сервер:

- socket (Тип коммуникаций, SOCK_STREAM, IPPROTO_SCTP)
- bind
- listen
- accept
- read/write/recv/send/sctp_sendmsg/sctp_recvmsg
- shutdown
- close

SCTP: пример

```
int listenSock, connSock, ret;
struct sockaddr_in servaddr;

listenSock = socket( AF_INET, SOCK_STREAM, IPPROTO_SCTP );

  bzero( (void *)&servaddr, sizeof(servaddr) );
  servaddr.sin_family = AF_INET;
  servaddr.sin_addr.s_addr = htonl( INADDR_ANY );
  servaddr.sin_port = htons(1234);
bind( listenSock, (struct sockaddr *)&servaddr, sizeof(servaddr) );
listen( listenSock, 5 );
connSock = accept( listenSock, (struct sockaddr *)NULL, (int *)NULL );
sctp_sendmsg( connSock,
                        (void *)buffer, (size_t)strlen(buffer),
                        NULL, 0, 0, 0, 1010, 0, 0 );
```

Вызовы для UDP

Сервер:

- socket(,SOCK_DGRAM, IPPROTO_UDP)
- bind
- sendto/recvfrom
- close