Сокеты и сети. Часть 2

АКОС #18

OSI

Open Systems Interconnections

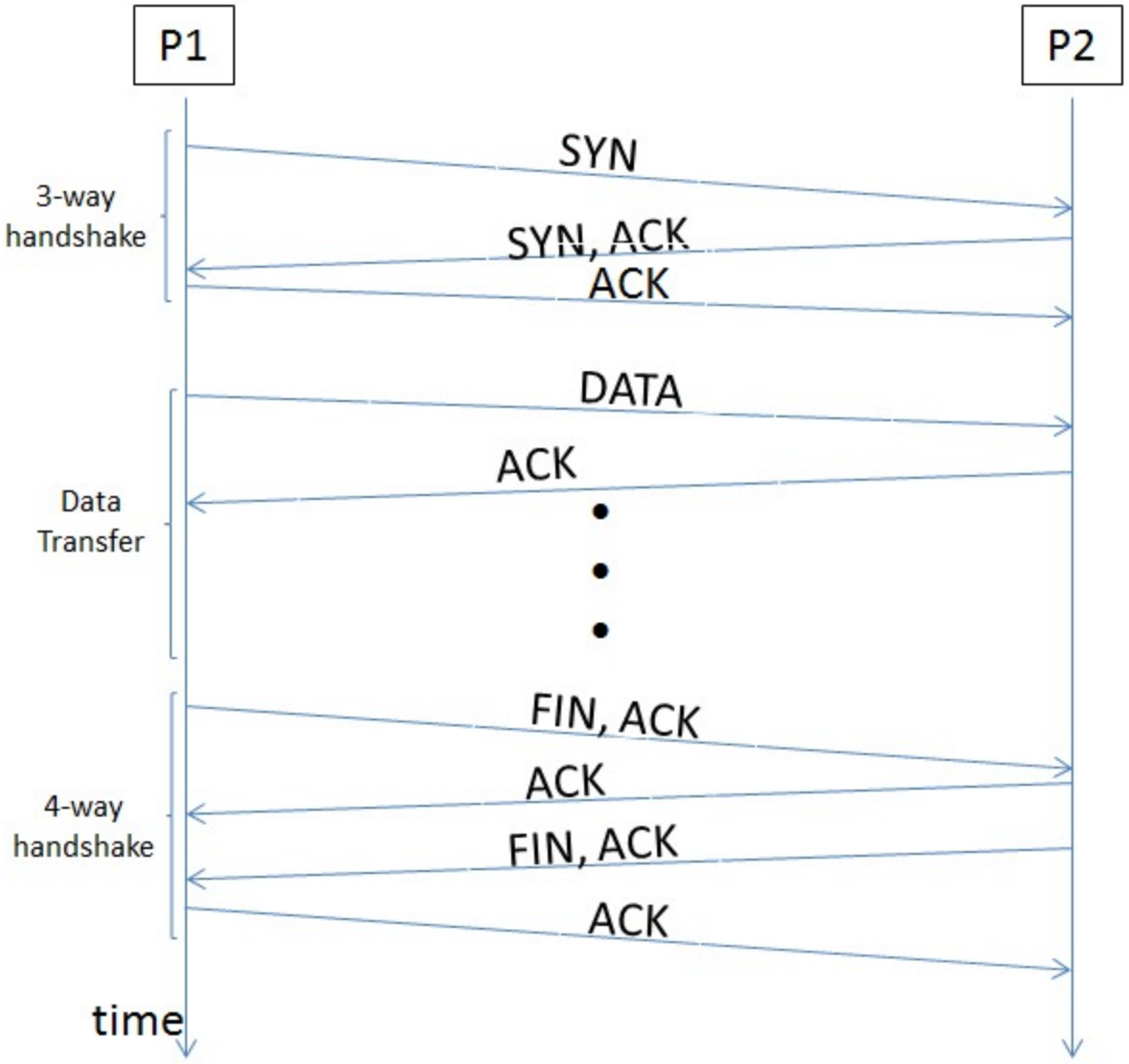
Стек TCP/IP		Модель OSI (Open Systems Interconnections)	Примеры
Уровень процессов		Уровень приложений (Application)	HTTP, FTP, SSH, Telnet
		Уровень представления (Presentation)	ASCII, GZIP, binary
		Уровень сеанса (Session)	NetBIOS, SSL
Транспортный уровень		Уровень транспорта (Transport)	TCP, UDP
Уровень Internet		Уровень сети (Network)	IPv4, IPv6, IPX, AppleTalk
Уровень сетевого интерфейса		Уровень канала (Data Link)	PPP, IEEE 802.2 (Ethernet)
		Физический уровень (Physical)	USB, IEEE 802.11 IEEE 802.3 (Ethernet)

Заголовок TCP

Байты	0	1	2	3
0...3	Порт отправителя		Порт получателя	
4...7	Порядковый номер пакета			
8...11	Порядковый номер подтверждаемого пакета (с флагом ACK)			
12...15	Размер заголовка в 32-битных словах	000 NS CWR ECE URG **ACK** PSH RST **SYN** **FIN**	Размер окна (буфера для приема данных, ожидаемых при ответе)	
16...19	Контрольная сумма заголовка и данных		Указатель на порядковый номер пакета, в котором заканчивается приоритетных блок данных	
20.....	Дополнительные опции			

Взаимодействие по TCP

1. Установка соединения (флаг SYM)
2. Подтверждение установки соединения (SYN+ACK)
3. Двусторонний обмен данными - с флагом ACK кроме полезной нагрузки указывается номер подтверждаемого пакета
4. Закрытие соединения - пакет с флагом FIN



Взаимодействие по TCP

- Все пакеты имеют порядковый номер, который присваивает ядро ОС
- Задача ядра - сделать видимость непрерывного потока данных в обе стороны
- Ввод-вывод - как с обычными каналами read/write
- `socket( AF_... , SOCK_STREAM, 0 )`

Операции ввода-вывода

- **read**(fd,buf,count) → **recv**(fd,buf,count,flags=0)
- **write**(fd,buf,count) → **send**(fd,buf,count,flags=0)

- **recv**(fd,buf,count,flags) →
  recvfrom(fd,buf,count,flags,
  struct sockaddr, *src_addr = NULL,
  socklen_t *addrlen = NULL)

- **send**(fd,buf,count,flags) →
  sendto(fd,buf,count,flags,
  struct sockaddr, *src_addr = NULL,
  socklen_t *addrlen = NULL)

SOCK_STREAM: кроме TCP

- Протокол Novell SPX (historic)
- Локальное взаимодействие AF_UNIX

```
socket( AF_... , SOCK_STREAM, 0 )
```

последний параметр – номер протокола; 0 – автоматический выбор для пары AF_..., SOCK_...

Инструменты для взаимодействия

- **telnet** - терминал для текстового ввода-вывода через сокеты
- **netcat** или **nc** - аналог **cat**, но работает с сетевыми соединениями

Демонстрация: telnet/nc и lorem-ipsum-server

Поточное взаимодействие: HTTP

Запрос

```
GET /index.html HTTP/1.1
Host: www.example.com
Connection: keep-alive
DNT: 1
User-Agent: Mozilla/5.0 . . .
Accept-Encoding: gzip, deflate
Accept-Language: ru, en
```

Ответ

```
HTTP/1.1 200 OK
Server: Apache
Content-Type: text/html
Transfer-Encoding: chunked
Date: Mon, 27 Apr 2015 13:40:00

<html>
  <head></head>
  <body><p>It works!</p></body>
</html>
```

Шифрование

- Уровень сеанса в модели OSI
- В модели TCP - это уровень процессов

- Поток данных передается по TCP, его интерпретация - задача процесса в пространстве пользователя

```
> openssl s_client -connect www.yandex.ru:443
```

Типы сокетов

- SOCK_STREAM - двунаправленных поток данных
- **SOCK_DGRAM** - односторонние сообщения (UDP)
- SOCK_RAW - пакеты IP (уровень сети)
- SOCK_PACKET (Linux) - фреймы Ethernet (уровень канала)

Пакеты UDP

Байты	0	1	2	3
0..4	Порт отправителя		Порт назначения	
5..8	Длина пакета		Контрольная сумма	

- Односторонние сообщения
- Максимальная длина:
  MTU - sizeof(ip_header) - sizeof(udp_header)
- Процесс - получатель сам обязан сформировать ответ

UDP v.s TCP

- Нет требуется установление соединения (SYN/SYN+ACK) и его завершение (FIN/FIN+ACK)
- Не контролируется порядок пакетов
- Не требуется отправка пакетов в подтверждение получения

Сценарии использования:

1. Торренты
2. Эмуляция *уровня канала* (VPN)

Канальный уровень сети

- Адресация - по MAC-адресам
- На физическом уровне - пассивные (хабы) или активные (свитчи) коммутаторы
- MAC-адрес связан с конкретным устройством и не меняется (в теории, но не на практике) при конфигурации

Канальный уровень сети

- Привязка к MAC-адресу устройства
- Достаточно знать только своих соседей
- Выделяется отдельное устройство - маршрутизатор для выхода во внешнюю сеть

- Соотвествие между IP-адресами и MAC-адресами через протокол ARP

Как узнать MAC-адрес

MAC-адрес получателя	MAC-адрес отправителя	Дополнительные опции	Длина	Данные	Контрольная сумма
6 байт	6 байт	4 байта	2 байта	от 46 до 1500 байт (параметр MTU)	4 байта

- Формируется Ethernet-кадр, содержимое которого не IP-пакет, а ARP-запрос с требуемым адресом
- MAC-отправителя - того, кто отправляет
- MAC-получателя - широковещательный FF:FF:FF:FF:FF:FF
- Хост с подходящим IP отправляет ARP-ответ
- Инициирующая сторона кэширует ответ (время жизни - 30 секунд для Linux)

```
/usr/sbin/arp # отобразить таблицу arp
```

Сетевой уровень IPv4

- У каждого хоста есть IP-адрес (для IPv4 не гарантируется уникальность)
- "Серые" адреса не доступны из сети Интернет (например 192.168.x.y)
- Для доставки IP-пакета может потребоваться цепочка маршрутизаторов

```
/bin/route # (в POSIX-системах)
  /usr/sbin/ip r # (в Linux)
```

Назначение IP-адресов

- Статическое - явное указание параметров сетевого интерфейса
- Динамическое (протокол DHCP)
  - **DHCPDISCOVER** от 0.0.0.0:68 к 255.255.255.255:67
    - запросить предложения от DHCP-серверов в локальной сети
  - **DHCPOFFER** от одного из серверов конкретному MAC
    - предложение свободного адреса
  - **DHCPREQUEST** к конкретному серверу
    - запрос получения ранее предложенного адреса
  - **DHCPACK** от сервера
    - подтверждение выдачи адреса

    .....
  - **DCHCPRELEASE** серверу
    - освобождение (перед выключением)

Реализация DHCP

- UDP пакеты поверх IP
- Сервер - на порту 67
- В сообщении OFFER кроме IP-адреса содержится дополнительная информация:
  - адрес маршрутизатора
  - адреса DNS-серверов

`/sbin/dhclient`

DNS

- IP адрес определяет маршрут до хоста
- Доменное имя - должно быть запоминающимся, например:
  `скатать-акос-без-регистрации-и-смс.рф`
- Служба DNS - иерархическая система каталогов соответствия имен и IP-адресов
- Сервер DNS - работает поверх UDP и TCP на 53 порту; на маршрутизаторах - кеширование
- Можно отправлять запросы любым серверам (Google: 8.8.8.8)

```
/usr/bin/nslookup # (простая команда)
/usr/bin/dig # (более продвинутая)
```

Доменные имена

- Полная форма заканчивается символом точки (но для упрощения это часто не требуется от пользователя)
- Несколько типов:
  - **A** - IPv4-адрес хоста для соединения
  - **AAAA** - IPv6-адрес хоста для соединения
  - **MX** - имя хоста для электронной почты
  - **CNAME** - имя хоста для соединения

Как посмотреть на разные пакеты

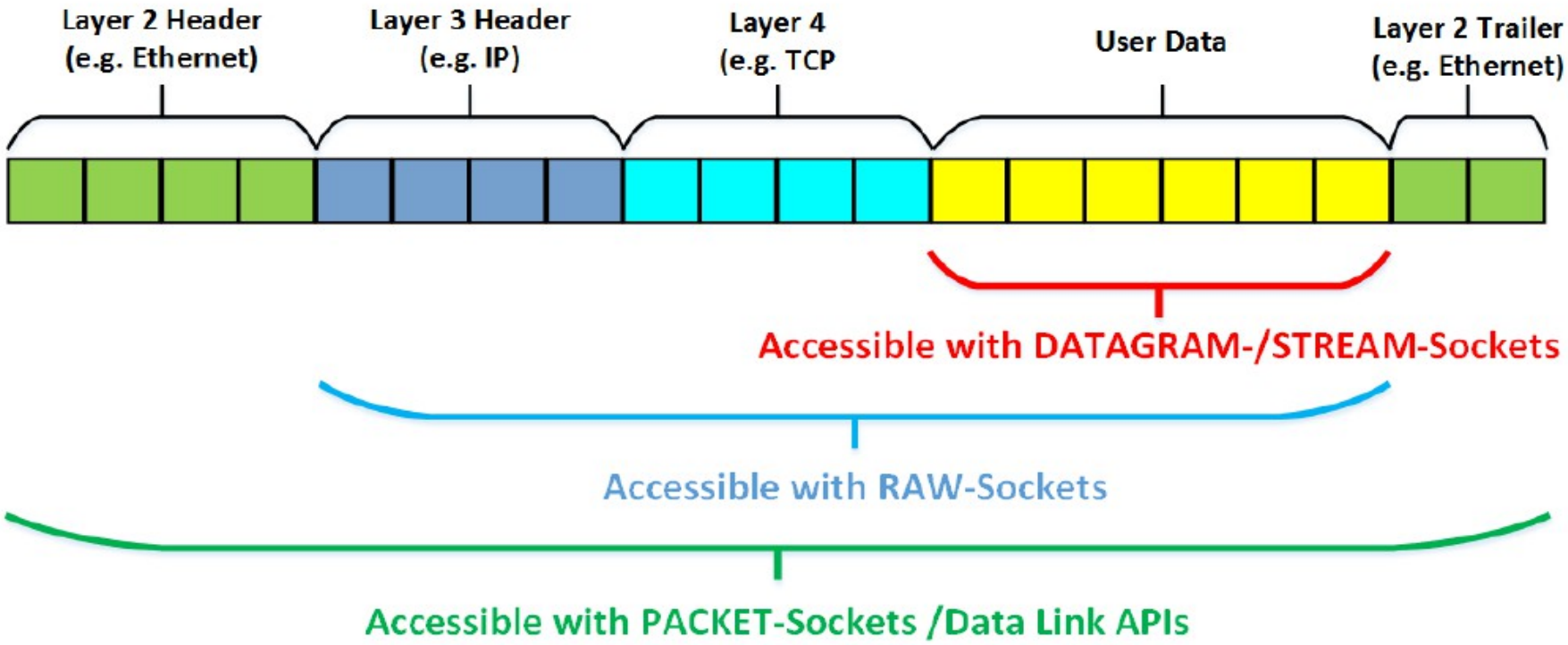
- Консольная утилита **`tcpdump`**
- Программа **`wireshark`**
- **Требуются права root**

Демонстрация wireshark

Типы сокетов

- SOCK_STREAM - двунаправленных поток данных
- SOCK_DGRAM - односторонние сообщения (UDP)
- **SOCK_RAW** - пакеты IP (уровень сети)
- **SOCK_PACKET** (Linux) - фреймы Ethernet (уровень канала)

Типы сокетов



Картинка из статьи ***Introduction to RAW-sockets.***
Jens Heuschkel, Tobias Hoffmar et al.Technische Universität Darmstadt. 2017

Работа с сокетами

- int fd = socket(...) - создать сокет
- connect(fd, ...) - подключиться
- send/recv - обмен данными

Стадия **`connect`** настраивает заголовки по умолчанию, связанные с *конкретным типом* сокета.

RAW-сокеты

```
sockfd = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)
```

- Работают только из-под рута или настроенным CAP_NET_RAW
- Можно настроить, чтобы добавлялись IP-заголовки (setsockopt IP_HDRINCL)

Сокеты канального уровня:

```
sockfd = socket(AF_PACKET, SOCK_RAW,
                htons(ETH_P_ALL) // фильтр
                )
```

Низкоуровневые операции

- Протоколирование действий
- Фильтрация траффика
- Реализация нестандартного протокола передачи данных