

# Сеть и сетевые протоколы: L4-сеть



Андрей  
Вахутинский



# Андрей Вахутинский

Заместитель начальника IT-отдела  
АО “ИНТЕКО”

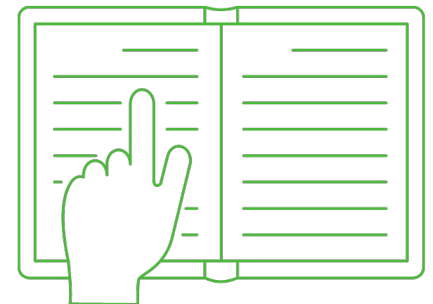


[Андрей Вахутинский](#)

# Предисловие

Эта лекция содержит **основные понятия**, связанные с 4-м уровнем модели OSI (Transport / Транспортный уровень).

Также вы познакомитесь с основными командами, которые позволяют получать информацию / вносить изменения в настройки ОС на 4-м уровне модели OSI.



---

# План занятия

1. [Предисловие](#)
2. [Транспортный уровень](#)
3. [Протокол TCP](#)
  - a. [Заголовок TCP](#)
  - b. [Сокеты](#)
  - c. [Установление соединения](#)
  - d. [Завершение соединения](#)
4. [Протокол UDP](#)
5. [Сравнение протоколов TCP и UDP](#)
6. [Популярные сетевые утилиты](#)
7. [Итоги](#)
8. [Домашнее задание](#)



# Транспортный уровень

---

## Вспомним основные понятия

**Транспортный уровень** (Transport layer) — определяет способы доставки данных (т.е. определяет сам механизм передачи данных).

**Тип взаимодействия:** точка – точка.

**Основные решаемые проблемы:**

- мультиплексирование (может работать с несколькими потоками данных между двумя устройствами);
- надежная передача данных;
- регулирование количества передаваемых данных;
- контроль доставки данных.

---

# Вспомним основные понятия

## Единица данных:

- сегмент (segment);
- дейтаграмма (datagram).

## Протоколы:

- TCP;
- UDP.



# Протокол ТСП



---

# Спецификация

Спецификация протокола TCP приведена в [RFC 675](#).

→ Протокол TCP является «рабочей лошадкой» Интернета.

Именно он передает данные между приложениям (например, интернет-браузером) и web-службами.



---

# Назначение протокола TCP

- надежная доставка данных;
- сборка сегментов на стороне получателя;
- контроль сессии;
- контроль скорости передачи данных.



---

# Надежность сетей передачи данных

Когда приложение посылает данные в сеть, никто не знает, какой путь будет выбран для каждого отдельного пакета. Также ничто не гарантирует, что до получателя дойдут все посланные данные.

Маршрут пакетов одной сессии может измениться по множеству причин:

- авария на одном из маршрутизаторов;
- работа балансировщика нагрузки;
- ошибки маршрутизации и т.п.

Таким образом, получатель, собрав сессию, может получить сегменты не в том порядке, в котором они отправлялись или не получить часть пакетов вообще.

---

# Надежная доставка данных протокола TCP

Надежная доставка осуществляется **автоматической повторной пересылкой пропавших сегментов**.

Каждый сегмент TCP содержит в заголовке специальное поле — «порядковый номер» → **Sequence number**.

После того, как отправитель выслал какое-то количества сегментов, он будет ждать подтверждения от получателя, с указанием порядкового номера следующего сегмента, который адресат желает получить → **Acknowledgment number**.

Если такое подтверждение не получено, отправка повторится.

---

# Сборка сегментов

Сборка сегментов пришедших в неправильном порядке также будет осуществлена автоматически, используя те же поля:

- Sequence number;
- Acknowledgment number.

Протокол TCP соберет сегменты сессии в нужном порядке и передаст правильные данные приложению получателя.

---

# Контроль сессии

Перед началом передачи, ТСП всегда проверяет, что **получатель существует и готов принимать данные**.

➔ Этот механизм называется «трехстороннее рукопожатие» (three-way handshake).

Во время сессии данные контролируются при помощи **Sequence number** и **Acknowledgment number**.

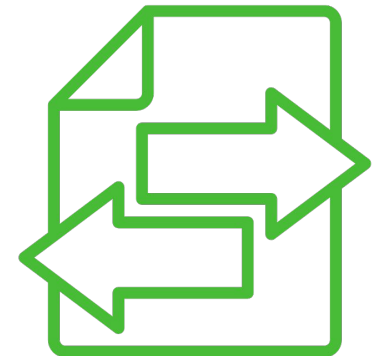
При закрытии сессии получатель и отправитель также извещают друг друга о том, что сеанс связи завершен.

# Контроль скорости передачи

В TCP существует механизм скользящего окна ➡ sliding window.

Он позволяет менять количество пересылаемых данных до следующего подтверждения.

Благодаря этому, отправитель может динамически менять размер пересылаемых данных, анализируя подтверждения от получателя.

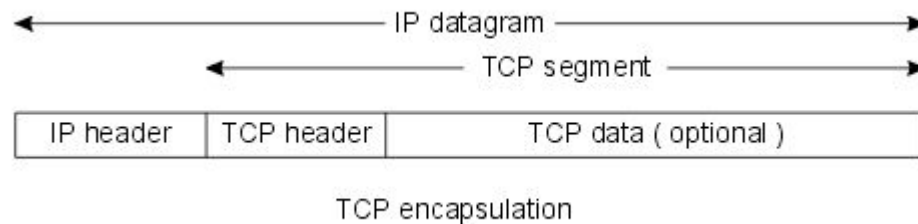




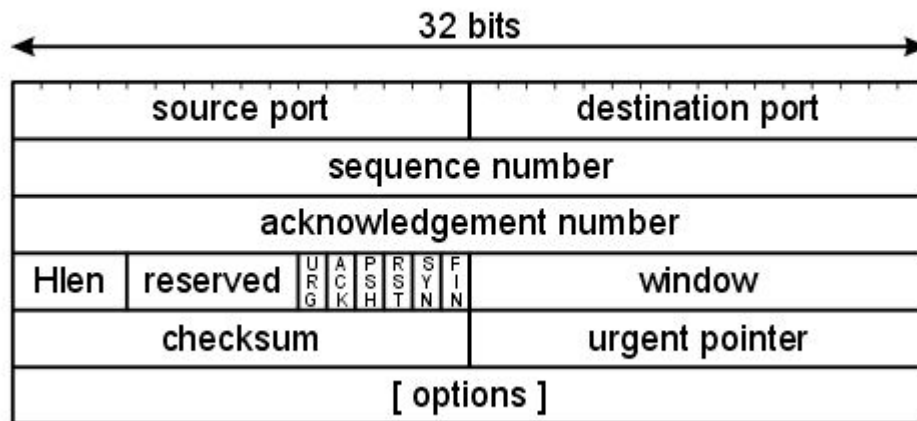
# Заголовок ТСР



# Заголовок TCP



## TCP header format



---

# Основные поля заголовка TCP

- **Source port / Destination port** — номера портов получателя и отправителя;
- **Sequence number** и **Acknowledgment number** — порядковый номер и номер подтверждения;
- **Hlen** (Header length) — длина самого заголовка TCP;
- **Window** — содержит размер окна, о чём было сказано выше;
- **Urgent** — признак важности (срочности) данного сегмента;
- **Options** — необязательное поле, которое может использоваться разработчиками

---

# Основные флаги заголовка TCP

- **ACK** – пакет содержит значение номера подтверждения в поле подтверждения (Acknowledgment number);
- **PSH** – получатель должен передать все данные из буфера в приложение и сразу же отправить сообщение с подтверждением;
- **RST** – сброс (reset) соединения;
- **SYN** – создает соединение;
- **FIN** – завершает соединение.

---

# Порты заголовка ТСР

**Номер порта** – это идентификатор приложения на хосте получателя и/или отправителя.

Это всегда целое положительное число (0 – 65535).

**Номера портов:**

- общеизвестные (well-known) – с 0 по 1023;
- зарегистрированные / пользовательские – с 1024 по 49151;
- динамические / частные – с 49152 по 65535.

---

# Well Known Ports

- 20 и 21 – FTP;
- 22 – SSH;
- 23 – TELNET;
- 25 – SMTP;
- 53 – DNS;
- 80 – HTTP;
- 123 – NTP;
- 443 – HTTPS.

# Wireshark

```
▶ Internet Protocol Version 4, Src: 192.168.88.254, Dst: 8.238.89.254
▲ Transmission Control Protocol, Src Port: 51354, Dst Port: 80, Seq: 1, Ack: 1, Len: 280
    Source Port: 51354
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 280]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 281    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
    ▶ Flags: 0x018 (PSH, ACK)
    Window size value: 256
    [Calculated window size: 65536]
    [Window size scaling factor: 256]
    Checksum: 0xb7a3 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
    ▲ [SEQ/ACK analysis]
        [iRTT: 0.033952000 seconds]
        [Bytes in flight: 280]
        [Bytes sent since last PSH flag: 280]
    ▲ [Timestamps]
        [Time since first frame in this TCP stream: 0.034211000 seconds]
        [Time since previous frame in this TCP stream: 0.000259000 seconds]
    TCP payload (280 bytes)
▶ Hypertext Transfer Protocol
```



# Сокеты

---

# Сокеты

**Сетевой сокет** – структура которая определяет конечную точку во время сетевого обмена данными.

Для TCP/IP сокетом является сочетание трех параметров сессии:

- транспортного протокола (TCP, UDP);
- номера порта (0 – 65535);
- IP-адреса.

Сетевой сокет TCP/IP также называют **интернет-сокетом**.

**Парные сокеты** – локальный и соответствующий удаленный сокеты.



---

## Аналогия

Аналогия сокетов – динамик и микрофон при разговоре по мобильному телефону: вы говорите в микрофон, который на вашем устройстве, а слышит вас собеседник в своём динамике.

Т.е. это такая «виртуальная» труба, соединяющая две точки в сети, где каждый пишет и читает в свой/из своего сокета, что приводит к автоматическому взаимодействию с удалённым сокетом.



---

# Сокеты

## Типы сокетов по подключению:

- потоковый (TCP);
- сокет дейтаграмм (UDP);
- сырой (raw) сокет (IP-пакет с данными).

## Типы сокетов по функциям:

- клиентский (всегда один);
- серверный (может быть несколько одинаковых на сервере).



# Установление соединения

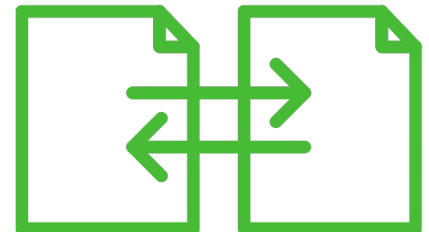
---

# Установка соединения

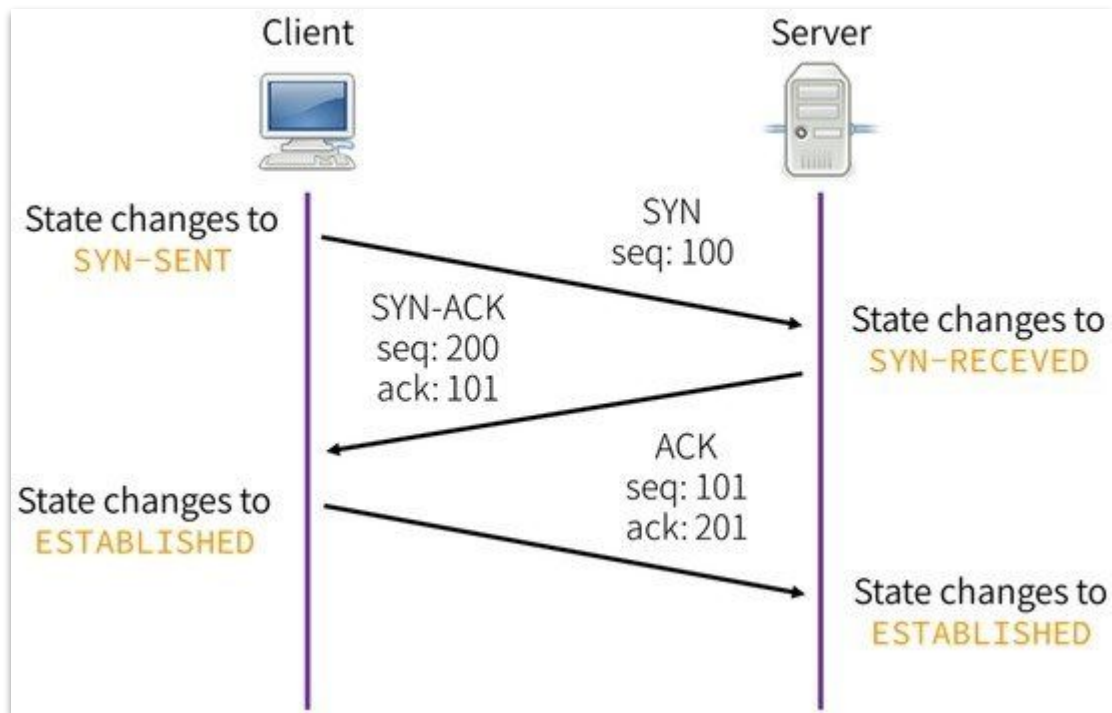
**Установка соединения** – важная стадия сетевого обмена.

Хосты могут убедиться, что они оба существуют и работают, что на обоих запущен один и тот же стек протоколов.

Протокол ТСР устанавливает соединение с помощью трехстороннего рукопожатия (в три этапа).



# Установление соединения



128	66	192.168.88.254	95.173.136.70	TCP	60524 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
131	66	95.173.136.70	192.168.88.254	TCP	80 → 60524 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1380 SACK_PERM=1 WS=128
132	54	192.168.88.254	95.173.136.70	TCP	60524 → 80 [ACK] Seq=1 Ack=1 Win=66048 Len=0
135	378	192.168.88.254	95.173.136.70	HTTP	GET /static/css/print.css HTTP/1.1

# Первый шаг установки соединения

Transmission Control Protocol, Src Port: 60524, Dst Port: 80, Seq: 2387450305, Len: 0

Source Port: 60524

Destination Port: 80

[Stream index: 8]

[TCP Segment Len: 0]

Sequence number: 2387450305

[Next sequence number: 2387450305]

Acknowledgment number: 0

1000 .... = Header Length: 32 bytes (8)

Flags: 0x002 (SYN)

Window size value: 8192

[Calculated window size: 8192]

Checksum: 0x38aa [unverified]

[Checksum Status: Unverified]

Urgent pointer: 0

Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted

[Timestamps]

1. Отправляем SYN-пакет.
2. SeqNum = 2387450305 (1-е случайное число).

## Второй шаг установки соединения

```
Transmission Control Protocol, Src Port: 80, Dst Port: 60524, Seq: 883876465, Ack: 2387450306, Len: 0
```

```
Source Port: 80
```

```
Destination Port: 60524
```

```
[Stream index: 8]
```

```
[TCP Segment Len: 0]
```

```
Sequence number: 883876465
```

```
[Next sequence number: 883876465]
```

```
Acknowledgment number: 2387450306
```

```
1000 .... = Header Length: 32 bytes (8)
```

```
► Flags: 0x012 (SYN, ACK)
```

```
Window size value: 29200
```

```
[Calculated window size: 29200]
```

```
Checksum: 0xcfb9 [unverified]
```

```
[Checksum Status: Unverified]
```

```
Urgent pointer: 0
```

```
► Options: (12 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK permitted, No-Operation (NOP), Window scale
```

```
► [Timestamps]
```

3. Приходит ответ SYN-ACK.
4. SeqNum = 883876465 (2-е случайное число).
5. AckNum = 2387450306 (1-е случайное число +1).

## Третий шаг установки соединения

```
Transmission Control Protocol, Src Port: 60524, Dst Port: 80, Seq: 2387450306, Ack: 883876466, Len: 0
  Source Port: 60524
  Destination Port: 80
  [Stream index: 8]
  [TCP Segment Len: 0]
  Sequence number: 2387450306
  [Next sequence number: 2387450306]
  Acknowledgment number: 883876466
  0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x010 (ACK)
  Window size value: 258
  [Calculated window size: 66048]
  [Window size scaling factor: 256]
  Checksum: 0x814a [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  ▸ [Timestamps]
```

6. Отправляем подтверждение **ACK**.
7. **SeqNum** = 2387450306 (1-е случайное число **+1**).
8. **AckNum** = 883876466 (2-е случайное число **+1**).





# Завершение соединения

---

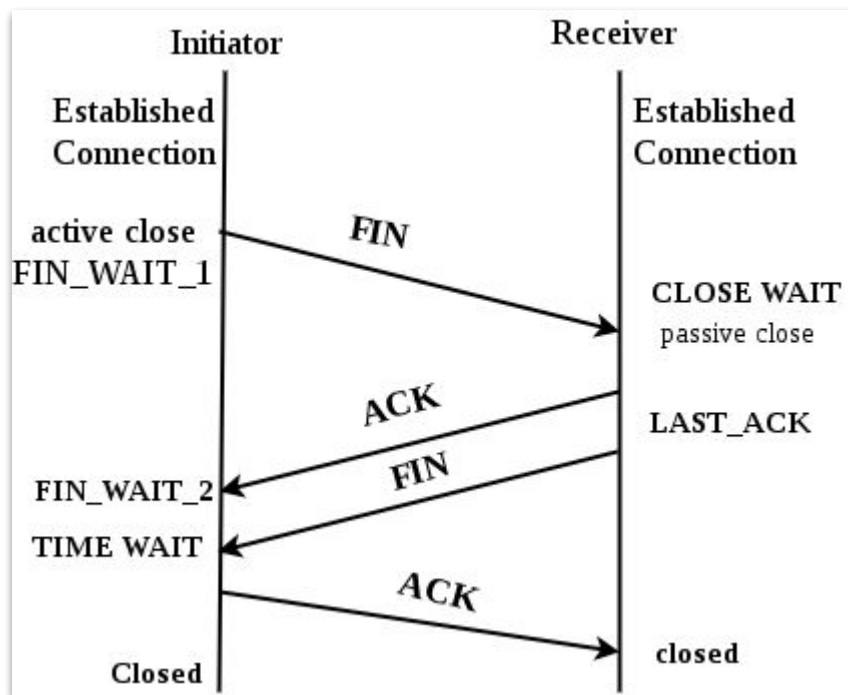
## Завершение соединения

Нормальное завершение TCP-соединения происходит **в виде двухстороннего рукопожатия.**

Во время этой процедуры оба хоста закрывают свои сокеты и освобождают занятые во время сеанса ресурсы.



# Завершение соединения



60	95.173.136.70	192.168.88.254	TCP	80 → 61114	[FIN, ACK] Seq=3751094172 Ack=1145032593 Win=30336 Len=0
54	192.168.88.254	95.173.136.70	TCP	61114 → 80	[ACK] Seq=1145032593 Ack=3751094173 Win=66048 Len=0
54	192.168.88.254	95.173.136.70	TCP	61114 → 80	[FIN, ACK] Seq=1145032593 Ack=3751094173 Win=66048 Len=0
60	95.173.136.70	192.168.88.254	TCP	80 → 61114	[ACK] Seq=3751094173 Ack=1145032594 Win=30336 Len=0

# Завершение соединения

```
Transmission Control Protocol, Src Port: 80, Dst Port: 61114, Seq: 3751094172, Ack: 1145032593, Len: 0
  Source Port: 80
  Destination Port: 61114
  [Stream index: 7]
  [TCP Segment Len: 0]
  Sequence number: 3751094172
  [Next sequence number: 3751094172]
  Acknowledgment number: 1145032593
  0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x011 (FIN, ACK)
  Window size value: 237
  [Calculated window size: 30336]
  [Window size scaling factor: 128]
  Checksum: 0xa33e [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
```

Посылаем пакет с **флагом FIN**.

В данном случае, **флаг ACK** – подтверждение передачи и не имеет отношение к закрытию соединения.

# Завершение соединения

```
Transmission Control Protocol, Src Port: 61114, Dst Port: 80, Seq: 1145032593, Ack: 3751094173, Len: 0
  Source Port: 61114
  Destination Port: 80
  [Stream index: 7]
  [TCP Segment Len: 0]
  Sequence number: 1145032593
  [Next sequence number: 1145032593]
  Acknowledgment number: 3751094173
  0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x010 (ACK)
    Window size value: 258
    [Calculated window size: 66048]
    [Window size scaling factor: 256]
    Checksum: 0xa329 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ [Timestamps]
```

Получаем в ответ подтверждение (ACK), что соединение закрыто.

# Завершение соединения

```
Transmission Control Protocol, Src Port: 61114, Dst Port: 80, Seq: 1145032593, Ack: 3751094173, Len: 0
  Source Port: 61114
  Destination Port: 80
  [Stream index: 7]
  [TCP Segment Len: 0]
  Sequence number: 1145032593
  [Next sequence number: 1145032593]
  Acknowledgment number: 3751094173
  0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x011 (FIN, ACK)
    Window size value: 258
    [Calculated window size: 66048]
    [Window size scaling factor: 256]
    Checksum: 0xa328 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ [Timestamps]
```

Получаем уведомление **ACK** + **FIN** от удаленного хоста, что соединение закрыто.

# Завершение соединения

```
Transmission Control Protocol, Src Port: 80, Dst Port: 61114, Seq: 3751094173, Ack: 1145032594, Len: 0
  Source Port: 80
  Destination Port: 61114
  [Stream index: 7]
  [TCP Segment Len: 0]
  Sequence number: 3751094173
  [Next sequence number: 3751094173]
  Acknowledgment number: 1145032594
  0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x010 (ACK)
    Window size value: 237
    [Calculated window size: 30336]
    [Window size scaling factor: 128]
    Checksum: 0xa33d [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ [Timestamps]
```

Подтверждаем закрытие **флагом ACK**.

# Завершение соединения

В заголовке сегмента TCP есть специальный флаг RST – сброс.

Если этот флаг установлен в 1, это означает, что хост, получивший такой пакет, должен немедленно прекратить обмен и закрыть со своей стороны сокет.

Данный механизм нужен для уведомления противоположной стороны о произошедшем сбое.

484	9.660973	0.000211	10.0.10.140	plus.google.com	255	TCP	54	55117→https(443)	[FIN, ACK]	Seq=455	Ack
485	9.661032	0.000059	10.0.10.140	plus.google.com	0	TCP	54	55117→https(443)	[RST, ACK]	Seq=456	Ack
490	9.712200	0.009154	plus.google.com	10.0.10.140	245	TCP	60	https(443)→55117	[FIN, ACK]	Seq=4598	Ac





# Протокол UDP

# Спецификация протокола UDP

Спецификация протокола UDP приведена в [RFC 768](#).

➔ Протокол UDP является «рабочей лошадкой» всех потоковых сервисов (видео / аудио стриминг) и геймдева.

Именно этот протокол обеспечивает работу современных цифровых сервисов.

Кроме этого, он является основой DNS.

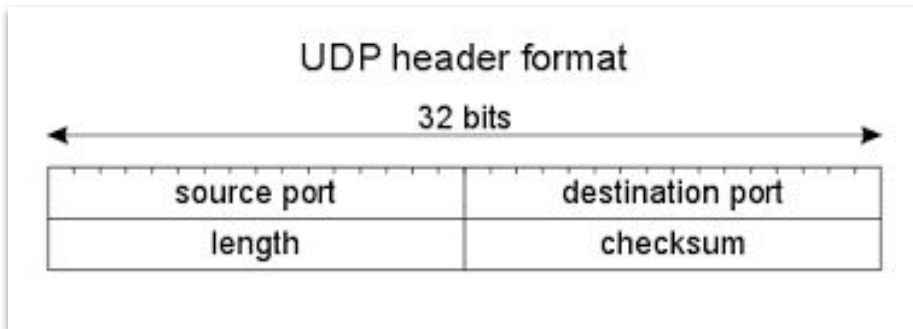


---

# Назначение протокола UDP

- ориентирован на транзакции (запрос – ответ), например. DNS или NTP;
- простой протокол, для которого не требуется сложная модель обмена данными;
- не сохраняет состояния соединения – подходит для рассылки большому количеству хостов (IPTV);
- отсутствуют повторные передачи, что позволяет работать в режиме реального времени (игры);
- поддерживает многоадресную рассылку (Precision Time Protocol and Routing Information Protocol).

# Заголовок протокола UDP



Как видно из рисунка, протокол крайне прост.

В протоколе нет:

- флагов;
- средств синхронизации;
- средств контроля сессии.

# Multicast

Вспомним, **multicast** (многоадресный) – режим передачи данных, в котором данные передаются группе хостов.

В сетях IPv4 для многоадресного вещания зарезервирована подсеть [224.0.0.0/4](#).

Преимущество многоадресного вещания в том, что снижается нагрузка на пропускную способность сети, т.к. часть информации для разных клиентов передается один раз по «общему» маршруту.

## Примеры сервисов:


- потоковое аудио и видео;
- интернет-телевидение.

# DNS

**DNS** (Domain Name System, система доменных имен).

Пока не будем углубляться  
в функции DNS, отметим,  
что данный протокол работает  
в режиме запрос-ответ,  
обмениваясь короткими  
сообщениями.

```
User Datagram Protocol, Src Port: 64459, Dst Port: 53
  Source Port: 64459
  Destination Port: 53
  Length: 53
  Checksum: 0x9277 [unverified]
  [Checksum Status: Unverified]
  [Stream index: 0]
  ▸ [Timestamps]
  Domain Name System (query)
    Transaction ID: 0xaca5
    ▸ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
    ▾ Queries
      ▾ chat-pa.clients6.google.com: type A, class IN
        Name: chat-pa.clients6.google.com
        [Name Length: 27]
        [Label Count: 4]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
      [Response In: 6]
```



# Сравнение протоколов ТСП и UDP

---

# Сравнение протокола TCP с UDP

- **надежность** – TCP управляет потоком данных через подтверждения, повторные передачи, изменение скорости.  
➔ В UDP ничего этого нет.
- **упорядоченность** – если пакеты попали к получателю не в том порядке, TCP сможет пересобрать нужные сегменты и приложение этого не заметит.  
➔ UDP получает все данные «по факту».
- **ресурсоемкость** – TCP требует много накладных расходов (по времени и по нагрузке на сеть). Особенно, при неустойчивой связи.  
➔ В UDP все очень просто, поэтому он работает быстрее и меньше нагружает сеть.



---

# Сравнение протокола UDP с TCP

- **широковещательный режим** – т.к UDP не требует установки соединения, то данные, которые посылаются всем узлам сети, будут обработаны одинаково.
- **многоадресный режим** – каждая датаграмма может быть передана группе подписчиков без дополнительных накладных расходов.

→ Эти два пункта невозможны в протоколе TCP как из-за требования к установлению соединения, так и к высокому расходу ресурсов для контроля каждой сессии.



# Популярные утилиты

# telnet

**telnet** – одна из самых популярных утилит для проверки «открытости» TCP порта.

```
> telnet ya.ru 80
[чёрный экран говорит о том, что соединение установилось]

[Enter],[Enter],[Enter]

HTTP/1.1 400 Bad request
Connection: Close
Content-Length: 0

Подключение к узлу утеряно.

> telnet ya.ru 81
Подключение к ya.ru..
Не удалось открыть подключение к этому узлу, на порт 81: Сбой подключения
```

# nmap

**nmap** – одна из самых популярных утилит для сканирования хостов в сетях – как внутренних, так и внешних – на открытые порты.

```
nmap -p U:53,79,113,T:21-25,80,443,8080 192.168.1.1 # сканирует определённые порты
nmap -top-ports 10 172.16.1.1 # сканирует топ 10 популярных портов
nmap -sT 172.16.1.1 # сканировать все TCP порты
```

```
# nmap -A -T4 scanme.nmap.org

Starting Nmap ( https://nmap.org )
Interesting ports on scanme.nmap.org (64.13.134.52):
(The 1663 ports scanned but not shown below are in state: filtered)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 3.9p1 (protocol 1.99)
53/tcp    open  domain
70/tcp    closed gopher
80/tcp    open  http     Apache httpd 2.0.52 ((Fedora))
113/tcp   closed auth
Device type: general purpose
Running: Linux 2.4.X|2.5.X|2.6.X
OS details: Linux 2.4.7 - 2.6.11, Linux 2.6.0 - 2.6.11
```

## SS

**socket statistics** – наиболее актуальная утилита для сбора информации о сокетах, в частности сетевых сокетах. Аналог **netstat**.

```
osboxes@osboxes:~$ ss -4 state listening state unconnected -n | column -t
Netid  State   Recv-Q  Send-Q  Local        Address:Port  Peer  Address:Port  Process
udp    UNCONN  0        0       127.0.0.53%lo:53  0.0.0.0:*
udp    UNCONN  0        0       0.0.0.0:5353    0.0.0.0:*
tcp    LISTEN  0        4096    127.0.0.53%lo:53  0.0.0.0:*
tcp    LISTEN  0        5      127.0.0.1:631    0.0.0.0:*
```

В домашнем задании подумайте, почему UDP в состоянии **UNCONN** = unconnected?

А так посмотрим установленные TCP соединения не на порт SSH:

```
osboxes@osboxes:~$ ss state connected sport != :ssh -t | column -t
State  Recv-Q  Send-Q  Local        Address:Port  Peer  Address:Port
Process
ESTAB  0        0       10.0.2.15:48668  104.26.8.143:http
```

# lsof

**lsof** – мощная утилита, в том числе для получения информации по сети.

Среди прочего он поможет вам узнать, какому процессу принадлежит прослушиваемый порт:

```
osboxes@osboxes:~$ sudo lsof -ni :22
COMMAND  PID    USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sshd     722    root    3u    IPv4  21337      0t0  TCP *:ssh (LISTEN)
sshd     722    root    4u    IPv6  21348      0t0  TCP *:ssh (LISTEN)
sshd     778    root    4u    IPv4  22479      0t0  TCP 10.0.2.15:ssh->10.0.2.2:52115
(ESTABLISHED)
sshd     1179   osboxes 4u    IPv4  22479      0t0  TCP 10.0.2.15:ssh->10.0.2.2:52115
(ESTABLISHED)
sshd     1538   root    4u    IPv4  30466      0t0  TCP 10.0.2.15:ssh->10.0.2.2:52283
(ESTABLISHED)
sshd     1607   osboxes 4u    IPv4  30466      0t0  TCP 10.0.2.15:ssh->10.0.2.2:52283
(ESTABLISHED)
```



# Итоги

---

## Итоги

Сегодня мы познакомились с базовыми представлениями о протоколах транспортного уровня:

- протоколе TCP;
- протоколе UDP.

Познакомились с популярными инструментами для работы на 4-м уровне модели OSI.







# Домашнее задание

---

# Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и  
пишите отзыв о лекции!**

**Андрей Вахутинский**