

Сеть и сетевые протоколы: L4-сеть

Ильмир Сахипов
Руководитель центра управления сетью АО “Уфанет”



Ильмир Сахипов

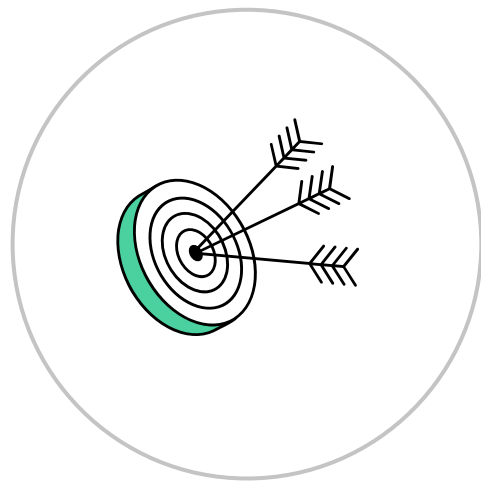
О спикере:

- Руководитель центра управления сетью АО “Уфанет”
- Более 10 лет опыта в области телекоммуникаций
- Эксперт в решении сложных клиентских и сетевых инцидентов на мультивендорной мультисервисной операторской сети



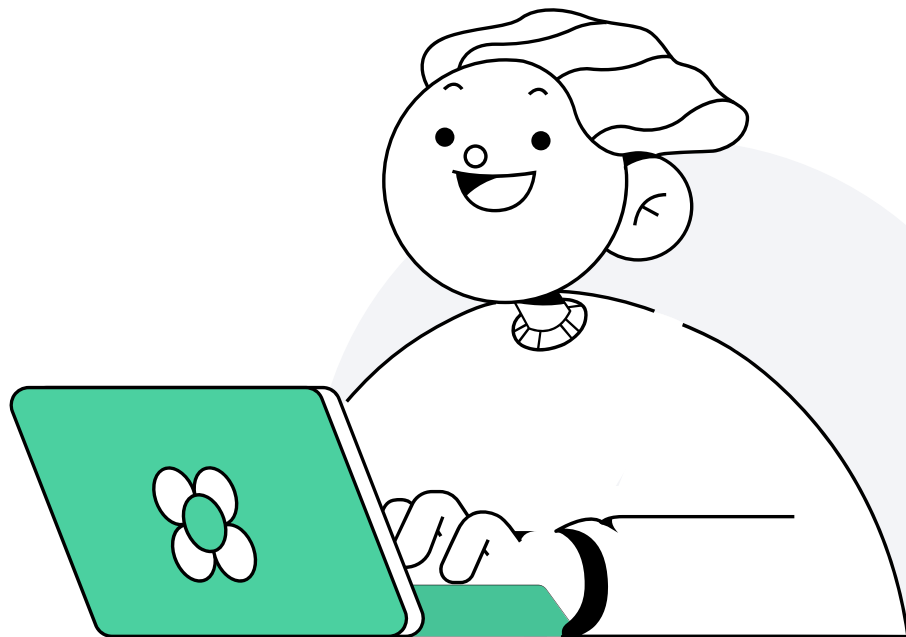
Цели занятия

- Изучить базовые протоколы транспортного уровня TCP и UDP
- Познакомиться с популярными утилитами для работы на транспортном уровне модели OSI
- На практике научиться низкоуровневому анализу TCP и UDP для поиска и устранения возникающих проблем соединения транспортного уровня

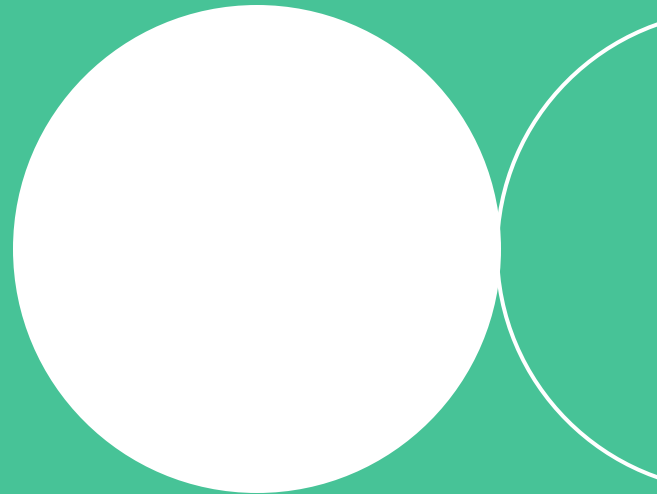


План занятия

- 1 [Транспортный уровень](#)
- 2 [Обзор протокола TCP](#)
- 3 [Заголовок TCP](#)
- 4 [Порты TCP](#)
- 5 [TCP: сокеты](#)
- 6 [TCP: установление соединения](#)
- 7 [TCP: завершение соединения](#)
- 8 [Обзор протокола UDP](#)
- 9 [Сравнение протоколов TCP и UDP](#)
- 10 [Популярные сетевые утилиты](#)
- 11 [Домашнее задание](#)

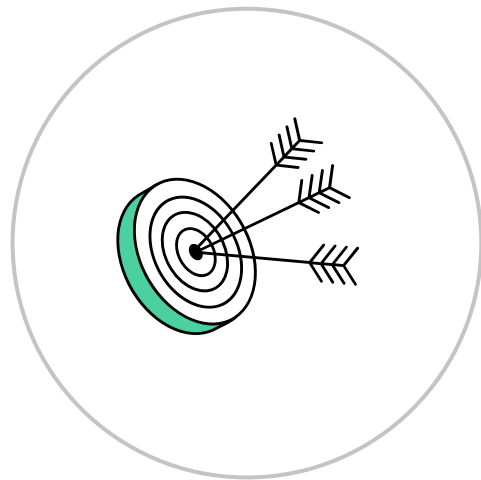


Транспортный уровень



Цели темы

- Вспомнить основные понятия, относящиеся к транспортному уровню
- Узнать об основных проблемах, которые решаются на транспортном уровне



Уровни модели OSI

Прикладной уровень

Application layer

Уровень представления

Presentation layer

Сеансовый уровень

Session layer

Транспортный уровень

Transport layer

Сетевой уровень

Network layer

Канальный уровень

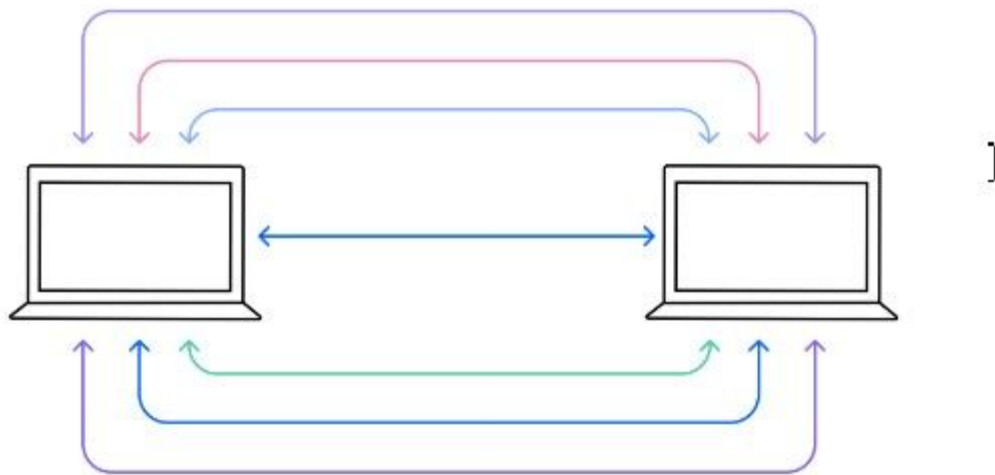
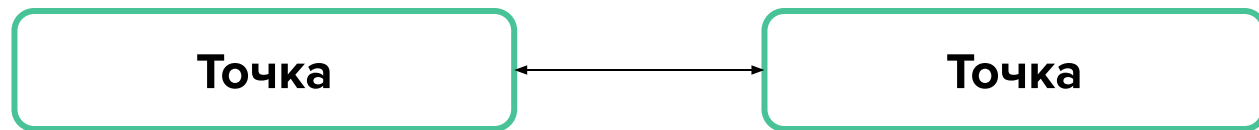
Data link layer

Физический уровень

Physical layer


Определяет способы доставки данных, то есть сам механизм передачи данных

Транспортный уровень: тип взаимодействия



Транспортный уровень: решаемые проблемы

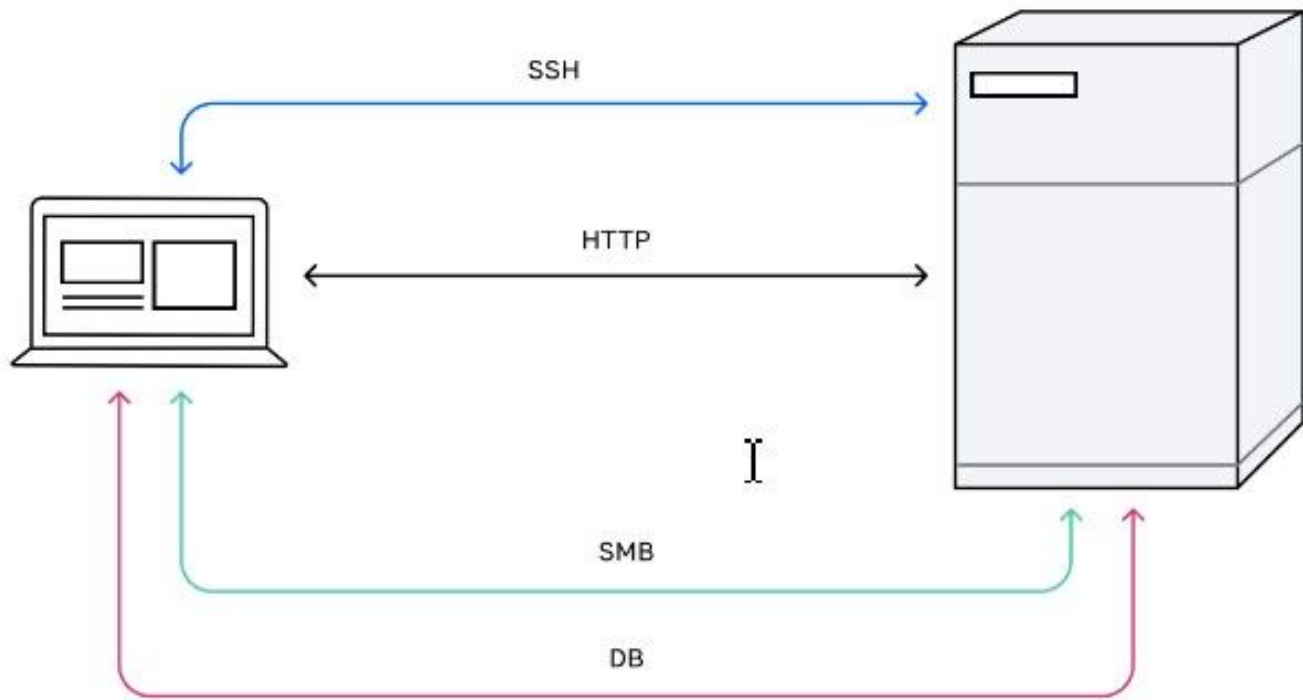
Любой протокол
уровня



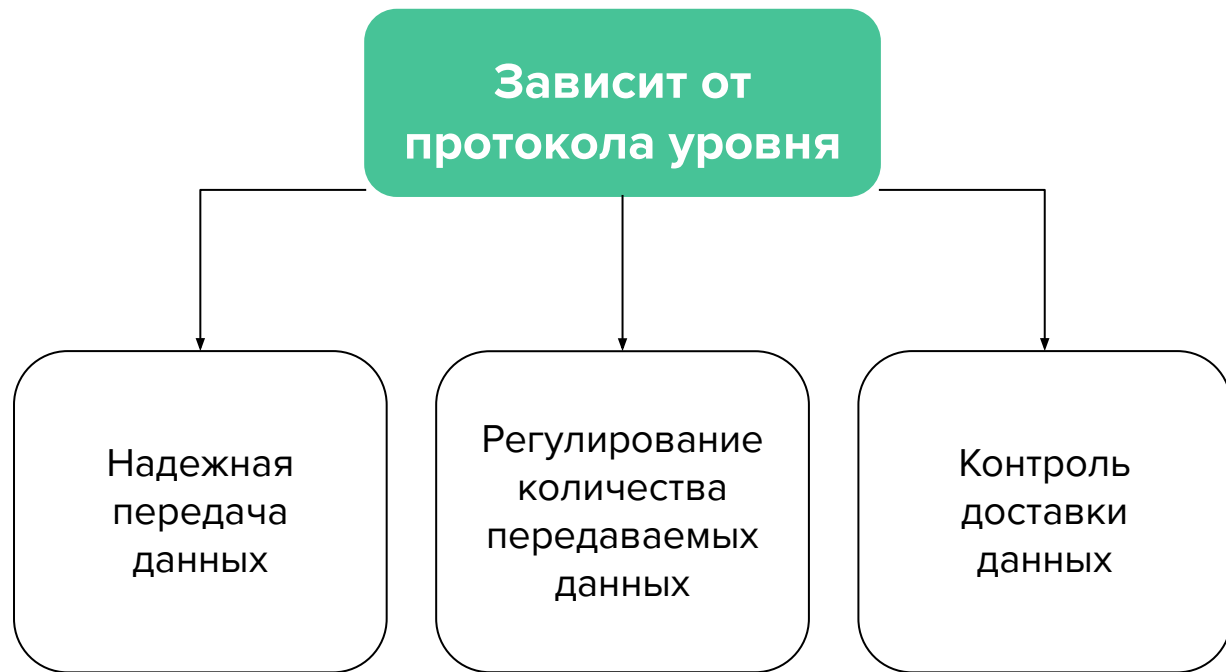
```
graph TD; A[Любой протокол уровня] --> B[Мультиплексирование может работать с несколькими потоками данных между двумя устройствами];
```

Мультиплексирование
может работать с
несколькими потоками
данных между двумя
устройствами

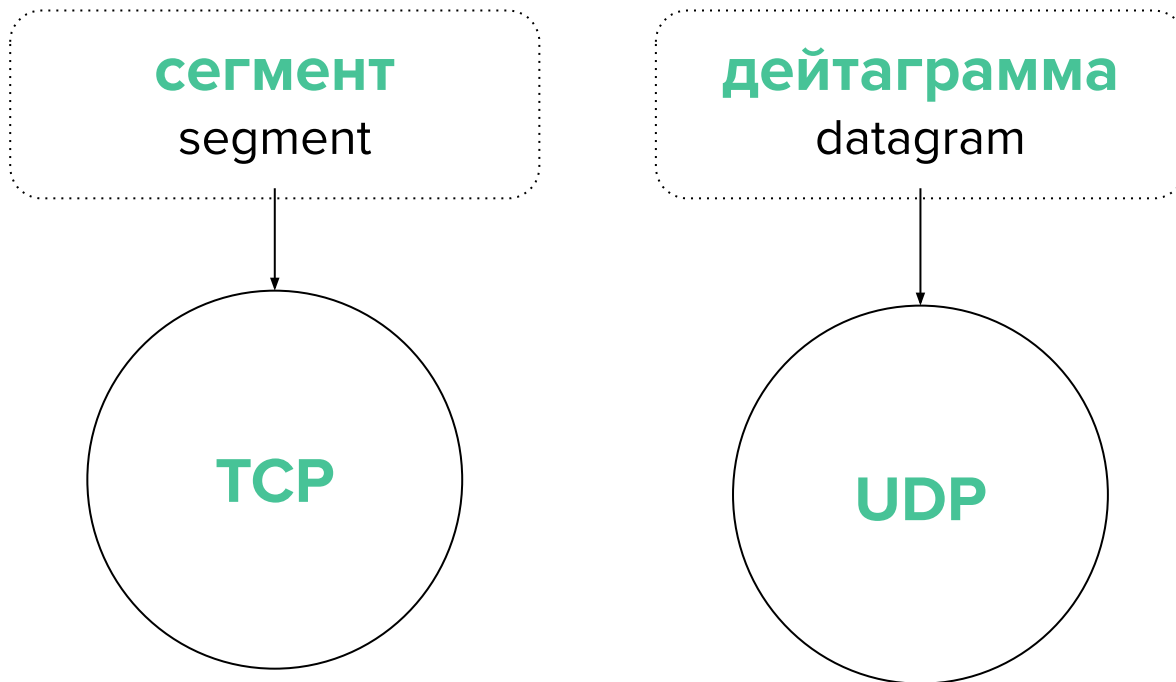
Транспортный уровень: решаемые проблемы



Транспортный уровень: решаемые проблемы



Транспортный уровень: единицы и протоколы

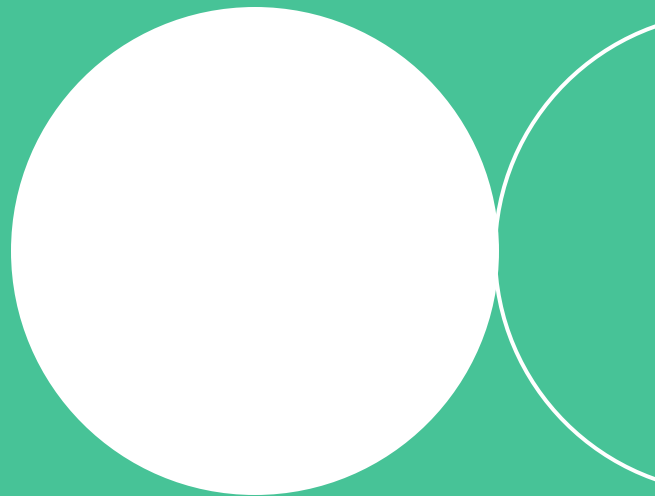


Итоги

- 1 Транспортный уровень устанавливает взаимодействие “точка-точка”. Его задача - собрать данные, полученные от сетевого уровня, и передать их на уровень выше
- 2 Мультиплексирование характерно для всех транспортных протоколов, остальные проблемы решаются только определенными протоколами

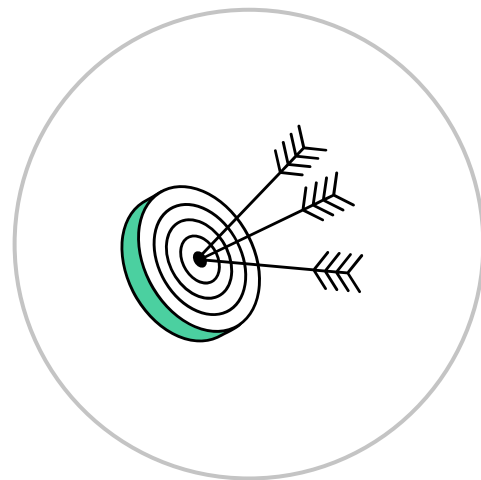


Обзор протокола TSP



Цели темы

- Детально познакомиться с протоколом TCP
- Рассмотреть проблемы, возникающие при передаче данных и понять, какие механизмы TCP гарантируют надежность доставки и упорядочивания данных
- Разобрать механизм контроля сессии и скорости передачи данных



Спецификация протокола TCP



[RFC 675](#)

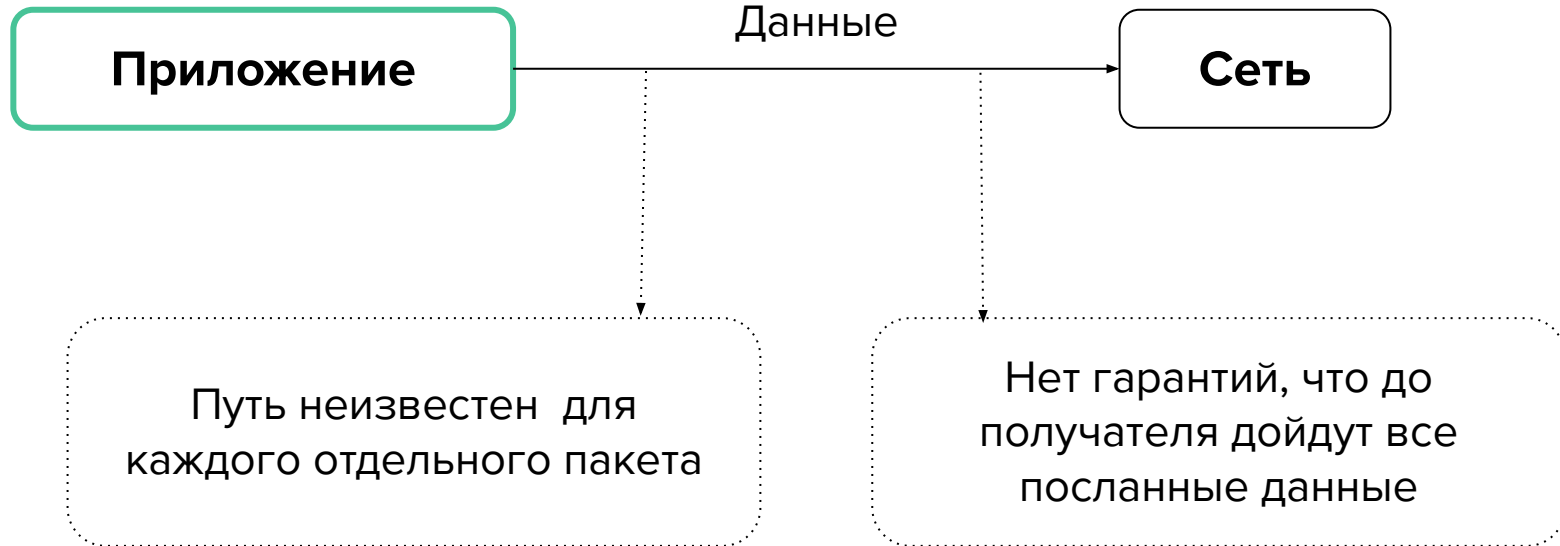
Суть TCP



Назначение протокола TCP

- Надежная доставка данных
- Сборка сегментов на стороне получателя
- Контроль сессии
- Контроль скорости передачи данных
- Мультиплексирование

Проблемы передачи данных



Причины смены маршрута пакетов одной сессии

1

**Авария на одном из
маршрутизаторов**

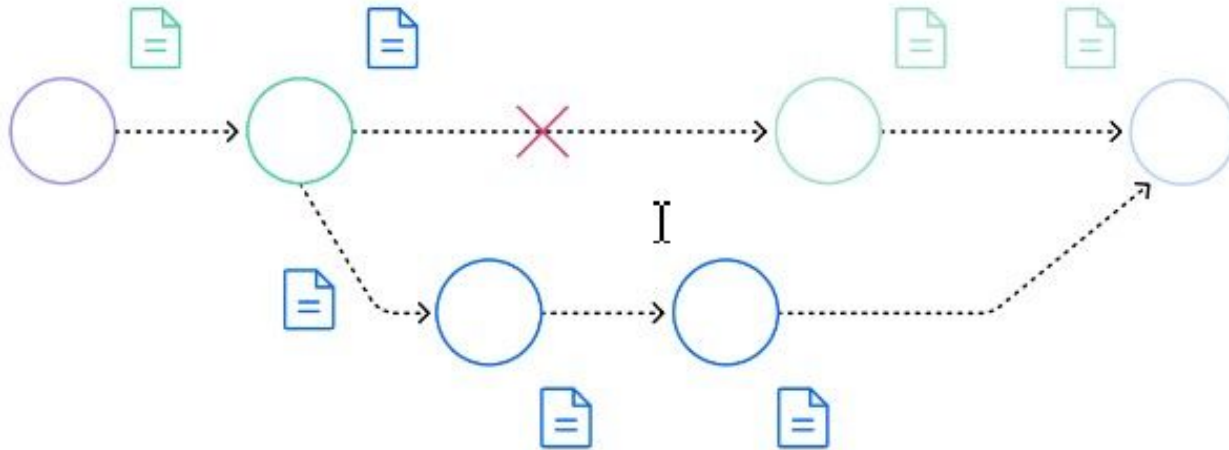
2

**Работа
балансировщика
нагрузки**

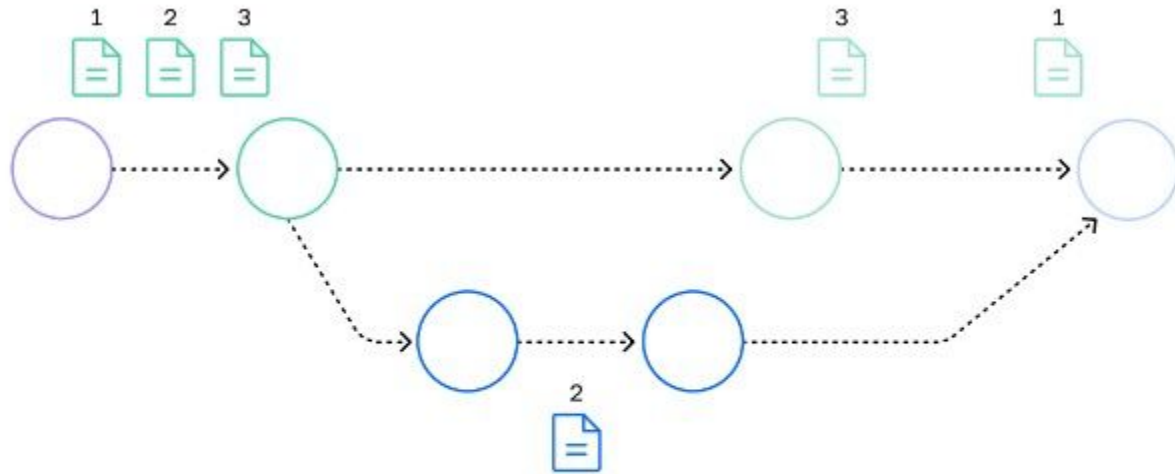
3

**Ошибки
маршрутизации**

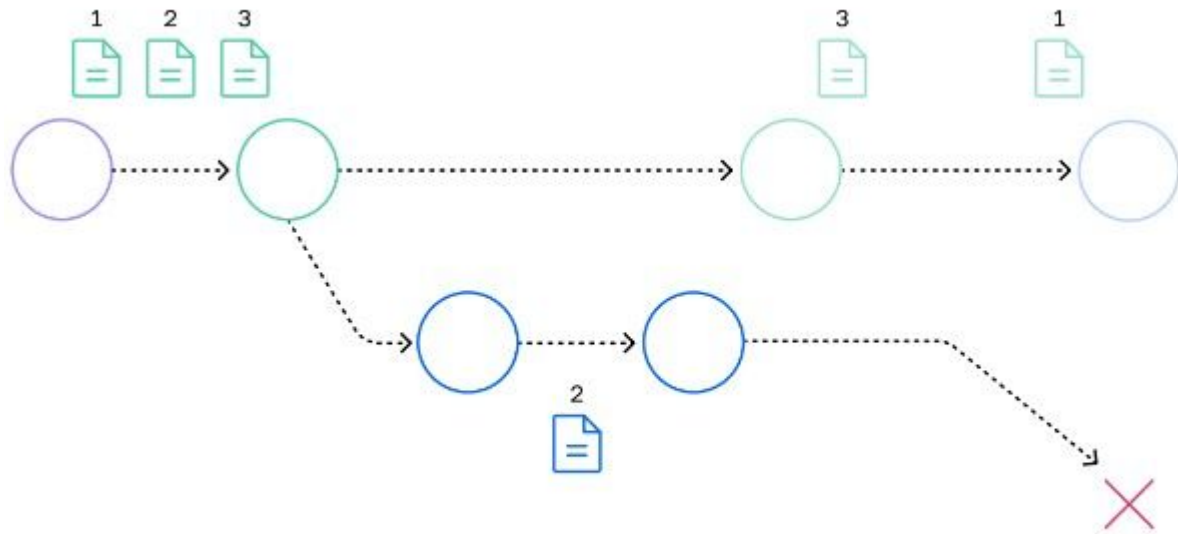
Авария на одном из маршрутизаторов



Работа балансировщика нагрузки

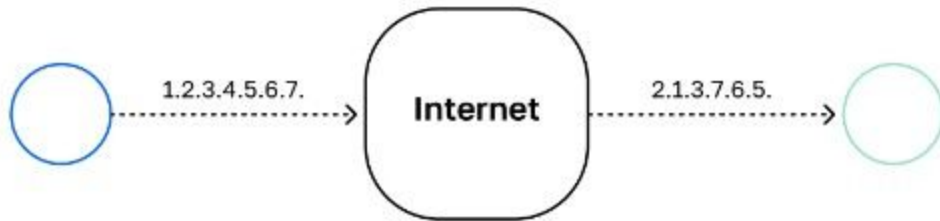


Ошибка маршрутизации

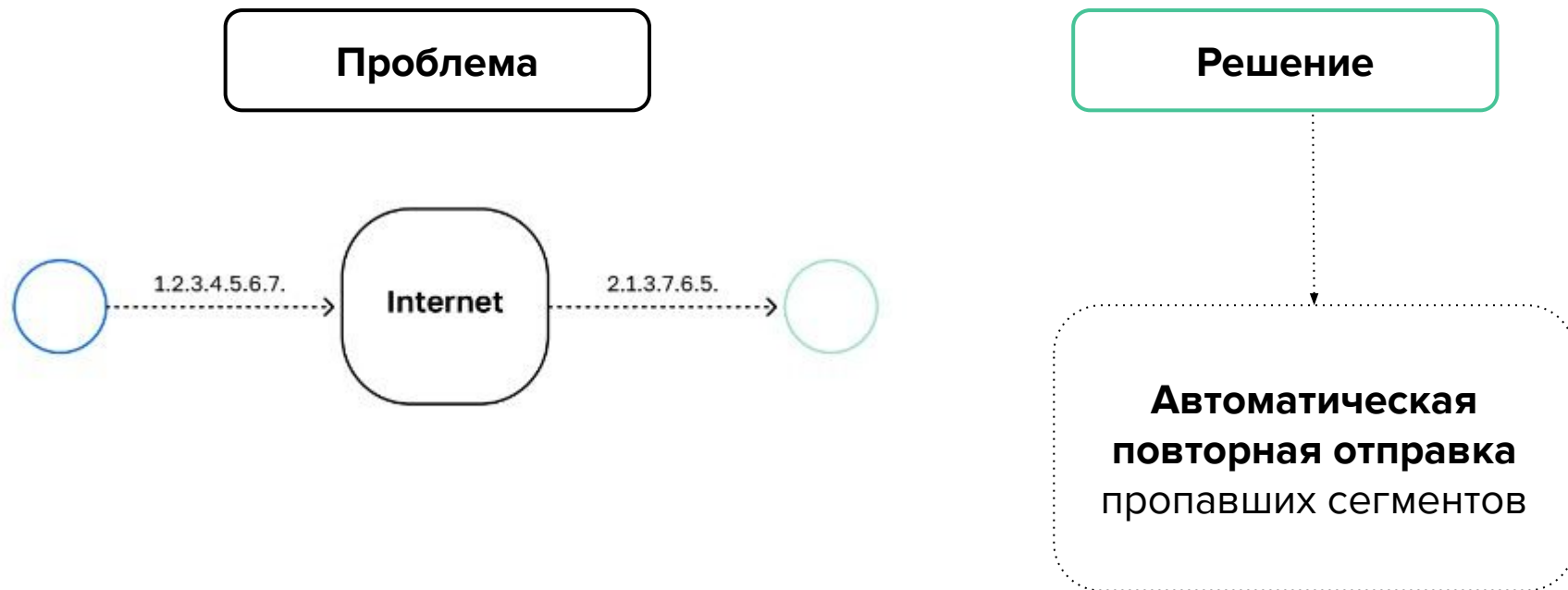


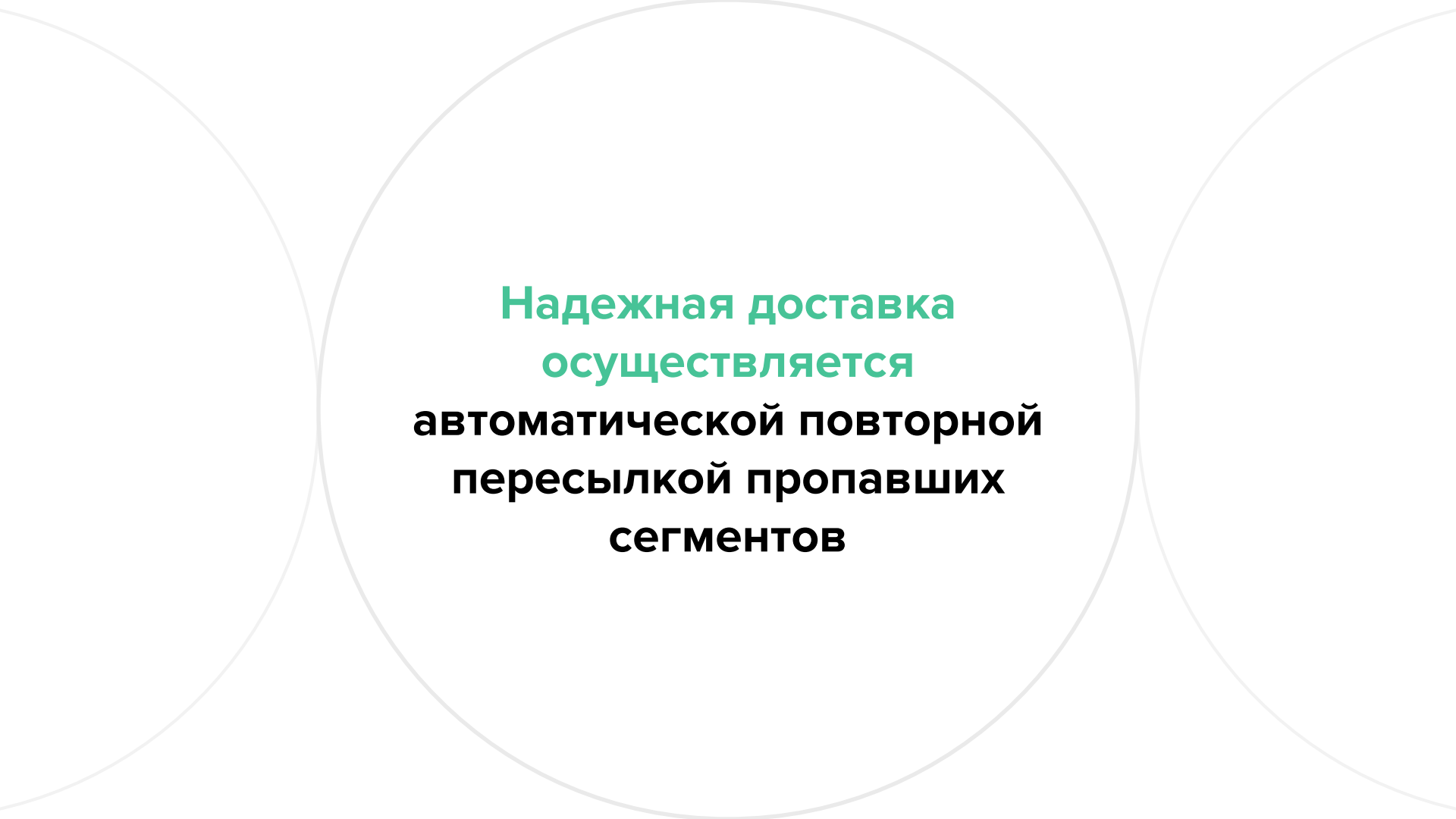
Доставка данных протокола ТСР

Проблема



Доставка данных протокола TCP





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

Надежная доставка данных протокола TCP

- ① Каждый сегмент TCP содержит в заголовке специальное поле
- ② Отправитель высылает какое-то количества сегментов и ждет подтверждения от получателя, с указанием порядкового номера следующего сегмента, который адресат желает получить
- ③ Если такое подтверждение не получено, отправка повторится

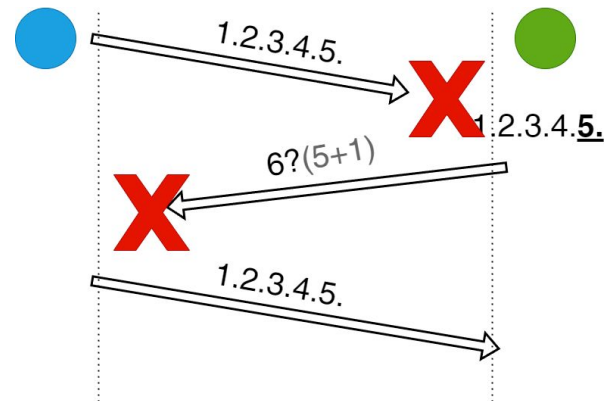
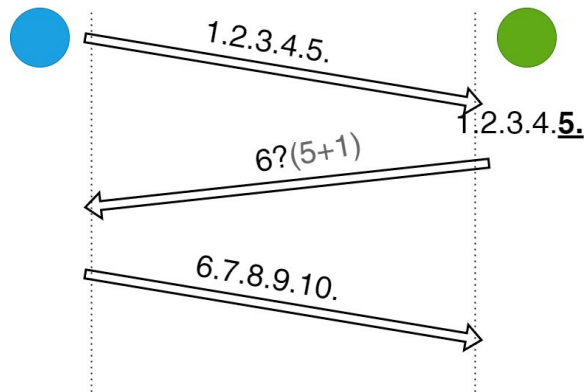
Sequence number

Порядковый номер

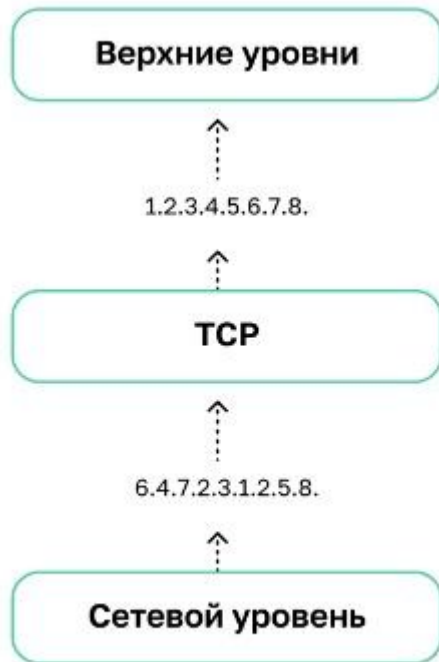
**Acknowledgment
number**

Номер подтверждения

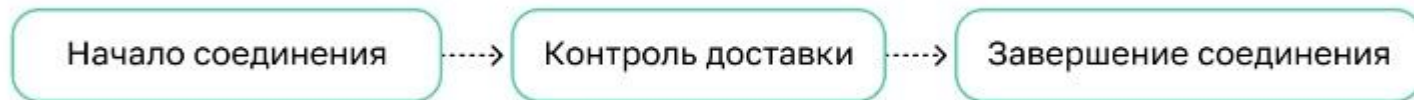
Надежная доставка данных протокола TCP



Сборка сегментов



Контроль сессии





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

Дословный перевод

Three-way handshake



Трехстороннее рукопожатие

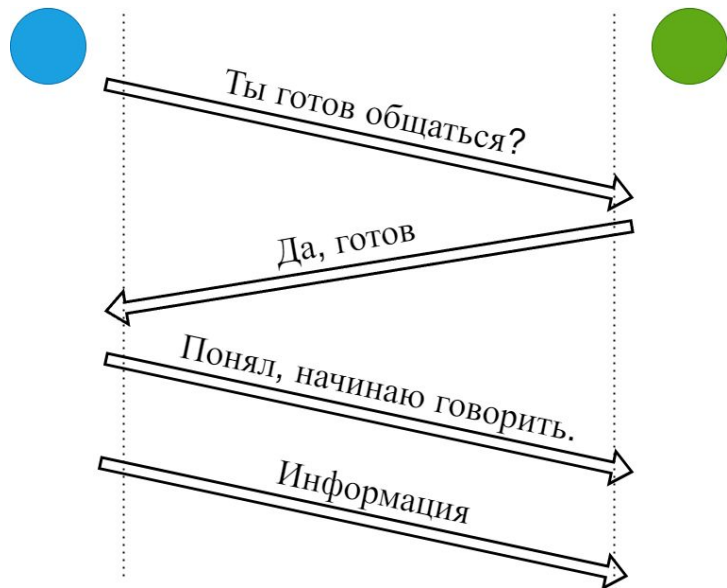
“

Трехстороннее рукопожатие

процесс установки надежного, полнодуплексного соединения, где оба канала могут передавать информацию одновременно, а также они синхронизируют и подтверждают друг друга



Механизм трехстороннего рукопожатия

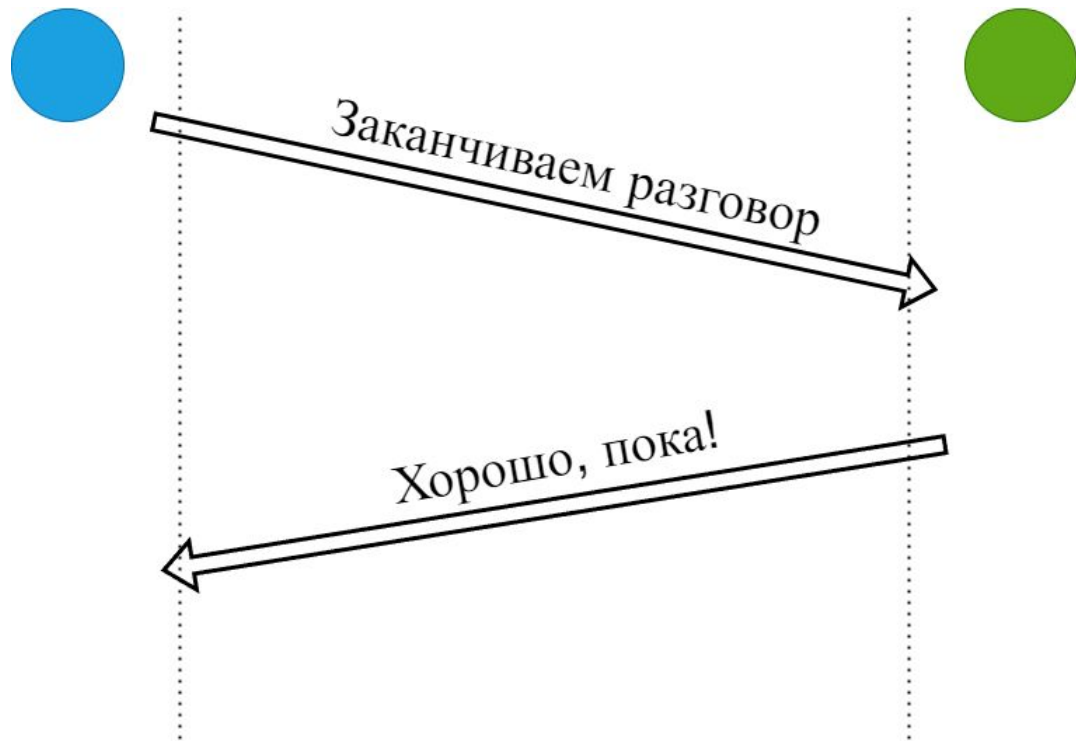


Во время сессии данные контролируются при помощи

Sequence number

**Acknowledgment
number**

Контроль сессии



Дословный перевод

Sliding window




Скользящее окно



Техника скользящего окна

особенность протокола ТСР, которая используется для подбора оптимального количества отправляемых пакетов с сохранением надежной упорядоченной доставки пакетов, а также используется для повышения эффективности, когда канал может иметь большую задержку

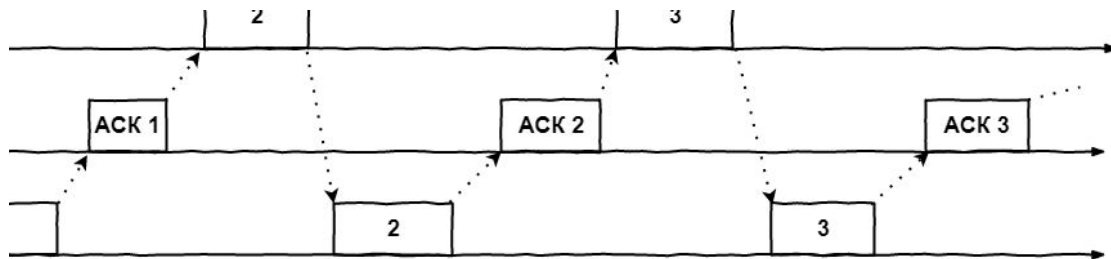




**Отправитель может
динамически менять размер
пересылаемых данных,
анализируя подтверждения от
получателя, благодаря
механизму скользящего окна**

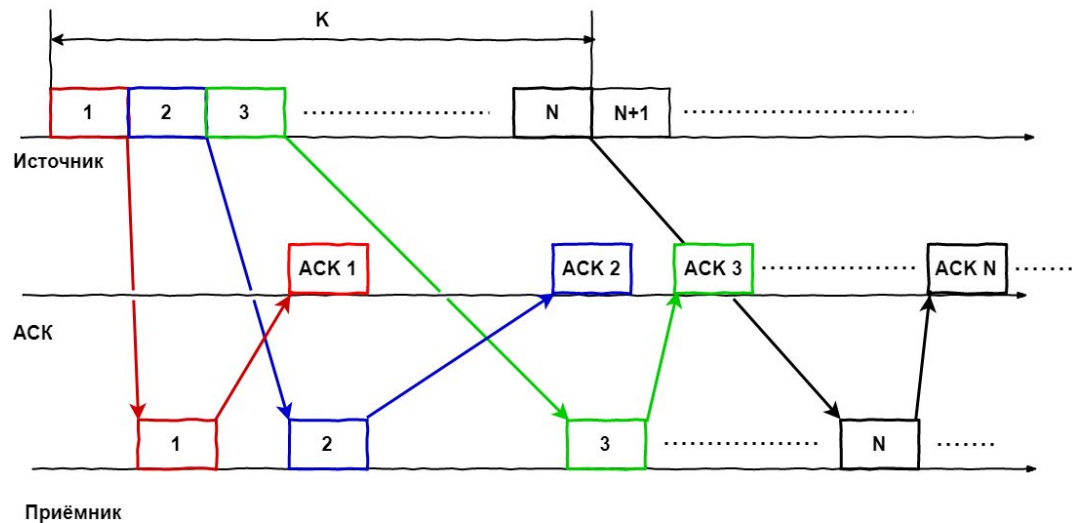
Контроль скорости передачи

Обмен без скользящего окна



Контроль скорости передачи

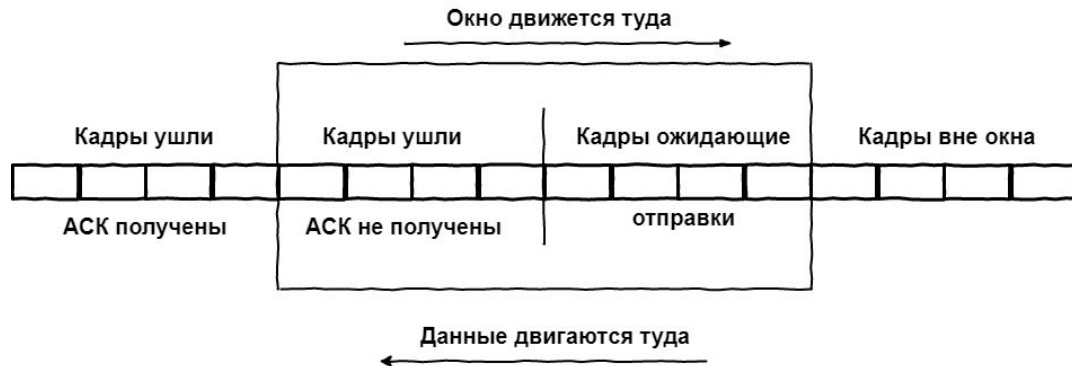
Обмен со скользящим окном



Источник

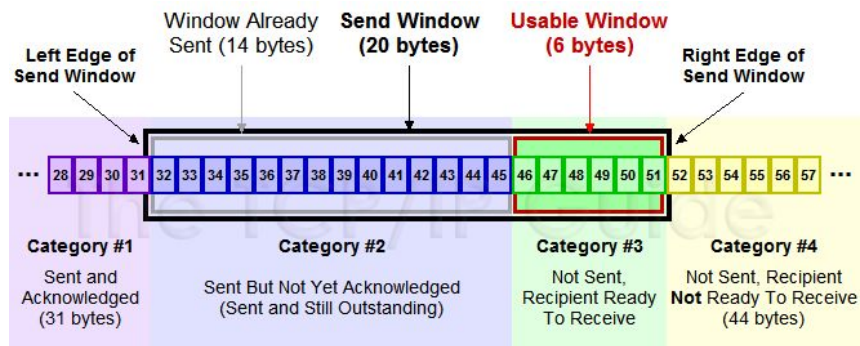
Контроль скорости передачи


Скользящее окно



Контроль скорости передачи

Скользящее окно





**ТСР постоянно
анализирует время прохождения
подтверждений и адаптирует
размер окна под пропускную
способность**

Скорость передачи в изначальной реализации TCP

2^{16}

байт

Максимальное
количество байтов
в одном окне

85

миллисекунд

Круговая скорость
передачи, например

6,17

Мбит/с

Максимальная
скорость передачи

$$2^{16} \text{ байт} * 8 / 0,085 = 6,17 \text{ Мбит/с}$$



RFC 1323

параметр, позволяющий включить масштабирование окон TCP, где значение некоего множителя выбирается в ходе установления сессии



Скорость передачи при использовании RFC 1323



2^{14}

Максимальное
значение множителя



1

Гбайт

Максимальный размер
окна с множителем

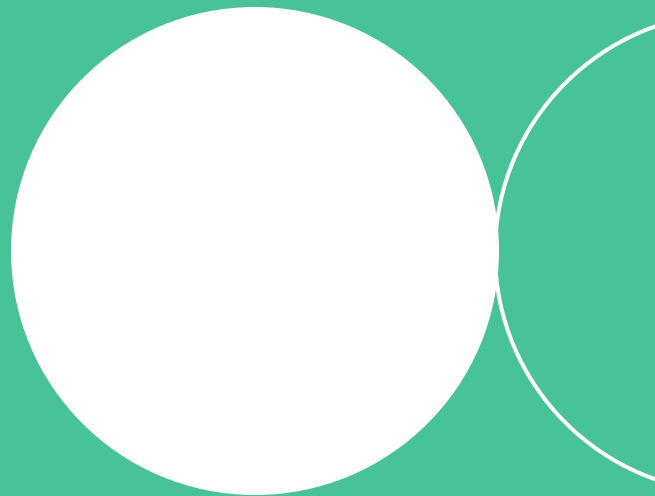
$2^{(16+14)}$ байт = 1 Гбайт

Итоги

- 1 Несмотря на ненадежность сетей IP, протокол TCP спроектирован так, чтобы гарантировать доставку информации получателю
- 2 Надежность доставки данных через TCP гарантируется контролем сессии с помощью сложных механизмов установления соединения («трехстороннее рукопожатие») и завершения («двухстороннее рукопожатие»)
- 3 Во время передачи информации протокол TCP контролирует доставку сегментов с помощью подтверждения получения, а скорость передачи механизмом «скользящее окно»

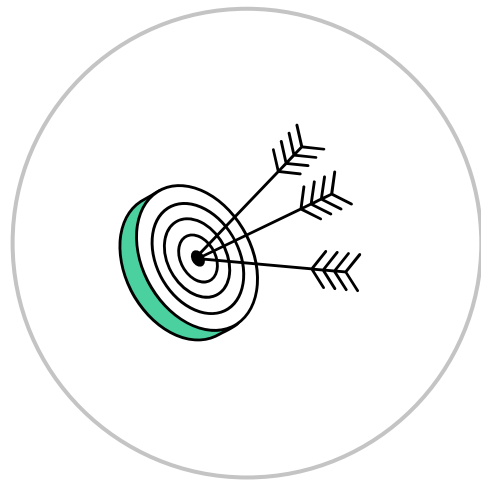


Заголовок ТСР



Цели темы

- Познакомиться с заголовком TCP
- Разобраться с назначением основных полей TCP
- Понять назначение основных флагов заголовка TCP



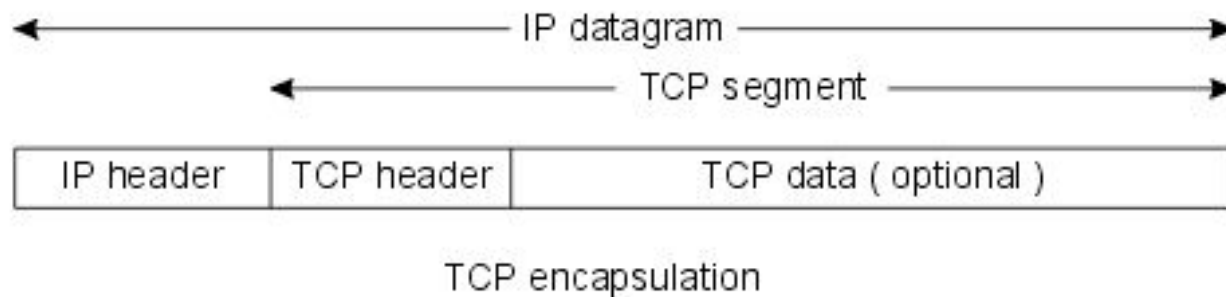


Заголовок ТСП

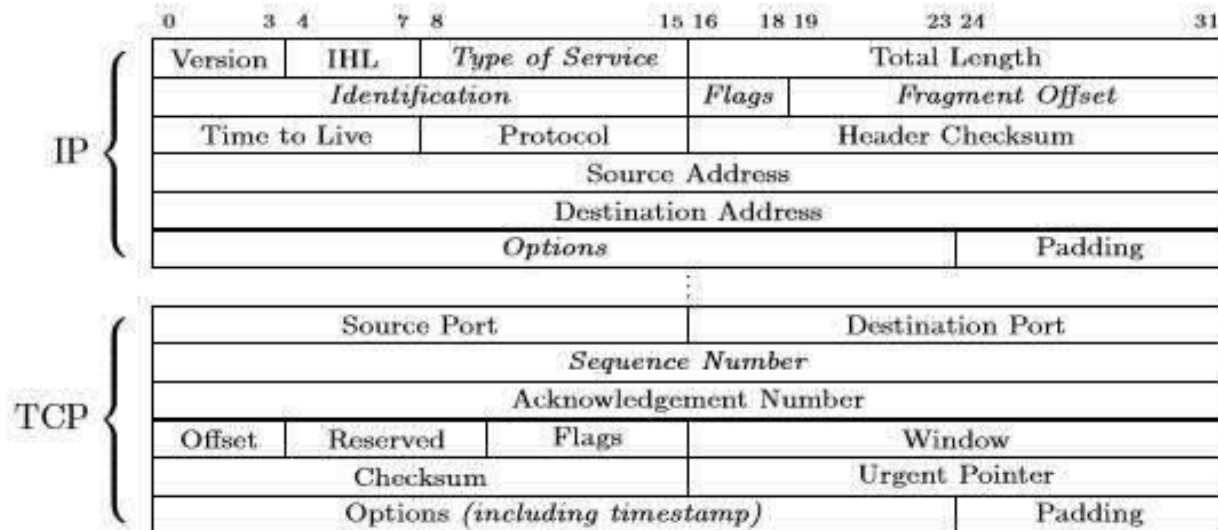
служебная информация (метаданные), которая добавляется к полезной нагрузке (payload) на транспортном уровне для решения всех задач протокола ТСП



Заголовок TCP в потоке информации

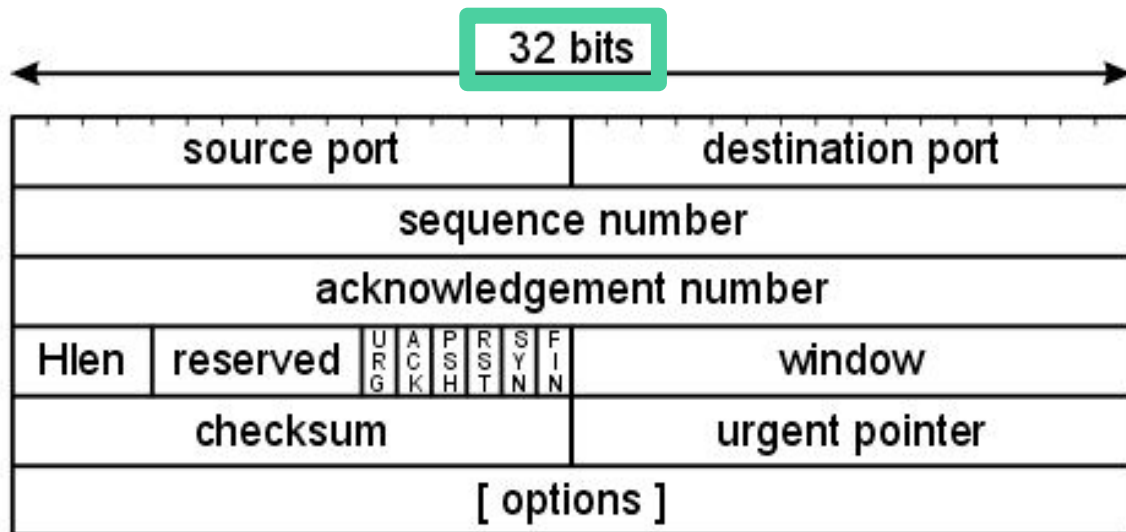


Заголовки протоколов IP и TCP вместе



Заголовок TCP

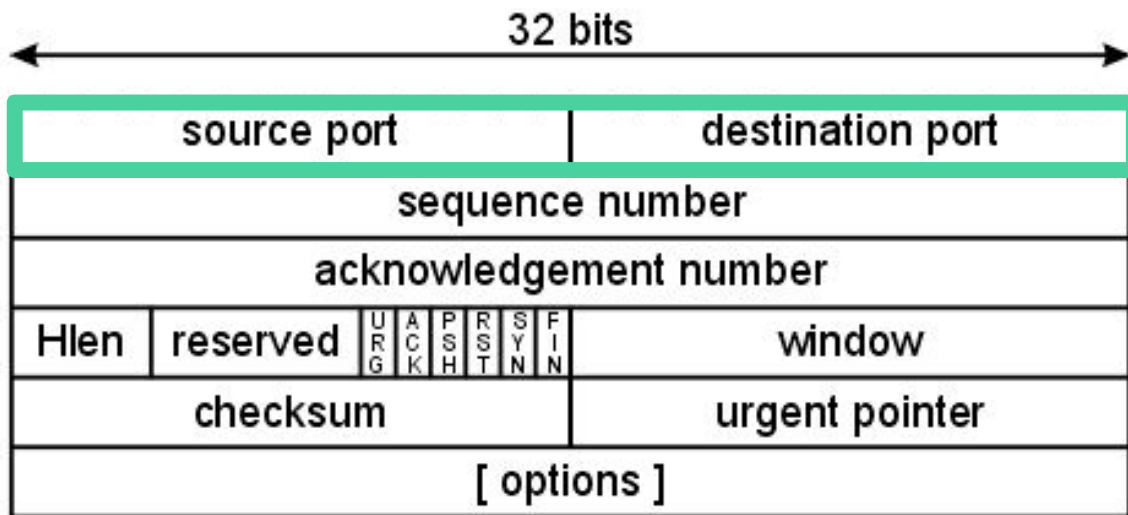
TCP header format



32-битные слова

Заголовок TCP: основные поля

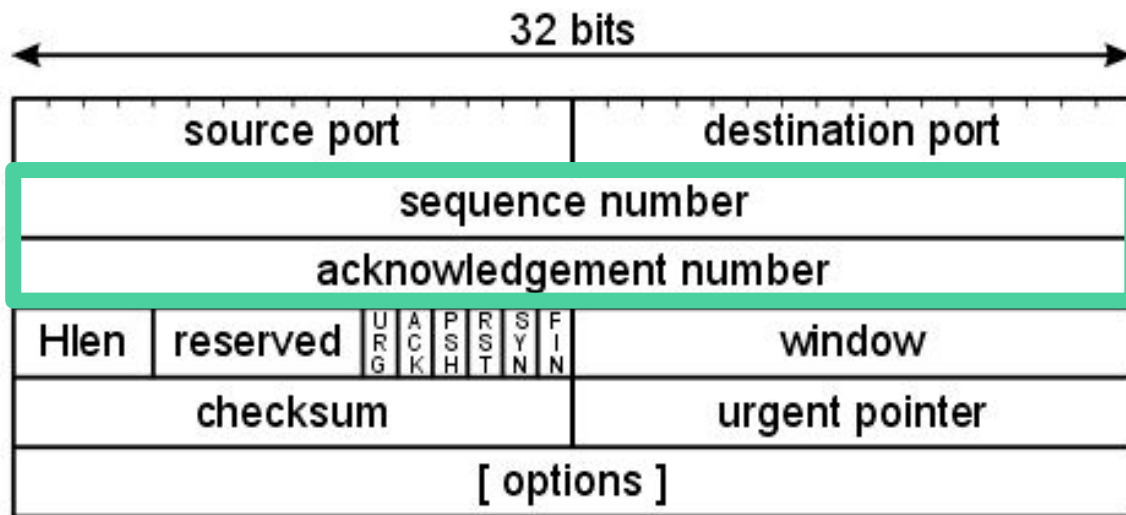
TCP header format



Номер портов
получателя и отправителя

Заголовок TCP: основные поля

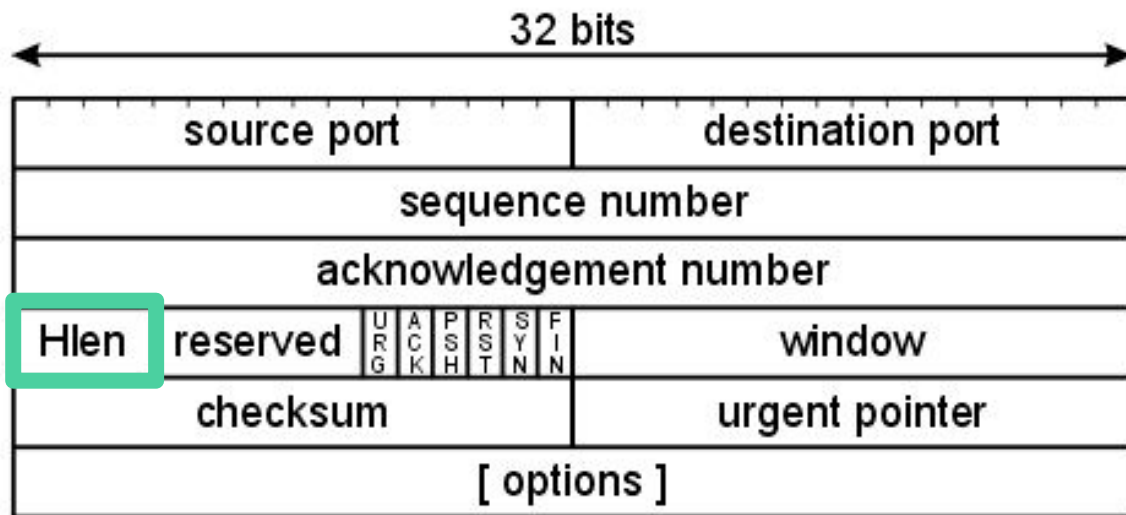
TCP header format



Порядковый номер
Номер подтверждения

Заголовок TCP: основные поля

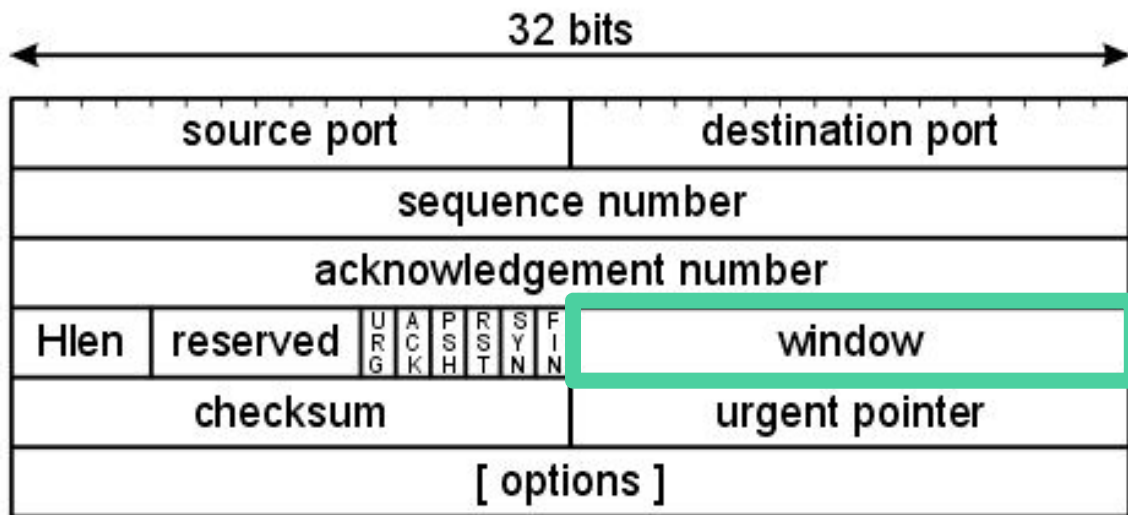
TCP header format



Длина
самого заголовка TCP

Заголовок TCP: основные поля

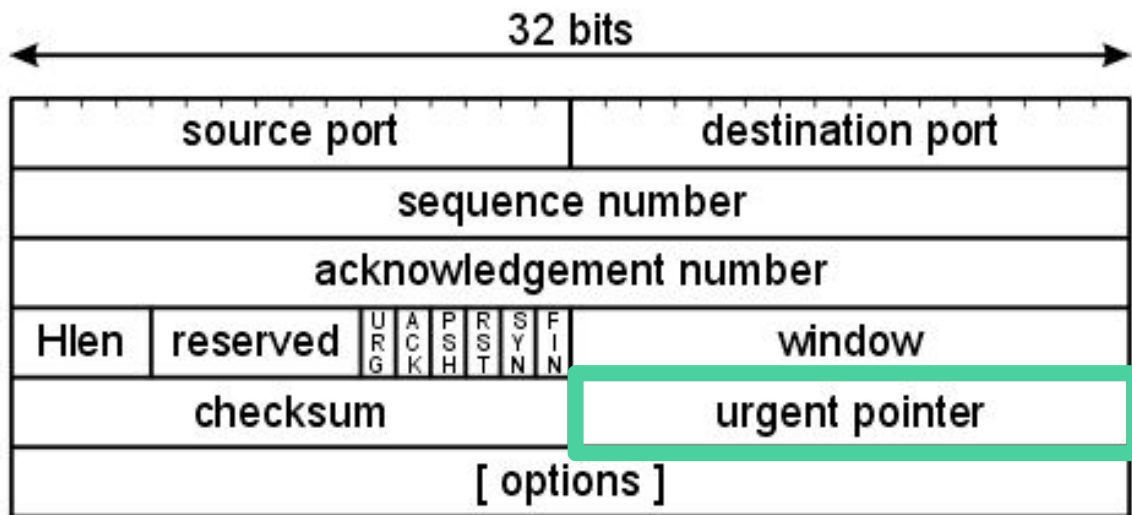
TCP header format



Размер окна

Заголовок TCP: основные поля

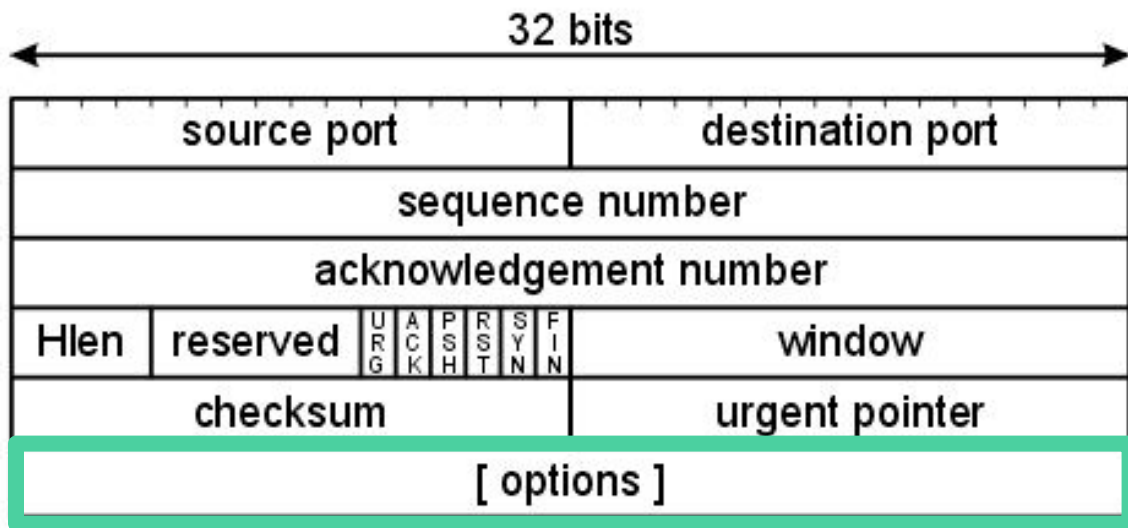
TCP header format



Признак важности
данного сегмента

Заголовок TCP: основные поля

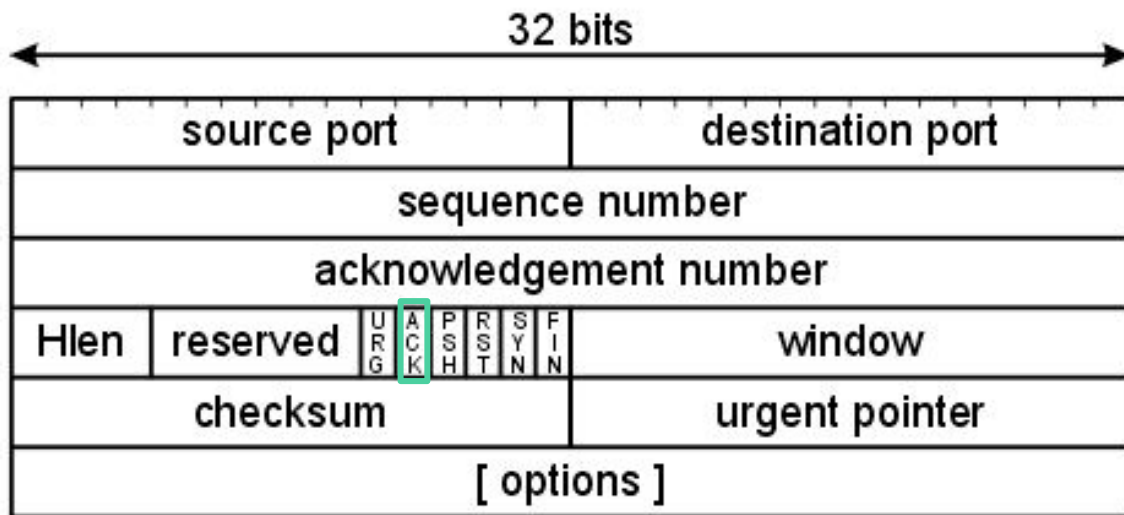
TCP header format



Необязательное поле
для разработчиков

Заголовок TCP: основные флаги

TCP header format

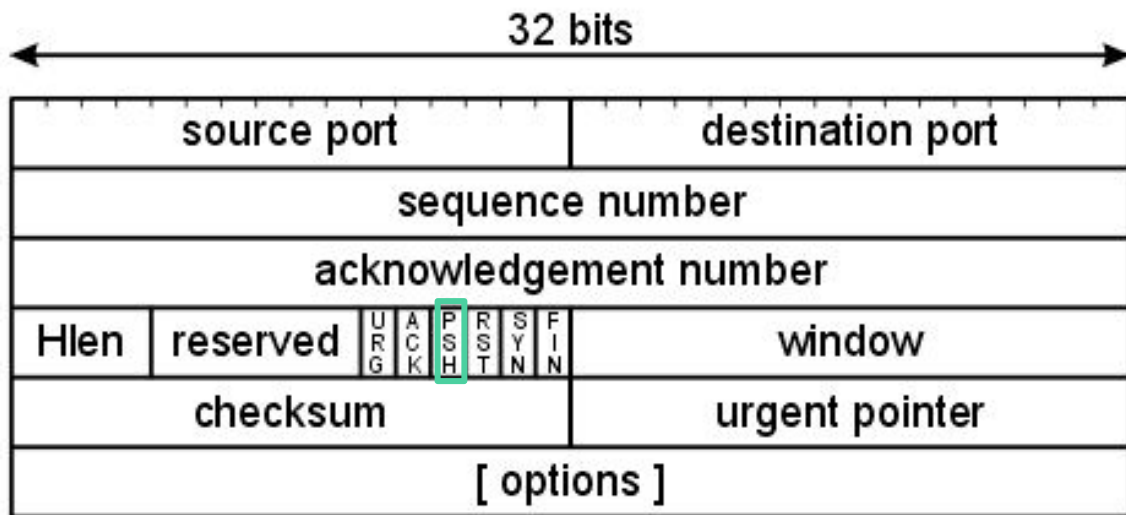


ACK

содержит значение
номера подтверждения
в поле подтверждения

Заголовок TCP: основные флаги

TCP header format

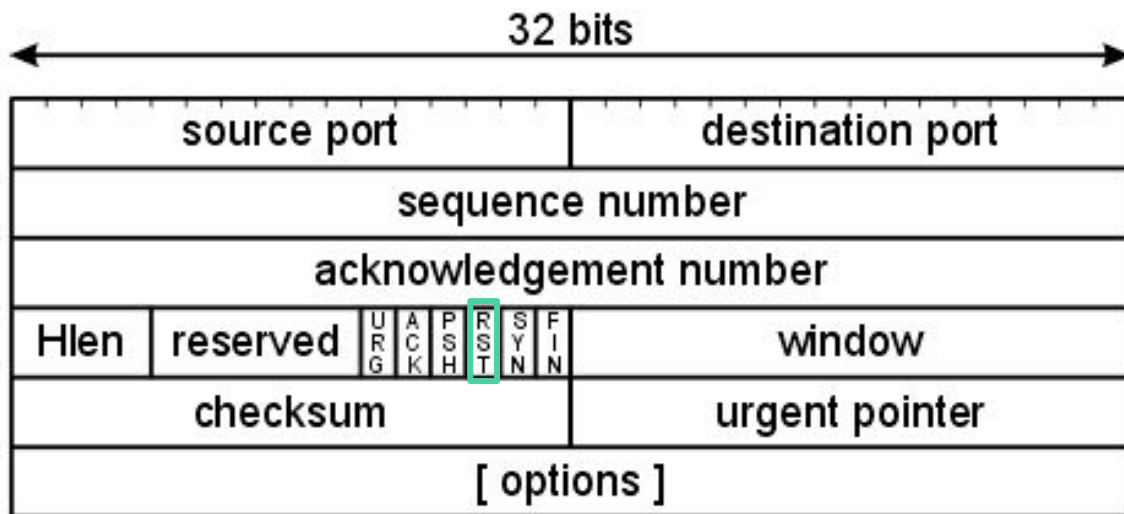


PSH

получатель передает данные из буфера в приложение и сразу же отправляет сообщение с подтверждением

Заголовок TCP: основные флаги

TCP header format

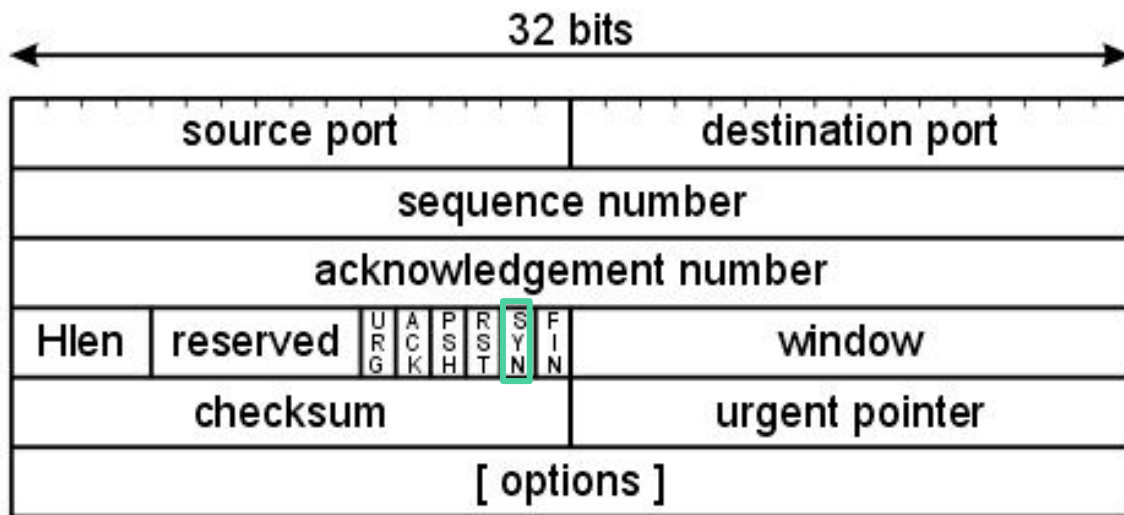


RST

сбрасывает соединения

Заголовок TCP: основные флаги

TCP header format

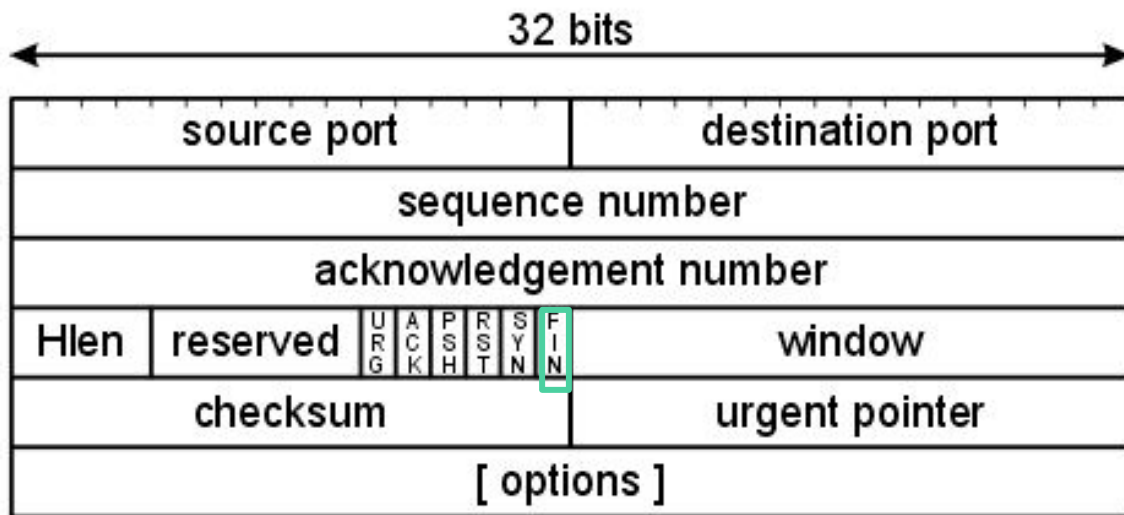


SYN

создает соединения

Заголовок TCP: основные флаги

TCP header format



FIN

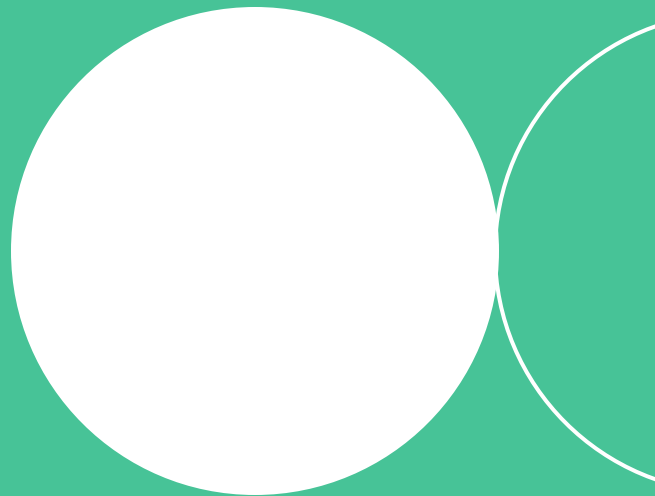
завершает соединения

Итоги

- 1 В отличии от метаданных предыдущих уровней, протокол TCP широко полагается на использование флагов для контроля передачи данных, важнейшие это SYN, ACK и FIN
- 2 Поля source port и destination port служат для реализации задач по мультиплексированию
- 3 Поле acknowledgement number обрабатывается только при наличии 1 во флаге ACK
- 4 Заголовок TCP все еще содержит ресурсы для улучшений, благодаря зарезервированным и неиспользуемым битам в заголовке

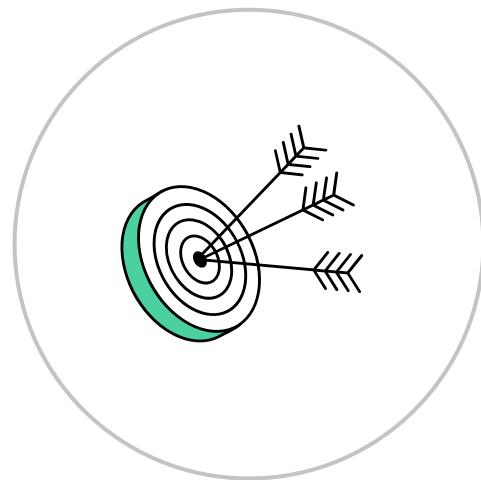


Порты ТСР



Цели темы

- Понять, каким образом порты транспортного уровня обеспечивают мультиплексирование
- Познакомиться с диапазонами портов и их назначением






Мультиплексирование

процедура приема данных протоколами TCP и UDP, поступающих от нескольких различных прикладных служб





**Для того, чтобы понимать
какой процесс посылает или
принимает данные, применяется
метод определения конечной
точки транспортного уровня**



Порт

адрес транспортного уровня



Аналогия сетевого / транспортного уровня

Адрес дома



/

Номер квартиры / офиса



Аналогия сетевого / транспортного уровня

Аэропорт



/

Терминалы в аэропорту





Номер порта

**идентификатор приложения на хосте получателя
и/или отправителя**



**Номер порта – всегда
целое положительное
число от 0 до 65535**

Номера портов

0 - 1023

общеизвестные /
системные

1024 - 49151

зарегистрированные /
пользовательские

49152 - 65535

динамические /
частные

Общеизвестные порты

20 и 21

FTP

22

SSH

23

TELNET

25

SMTP

53

DNS

80

HTTP

123

NTP

443

HTTPS

Зарегистрированные / пользовательские порты

**Зарегистрированные
программы или
протоколы**

1721 - PPTP VPN
2041, 2042 - Mail.ru Агент
3389 - Microsoft RDP
13000 - Kaspersky Security
Center

**Назначение
нестандартных
значений протоколов
для сокрытия от
внешнего
обнаружения рабочих
процессов**

**В любых
пользовательских
целях**

Частные / динамические порты

Создания исходящих
соединений с различными
сервисами

Пример

Открытие сайта netology.ru в
новой вкладке браузера приведет
к тому, что новому процессу от
браузера будет назначен
случайный порт из диапазона

Wireshark

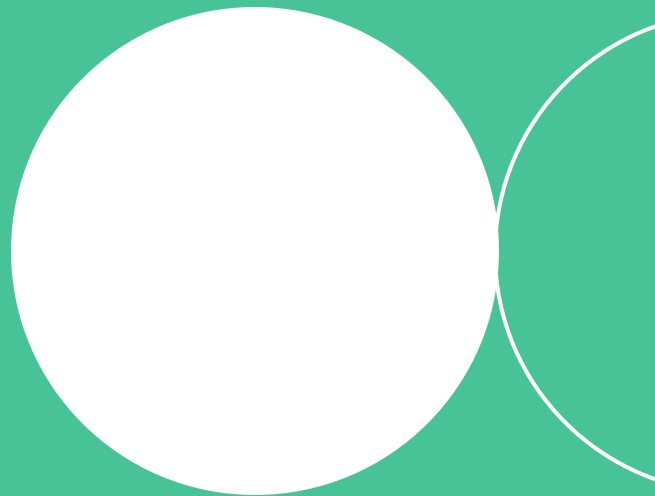
```
▶ Internet Protocol Version 4, Src: 192.168.88.254, Dst: 8.238.89.254
▲ Transmission Control Protocol, Src Port: 51354, Dst Port: 80, Seq: 1, Ack: 1, Len: 280
    Source Port: 51354
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 280]
    Sequence number: 1 (relative sequence 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
```


Итоги

- 1 Порты - это адреса транспортного уровня, помогающие определить к какому процессу относится полученная или отправленная информация
- 2 Динамический диапазон портов нужен для создания исходящих подключений к удаленным ресурсам. Если весь диапазон будет исчерпан, могут возникнуть проблемы с новыми подключениями к сетевым ресурсам
- 3 С помощью утилиты Wireshark посмотрели, как выглядит реальный заголовок TCP

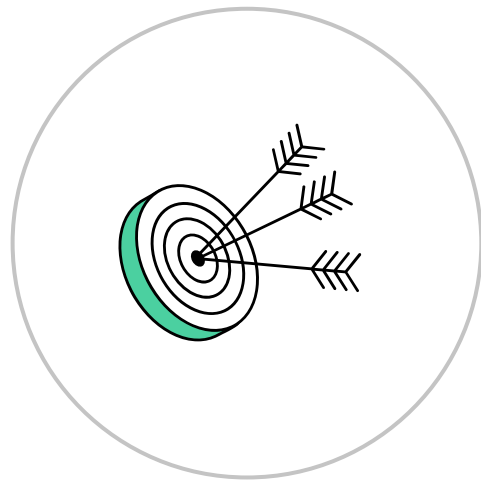


ТСР: сокет



Цели темы

- Познакомиться с концепцией сокетов, историей появления
- Узнать, чем сокет улучшает сетевое взаимодействие
- Разобраться в типах сокетов и их особенностях





Сетевой сокет

структура, которая определяет конечную точку во время сетевого обмена данными



**Сетевые сокеты впервые
появились в 1983 году в
ОС UNIX 4.2BSD**

Для TCP/IP сокетом является сочетание трех параметров сессии:

1

Транспортного
протокола

(TCP, UDP)

2

Номера порта

(0 - 65535)

3

IP-адреса



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

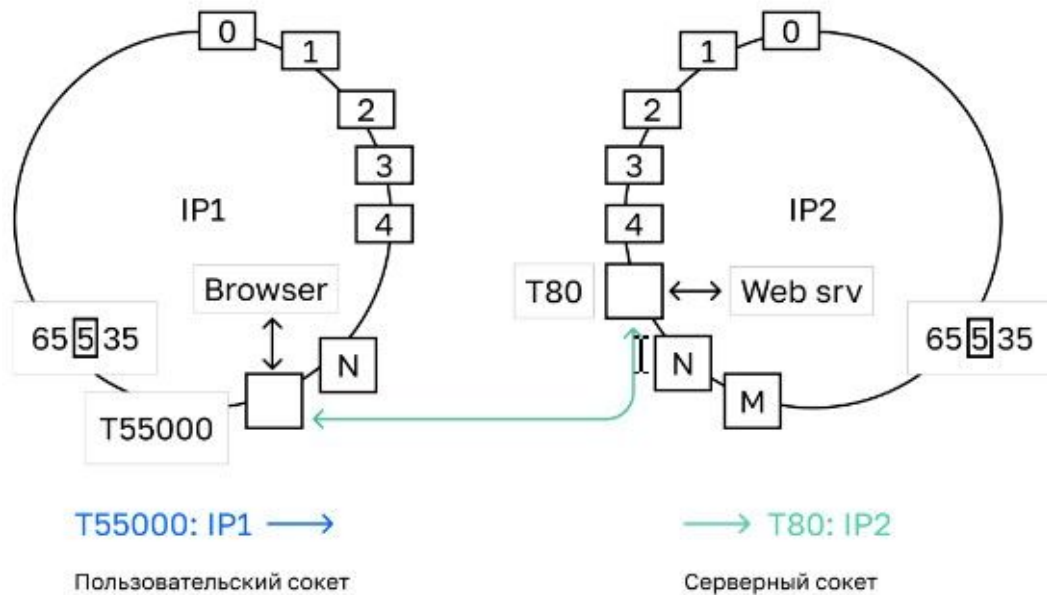


Парные сокеты

локальный и соответствующий удаленный сокеты



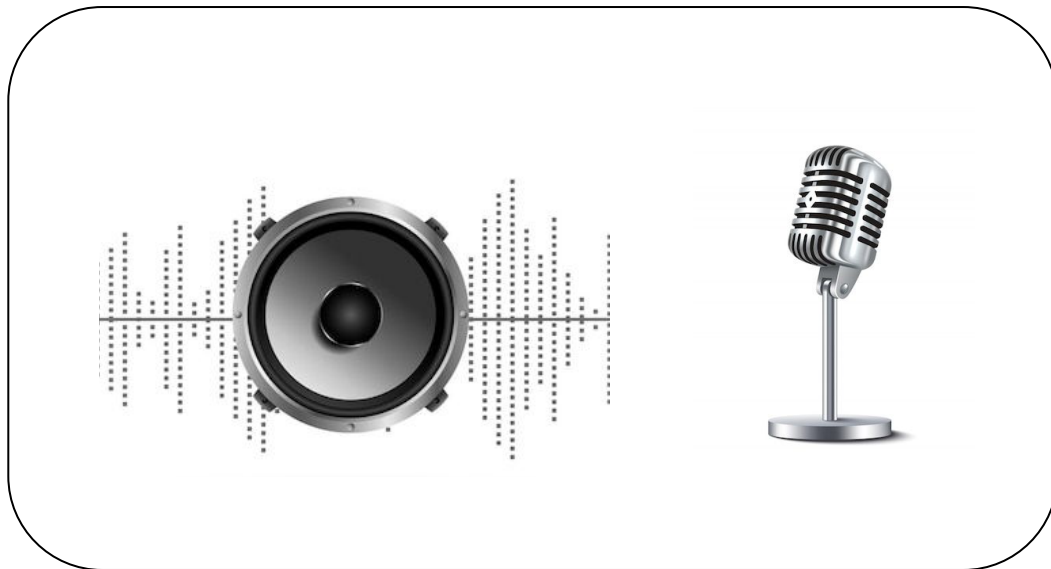
Сокеты



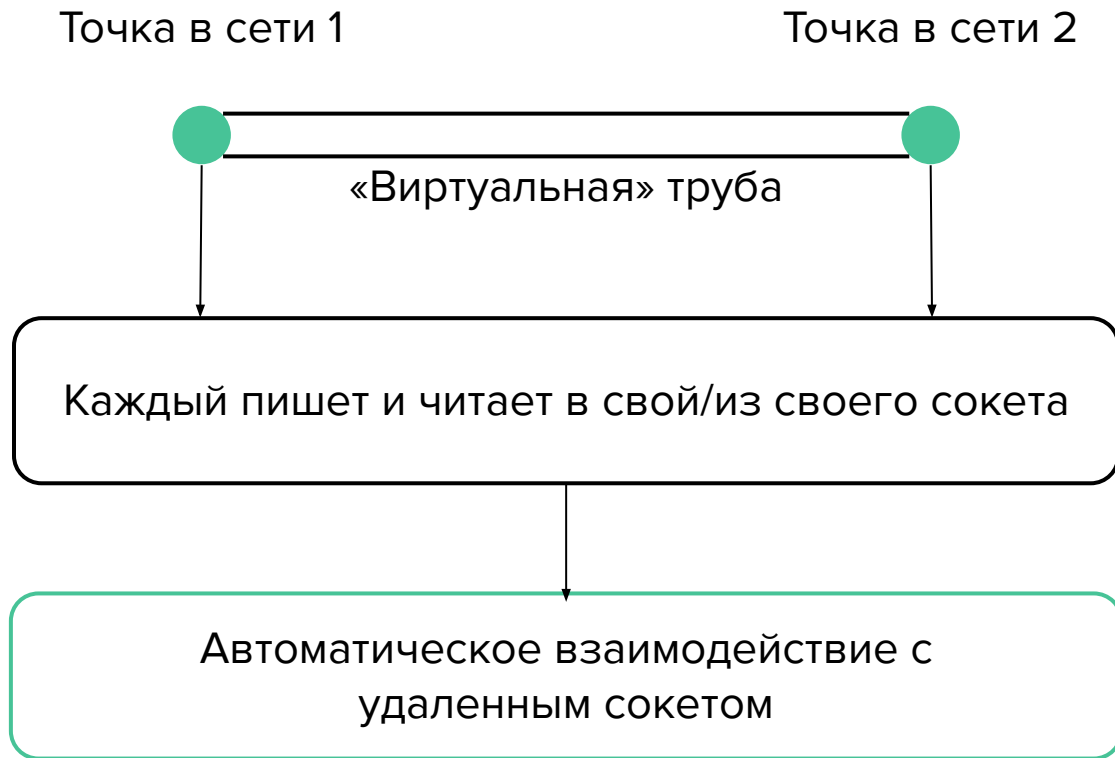
Парный сокет имеет вид T.55000:IP1<->80:IP2

Аналогия сокетов

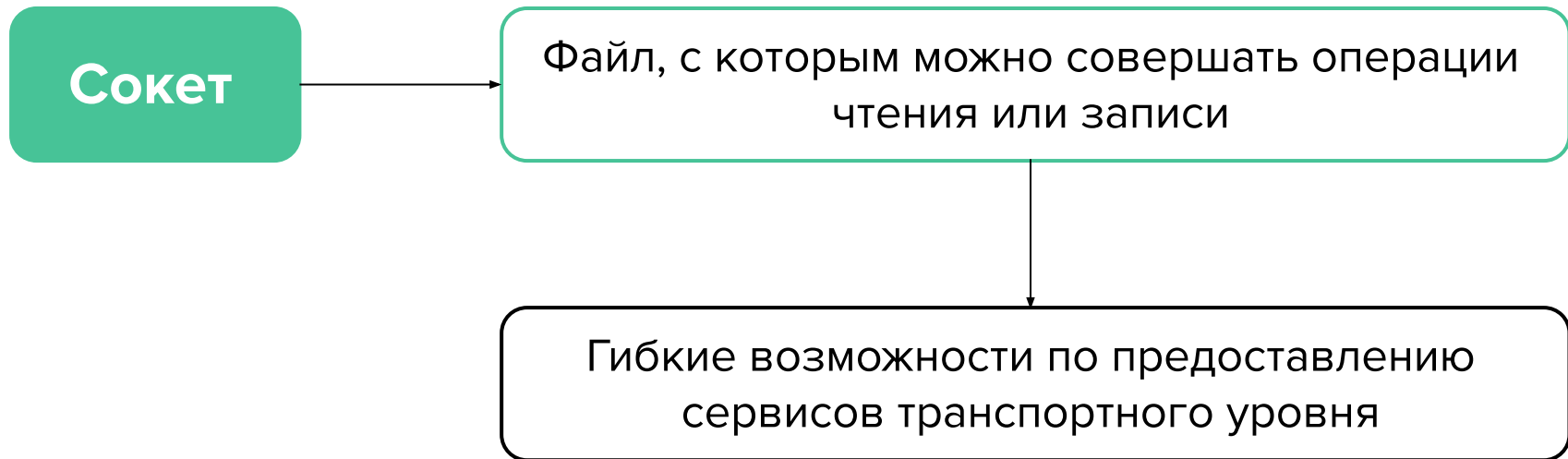
Динамик и микрофон
при разговоре по
мобильному телефону



Аналогия сокетов



Суть сокета



Типы сокетов по подключению



Потоковый
(TCP)

Сокет
дейтаграмм
(UDP)

Сырой (raw)
сокет
(IP-пакет с
данными)

Типы сокетов по функциям

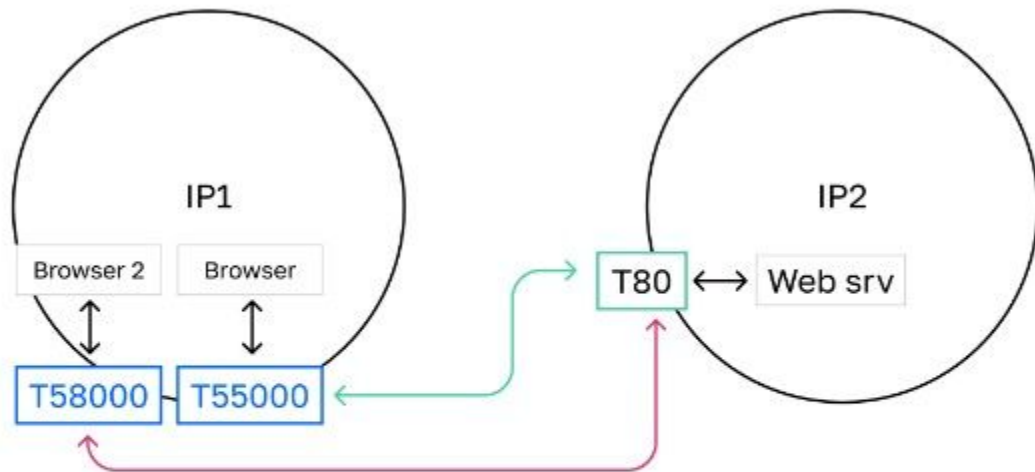
Клиентский

Всегда один

Серверный

Может быть несколько
одинаковых на сервере


Сокеты



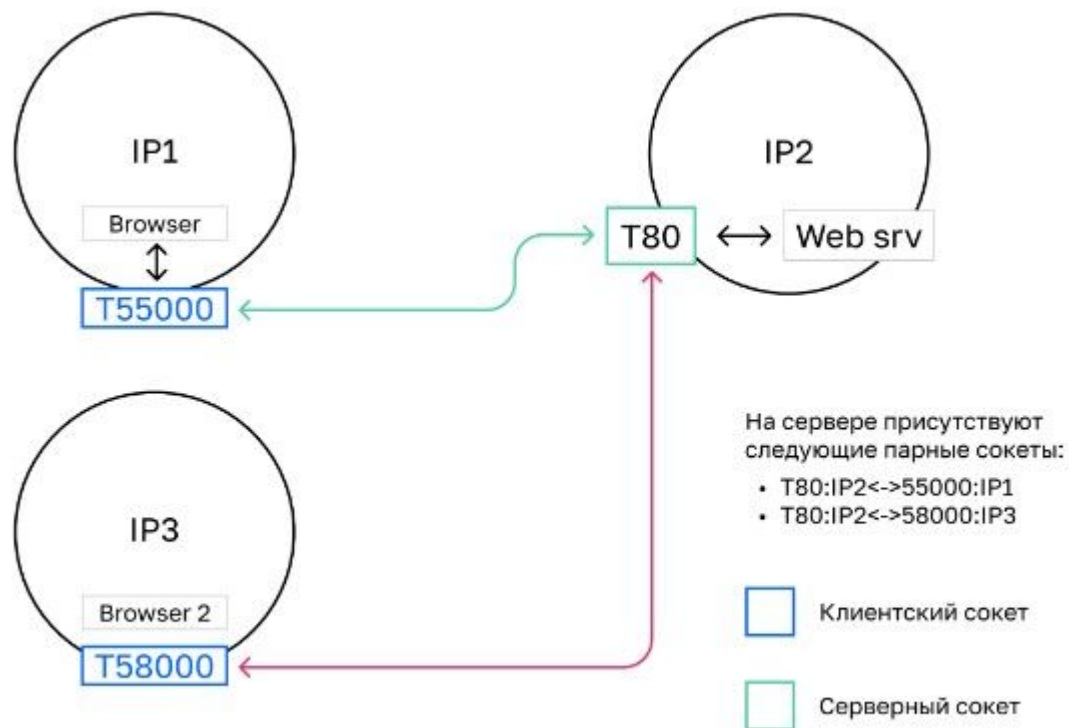
На сервере присутствуют следующие парные сокеты:

- T80:IP2<->55000:IP1
- T80:IP2<->58000:IP1

 Клиентский сокет

 Серверный сокет

Сокеты

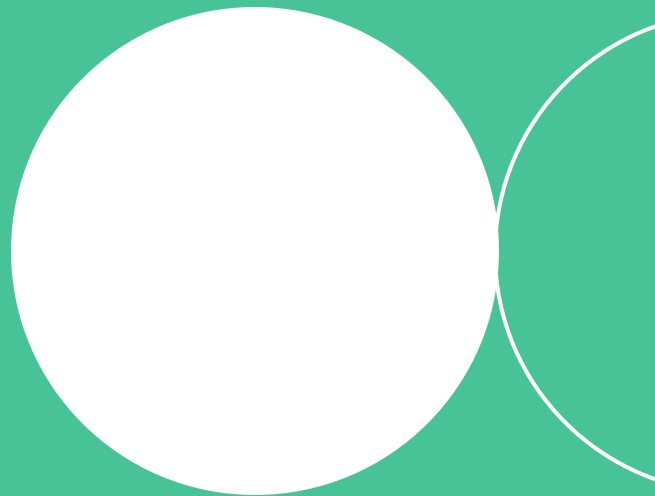


Итоги

- 1 Сокеты как абстракция помогают гибко управлять сервисами транспортного уровня
- 2 Комбинация из локального и удаленного сокета составляет парный сокет, позволяющий однозначно идентифицировать соединение
- 3 Сокеты могут делиться как по типу протокола (TCP, UDP, RAW), так и по типу функционирования (клиентские/серверные)

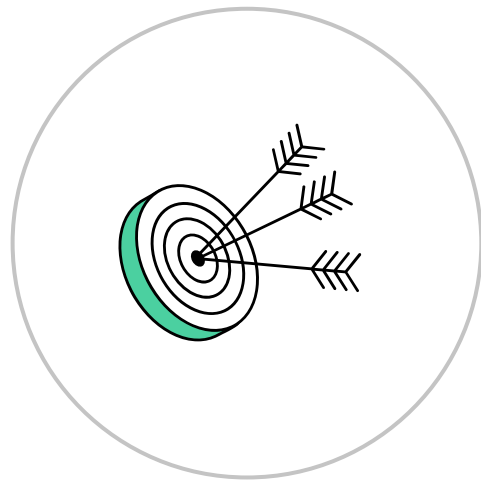


ТСР: установление соединения



Цели темы

- Узнать об установке соединения в протоколе TCP
- Разобраться, из каких шагов состоит установление соединения
- Понять, как меняется состояние сокетов на каждом этапе



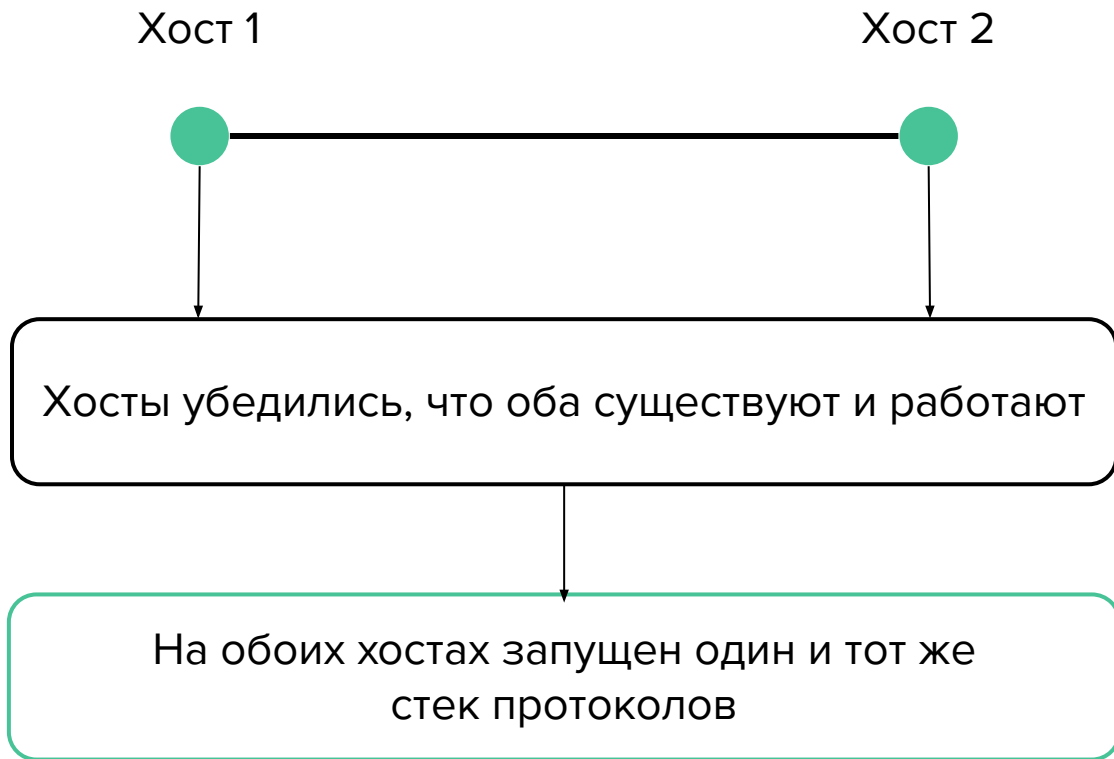



Установление соединения

**процесс обмена между двумя удаленными точками,
в результате которого стороны начинают
обмен информацией**



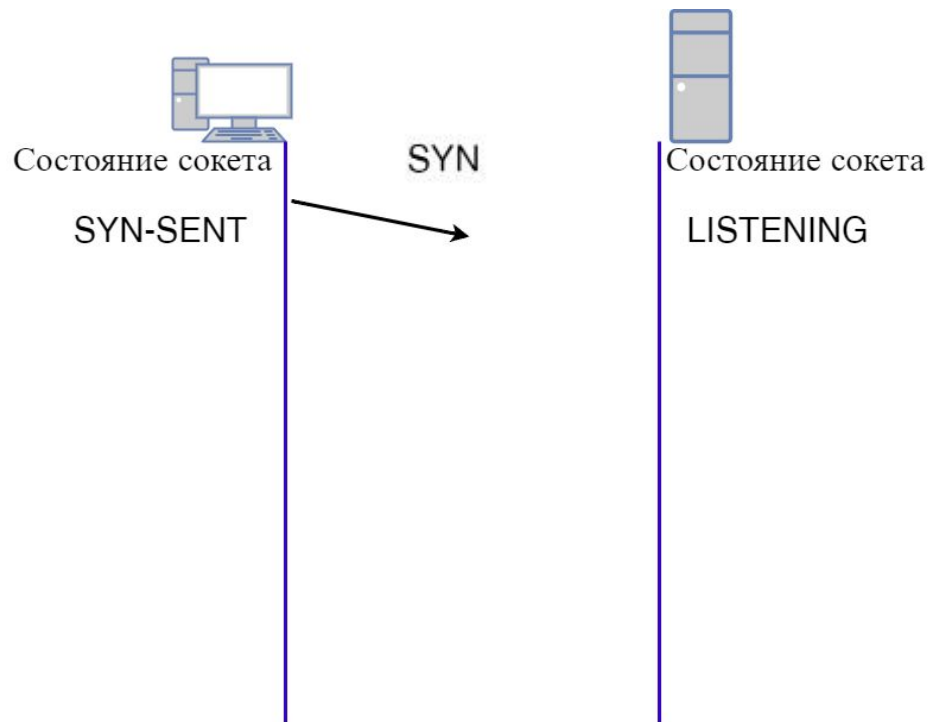
Установление соединения между двумя хостами



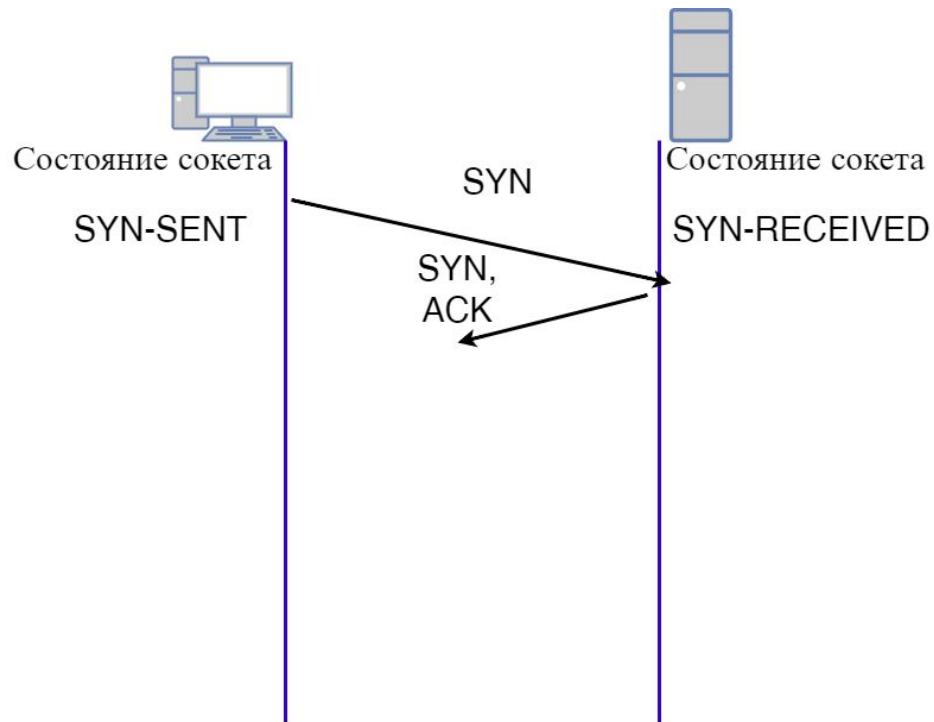


**Протокол ТСР устанавливает
соединение с помощью
трехстороннего рукопожатия**

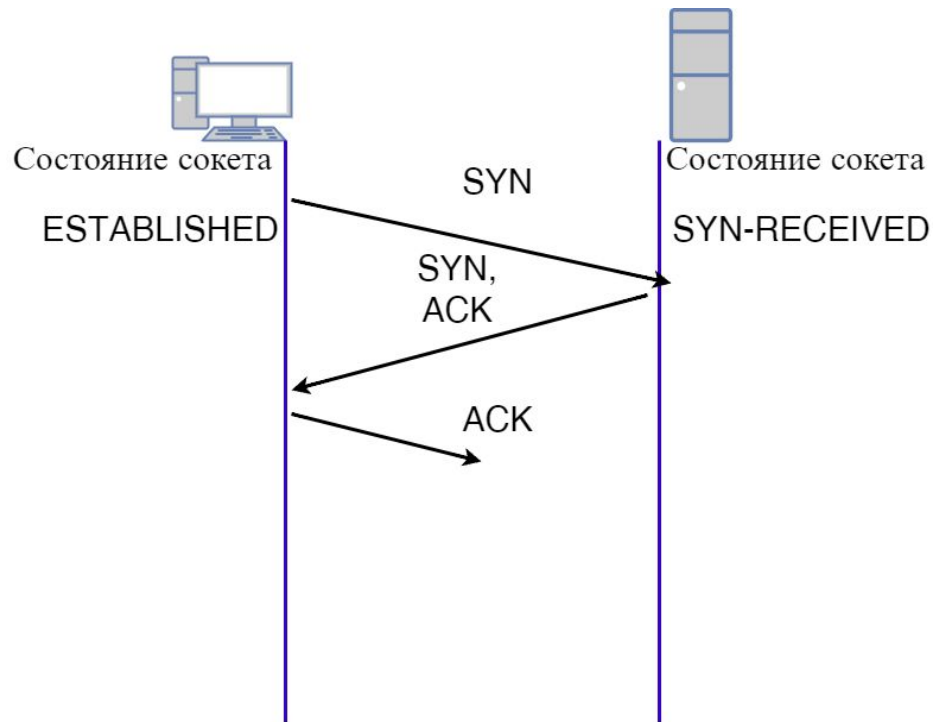
Установка соединения



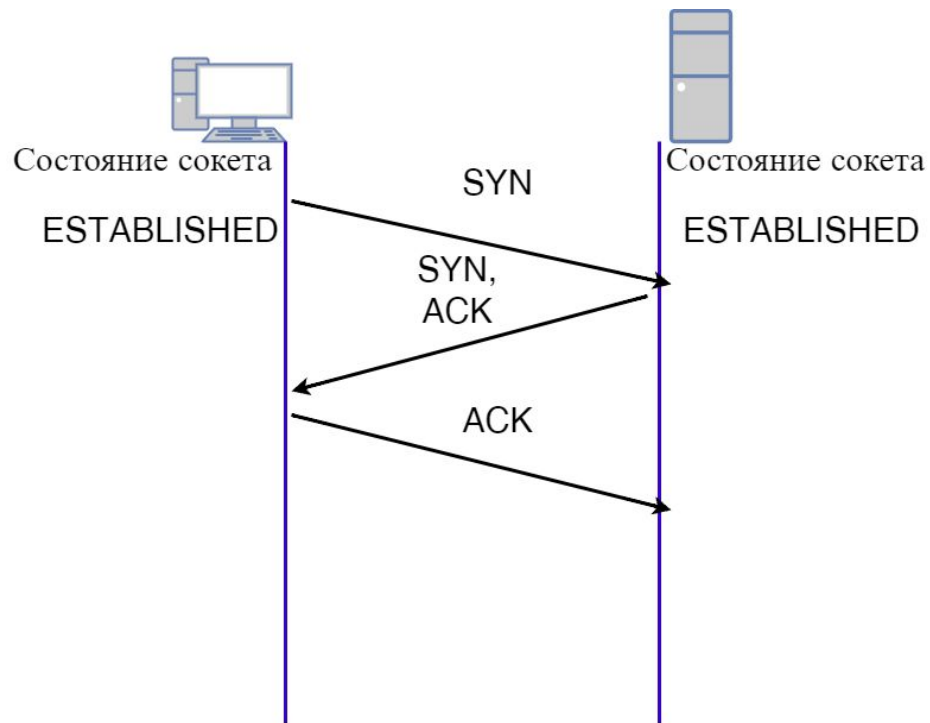
Установка соединения



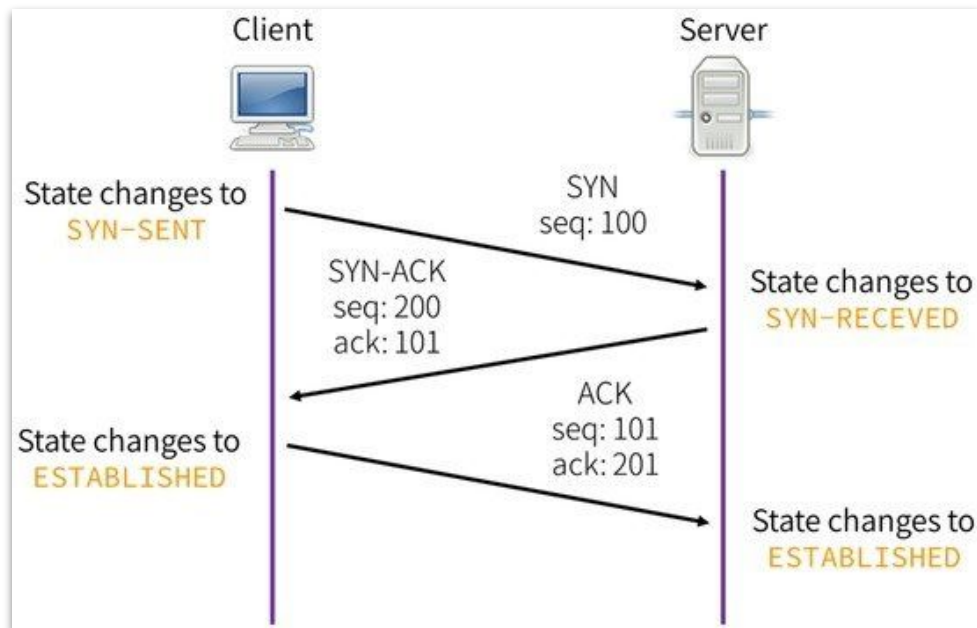
Установка соединения



Установка соединения



Установка 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]
```

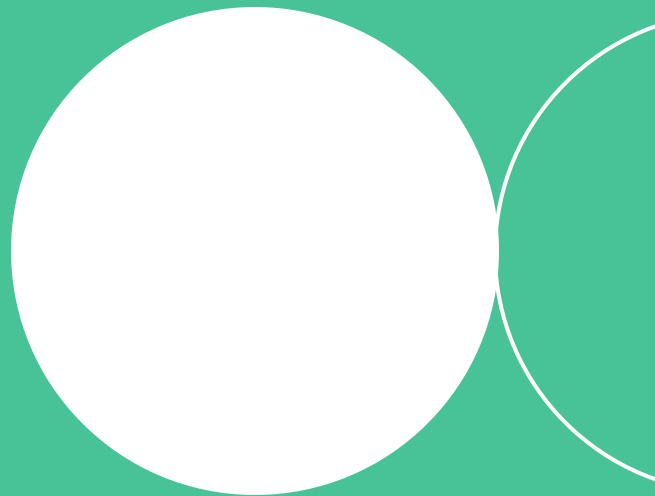
6. Отправляем подтверждение ACK
7. SeqNum = 2387450306 (1-е случайное число +1)
8. AckNum = 883876466 (2-е случайное число +1)

Итоги

- 1 Трехстороннее рукопожатие помогает установить надежное соединение, используя ненадежные сети передачи данных
- 2 Три этапа помогают обеим сторонам убедиться в функционировании канала связи, а также измерить его пропускную способность для установления размера sliding window
- 3 Сокет инициирующей стороны меняет свое состояние SYN-SENT - ESTABLISHED
- 4 Сокет принимающей стороны меняет свое состояние LISTEN - SYN-RECEIVED - ESTABLISHED

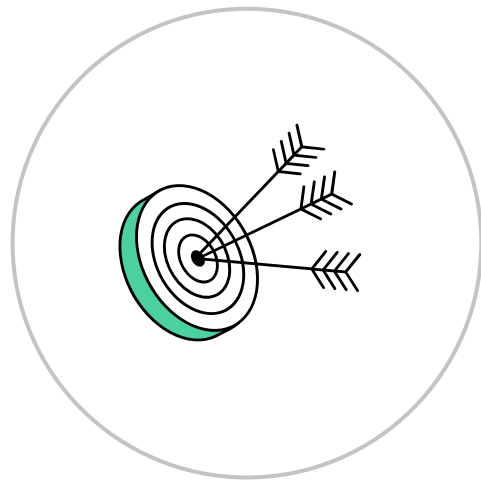



ТСР: завершение соединения



Цели темы

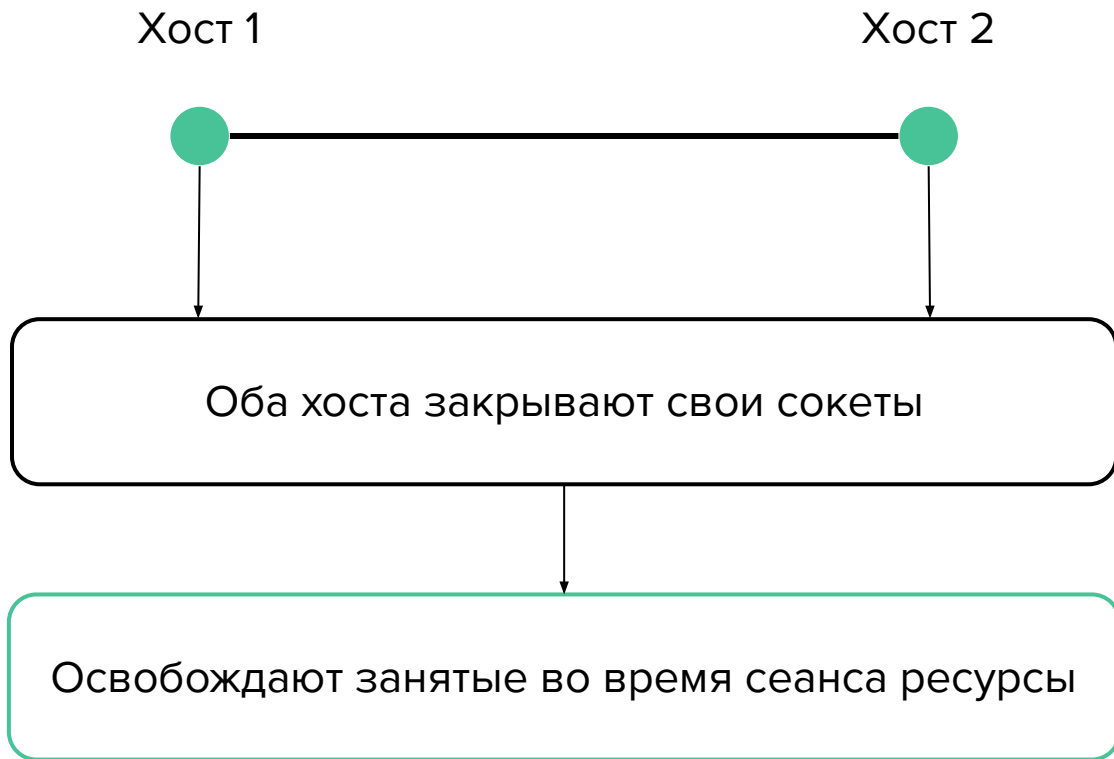
- Понять процесс завершения соединения
- Узнать, как меняется состояние сокетов во время этого процесса



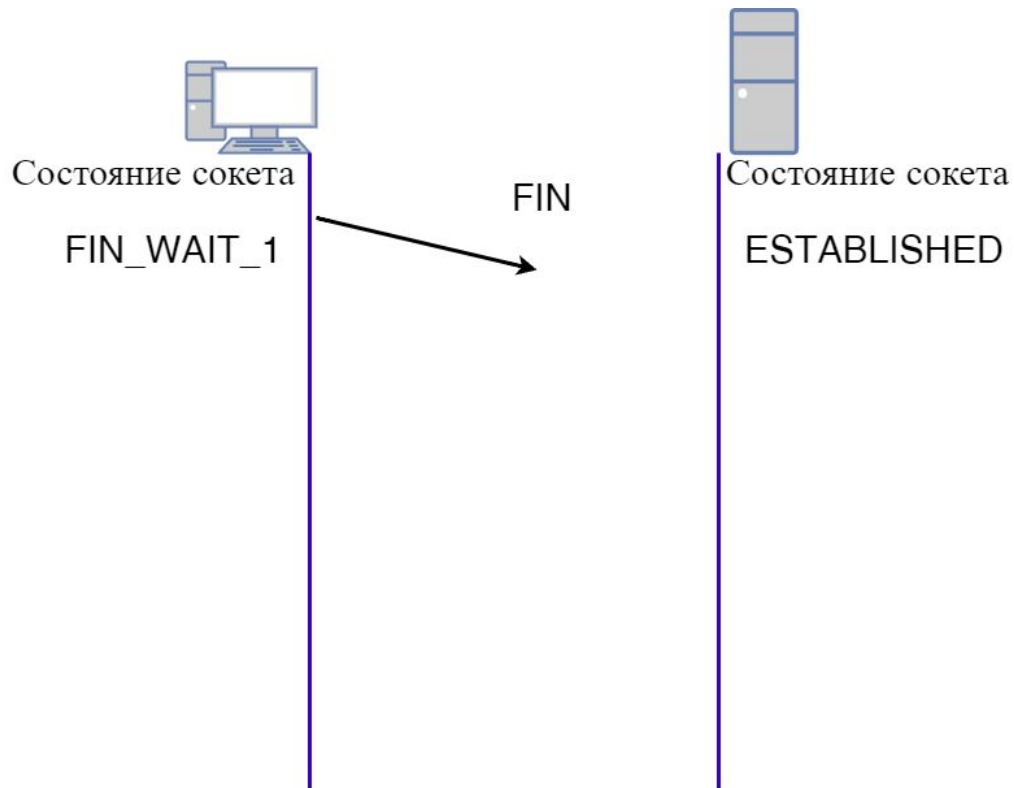


**Нормальное завершение TCP-
соединения происходит
в виде двухстороннего
рукопожатия**

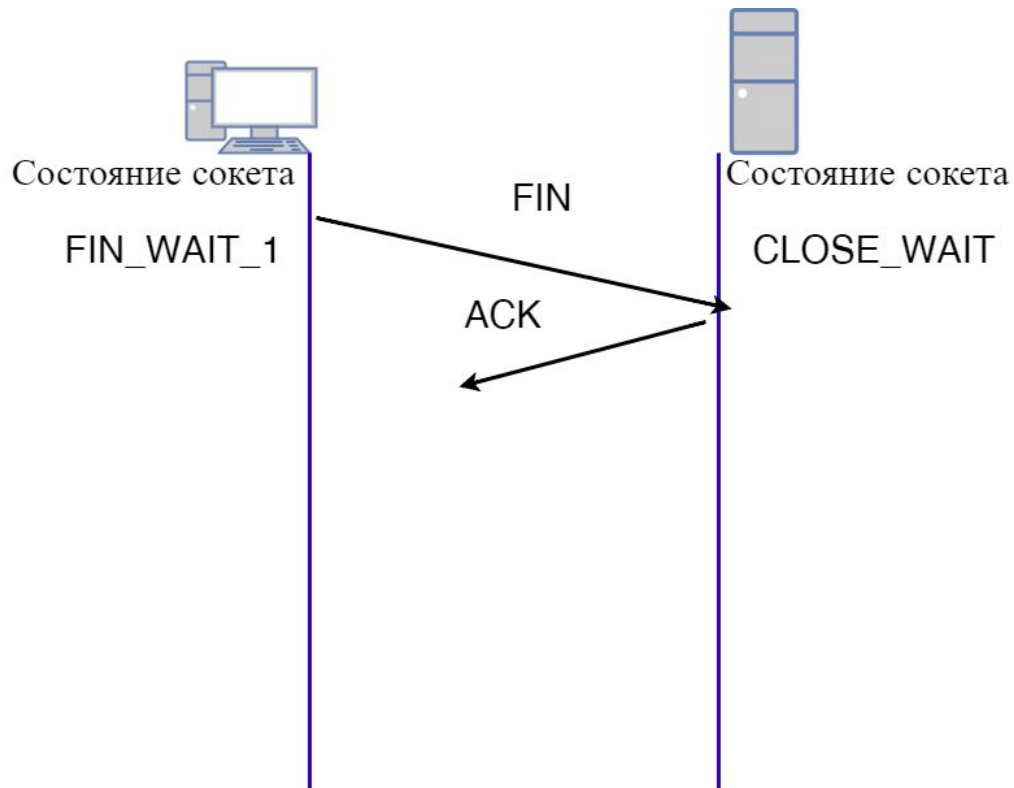
Завершение соединения между двумя хостами



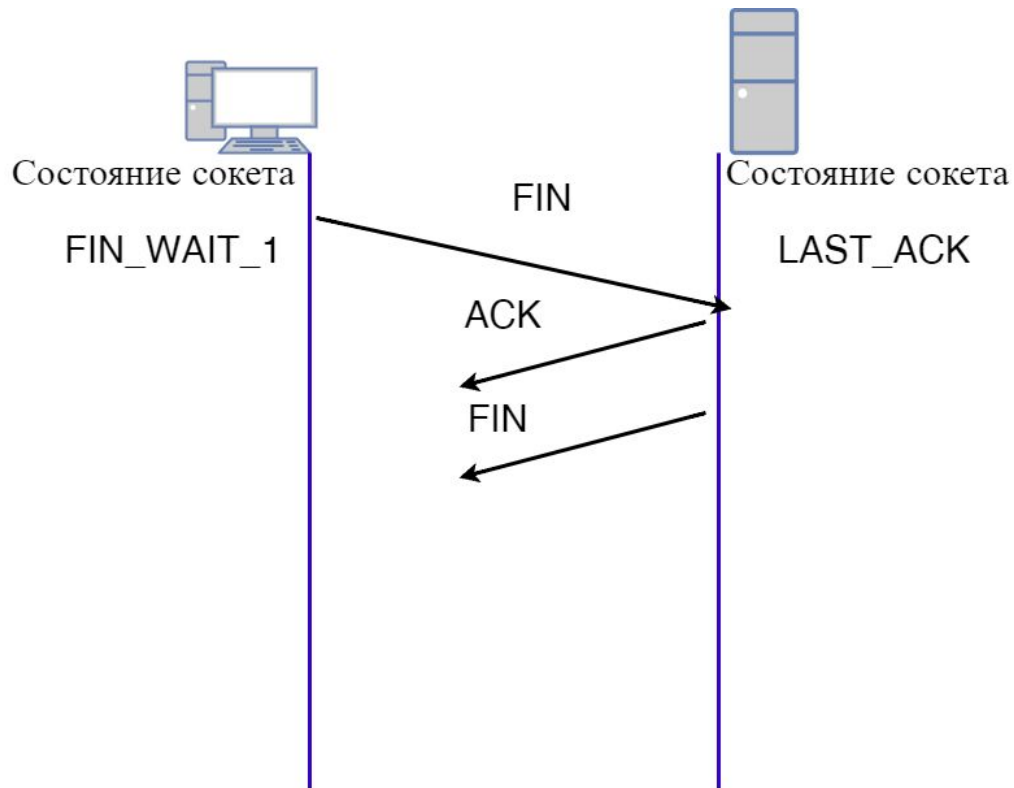
Завершение соединения



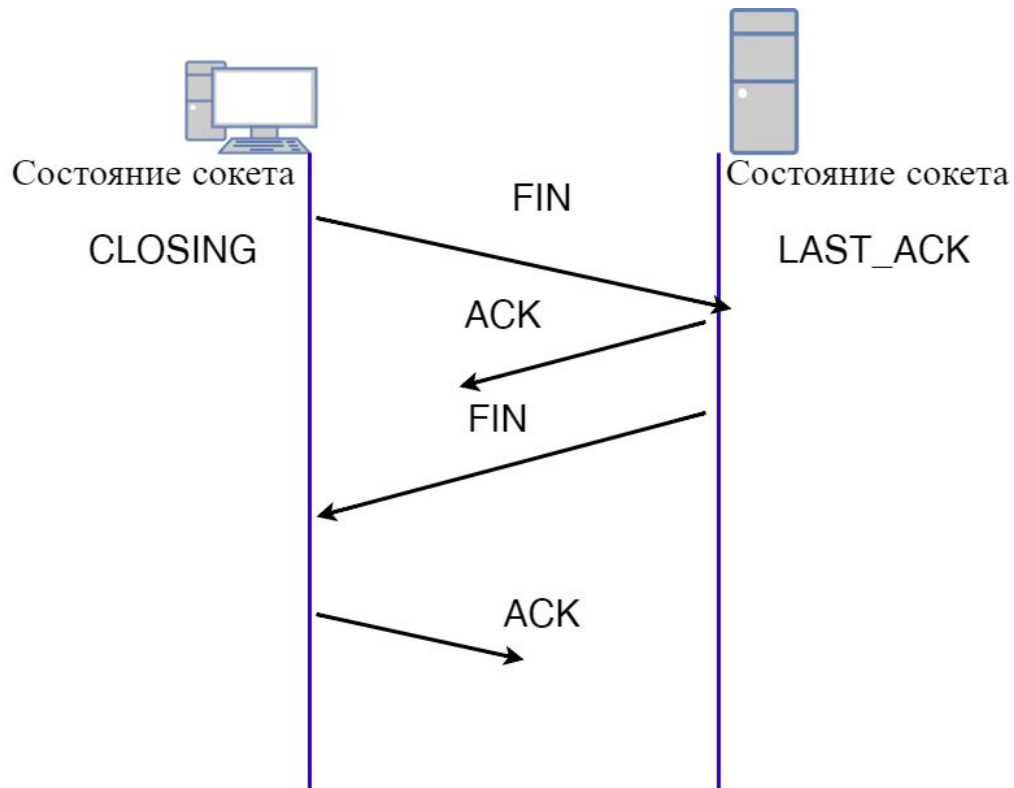
Завершение соединения



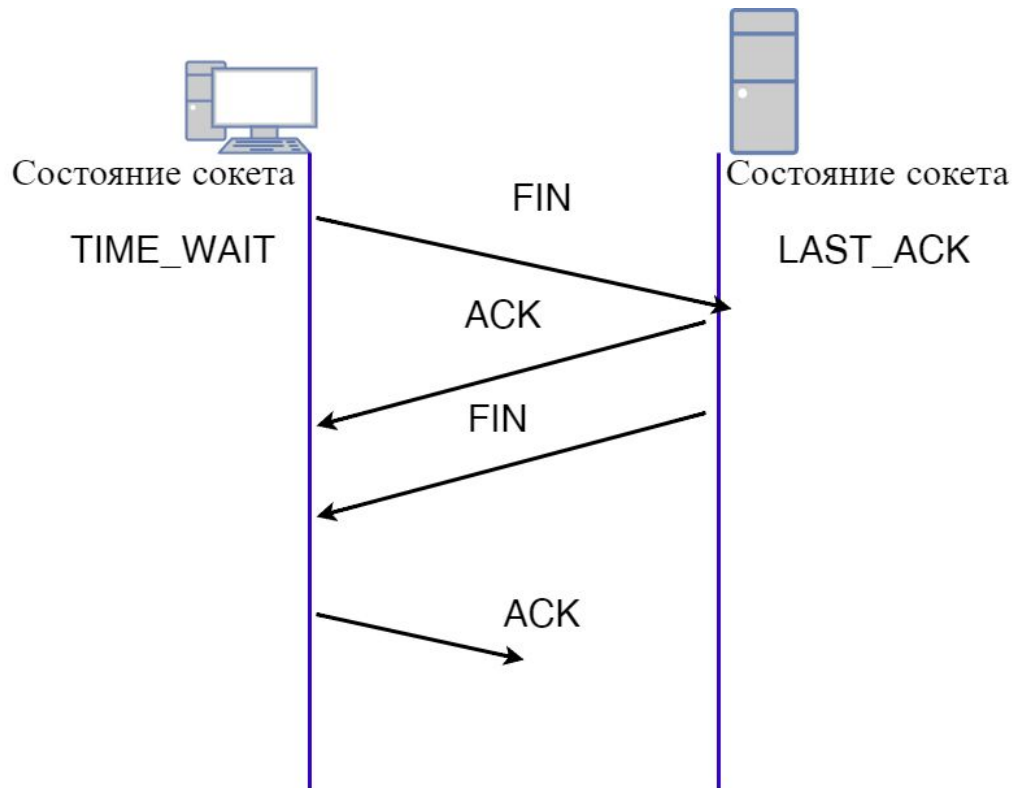
Завершение соединения



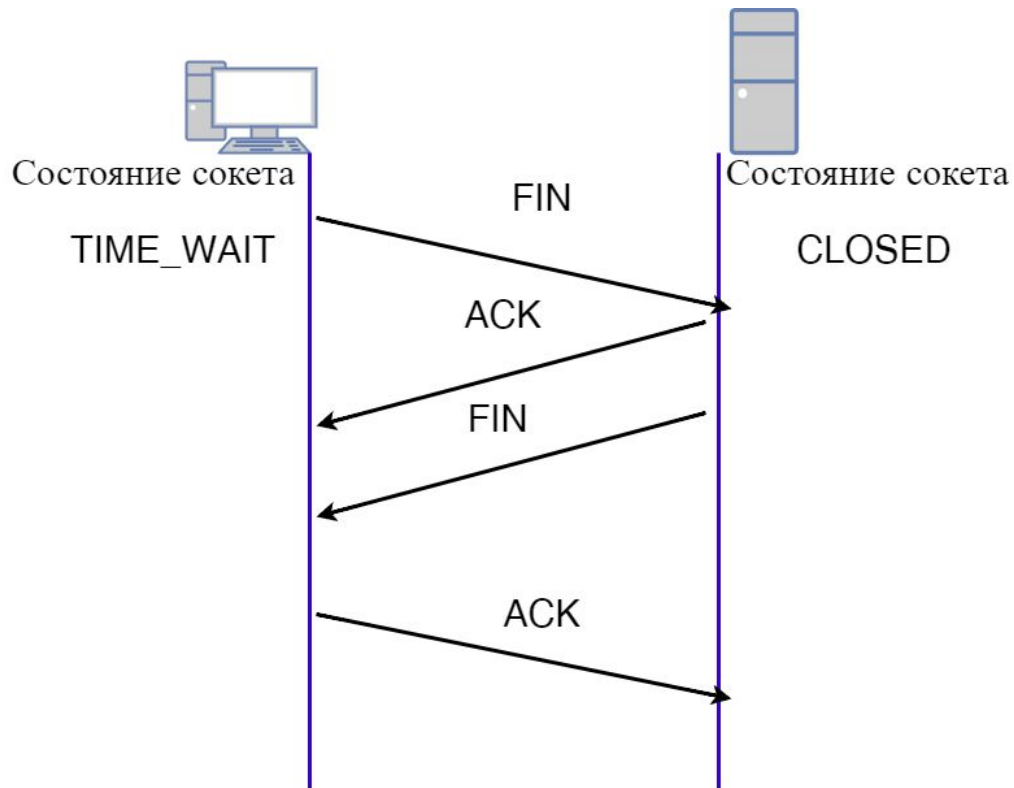
Завершение соединения



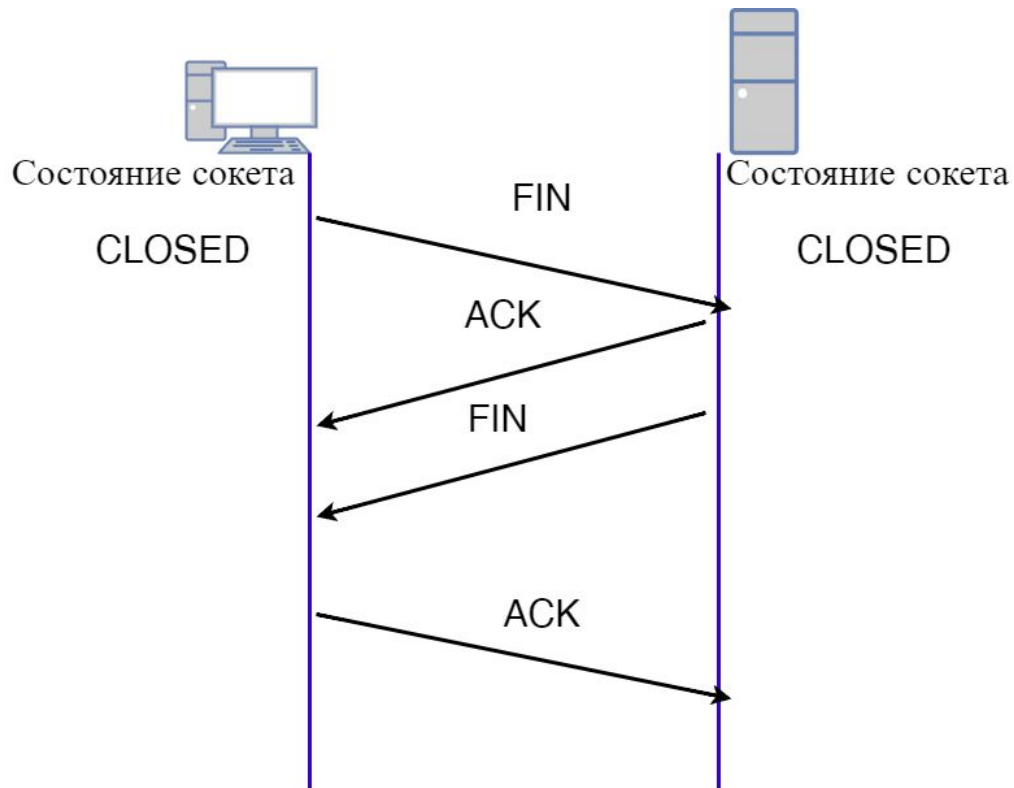
Завершение соединения



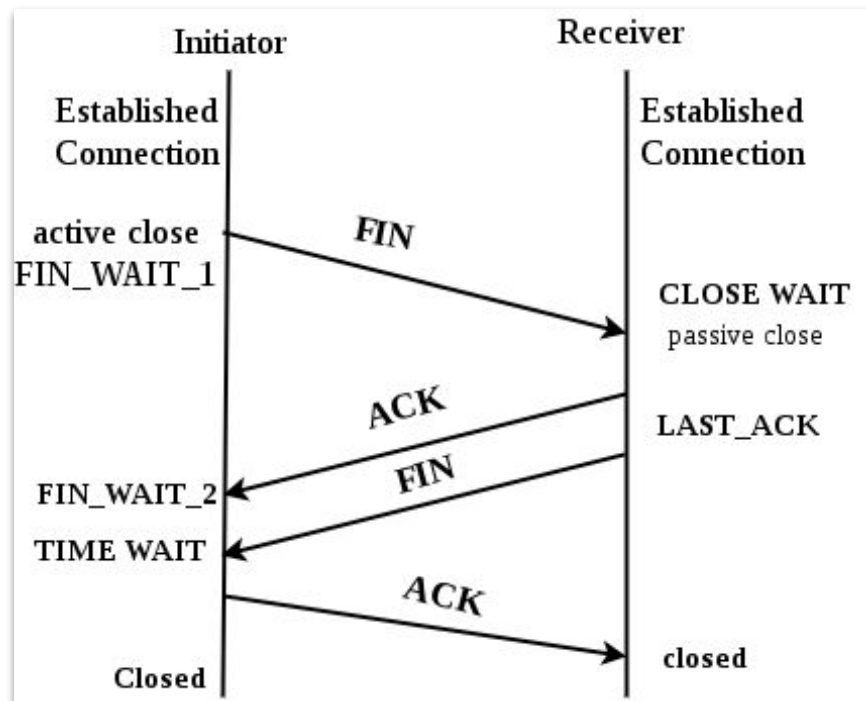
Завершение соединения



Завершение соединения



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**



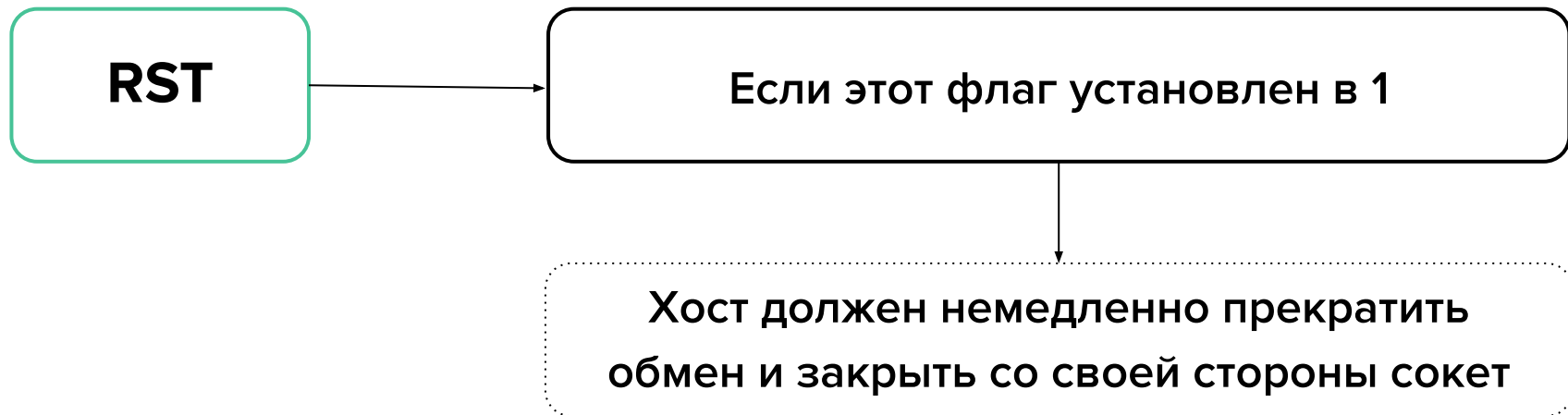
RST

флаг прерывания соединения, используется для отказа в соединении

флаг RST – сброс



Сбросы TCP



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

Итоги

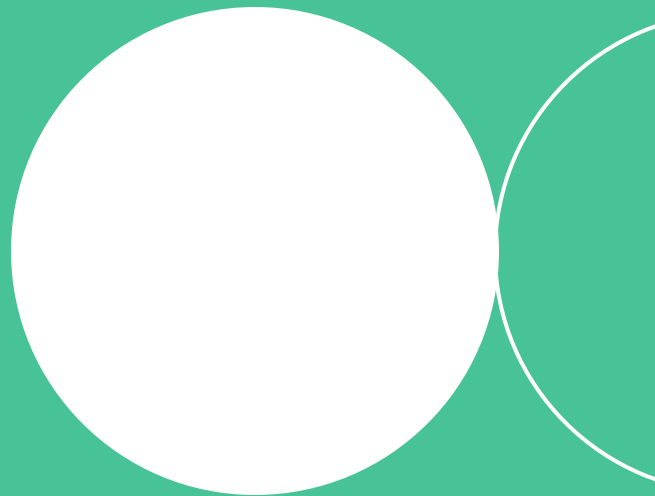
- 1 В протоколе TCP завершение соединения происходит по схеме двухстороннего рукопожатия, чтобы не потерять передаваемые данные
- 2 Флаг RST применяет в крайних случаях, когда одна из сторон обнаружила серьезную ошибку или сбой соединения
- 3 С помощью Wireshark увидели, как в реальности выглядит завершение соединения при нормальном сценарии и при возникновении сбоя



Перерыв

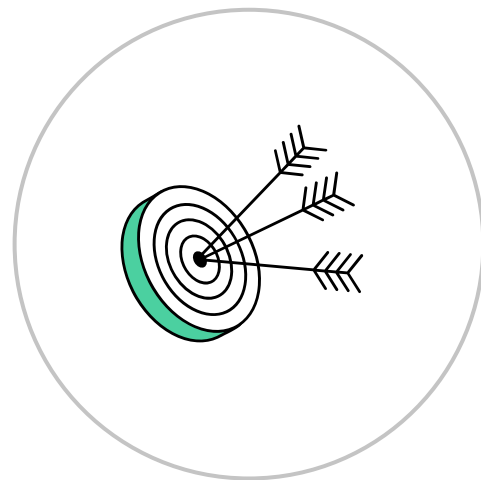


Обзор протокола UDP



Цели темы

- Рассмотреть протокол UDP
- Познакомиться с заголовком UDP
- Узнать о ключевых отличиях протокола UDP от TCP



Спецификация протокола UDP



RFC 768

Протокол UDP

1

«Рабочая лошадка»

всех потоковых
сервисов и геймдева

2

Обеспечивает
работу
современных
цифровых сервисов

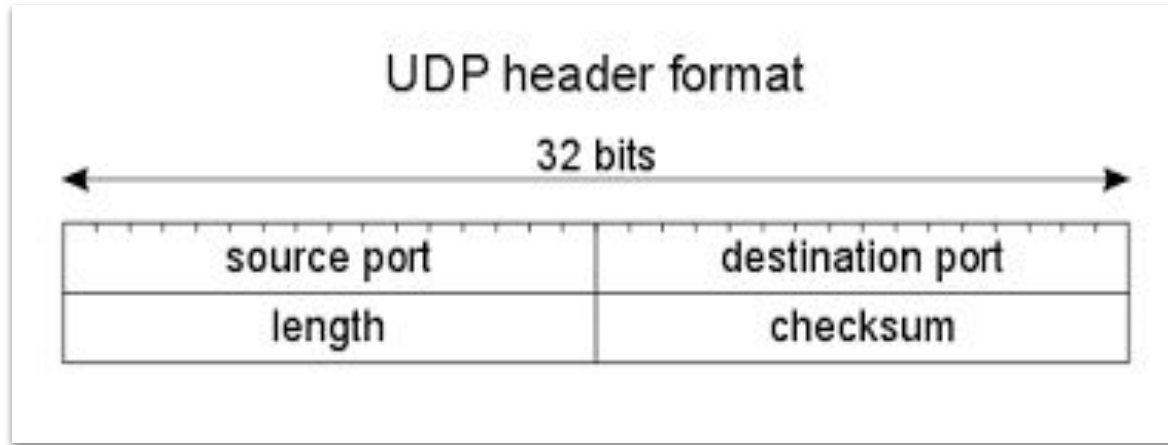
3

Основа 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



Заголовок протокола UDP

UDP header format



В протоколе нет:

- флагов
- средств синхронизации
- средств контроля сессии



Multicast

режим передачи данных, в котором данные передаются группе хостов



**В сетях IPv4 для многоадресного
вещания зарезервирована
подсеть 224.0.0.0/4**

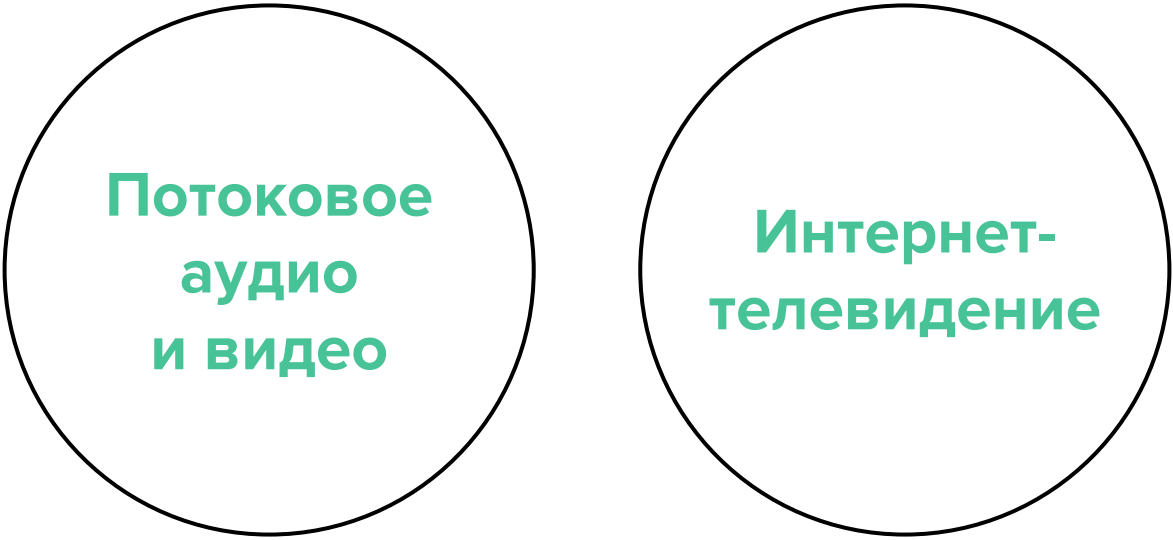
Преимущество многоадресного вещания

Часть информации для разных клиентов
передается один раз по «общему» маршруту



Снижается нагрузка на пропускную
способность сети

Примеры сервисов



Потоковое
аудио
и видео

Интернет-
телевидение

Дословный перевод DNS

Domain Name System



```
graph LR; A[Domain Name System] --> B[система доменных имен];
```

The diagram consists of two rounded rectangular boxes connected by a horizontal arrow pointing from left to right. The left box has a solid black border and contains the text 'Domain Name System' in a teal color. The right box has a dotted black border and contains the Russian text 'система доменных имен' in black. The arrow is a simple black line with a triangular head.

**система
доменных имен**



DNS

система, предназначенная для получения информации о доменах



Режим работы DNS

Запрос – ответ

**обмениваясь короткими
сообщениями**

Режим работы DNS

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)

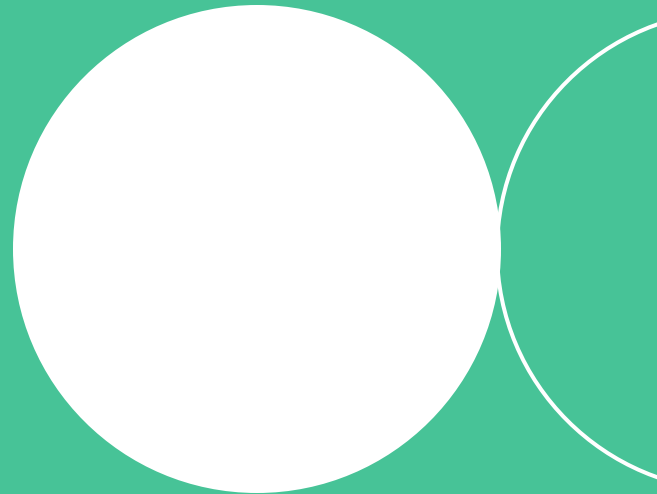
[\[Response In: 6\]](#)

Итоги

- 1 Протокол UDP крайне простой, ориентирован на транзакции, а не на соединение
- 2 UDP не гарантирует надёжность доставки, однако позволяет использовать multicast соединения для доставки одной информации большому количеству клиентов (стриминг, IP-телефония, IPTV)
- 3 UDP подходит как основа для реализации собственных протоколов

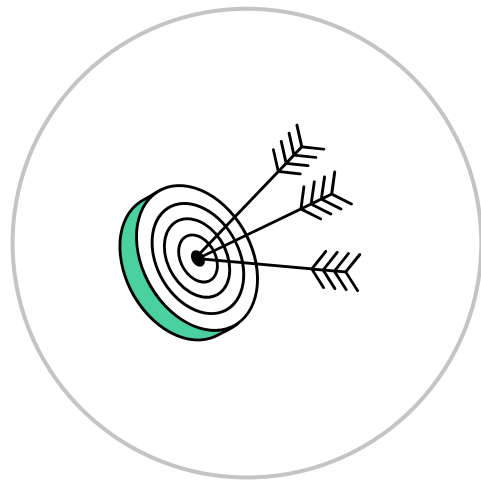


Сравнение протоколов TCP и UDP



Цели темы

- Сравнить основные протоколы TCP и UDP
- Понять преимущества TCP, в каких случаях они необходимы
- Понять преимущества UDP, каким образом их используют



Сравнение протокола TCP и UDP по надежности

TCP

Управляет потоком данных
через подтверждения,
повторные передачи,
изменение скорости

UDP

Ничего этого нет

Сравнение протокола TCP и UDP по упорядоченности

TCP

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»

**Если необходимо передать
информацию точно и критичен
каждый байт, то используем TCP**

Сравнение протокола TCP и UDP по ресурсоемкости

TCP

Требуется много накладных расходов (по времени и по нагрузке на сеть). Особенно, при неустойчивой связи

UDP

Все очень просто, поэтому он работает быстрее и меньше нагружает сеть

**Если связь неустойчивая, а
нужно отправить звук в
реальном времени, то можно
возложить коррекцию ошибок
на вышестоящие протоколы
Можно использовать UDP**

Сравнение протокола UDP и TCP в широковещательном режиме

UDP

Не требует установки соединения. Данные, которые посылаются всем узлам сети, будут обработаны одинаково

TCP

Невозможно, как из-за требования к установлению соединения, так и к высокому расходу ресурсов для контроля каждой сессии

Сравнение протокола UDP и TCP в многоадресном режиме

UDP

Каждая датаграмма может быть передана группе подписчиков без дополнительных накладных расходов

TCP

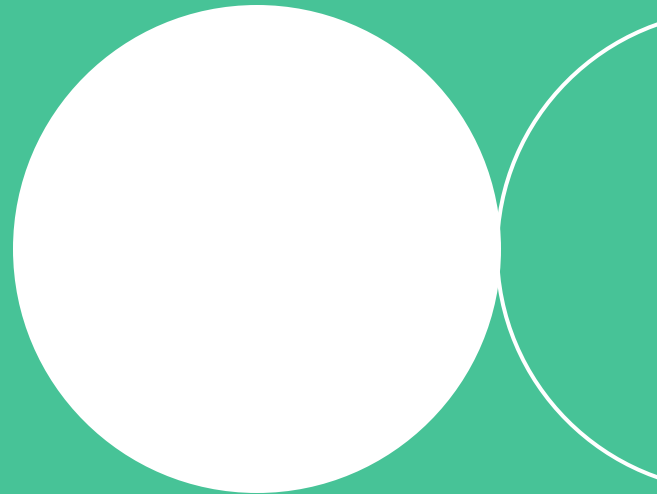
Невозможно, как из-за требования к установлению соединения, так и к высокому расходу ресурсов для контроля каждой сессии

Итоги

- 1 Если необходимо точно передать данные, без малейших искажений, мы используем протокол TCP
- 2 Если нам важна скорость, возможности multicast, а контроль над ошибками возложен на вышестоящие уровни, то мы используем протокол UDP

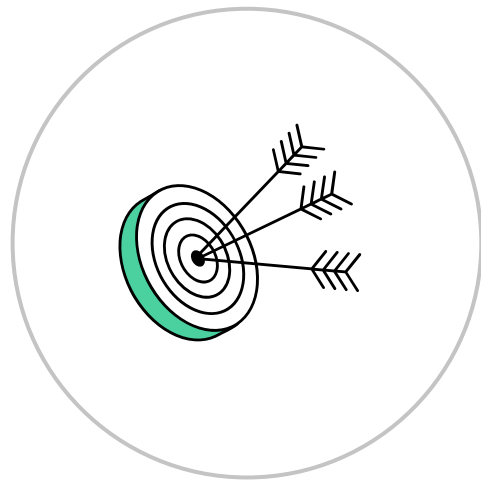


Популярные сетевые утилиты



Цели темы

- Познакомиться с популярными утилитами транспортного уровня: telnet, nmap, ss и lsof
- Научиться проводить диагностику проблем на транспортном уровне с использованием данных утилит





Telnet

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



Telnet

```
> telnet ya.ru 80
```

[чёрный экран говорит о том, что соединение установлено]

```
[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

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



Nmap

`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



Socket statistics

```
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:*
```

TCP соединения, установленные 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

мощная утилита, в том числе для получения информации по сети



Lsof

Lsof поможет 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

- 1 В качестве быстрой проверки доступности удаленного порта можно использовать утилиту **telnet**, для полного сканирования - **nmap**
- 2 Комбинируя утилиты **ss** и **lsof**, можно не только выяснить, какие порты используются на локальном компьютере, но и к каким процессам они относятся



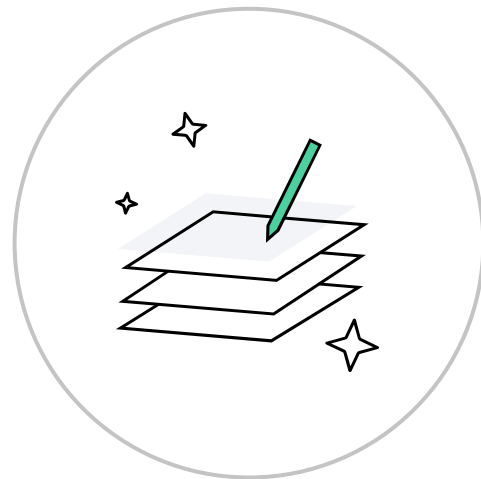
Практика



Домашнее задание

Давайте посмотрим вашу практику после лекции

- 1 Практика состоит из обязательного теста и домашнего задания со звездочкой (необязательное)
- 2 В тесте 14 вопросов, на 10 нужно ответить верно. Есть 2 попытки
- 3 Вопросы по домашнему заданию со звездочкой задавайте в чате группы
- 4 Задачи можно сдавать по частям.
Зачёт по домашней работе ставят после того, как приняты все задачи



Задавайте вопросы. Оставляйте обратную связь по вебинару

Ильмир Сахипов

Руководитель центра управления сетью АО “Уфанет”

