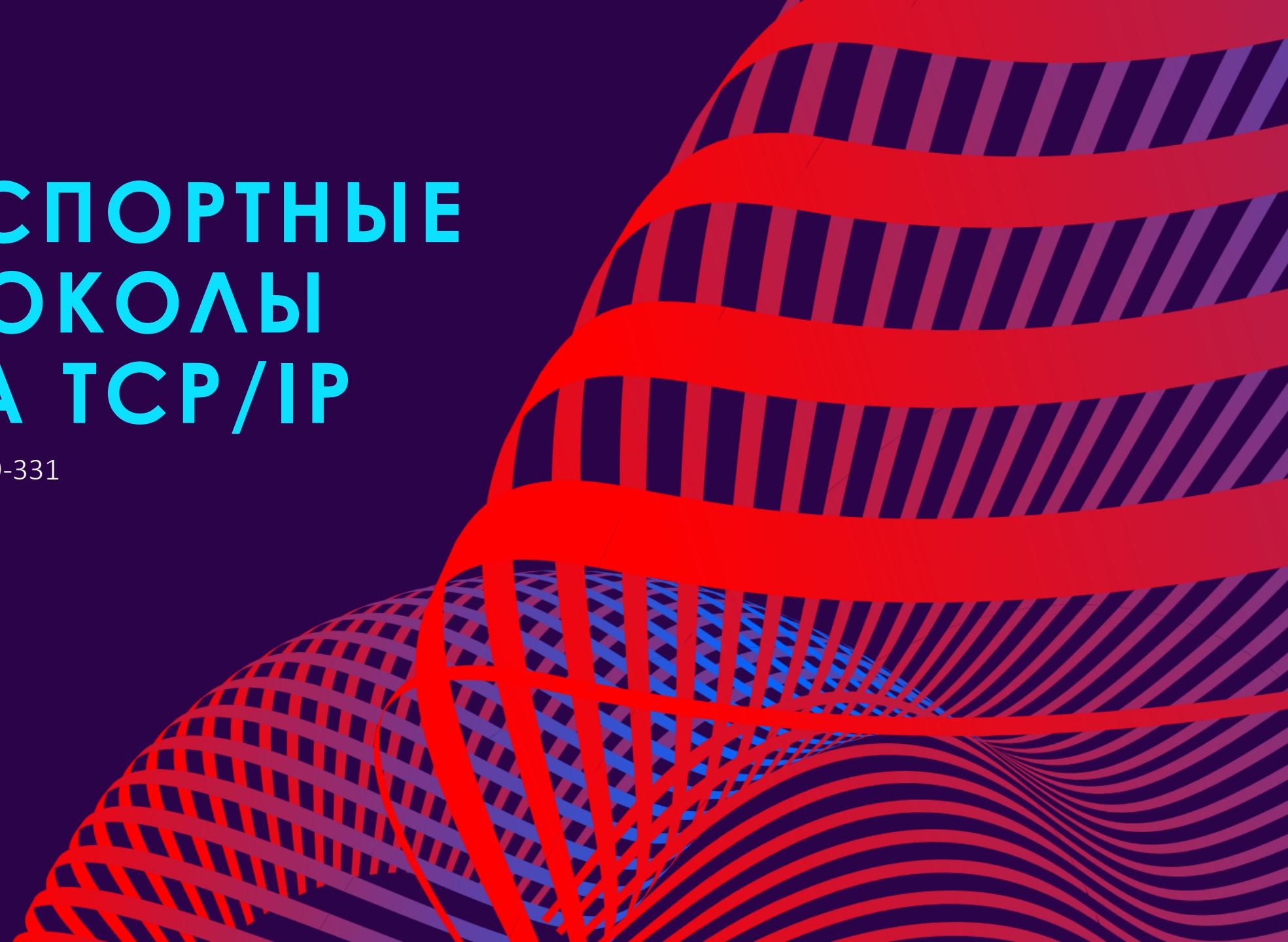
# ТРАНСПОРТНЫЕ ПРОТОКОЛЫ СТЕКА TCP/IP

Николаева Анна, 09-331



# РОЛЬ ТРАНСПОРТНОГО УРОВНЯ В МОДЕЛИ TCP/IP

## Место в иерархии:

- Прикладной уровень: HTTP, FTP, DNS, SMTP
- **Транспортный уровень: TCP, UDP**
- Сетевой уровень: IP, ICMP
- Канальный уровень: Ethernet, Wi-Fi

## Критические функции:

- Мультиплекксирование/демультиплекксирование потоков данных
- Сквозной контроль ошибок
- Управление потоком между процессами
- Обеспечение качества обслуживания (QoS)

# КРИТИЧЕСКИЕ ФУНКЦИИ — ЭТО ОСНОВНЫЕ, ОБЯЗАТЕЛЬНЫЕ ЗАДАЧИ, КОТОРЫЕ ТРАНСПОРТНЫЙ УРОВЕНЬ РЕШАЕТ ДЛЯ ОБЕСПЕЧЕНИЯ КОРРЕКТНОЙ СВЯЗИ МЕЖДУ ПРИКЛАДНЫМИ ПРОЦЕССАМИ В СЕТИ.



Критические функции:

Мультиплекксирование/демультиплекксирование потоков данных. Это разделение данных от разных приложений на одном хосте с помощью портов. Как работает: С помощью номеров портов транспортный уровень определяет, какому приложению принадлежат данные. *Например:* Веб-браузер (порт 80) и почтовый клиент (порт 25) работают одновременно через один сетевой интерфейс.

# Критические функции:

**Сквозной контроль ошибок**. Это гарантия целостности данных между конечными приложениями, а не только между соседними узлами. *Механизмы:* Контрольные суммы, подтверждения (ACK) и повторные передачи (в TCP). **Как работает:** Использование **контрольных сумм** для обнаружения ошибок и механизмов **подтверждения и повторной передачи** для их исправления. **Пример:** TCP отправляет подтверждение (ACK) для каждого полученного сегмента и повторяет передачу при обнаружении потери.

Критические функции:

**Управление потоком между процессами**. Это регулировка скорости передачи данных, чтобы отправитель не "перегружал" получателя. **Как работает:** Получатель сообщает отправителю, сколько данных он готов принять (размер окна).

**Пример:** В TCP используется механизм **скользящего окна**, где окно динамически меняется в зависимости от возможностей получателя, сколько данных он готов принять.

Критические функции:

**Обеспечение качества обслуживания**. Это приоритизация трафика и управление задержками для чувствительных к времени приложений.

**Как работает:** Назначение приоритетов различным типам трафика (например, голосовой связи или потоковому видео). VoIP (или же голос) может получить более высокий приоритет, чем загрузка файла.

## Почему эти функции КРИТИЧЕСКИЕ?

Без этих функций сетевое взаимодействие было бы невозможным или крайне неэффективным:

**Без мультиплексирования:** Невозможно было бы запускать несколько сетевых приложений одновременно.

**Без контроля ошибок:** Данные приходили бы с ошибками, а потерянные пакеты не восстанавливались.

**Без управления потоком:** Быстрые отправители "заваливали" бы медленных получателей.

**Без обеспечения качества обслуживания:** Чувствительные к задержкам приложения (видеозвонки, игры) работали бы с перерывами.

# КОНЦЕПЦИЯ СКВОЗНОЙ ДОСТАВКИ (END-TO-END)



**Принципы:**

- Доставка "от процесса к процессу", а не "от хоста к хосту"
- Независимость от нижележащих сетевых технологий
- Единый интерфейс для прикладных программ

# АРХИТЕКТУРА TCP

Ключевые механизмы надежности

## КОНТРОЛЬНЫЕ СУММЫ:

- 16-битная контрольная сумма покрывает заголовок и данные
- Обнаружение ошибок передачи на уровне канала
- При ошибке - сегмент отбрасывается без подтверждения

## ТАЙМЕРЫ И ПОВТОРНЫЕ ПЕРЕДАЧИ:

- RTT (Round-Trip Time) - время обхода туда и обратно
- Алгоритм вычисления тайм-аута на основе скользящего среднего RTT
- Экспоненциальное увеличение тайм-аута при повторных передачах

## 1. Контрольные суммы (Checksum)

**Что это:** 16-битный механизм проверки целостности данных в сегменте TCP.

**Как работает:**

**Вычисление:** Суммирование всех 16-битных слов заголовка и данных с последующим дополнением до единицы (one's complement)



**Покрытие:** Включает не только TCP-заголовок и данные, но и псевдозаголовок (IP-адреса отправителя и получателя)

**Цель:** Обнаружение ошибок, вызванных сбоями в оборудовании, помехами в канале связи или проблемами маршрутизации

**Что происходит при ошибке:**

Получатель **отбрасывает поврежденный сегмент без отправки ACK**

Отправитель, не получив подтверждения, повторяет передачу по тайм-ауту

**Важно:** Контрольная сумма не исправляет ошибки, только обнаруживает их

## 2. Таймеры и повторные передачи

### RTT (Round-Trip Time):



**Определение:** Время между отправкой сегмента и получением подтверждения (ACK)

## Пример в действии:

1. Отправлен сегмент №1, RTO (тайм-аут для повторной передачи) = 1 секунда



2. ACK не пришел за 1 секунду → повторная передача

3. Новый RTO = 2 секунды

4. Если снова нет ACK → RTO = 4 секунды

5. И так до максимума 60 секунд

# Дополнительные механизмы надежности



## Выборочные подтверждения (SACK - Selective Acknowledgments):

- Позволяет подтверждать получение отдельных блоков данных
- Особенно эффективно при потере нескольких сегментов подряд
- Передает "дыры" в полученных данных отправителю

# Дополнительные механизмы надежности



## Быстрая повторная передача (Fast Retransmit):

- При получении **трех дубликатов ACK** для одного сегмента
- Не ждет тайм-аута, сразу повторяет передачу
- Уменьшает задержки при потере отдельных сегментов

# Дополнительные механизмы надежности



## Быстрое восстановление (Fast Recovery):

- После быстрой повторной передачи не сбрасывает окно перегрузки
- Сохраняет данные в "конвейере" передачи
- Плавный переход к нормальному режиму работы

# ФОРМАТ СЕГМЕНТА TCP

| Формат TCP-сегмента | | | | |
|---|---|---|---|---|
| **Бит** | **0 - 3** | **4 - 7** | **8 - 15** | **16 - 31** |
| **0** | Порт источника | | | Порт назначения |
| **32** | Номер последовательности | | | |
| **64** | Номер подтверждения | | | |
| **96** | Смещение данных | Зарезервировано | Флаги | Окно |
| **128** | Контрольная сумма | | | Указатель важности |
| **160** | Опции (необязательное) | | | |
| **160/192+** | Данные | | | |

Описание полей:

**Строка 1 (Биты 0-31): Порта:**

- Порт источника (биты 0-15): Номер порта отправителя
- Порт назначения (биты 16-31): Номер порта получателя



**Строка 2 (Биты 32-63): Номер последовательности:**

- Уникальный номер первого байта в этом сегменте
- Помогает собрать данные в правильном порядке

**Строка 3 (Биты 64-95): Номер подтверждения:**

- Номер следующего ожидаемого байта
- Действителен только при установленном флаге ACK

Строка 4 (Биты 96-127): Служебная информация:

4 части:

1) Смещение данных (4 бита): Длина заголовка в 32-битных словах

- Минимум: 5 (5×4 = 20 байт)
- Максимум: 15 (15×4 = 60 байт)

2) Зарезервировано (6 бит): Всегда 0, запас на будущее

3) Флаги (6 бит): Управляющие биты:



- URG - срочные данные
- ACK - подтверждение действительно
- PSH - протолкнуть данные приложению
- RST - сбросить соединение
- SYN - установить соединение
- FIN - завершить соединение

4) Окно (16 бит): Размер окна для управления потоком

- "Сколько байт я готов принять"
- Динамически меняется

**Строка 5 (Биты 128-159): Контроль:**

Контрольная сумма (16 бит): Проверка целостности данных

Указатель важности (16 бит):

Смещение до срочных данных

Работает только при установленном флаге URG

**Строка 6 (Биты 160+): Опции (необязательные):**

Дополнительные параметры:

MSS (Maximum Segment Size) - максимальный размер сегмента

Window Scale - масштабирование окна для высокоскоростных сетей

SACK (Selective ACK) - выборочные подтверждения

Timestamp - метки времени

Выравнивание (Padding): Если опции не заполняют строку полностью, добавляются нули

**Флаги управления:**

**URG (Urgent - Срочный):**

Указывает на наличие срочных данных

Используется редко в современных приложениях

**ACK (Acknowledgment - Подтверждение):**

Поле номера подтверждения действительно

Устанавливается во всех сегментах, кроме начального SYN

**PSH (Push - Проталкивание):**

Требование немедленной доставки данных приложению

Отключает алгоритм Нейгла (Nagle's algorithm)

**Флаги управления:**

**RST (Reset - Сброс):**

Аварийный разрыв соединения

Используется при обнаружении неисправимых ошибок

**SYN (Synchronize - Синхронизация):**

Запрос на установление соединения

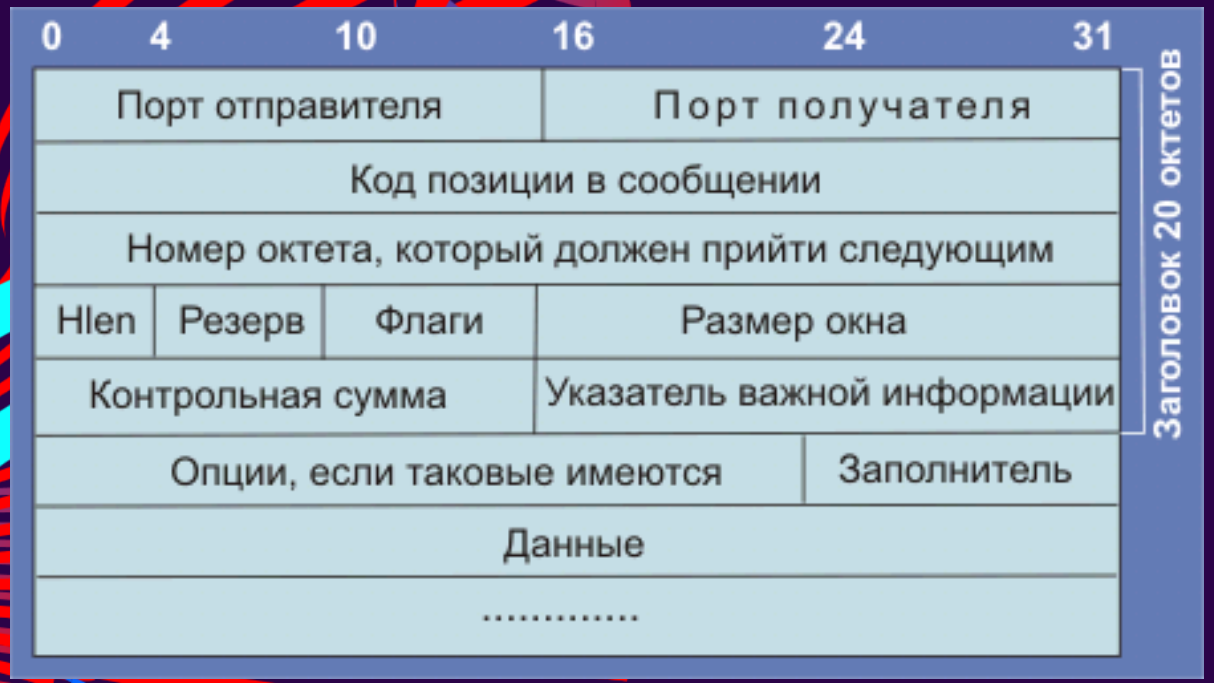
Синхронизирует начальные номера последовательности

**FIN (Finish - Завершение):**

Уведомление о завершении передачи данных

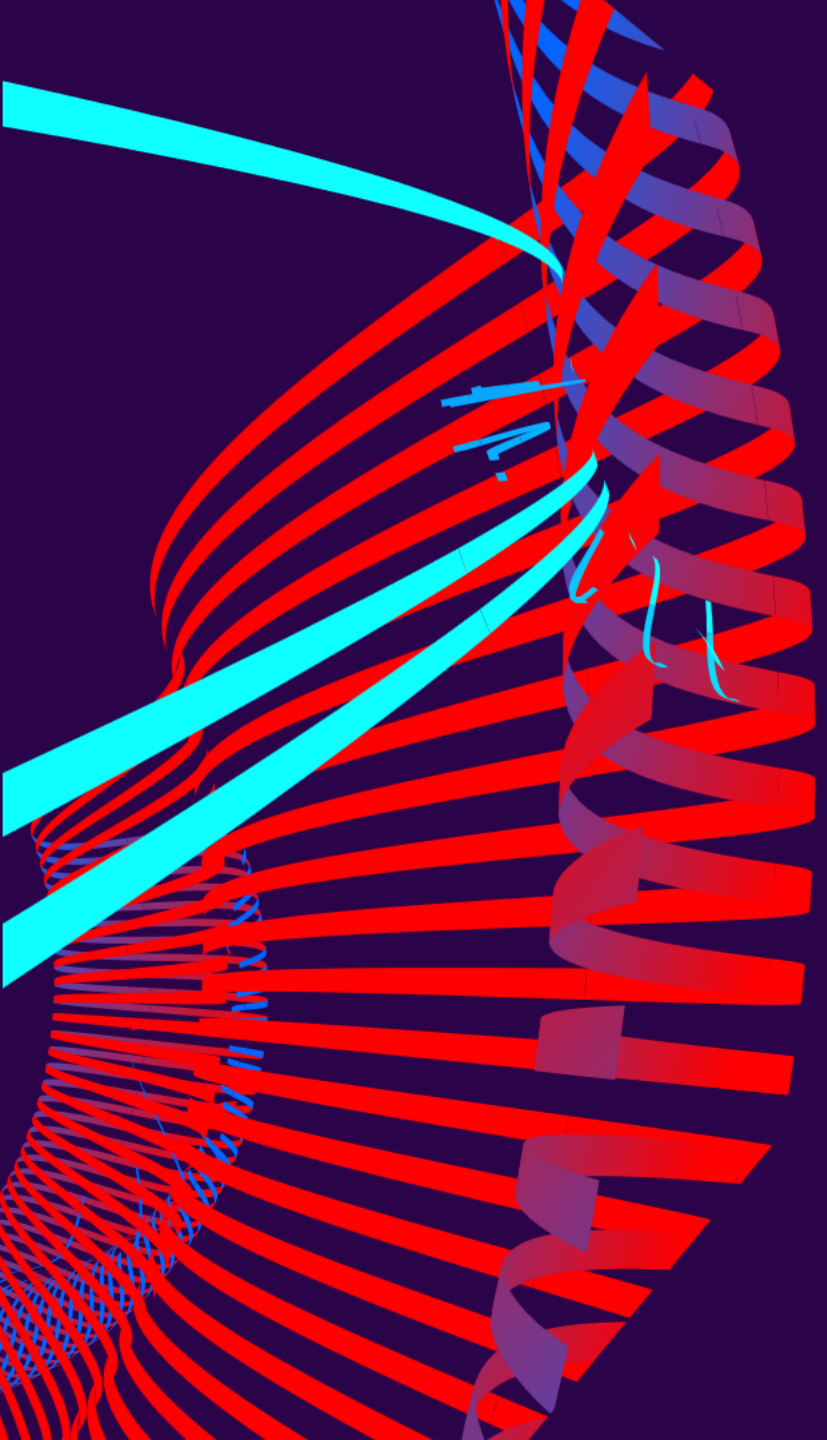
Корректное закрытие соединения

# ФОРМАТ СЕГМЕНТА TCP




Поле ***опции*** зарезервиров
ано на будущее и в
заголовке может
отсутствовать, его размер
переменен и дополняется
до кратного 32-бит с
помощью
поля *заполнитель*.

октет — это **группа из 8 бит**



- В поле ***вид*** записывается код опции, поле ***LEN*** содержит число октетов в описании опции, включая поля *вид* и *LEN*. Поле ***данные*** может иметь переменную длину, верхняя его граница задается значением **MSS** (Maximum Segment Size). Значение MSS может быть задано при установлении соединения каждой из сторон независимо.

# ФОРМАТ ОПЦИЙ

Формат опции MSS

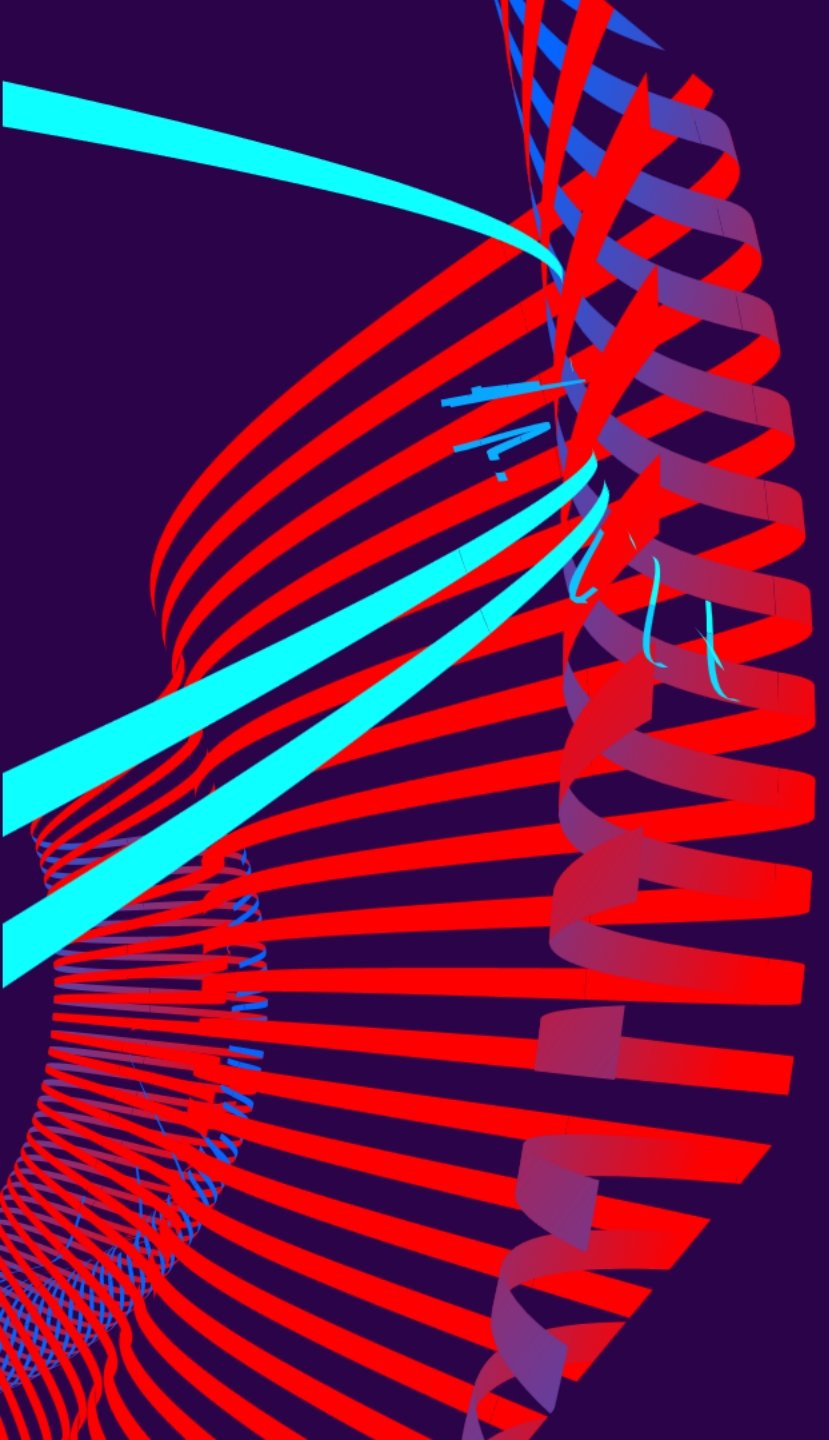
| Вид=2 | Len=4 | MSS |
|---|---|---|
| 1 октет | 1 октет | 2 октета |

Формат опции *масштаб окна*

| Вид=3 | Len=3 | Счетчик |
|---|---|---|
| 1 октет | 1 октет | 1 октет |

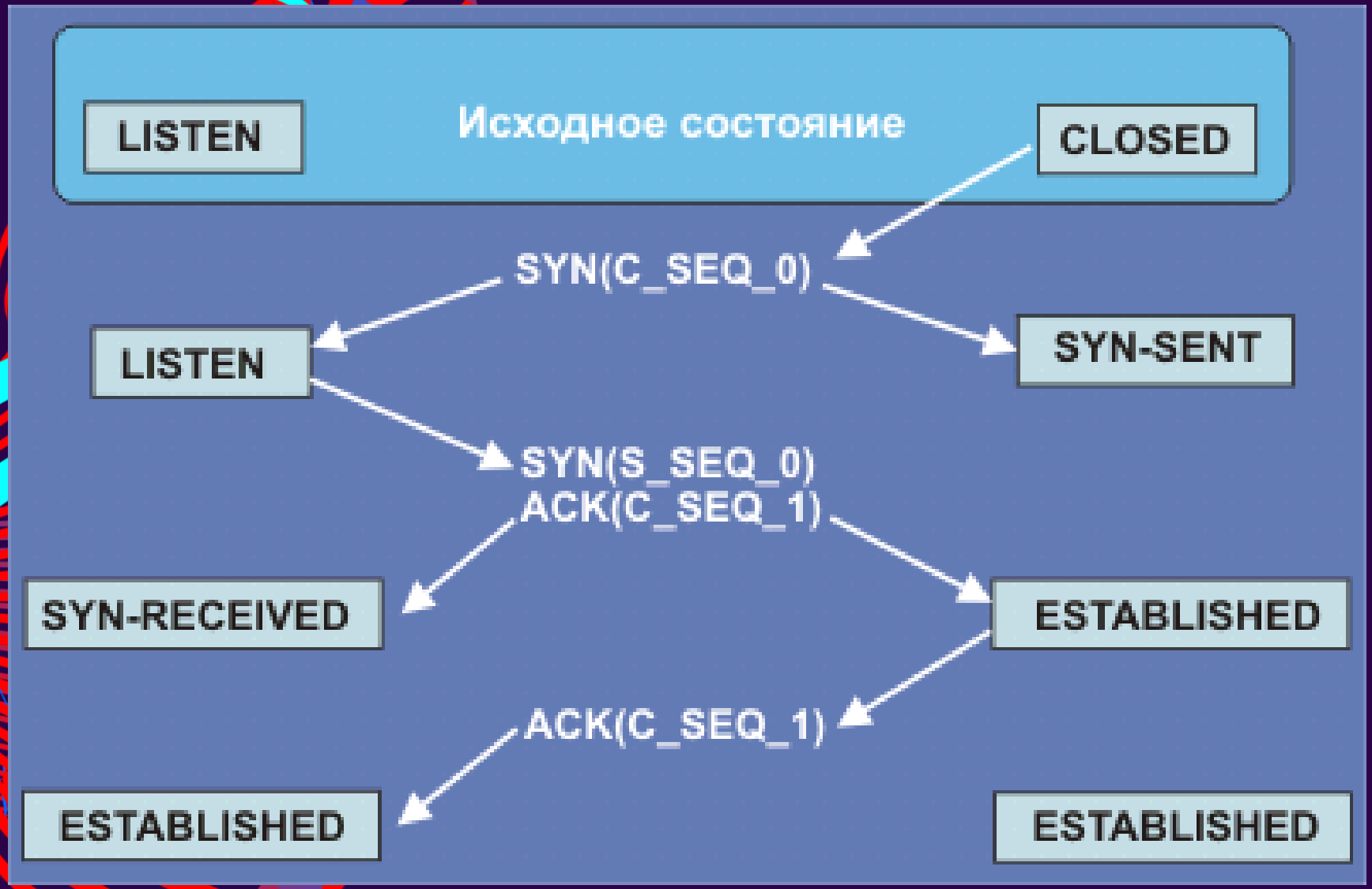
Формат опции временной метки

| Вид=8 | Len=10 | Значение временной метки | Отклик |
|---|---|---|---|
| 1 октет | 1 октет | 4 октета | 4 октета |

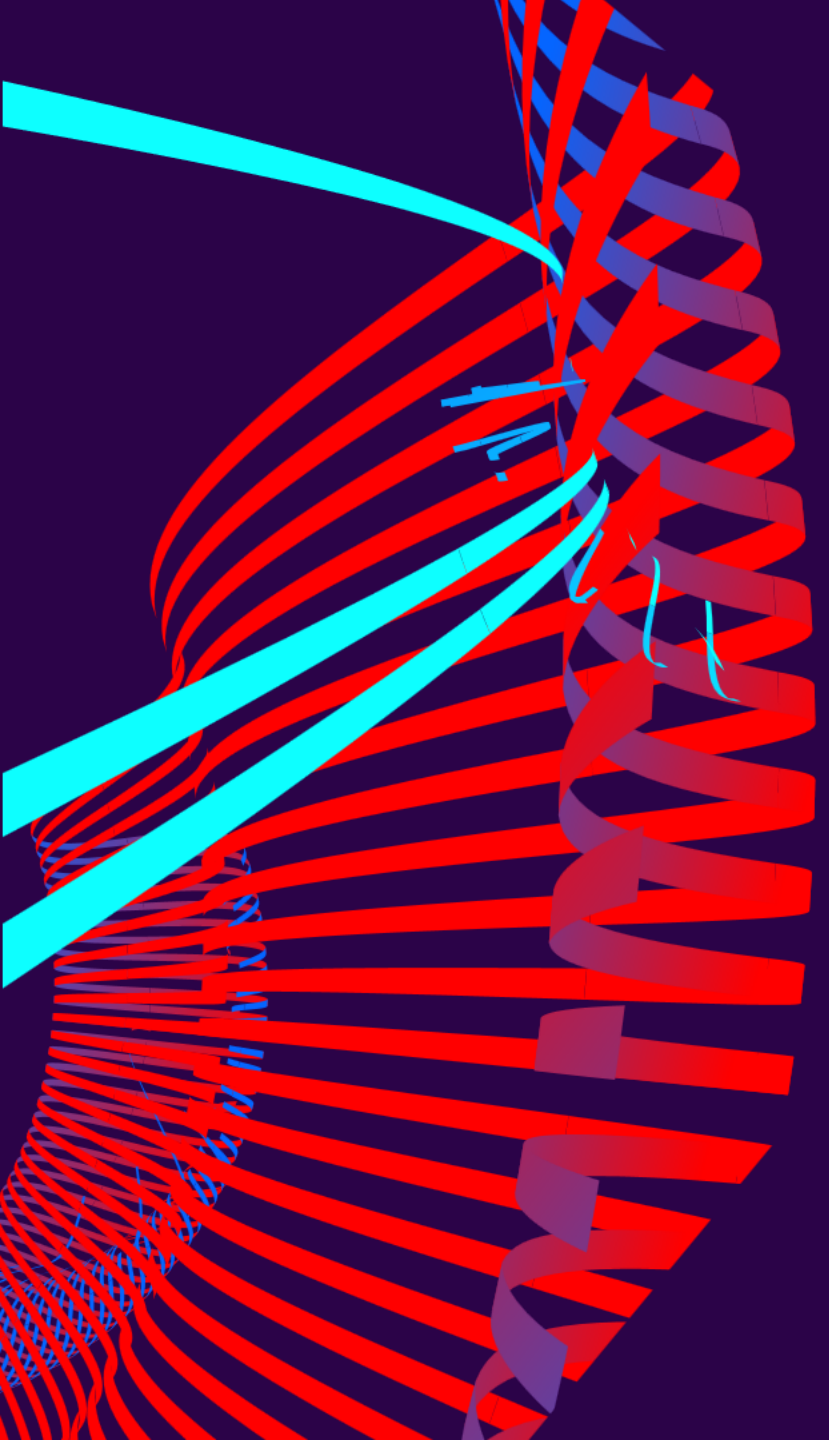


- Поле *данные* в TCP-сегменте может и отсутствовать, характер и формат передаваемой информации задается исключительно прикладной программой, теоретически максимальный размер этого поля составляет в отсутствии опций 65495 байт. TCP является протоколом, который ориентируется на согласованную работу ЭВМ и программного обеспечения партнеров, участвующих в обмене информацией.

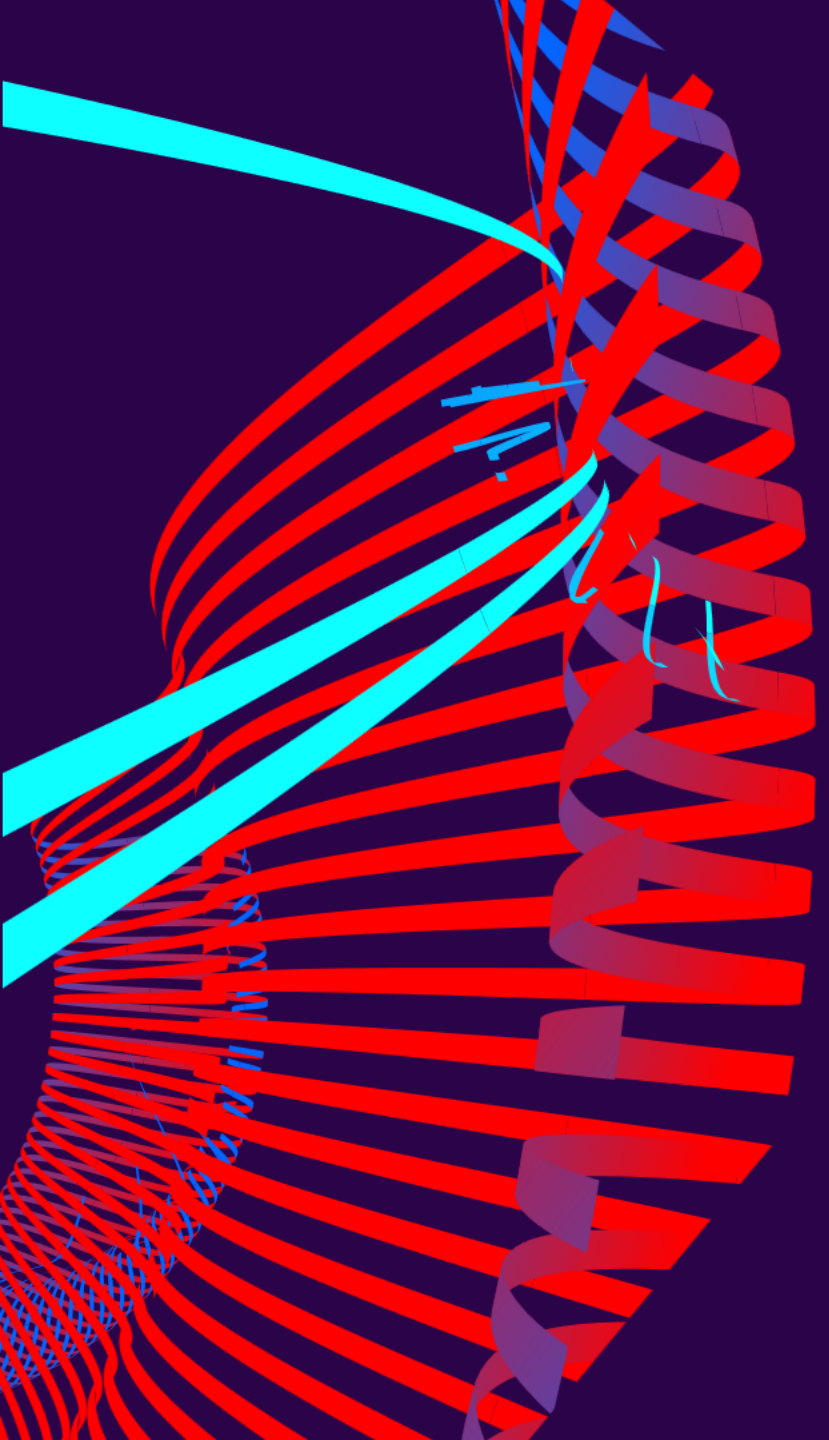
# АЛГОРИТМ УСТАНОВЛЕНИЯ СВЯЗИ



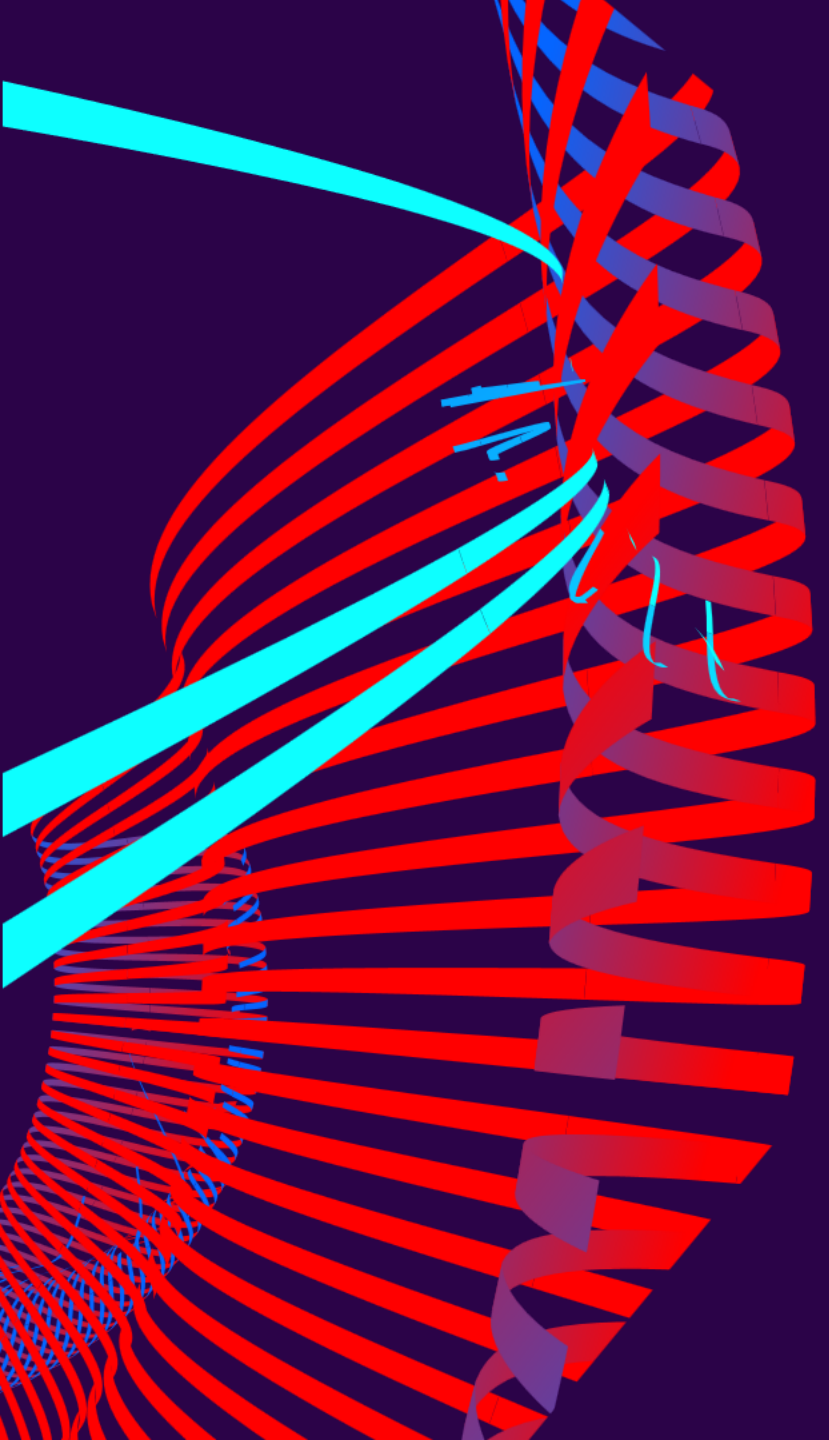

После установления соединения ISN(S) = s_seq_1, а ISN(C) = c_seq_1



- Установление связи клиент-сервер осуществляется в три этапа:
- Клиент посылает SYN-сегмент с указанием номера порта сервера, который предлагается использовать для организации канала связи (active open).
- Сервер откликается, посылая свой SYN-сегмент, содержащий идентификатор **ISN** (Initial Sequence Number). Начальное значение ISN не равно нулю. Процедура называется *passive open*.
- Клиент отправляет подтверждение получения SYN-сегмента от сервера с идентификатором равным ISN (сервера)+1.

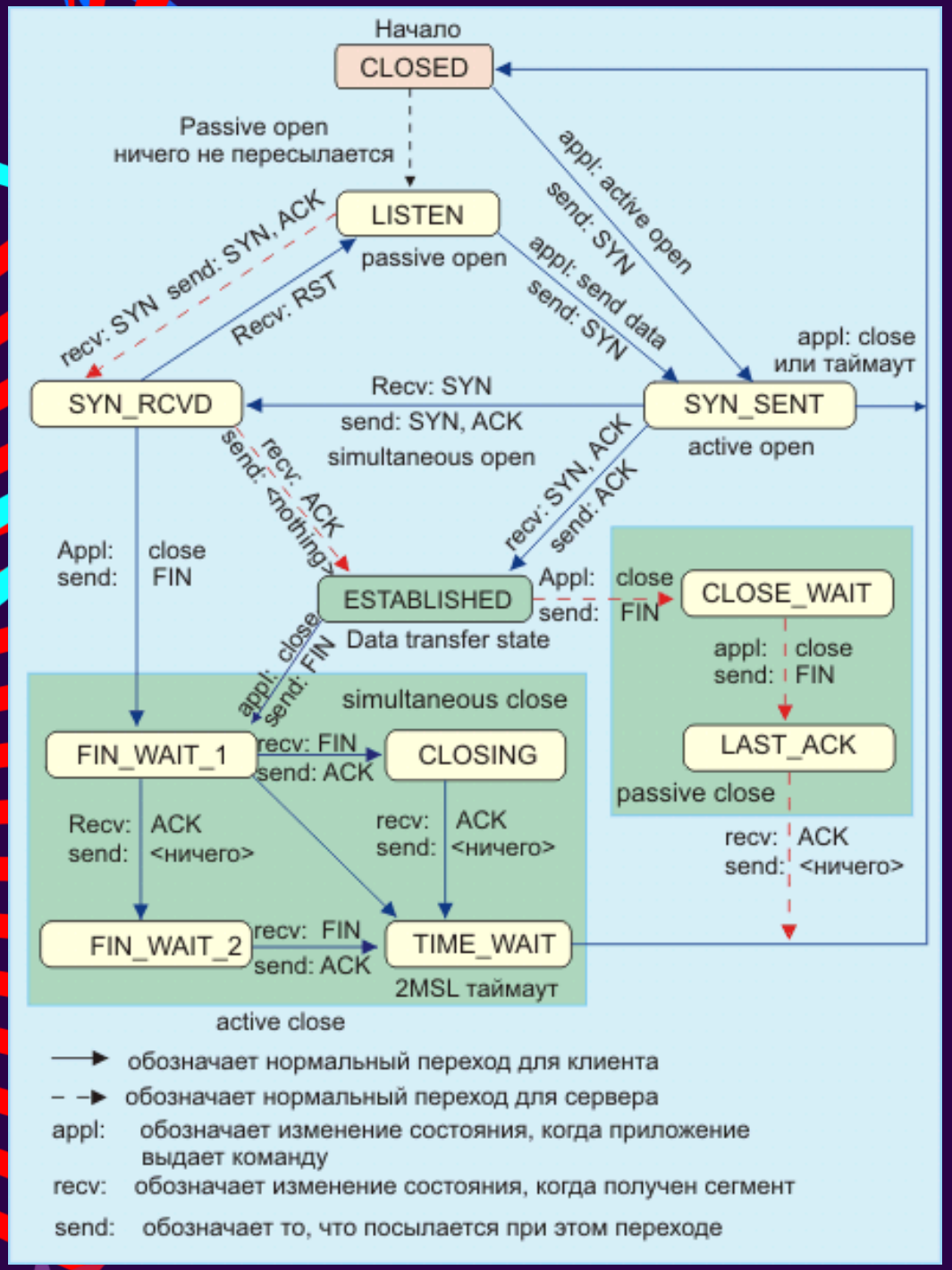


- Префикс **S** на рисунке указывает на сервер, а **C** - на клиента. Параметры в скобках обозначают относительные значения ISN.
- Каждое соединение должно иметь свой неповторимый код ISN. Для реализации режима соединения прикладная программа на одном конце канала устанавливается в режим пассивного доступа ("passive open"), а операционная система на другом конце ставится в режим активного доступа ("active open"). Протокол TCP предполагает реализацию 11 состояний established, closed, listen, syn_sent, syn_received и т.д.



- Здесь состояние closed является начальной и конечной точкой последовательности переходов. Каждое соединение стартует из состояния closed. Из диаграммы машины состояний видно, что ни одному из состояний не поставлен в соответствие какой-либо таймер. Это означает, что машина состояний TCP может оставаться в любом из состояний сколь угодно долго. Исключение составляет keep-alive таймер, но его работа является опционной, а время по умолчанию составляет 2 часа. Это означает, что машина состояния может оставаться 2 часа без движения.

# МАШИНА СОСТОЯНИЙ ДЛЯ ПРОТОКОЛА TCP



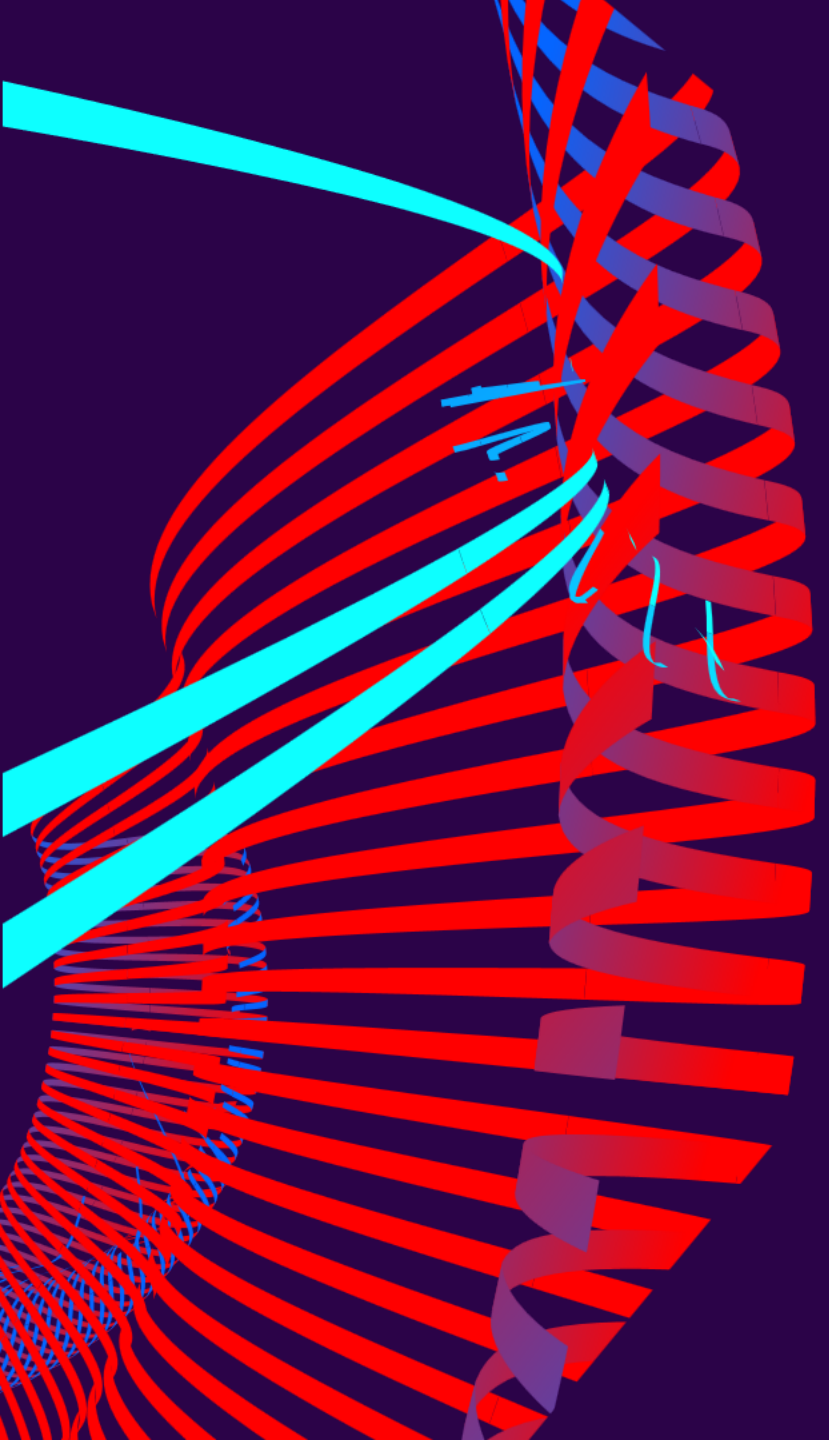
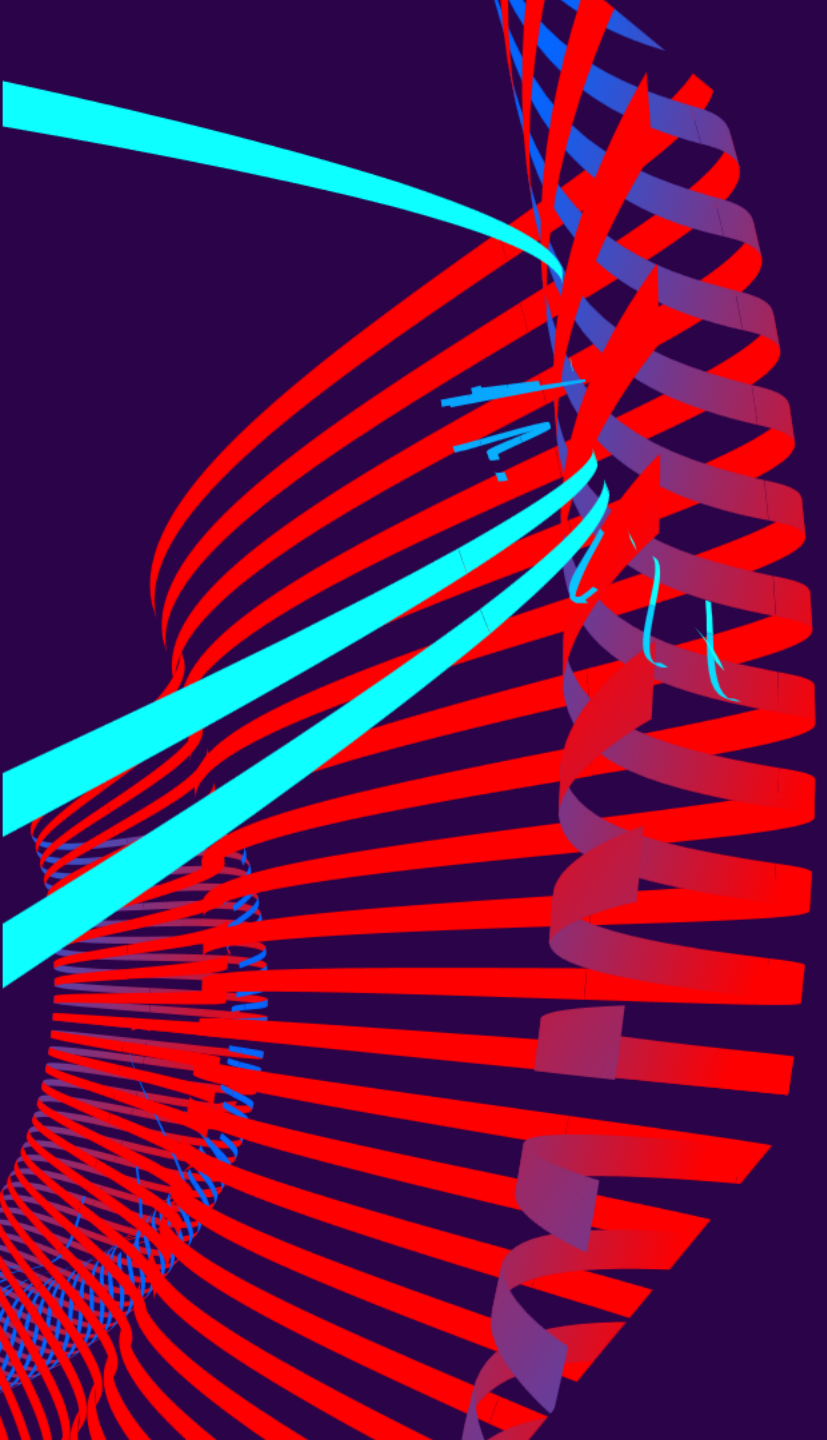


Диаграмма показывает все возможные состояния TCP-соединения и переходы между ними

Начальные и промежуточные состояния:

- CLOSED - соединение не существует
- LISTEN - сервер ждет входящих соединений
- SYN_SENT - клиент отправил SYN, ждет ответ
- SYN_RCVD - сервер получил SYN, отправил SYN-ACK
- ESTABLISHED - соединение установлено, идет передача данных
- Состояния закрытия:
- FIN_WAIT_1 - отправили FIN, ждем ACK
- FIN_WAIT_2 - получили ACK на FIN, ждем FIN от партнера
- CLOSE_WAIT - получили FIN, отправили ACK
- CLOSING - одновременное закрытие
- LAST_ACK - отправили FIN, ждем ACK
- TIME_WAIT - ожидание 2*MSL перед полным закрытием



ПРИМЕР:

Начальная ситуация

Клиент: Ваш компьютер с браузером Chrome

Сервер: Веб-сервер Google на порту 443

Изначально соединения нет. Сервер Google уже запущен и "слушает" порт 443, ожидая подключений. Ваш браузер только что был открыт.

**Часть 1:** Установление соединения

Вы вводите https://google.com и нажимаете Enter

Браузер решает установить TCP-соединение:

Он выбирает случайный локальный порт (допустим, 54321)

Формирует специальный пакет "SYN" с начальным номером последовательности (например, 100)

Отправляет его на адрес Google, порт 443



В этот момент TCP на вашем компьютере переходит в состояние "SYN_SENT" — отправили запрос на соединение, ждем ответа.

Сервер Google получает запрос:

Он отвечает пакетом "SYN-ACK"

В нем свой начальный номер (например, 500) и подтверждение вашего номера 101 (т.е. "я получил ваш SYN с номером 100, теперь жду байт 101")

Сервер переходит в состояние "SYN_RCVD" — получил запрос, отправил ответ.

Ваш компьютер получает ответ Google:

Отправляет финальное подтверждение "ACK" с номером 501 ("я получил ваш SYN с номером 500, теперь жду байт 501")

Теперь обе стороны в состоянии "ESTABLISHED" — соединение установлено!

Результат: За 3 обмена пакетами установлено надежное соединение. Теперь можно передавать HTTPS-данные.



**Часть 2:** Работа соединения

Браузер отправляет запрос: "Дай мне главную страницу Google"

Данные разбиваются на сегменты

Каждый сегмент имеет порядковый номер

Сервер Google отправляет ответ:

HTML-код страницы, CSS-стили, JavaScript-файлы, Изображения

Ваш компьютер подтверждает получение каждого сегмента

Отправляет "ACK" на каждый полученный блок данных

Если какой-то сегмент теряется, он запрашивается повторно

Соединение активно работает несколько секунд или минут, пока вы просматриваете страницу.

**Часть 3:** Закрытие соединения

Сценарий А: Вы закрываете вкладку



Вы нажимаете крестик на вкладке Google

Браузер говорит TCP: "Закрой соединение"

Ваш компьютер отправляет пакет с флагом "FIN" (завершение)

Состояние меняется на "FIN_WAIT_1" — отправили запрос на закрытие, ждем подтверждения.

Сервер Google получает FIN:

Отправляет подтверждение "ACK": "Получил ваш FIN"

Уведомляет свое приложение (веб-сервер), что клиент хочет закрыться

Сервер в состоянии "CLOSE_WAIT" — получил запрос на закрытие, но еще не завершил свою работу.

Ваш компьютер получает ACK:

Переходит в состояние "FIN_WAIT_2" — наша часть закрыта, ждем когда сервер тоже закроется.

Сервер Google завершает свою работу:



Веб-сервер отправляет оставшиеся данные (если есть)

Затем отправляет свой пакет "FIN"

Состояние сервера меняется на "LAST_ACK" — отправили свой FIN, ждем последнее подтверждение.

Ваш компьютер получает FIN от сервера:

Отправляет финальное "ACK"

Переходит в состояние "TIME_WAIT"

Ваш компьютер ждет 1-2 минуты в состоянии TIME_WAIT:

Гарантирует, что все пакеты "умерли" в сети

Предотвращает проблемы с новыми соединениями

После этого соединение полностью закрыто.

Сценарий B: Сервер закрывает соединение первым

Сервер Google решает закрыть соединение (например, из-за таймаута неактивности):



Отправляет пакет "FIN" вашему компьютеру

Ваш компьютер получает FIN:

Отправляет "ACK": "Получил ваш запрос на закрытие"

Уведомляет браузер, что сервер хочет закрыться

Браузер завершает свою работу и отправляет свой FIN

Сервер получает FIN и отправляет финальное ACK

Теперь сервер ждет в TIME_WAIT, а ваш компьютер уже закрылся

## Что видит пользователь?

При открытии сайта:

Задержка 0.1-0.3 секунды на установление соединения (3 пакета туда-обратно)

Затем страница начинает загружаться

При закрытии:



Вкладка закрывается мгновенно (пользователь не видит 4-этапного закрытия)

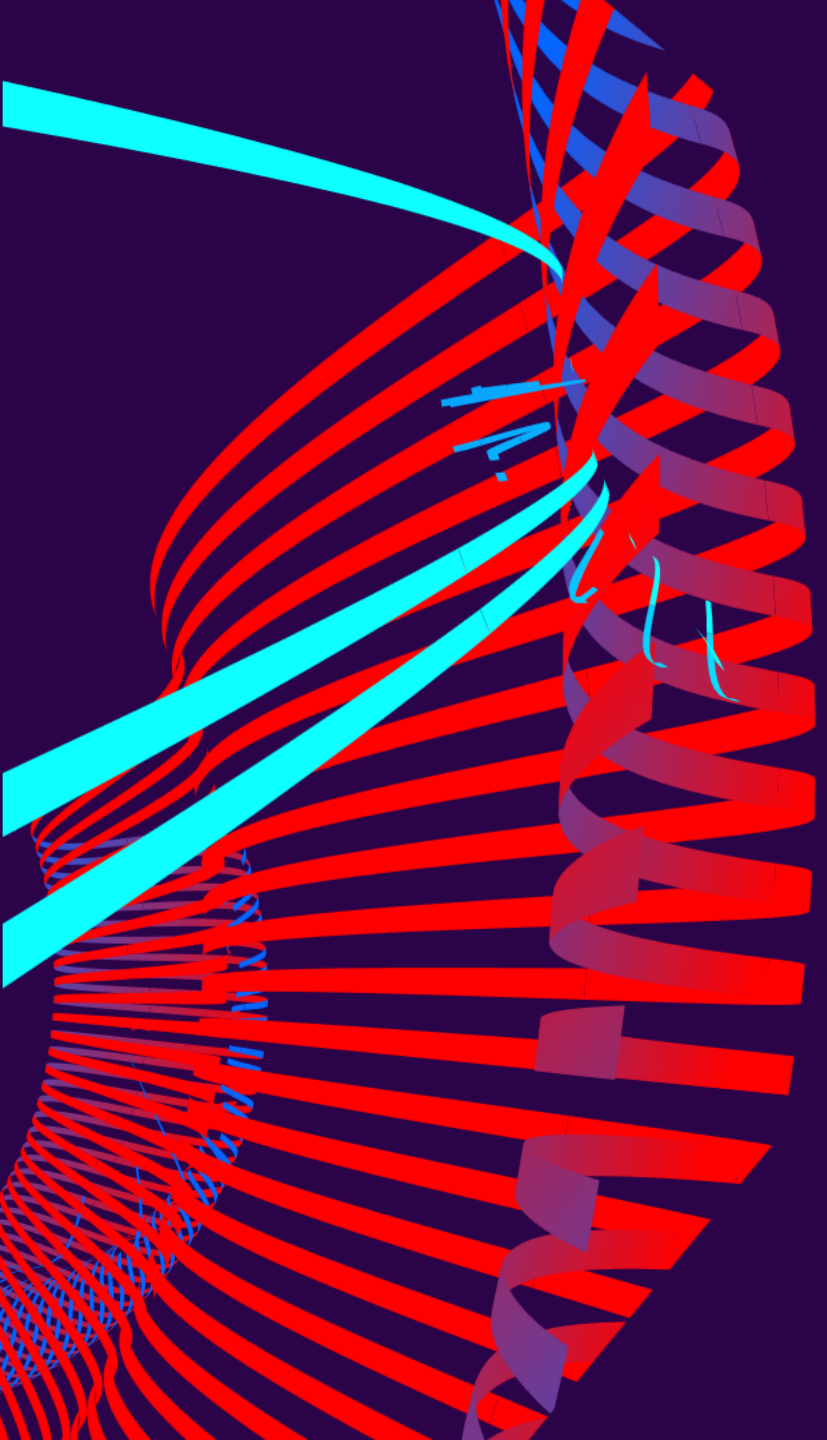
Но в фоне система еще 1-2 минуты "помнит" это соединение

Эта многоэтапная процедура обеспечивает:

Надежность: Никакие данные не теряются при закрытии

Корректность: Обе стороны знают, что соединение действительно закрыто

Безопасность: Предотвращаются конфликты между старыми и новыми соединениями



# КОНЦЕПЦИЯ КВИТИРОВАНИЯ

**Концепция квитирования** в протоколе TCP/IP — это механизм, при котором **правильность передачи каждого сегмента (единицы данных) подтверждается квитанцией получателя**. Это один из традиционных методов обеспечения надёжной связи, который позволяет организовать повторную передачу искажённых данных.

**Идея квитирования:** отправитель нумерует отправляемые данные (например, сегменты) и ожидает от приёмника положительную квитанцию — служебное сообщение, извещающее о том, что исходный сегмент был получен и данные в нём оказались корректными.

## Принцип работы

- Отправитель приостанавливает передачу очередного сегмента до получения подтверждения о приёме предыдущего сегмента. Интервал ожидания устанавливается равным значению задержки повторной передачи (retransmit timer). Если в течение этого интервала времени не будет получено подтверждение, передача сегмента выполняется повторно.
- В некоторых протоколах приёмник в случае получения сегмента с искажёнными данными отправляет отрицательную квитанцию — явное указание, что данный сегмент нужно передать повторно.
- Время ожидания квитанции ограничено — при отправке каждого сегмента передатчик запускает таймер, и если по его истечению положительная квитанция не получена, сегмент считается утраченным (потерянным).

# ПЕРЕДАЧА ДАННЫХ

**Передача данных по протоколу TCP/IP** — это процесс обмена информацией между устройствами в сети, основанный на стеке протоколов TCP (Transmission Control Protocol) и IP (Internet Protocol).

# УПРАВЛЕНИЕ ПОТОКОМ ДАННЫХ

**Управление потоком данных в протоколе TCP/IP** — это механизм, который позволяет контролировать скорость передачи информации, предотвращая перегрузку сети.

# УСТАНОВКА И РАЗРЫВ СЕАНСА

Установление TCP-соединения происходит до того, как любая из других функций TCP сможет начать свою работу. Установление соединения относится к процессу инициализации полей "**Sequence**" и "**Acknowledgment**" и согласования используемых номеров портов. На рисунке показан пример процесса установления соединения.




**Трехэтапный** обмен пакетами:
SYN
SYN-ACK
ACK

Установление соединения относится к процессу инициализации полей "Sequence" и "Acknowledgment" и согласования используемых номеров портов. На рисунке (слайд 43) показан пример процесса установления соединения.

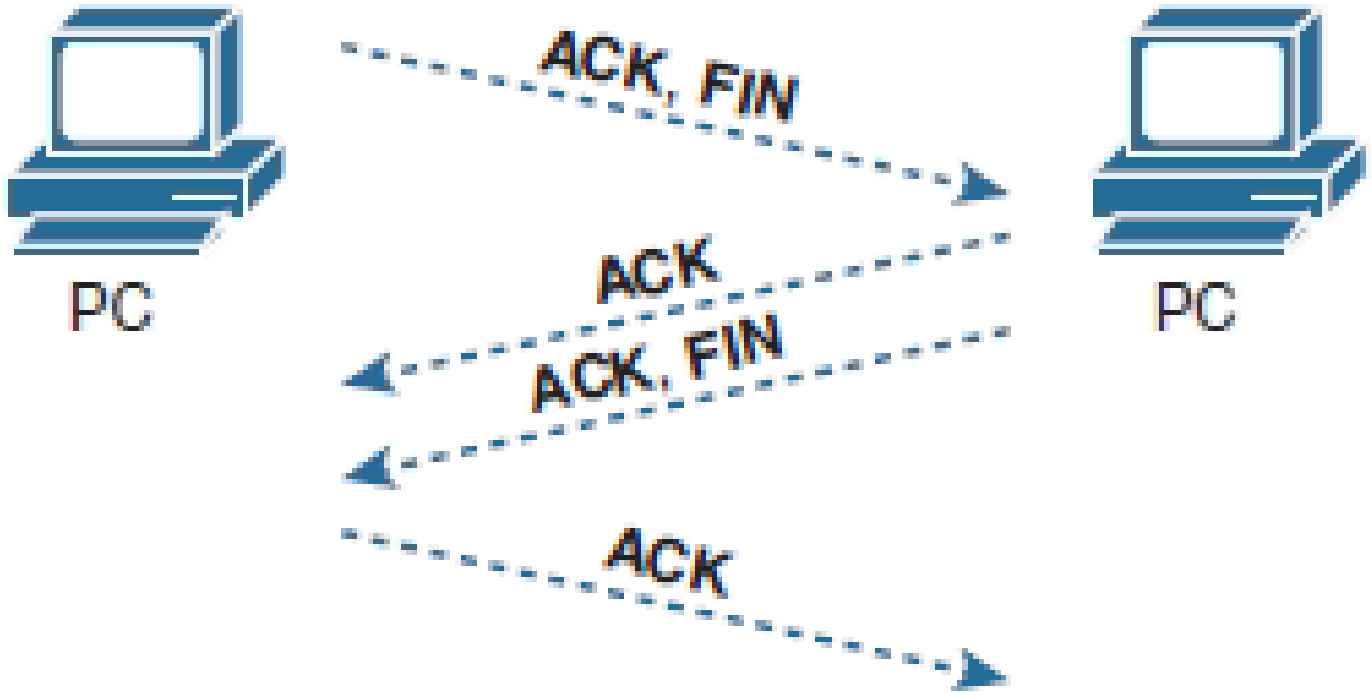
Этот трехэтапный процесс установления соединения (также называемый трехэтапным рукопожатием) должен завершиться до начала передачи данных. Соединение существует между двумя сокетами, хотя в заголовке TCP нет отдельного поля 'сокет'. Сокет состоит из IP-адреса и номера порта. IP-адреса берутся из IP-заголовка пакета, поэтому в TCP-заголовке нужно указать только номера портов.

TCP сообщает об установлении соединения, используя 2 бита в полях флагов заголовка TCP. Эти биты (SYN и ACK) имеют следующее значение:

➔ SYN означает "синхронизировать порядковые номера", что является одним из необходимых компонентов при инициализации TCP.

➔ ACK означает 'подтверждение' (acknowledgment) и указывает, что поле 'Номер подтверждения' в заголовке содержит корректный номер следующего ожидаемого байта.

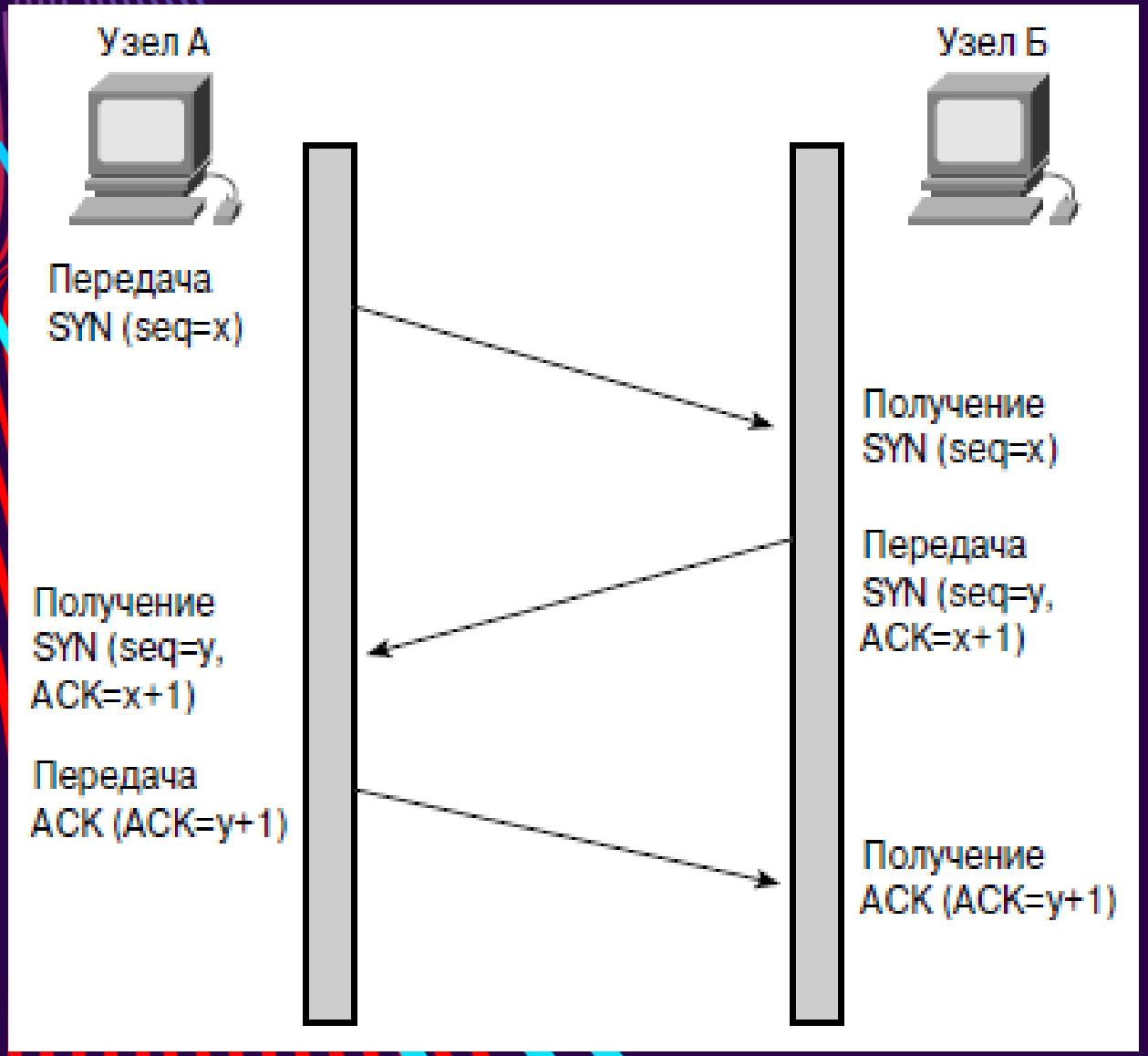
# ЗАВЕРШЕНИЕ TCP-СОЕДИНЕНИЯ




Четырехэтапная последовательность завершения. Используется дополнительный флаг, называемый битом FIN.

- **TCP устанавливает и завершает соединения между конечными точками, а UDP - нет.**
- Протокол, ориентированный на соединение: протокол, который требует обмена сообщениями до начала передачи данных или который имеет требуемую предварительно установленную корреляцию между двумя конечными точками.
- Протокол без установления соединения: протокол, который не требует обмена сообщениями и не требует предварительно установленной корреляции между двумя конечными точками.

# ТРЕХЭТАПНОЕ КВИТИРОВАНИЕ




**1. А -> Б SYN.** Мой начальный порядковый номер ISN равен X, номер ACK = 0, бит SYN установлен, однако бит ACK не установлен.

**2. Б -> А ACK.** Твой порядковый номер равен X+1, мой ISN-номер равен Y, биты SYN и ACK установлены.

**3. А -> Б ACK.** Твой порядковый номер равен Y+1, мой порядковый номер = X+1, бит ACK установлен, а бит SYN не установлен.

# ТРЕХЭТАПНОЕ КВИТИРОВАНИЕ

Синхронизация ISN достигается путем обмена специальными сегментами, содержащими:

- Бит SYN (Synchronize) — указывает на запрос синхронизации.
- Начальный порядковый номер (ISN) — случайное число, с которого начинается нумерация.

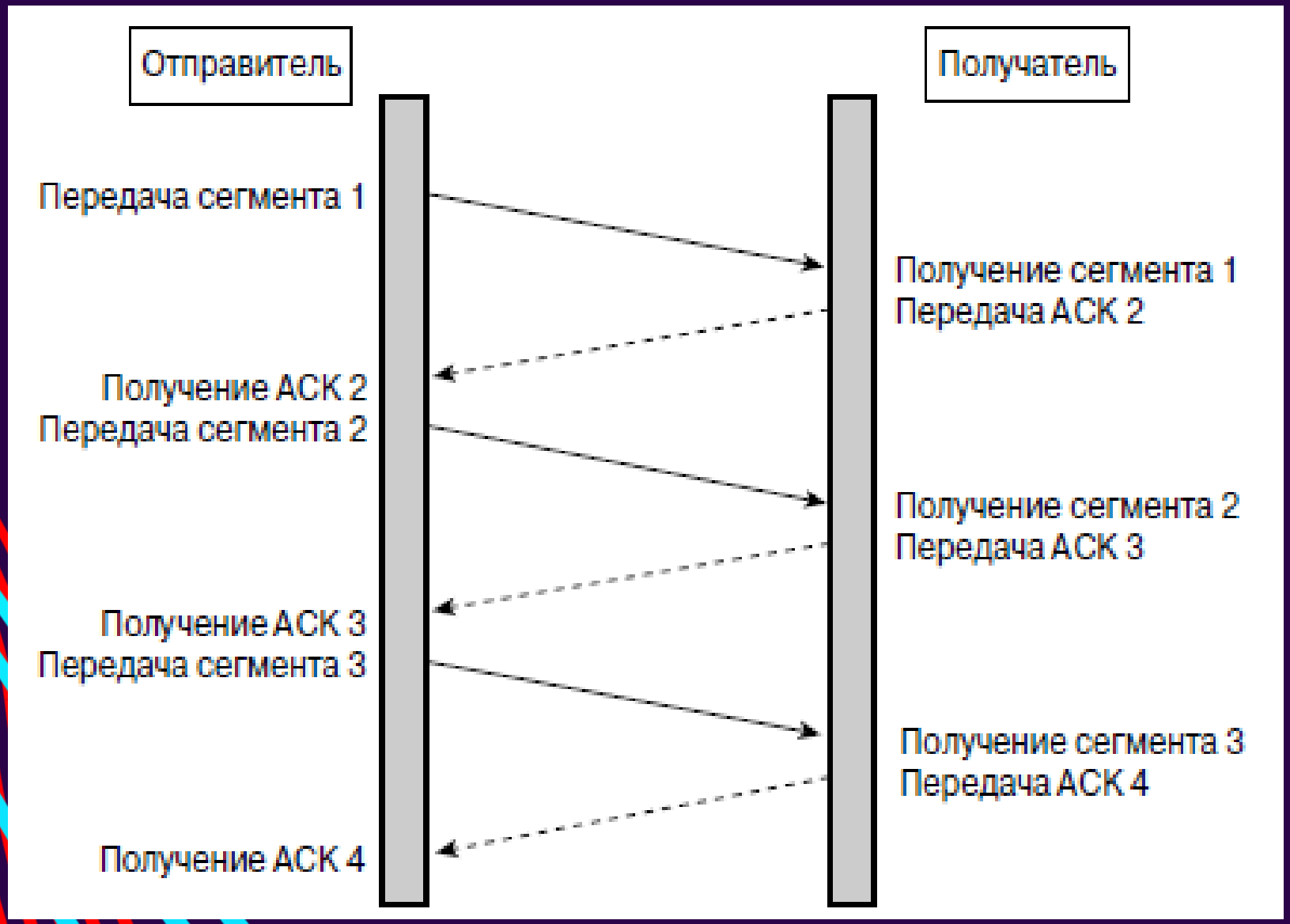
Такие сегменты называются SYN-сегментами или SYN-сообщениями.



Порядковые номера в TCP не берутся из какого-либо глобального сетевого счетчика. Вместо этого:

- Каждая сторона самостоятельно выбирает свой ISN
- ISN должен быть случайным (по соображениям безопасности)
- Номера синхронизируются в обоих направлениях.

# МЕХАНИЗМ СКОЛЬЗЯЩЕГО ОКНА



В наиболее общей форме надежной пересылки данных с установлением соединения пакеты данных должны доставляться принимающей стороне в том же порядке, в котором они передавались. Протокол сигнализирует о сбое, если какие-либо пакеты данных теряются, повреждаются, дублируются или принимаются в другом порядке. Наиболее простым решением такой задачи является использование подтверждений получателя о приеме каждого сегмента данных.

Однако если отправитель вынужден ждать подтверждения после посылки каждого сегмента, как показано на слайде 50, то скорость передачи при таком способе значительно снижается. Поскольку с того момента, как отправитель заканчивает отсылку пакета данных, до момента завершения обработки какого-либо принятого подтверждения проходит определенный интервал времени, он может быть использован для передачи дополнительной порции данных. Количество пакетов данных, которое разрешается пересылать отправителю без получения подтверждения, называется *окном* (window).

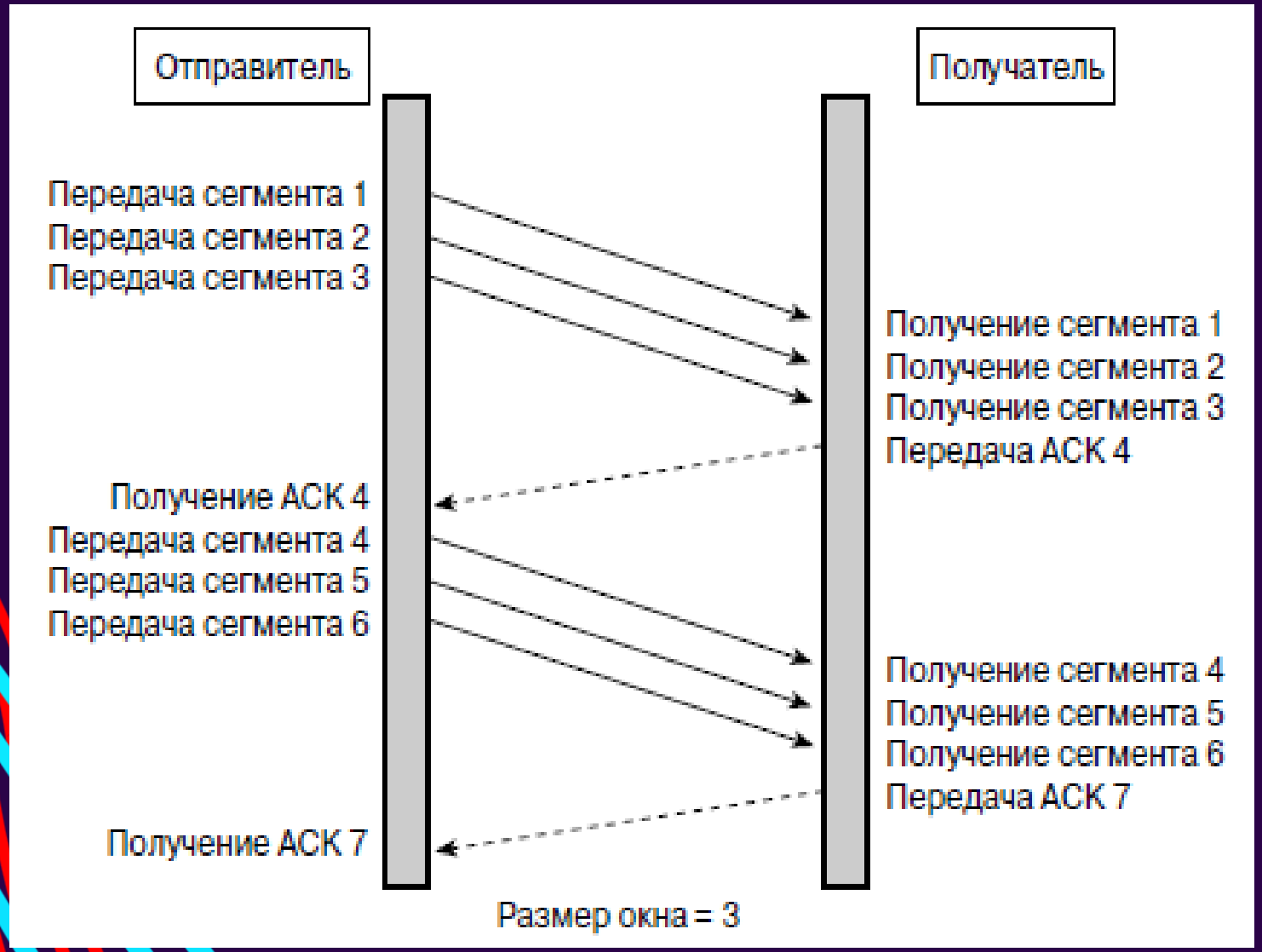
В протоколе TCP используются так называемые ожидаемые подтверждения; они содержат номер, относящийся к октету, который ожидается следующим. Механизм скользящего окна заключается в том, что согласование размеров окна происходит динамически в течение TCP-сеанса.

*Механизм скользящего окна* — это механизм управления потоком данных, который требует, чтобы получатель принимал подтверждение от отправителя после передачи некоторого количества данных.

Получатель докладывает отправителю о получении данных; получение такого уведомления позволяет установить размер окна. Окно определяет количество октетов, отсчитываемое от текущего номера подтверждения, которое TCP-устройство способно принять в заданный момент времени.

Размер окна может изменяться в процессе передачи потока данных между двумя сетевыми устройствами. В каждом подтверждении, отправленном от получателя, содержится информация о количестве байтов, которые получатель способен принять. В протоколе TCP предусмотрено использование окна управления заторами, которое в нормальном состоянии равно окну устройства-получателя, но его размер уменьшается вдвое, если теряется какой-либо сегмент данных (например, при перегрузке в сети).

# МЕХАНИЗМ ПОДТВЕРЖДЕНИЙ

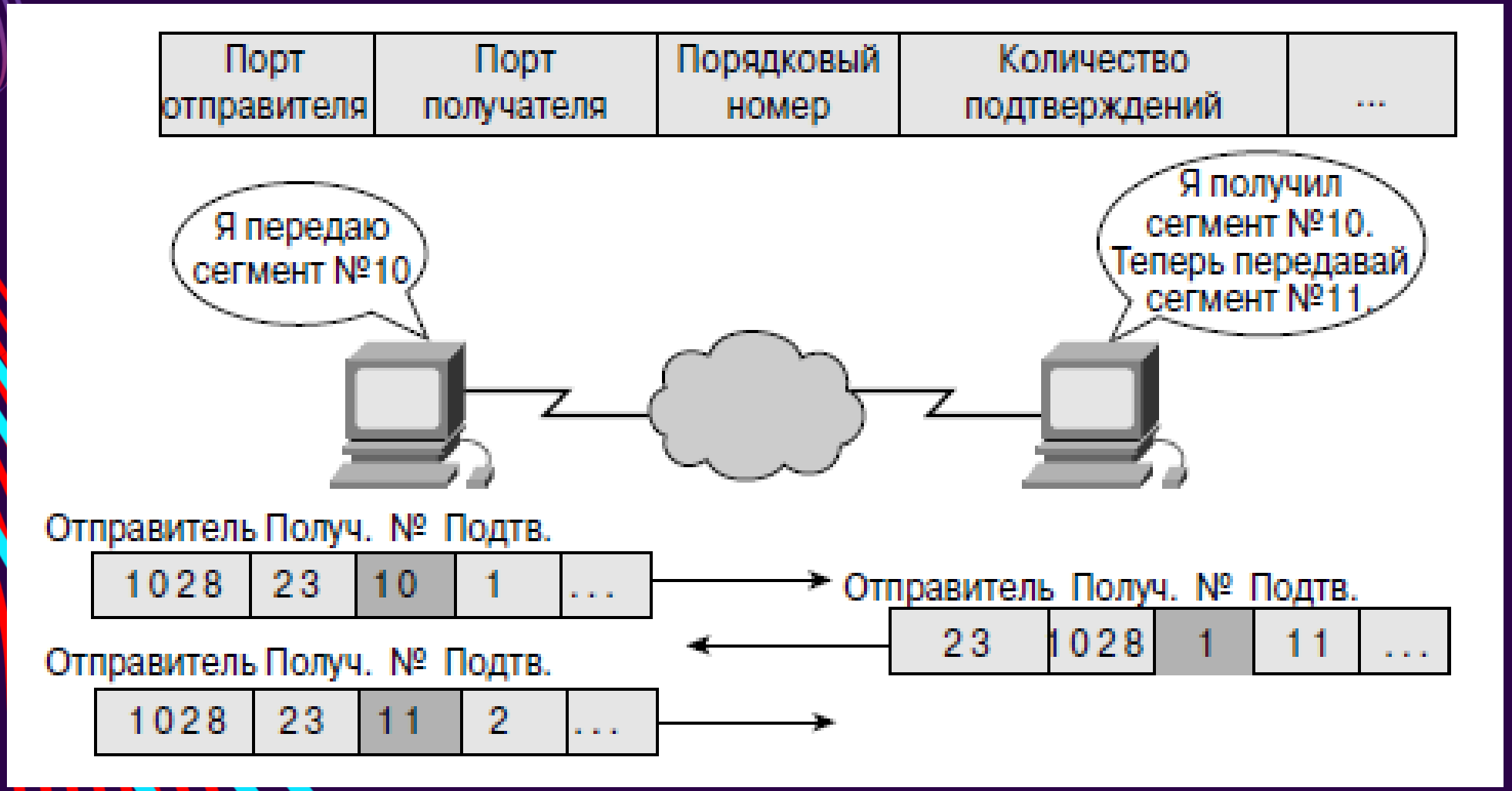


Отправитель регистрирует каждый переданный им пакет и перед отправкой следующего пакета данных ждет подтверждения. В момент пересылки сегмента отправитель также запускает таймер и повторно передает блок данных, если установленное таймером время истекает до поступления подтверждения.

На слайде 57 показан отправитель, который передает пакеты 1, 2 и 3. Получатель подтверждает прием пакетов, запрашивая пакет 4. Отправитель, получив подтверждение, посылает пакеты 4, 5 и 6.

Если пакет 5 не доставляется получателю, он посылает соответствующее подтверждение с запросом о повторной отправке пакета 5. Отправитель повторно отсылает пакет 5 и должен получить соответствующее подтверждение, чтобы продолжить передачу пакета с номером 7.

# МЕХАНИЗМ ПОДТВЕРЖДЕНИЙ



Протокол TCP обеспечивает соблюдение последовательности сегментов с последующим подтверждением. Каждой дейтаграмме перед передачей присваивается номер (можно посмотреть на слайде 59). После того как получатель принял все дейтаграммы, они собираются в завершенное сообщение. В обязанности протокола TCP входит восстановление поврежденных, утерянных, дублированных или пришедших в неверном порядке данных, которые передавались через сеть Internet.

Механизм восстановления функционирует за счет назначения порядкового номера каждому переданному октету, после приема которого получатель должен отправить подтверждение (ACK). Если же в течение интервала времени ожидания подтверждение не было получено, данные передаются отправителем повторно. После доставки октетов получателю их порядковые номера используются для сборки сообщения из фрагментов и устранения дубликатов. Поврежденные данные восстанавливаются при помощи контрольной суммы, которая добавляется к каждому передаваемому сегменту. Контрольная сумма проверяется получателем, и, если она не совпадает, поврежденные данные отбрасываются.

# ПРОТОКОЛ ПОЛЬЗОВАТЕЛЬСКИХ ДЕЙТАГРАММ UDP

*UDP (User Datagram Protocol - протокол передачи дейтаграмм пользователя*) является транспортным протоколом без установления соединения в стеке протоколов TCP/IP.



UDP — это простой протокол, который осуществляет обмен дейтаграммами без подтверждения и без гарантии доставки. Простота протокола становится очевидной при сравнении форматов сегментов протоколов UDP и TCP. При использовании протокола UDP обработка ошибок и повторная передача данных должна осуществляться протоколом более высокого уровня.

Список полей UDP-сегмента:

**Порт отправителя -** номер вызывающего порта.

**Порт получателя -** номер вызываемого порта.

**Длина** - количество байтов, включая заголовок и данные.

**Контрольная сумма** - расчетная контрольная сумма заголовка и полей данных.

**Данные** - данные протокола более высокого уровня.

В протоколе UDP не используется механизм скользящего окна, поэтому надежность передачи данных должна обеспечиваться *протоколами уровня приложений*.

Протокол UDP используют такие службы и протоколы верхнего уровня:

- TFTP (Trivial File Transfer Protocol - простейший протокол передачи файлов);
- SNMP (Simple Network Management Protocol - простой протокол управления сетью);
- DHCP (Dynamic Host Configuration Protocol - протокол динамической конфигурации узла);
- DNS (Domain Name System - служба доменных имен).

# Служебные порты протокола UDP



**Служебные порты протокола UDP** — это номера портов, которые идентифицируют приложения и службы, использующие UDP. Порты представлены 16-битными целочисленными значениями (от 0 до 65 535).

Номера портов UDP разделены на три группы:

1. Порты с номерами от 0 до 1023 — для обычных, хорошо известных служб. В Unix-подобных операционных системах для использования таких портов необходимо разрешение суперпользователя.

2. Порты с номерами от 1024 до 49 151 — для зарегистрированных IANA служб.

3. Порты с 49 152 по 65 535 — могут быть использованы для любых целей, поскольку официально не разработаны для какой-то определённой службы. Также используются как динамические (временные) порты, которые запущенное на хосте программное обеспечение может случайным образом выбрать для самоопределения.

# ФОРМАТ ПАКЕТА UDP

# Структура:

Заголовок UDP имеет фиксированный размер — 8 байт. Он включает следующие поля:

**Порт источника** — 16 бит, указывает порт отправителя. Позволяет получателю определить, откуда пришёл пакет, и при необходимости отправить ответ.

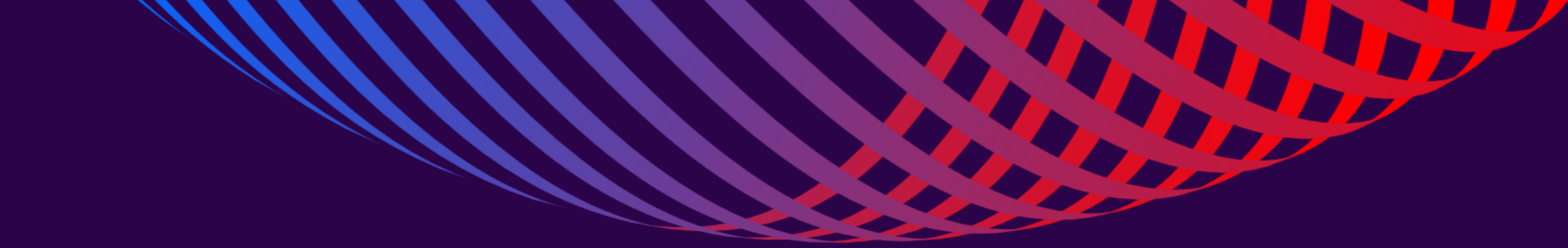
**Порт назначения** — 16 бит, указывает порт получателя. Позволяет отправителю указать, куда должен быть доставлен пакет данных.

**Длина** — 16 бит, указывает общую длину UDP-пакета, включая заголовок и данные. Позволяет получателю определить, сколько данных содержится в пакете.

**Контрольная сумма**— 16 бит, используется для проверки целостности данных. Позволяет получателю проверить, не были ли данные повреждены при передаче.

Не все поля обязательно должны быть заполнены. Например, если посылаемая дейтаграмма не предполагает ответа, то на месте адреса отправителя могут помещаться нули.

UDP применяется в сценариях, где приоритетом является скорость передачи данных, а не их гарантированная доставка.



# СРАВНЕНИЕ TCP И UDP

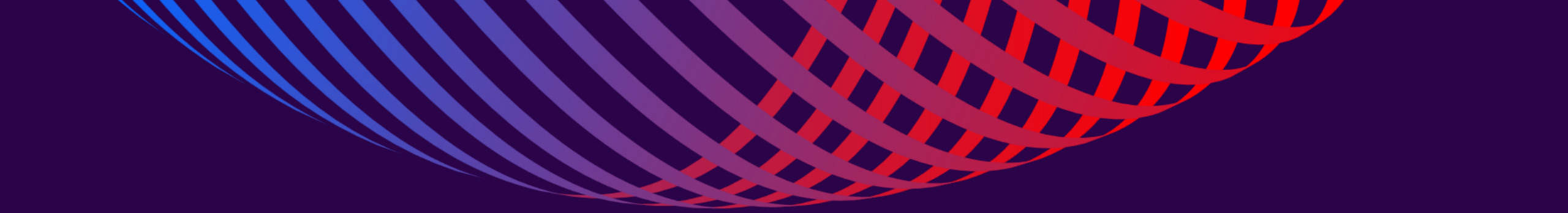
TCP

UDP



TCP применяется там, где требуется точная и подтверждаемая передача данных – например, отправка фотографий, или переписка между пользователями. UDP, в свою очередь, нужен для общения в голосовом формате, или при передаче потокового видео, например, с веб-камер или IP-камер.

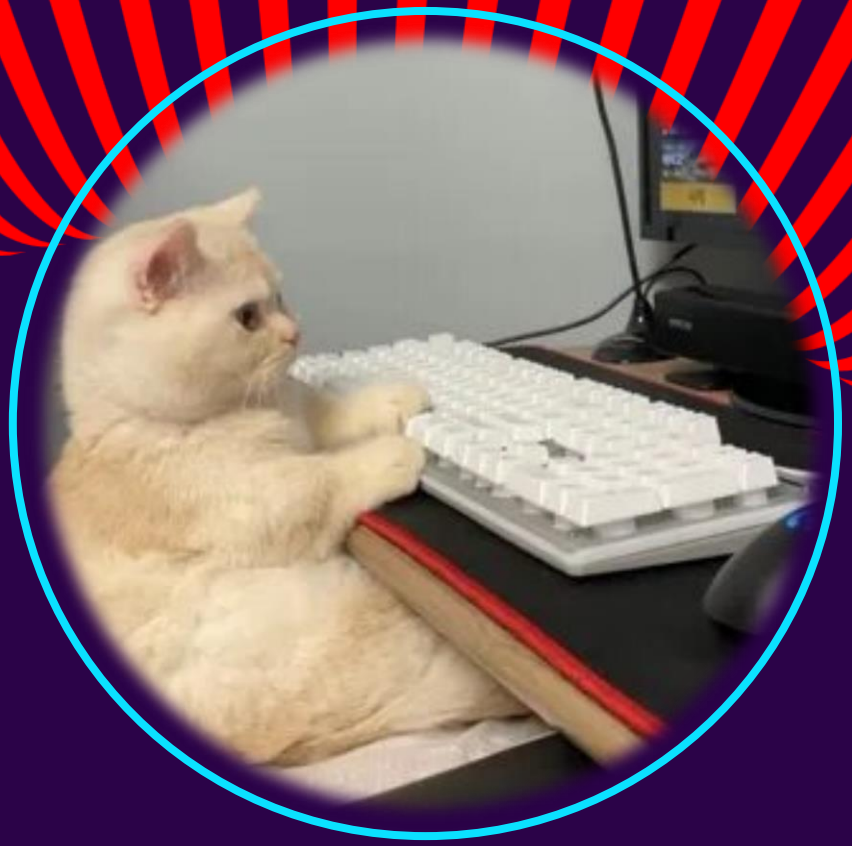
Протокол TCP – это сетевой протокол, который «заточен» под соединение. Иными словами, прежде, чем начать обмен данными, данному протоколу требуется установить соединение между двумя хостами. Он имеет высокую надежность, поскольку позволяет не терять данные при передаче, запрашивает подтверждения о получении от принимающей стороны и в случае необходимости отправляет данные повторно. При этом отправляемые пакеты данных сохраняют порядок отправки, то есть можно сказать, что передача данных упорядочена. Минусом данного протокола является относительно низкая скорость передачи данных, за счет того, что выполнение занимает больше времени, чем в альтернативном протоколе UDP.



Протокол UDP, в свою очередь, более прост. Для передачи данных ему не обязательно устанавливать соединение между отправителем и получателем. Информация передается без предварительной проверки готовности принимающей стороны. Это делает протокол менее надежным – при передаче некоторые фрагменты данных могут теряться. Кроме того, упорядоченность данных не соблюдается – возможен непоследовательный прием данных получателем. Зато скорость передачи данных по данному транспортному протоколу будет более высокой.

| Критерий | TCP | UDP |
|---|---|---|
| **Тип соединения** | Протокол с установлением соединения (connection-oriented) | Протокол без установления соединения (connectionless) |
| **Надежность доставки** | Гарантированная доставка данных, подтверждение получения | Доставка не гарантируется, подтверждения отсутствуют |
| **Упорядоченность данных** | Данные передаются и принимаются в строгом порядке | Данные могут приходить в произвольном порядке |
| **Метод передачи** | Потоковый (передача непрерывного потока байтов) | Дейтаграммный (передача отдельных независимых пакетов) |
| **Скорость передачи** | Относительно низкая из-за накладных расходов на контроль и подтверждение | Высокая, минимальные накладные расходы |
| **Контроль перегрузок** | Реализован механизм управления перегрузками и потоком данных | Отсутствует контроль перегрузок |
| **Использование** | HTTP, HTTPS, FTP, SMTP, SSH, Telnet — где важна надежность | DNS, DHCP, SNMP, VoIP, онлайн-игры — где важна скорость и малая задержка |
| **Размер заголовка** | 20–60 байт (переменный) | 8 байт (фиксированный) |
| **Пример аналогии** | Заказное письмо с уведомлением о вручении | Обычная почтовая открытка без отслеживания |

# СПАСИБО ЗА ВНИМАНИЕ!



# ПОЛЯ TCP-СЕГМЕНТА

- **Порт отправителя -** номер вызывающего порта.
- **Порт получателя -** номер вызываемого порта.
- **Порядковый номер -** номер, используемый для расположения поступающих данных в правильной последовательности.
- **Номер подтверждения** - номер следующего ожидаемого TCP-октета.
- **HLEN -** количество 32!разрядных слов в заголовке.
- **Зарезервированное поле** - все биты установлены в значение 0.
- **Биты кода** - служебные функции (например, установка и завершение сеанса).
- **Окно** - количество октетов, с которым отправитель готов согласиться.
- **Контрольная сумма** - расчетная контрольная сумма заголовка и полей данных.
- **Указатель срочных данных** - указывает конец срочных данных.
- **Параметры** - в настоящее время определен один параметр: максимальный размер TCP-сегмента.
- **Данные** - данные протокола более высокого уровня.

# ФЛАГИ УПРАВЛЕНИЯ:

- **URG (Urgent - Срочный):**

Указывает на наличие срочных данных

Используется редко в современных приложениях

- **ACK (Acknowledgment - Подтверждение):**

Поле номера подтверждения действительно

Устанавливается во всех сегментах, кроме начального SYN

- **PSH (Push - Проталкивание):**

Требование немедленной доставки данных приложению

Отключает алгоритм Нейгла (Nagle's algorithm)

- **RST (Reset - Сброс):**

Аварийный разрыв соединения

Используется при обнаружении неисправимых ошибок

- **SYN (Synchronize - Синхронизация):**

Запрос на установление соединения

Синхронизирует начальные номера последовательности

- **FIN (Finish - Завершение):**

Уведомление о завершении передачи данных

Корректное закрытие соединения