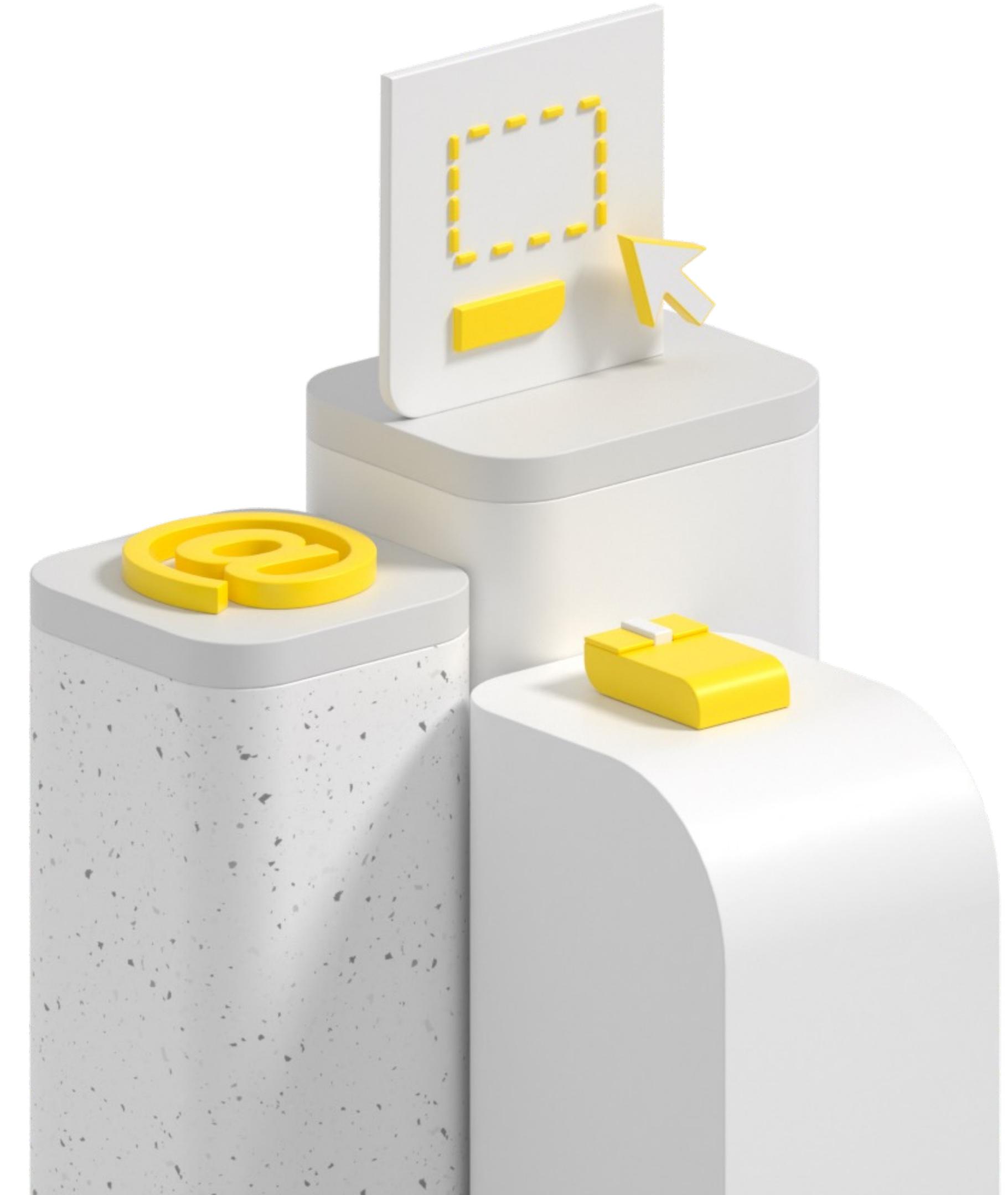




Основы сетей





Катаев Виталий | Сеньор Формошлеп



vita@kataev.pro



@vita1101

План



Транспортный

- TCP
- UDP



Сеансовый

- SSL/TLS



Представления данных

- SSL/TLS

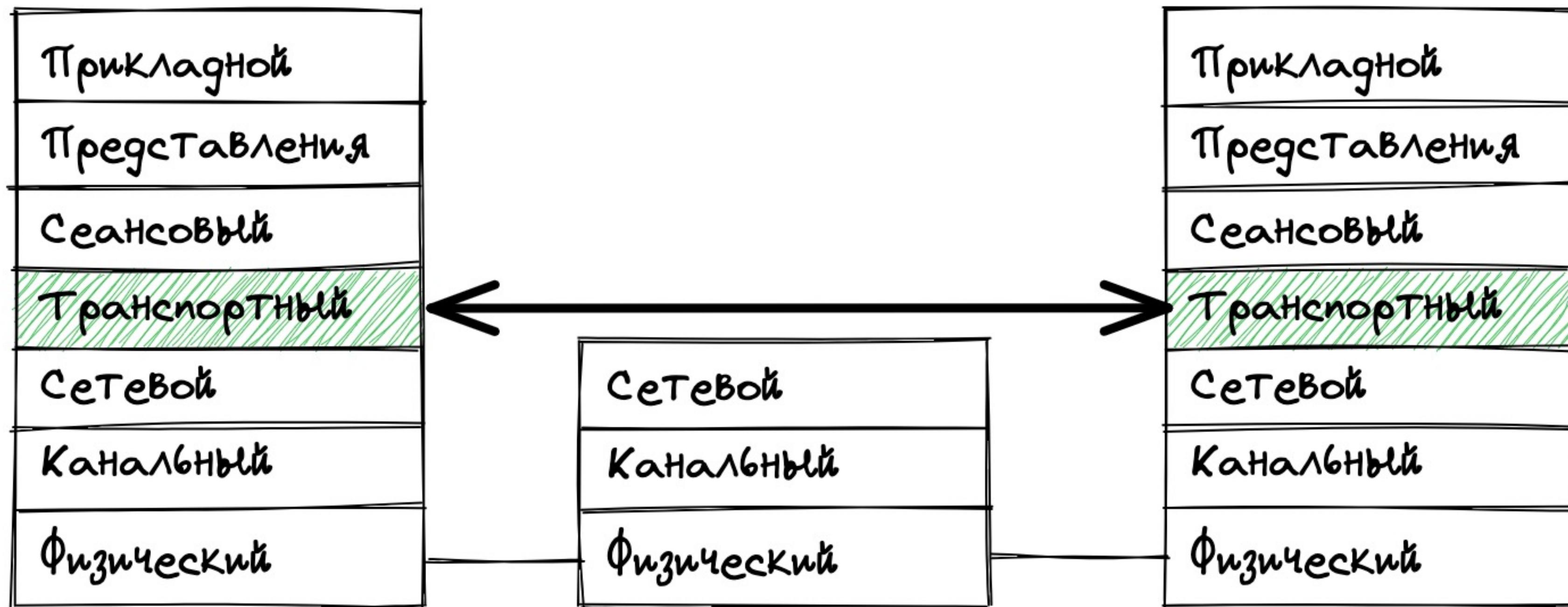


Прикладной

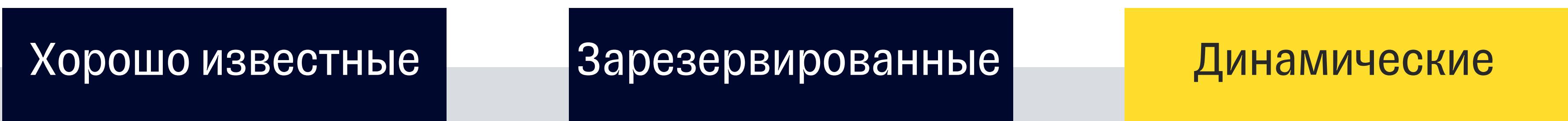
- HTTP(s)
- DNS
- WebSockets

Транспортный уровень

Транспортный уровень



Порты



1-1024

- 80 – HTTP
- 443 – HTTPS
- 22 – SSH
- 53 – DNS

1025-49151

Регистрируются в
Internet Assigned
Numbers Authority (IANA)

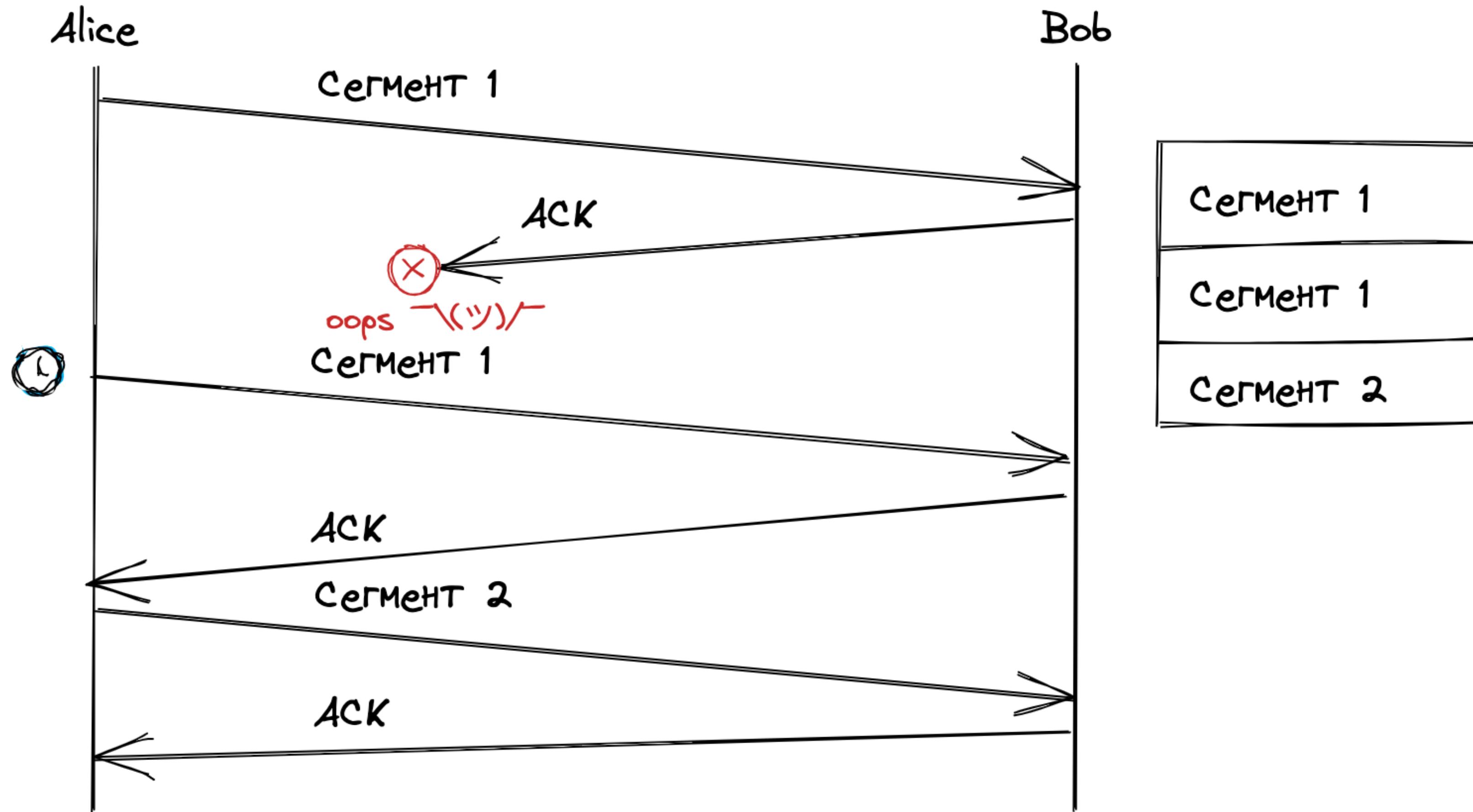
49152-65535

Автоматически
назначаются ОС

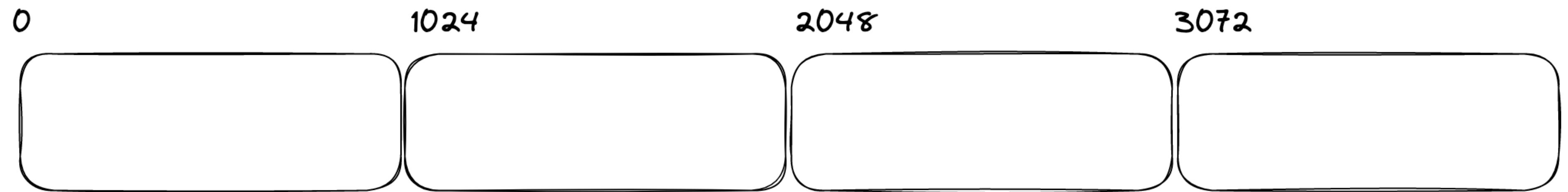
TCP



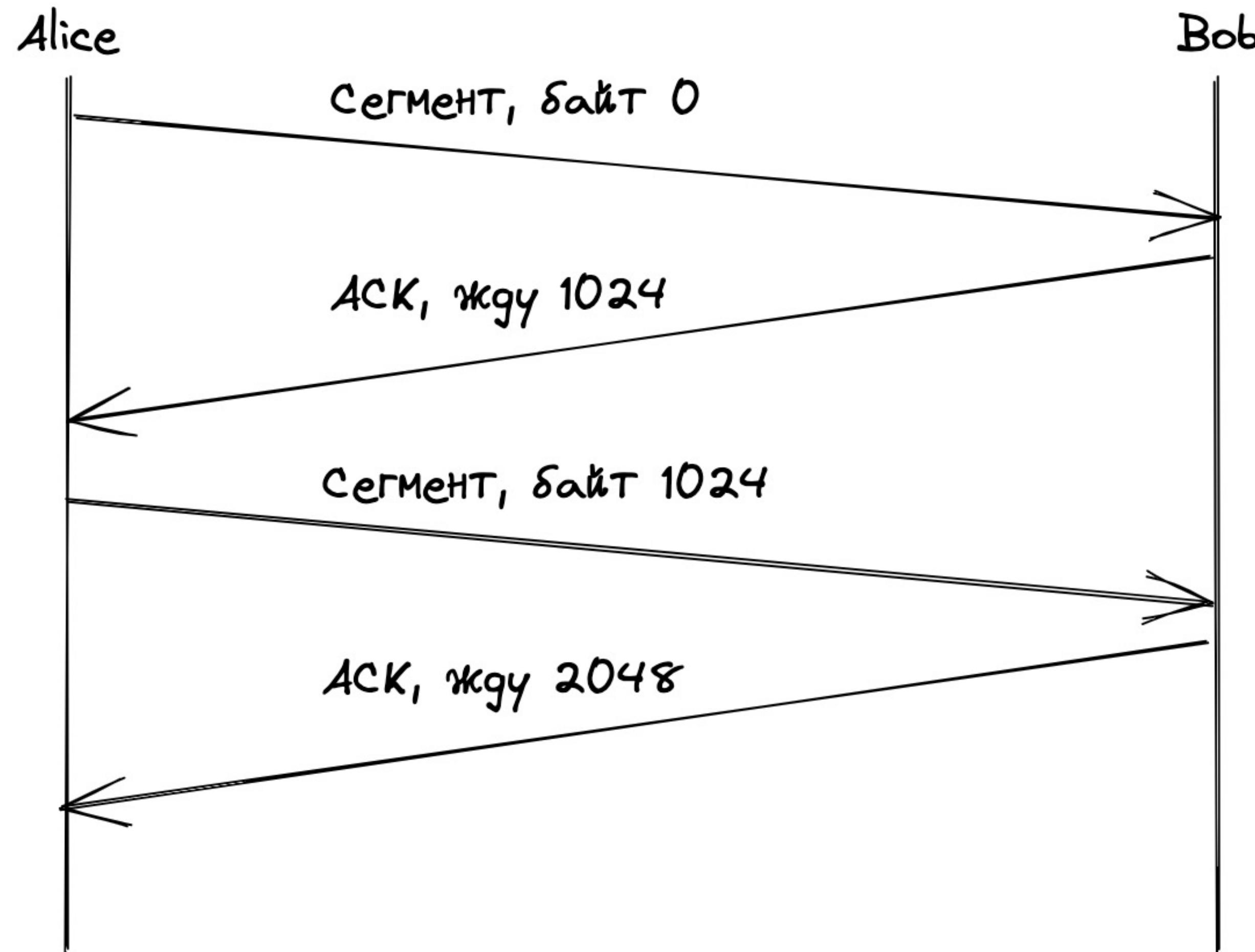
Ошибка фрагментации



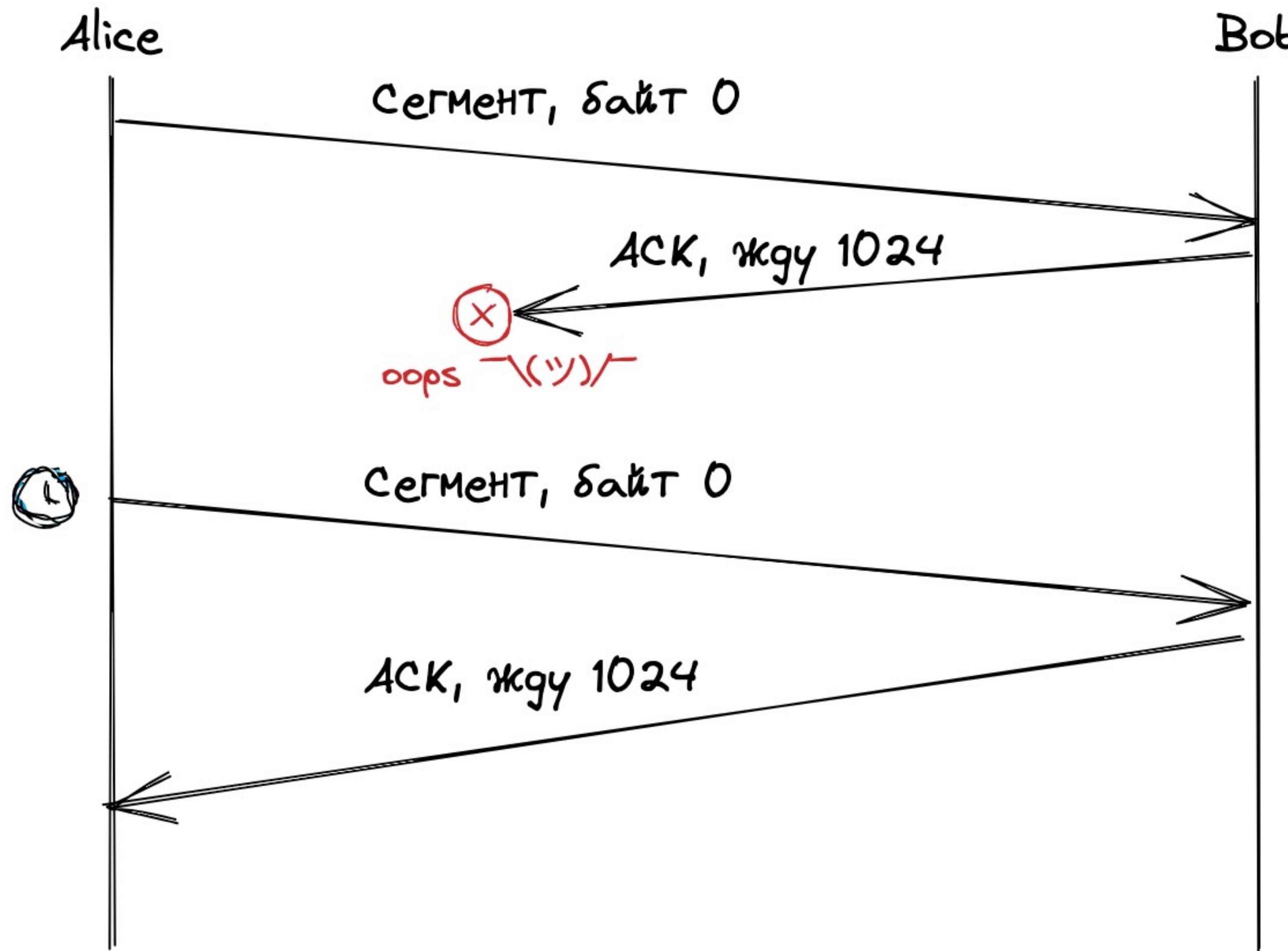
Нумерация сегментов



Нумерация сегментов



Нумерация сегментов



Механизмы подтверждения



Остановка и ожидание

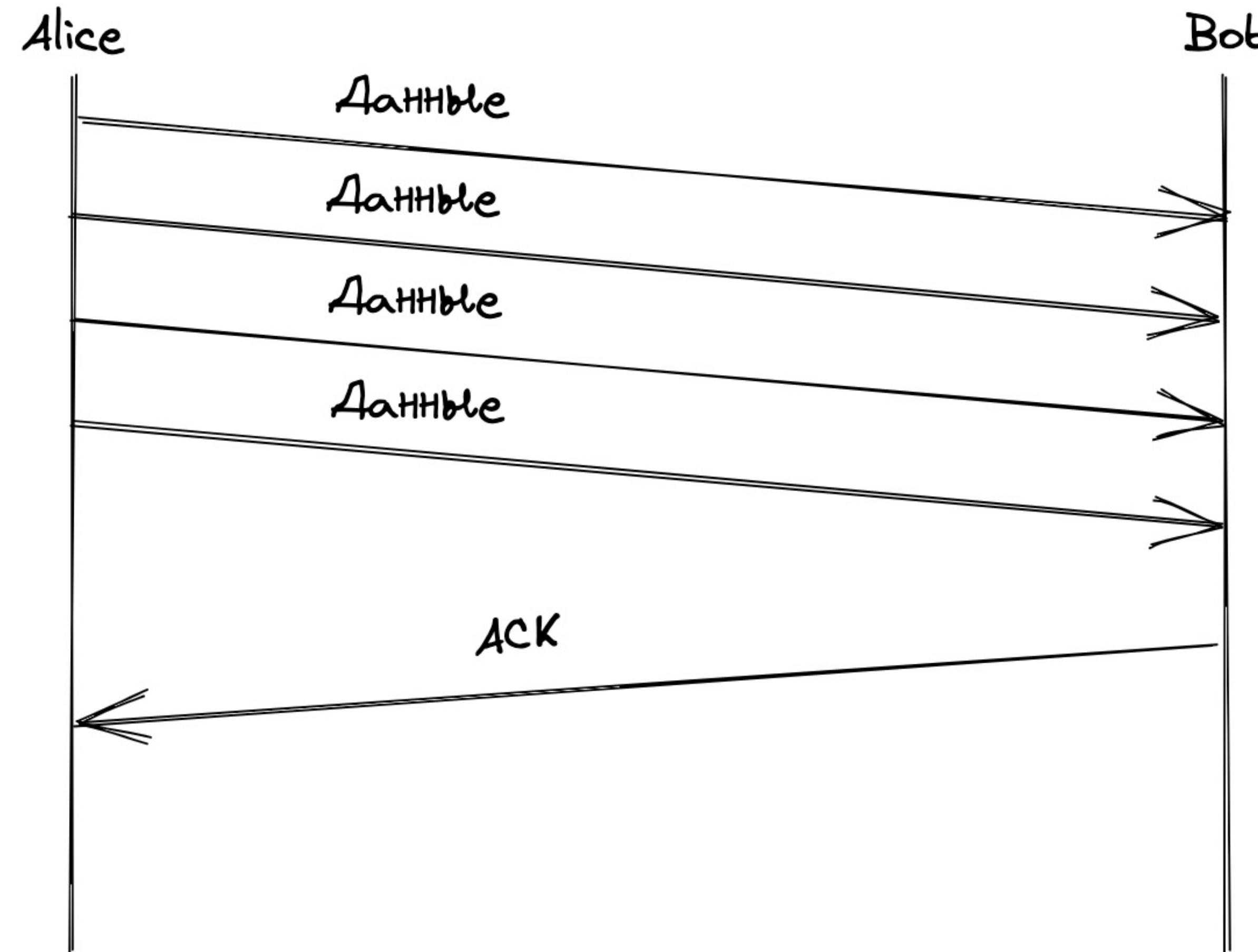
Каждый фрагмент
подтверждается отдельно



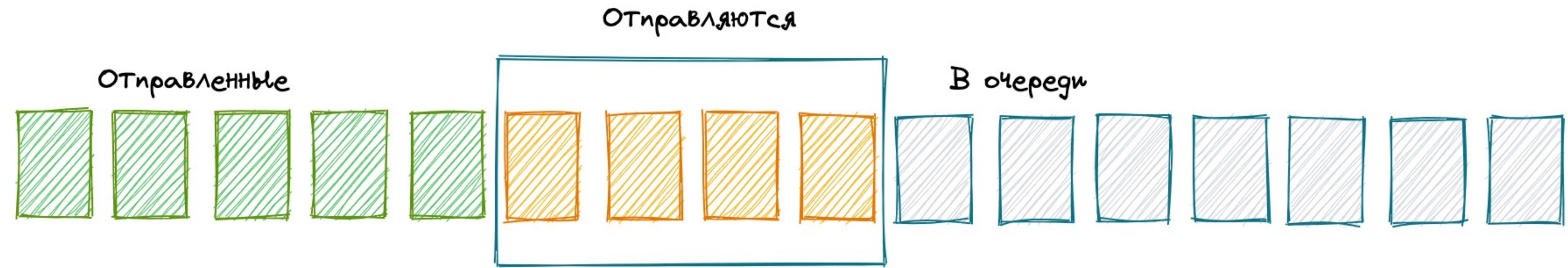
Скользящее окно

Одно подтверждение на
несколько фрагментов

Скользящее окно



Скользящее окно



Варианты подтверждения



Кумулятивное

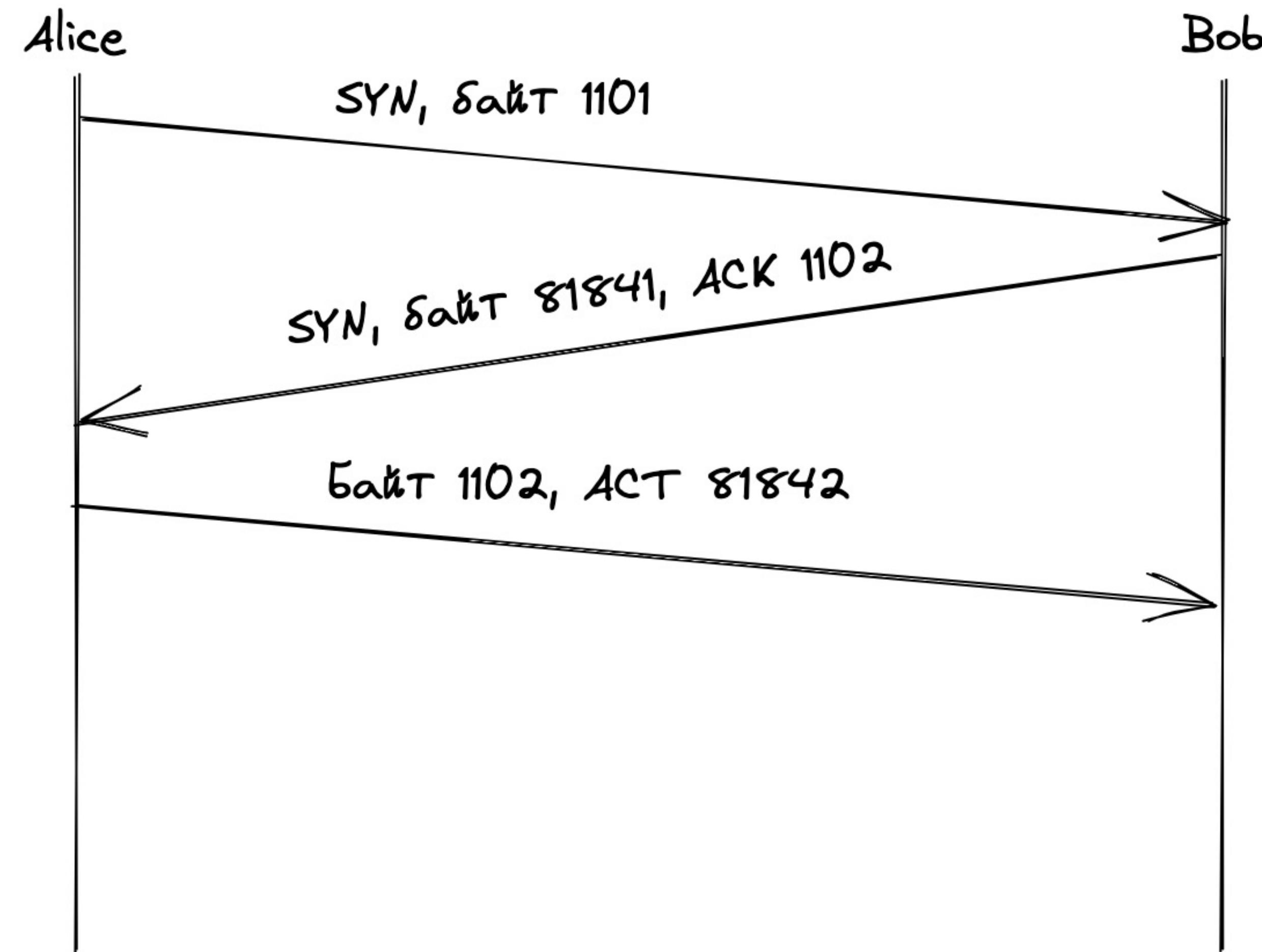
Подтверждение, что все фрагменты получены



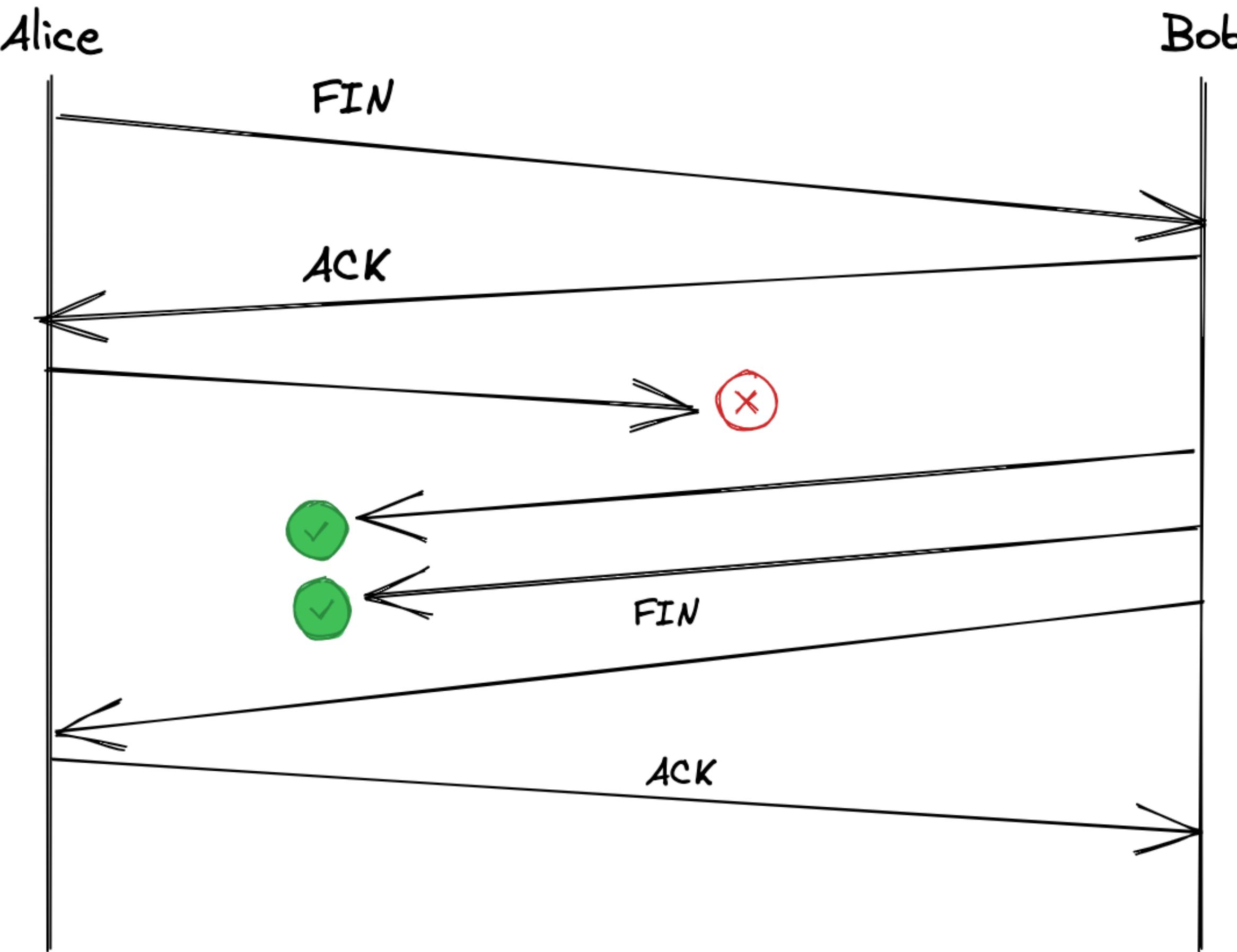
Селективное

Подтверждение только принятых фрагментов

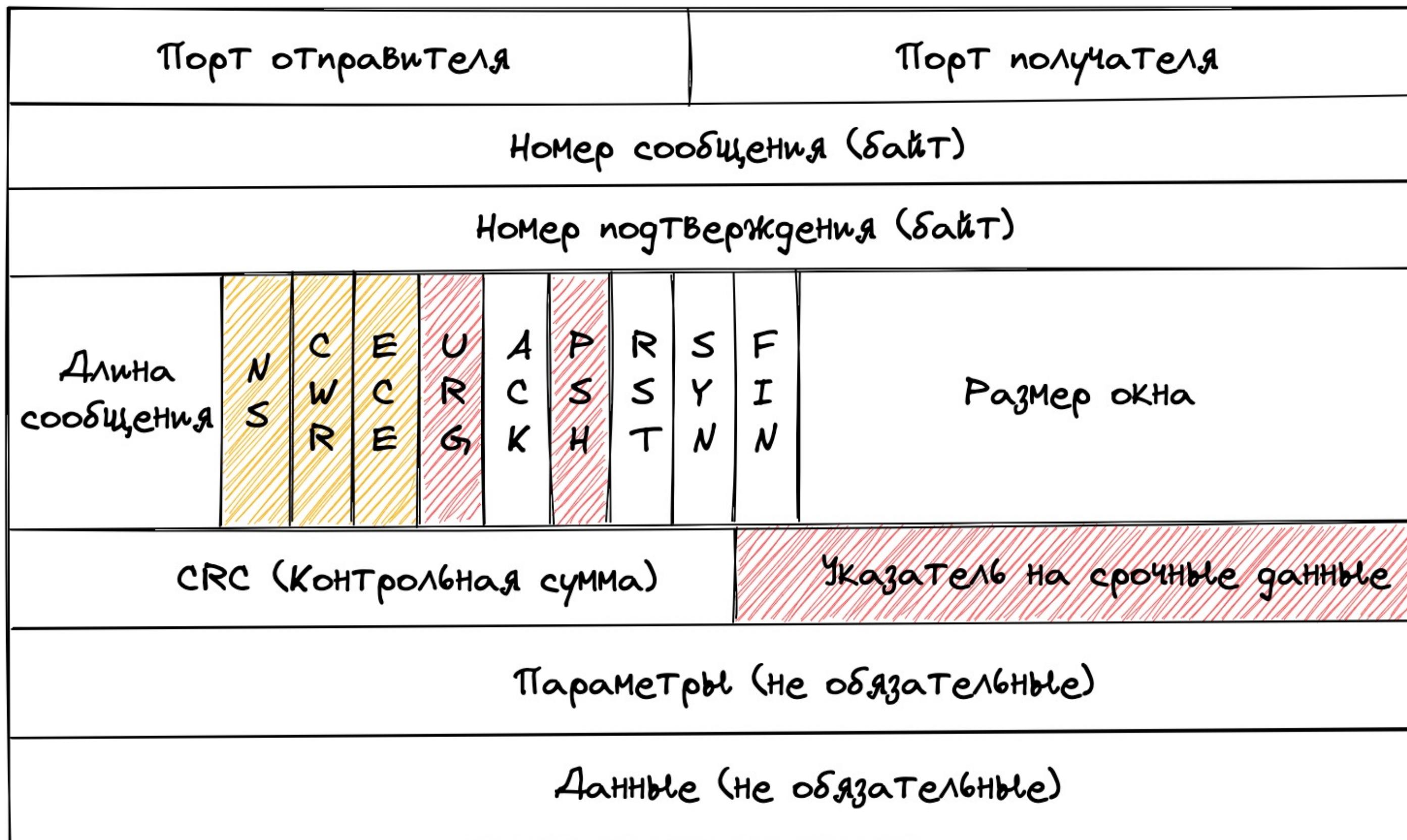
Установка соединения



Разрыв соединения



формат TCP сообщения



UDP



UDP



Недостатки

Нет гарантии доставки

Нет гарантии сохранения порядка

Нет соединения

Нет сегментации



Преимущества

Простота

Скорость

формат UDP дейтаграммы



Пример

TCP

1. Трехкратное рукопожатие - 3 сообщения (SYN, SYN-ACK, ACK)
2. Отправка пакета и получение подтверждения - 2 сообщения (сообщение + ACK)
3. Получение ответа и отправка подтверждения - 2 сообщения (ответ + ACK)
4. Завершение соединения и подтверждение - 2 сообщения (FIN + ACK)
5. Завершение соединения с другой стороны и подтверждение (FIN + ACK)

Пример

UDP

1. Отправка сообщения
2. Получение результата



DNS

DNS



DNS – Domain Name System

*nix - /etc/hosts

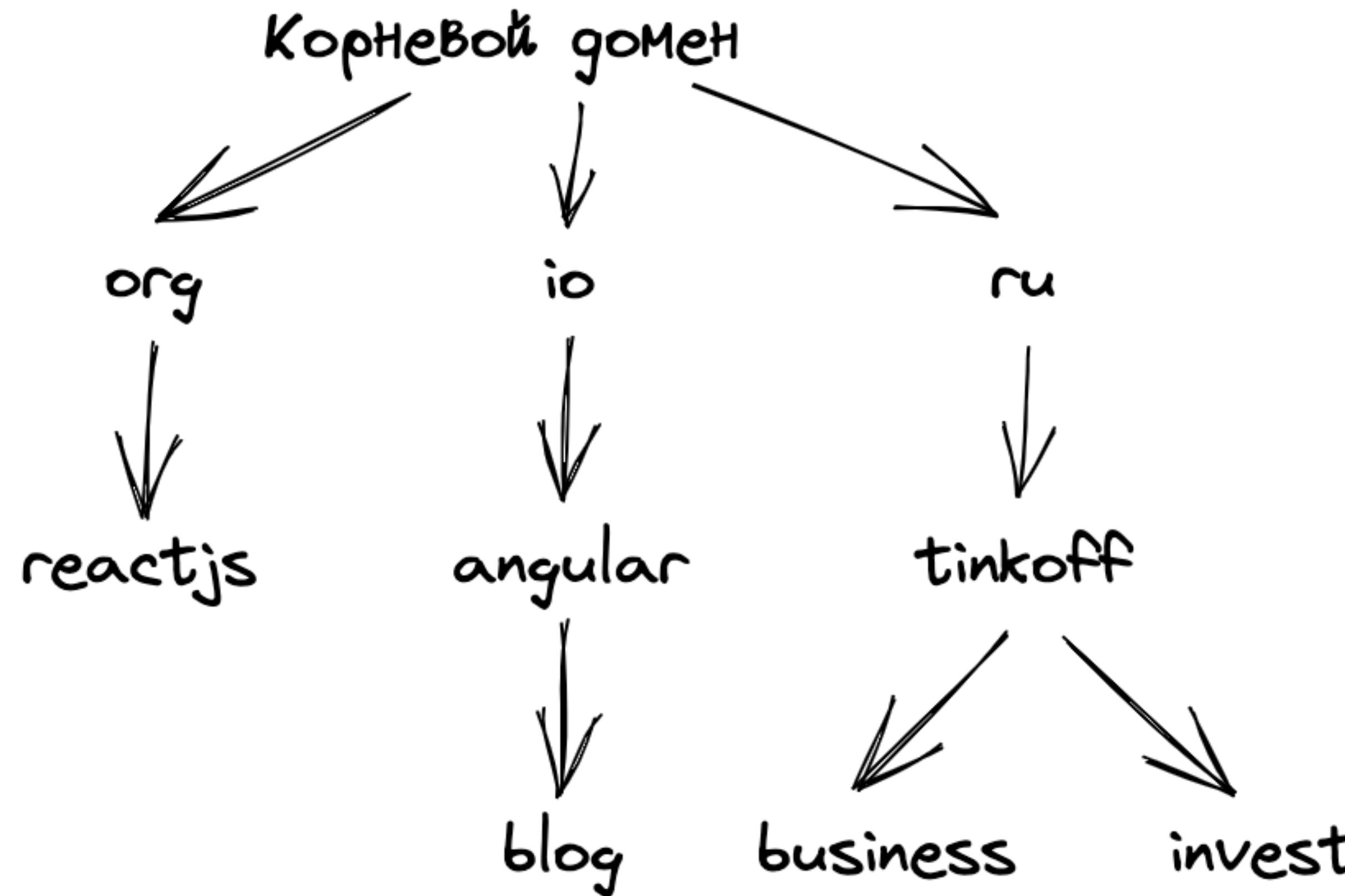
Windows – C:\windows\system32\drivers\etc\hosts

Работает по протоколу UDP на 53 порту

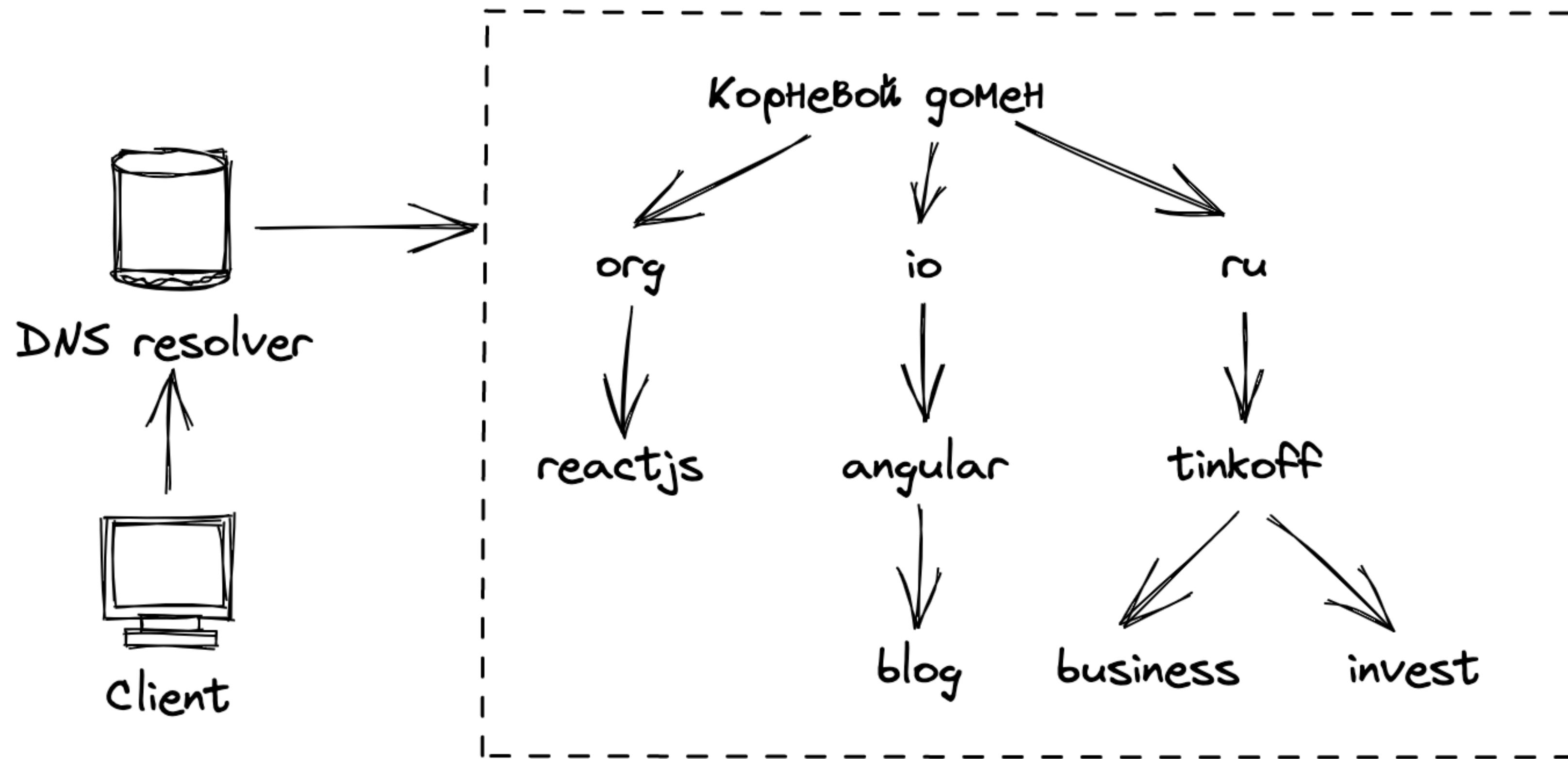
Структура домена



Иерархия доменных имен



Инфраструктура DNS



Протокол DNS



Регистрация домена

Корневой регистратор

ICAN (Internet Corporation
for Assigned Names and
Number)



Режимы работы

1. Итеративный
2. Рекурсивный



Типы DNS серверов

1. DNS resolver
2. DNS сервер зоны

Типы ответов DNS

1. Authoritative (заслуживающий доверия)
2. Non-Authoritative (не заслуживающий
доверия)

Открытые DNS сервера

- 8.8.8.8 – Google
- 1.1.1.1 – Cloudflare
- 77.88.8.88 – Yandex

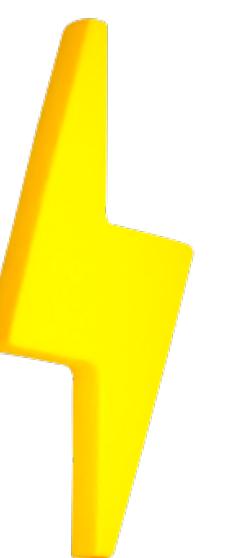
Задачи DNS

-  Определение IP адреса по доменному имени (ipv4 и ipv6)
-  Определение нескольких доменных имен на один ip адрес
-  Поиск адреса почтового сервера на домене
-  Определение ip адреса и порта некоторых сервисов (jabber)
-  Определение DNS сервера для нижестоящих зон (делегирование)
-  Определение доменного имени по IP адресу (reverse DNS)

Типа записей



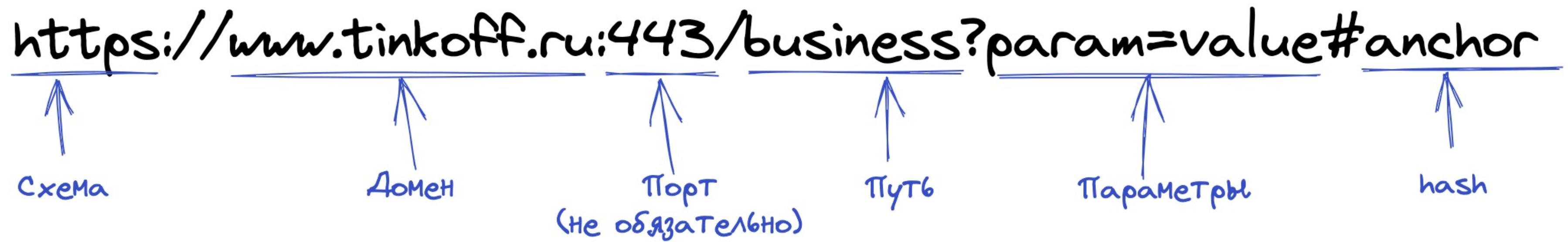
1. А запись – определение ipv4 адреса по доменному имени
2. AAAA – определение ipv6 адреса по доменному имени
3. CNAME – (Canonical name) определение дополнительного доменного имени для адреса
4. MX – (Mail eXchange) определение почтового сервера на доменном имени
5. SRV – (Service) запись сервиса
6. NS – (Name Server) делегирование зоны
7. PTR – (Pointer) обратный DNS
8. TXT – (Text) текстовая запись



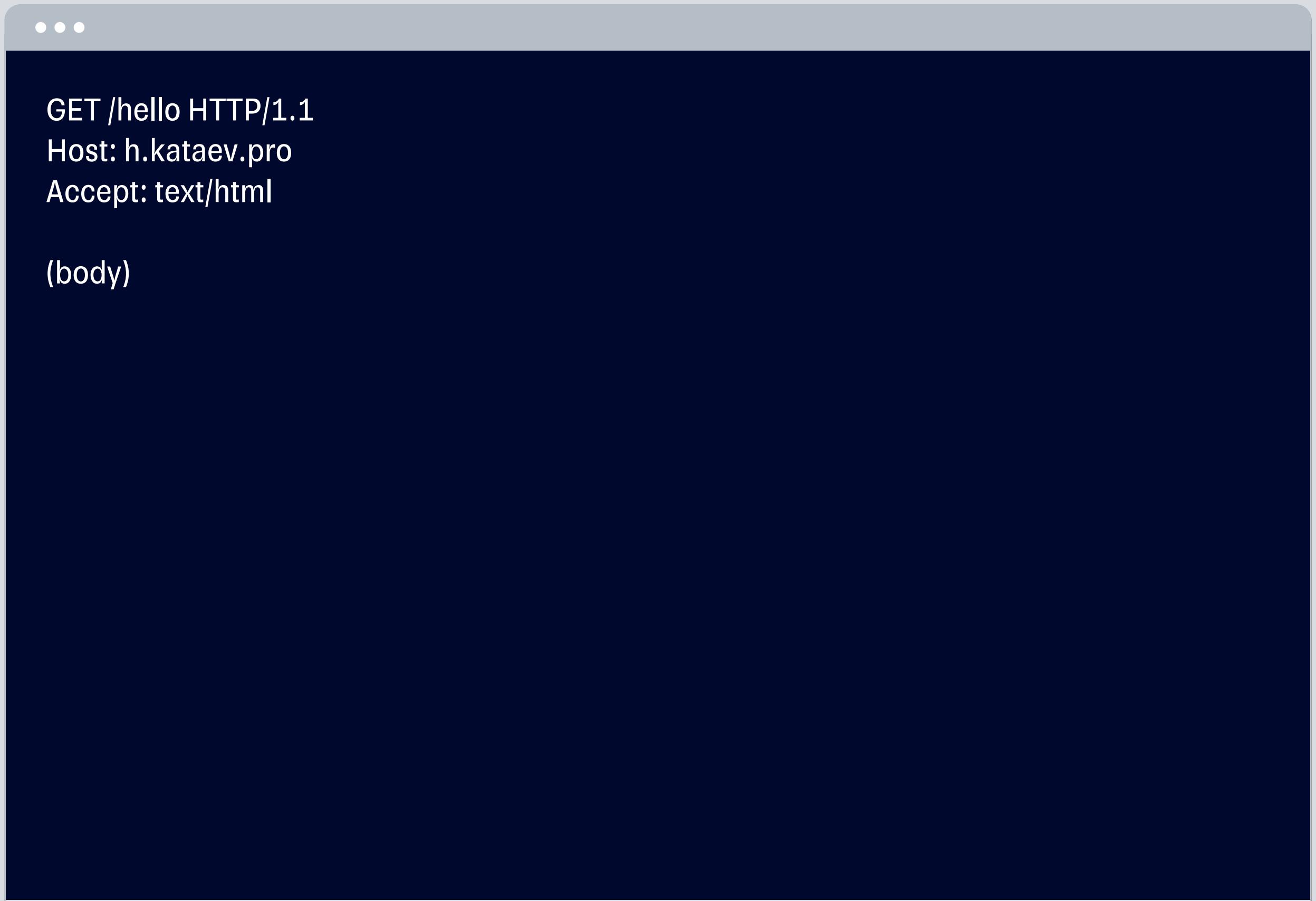
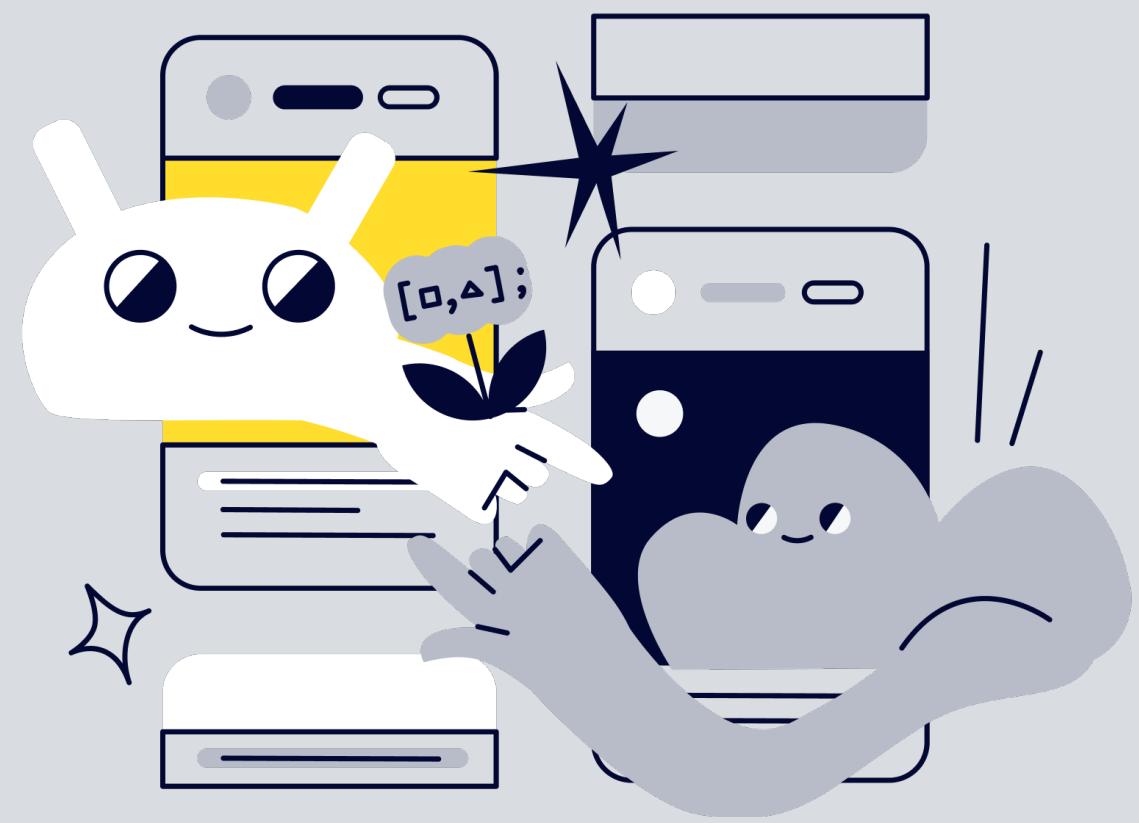
HTTP



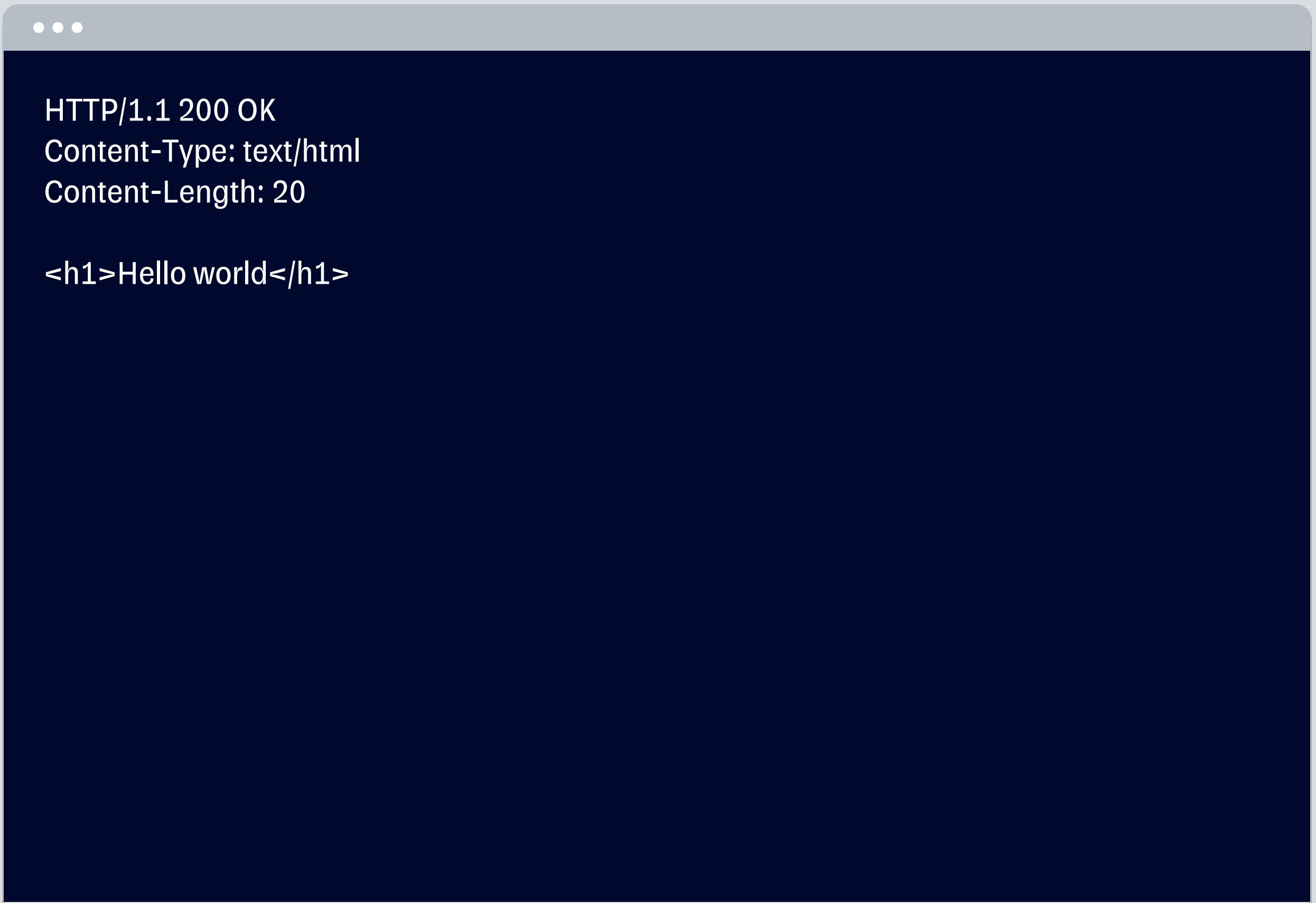
URI



Формат запроса



Формат ответа



Методы

GET/ HEAD

Получение данных от Сервера. Метод HEAD не содержит body, только заголовки

PUT

Создание или изменение ресурсов

POST

Передача данных, запуск процедур, загрузка файлов

PATCH

Аналогичен PUT, но в запросе используется фрагмент ресурса

OPTION

Определение возможностей веб сервера

DELETE

Удаление ресурса

Статус коды



1xx – Информационные



2xx – Успех!



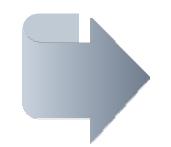
3xx – Ресурс перемещен



4xx – Ошибки клиента



5xx – Ошибки сервера



1xx



2xx



3xx

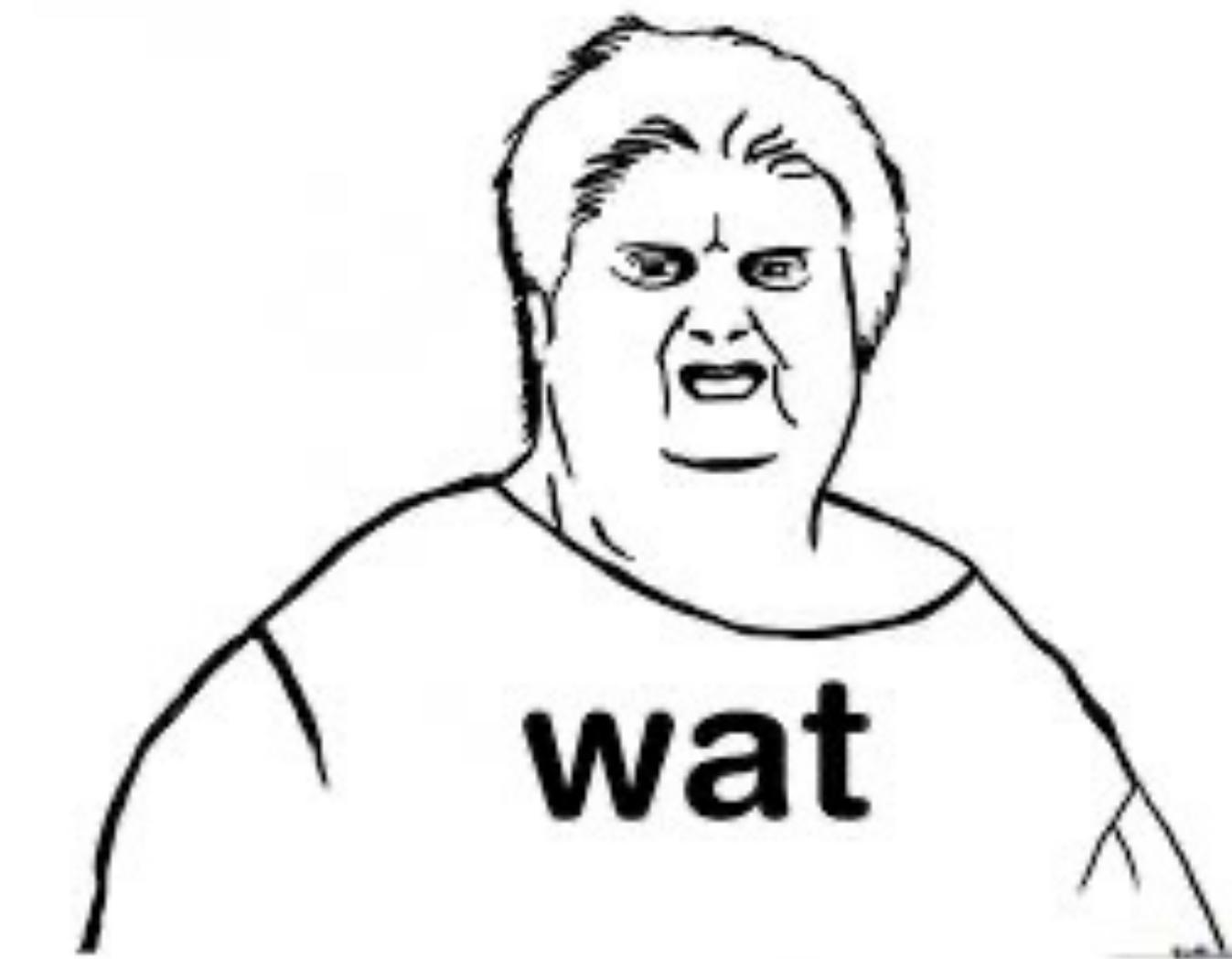


4xx



5xx

Информационные



Успех!

1xx

2xx

3xx

4xx

5xx

- 200 OK
- 201 Created
- 202 Accepted
- 206 Partial Content

Ресурс перемещен

1xx

2xx

3xx

4xx

5xx

- 301 Moved Permanently
- 302 Found
- 307 Temporary Redirect



Ошибки клиента

- 400 Bad requests
- 401 Unauthorized
- 402 Payment Required
- 403 Forbidden
- 404 Not found
- 405 Method not allowed
- 406 Not acceptable
- 408 Request timeout
- 411 Length Required
- 413 Payload Too Large
- 414 URI Too Long
- 417 I'm a Teapot

Ошибки Сервера

1xx

2xx

3xx

4xx

5xx

- 500 Internal Server Error
- 501 Not Implemented
- 502 Bad Gateway
- 503 Service Unavailable
- 504 Gateway Timeout

Основные заголовки



Клиент

Host
Accept
Authorization
Accept-Encoding
Accept-Charset
Accept-Language
User-Agent



Сервер

Content-Encoding
Content-Type
Content-Length
Content-Language

Авторизация\Аутентификация\Идентификация



01110110010

HTTP Basic Auth



Механизм



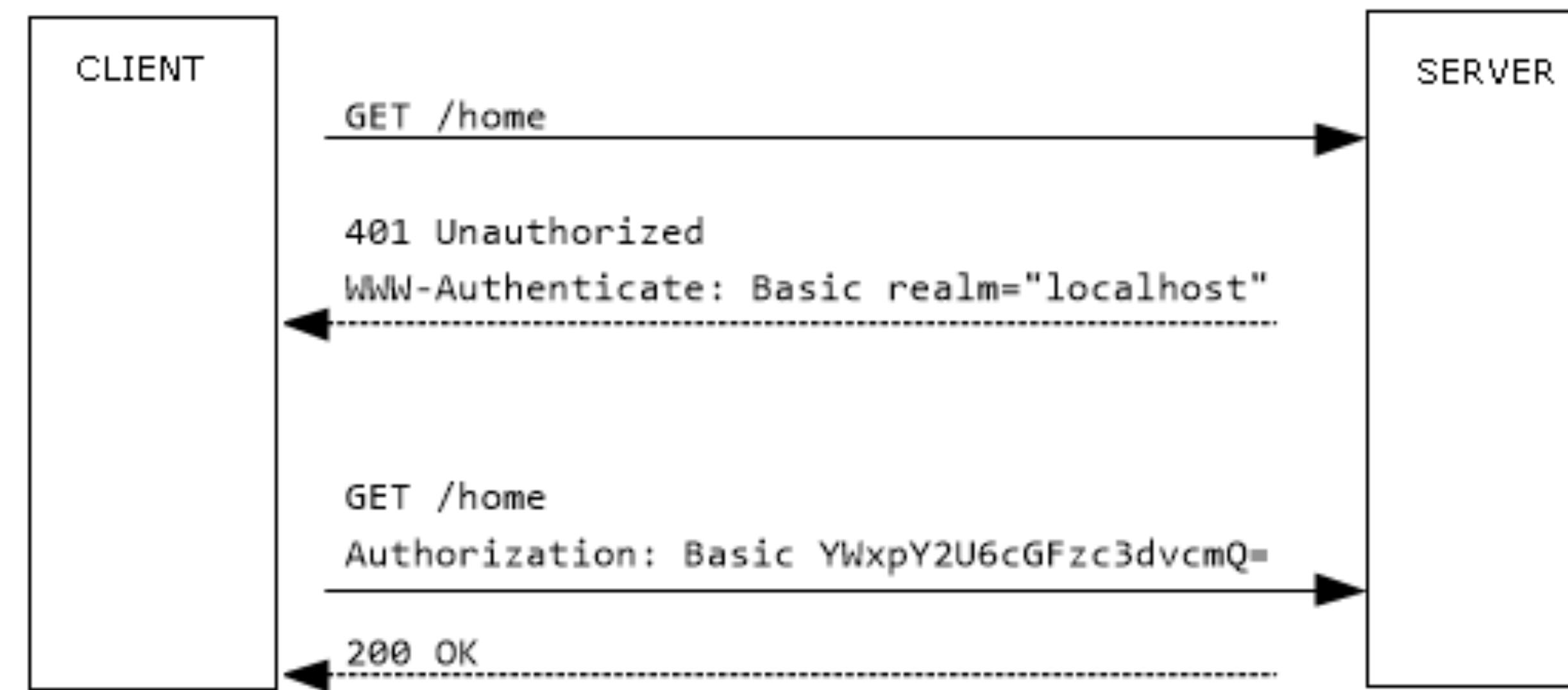
Сервер отвечает Клиенту кодом 401 и заголовком WWW-Authenticate



Клиент генерирует токен и повторяет запрос, передав токен в заголовок Authorization



Сервер проверяет токен и если он корректные отвечает на запрос



Генерация токена

- Конкatenируем логин и пароль через :
- vita:hello
- Переводим получившуюся строку в base64



```
const login = 'vita';
const password = 'hello';
const authData = login + ':' + password;
const token = window.btoa(authData); // 'dml0YTpoZlwxsbw=='
```

Cookie



Механизм работы



Сервер отправляет заголовок Set-Cookie с именем Куки и ее значением



Клиент сохраняет Куку



Все последующие запросы клиент в заголовке Cookie прикладывает сохраненную куку

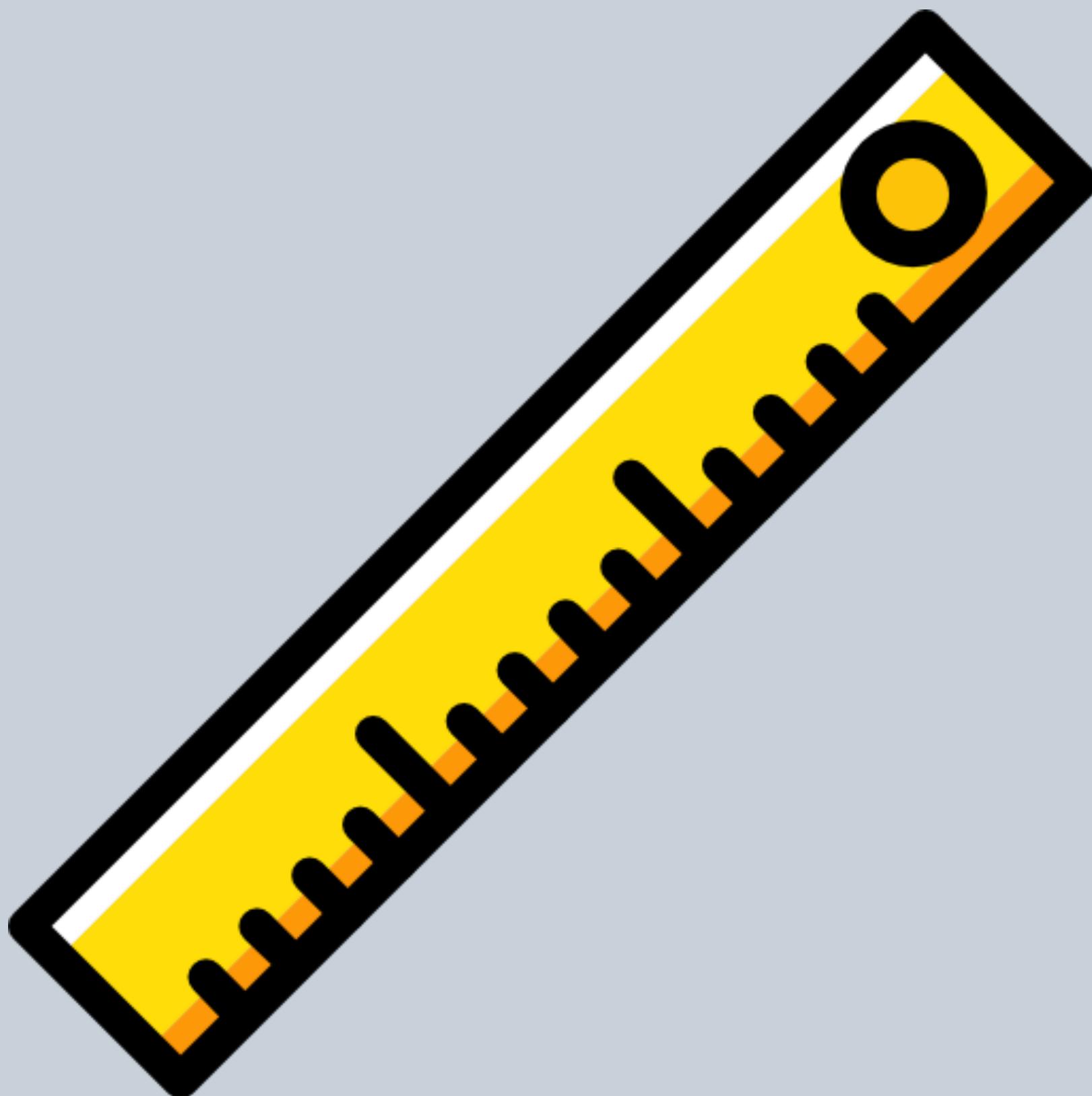


Дополнительные параметры

- expired и max-age
- path
- domain
- secure
- httponly
- samesite



Range



Механизм работы

Если сервер прислал заголовок Accept-Ranges, то сервер поддерживает частичную отправку данных. В значении будет единица измерения, в которых мы можем определить порцию данных. В большинстве случаев это bytes

```
HTTP/1.1 200 OK  
Accept-Ranges: bytes  
  
(body)
```



```
GET /img.png HTTP/1.1  
Range: bytes=0-3
```

Для частичного получения контента отправляем заголовок Range и в нем указываем единицу измерения и диапазон.

Механизм работы

В ответ сервер вернет частично запрошенные данные.

HTTP/1.1 206 Partial Content

Accept-Ranges: bytes

.PNG

Ещё больше примеров

Range: bytes=0-10

Range: bytes=10-

Range: bytes=0-10;15-20



WebSocket

Способы передачи данных от сервера клиенту

Pooling

Делаем http запрос раз в n секунд

Long pooling

Делаем http запрос, сервер на него не отвечает и как только у него появляются данные для отправки готовит ответ. После ответа клиент повторяет запрос

WebSocket

Дуплексное соединение между клиентом и сервером

Установка соединения

Клиент отправляет http запрос. В качестве протокола используется ws или wss для шифрованного. На 80 или 443 порт в случае с шифрованным.

```
GET /chat HTTP/1.1  
Host: chat.example.com  
Upgrade: websocket  
Sec-WebSocket-Key: <token>  
Sec-WebSocket-Version: 13
```



```
HTTP/1.1 101 Switching Protocols  
Upgrade: websocket  
Sec-WebSocket-Accept: <server-token>
```

После этого TCP соединение не разрывается как в случае в http, а остается активным. И дальше через это TCP соединение можно обмениваться данными без использования протокола HTTP

Итого

Поддержка существующей инфраструктуры Web (прокси, балансировки нагрузки, файрволлы и т.п.)

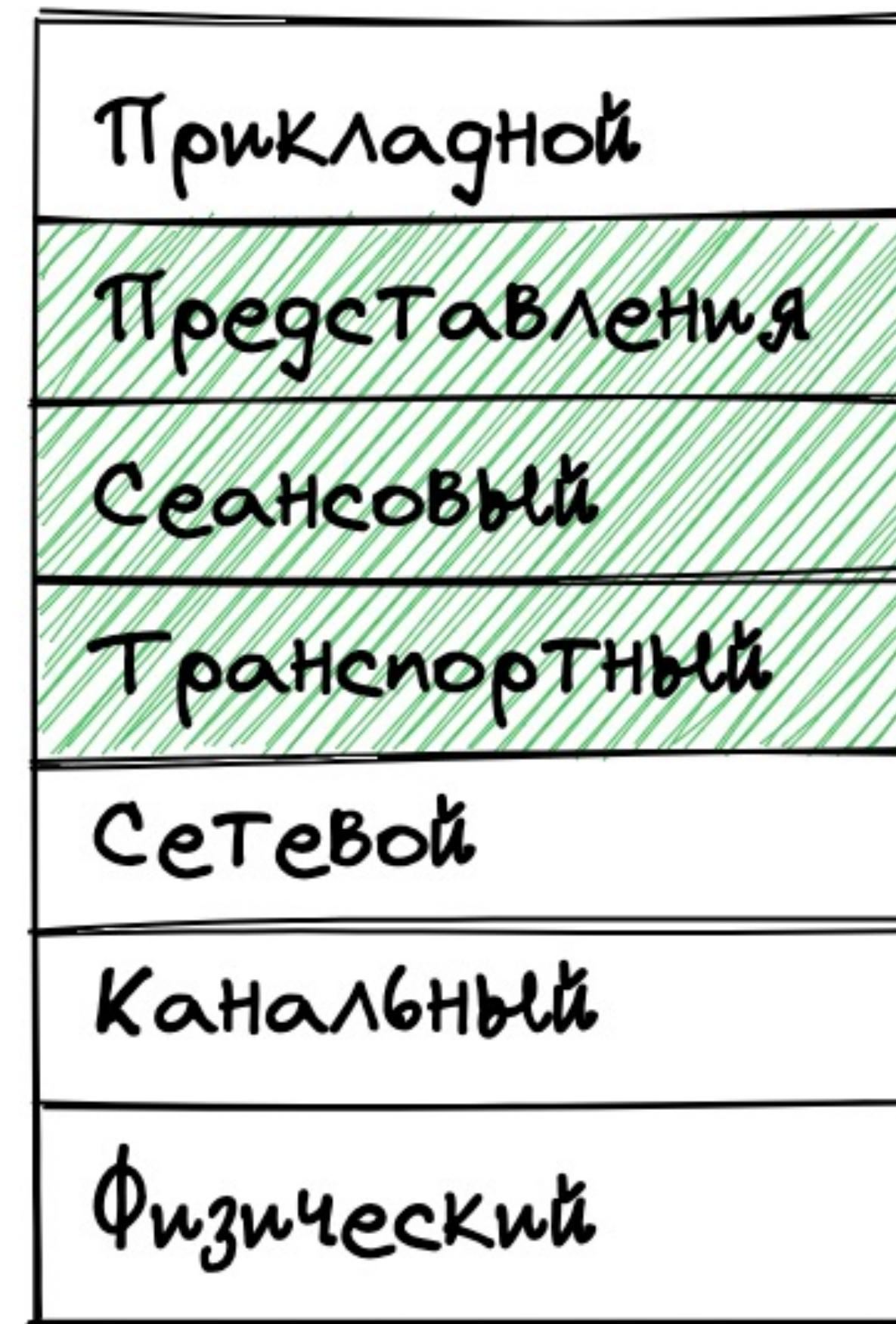
Высокая производительность за счет уменьшения накладных расходов

Нативно реализована во всех современных браузерах.

SSL/TLS



Место в модели ISO OSI



SSL/TLS

TLS - Transport Layer Security

SSL - Secure Sockets Layer

На данный момент считаются актуальными версии протоколов TLS 1.3 и 1.2. TLS 1.0 и 1.1 равно как и SSL не используется

Аспекты безопасности



Privacy



Integrity



Authentication

Шифрование



Симметричное

Используется один ключ как для шифрования, так и для расшифровки данных

Алгоритмы: AES, 3DES

Асимметричное

Используется 2 ключа: открытый и закрытый. Открытым шифруют данные, закрытым расшифровывают

Алгоритмы: RSA, DSA, DSS, Diffie-Hellman

Стек алгоритмов в TLS/SSL

1. Алгоритмы обмена ключами

- RSA
- Diffie–Hellman

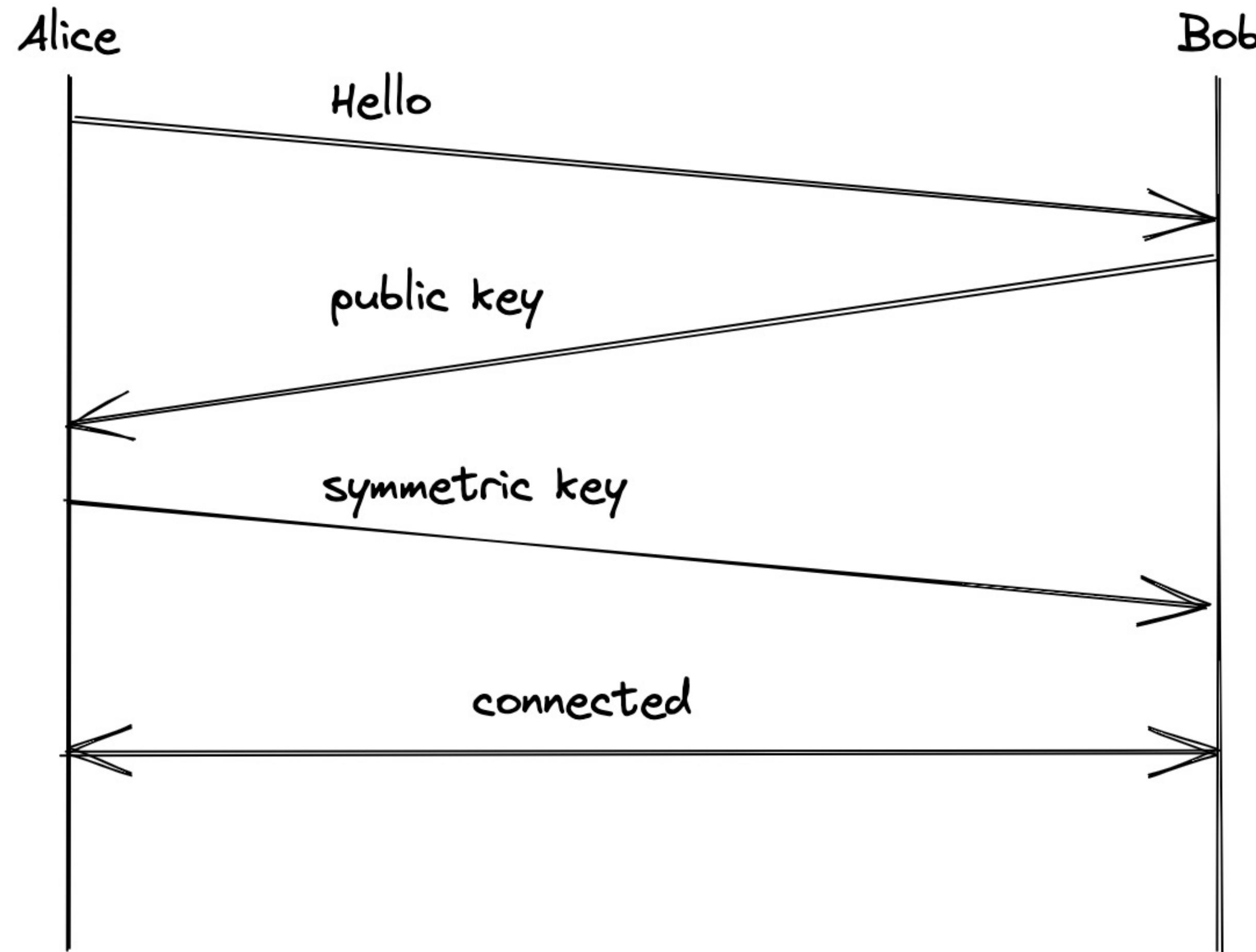
2. Алгоритмы симметричного шифрования

- AES
- 3DES

3. Хэш функции

- MD5
- SHA-1, SHA-224, SHA-256, SHA-384, SHA-512

Создание сеанса связи (Handshake)



Handshake Diffie–Hellman

1. Клиент и сервер договариваются о 2 числах p и g , которые можно передать в открытом виде, допустим $p = 29$, $g=3$
2. Клиент и сервер генерируют 2 числа, каждый свое. Например клиент генерирует число 5 , а сервер число 8 . Эти числа нельзя передавать по открытым каналам связи.
3. Клиент и сервер считает $g^r \ mod \ p$
 - Клиента - $3^5 \mod 29 = 11$
 - Сервера - $3^8 \mod 29 = 7$
4. Клиент и сервер обмениваются этими ключами и рассчитывают новый ключ
 - Клиент - $7^5 \mod 29 = 16$
 - Сервер - $11^8 \mod 29 = 16$

Handshake Diffie–Hellman

Условия работы алгоритма

1. p - большое простое число, минимум 1024 бита
2. g - первообразный корень по модулю p , небольшое

Преимущества

1. При достаточно большом p подбор ключа займет тысячи лет на современных суперкомпьютерах
2. Обеспечивает совершенную прямую секретность

Integrity

Алгоритмы

1. MD5
2. SHA-1,SHA-224,SHA-256,SHA-384,SHA-512(число-это длина результирующего хэша в битах)

Message Authentication Code (MAC)

При установки соединения клиент и сервер договариваются о **Разделяемом ключе**, который будет использоваться для подписания.

$$\text{MAC} = f(\text{data}, \text{key})$$

Authentication

Решает задачу удостоверения, что сервер тот, за кого себя выдает

Для решения этого проблемы используется **Подпись и Инфраструктура открытых ключей**

Подпись

1. Когда сервер отвечает клиенту он рассчитывает хэш от данных
2. Хэш шифруется закрытым ключом сервера и этот шифр прикладывается к сообщению
3. Клиент извлекает шифр и расшифровывает его открытым ключом сервера
4. Клиент рассчитывает хэш от данных
5. Сравнивает два хэша (расшифрованный от сервера и рассчитанный).
Если хэши совпадают, то можно гарантировать, что сообщение пришло от владельца публичного ключа

Инфраструктура открытых ключей

Предполагается, что в сети есть множество узлов, которые не доверяют друг другу, но все они доверяют третьему, не заинтересованному лицу, которого называют **Удостоверяющий центр (УЦ)**

УЦ выдает серверу сертификат в формате X509, который содержит публичный ключ сервера

Сертификат подписан приватным ключом УЦ

На практике используется иерархия УЦ, где есть корневые УЦ и делегированные им УЦ

Клиент должен проверить подписи всей цепочки в иерархии УЦ

Список корневых УЦ и их публичные ключи хранятся в хранилище сертификатов ОС

Само подpisанный сертификат

Применение:

1. Организации шифрованных соединений в закрытых сетях
2. Для задач отладки на localhost

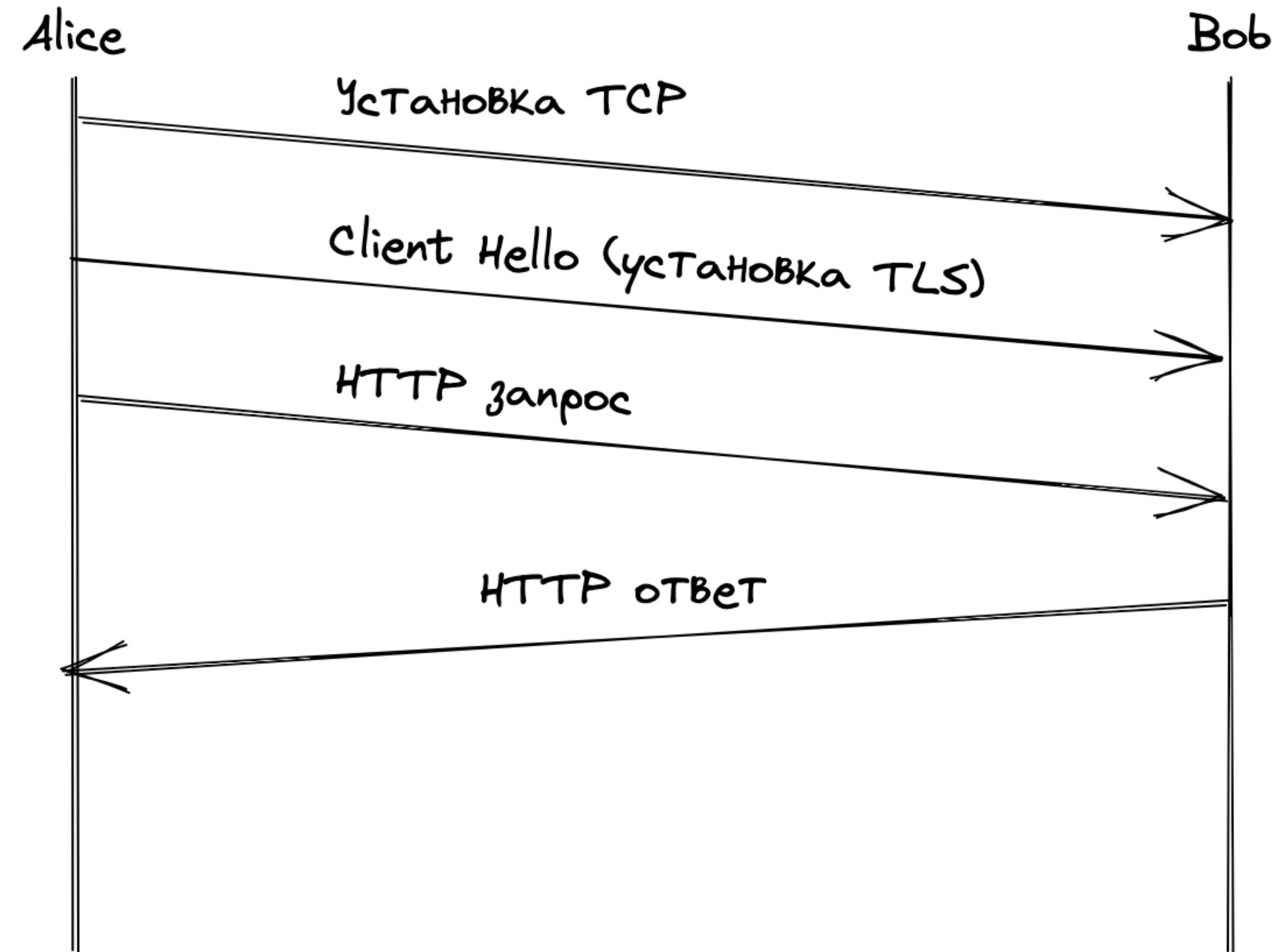
Само подpisанный сертификат можно создать самостоятельно, добавить в хранилище сертификатов ОС и сделать его доверенным

Важно: в целях безопасности не устанавливайте сертификаты из непроверенных источников!



HTTPS

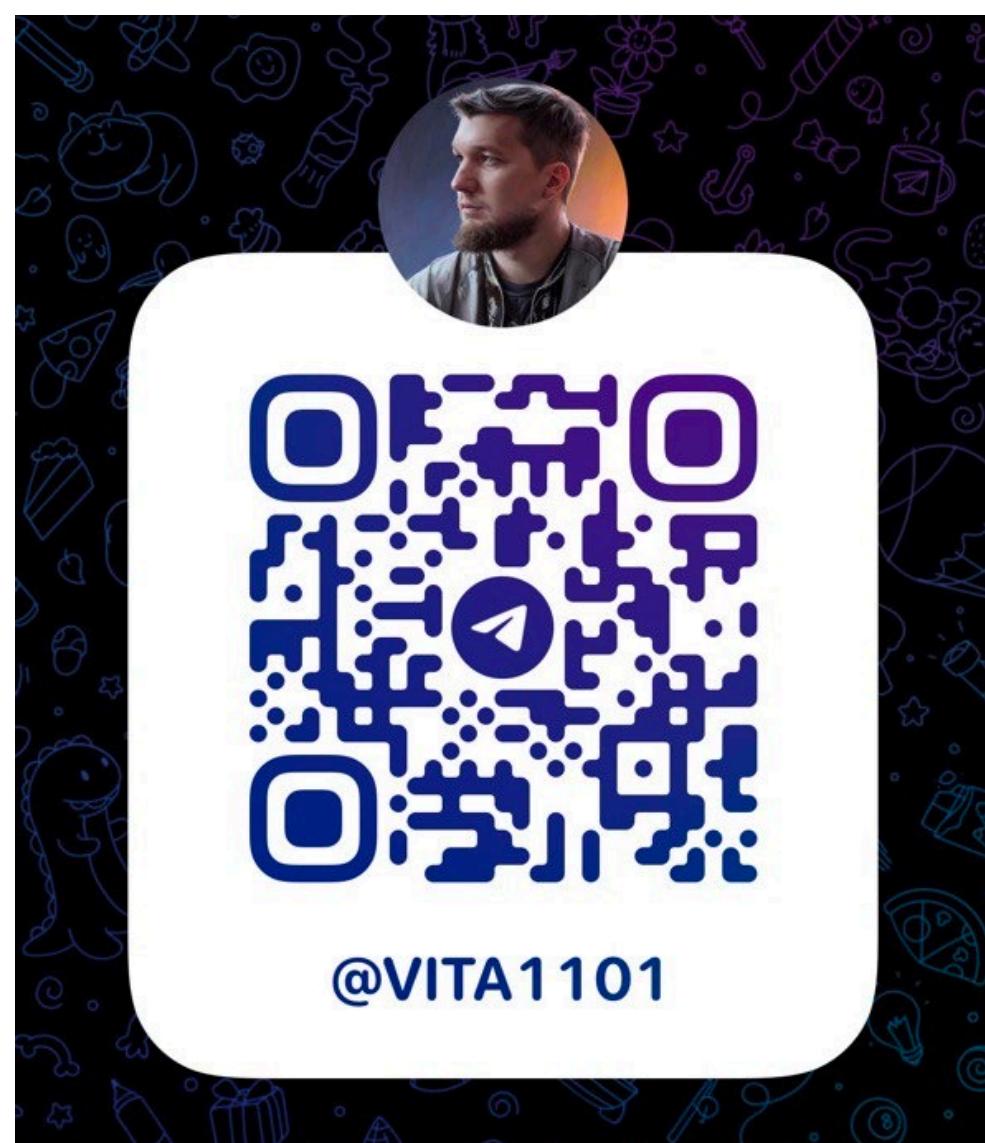
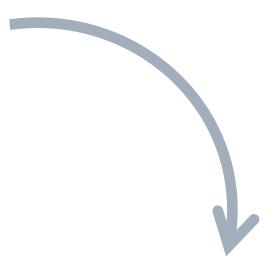
HTTPS



Катаев Виталий

Сеньор Формошлеп

Мне можно
написать



vita@kataev.pro

