

Websocket - когда, как и сколько стоит

Лавриненко Александр,
ManyChat

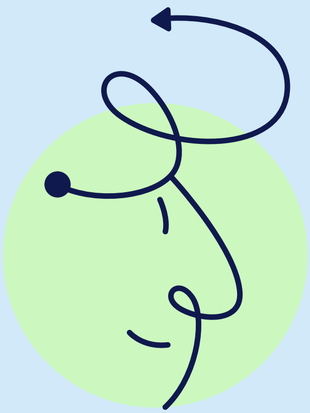
Пару фактов обо мне

6 лет в мобилке

Несколько стартапов

Сбербанк онлайн

ManyChat и погружение в Websocket



ManyChat

Messenger Marketing (Facebook, SMS, Email)

Бизнес хочет ответить клиенту максимально быстро

Мобильное приложение поддержки LiveChat

Более 1 000 000 клиентов по миру



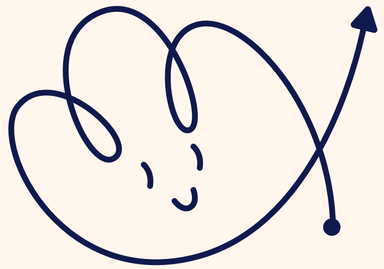
Websocket - это круто

Двунаправленный

Одно соединение (в идеале)

Маленький размер «хедеров» сообщения (+ 2-14 байт)

Передает бинарные и строковые данные



Websocket vs HTTP

Критерии	Websocket	HTTP 1.1	HTTP 2.0
Обмен данных	Двунаправленный	Однонаправленный	Однонаправленный, несколько ответов
Соединение	Одно	Несколько *	Несколько *
Размер сообщения	15 байт	500-800 байт	От 10 до 100 (при кэшировании)
Тип данных	Бинарные и строковые	Бинарные и строковые	Бинарные
Гарантия доставки	Нет	Есть	Есть

* Keep alive позволяет переиспользовать одно соединение в рамках таймаута

«Двунаправленный» HTTP

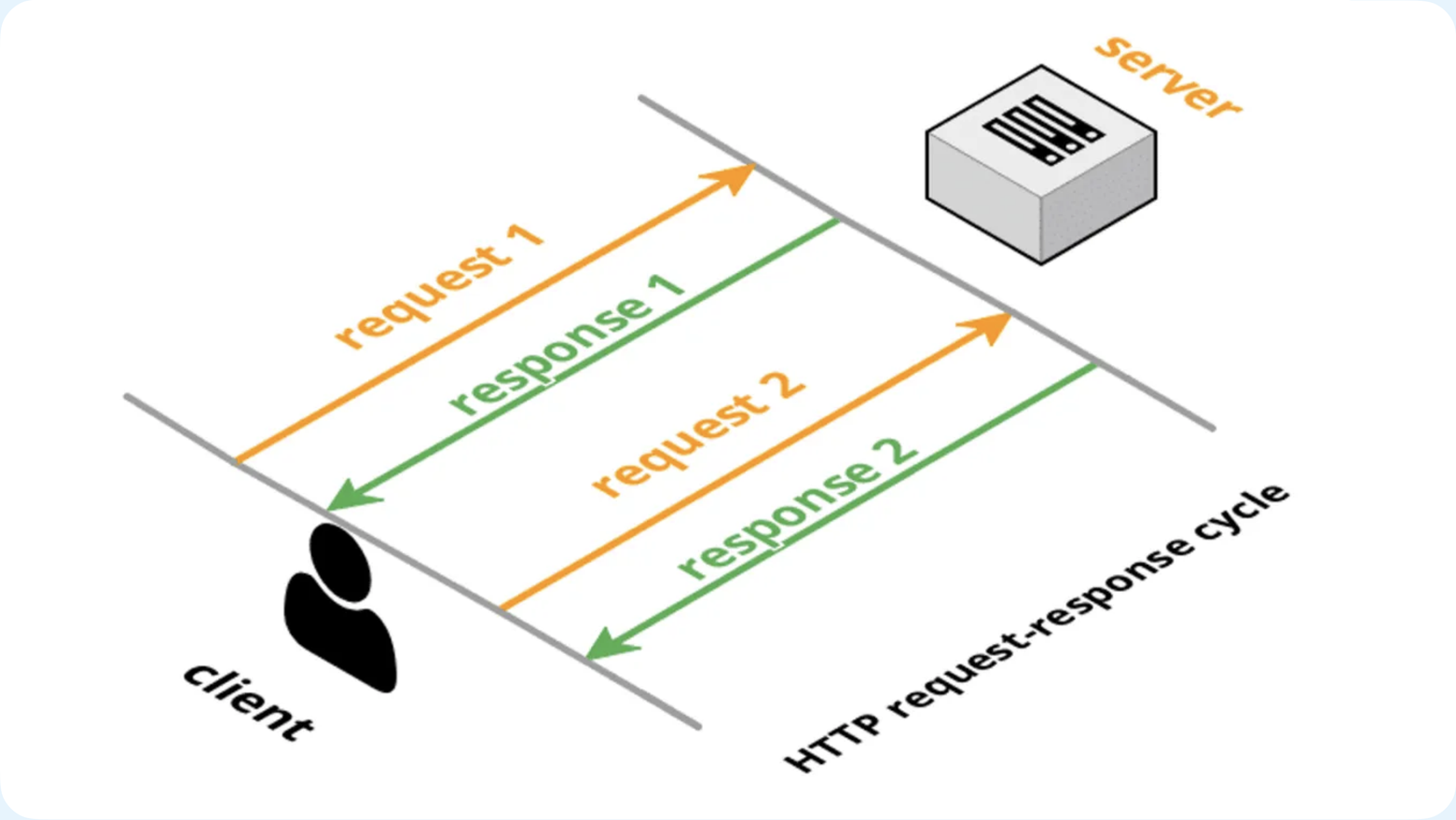
Polling



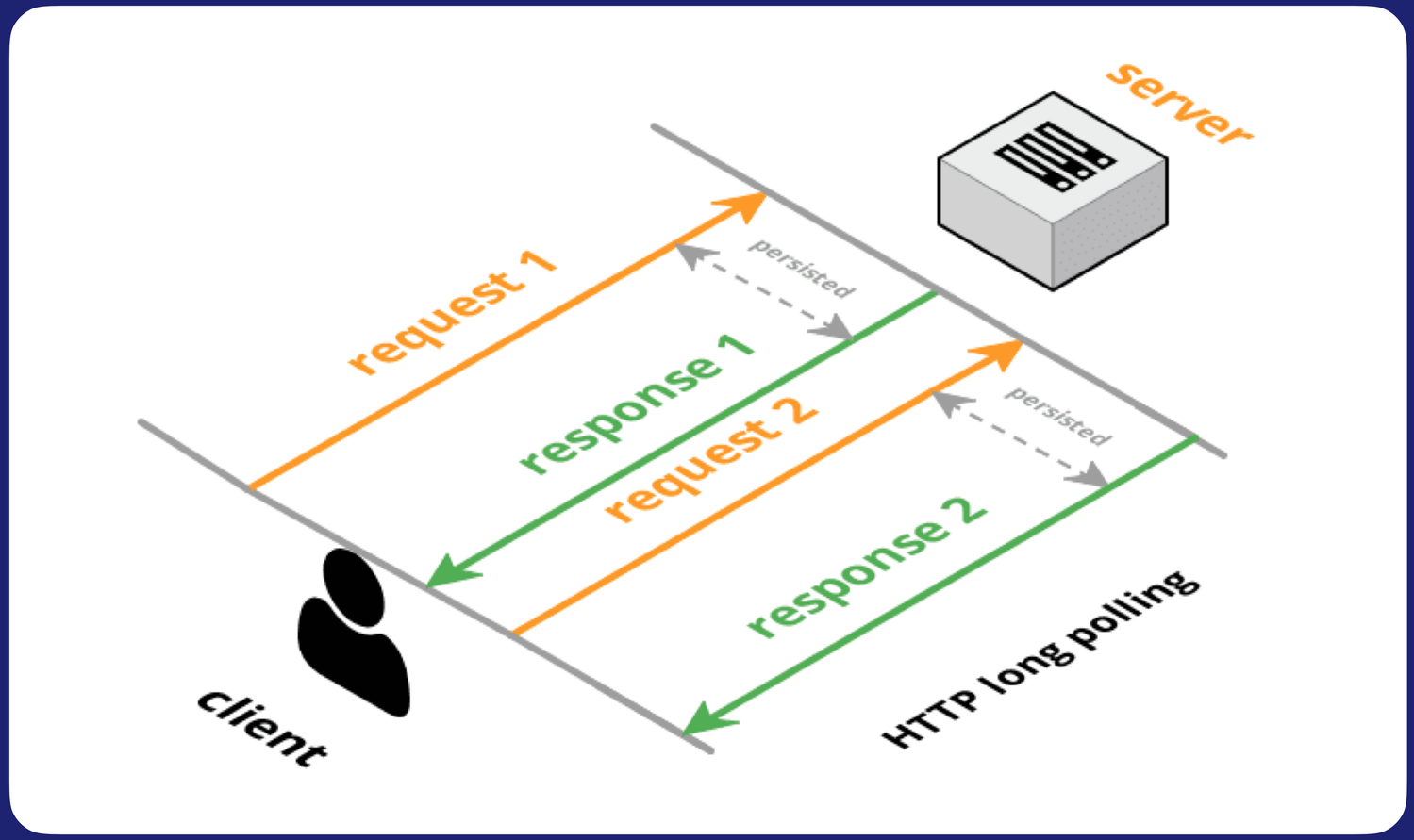
Long-polling



Polling


server
request 1
response 1
request 2
response 2
client
HTTP request-response cycle

Long-Polling



Websocket vs HTTP

Критерии	Websocket	HTTP 1.1	HTTP 2.0
Обмен данных	Двунаправленный	Однонаправленный	Однонаправленный, несколько ответов
Соединение	Одно	Несколько *	Несколько *
Размер сообщения	15 байт	500-800 байт	От 10 до 100 (при кэшировании)
Тип данных	Бинарные и строковые	Бинарные и строковые	Бинарные
Гарантия доставки	Нет	Есть	Есть

* Keep alive позволяет переиспользовать одно соединение в рамках таймаута

Websocket vs HTTP

Критерии	Websocket	HTTP 1.1	HTTP 2.0
Обмен данных	Двунаправленный	Однонаправленный	Однонаправленный, несколько ответов
Соединение	Одно	Несколько *	Несколько *
Размер сообщения	15 байт	500-800 байт	От 10 до 100 (при кэшировании)
Тип данных	Бинарные и строковые	Бинарные и строковые	Бинарные
Гарантия доставки	Нет	Есть	Есть

* Keep alive позволяет переиспользовать одно соединение в рамках таймаута

Размер Header HTTP 1.1

GET / HTTP/1.1‹CR›‹LF›

Host:x‹CR›‹LF›

‹CR›‹LF›



26 байт

Размер Header HTTP 1.1



500-800 байт

Размер Header HTTP 2.0



10-100 байт

Использует таблицу Header’ов на бекенде.

Клиент ссылается на элемент в таблице, а не шлет заголовок.

Почему размер это плохо?

Скорость передачи

Ресурсы

Скорость обработки



Websocket vs HTTP

Критерии	Websocket	HTTP 1.1	HTTP 2.0
Обмен данных	Двунаправленный	Однонаправленный	Однонаправленный, несколько ответов
Соединение	Одно	Несколько *	Несколько *
Размер сообщения	15 байт	500-800 байт	От 10 до 100 (при кэшировании)
Тип данных	Бинарные и строковые	Бинарные и строковые	Бинарные
Гарантия доставки	Нет	Есть	Есть

* Keep alive позволяет переиспользовать одно соединение в рамках таймаута

Websocket vs HTTP

Критерии	Websocket	HTTP 1.1	HTTP 2.0
Обмен данных	Двунаправленный	Однонаправленный	Однонаправленный, несколько ответов
Соединение	Одно	Несколько *	Несколько *
Размер сообщения	15 байт	500-800 байт	От 10 до 100 (при кэшировании)
Тип данных	Бинарные и строковые	Бинарные и строковые	Бинарные
Гарантия доставки	Нет	Есть	Есть

* Keep alive позволяет переиспользовать одно соединение в рамках таймаута

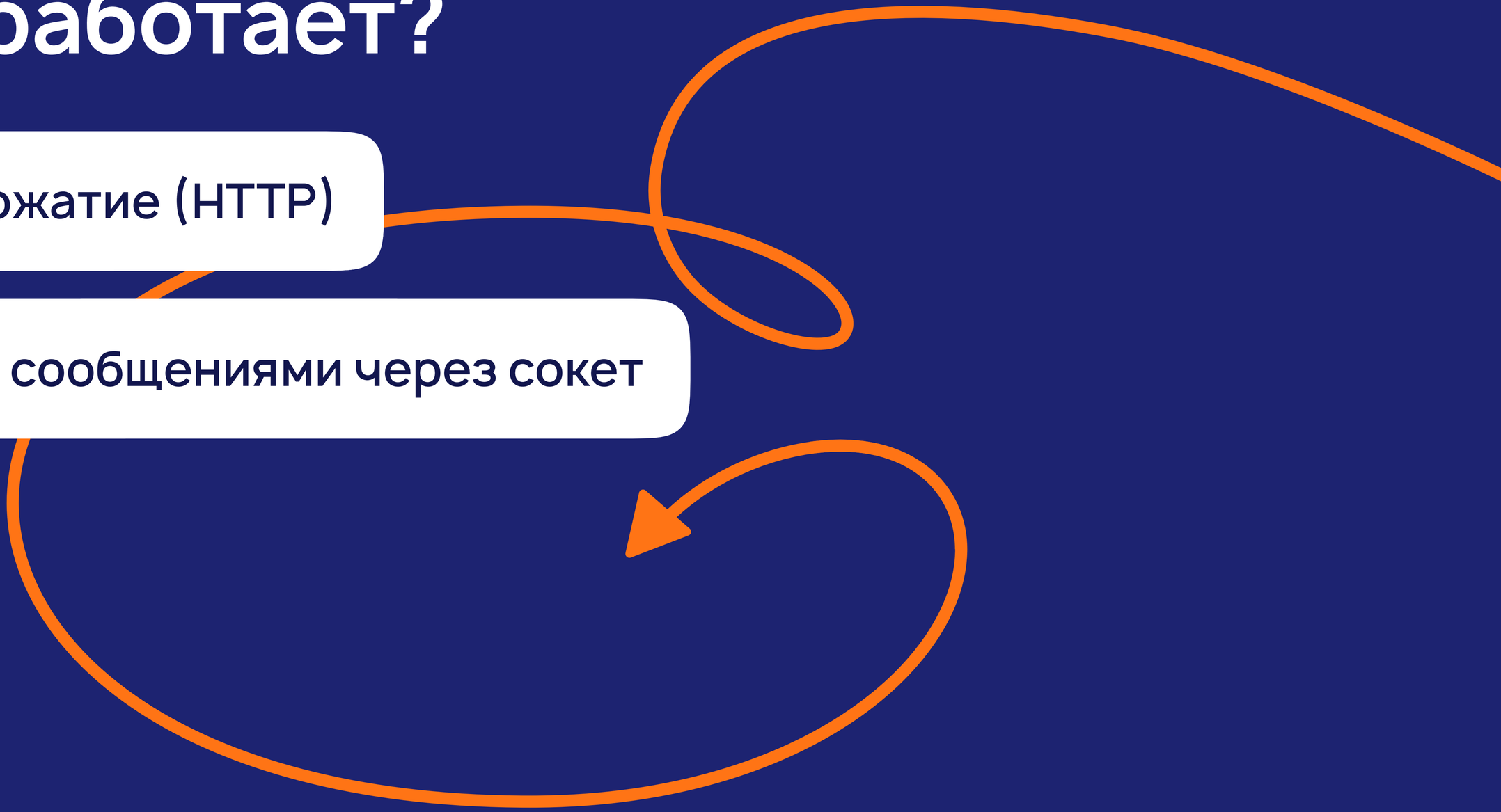
Основные различия Websocket и HTTP

- Количество соединений
- Размер сообщения
- Гарантия доставки
- Обмен данных

Как работает?

Рукопожатие (HTTP)

Обмен сообщениями через сокет



Рукопожатие



Клиенты

GET /chat

Host: javascript.info
Upgrade: websocket
Connection: Upgrade
Origin: https://javascript.info
Sec-WebSocket-Key: Iv8io/9s+lYFgZWcXczP8Q==
Sec-WebSocket-Version: 13
Sec-WebSocket-Extensions: deflate-frame
Sec-WebSocket-Protocol: soap, wamp

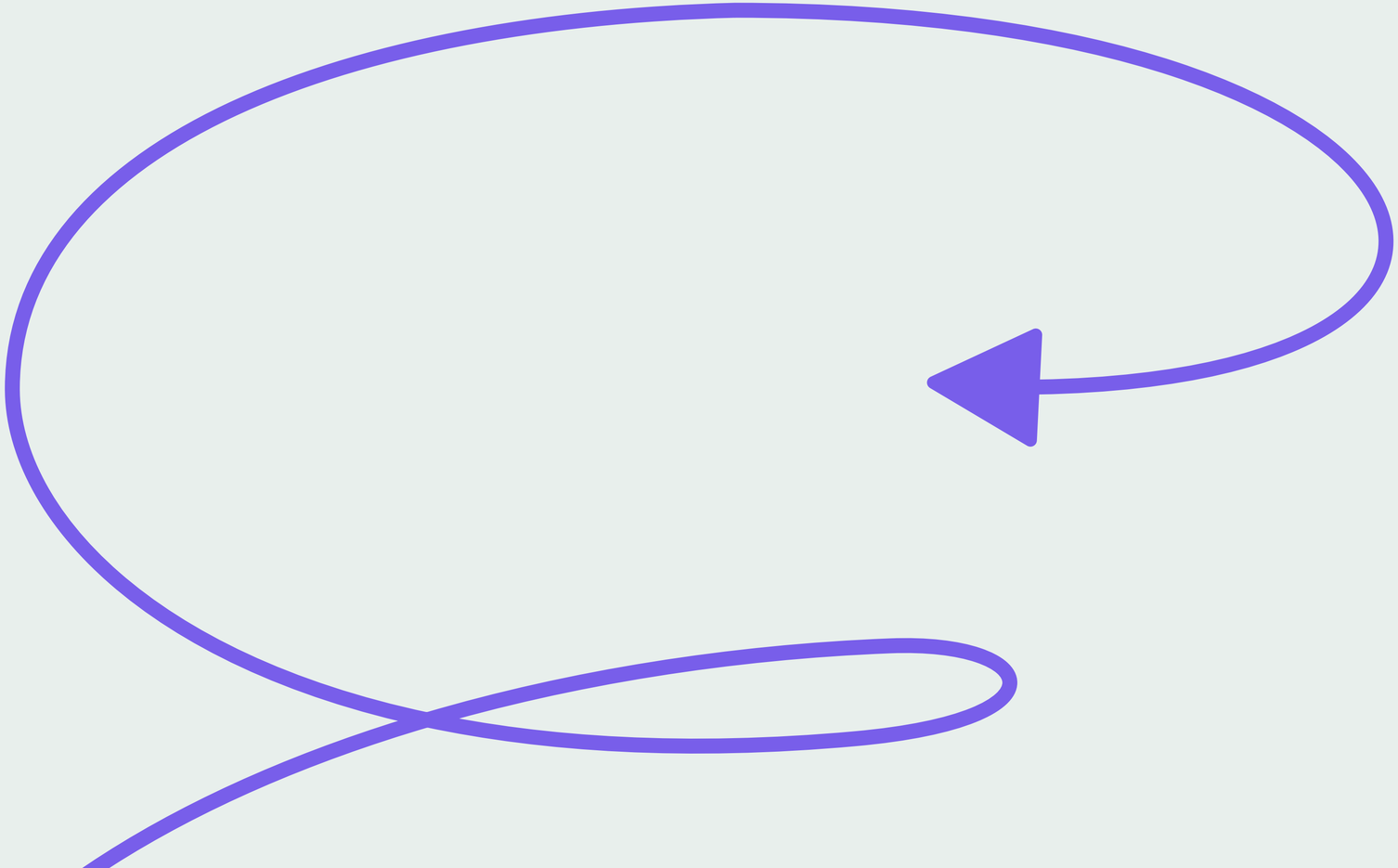
101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: hsBlbuDTkk24srzEOTBUlZAlC2g=
Sec-WebSocket-Extensions: deflate-frame
Sec-WebSocket-Protocol: soap



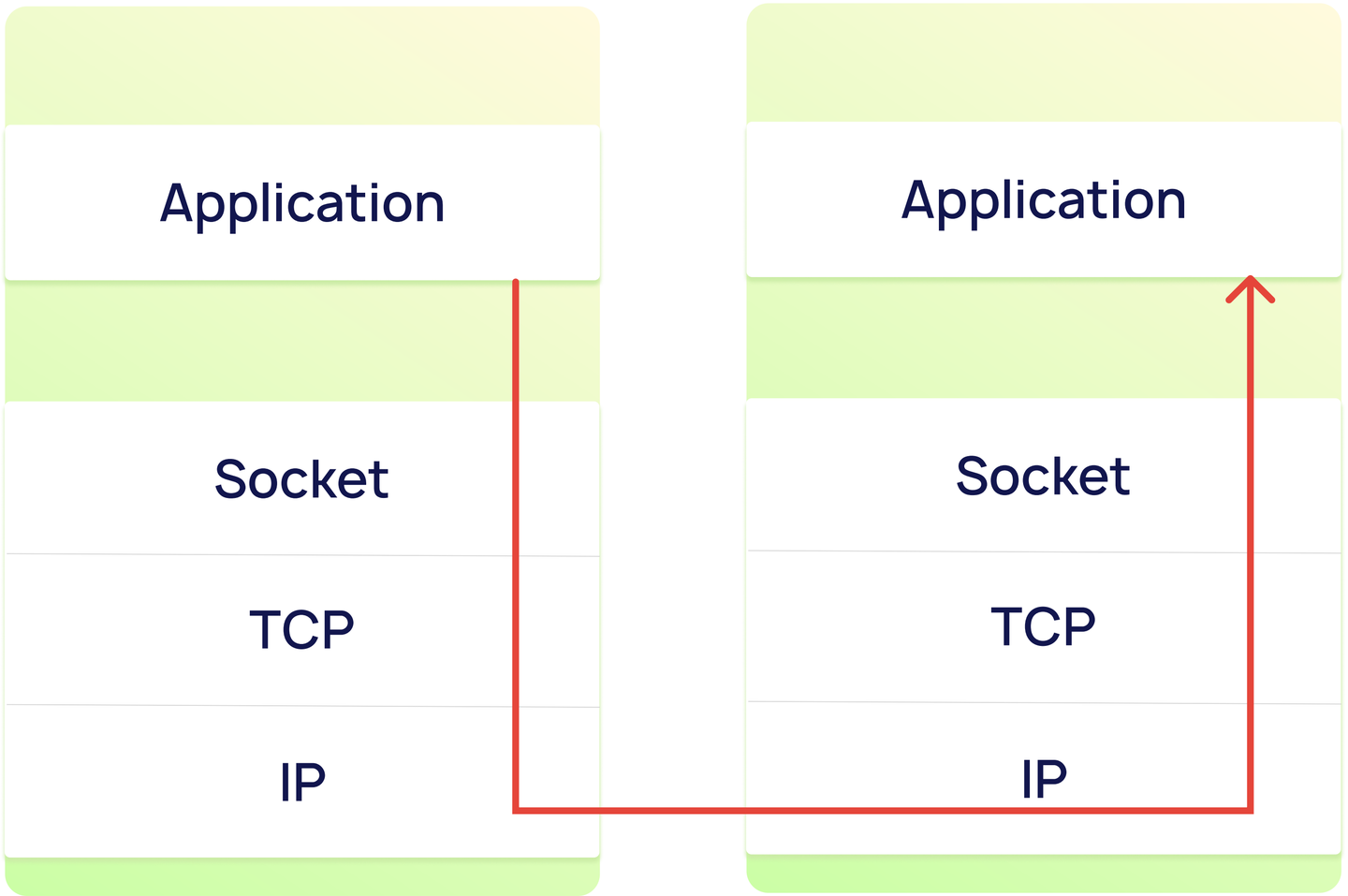
Server

Передача сообщений через сокеты

Что такое сокеты?



TCP/IP



TCP

Надежная доставка

Последовательное получение

Сегментация



Сокет

Socket = IP: Port



Что добавил Websocket

Концепция сообщений

Свой пакет/фрейм

Несколько типов сообщений

Extensions

Subprotocols



Концепция «сообщений»

	TCP отправляет 8 байт	Websocket - 1 сообщение размером 8 байт
Пакет 1	6 байт	6 байт
Пакет 2	2 байта	2 байта
Всего сообщений:	?	1

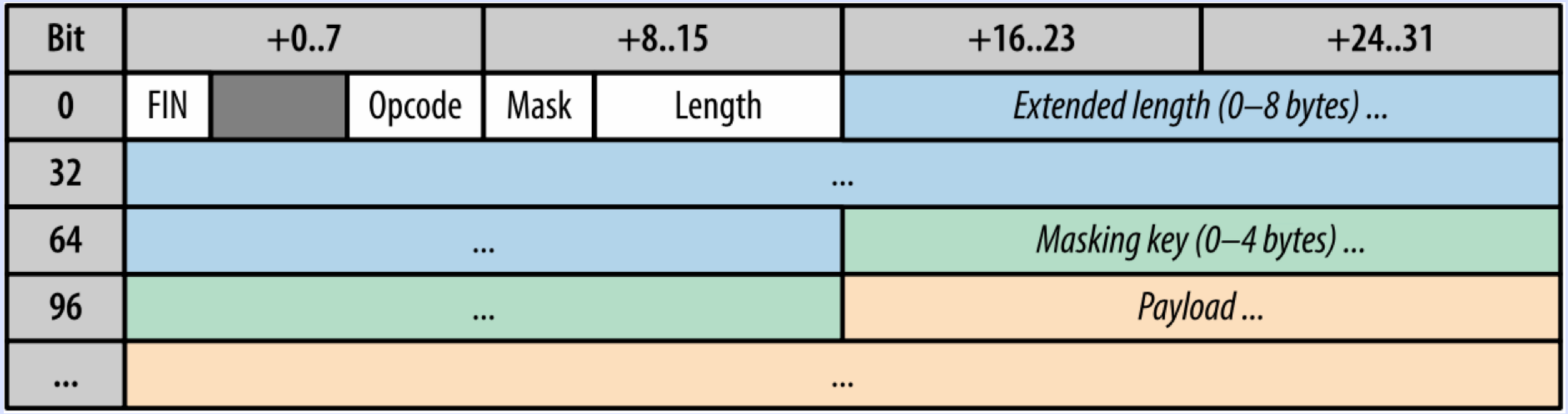
Концепция «сообщений»

	TCP отправляет 8 байт	Websocket - 1 сообщение размером 8 байт
Пакет 1	6 байт	6 байт
Пакет 2	2 байта	2 байта
Всего сообщений:	?	1

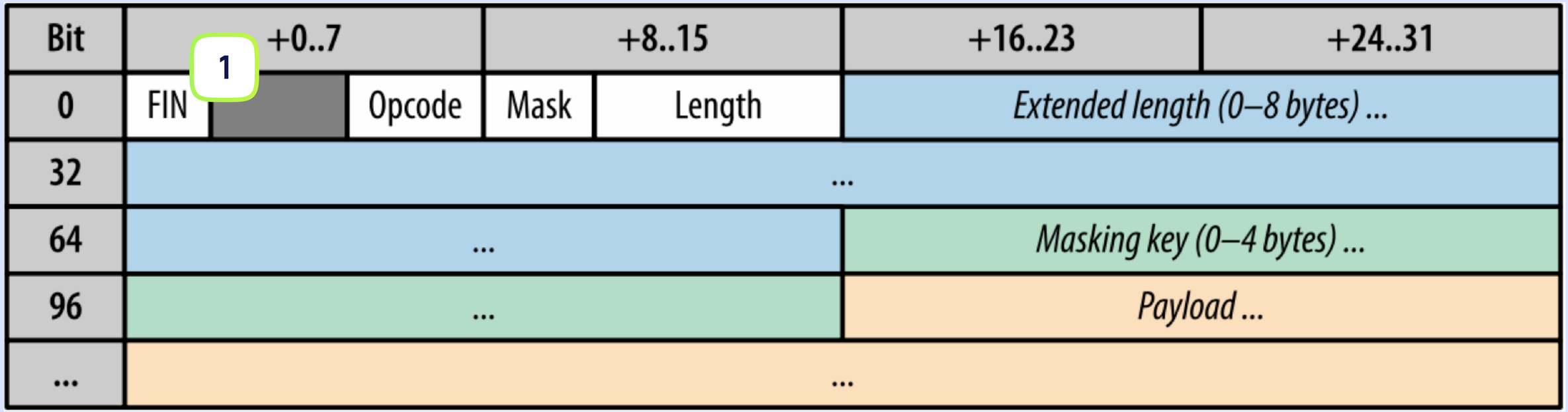
Концепция «сообщений»

	TCP отправляет 8 байт	Websocket - 1 сообщение размером 8 байт
Пакет 1	6 байт	6 байт
Пакет 2	2 байта	2 байта
Всего сообщений:	?	1

Websocket frame

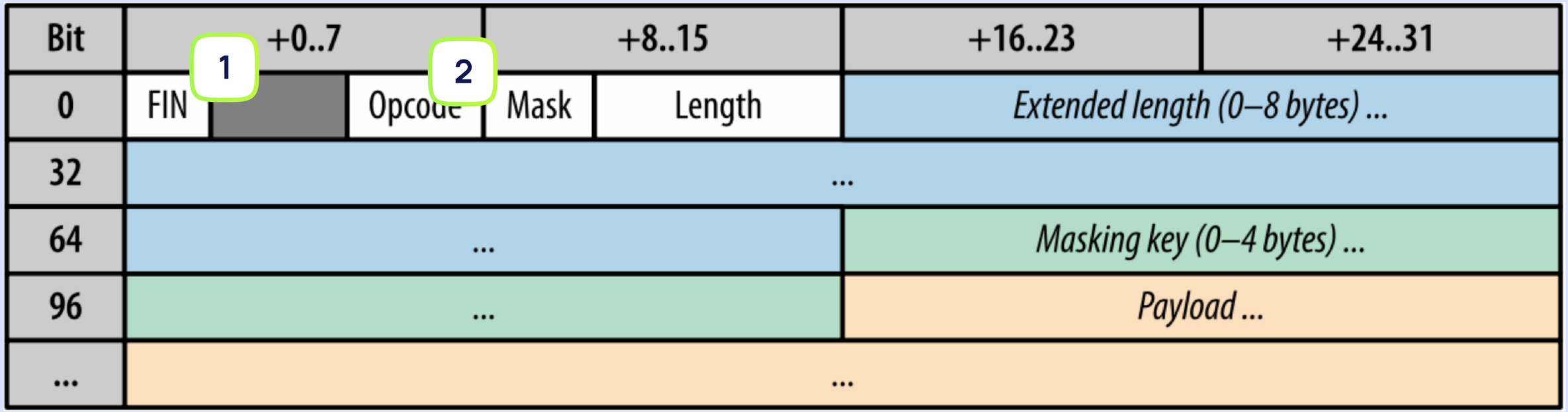


Websocket frame



1. FIN - индикатор последней части сообщения

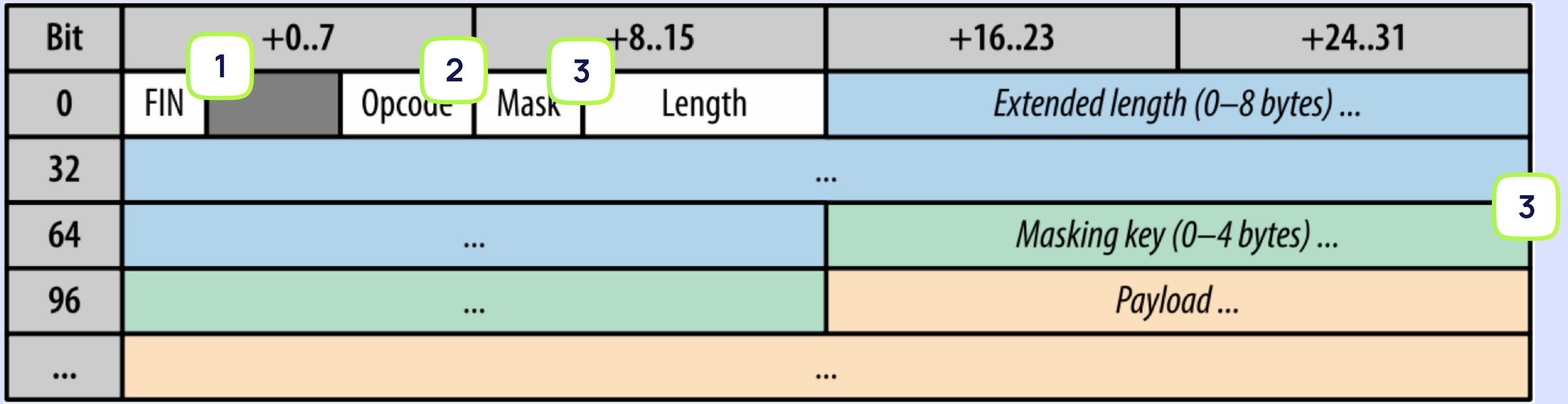
Websocket frame



1. FIN - индикатор последней части сообщения

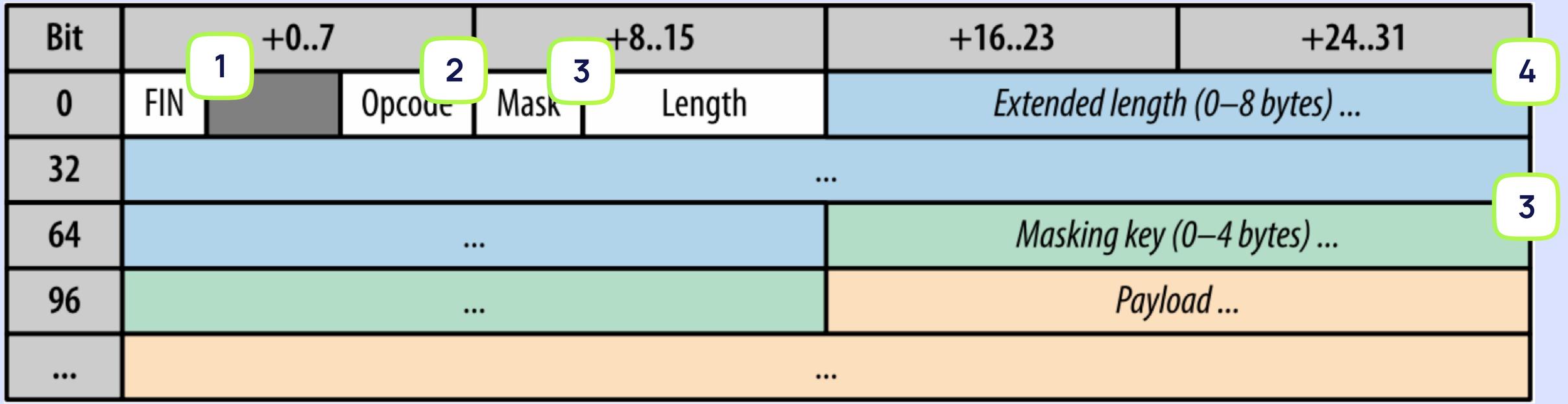
2. Opcode - Тип сообщения

Websocket frame



1. FIN - индикатор последней части сообщения
2. Opcode - Тип сообщения
3. Флаг маскирования и маска

Websocket frame



1. FIN - индикатор последней части сообщения
2. Opcode - Тип сообщения
3. Флаг маскирования и маска

4. Длина сообщения

Websocket frame

2-10 bytes - размер хедера

6-14 bytes - если используем http и маскирование



Типы сообщений

Строковые данные

Бинарные данные

Ping

Pong

Закрытие сокета



Extension

Extension = Преобразования

Например, сжатие данных

Фиксируется на этапе рукопожатия, применяется к каждому сообщению на обеих сторонах

Subprotocols

Дополнительные условия к общению, например:

- Новые типы сообщений
- Дополнительные параметры в сообщениях
- Структура сообщений

Фиксируется на этапе рукопожатия
Клиент может предложить несколько вариантов, сервер выбирает один

Websocket - это боль

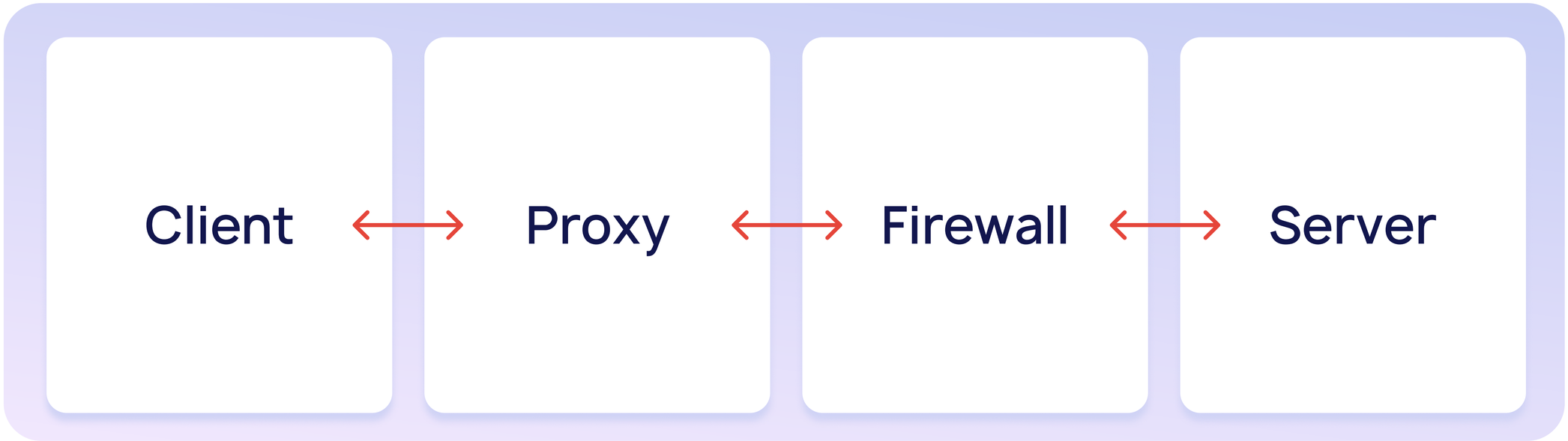
Стабильнось коннекта

Гарантия успешности операции

Батарейка/ресурсы

Переподключение и история

Стабильность коннекта

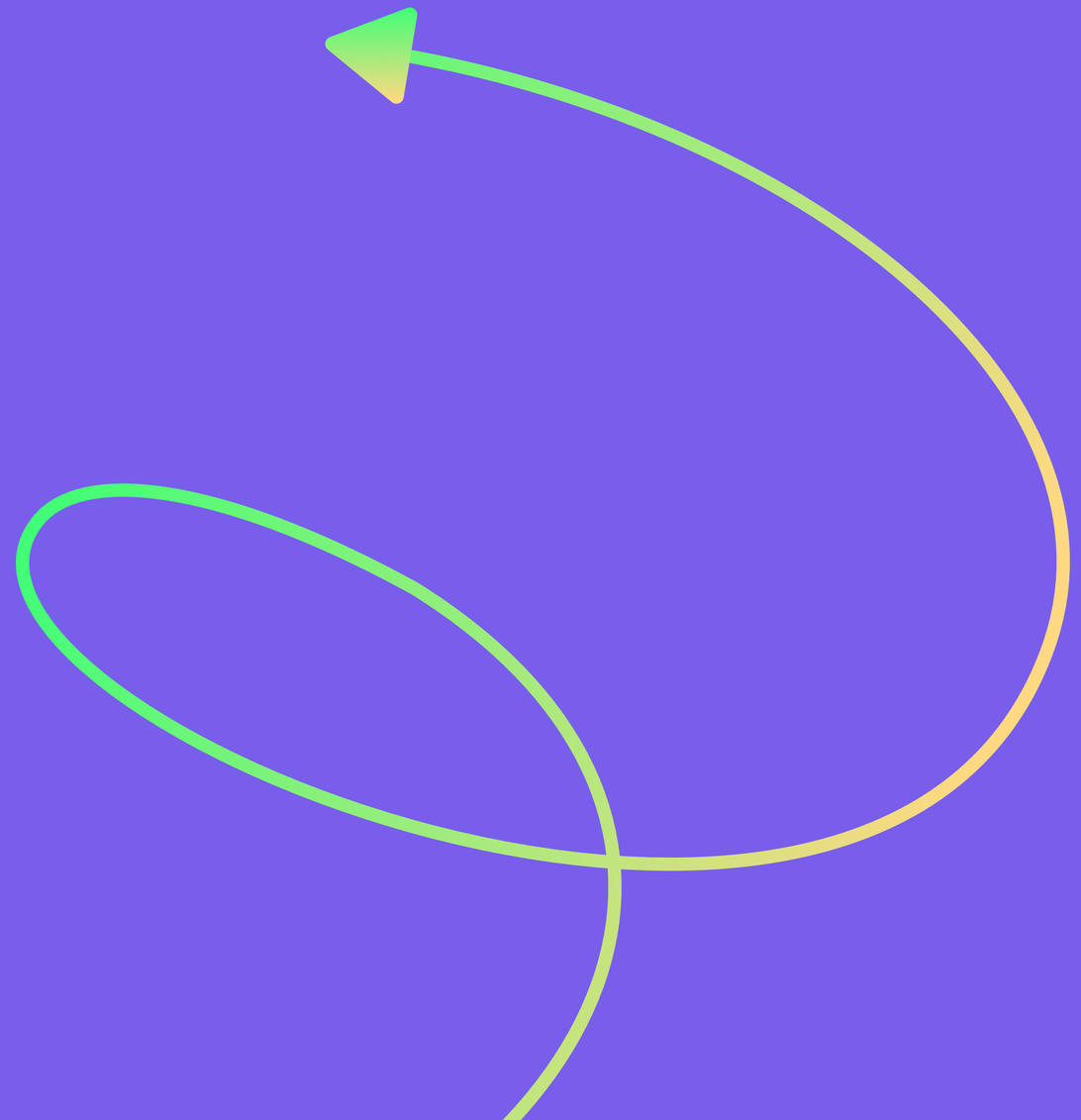


Ping Pong

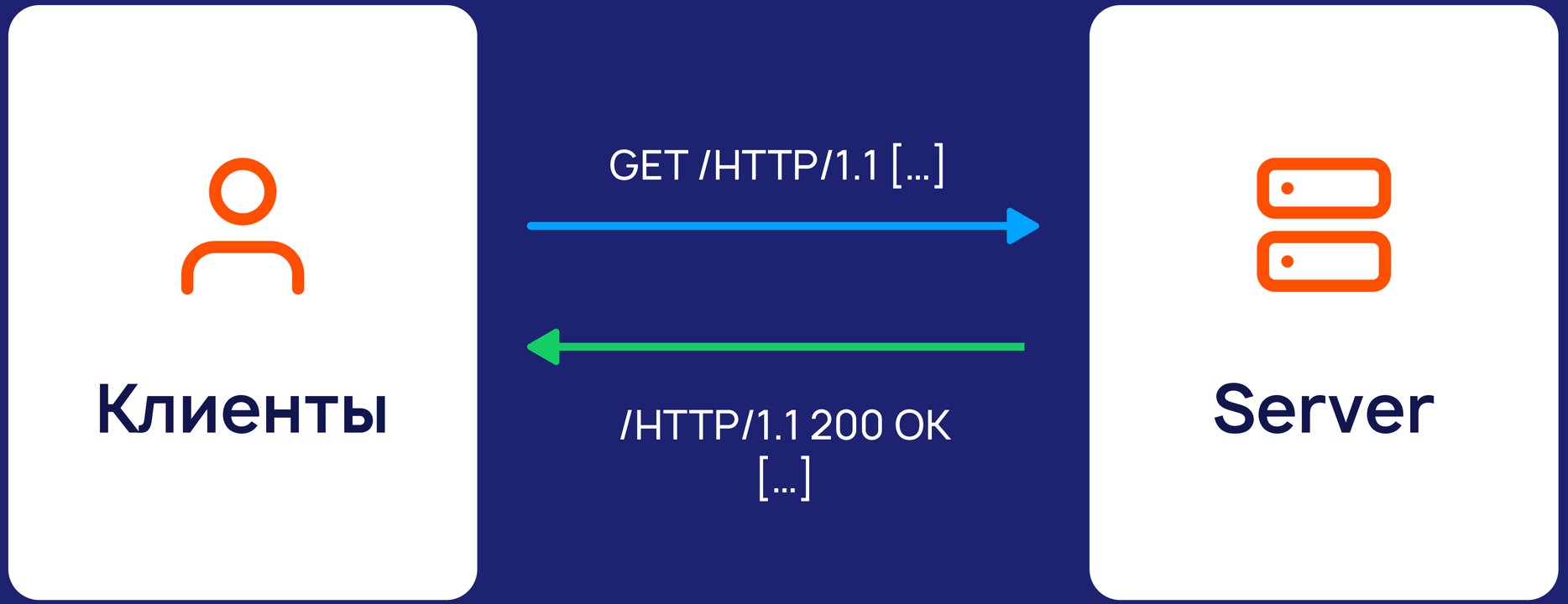
- Периодически шлем сообщения ping в эфир
- Абонент обязан ответить pong
- Нет ответа? Закрываем коннект
  и пытаемся открыть новый

Ping Pong - как часто?

20-30 секунд

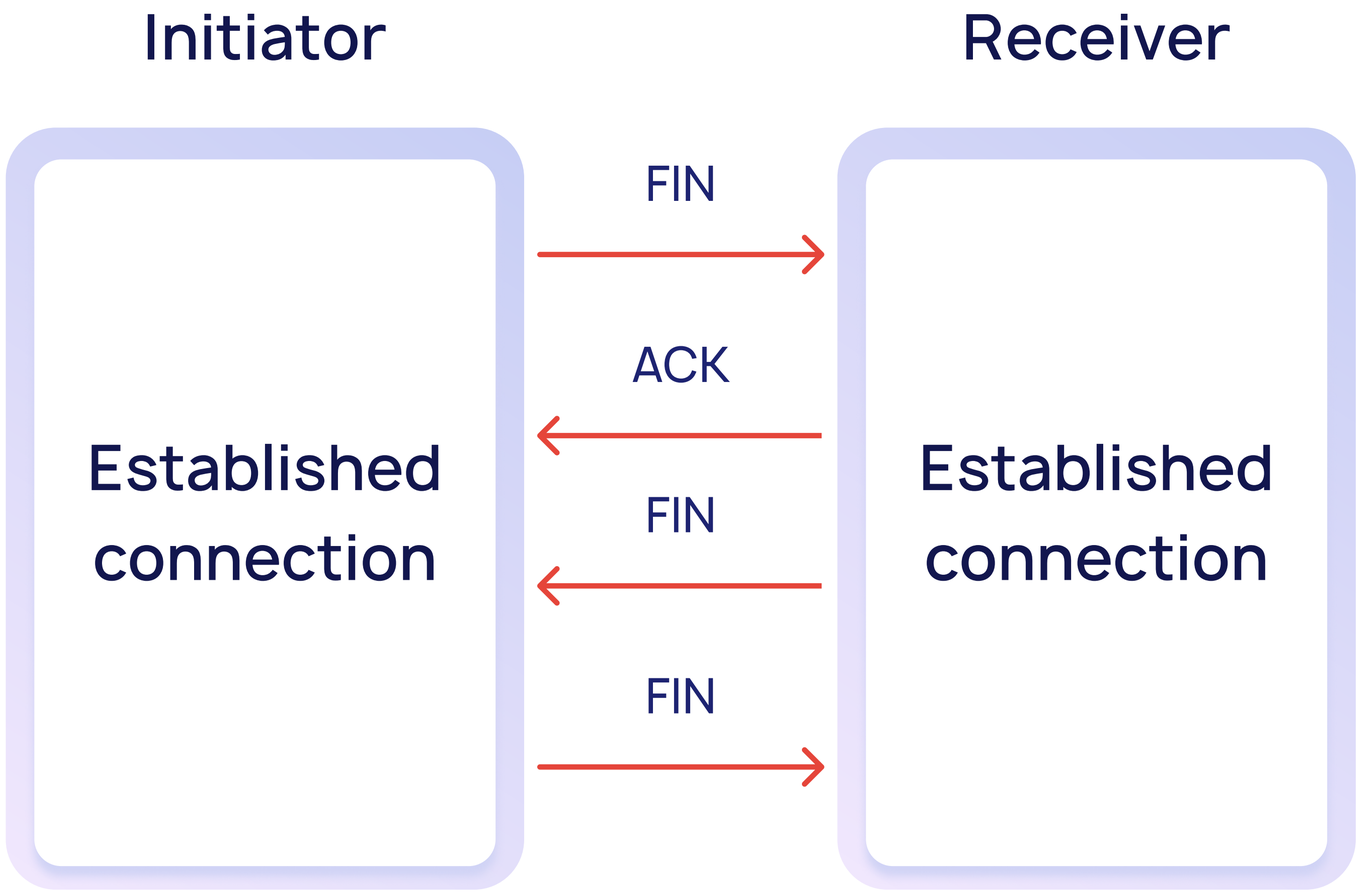


Гарантия успешности



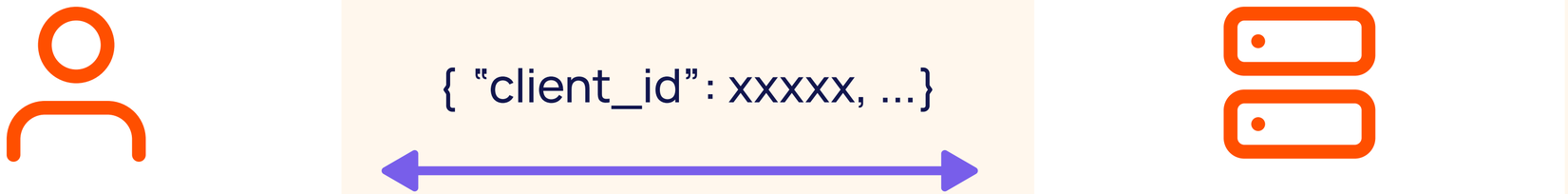
HTTP

ACK


Initiator
Receiver
Established connection
Established connection
FIN
ACK
FIN
FIN

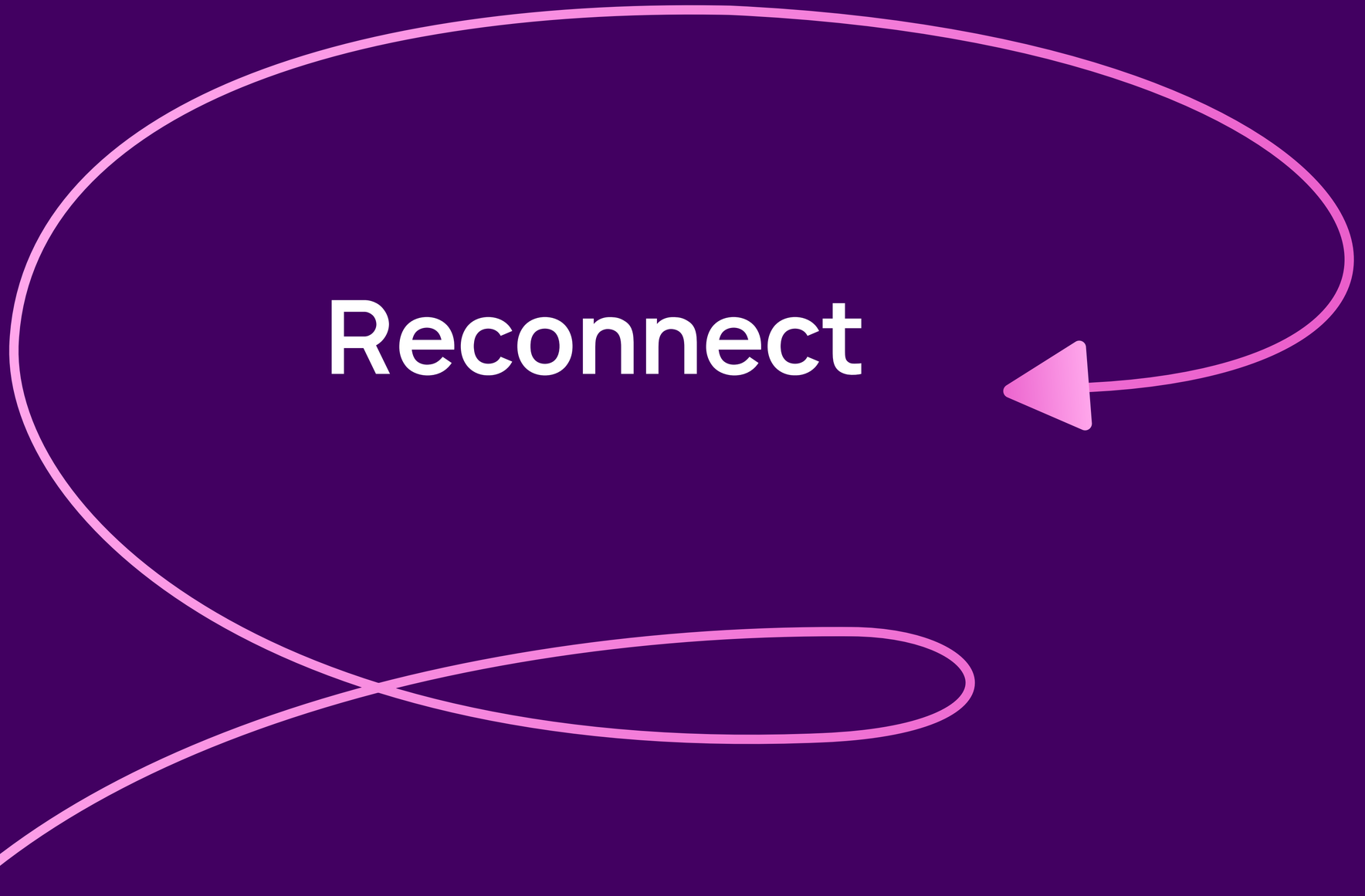
TCP Ack

ACK - пишем сами



Клиенты

Server


Reconnect

Запускаем Watchdog

Переодически пытаемся установить коннект

Лучше использовать рандомный интервал



История

- Нет в протоколе



История - идеи

Отправляем на сервер id/timestamp последнего сообщения

Бек досылает пропущенные сообщения



Ресурсы

По нему постоянно ходят данные

Открыт коннект



Ресурсы - что важно делать ?

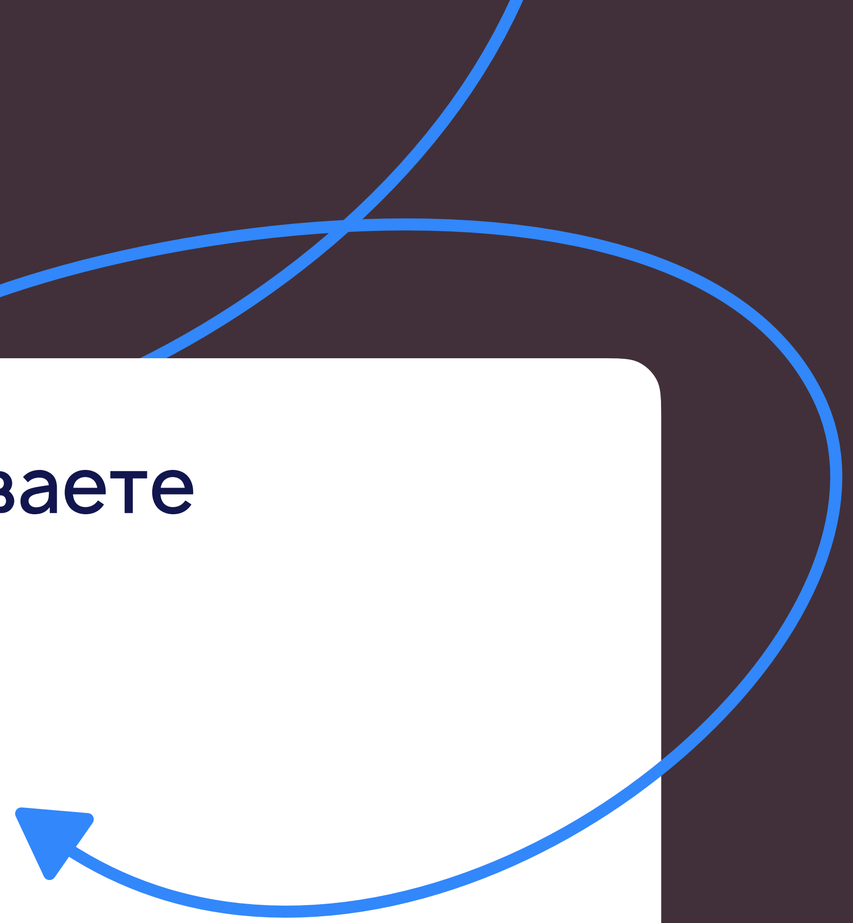
- Важно закрывать сокет при сворачивании приложения
- По нему постоянно ходят данные = постоянно тратим энергию

Сколько тратится энергии на socket?

- Открытие сокета в пустом приложении переводит затраты энергии из Low в High
- Приблизительно 15-20% ресурсов тратится на обеспечение конекта и обработку сообщений

Как же подключить?

- CFNetworking (если вы поддерживаете iOS младше 13 и с бородой)
- Networking (iOS 13+)
- Starscream
- socket.io



Все решения полностью реализуют стандарт Websocket (RFC6455)

Какие проблемы придется решать?

Проблема	Networking	Starscream	Socket.io
Стабильность коннекта	Автоответ, с клиента шлем сами	Автоответ, с клиента шлем сами	Автоматически
Гарантия доставки	Нет	Нет	Есть свой ACK
Reconnect	Пишем сами	Пишем сами, есть подсказки	Пишем сами
История	Пишем сами	Пишем сами	Пишем сами

Какие проблемы придется решать?

Проблема	Networking	Starscream	Socket.io
Стабильность коннекта	Автоответ, с клиента шлем сами	Автоответ, с клиента шлем сами	Автоматически
Гарантия доставки	Нет	Нет	Есть свой ACK
Reconnect	Пишем сами	Пишем сами, есть подсказки	Пишем сами
История	Пишем сами	Пишем сами	Пишем сами

Какие проблемы придется решать?

Проблема	Networking	Starscream	Socket.io
Стабильность коннекта	Автоответ, с клиента шлем сами	Автоответ, с клиента шлем сами	Автоматически
Гарантия доставки	Нет	Нет	Есть свой ACK
Reconnect	Пишем сами	Пишем сами, есть подсказки	Пишем сами
История	Пишем сами	Пишем сами	Пишем сами

Какие проблемы придется решать?

Проблема	Networking	Starscream	Socket.io
Стабильность коннекта	Автоответ, с клиента шлем сами	Автоответ, с клиента шлем сами	Автоматически
Гарантия доставки	Нет	Нет	Есть свой ACK
Reconnect	Пишем сами	Пишем сами, есть подсказки	Пишем сами
История	Пишем сами	Пишем сами	Пишем сами

Особенности Networking

Сделано поверх URLSessionTask → авторизация и redirect

Completion на получения сообщений работает один раз

Особенности Networking

```
func recieveMessage() {
  websocketTask.receive { result in
    switch result {
    case .failure(let error):
      print("Error in receiving message: \(error)")
    case .success(let message):
      switch message {
      case .string(let text):
        print("Received string: \(text)")
      case .data(let data):
        print("Received data: \(data)")
      @unknown default:
        print("Unknown case")
      }
    }
  }
}
```

Особенности Starscream

```
// MARK: - WebSocketDelegate
func didReceive(event: WebSocketEvent, client: WebSocket) {
    switch event {
    case .connected(let headers):
        isConnected = true
        print("websocket is connected: \(headers)")
    case .disconnected(let reason, let code):
        isConnected = false
        print("websocket is disconnected: \(reason) with code: \(code)")
    case .text(let string):
        print("Received text: \(string)")
    case .binary(let data):
        print("Received data: \(data.count)")
    case .ping(_):
        break
    case .pong(_):
        break
    case .viabilityChanged(_):
        break
    case .reconnectSuggested(_):
        break
    case .cancelled:
        isConnected = false
    case .error(let error):
        isConnected = false
        handleError(error)
    }
}
```

1

1. Информирует об изменении статуса подключения сокета

Особенности Starscream

```swift
// MARK: - WebSocketDelegate
func didReceive(event: WebSocketEvent, client: WebSocket) {
    switch event {
    case .connected(let headers):
        isConnected = true
        print("websocket is connected: \(headers)")
    case .disconnected(let reason, let code):
        isConnected = false
        print("websocket is disconnected: \(reason) with code: \(code)")
    case .text(let string):
        print("Received text: \(string)")
    case .binary(let data):
        print("Received data: \(data.count)")
    case .ping(_):
        break
    case .pong(_):
        break
    case .viabilityChanged(_):
        break
    case .reconnectSuggested(_):
        break
    case .cancelled:
        isConnected = false
    case .error(let error):
        isConnected = false
        handleError(error)
    }
}
```

1

2

1. Информирует об изменении статуса подключения сокета

2. Дергается если пользователь из мобильной сети перешел на wifi

Пару советов по проектированию

Документируйте протокол

Завести уникальное поле/ключ type для каждого сообщения

На старте заложить ACK и получение истории



Особенности socket.io

Легкий старт

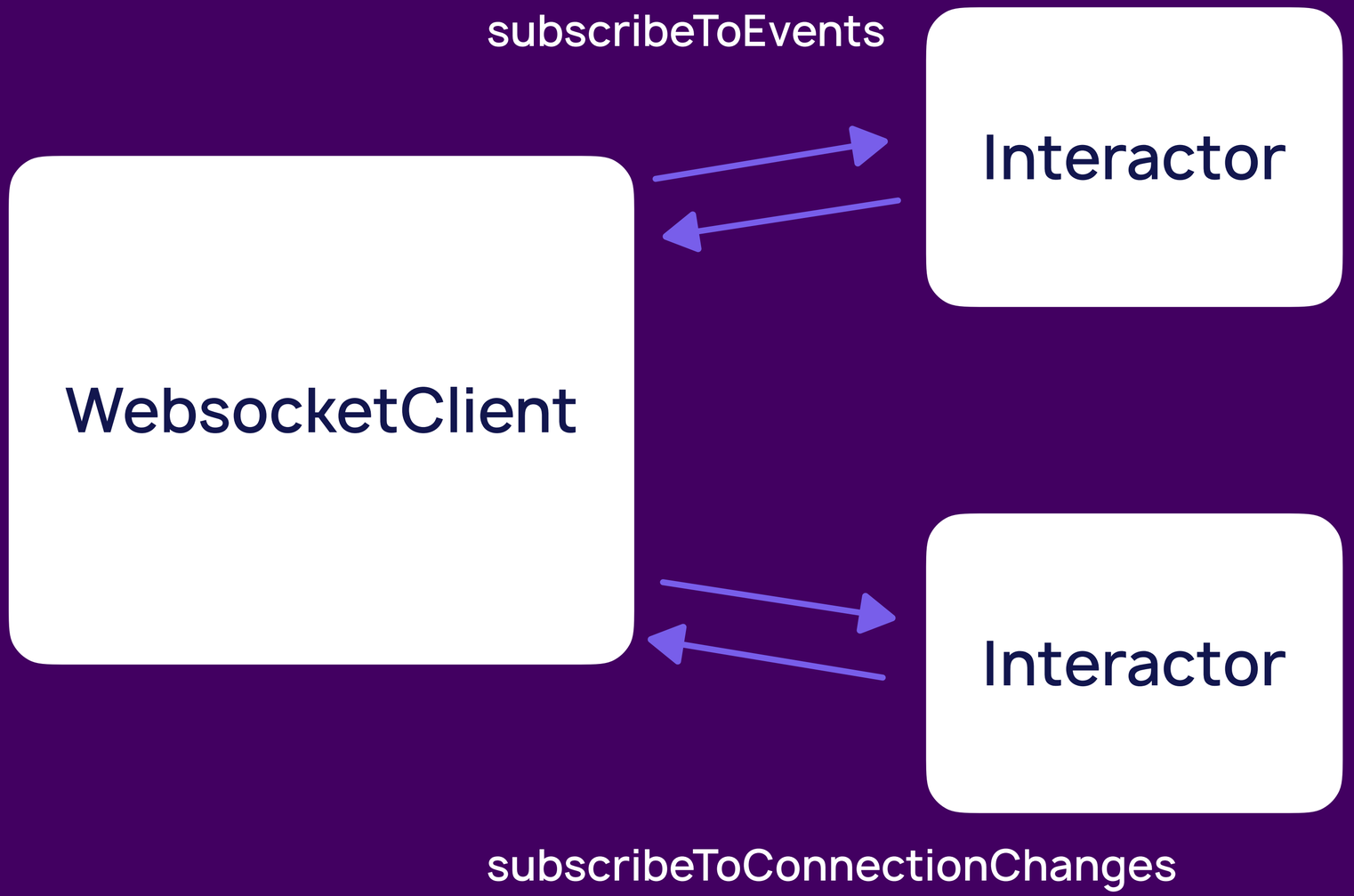
Polling из коробки

Polling повышает нагрузку на бек

Масштабирование бекенда



Архитектура ManyChat



Websocket Client

Connect/disconnect

Watchdog для reconnect’a

Преобразовывает полученные json’ы в модели

Поддерживает подписку на получение новых сообщений, изменения статуса Websocket

Парсинг сообщений

```
init(from decoder: Decoder) throws {
    let container = try decoder.container(keyedBy: Keys.self)

    let event = try container.decode(String.self, forKey: .event)

    switch event {
    case EventKey.thread.rawValue:
      let dialog = try container.decode(Dialog.self, forKey: .model)
      self = .thread(model: dialog)

      …
    default:
      self = .unknown(event: event)
    }
```

Обработка сообщений в бизнес-логике

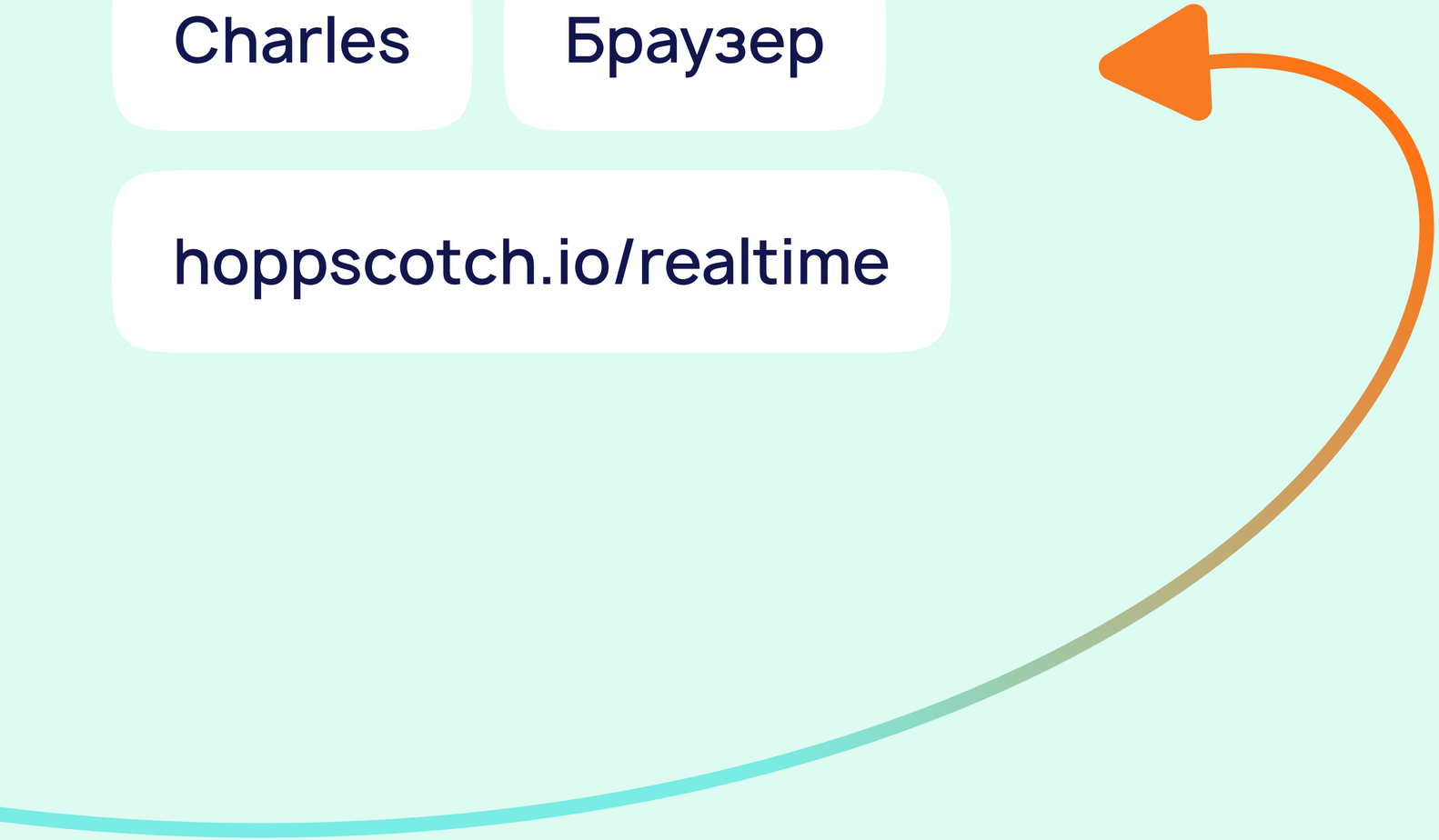
```
func subscribeToWebsocket() {
      websocket.addMessageObserver(self, for: channel) { (interactor, _, event) in
        switch event {
        case .threadUpdated(let dialog):
          interactor. updateDialog(dialog)
          …
        default:
          break
        }
    }
```

Как тестировать?

Charles

Браузер

hoppscotch.io/realtime





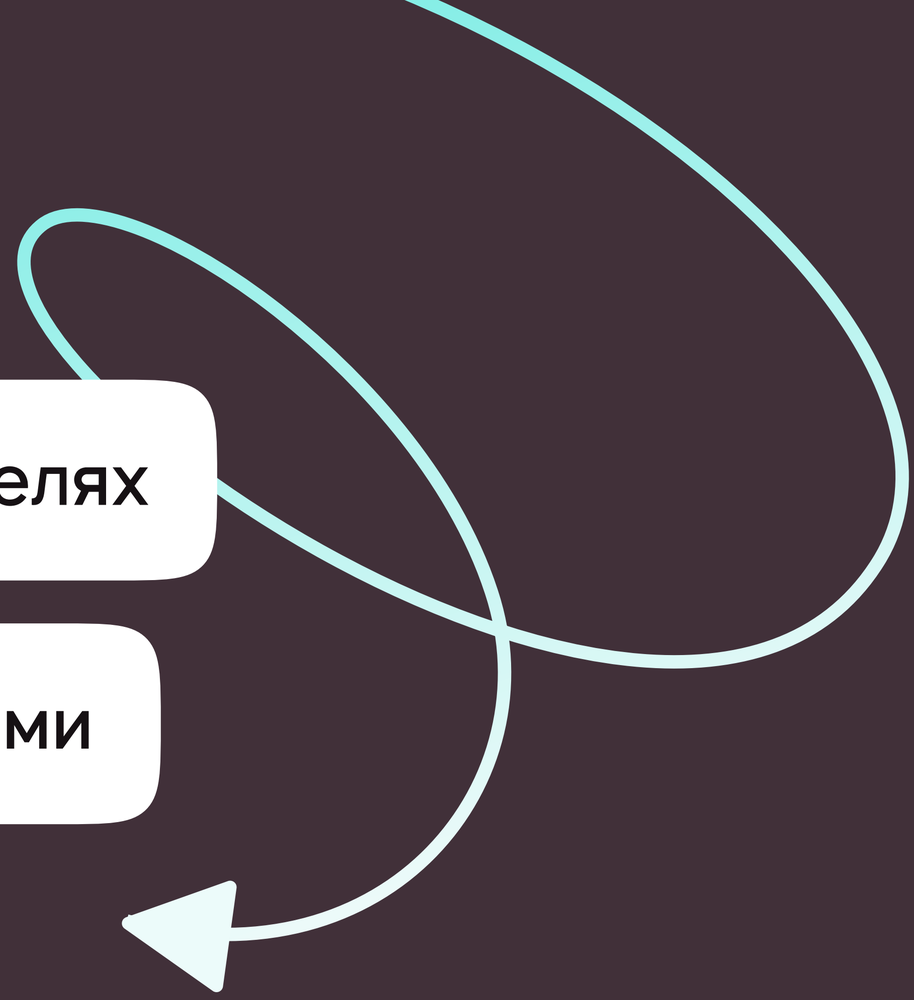
One more thing?

Pub Sub

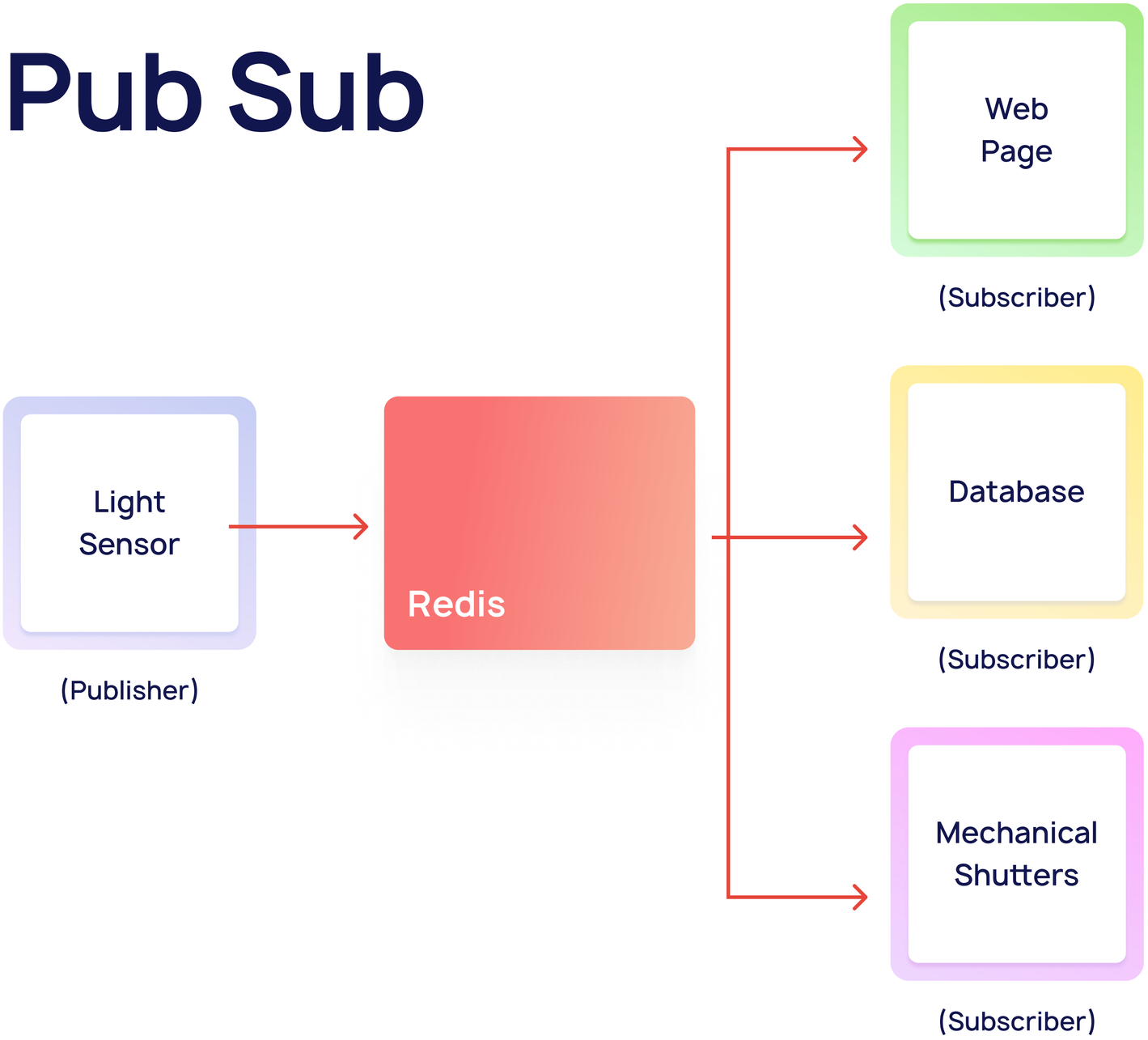
Не нужно знать о конкретных пользователях

Работка с каналами, а не с пользователями

Удобно масштабируется на бекенде



Pub Sub


Light
Sensor
(Publisher)
Redis
Web
Page
(Subscriber)
Database
(Subscriber)
Mechanical
Shutters
(Subscriber)

Как завести?

- Реализовать логику самим поверх вашего решения
- Докрутить socket.io
- Есть решение на google app engine
- Centrifugo

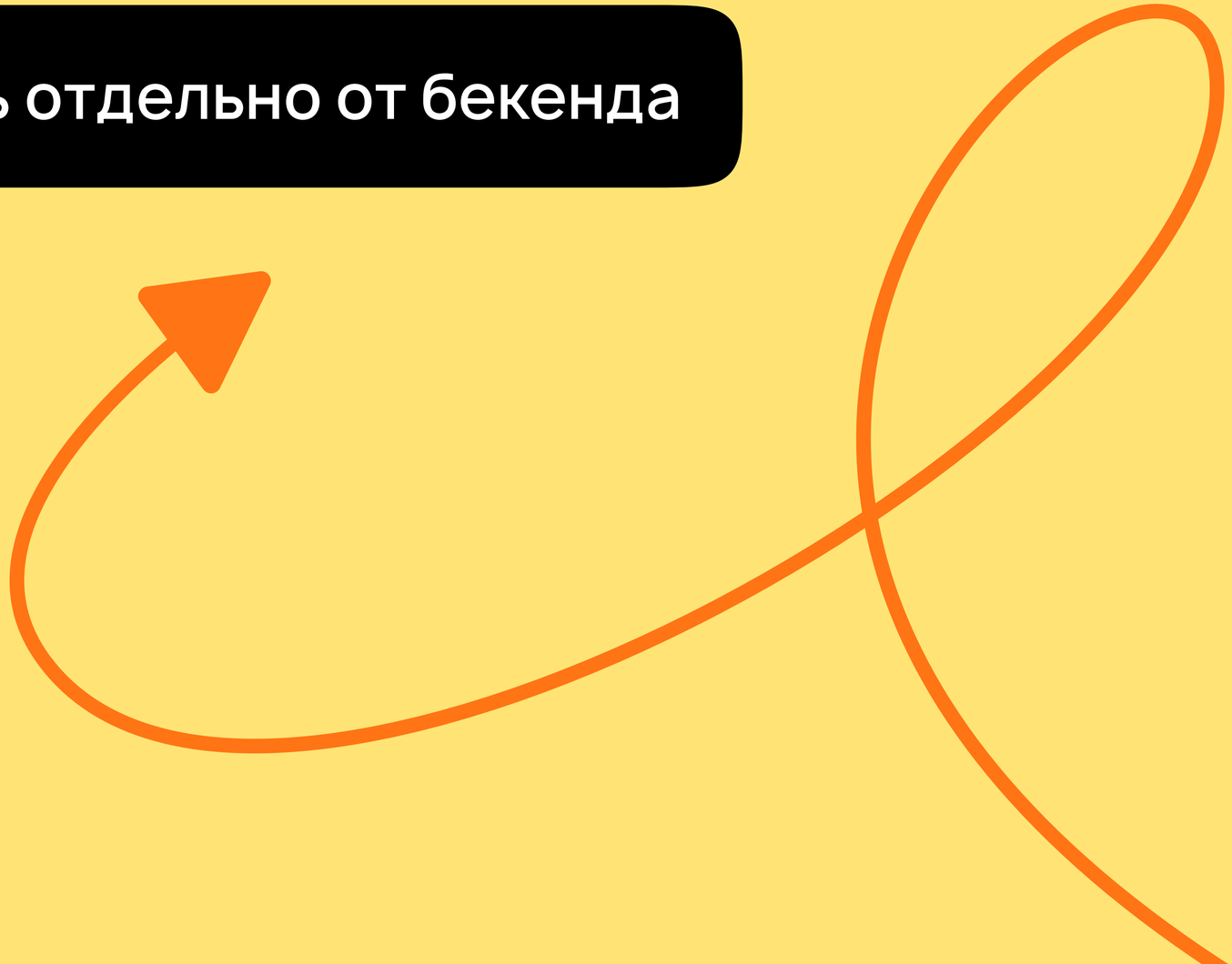
Centrifugo

- Каналы, события с ними, настройки видимости и ограничения по размеру
- Получение истории канала, восстановление сообщений между reconnect’ам
- Отправку с клиентов лучше делать через HTTP
- Отдельный сервис
- Хорошо масштабируется горизонтально (держит 1,000,000 + connections)
- Json/Protobuf

Centrifugo - отдельный сервис

Ее можно масштабировать отдельно от бекенда

Никакой бизнес-логики



Как отправлять данные с клиента при Centrifugo

Шлем HTTP на сервер

Сервер валидирует, выполняет бизнес логику

Сервер шлет на Centrifugo

Centrifugo рассылает клиентам



Если отправлять по HTTP

Уйдет проблема с гарантией доставки

Нужно следить за дублями (ACK)

Можно переиспользовать текущий Rest API



Как отправлять данные с клиента при Centrifugo

Шлем специальное сообщение по socket’у

Centrifugo проксирует запрос на бекенд

Сервер выполняет проверки, логику и отправляет ответ Centrifugo

Centrifugo рассылает клиентам сообщения

Как отправлять данные с клиента при Centrifugo

```
{
  "client":"9336a229-2400-4ebc-8c50-0a643d22e8a0",
  "transport":"websocket",
  "protocol": "json",
  "encoding":"json",
  "user":"56",
  "method": "getCurrentPrice",
  "data":{"params": {"object_id": 12}}
}
```

Для клиентов есть библиотека Centrifigo

- Автоматический reconnect, отправка ping/pong
- Поддержка всех основных методов Centrifugo
- Развивается силами сообщества
- Поддерживает только protobuff

Выводы

Когда

Вам нужно регулярное общение между бекендом и клиентом

Для нерегулярных обновлений на определённых экранах - polling/long-polling (лучше с HTTP 2.0)

Как

Есть нативный способ запустить и несколько библиотек

Для реализации PubSub существует хорошо масштабируемое решения Centrifugo

Сколько стоит

Для Вас:

- На backend’е придется разворачивать дополнительную инфраструктуру и поддерживать ее. Может быть дорого
- Внедрение ACK, алгоритма Reconnect’а, история - потребует усилий
- Подключение библиотеки и интеграция - легко

Для пользователей:

- Возрастет энергопотребление. Желательно помнить об этом и не забывать закрывать, когда websocket не нужен:)

Материалы

Frame websocket’a на Swift

Советы по безопасности websocket’a

Стандарт Websocket

Centrifugo

Подробный разбор работы Websocket

Extensions и Subprotocols

Вопросы

